Health Link: A Wide Area Telecommunication Network for Health Care Providers

by

James Grennell McDaniel
B.S., Case Western Reserve University, 1968
B.Sc., University of Victoria, 1990
M.S., Cornell University, 1970

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the School of Health Information Science

We accept this thesis as conforming
to the required standard

Dr. J.R. Moehr, Supervisor (School of Health Information Science)

Dr. H.A. Müller, Co-supervisor (Department of Computer Science)

Professor D.J. Protti, Departmental Member (School of Health Information Science)

Dr. D.M. Hoffman, Outside Member (Department of Computer Science)

Dr. K.F. Li, Outside Member (Department of Electrical and Computer Engineering)

Dr. C.A. Laszlo, External Examiner (Department of Electrical Engineering)

Name JAMES G. McDANIEL

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

HEALTH SCIENCES - GENERAL
SUBJECT TERM

0566
SUBJECT CODE

UMI

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Supervisor: Dr. J.R. Moehr
Co-supervisor: Dr. H.A. Müller

Abstract

Early computerized health information systems supported applications in hospital records and laboratory data collection. Since that time, software has been developed for a number of health care providers such as doctors and pharmacists. Although local area networks are installed at larger institutions, only a few small-scale, special-purpose, wide-area networks are installed for external providers. To be adopted, wide-area networks should provide greater functionality than, and be cost-competitive with, conventional communication methods. Several projects are underway in Health Information Science to develop and evaluate generic, wide-area networks.

This dissertation describes the design, analysis, development, implementation and evaluation of a prototype health care network which would be accessible to providers using existing computer equipment and the public switched telephone system. The network software, ***Health Link***, supports reliable, automatic, store-and-forward messaging of medically-sensitive information. Encrypted messages can be authenticated and the software features registered delivery. An application programming interface formats messages in accordance with the HL7 data interchange standard.

Simulation studies have been conducted which demonstrate the steady state characteristic behaviour of a node in a uniform cluster. Further studies have investigated a realistic, dynamic, large scale network. A peer-to-peer model and client-server model were analyzed and both were found to be feasible with respect to certain performance and cost criteria. The client-server model was found to be less costly to operate than the peer-to-peer model. The peer-to-peer model can transfer messages in a shorter time than the client-server model.

The network software was verified in a field test involving four clinics, one medical laboratory, and one hospital. Data collected in the test provide performance benchmarks, an estimate of message sizes and frequencies, network reliability statistics,

and a wealth of observations. Performance benchmarks and message traffic measurements were used to calibrate the simulation models.

Results from this and other research indicate that, although most of the technical networking problems can be readily overcome, consensus on standards, health care applications, and initiatives should be promoted before a wide-spread, production network is implemented.

Examiners:

Dr. J.R. Moehr, Supervisor (School of Health Information Science)

Dr. H.A. Müller, Co-supervisor (Department of Computer Science)

Professor D.J. Protti, Departmental Member (School of Health Information Science)

Dr. D.M. Hoffman, Outside Member (Department of Computer Science)

Dr. K.F. Li, Outside Member (Department of Electrical and Computer Engineering)

Dr. C.A. Laszlo, External Examiner (Department of Electrical Engineering)

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to recognize and thank the members of my committee, Jochen Moehr, Hausi Müller, Dan Hoffman, Kin Li, and Denis Protti, whose support and advice has made this dissertation possible. The guidance and encouragement provided by Jochen Moehr and Hausi Müller have been invaluable in this research.

Many people and organizations have been involved in developing and promoting *Health Link*. Particular recognition is given to the participants of the field test: Clinicare Corporation, Cube Computer Systems, Cumberland Medical Clinic, Doctors Medical Clinic, Island Medical Laboratories, Medical Associates Clinic, Saint Joseph's Hospital and University Health Services. Special thanks are given to Dr. Richard Backus, John Beintema, Dr. Ronald Brown, Bev Gemmell, Stewart Jack, Donna Kingston, Eric Macdonald, Dr. Michael McNeely, Dr. David Musgrave, Dennis Niebergal, Dr. Jack Petersen, Jill Parker, Jean Preece, Elaine Sawatsky, Dr. Diana White and Yokee Wong.

As required by the University of Victoria, a description of my contribution to this research follows. I designed the *Health Link* system and programmed the production version of the adapter software with the exception of certain variants introduced by the gateway. I designed and implemented all of the simulation programs and conducted the simulation analyses. I participated in the installation of the field test sites, conducted the stress testing of the prototype network and monitored the field test.

I was assisted by a project team whose members I wish to thank. Shelly Anderson conducted the site surveys, monitored the progress of the field test and maintained contact with the field test partners. David Bakke assisted in the field test installation, monitored the field test, adapted the software to the requirements of the gateway, programmed the additional modules required for the gateway program and adapted the HL7 Application Programming Interface. Sarma Vempati assisted in the field test installation, monitored the field test, and converted the terminal emulation software to run on the DEC VAX and RISC SYSTEM/6000.

I would like to give thanks to Sue Dier, my wife, who has provided invaluable advice and moral support. I thank my parents who have given me encouragement and the benefit of their experiences.

Support for this research was given in part by the Science Council of British Columbia under the HDF #43-89, MART #90/91-11, HDF #7-91, and Technology B.C. #179 (T-3) Grants. Much of the financial support for my studies has been given by the University of Victoria and the Medical Research Council of Canada.

Trademarks

1 Introduction

1.1 Electronic Communication in Health Care

Information is essential for effective health care delivery. Health care providers, both individual and institutional, rely heavily on medical, environmental, occupational, psychological and social research; observations of symptoms exhibited by one or more clients; observations of program or treatment outcomes; evaluation of program or treatment effectiveness, and health care accounting, finance and policy administration. Health care services are provided through specialization. Different types of organizations supply different needs which range from scientific research performed by government agencies, educational institutions and research departments of major suppliers to delivery of patient care by individual practitioners. The distribution of health care services is non-uniform. Economic, political, geographic, demographic and epidemiological factors influence the placement of specific health care providers and resources. A client of health care services usually requires services from more than one provider. For example, even a simple visit to the dentist demands co-ordination among the dentist, pharmaceutical suppliers and dental equipment suppliers, not to mention other non-medical agencies such as dental insurers and government social services.

The health care system is extensive and highly integrated. It relies on many types of communication to effect co-ordination. Traditional communication is supported by newspapers, professional and organizational journals, drop boxes, mail, courier, facsimile, television, telephone, meetings and personal discussions. Lately, digital electronic communication has emerged as a communication modality. Although electronic data communication has been present for over twenty years within large health care organizations such as major hospitals, it is only now that an effort is being made to integrate it into a wide area context among diverse health care providers.

Just as telephone communication has intrinsic characteristics which differ with those of other communication modes, so does digital electronic communication. Electronic data communication has several attractive properties.

- Data can be easily stored, indexed, retrieved, copied and shared.
- Data which can be digitized, including image and voice, can be communicated and processed using digital computers.
- Data can be transmitted without significant delay.

- The current communication technology places few meaningful restrictions on the volume of health data which can be transmitted.
- Data communication can be less expensive than other traditional communication modalities.

However, electronic data communication has some unattractive properties as well.

- Data can be easily altered, lost or destroyed by accident or with intent.
- Data can be falsely attributed to a sender.
- Data can be disclaimed by a recipient.
- Data can be intentionally or unintentionally disclosed to unauthorized people and organizations.
- Data can be misinterpreted by incompatible end-user applications.
- Analog to digital data conversion for digital communication is not completely reversible and can require sophisticated equipment.

Like the more traditional methodologies, techniques can be applied to electronic data communication to compensate for its unattractive properties with varying degrees of success. For example, authenticity codes can be added to messages and files to ensure that they have not been altered. However, digital communication can never completely overcome the problem of detail lost due to digital encoding of analog data as this is an intrinsic characteristic of digitization.

Digital computers have been acquired by most hospitals in the developed world. Many physicians and dentists now use computers to manage their practices. Electronic patient record software is available for both hospitals and private medical practices. It is in this computerized environment that electronic data communication has its greatest benefits as data can be exchanged directly among providers with a minimum of manual intervention. One issue, then, is for electronic data communication technology to be adopted for those applications for which it is better suited than competing communication modalities.

Widespread use of digital communication networks has given rise to new applications. Electronic bulletin boards and electronic bibliographic reference services are examples of applications which, although they have manual analogs, have received new definition and usefulness with electronic communication. Intrinsic properties of

traditional communication modalities have suppressed the growth of some applications which would flourish with electronic data communication. A second issue is to explore new applications which might increase the effectiveness and efficiency of health care delivery.

It is perhaps unexpected that no general purpose wide area network for health care providers is yet in operation. A number of experimental pilot projects have been conducted but they fall short of full-scale production networks. An even greater number of commercial networks have been developed to support single health care applications but they show few signs of evolving into networks which support a range of applications for many classes of providers. The benefits of electronic data communication cannot be fully realized without promoting the development of general purpose health care networks.

1.2 Context of Research

In order to address the issues above, wide area health care networks require further attention. Past research and development has been sporadic, ephemeral and disorganized. Very few of the pilot projects have received formal analysis and reporting. It is impossible to describe all of the research and commercial attempts that have been made and why they have failed. The outcomes of these experiments are now lost.

There is abundant evidence in other fields that large wide area networks are practical. For example, banking and finance have employed wide area networks for over a decade. Like these fields, health care relies heavily on communication and it is probable that an appropriately implemented network would be viable given the proper nurturing.

The development of a health care network demands more than simply the installation of networking software and hardware. The nature of the information exchanged over a health care network places stringent requirements on data confidentiality, authentication, authorization, non-repudiation and interpretation. Health care data is multi-media: it can be text, video, image, biometrics, audio or simply application-specific analog or binary encoded. Data access time requirements

may vary by application which may depend on real-time communication, on-line connections or store-and-forward messaging.

Health care providers are geographically distributed. Some are situated in areas which have few communication media alternatives. Many of the providers act independently as private agents whose participation in a network must be individually solicited. A number of providers already have selected their preferred end-user application software and computer hardware with little attention to complementary functionality or homogeneity. Indeed, it is these characteristics of the data and of the health care providers which distinguish health care networks from many other networks.

Many of these issues can be resolved technically by developing the appropriate networking software and tools. In the past, some developers have resorted to partial solutions simply because of their ready availability. Lack of information makes evaluation of these efforts difficult, but they may have been unsuccessful partly because of their insular approach.

Some of these issues pose a much greater difficulty because they require compatibility among end-user systems and application software. Differences in data interpretation can be the result of *syntax*, *semantics* and *structure*. Syntax is the format and grammar of data representation. Semantics relates to the meaning of the data, such as how a test value relates to the normal range for that test or how the value of one field relates to another. Structure is taken here to mean the relationships among data induced by temporal and procedural properties of the application system structure and operation. For example, there is a structural difference between a preliminary and a final test result. Adoption of open standards for data interchange would alleviate problems with incompatible data syntax. However, semantic and structural differences among different systems would still exist unless there were strict adherence to guidelines for software development. This approach precludes the application software already installed that has been written without regard to these as-yet, non-existent guidelines.

If we assume that the technical issues can be satisfactorily resolved, a viable network is contingent on its acceptance by the health care providers who must address legal, financial, social and cultural concerns within their organizations and the health

care community. There are outstanding legal issues regarding the release of electronic data to other providers and institutions; data security; data non-repudiation procedures; and data authentication procedures.

A network has a financial impact on all health care providers, even though they may not participate in the network. The benefits of a network may accrue to certain classes of providers whereas the costs may be assumed by others. If not everyone joins a network, those providers with manual procedures may be obligated to assume the consequential manual processing costs incurred by automated providers. New applications made available by a network may carry increased costs although with correspondingly increased benefits that are cost-unrelated. There may be significant capital costs for those providers who join the network either with no computer system or with an inappropriate system.

A network has a social impact on health care providers and their workers. Responsibilities and procedures change as a result of automation. New skills must be developed. In some cases, jobs may be lost or created but more likely, jobs will be reclassified. Second-order social-psychological effects related to fear of change, self-worth, self-image and working relationships may impede or accelerate acceptance of network-related automation.

Culture within an organization and in larger contexts, such as professional organizations, the client base and the general public, influence the success of new technologies. A network is more likely to succeed if there is already a secure base of computerized applications in health care and if, historically, there has been a positive attitude toward computerization [1, 2].

The research described here focuses on the development of a wide area health care network for the North American context. The emphasis of this research is placed on the exchange of generic health care data, which includes clinical data as a major component, rather than simply administrative or financial data. The work produces a solution to many of the technical problems; examines network advantages and network costs; and identifies several critical factors affecting network installation.

1.3 Problem Statement

This research includes the implementation of a prototype wide area network providing secure and confidential exchange of health-related data among health care providers using store-and-forwarding of character-based messages. The specific objectives of the research are to:

- develop prototype messaging software to support wide area telecommunication for health care providers which relies minimally on the public switched telephone system and on existing heterogeneous health care systems,
- implement techniques to promote data security and confidentiality within the network,
- implement techniques to provide message tracking, authentication and non-repudiation,
- examine the impact of network scaling on feasibility and performance,
- resolve technical problems related to installing and managing the network,
- investigate the applications appropriate to a messaging network,
- analyze the behaviour and costs of a messaging network with respect to topology, and
- test and evaluate a pilot network.

The research investigates the viability of a network which:

- can be operated automatically and non-invasively from those computer platforms currently used by providers,
- can reliably use the public switched telephone system or a more sophisticated underlying network as a communications medium,
- costs approximately $50/month per physician to operate, and
- is easily scaled up to a large user base like that found in a province or state.

The success of the prototype network and its evaluation form a base for future research relating to provider acceptance of telemedicine and its impact on resource utilization and delivery of health care services.

1.4 Research Methodology

Prototype communications software is developed which captures appropriately addressed files stored on an end-user's application computer. A file is transferred to a message base where it 1) receives a Message Authenticity Code (MAC), 2) is compressed and 3) is encrypted using the MAC. The message receives an electronic signature which is built from the MAC and other message-specific data using the sender's private encryption key. Using the Rivest Shamir Adleman (RSA) public key crypto-system [3], the signature is encrypted with the receiver's public encryption key to ensure that only the destination node can decrypt the signature. The messages are held at the node until a connection is made. Connections can be initiated at either the destination or the source. Several connection scheduling and routing algorithms are employed to minimize the overhead incurred by the connect/disconnect procedures. A message is broken into frames and the frames are encrypted once again and then translated into printable characters before being transmitted. Once the message is transferred to its destination, its signature is validated and it is decrypted, decompressed and transferred as a file to the destination end-user system. A tool box of routines is constructed which permits messages to be formatted according to the Health Level 7 (HL7) data interchange standard. The entire procedure is executed automatically. The prototype software is designed to run either on the application computer or from a network front-end processor. In the latter case, an unobtrusive background process on the end-user computer performs the local file transfer to and from the front-end processor.

User authorization, route resolution and public encryption key tables are maintained at a subnet server or gateway which also stores and forwards those messages which are not immediately deliverable. The gateway monitors the state of a subnet of nodes through the use of control messages.

The software is written in the C language and is easily transported to heterogeneous platforms. The software is implemented for the IBM PC, the IBM RS6000 and the DEC VAX.

Performance benchmarks are taken of the prototype software to be used in network simulation studies. Discrete event and analytic simulation programs are written which predict the steady state behaviour of mesh and star clusters ranging from 3 to 1,000 nodes.

Surveys are taken at two medical clinics to determine clinic procedures and the type and volume of communication traffic. The surveys supply data for further discrete event simulations of a Saskatchewan model network having physicians, medical laboratories, hospitals and the Medical Care Insurance Commission. Simulation data is collected from the model for peer-to-peer and client-server topologies over a daily cycle. The data is analyzed and used to extrapolate network performance and communication costs for the two topologies.

A field test of the prototype is conducted among six nodes consisting of four clinics, one medical laboratory and one hospital. Data is collected from the field test to determine the reliability of the network. Observations are made regarding network installation and operation.

1.5 Dissertation Outline

This dissertation describes the methods used to develop the prototype network software. The behaviour of the software is measured and used to predict the behaviour and costs of large scale health care networks employing a similar design and operating rubric to that of the prototype. The software is demonstrated, tested and observed in a live field test.

Chapter 2 presents an argument for establishing health care networks. It outlines the context of such a network by giving an overview of appropriate applications, benefits and infrastructural concerns. It describes the areas which require further research and development and gives the rationale for this research.

Chapter 3 describes three other pilot projects which have a similar research focus.

Chapter 4 provides conceptual and design information about the network software. The network topology is described and terms are defined. The design criteria imposed by the operating context is given. A functional description of the software and a description of the network architecture are provided with benchmark data for those processes consuming significant processor resources.

Chapter 5 reports results of steady state simulation studies made for two network topologies: star and mesh. Discrete-event and analytic simulation models are described and compared. An analysis of model initialization effects and statistical standard error is presented.

Chapter 6 extends the simulation research begun in Chapter 5 to two large-scale, cyclic models of the Saskatchewan health care system. The performance characteristics of peer-to-peer and client-server topologies are compared. The cost of communications for both topologies is estimated based on prevailing tariffs and estimated loads. A business model is constructed and an operating cost estimate is presented.

Chapter 7 describes a field test. The composition of the pilot network and the schedule of activities is given. Observations are made with respect to network reliability, communication requirements, and network installation and operation. Several recommendations are made for future network installations and prototype improvements.

Chapter 8 reviews the significant findings of the research and outlines future work which can use the prototype network as a platform. Future improvements and extensions to the software are proposed.

2 Health Care Network Implementation Objectives

2.1 Introduction

Health care providers constitute a large segment of the service providers in developed countries. In 1987, Canada had more than 241,000 registered nurses [4:3-34], 55,000 physicians [4:3-32], 16,000 pharmacists [4:3-35], 13,000 dentists [4:3-33], and 1,000 hospitals [4:3-27] with thousands of ancillary staff. In addition to these providers, there are a number of other orthodox, alternative and ancillary service providers which include physiotherapists, optometrists, chiropractors, naturopaths, community home care workers, health insurance agencies, social service organizations, medical suppliers, pharmaceutical companies and government departments. Health Canada estimates the 1991 overall cost of health care in Canada to be approximately $67.1 billion dollars [5:16] which constituted about 10 percent of the Gross National Product (GNP) [5:11]. If it were assumed that expenditures in health care are entirely in labour, this would represent the income of over one million workers.

Individual clients of the health care system receive a spectrum of specialized services from health care providers. For example, the treatment plan for a patient in an intermediate care hospital might require services from a General Practitioner (GP), one or more medical specialists, a physiotherapist, nurses, dietitians and a pharmacist. Co-ordination among these providers is achieved through paper, voice, telephone and facsimile communication. Little electronic data transfer is used for several reasons:

1) A computer communications network relies on a widely installed base of complementary computerized information systems.

2) Electronic Data Interchange (EDI) standards are not sufficiently comprehensive to support the wide range of transactions used by providers.

3) A network which supports text, image, audio, video and biometric signaling and which is able to accommodate both real-time communication and message store-and-forwarding is expensive to install.

4) Security mechanisms ensuring information confidentiality for end-user sites using a network are difficult to install and maintain for a large, heterogeneous user base.

5) A network developed for densely populated areas is likely to be less cost-effective for sparsely populated areas than other communication technologies.

6) A network product is difficult to market without featuring highly visible value-added features not available to competing communication technologies.

7) A network requires rapid and near-total adoption by groups of interacting health care providers to make subscription by any individual provider worthwhile.

8) The inertia of procedures based on paper systems is difficult to overcome for a variety of reasons which include fear of change, additional training requirements and transitional expenses.

9) Electronic communication is often thought to be less reliable than paper because paper is tangible and because people have well-established procedures to manage and audit paper communication.

The following section lists the applications which can be supported by, and the benefits which might accrue from, a health care network. Section 2.3 presents and attempts to answer two fundamental questions relating to the development of health care networks. Section 2.4 summarizes the areas which would benefit from further research and Section 2.5 describes a rationale for conducting this particular research project.

2.2 Benefits

A network is an enabling technology for access to:

1) automated direct purchasing and invoicing of supplies,

2) co-ordinated (group) purchasing and centralized distribution of supplies,

3) such registries as cancer and organ-donor,

4) distance education and programmed learning facilities,

5) bibliographic searches and references,

6) distributed electronic patient records,

7) automated submission and reconciliation of service billings,

8) collection of statistics for research, epidemiological analyses, resource management and environmental monitoring,

9) ad hoc professional and personal communication which employ electronic mail and electronic bulletin boards,

10) research facilities supporting advanced or experimental computerized diagnostic and treatment systems which might rely on Artificial Intelligence (AI) or radically different regimens,

11) telemedical consultation and investigation,

12) co-ordinated diagnosis and treatment plans which reduce duplications, omissions, and scheduling delays, and

13) patient monitoring devices.

These are features which are made obvious by today's applications in health care. If a full-featured network were available, there would likely be an emergence of new applications which would rely on the existence of a network.

Table 1 provides some insight into what might constitute a full-featured network by showing the network properties which might be expected for each of the applications. The different types of access are store-and-forward messaging and on-line, broadcast and real-time communication. Levels of confidentiality range from none, where there is no concern for confidentiality, to high, where there is great concern about unauthorized disclosure of information. Access restrictions refers to the types of permission which must be granted to a user in order to send, receive or retrieve data. The media type refers to the format of the data. In the cases where different qualifications might exist for a given application depending on the nature of the data, two or more are shown.

Table 1 is based on an optimistic model for network communication in which multi-media data transmission and broadcast, on-line and real-time connections are practical. Although some of the applications would be severely restricted by the exclusive use of text messaging, all applications could still be supported. Ideally, Table 1 would show the network applications ranked by their cost benefits. Since there is no network in place which provides all of these features, there is no direct method to obtain measurements of network utilization and the relative importance of each

application. A needs survey could be conducted but, for many of the applications, the respondents would be asked to estimate their use of services which are not available and for which the respondents might have only a vague appreciation.

Table 1. Network Applications and Their Properties

Network Application	Type of Access	Confi-dentiality	Access Restrictions	Media Type
1) Querying registries	messaging	high	by discipline, by individual	text
2) Educating	messaging, broadcast	low	by discipline, by enrollment	multi-media
3) Referencing	on-line	none	by enrollment	multi-media
4) Querying patient records	messaging, on-line	high	by entitlement	multi-media
5) Billing for services	messaging	moderate	by entitlement	text
6) Direct purchasing	messaging	low	by entitlement	text
7) Group purchasing	messaging	low	by entitlement	text
8) Collecting statistics	messaging	variable	by entitlement	text
9) Communicating *ad hoc*	messaging, on-line	high	by individual	multi-media
10) Using advanced systems	messaging, on-line, real-time	high	by entitlement	multi-media
11) Consulting	messaging, on-line	high	by individual	multi-media
12) Co-ordinating treatment plans	messaging, on-line	high	by entitlement	text
13) Monitoring patients	messaging, real-time	high	by individual	text, biometric

Certain applications are already available through local or regional networks. A number of hospitals are using EDI to place orders with suppliers and to make insurance claims. Physicians are also beginning to use networks for a few applications. Some of the more common applications summarized in Chapter 3 are the submission of billings and insurance claims by physicians, electronic distribution of laboratory test results to

physicians, and hospital Admission/Discharge/Transfer reporting to physicians. These applications are typically supported by electronic mail systems. In the research described in the ensuing chapters, it is observed that communication activities associated with the patient record such as the distribution of laboratory test results are traffic intensive and offer potential cost benefits.

In most applications, telematic procedures could offer improvements over the currently used procedures. Electronic transfer of data from one computerized data base to another reduces or eliminates transcription. Electronic data transmission is potentially faster than mail or telephone-supported voice communication. Networks can support multiple media conveniently and compactly whereas paper-based systems require manual handling of bulky, non-uniform documents. Networks facilitate accurate data duplication, distribution and storage so that more than one person can share a source document or a certified copy of the source document. Distributed data bases provide an economical means to scan, retrieve and analyze data from a much larger pool than would otherwise be available through manual procedures. Use of encryption can make electronic transmittal of data more secure than mail.

Networks could positively affect the quality of patient care. No transcription errors are introduced if no transcription is performed. Treatment plans are more easily co-ordinated if providers have a convenient means of communicating and exchanging observations. Increased access to records maintained by different institutions and providers provides a more complete description of the client. Ease of electronic transmission and inexpensive storage of observations encourages providers to maintain more comprehensive records (such as baseline observations) which can be recalled when needed to provide comparative client histories. Timely delivery of information used in decision making can result in more rapid treatments for deteriorating pathological conditions. Programs are more easily evaluated if data can be collected from wider a range of observers who impinge upon the activities of the clients rather than simply the program managers.

Networks could reduce health care costs. Data sharing decreases the number of redundant tests ordered by different practitioners for the same patient. Electronic storage, which is facilitated by electronic data interchange, is far less expensive to maintain than is paper. Electronic data is more easily distributed than is paper which must be physically transported. Less time is wasted waiting for receipt of client

records from other providers. No particular clerical skills are required to retrieve and transmit records to other providers.

Health care networks support a large number of applications, some of which are practical only in a network environment. The networks could simplify and improve procedures, improve the quality of patient care and reduce health care costs.

2.3 Orientation

No widely adopted health care network yet exists even though the technology for text based telecommunications has existed for over 30 years. One might assume that major reason for the delay has been the unavailability of low-cost computers. However, the Apple computer was introduced in 1975. This was followed six years later by the IBM PC. The MacIntosh computer which provided the first widely accepted graphical user interface (GUI) was released in 1984 [6]. Not only have low-cost computers been available for almost two decades but wide area networks have been commonplace in the retail, banking and travel sectors, since the mid-1980's. This gives rise to two fundamental questions regarding the implementation of health care networks:

> Why do health informaticians focus on health care specific networks which might be considered a subset of existing general-purpose wide area networks?
>
> Why does the evolution pattern of a wide area health care network differ from that of networks used for other purposes?

Use of telecommunication for the exchange of administrative and financial data is more prevalent than for clinical data. The rapid acceptance of EDI networks gives rise to a further question:

> What role do health care EDI networks play with respect to the exchange of clinical data?

2.3.1 Why the Exclusive Focus on Health Care Networks?

Although many of the prototype networks employ underlying networks supplied by a third-party network vendor, it appears that few health informaticians view networks for health care providers as merely a non-exclusive component of a much wider offering. In fact, a number of networks are specifically designed for an installation base in the health care sector [1, 7]. It is noteworthy that one report produced by the Science Council of British Columbia, which overviews fifty different information sharing projects in the health care sector, makes no effort to provide the reason for the exclusive focus. This report merely states "In most cases the users determined the orientation of the system." [7:i]. Another example of this peculiar oversight is evidenced in the proceedings of the IMIA Working Conference on Telematics in Medicine (held 18-21 November, 1992 in Rotterdam, The Netherlands). Here only a brief mention of the uniqueness of health care networks is given in the preface: "It is important that the exchange and use of information is well integrated in the information systems used in Health Care. Information models of Health Care applications are needed." [8:v]. What then might explain this orientation?

2.3.1.1 Data Interchange Standards

Data exchange among any group of network users requires a commonly accepted data format. User applications which access the data exchanged must conform to a common semantic interpretation which is uniquely oriented to health care.

Health care data are derived from a number of sources ranging from manually-recorded physician-patient consultations to highly-automated Magnetic Resonance Imaging (MRI). Data interchange standards are being developed for different areas [9, 10] such as laboratory data reporting (ASTM E-31) [11] and radiology (ACR/NEMA) [12]. EDI standards, or more precisely, the American National Standards Institute ANSI X12 standards, are being used for fiscal transactions relating to insurance claims and payments [13, 14].

More encompassing standards such as HL7 and IEEE MEDIX are evolving and converging to a single universal health care standard [15, 16]. A collection of EDIFACT standards (ISO 9735) are being developed and codified in the Netherlands [17]. These standards have been slow to develop. For example, the HL7 working

committee was established in 1987 [18]. HL7, version 3.0, which was to be released in 1991 [9] was postponed until mid-1992 [16] and was still not available at the close of 1993. None of these standards offers a comprehensive repertory of transactions which cover the full spectrum of health care.

2.3.1.2 Data Confidentiality, Authenticity, Non-repudiation

An *ad hoc* survey conducted by Jan Weingarten for *M.D. Computing* notes that improper access to computerized patient data presents a high level of concern to physicians. She concludes "... computer experts seem to agree that physicians and those they treat are subject to worrisome trends in access to personal information" [19]. Some of the trends referred to in the article are the unintentional disclosure of patient data through unauthorized access to computer systems, osmosis of sensitive information into financial systems, automated release of computerized patient medical records to other health care providers, collection and sale of mailing lists, and disclosure of patient information to employers. These concerns relate to policy and procedure weaknesses. There are several approaches being developed to control these types of unauthorized disclosure. The law provides some controlling criteria over confidentiality and authentication procedures [20]. There are also various schemes for data access controls which restrict data access to authorized users [21, 22, 23]. As the law is non-procedural and as criteria for establishing data access controls are not yet standard, network implementation and acceptance has been impeded.

There are fundamental ethical problems related to the ownership and use of medical data which have yet to be resolved [24, 25]. Without a clear policy of what constitutes consent, sharing of medical data is problematic. Networks which promote data sharing are adopted with some reluctance.

Theft, repudiation and falsification of data also present concerns to the health care and legal communities [20, 26]. Several common approaches to maintaining network security can be seen in the electronic banking, stockbroking and insurance industries [27]. Keyed Programmable Read Only Memory chips (PROM) provide a hardware signature for computer systems. Magnetic stripe cards, smartcards and passwords are frequently used to identify users. Data encryption is a common method used to prevent theft and electronic signatures provide positive user authentication. In effect, these procedures and devices ensure that the system is *closed* to all but entitled

users and computer systems. These provisions add cost and implementation overhead to a network. They also give application level network services a uniquely medical flavour.

2.3.1.3 Organizational Initiative

One of the largest general-purpose networks, the Internet, was originally developed by the U.S. Defense Advanced Projects Research Agency (DARPA). The project, which had its first node installed in 1969 at the University of California at Los Angeles, has been restructured and re-installed [28]. In 1986, the U.S. National Science Foundation (NSF) constructed a backbone network called DARPANET. A number of independent organizations joined the network and an Internet Activities Board was established to oversee the research, development and standards used by the network [29, 30]. The Internet, now numbering an estimated 1.5 million computers and 10 million users, derives its user base from governments, educational facilities and research facilities [31].

The Internet has been a highly successful network implementation. Government support has been responsible for its early rapid development. A very large user community which embraced computer technology at its inception has been responsible for its acceptance. A critical threshold has been reached where now many educational and research organizations find Internet participation a necessity for information exchange.

In comparison to the Internet, Health care network initiatives in most countries have not received the same level of funding nor are their user bases as receptive to computerization and information exchange. Although a health care network would be able to take advantage of a highly-evolved underlying network such as the Internet, it still entails a directed effort by the health care community to develop application protocols, define data interchange standards and promote wide-spread acceptance. Thus, the supporting organizations, which would be representative of the health care sector, help to determine the exclusive character of the resulting network.

2.3.2 Why Does the Evolution Pattern Differ?

There are a number of highly successful wide area networks to be found. Wide area point-of-sale systems are installed in most of the major department store chains. Many hardware stores make use of electronic ordering networks provided by major distributors. Credit card confirmation is performed at nearly every retail store in North America. The Interact system supports automatic banking machines installed by nearly all of the major North American commercial banks. Similarly, airline reservation networks like Gemini are used by many travel agencies. Telephone companies usually sell packet-switched and electronic mail services to the general public and companies such as CompuServe extend these services to include a range of value-added applications.

Although a number of network implementation attempts have been made by researchers, health care organizations and private enterprise, only limited health care networks exist. Typically the network applications which are supported are very specific, such as the submission of insurance billing claims. Often the networks exhibit only a short period of growth in functionality and membership before reaching stagnating maturity. Why then is the pattern of evolution for health care networks so different from other commercial networks when the subscriber base is so extensive and the need for data exchange so apparent?

2.3.2.1 Segmentation, Specialization and Private Enterprise

Many different types of enterprises comprise the health care community in North America. The types include government-operated organizations, government supported corporations, publicly-owned corporations, private limited corporations, not-for-profit corporations, community organizations, voluntary organizations, partnerships and private practices. Furthermore, many institutions which are generally classified as providing the same type of care, in fact specialize in the services they supply.

Canadian hospitals are a prime example of segmentation and specialization. Soderstrom constructs a matrix of hospitals by ownership and type of service that contains 10 active cells [32:24] which include, for example, publicly owned general acute care hospitals and privately owned mental hospitals. He also tabulates hospitals by eight mutually exclusive types of operators which include religious organizations,

the Red Cross, the Federal Government and others [32:25]. Although 94 percent of Canadian hospitals are public [32:25] and would superficially appear to be promoting common strategic and operational objectives, they are structured as separate corporations, each with an administration which sets its own priorities and reports to its own community-appointed board of trustees [33, 34]. Consequently, operational management of hospitals is widely dispersed. Only broad policy and high-level strategic co-ordination among hospitals is achieved through government ministries which provide registration, certification and/or funding, and voluntary organizations such as the Canadian Hospital Association.

Fragmentation of health care delivery is actually increasing as the emphasis changes from hospital acute care to ambulatory care, community care, home care and palliative care. The current trend toward re-privatization results in the further growth of private, independent health care institutions which provide services not only by care specialty but by client's ability to pay [35].

Because there are a large number of independent health care providers, most operational decisions and many strategic decisions are made independently. This is particularly true for the provider working in a private practice. Physicians, for example, are usually private entrepreneurs. Soderstrom finds that in 1973, only 2,760 of the 36,095 physicians were not engaged in private practice [32:104]. If we assume that the same ratio is applied to the 55,000 physicians registered in Canada in 1987, then there are potentially greater than 50,000 autonomous physicians making entrepreneurial decisions.

Other networking communities such as those found in banking, retail and travel usually have strong ties based on highly structured, well-established business practices. For example, electronic funds transfer is a very high-volume application which has little or no variation in interpretation from one banking institution to another. Although there might be differences in bank ownership, the practices relating to funds transfer have been used for centuries and the co-ordination between institutions has been well-established.

It is not obvious how major resource development and utilization decisions are to be made among independent individual and institutional health care providers. Many of the applications which could benefit from network communication have not been subjected to the same rigid practices that are found in other business sectors.

Furthermore, the degree of segmentation and specialization is possibly an order of magnitude greater in health care than it is in other industries. Co-ordinating organizations certainly have a role to play but these organizations often have a specific charter and a specialized membership. Government regulation and/or sponsorship may be the most likely method of ensuring co-ordination and accelerating the process of network implementation.

2.3.2.2 Fragmented Development and Competition

Most health care networks in North America are owned and operated by private sector companies. Governments have taken part in developing a few of the larger networks. For example, the Ontario Provincial Government awarded a contract to Green Shield in May 1993 to develop a network connecting 2,500 pharmacies and 80 dispensing physicians [36, 37]. A similar effort has begun in British Columbia with the Pharmacy Network Project [38, 39]. In these cases, although the initiative is sponsored by government, the operation of the network resides with a private insurer and a crown corporation, respectively.

The two pharmacare projects are very large in comparison to many other proprietary networks which are found throughout North America. Hospitals, insurance companies, private medical laboratories, clinic software developers and other companies have provided the base for a number of application-specific proprietary networks. The Province of British Columbia, has developed and is operating application-specific networks such as the Teleplan network [40] for physician billings and the Communicable Disease Surveillance System [41].

The competitive nature of private business does not encourage the co-operative development of a network unless the network is perceived as a marketing support for a revenue-generating product rather than being the product itself. Our experience in this research project indicates that co-operation from a private medical laboratory in network development is fairly easy to obtain since the enterprise is marketing laboratory services, not network services. However, a network supplier, such as a telephone company, is much less likely to embrace co-operative development if there is any danger of decreasing its share of market.

Another consequence of private sector development of network facilities is the proprietary nature of their networks. If an enterprise develops a network to promote its product line, unique protocols may be employed because the proprietary network can:

- be used to bond the company's customer base to a product which would otherwise be easily obtained from competitors,
- offer proprietary security features,
- be less expensive to develop, install and operate, and
- be customized to optimize the performance of the end-products.

Each application-specific network operating in a region lessens the incentive to develop an application-independent network. Private companies which operate these networks have no justification to make further investment in an open network unless there is a potential for increasing their revenue or prolonging their share of market. Public non-profit organizations are inhibited from expanding their networks because of the difficulty in obtaining funding for further development. Each proprietary network reduces the potential usefulness of an application-independent network because 1) the most cost-effective applications are already implemented, and 2) the added cost and complexity of accessing redundant networks discourages users from new non-integrated offerings.

If the premise is accepted that special-purpose, proprietary networks inhibit development of application-independent large-scale networks, why then should the health care sector suffer specifically from this problem while other sectors do not?

- Health care applications are diverse. There are some highly cost-justifiable applications such as the distribution of laboratory results, which receive early attention. Applications which are not as computerized, not as standardized, not so easily cost-justified, and not as closely controlled are invariably omitted from the early networking strategies.
- The health care sector already has an established base of application-specific networks. This inertia is difficult to overcome. From the perspective of the operators of these networks, a transition to a new, more open network represents loss of investment, loss of customers and increased risk.

- Health care organizations are varied and abundant. Many providers are interested in networks, however, only a few have a global perspective and even fewer have sufficiently large research and development budgets to undertake a project of this magnitude. Consequently, providers make smaller, proprietary efforts to resolve pressing application-specific problems.

2.3.2.3 Treatment Orientation and Product Packaging

Application-specific networks are the result of a treatment orientation which also characterizes a common approach to health care delivery. Rachlis and Kushner quaintly make the following observation regarding the treatment of illness:

> The Autobahn, Germany's most famous expressway, has no speed limit, and motor vehicle accidents are frequent and often result in loss of life. All along its length, Germany has established trauma centres to deal with the carnage. ... This "solution," in short, requires enormous resources to maintain, even though the problem could be handled more effectively by a simple preventive measure - establishing and enforcing a safe speed limit. [33:11-12]

This quotation highlights how organizations often take a reactive, treatment approach rather than the proactive, preventive approach to health care. A similar approach is taken when a health care network is developed: a specific communications *illness* is addressed by a timely implementation of a network which addresses the specific problem. The resulting network solves the immediate problem but it is rarely used or usable in its present form to solve other problems. From this perspective, the issue of network implementation simply resolves itself into being a lack of strategic planning.

Health care networks are promoted as part of an end-product. Consider Medline, a bibliographic referencing service of the U.S. National Library of Medicine. The product that is promoted is the referencing service, not the communications software. In another example, Ameritech, a private corporation marketing "a comprehensive information and communications solution for healthcare," states in their promotional literature: "We provide clinical databases and select hospital applications to build the foundation for computer-based patient records" [42]. In fact, the

proprietary network which Ameritech has developed is wholly application oriented; the user is unable to gain direct access to the network for non-Ameritech applications [43].

The health care field has many applications which can derive enhanced functionality from networks. As each health care provider is a potential purchaser, the application software is promoted by emphasizing its particular suitability for the end-user. The marketing of unique end-products cloaks an underlying need for an application-independent network which by itself is unmarketable.

2.3.3 What Is the Role of Health Care EDI Networks?

EDI networks for the exchange of health-related administrative and financial data are becoming common. For example, in the U.S.-based Virginia Project, the University of Virginia Medical Centre (UVAMC), Blue Cross and Blue Shield of Virginia, and over 200 other providers are involved in EDI transmission of electronic insurance claims and remittance advices [13, 14]. This project is one of nine pilots organized by the Workgroup for Electronic Data Interchange (WEDI).

WEDI is an unofficial, public-private task force composed of 53 representatives from provider organizations, payors and insurance companies, government, and vendors. The objective of the Workgroup is "to streamline health care administration by standardizing electronic communications across the industry" [13:1]. The committee was born out of a forum "convened by the Secretary of Health and Human services to address administrative costs in the nation's health care system [13:1]." Its chair and co-chairs are drawn from two insurance agencies: B.R. Tresnowski, president of Blue Cross and Blue Shield Association; and J.T. Brophy, past president of The Travelers Insurance Company. Sitting on the Steering Committee are such members as L.E. Jensen, Vice President of the American Medical Association; G. Belsey, Executive Vice President of the American Hospital Association, and Dr. A.H. Guay, Assistant Executive Director of the American Dental Association. The staff of the organization is drawn exclusively from the Blue Cross and Blue Shield Association and The Travelers Insurance Company.

The latest report submitted by WEDI in October, 1993 [13] makes a wide range of recommendations. Among the recommendations are:

- that there be enacted U.S. Federal regulations to *mandate* the use of data interchange standards defined by an accredited ANSI committee, ASC X12,
- that any Category I organization, "which includes major public and private payors, hospitals, employers with greater than 100 employees, self-insured plans, clinics, laboratories, DME suppliers, and group practices of 20 physicians or more or volume of 50,000 claims/encounters per year must implement ASC X12 standards by 4th Qtr of 94" [13:1-14],
- that any other organization (Category II), "must implement approved ASC X12 standards by 4th Qtr 95" [13:1-14],
- that WEDI, for lack of an existing *action group*, become the organization which ensures "the implementation guides for the core health care transactions until the action group is identified" [13:1-15], and
- that a health care action group be "empowered ... to work with ASC X12 to develop criteria for voluntarily certifying that health care participants meet minimum requirements necessary to conduct business in an EDI environment" [13:1-16].

The WEDI report is interesting for a number of reasons. WEDI has strong representation from the health insurance industry which is currently under pressure to simplify, rationalize and expedite claim procedures. WEDI predicates its recommendations on Federal and State intervention and regulation. (In 1992, two bills were introduced in the U.S. Congress to ensure electronic submission of claims [14].) These recommendations, if implemented, would result in many changes to existing claims procedures for large and small provider organizations, alike. within two years. Health care EDI providers will play a larger role in delivery of health care services. The administrative and financial aspects of health care delivery will dominate applications emerging from this initiative.

This initiative is a response to an urgent need of the U.S. health care system to deliver care more efficiently. While providing automation, the EDI initiative leaves the provider/payer/supplier relationships intact. The WEDI report projects savings of US $42 billion over a period of six years (including the cost of implementation) due to simplified settlement of claims [13:14]. It does not, because of its genesis, address the

basic need for generic telecommunication among health care providers. This does not imply that these needs will always be ignored, but that they are of a lower priority.

2.4 Areas Requiring Further Research and Development

At best, it is difficult to predict which path telemedicine will follow but there are certain activities which will foster its future development. Further work in the following areas is required:

- development of universally acceptable data interchange standards for major applications,
- implementation of Application Programming Interfaces (API's) to permit third-party application programs to access the network for messaging, on-line and real-time communications,
- selection of a transport protocol that operates cost-efficiently with different communications media for a range of data media,
- adoption and, if necessary, development of encryption algorithms,
- adoption and, if necessary, development of algorithms to generate electronic signatures and authenticity codes,
- adoption and, if necessary, development of algorithms to index, locate and retrieve data,
- development of access control algorithms and rules which could be used to release data automatically according to user need, user privilege, data ownership, data sensitivity and data validity, and
- development of a repertory of useful, appealing end-user applications with built-in data communication capabilities.

Most of all, the success of a health care network requires co-ordination among health care providers, researchers, software developers and telecommunications specialists. The existing technology is capable of fully supporting the applications outlined in Table 1. However, a network requires that users universally accept the communications environment.

Committees have been struck to co-ordinate several of these activities. For example, the ANSI Health Information Standards Planning Panel and the Healthcare Information Standards Committee are two examples of multi-disciplinary committees that are guiding the development of a unified data interchange standard. The use of committees introduces a problem: their recommendations are often an aggregation rather than a selection of the diverse design objectives proposed by members of the committee. Furthermore, the committees that have been formed do not cover all of the areas of development and there is no supervising committee whose goal is to establish a functioning network.

It may be more practical to adopt and build on a *de facto* network rather than to construct one from the top down. Ongoing development of the very successful DARPA Internet is an example of this approach. Committees play a role in accepting and certifying additions and changes in the core offering but individual member organizations assume the responsibility of performing the research and making the recommendations [30].

2.5 A Rationale for the Research in this Dissertation

British Columbia (B.C.) presents a good opportunity to develop a health care network because over 60 percent [44] of the 6,420 practicing physicians [4] already own personal computers in order to submit their Medical Service Plan (MSP) billings. The large installed base and the familiarity with an application requiring network support contribute to a favourable development environment for a more extensive, more functional wide area network. Except for the Teleplan network, several small experimental electronic mail projects and a hospital network, no health care network yet exists in B.C. Publicly available networks do not provide data security, data interchange standards and automatic background operation which are thought to be a prerequisite of a health care network. The network software must be able to 1) provide reliable communication over unreliable communication links such as the voice-grade telephone and 2) accommodate heterogeneous computer platforms, and 3) operate without interrupting the normal operation of application computers.

The research pursued in this dissertation is intended to produce prototype software appropriate for use in a health care network. The research includes investigation into:

- the feasibility of large-scale networks,
- the technical problems associated with providing secure and reliable communications to a large number of existing heterogeneous health care computers,
- the types of applications which would prove cost-effective,
- the effects of different topologies and call connection regimens on network loading, and
- the network operating costs.

The prototype forms the basis for further development which may lead to the successful implementation of a health care network in B.C. and in other regions.

3 Related Research

3.1 Introduction

A number of research initiatives into the development of telemedical networks are in progress. Some of the efforts are extensive multi-national efforts. Some of the projects are very large in scope and are in the process of assessing the requirements of health care providers, defining data interchange standards in specific applications, and implementing networks. A few are operational in a limited sense; restricted by function or by membership. The following sections describe three initiatives which are particularly significant.

3.2 The Netherlands: Inter-Institutional Information Exchange (3I) and Associated Projects

A co-ordinated effort to evaluate the benefits of electronic data interchange among health care providers has been ongoing since 1985 in The Netherlands. The Inter Institutional Information (3I) project is an initiative to develop electronic messages which can be exchanged and to study the costs, benefits and effectiveness of electronic communication in general practice [2, 45]. The 3I project has two participating pilot studies: MIRAGE [46, 47] and Communications Project Apeldorn (COPA) [48, 49].

In the course of the project, at least nine Electronic Data Interchange (EDI) messages have been defined in accordance with the Electronic Data Interchange For Administration Commerce and Transport (EDIFACT) syntax. These messages include the reporting of laboratory results; hospital admission, transfer and discharge; and various types of patient administration transactions.

The MIRAGE project was initiated by the Dienst Informatie Verwerking (DIV), the Electronic Data Processing Service in the city of Tilburg. After initial stirrings in 1985, when a network was devised with four GP's linked to the BAZIS hospital system, the project began in earnest in 1988 with 18 GP's and five hospitals [50, 51]. The original studies used the Dutch public telephone electronic mail system, MEMOCOM, to exchange hospital laboratory results, hospital admission and discharge messages, radiology reports and specialists' reports. At the end of 1990, the experimental system entered into production mode. At the end of 1991, nine hospitals

and over 100 GP's had joined the system. Further expansion as a private health care network is now being examined.

The COPA project, undertaken in Apeldorn by Erasmus University, was begun in 1988 with 27 GP's, 12 pharmacists and two hospitals. After a period of evaluation in 1990, this network continued to operate. By 1992, COPA had increased its user base to 34 GP's.

Like MIRAGE, COPA uses the MEMOCOM electronic mail system. The messages exchanged consist of hospital laboratory results, admission and discharge messages and free text messages. All physicians participating in the pilot have the same electronic patient record system, ELIAS. Both participating hospitals have computerized patient administration and laboratory reporting.

The conclusions of the 31 project were mixed because separate analyses were performed by 31, MIRAGE and COPA researchers. The MIRAGE research indicates that electronic transfers to an integrated electronic patient record system result in:

- a more rapid delivery of laboratory tests,
- an automatic facility for data input, and
- an increase in the net cost of record keeping.

Although GP's do not consider that rapid delivery of laboratory tests is important, automatic receipt and storage of electronic data is. Contrary to expectations, the cost of record keeping increases because manual systems must be maintained to accommodate data received from those GP's lacking electronic patient record systems.

The COPA study indicates that electronic transfers to an integrated electronic patient record system result in:

- a more rapid delivery of laboratory tests,
- a decrease in workload for 45 percent of the responding GP's,
- an elimination of data entry errors,
- an increase in the number of laboratory tests retained,
- an increase in knowledge about patient treatment, and
- an increase in the expense of communication to the GP which might be offset by a corresponding decrease by the hospitals.

The 31 analyses indicate that

- in fewer than 10 percent of the cases, earlier message arrival is important to the GP,
- electronic mail may result in savings of approximately Dfl. 1,000 (Cdn $700) per year per GP by the hospitals but there is little savings to be had overall,
- little benefit is realized through the reduction of manual transcription errors because a GP refers back to source documents on critical decisions, and
- little or no benefit results if there is no electronic patient record system present at each GP's office to receive the electronic data.

In summary, both pilot projects have been successful. The networks are slowly expanding in size. There appears to be little or no cost reduction directly attributable to electronic communication. The GP's who benefit most from the networks are those with fully integrated electronic patient record systems.

3.3 The United Kingdom: National Health Service Initiatives

The National Health Service (NHS) in the United Kingdom has been developing an information technology strategy since 1986. Included in this strategy is the development of a wide area network for data interchange. An organization known as the NHS Central OSI Team (COSIT) has been established to implement 1) a prototype OSI network (POSINET) as a reference site and as a consultancy network and 2) a demonstrator network in a health care environment [52, 53, 54, 55, 56]. As might be inferred from the name of the organization, COSIT, a heavy emphasis is placed on the Open System Interconnect (OSI) standards adopted by the International Standards Organization (ISO).

The Demonstrator Project has two components. The Oxford Regional Health Authority is participating with COSIT to provide doctors with electronic pathology reports mailed electronically from the John Radcliffe Hospital. As well, the Northampton General Hospital is using electronic mail to provide GP's with out-patient waiting times and in-patient waiting lists.

A number of application systems developers have co-operated with COSIT in integrating network functionality into their patient record software used by physicians.

This is particula ly important in the Oxford project as 79 percent of the practices are computerized [55:1].

Preliminary evaluation of the Oxford project has placed operating costs for the transmission of pathology reports in the range of £50 to £450 (Cdn $100 to $900) per year [54:1]. However, it is difficult to determine from this report exactly which costs have been included or what level of message traffic is supported. Further quantitative and qualitative evaluation of this pilot project is forthcoming.

In another project outside of COSIT supervision, the NHS Family Practitioner Service is developing a wide area network for data interchange among dentists, opticians, family doctors and pharmacists in England and Wales [57]. The Dental Estimates Board (DEB), the Prescription Pricing Authority (PPA) and the NHS Central Registry (NHSCR) are the major institutional nodes in the proposed network. Several small pilot projects have been conducted by the NHS Information Technology branch in preparation for the family practitioner network.

The British commitment to the European Community has imposed the objective of standards conformity on these exploratory projects. Vendors supplying network products are required to conform to a U.S. Government OSI Profile (GOSIP). For example, the CCITT X.400 recommendations for message handling have been implemented. Efforts to implement OSI standards have met with limited success because of heterogeneous computer platforms and restricted availability of conforming third-party network services. COSIT has been able to implement full protocol suites at only the major nodes in the POSINET network. At practitioner sites, non-standardized telephone communication is used. At many of the other sites, either the computer vendor's proprietary network product or PC-based protocol engines have been installed.

These pilots have experienced a problem with data interchange protocols. The EDIFACT standard has been adopted by the NHS but in many cases no suitable formats have been defined for the applications in place. Transaction formats are being developed for physician referrals and hospital discharges.

3.4 Europe: Advanced Informatics In Medicine (AIM)

The European Advanced Informatics in Medicine (AIM) initiative has 36 projects including two that are concerned with the development of wide area

telecommunication networks [58, 59]. Integrated System Architecture in Advanced Primary Care (ISAAC) supports research into the development of a medical workstation and the support system needed to connect GP's, specialists, laboratories, pharmacies and hospitals. Pilot studies are underway in Denmark, Germany, Greece, Italy and Portugal. Strategic Health Informatics Network for Europe (SHINE) is focused on community health support through regional health care networks. SHINE pilot studies are being conducted in Ireland, Italy and Portugal.

One of the early AIM projects, a European standard for Clinical Laboratory Data Exchange (EUCLIDES), was conducted in Belgium [26, 60]. This research focused on the definition of a standardized electronic mail network for the delivery of health care data. EUCLIDES used an X.400 message handling system which had a specialized User Agent (UA) designed for PC's. The Data Encryption Standard (DES) was used to selectively encrypt the message body. The DES encryption key was transmitted securely using the RSA public key crypto-system. EUCLIDES researchers constructed a proprietary message format which enclosed metadata describing the health data objects with the actual health data, in a manner somewhat similar to Abstract Syntax Notation 1 (ASN.1). It was hoped that the EUCLIDES data interchange format would become the accepted standard for Europe. EUCLIDES objects were defined for clinical laboratory tests, technical messages and billing messages.

By 1992, the EUCLIDES project had ended but given rise to a commercial venture, EUCLIDES Foundation International, which administers a wide area health care network [61]. The EUCLIDES message objects now include, among others, prescriptions, hospital admission and discharge summaries. EUCLIDES now supports EDIFACT and ASN.1 as well as the original EUCLIDES data interchange format.

Another example of the work associated with AIM is the German RACE project, TELEMED [62], which is being conducted in collaboration with SHINE. This is research into the development of a regional broadband network and PC-based multi-media workstations to be used by hospitals, specialists and GP's.

These projects are directed toward achieving a comprehensive open system for the exchange of health data among providers and associated payers and suppliers. Some of the pilots were already established before they became members of the AIM initiative. For example, the SHINE pilot in Lombardia, Italy, makes use of an existing 300 node wide area network used for health care administration. The purpose of these

pilots is to explore existing telematic applications, develop new telematic services, investigate and understand the uses of information within and among organizations, and align the information needs and standards of adjacent organizations.

3.5 Summary

This chapter gives an overview of three major co-ordinated research efforts in The Netherlands, the United Kingdom and Europe. Each of these projects involves hundreds of participants. The projects are funded directly or indirectly by governments. The most mature of the projects is that of The Netherlands which has attempted to assess the cost-benefits of data exchange for specific applications involving distribution of laboratory test results and hospital admission/discharge data. The scope of the other projects, particularly the AIM projects, is so vast that they defy timely, comprehensive evaluation. In these projects, what little is reported is fragmented and inconclusive.

Some of the prototypes such as MIRAGE and EUCLIDES exist now as commercial ventures. In addition, a consortium of vendors and researchers are involved in a commercial project, EUREKA EU645 MEDICINE, which proposes to develop a pan-European open health care network [63]. MEDICINE is built on the NHS Demonstrator and EUCLIDES projects and has the support of several major telecommunication and computer vendors.

A number of specialized networks are being developed in North America. For example, UVAMC is using EDI to exchange claims and remittances with Blue Cross and Blue Shield of Virginia [14]. Blue Cross and Blue Shield of Virginia, through its subsidiary, Health Communication Services, is supplying the network facilities to nearly all of Virginia's hospitals and 50 percent of its doctors [14:92]. There are some commercial ventures where Value Added Networks (VAN's) provide end-user applications and telecommunication such as where Ameritech operates the Wisconsin Health Information Network (WHIN) [42, 43]. Others, like Synapse developed by the University of Nebraska Medical Center, are co-operative ventures among health care providers [64]. An example of a large co-operative hospital network used primarily for transmission of administrative and fiscal data is that maintained by the University Hospitals Consortium in Illinois [65].

In British Columbia, several pilots have been conducted. The British Columbia Systems Corporation has constructed two prototype wide area networks among physicians and hospitals [66, 67]. Electronic mail among hospitals is now supported by the telecommunication network operated by the Province [68]. There is at least one commercial network, Medinet, which distributes medical laboratory tests using electronic mail.

Unlike the other projects described in this chapter, the North American health care networks are not well publicized and offer few formal publications. Many of these networks are developed for specific applications in small geographic regions and few have undergone any formal evaluation.

4 *Health Link*

4.1 Introduction

Health Link is a prototype set of programs developed to provide secure, networked, communication among heterogeneous computers supporting a wide range of health care applications. The original proposal, submitted in 1989, positioned *Health Link* as a support product to provide a convenient and secure communications vehicle for new and existing end-user applications. It was and is not intended to be marketed as value-added network service in itself. *Health Link* is characterized by the following design considerations:

1) it makes use of low cost, underlying communications technology such as the public switched telephone system,
2) it can operate on a personal computer running under Microsoft MS-DOS,
3) it requires only a small hardware configuration such as an IBM PC with 640 kilobytes of base memory and three megabytes of non-volatile storage,
4) it provides *automatic* pickup and delivery of messages in a store-and-forward mode,
5) it provides data encryption, message integrity validation, message authentication, network access authorization, sender identification and message receipt acknowledgment.
6) it provides gateway communications among multiple *Health Link* networks,
7) it provides data collection and reporting for network accounting, trouble shooting and monitoring, and
8) it supports message translation to and from the HL7 data interchange format.

Although these design parameters remain unchanged, the technology has advanced appreciably over the four year development period. The DOS platform is no longer as significant as it was in 1989 because sophisticated multi-tasking workstations running under operating systems like UNIX, Windows NT and OS/2 are available at competitive prices.

The prototype software that has been developed is unique because:

1) it has a low operating cost of approximately $50 per month per node,
2) it provides secure message store-and-forwarding among heterogeneous computer platforms,
3) it employs a simple, yet automatic, interface which necessitates little or no modification of existing application software,
4) it makes provision for use of a standard, application level, data interchange format for data transfer among heterogeneous application software packages, and
5) it provides reliable communication among non-dedicated nodes over a wide geographic area.

Most commercially available communication software packages do not support message store-and-forwarding with telephone communication, do not operate automatically, have no provision for data security, and do not support data interchange standards.

Health Link is targeted for use by all members of the health care community. It is perhaps best suited for those providers, such as doctors, dentists, pharmacists and community care workers, who have a low-volume communications requirement. It is suited less well for communication within a large institution or complex of institutions where a high number of transactions or a large amount of data is exchanged locally such as a hospital or a government ministry. *Health Link* can be used in contexts where the communications load is heavily unbalanced. For example, laboratories distribute test results with high frequency to a large number of doctors; but doctors communicate comparatively infrequently with other providers and with laboratories. A survey conducted at two Victoria medical clinics indicates that the load of a high-volume node might exceed that of a low-volume node by a ratio of 20 to one (see Chapter 6). To further exacerbate the imbalance, loading demands vary significantly according to time in ways which are characteristic of the provider or institution, not of the health care community in general.

This chapter addresses the design and implementation of the *Health Link* prototype. Timing and performance measurements derived from the prototype have been used to predict the behaviour of a large *Health Link* network through simulation

studies described in Chapters 5 and 6. Chapter 7 describes a prototype network which uses Health Link software to deliver messages among several medical clinics, a medical laboratory and a 200-bed hospital.

The next section of this chapter defines terms used to describe *Health Link*. This is followed by a description of the topology and platform specifications. The major software design decisions are presented: software structure, network architecture, message processing, message security, message scheduling, system integrity, gateway services and data interchange standards. Section 4.13 is a brief section which describes an abbreviated set of interactive features supported by the prototype. This is followed by a description of the software testing and validation techniques used in development of the software.

4.2 Definitions

Although Tanenbaum formally defines a *computer network* as "an interconnected collection of autonomous computers," [69:2] he intimates that it is an interconnected collection of autonomous computers which employ dedicated communication channels to exchange information. Similarly, Tanenbaum refers to a subnet as a set of interconnected hosts or end-user computers which access the network through an Interface Message Processor (IMP). A network, according to Tanenbaum's concept, is then a set of interconnected, autonomous IMP's which provide communication services to subnets of application computers. This approach is consistent with the ARPANET terminology and the Internet implementation [69, 30].

Clearly, *Health Link* fails to qualify as either a subnet or a network as its most commonly used medium is the public switched telephone network. *Health Link* most closely resembles PHONENET, which is a member of National Science Foundation's CSNET [69], and FidoNet, which is a widely used *ad hoc* network [70]. Use of the telephone network provides only intermittent connection and is used in a non-sharable mode by more than one device, for example, computer modem, facsimile (FAX) machine or telephone. In order to discuss *Health Link*, it is necessary to extend the meaning of some terms used to describe networks and network software.

A *network*, for our purposes, is a collection of autonomous computers which can gain access to a communications medium in accordance with a commonly agreed-upon schedule or by demand. A *subnet* is taken to be a collection of *Health Link* nodes

which are capable of communicating directly through point-to-point connections. An *internet* (small *"i"*), similar to the ARPANET-based Internet, is a collection of subnets which are accessible through *gateway* nodes (each of which exhibits functionality somewhat similar to the Internet IMP). The *Health Link* internet typically uses dedicated communication channels which are provided by a third-party network supplier in the form of packet switched services.

An *adapter* is a *Health Link* program which accesses a subnet and which can be run on an end-user application host or on a stand-alone *Health Link* computer which may, optionally, be an intermediary to an application host. Gateway functions are a superset of adapter functions.

Terminal emulation designates the functionality supported between an adapter acting as an intermediary to the subnet and an application host. This functionality is configurable and includes file copying as well as interactive terminal entry and display.

Topology, depending on the term's point of reference, relates to the organization of a typical constellation of either adapters or subnets. With *Health Link*, the specification of the communication media are non-critical because of the network's reliance on a very low common denominator which provides connection-oriented, low speed services.

Network architecture herein refers to the hierarchy of communication protocol layers which provide communication services.

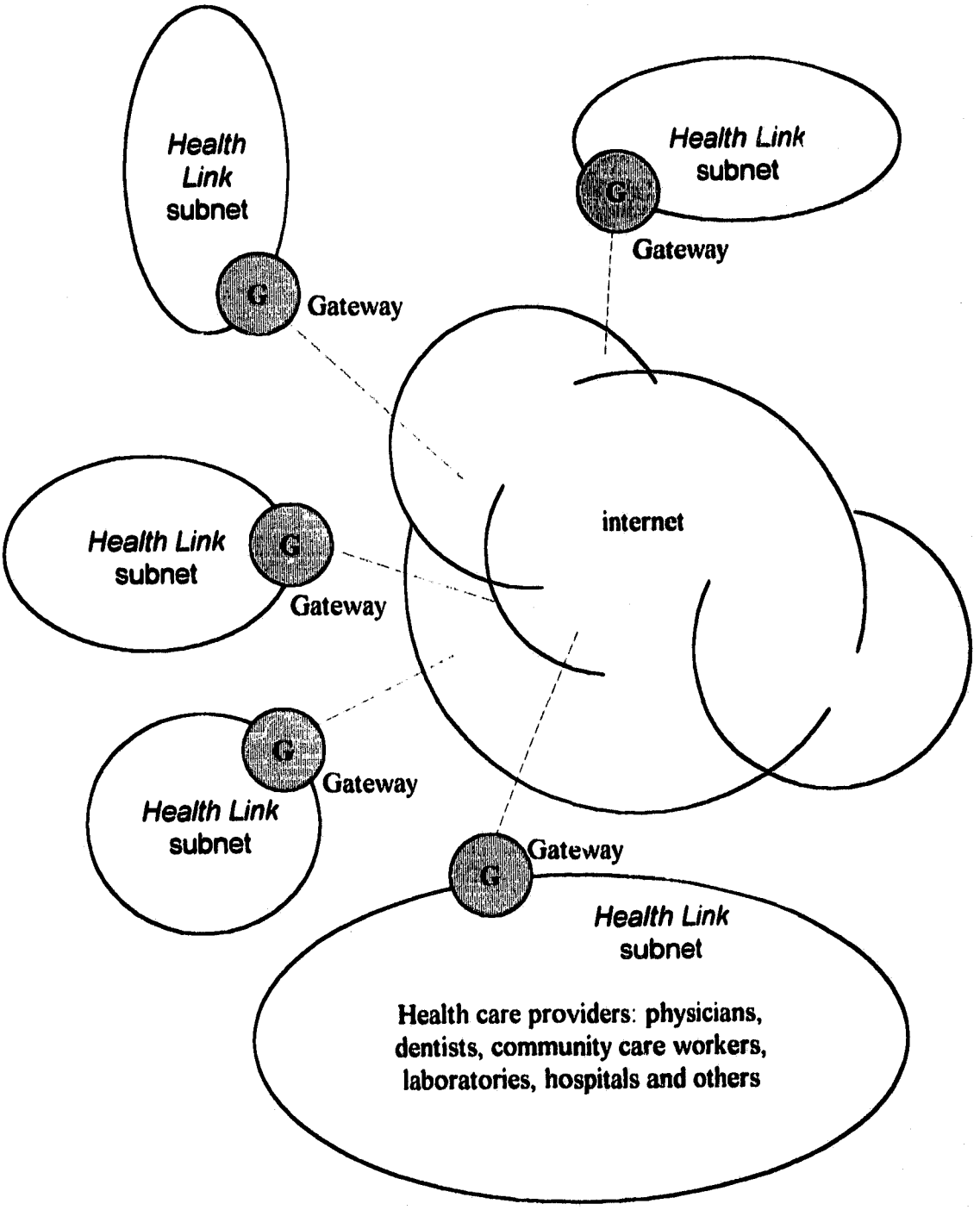
Software structure refers to program modularization [71] and call (or "uses") structure [72]. *Health Link* has been structured to take advantage of modularization for ease of testing, functional extensibility and software portability.

Message scheduling refers to the technique of triggering message delivery according to a preset plan which accounts for different levels of message priority and which attempts to optimize resource use.

4.3 Topology

In Figure 1, a *Health Link* network is shown as consisting of one or more subnets. If there is more than one subnet, interconnection is achieved by using a third-party underlying network such as Datapac [73]. The *Health Link* gateways provide access to the underlying network, which is referred to as an *internet* (which may or may not be the ARPANET-based Internet).

Figure 1. ***Health Link*** **Network**



A subnet consists of one or more *Health Link* adapters and one gateway connected, typically, by the public switched telephone system as shown in Figure 2. Although any underlying network can be used to support communication within a subnet, the adapters use an RS-232C serial interface and their own communications protocol which is designed for low speed analog communication. The maximum number of adapters in a subnet depends on the loading characteristics of each adapter and the gateway. To communicate within a subnet, usually one adapter makes a direct point-to-point connection with another. To communicate from an adapter in one subnet to an adapter in another, a message is stored and forwarded by the two intervening gateways.

An adapter, which is a node in a subnet, can be configured in one of three ways: 1) as a single purpose, dedicated, stand-alone computer, 2) as a single-purpose, dedicated computer which acts as an intermediary to an end-user computer, and 3) as a process on an end-user application computer (see Figure 3). It is emphasized that the adapter is merely a program which performs network-related functions. The three configuration options are provided in order to reduce installation complexity given the wide diversity of existing health care application systems.

The network topology is designed to be flexible for several reasons:

1) *Health Link* is readily expandable region by region as health care providers are clustered both in terms of geography and in terms of concentrated information interchange. The regional nature of communications is illustrated by the way in which physicians make frequent contact with local laboratories, pharmacies and acute care facilities.

2) *Health Link* can be introduced gradually without incurring major startup costs. An incremental approach is possible with the subnet topology. Each subnet carries the overhead cost of a gateway but no oversized central facility is needed in order to meet the projected growth of the entire network.

3) A densely subscribed regionalized subnet can be introduced more easily than a sparsely subscribed non-regionalized network [1]. Networks are more likely to be used if they are able to provide services to most of the communicating parties.

Figure 2. ***Health Link*** **Subnet**

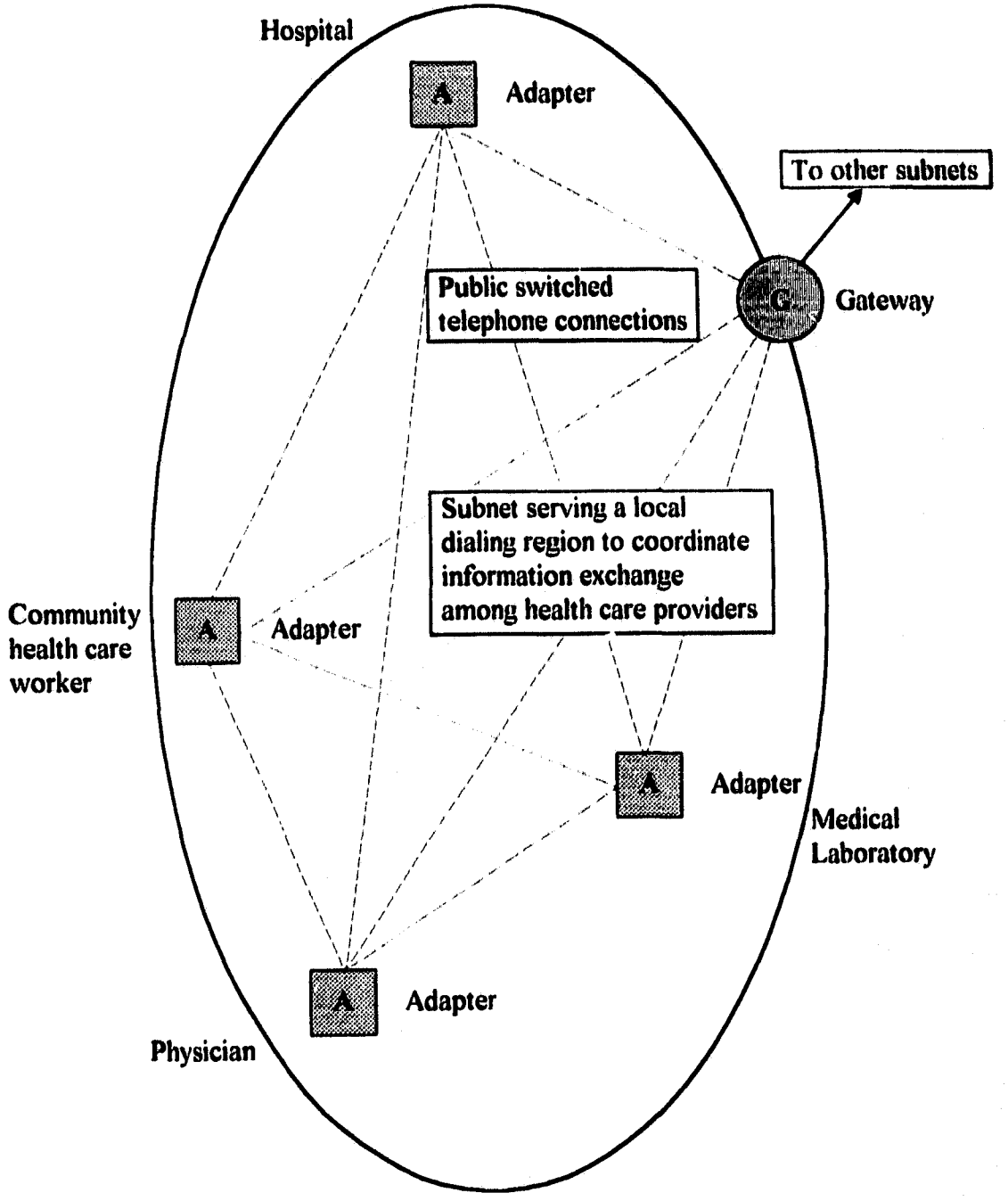
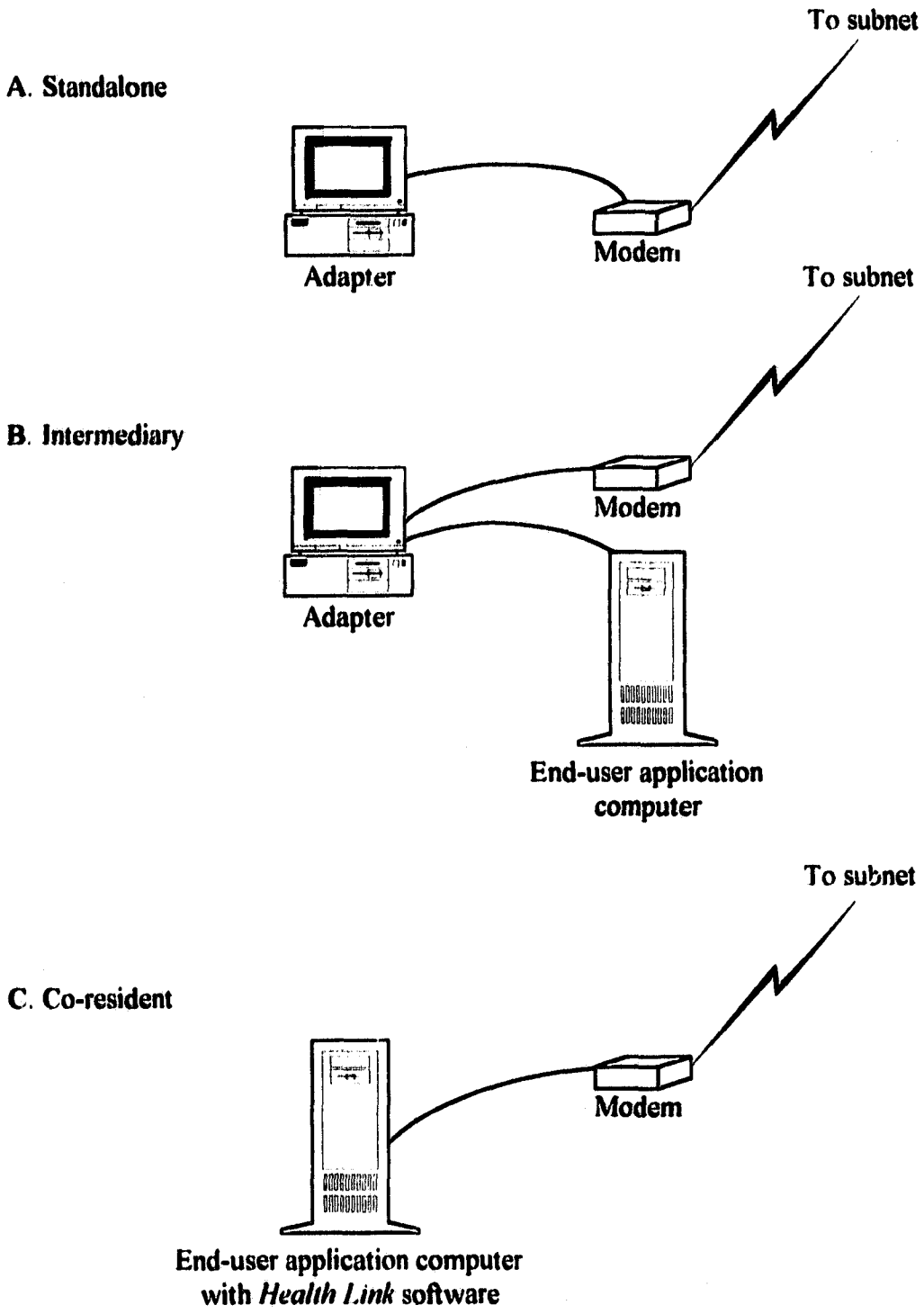


Figure 3. Adapter Configurations



4) Although some network management support is required for maintenance of the gateway nodes, much of this support could be provided remotely from a single network control centre. The more complicated network management functions can be safely left to the communications carrier supplying the underlying network.

5) Costs relating to network usage can be reduced as a result of reliance on point-to-point connections within a local telephone dialing region.

6) Gateway failure only incapacitates internet communication. For most transactions, an adapter requires no access to a gateway. If a gateway is unavailable for a few hours, there is no major impact on network behaviour because most communication is within a subnet.

7) Although *Health Link* is designed to store and forward messages, it is still possible to make use of on-line, value-added services such as health registries and bulletin boards. For on-line services to be implemented with *Health Link*, special inquiry nodes can be added which have a large number of ports to accept connections. The adapters themselves require no additional ports as the standard *Health Link* message-passing protocols are designed to compensate for unavailable adapters.

8) For typical intersubnet messages, only the source and destination adapters have a copy of a message. This provides a greater assurance of message confidentiality.

9) The subnet topology is flexible. Although no central message store is required, central message stores can be easily implemented. There is no reliance on any particular communications technology as *Health Link* can "tunnel" through any underlying network which provides connection-oriented services.

4.4 Platform Specifications

Health Link software is written primarily for an IBM-compatible PC operating under Microsoft MS-DOS (version 3.1 or higher). The following considerations resulted in the selection of the PC as the main development platform:

1) many health care providers have existing systems which are IBM-compatible PC's,

2) the PC has a wide variety of libraries and compilers for use in program development,

3) there is a large base of both users and program developers who are familiar with the PC, and

4) the IBM-compatible computers are sold at attractive prices.

Care has been taken in the software design and programming to permit easy adaptation to other platforms. Only one module is written in a language other than the C programming language. Third party libraries, Code Base 4 [74] and Essential Communications Library [75], have been incorporated only because they are written and supplied in the C programming language. The C source has been compiled using the Zortech C++ compiler [76] which provides a higher degree of error checking than do many of the standard C compilers.

Minimum hardware requirements for an adapter are:

- Intel 8088 central processing unit (CPU),
- 640 kilobytes of random access memory (RAM),
- 3 megabytes of non-volatile storage, either RAM or hard disk, and
- 1 RS-232C serial port(s)

A keyboard and monitor are required only if direct interaction with the adapter is desired. It is possible to access the adapter through an end-user application system, but, in this case, a second RS-232C serial port for the adapter is required. The amount of non-volatile storage required depends mostly on the maximum number of messages retained on the adapter, regardless of whether they are automatically stored for subsequent forwarding or they are manually saved by a user. A rough estimate of the non-volatile storage requirement is calculated by assuming two megabytes of fixed overhead and one megabyte for every 1,250 messages stored. It is further assumed that only text messages are exchanged. A survey of two medical clinics in Victoria (see Chapter 6) has revealed that the average text message for a GP is roughly 200 characters long and that approximately 106 messages are sent and received per day. (In calculating this mean, it is assumed that every data message sent results in a zero-length acknowledgment message.)

A subset of the adapter software has been converted to run on the application host to provide terminal emulation services. This is currently available on three

different platforms: 1) the IBM-compatible personal computer meeting the specifications above, 2) the Digital Equipment VAX operating under VMS and 3) the IBM RISC System/6000 with a UNIX-based operating system.

Gateway software is a superset of the adapter software. The gateway platform must meet the minimum specifications given for the adapter. It is recommended that it be configured with an Intel 80386 or 80486 CPU, 3 megabytes of extended RAM, a 120 megabyte fixed disk with a 15 millisecond access time, and at least two RS-232C ports. A rough estimate of the non-volatile store requirements is calculated by assuming five megabytes of fixed overhead and one megabyte for every 250 messages stored in transit (where a mean message size is taken to be 1,000 characters).

Usually physical communication links within a subnet are analog telephone lines. Modems, with a speed no slower than 2,400 bits per second, should be used for connections to the telephone system. These modems should conform to the Hayes command set. Not all modems need to be identical. The adapter software is capable of negotiating data rates and parity between dissimilar modems according to preset connection parameters.

4.5 Software Structure

There are three main programs used by *Health Link*: 1) the terminal emulation program (hterm) installed on the end-user application host, 2) the adapter program (adapter), and 3) the gateway program (gateway). These programs share many of the same modules. The terminal emulation program has the smallest, most restrictive combination of modules; the gateway program has the largest, most encompassing combination. Conditional compilation is used to determine which objects are included in a particular program. Conditional compilation is also used to control for coding variations targeted at different run-time platforms.

There are more than 60 modules which can be classified as follows:

- low level routines which include low level data base functions, physical device access, memory management functions, queuing functions and timer functions,
- high level data base routines which access individual, specific data bases,
- message processing routines,

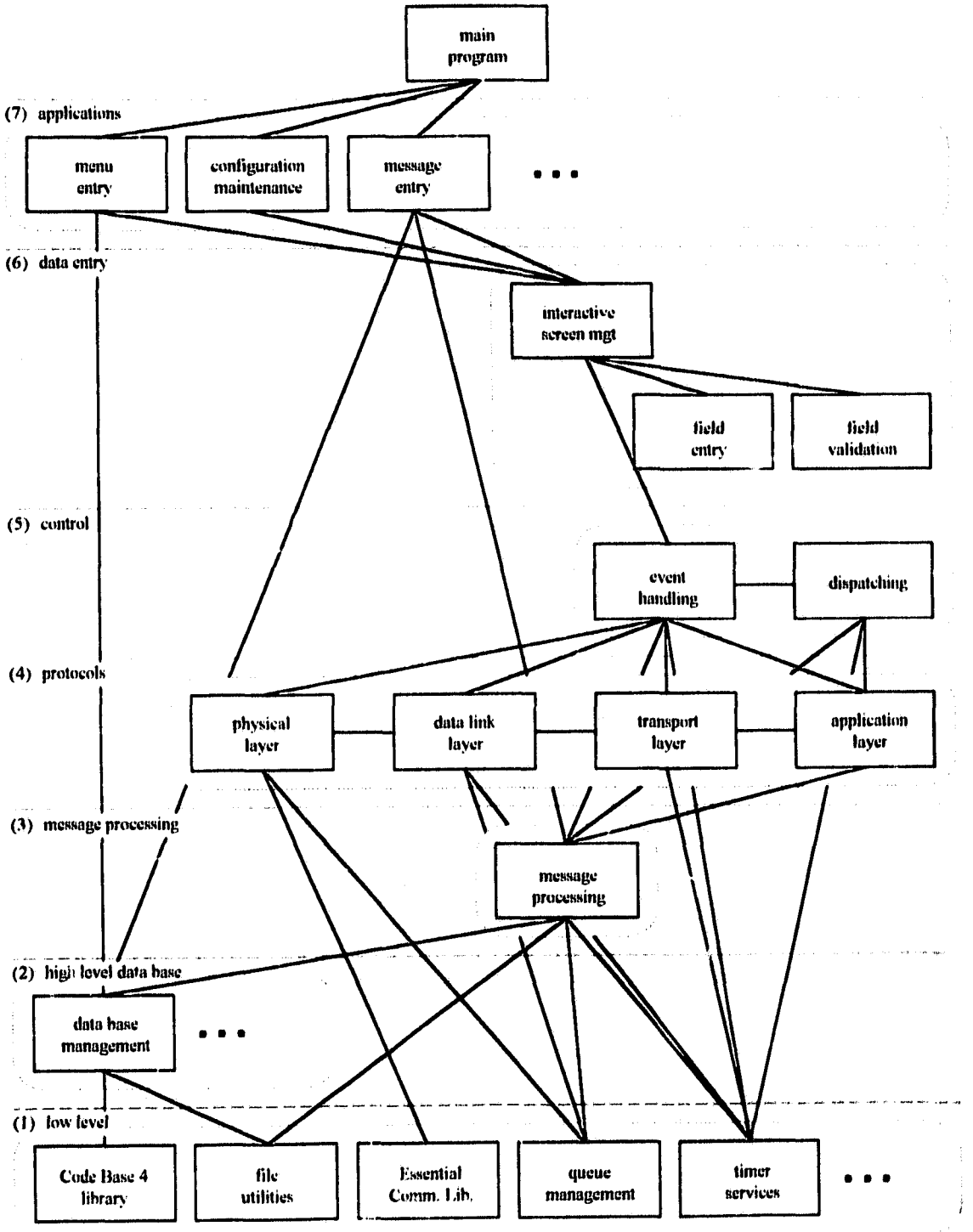
- communications and application protocols,
- control routines for event handling and dispatching,
- interactive data entry and display routines, and
- interactive application routines.

The call structure of the modules is, to a large degree, hierarchical. An outline of the call structure showing these classes and significant call dependencies is shown in Figure 4. Timer events, event handling and dispatching is implemented through call back functions.

Low level modules have been designed to minimize the impact of changes to the hardware and operating system on the programs. Similarly, an effort has been made to facilitate testing by maintaining a hierarchy similar to Dijkstra's "necklace" [77:47]. This approach promotes proof of module correctness based on the provable correctness of lower level, called modules. In general, the modules have been designed with attention to information hiding and encapsulation [72]. This has significantly eased development of multiple module versions where either improved performance or alternate implementation due to algorithmic changes or platform conversion is required.

The modules relating to communication protocols have complicated the program structure. This complexity is primarily due to the hardware and operating system limitations of the PC. The generic DOS-based PC is unable to provide true pre-emptive multi-tasking. The advantages that a sophisticated multi-tasking operating system provides are only available under later versions of Microsoft Windows which carries a high program development overhead. (*Health Link* has not yet been implemented under Microsoft Windows.) The DOS-based PC is also unable to provide a convenient set of user-accessible software interrupts. To avoid this difficulty, the *Health Link* programs simulate a multi-tasking environment by cycling perpetually from one task to another in a round robin. The modules are designed to provide their services within a short elapsed time in order for the program to give a reasonable interactive response.

Figure 4. Abbreviated *Health Link* Call Structure



Complexity in the program structure is further increased by the layered nature of the ISO OSI reference model [69, 78]. In order to make the module interfaces match those recommended by the OSI model, queues are used to provide service access points between layers. With the PC platform, an event loop is executed in a busy wait for an I/O event rather than trying to implement a system of software interrupts [79].

Specific considerations concerning program structure and modularization are given to the special requirements of the terminal emulation program, hterm. This program is designed to be run on a PC in the background as a terminate-stay-resident program (TSR) in order for other application programs to be run concurrently. Consequently, specific care has been taken to minimize the space requirements (executable code, static areas, stack and heap) of the program and to inhibit operations which dynamically restructure memory allocation during run-time.

Two libraries are used on the PC which provide low level modules. Access to physical communications ports is accomplished by calling modules in the Essential Communications Library (ECL) [75]. This library comes with a TSR program, xcomms, which services interrupts and copies data to and from buffers which are globally accessible in the PC's RAM. The library provides an interface to the global buffers and the xcomms program. For the VAX version of hterm, which does not use ECL, VMS system routines are called to service I/O interrupts.

A second library of routines is provided by the Code Base 4 library [74]. This library builds and maintains dBASE III Plus files using an interface which resembles the verbs found in the dBASE III Plus language [80]. *Health Link* modules, which make calls to the Code Base 4 library, are written for each data base file. In this way, the interface to library routines can be hidden from higher level modules and records stored with dBASE field definitions can be translated to C language structures.

Tables 2 and 3 give statistics on the software modules implemented. Overall, there are 93 production modules in the system. As well, there are five simulation programs (described in Chapters 5 and 6) which constitute an additional 19 modules. Table 3 lists the modules in the adapter program by their call level as reflected in Figure 4. Call level 8 represented by pmenu, is the main program

Table 2. Software Statistics

	Health Link Prototype	Simulation Programs	Total
Number of Modules	93	19	112
Lines of Production Code	32,000	10,000	42,000
Lines of Test Code	21,000	0	21,000
Lines of Documentation	11,000	600	10,600

Table 3. Lines of Source Code and Module Documentation for the Adapter Program

Call Level	Module Mnemonic	Lines of Code/Text Prod-uction	Test	Docu-ment
8	PMENU	237	40	72
7	BCKUP	196	0	56
7	CFGPB	616	183	30
7	FILDIR	145	150	69
7	FNDPB	195	90	76
7	FNDUSR	194	99	81
7	MDYSCR	344	107	84
7	MENU	185	80	75
7	MFWSCR	55	102	50
7	MSGDIR	317	101	105
7	MSGENT	310	141	85
7	MSGHDR	252	160	91
7	PMPRV	351	71	108
7	PMRTE	209	68	71
7	PMSCHD	458	69	70
7	PREORG	142	105	69
7	PSTART	284	304	68
7	PSWRD	107	133	75
7	SYSTAT	400	179	64
7	ULOGON	107	99	78
6	EDTFLD	374	176	108
6	EMUTRM	424	153	103
6	GETFLD	222	118	155
6	GETSCR	401	0	66
6	SCRMGT	385	174	216
5	DSPTCH	980	237	116
5	EVNTLP	203	182	77
4	FAPP	1,769	653	409
4	LLAY	975	620	82
4	PLAY	1,356	435	483
4	TLAY	1,237	659	0
3	CURO	832	52	332
3	PMMSG	1,445	949	350
2	C00MGT	464	317	174
2	C01MGT	286	318	169
2	C02MGT	391	344	174
2	C03MGT	390	491	184
2	C05MGT	220	268	165
2	C06MGT	204	254	165
2	C14MGT	227	294	166
2	CB4IF	399	92	192
1	ALLOCB	88	105	35
1	CDPRES	428	237	100
1	CIPHER	297	539	92
1	DIGEST	51	225	43
1	F00MGT	88	99	34
1	F01MGT	419	517	183
1	F02MGT	99	110	76
1	FILHDR	697	274	136
1	FMTASC	159	232	37
1	FMTFRM	93	145	51
1	FUTIL	667	318	187
1	HARITH	441	208	71
1	LGCLK	61	95	56
1	PCIPHR	105	284	45
1	QUEUE	93	173	52
1	RUNLEN	69	90	39
1	STACK	51	78	47
1	TIMER	148	279	64
1	UTIL	299	126	62
	Total	22,641	13,201	6,773

4.6 Network Architecture

Health Link uses a layered communications protocol which follows the OSI reference model [69, 78]. The software has distinct physical, data link and transport layers. The network layer has been omitted since the primary services offered by this layer deal primarily with internetwork routing, a problem which lies beyond the subnet. These services, instead, are provided at the internet. The gateway nodes, too, avoid these problems by assuming that the internet services are provided by a third party supplier and are fully reliable. Although the network layer can be disregarded, most of the other layers are needed to provide reliable connection-oriented communication between any two nodes in the subnet.

The *Health Link* network architecture is non-standard with respect to the *de facto* TCP/IP standard and the formal OSI standard. Should the need ever arise that *Health Link* must conform to either of these protocols, it is easy to remove the link and physical layers and replace them with a BSD UNIX socket [81] or an appropriate OSI link and physical layer protocol, respectively. It should be remarked that Health Link is similar to other serial link protocols, e.g., Kermit, X-Modem, Y-Modem, Z-Modem or SLIP, in its lack of conformity to any formal standard. The *Health Link* protocol is designed to be at least as reliable as any of the comparable serial link protocols.

Protocol substitution becomes more difficult at layers above the link layer because of software extensions at the transport and application layer which, among other functions, are designed to queue and retrieve messages, schedule connections, and control data flow. If the replacement protocol is fully-featured, *Health Link* can use *tunneling* to exchange data [30]. In this case, the protocol on the supporting network is merely treated like any physical connection by the *Health Link* protocol. This effectively encapsulates the *Health Link* protocol in that of the supporting network.

At the top of the protocol stack, there is a file transfer protocol implemented in the application layer which supplies services to a dispatcher. The application layer also incorporates those features of the session layer needed in order to execute multiple concurrent operations. The presentation layer is omitted as the main presentation issues are handled off-line (not in real time) in the message processing module.

Terminal emulation is achieved through inband signaling on the same physical link that is used by the data link, transport and application layers. A full-duplex stream

of emulation characters, conforming to a subset of the Digital Equipment Corporation's VT100 codes [82], are interspersed among the frames exchanged by the link layer. To avoid the overhead incurred by using such presentation/application protocols as Virtual Terminal (VT), this stream of character codes is redirected by the physical layer to an independent interactive services layer. Figure 5 is a diagram of the network architecture which shows both the layers belonging to the OSI protocol stack and the interactive services layer.

This section presents a discussion of each layer implemented in *Health Link* and then presents a few benchmark timings for modem-based communications which will be referenced in later chapters.

4.6.1 Physical Layer

The physical layer provides the following functions:

1) Port selection and allocation

   It searches for an available physical port, calculates a set of negotiated parameters compatible with the destination (if one is specified), and opens the port.

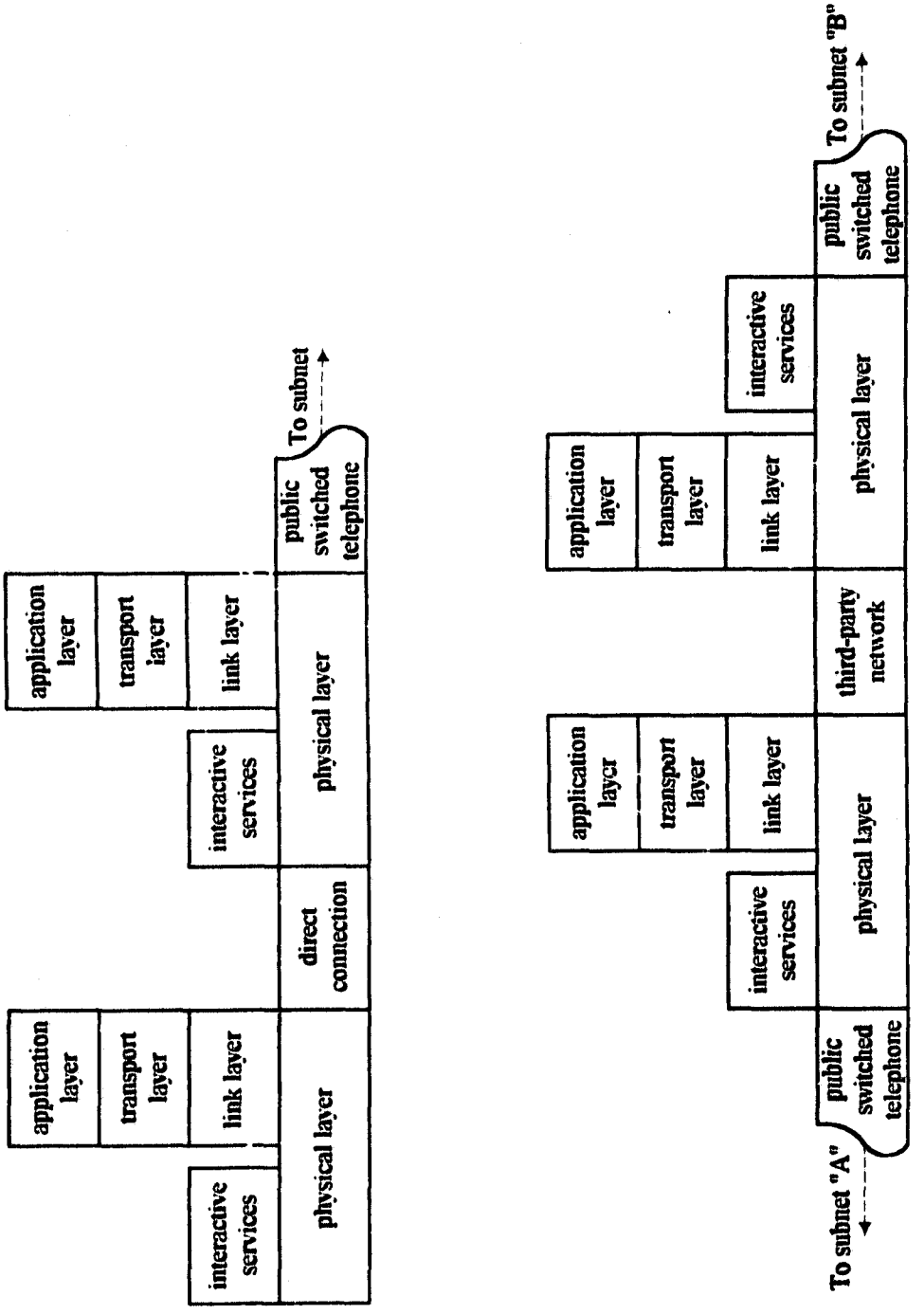
2) Physical connection and disconnection

   It verifies that the destination is reachable. It executes a connection script to a) initialize the local Data Communicating Equipment (DCE) (such as a modem or packet assembler/disassembler), b) establish the route to the destination, c) initialize the remote DCE, and d) verify that a successful connection has been made. Once the connection is made, it monitors various time out conditions, data carrier detect, data set ready and flow control conditions.

3) Virtual connection

   This function establishes a virtual connection (channel), one which loops back to the protocol stack. The channel permits applications to be executed locally on the same computer even though they are designed for execution on remote systems. A virtual connection is broken only if the physical layer is terminated.

Figure 5. ***Health Link*** **Network Architecture**



4) Emulation connection

This function checks the setup configuration parameters and, if an emulation connection is defined, it automatically establishes a physical connection. Subsequent inband signals are re-directed to and from modules outside of the standard protocol stack. An emulation connection is broken only if the physical layer is terminated. Only one emulation connection is supported at the physical layer.

5) Packet processing

Packets sent and received are prefixed with an ASCII character 1 (SOH) and suffixed with an ASCII character 13 (CR). To ensure packet integrity, a CRC-CCITT is affixed [83]. If an invalid packet arrives, it is merely discarded. Any character which is not within a packet is treated as a terminal emulation character. Characters within packets are translated to ASCII printable characters to prevent them from being interpreted as data link escapes by the DCE. Packet data are encrypted using a polyalphabetic algorithm.

These functions are an extended set of functions commonly provided by the OSI reference model. Most notable is the packet processing function which typically is the responsibility of the link layer. Packet framing is required to ensure that inband signals are detected and re-directed. Printable character translation is needed to prevent accidental data link escapes due to illegal characters, (the physical layer makes only limited assumptions about the DCE). Data encryption is considered desirable for transmission on insecure data links as the message header information would otherwise be clear text. (No data encryption is performed on the virtual or emulation channels.)

The physical layer has two service access points: one for packet transmission and one for terminal emulation. The packets received by the link layer have data integrity but may have sequencing errors. The terminal emulation stream received by the interactive services layer does not have guaranteed data integrity (other than that provided by parity checks if the link is so configured), but the characters in the stream are received in sequence.

4.6.2 Data Link Layer

Functions provided by the data link layer are as follows:

1) Connection establishment and release

The link layer services connection requests and disconnect requests made by its caller, the transport layer. These requests are passed to the physical layer to establish a physical connection. If the connection is successful, a connect confirm is passed to the transport layer. The connection Service Data Unit (SDU) contains a connection identifier for the remote node which is the local node's address.

2) Formatting of Protocol Data Units (PDU's)

PDU's being sent (or received) by the transport layer are presented to (or delivered by) the physical layer with associated Interface Control Information (ICI) containing the frame type, the framing sequence numbers (both for sending and receiving), the length of data in the frame and the data payload.

3) Sequencing of data units

The sequencing of PDU's is maintained by the link layer. A one-bit sliding window protocol [69] is used to verify that frames are received in sequence. The protocol is full-duplex which allows frame acknowledgments to be piggy backed.

The one-bit sliding window implies that a sender permits only one frame to be left unacknowledged at any time. This protocol has the advantage that it is full-duplex, simple to implement and frugal with the buffer space needed to cache transmitted frames. The protocol has the disadvantage that it is a variation of the *stop-and-wait* protocol where the sender pauses for an acknowledgment after each frame is sent. When the protocol is operating in full-duplex mode (while sending and receiving frames of the same size) the speed of each channel is roughly half that of a single channel operating in half-duplex mode.

4) Error recovery

Frame sequencing errors are detected and corrected. Duplicate frames are discarded. Frames with sequence numbers lying outside the one-bit sliding window are ignored. Frames not received, as determined by a missing acknowledgment, are retransmitted. A sender will attempt up to four frame retransmissions spaced at 10 second intervals.

5) Flow control

Flow control for incoming packets is accomplished by sending back "busy" frames. A node is deemed busy if it has insufficient memory to process the frame received. A node receiving a "busy" frame waits 20 seconds and then retransmits the same frame.

6) Listening

The link layer manages those data links which can accept calls by issuing listen connect and listen disconnect requests to the physical layer. Links which accept calls are also considered to originate calls. If no idle links are available for an outgoing call, the listen links are reviewed. If a listen link is found, a listen disconnect request is issued followed by a connect request.

After 20 minutes of listening without any activity, a link is disconnected and a new listen request is issued. This corrects for a DCE which has gone off-line and needs to be re-initialized.

The transport layer is permitted to issue listen requests to the link layer even though no listen links may be available; positive listen confirms are always returned. When a link, which accepts calls, goes to an idle state, a listen request is automatically issued to the physical layer.

The link layer has an abbreviated set of IEEE 802 Logical Link Control (LLC) primitives: L-CONNECT.request, L-CONNECT.indication, L-CONNECT.response, L-CONNECT.confirm, L-DISCONNECT.request, L-DISCONNECT.indication, L-DISCONNECT.response, L-DISCONNECT.confirm, L-DATA-CONNECT.request, L-DATA-CONNECT.indication, L-DATA-CONNECT.response and L-DATA-CONNECT.confirm. The standard is supplemented with L-LISTEN.request and L-LISTEN.confirm. Tags assigned by the transport layer to a service request are returned with the confirmation. A confirmation or indication is presented to the transport layer through a Service Access Point (SAP) which queues the SDU's in a First-Come, First-Served (FCFS) order.

There is no provision to expedite data in the *Health Link* protocol. In the OSI protocol, this service resides in the network layer, but because *Health Link* has no network layer, it would logically be implemented in the link layer. It has not been implemented here because 1) it requires pre-emptive queueing which adds additional

programming complexity to the prototype and 2) it is not needed by any of the higher layers in the protocol stack.

4.6.3 Transport Layer

The transport layer lies directly above the data link layer as no network layer functions are necessary for *Health Link*. The transport layer provides the following functions:

1) Mapping of transport addresses

   Transport connections to a single destination are mapped to a data link address. Each connection is uniquely defined with a transport handle which is passed back to the application layer whenever an association request is made. This handle is cross-referenced two ways: 1) to the link destination address and 2) to a handle supplied by the application.

2) Multiplexing

   More than one transport connection can be established concurrently with a given destination. Data requests are queued for sending on the same data link. Similarly, data indications received by the transport layer are distributed and queued according to the designated transport connection.

3) Connection establishment and release

   The transport layer accepts connection requests for a specific destination transport identification on a specific link. It, in turn, executes an L-CONNECT.request primitive if no link exists already. If a link exists, it establishes a transport connection on the existing link by issuing a connect PDU to the remote node. When the PDU is confirmed, the transport layer notifies the application layer by queuing a connect confirm.

   When a disconnect SDU is received from the application layer, the transport connection is released after a disconnect PDU is confirmed by the remote node. The application layer is notified of the connection release with a disconnect confirm. If no transport connections remain on the link, an L-DISCONNECT.request primitive is executed.

4) Data segmenting, blocking and concatenation

The transport layer can accept an SDU from the application layer which may be too large for transmission. The SDU is split into blocks which are then sent in sequence. At the destination, the segments are reassembled into the original SDU which is presented at the transport SAP.

5) Flow control for individual connections

The receiving node can control the rate at which data is sent by using the technique of issuing data credits. The receiver reserves a buffer and then issues a credit to the sender for the amount of data needed to fill the buffer. The sender is permitted to send data up to the amount credited. Once the transfer is completed and a new empty buffer found, the receiver can issue a new credit to the sender.

6) Name serving

The transport layer provides for incoming calls by setting up a name server for each listening application process. Each process names itself according to network-wide standard identifier and notifies the transport layer that it is accepting calls (listening). Upon receipt of an incoming connect indication, the transport layer searches for the name of the destination process. If it is found, a unique, unused name is substituted and an association indication is queued for the application process. The application process can then establish another name server.

Like the data link layer, the transport layer supports a subset of IEEE primitives: T-CONNECT.request, T-CONNECT.indication, T-CONNECT.response, T-CONNECT.confirm, T-DISCONNECT.request and T-DISCONNECT.indication. It supplements these primitives with T-LISTEN.request and T-LISTEN.confirm. T-SEND.request, T-SEND.confirm, T-RECEIVE.request and T-RECEIVE.confirm, implemented for flow control, replace the standard T-DATA.request and T-DATA.indication. Several other routines are available to abort connections or prevent establishment of new connections.

Virtual and emulation channels are never disconnected at the physical layer regardless of time outs or other communication errors. Consequently, if either the link layer or the physical layer should fail, no disconnect indication is received by the transport layer and connections on these channels are not released. To avoid stagnated

connections on the virtual or emulation channels, connections are aborted after one minute of inactivity.

4.6.4 Application Layer

The application layer is a routine which performs basic file and message operations in a manner similar to TCP/IP File Transfer Protocol (FTP) [30]. Unlike the OSI reference model, this layer does not provide a set of Association Control Service Element (ACSE) primitives with an associated Common Application Service Element (CASE) such as File Transfer, Access and Management (FTAM) [69]. It is referred to as a layer because all message and file transfer procedures use this routine to transfer data. As such, it performs the same role as does a CASE in conjunction with ACSE's [69].

The application layer performs operations which are known to the client as:
Push: a file or message is sent to a remote address,
Pull: a file or message is received upon demand,
Fildir: a file directory is received upon demand,
Fildel: a file is deleted upon demand, and
Noop: an association is made with a remote address.

An application procedure queues a request to be serviced by the application layer. This causes the application layer to set up a client which initiates the operation. When the operation is complete, the application layer notifies the registered call back routine, typically the application procedure which originated the request. A server at the remote node performs the operation specified by the client.

Each operation is initiated by executing a T-CONNECT.request. When the operation is complete, a T-DISCONNECT.request is made. For the link to remain open, at least one operation must be established. A link is permitted to carry a maximum of two operations at any time, one initiated by either node. To prevent premature link disconnect, a node is permitted to review its queue for additional pending operations as each operation completes. A Noop operation causes a connection to be made at the transport layer without actually causing a file or message operation to be performed. This operation is implemented for the scheduled message pickup feature

described in Section 4.9. The Noop initiates a scheduled connection with a remote node so that the remote node can deliver any pending messages.

The application layer permits copying operations to be mixed between files and messages. For example, a file containing a message header and data can be copied into the message base as a message. Similarly, a message in the message base can be copied to a file either with or without a message header. The data in a file or message may be text or binary. A copy operation can specify that the source message be destroyed after it has been successfully transferred.

The application layer uses a protocol which specifies the operation to be performed, transfers data and then sends a status SDU upon completion. A status SDU can be sent at any time to cause an operation to terminate. To ensure that the source file or message is not inadvertently destroyed, any copy operation requires a two-way status handshake when it completes.

The application procedure can prescribe one of four actions to be taken if an operation fails:

1) take no action, discard the operation,

2) retry the operation for up to MaxNumberRetries,

3) retry the operation for up to MaxNumberRetries, then requeue the operation while designating the remote node as the gateway, and

4) notify the application call back routine.

In those situations where retries are required, the setup configuration determines the number of retries permitted and the time interval between retries.

4.6.5 Interactive Services Layer

The interactive services layer lies above the physical layer and supplies four access routines: SendInband, RecvInband, RecvKb and SendScreen. The first two, SendInband and RecvInband, access a pre-established communication channel to send and receive a buffer of characters. The RecvKb routine returns characters typed on the local keyboard. The SendScreen routine displays characters on the local screen. All routines are written for VT100-compatible devices, that is, VT100 escape sequences are presented to the interface.

The interactive services layer is accessed through two routines: GetKbCharacter and SetScreenString. The function of this layer is predominantly one of control: it determines, according the program configuration, how the signals sent (received) by the physical layer are to be distributed (combined). For example, if the program is configured to accept characters from an emulation channel as keyboard input, the keyboard input at the adapter, if any, is merged with those characters received on the channel. For this same case, characters sent for screen display at the adapter are also transmitted on the emulation channel. This is a typical configuration for an adapter attached to a PC application host.

Services from this layer are used predominantly by a set of modules which provide full-screen entry and display to interactive applications.

4.6.6 Benchmark Timings

Connect and disconnect times and data transfer rates are available from benchmark tests. The timing measurements presented in Table 4 were taken using an adapter with a 2,400 bits per second (bps) modem. These measurements, although representative, are very much subject to changes in the modem command scripts and the physical characteristics of the modems and telephone line. The LineIdle time, a period of inactivity which is taken to indicate that there is no connection, is a configurable parameter which has been set to 60 seconds for all results reported in this document.

Table 4. Connect and Disconnect Timing Benchmarks (seconds)

Procedure	Result	Time (seconds)
Connect	Carrier detected	27
	Busy	15
	No dialtone	10 + LineIdle
	No answer	42 + LineIdle
	No carrier	43 + LineIdle
Listen	Port listening	5
Disconnect	Port Idle	7

A number of factors influence the effective data transfer rate beyond the quality of line service, the speed of the DCE and the response time of the adapter. The factors are as follows.

1) The length of the transmitted message is different from the clear text message because of the data compression and data encoding routines.

2) The message is blocked and framing characters are added.

3) The message is accompanied by a 292 character header which carries addressing information.

4) The adapter may be both sending and receiving data packets as opposed to simply sending a message and receiving acknowledgment packets.

5) Upper layer protocols add additional control SDU's which do not actually carry a data payload.

Tests made with 1,000 character messages composed of lines using an 80 character alphabet, have reported best case effective data transfer rates of 770 bps with 2,400 bps modems. Best case raw data transfer rates are in excess of 1,230 bps, a throughput efficiency of better than 51 percent. A constant initialization time delay of several seconds distorts these data rates for small messages. The throughput efficiency is comparable to that for *classic Kermit* which is reported by Lee and Chanson [84] to be 62 percent at 4,800 bps.

4.6.7 Summary

Health Link has attempted to conform to the architecture of the OSI reference model. The network and presentation layer have been omitted and the application layer has been significantly simplified. The interactive services layer has been added. These digressions from the model are in order to simplify the implementation, to improve response and to work within the restrictions imposed by a limited-featured platform. Although *Health Link* is not an implementation of an open system, this restriction is limited only to the subnet because third party internetworking is used. Furthermore, if full compliance to ISO standards is required, the network architecture of *Health Link* is sufficiently similar to make protocol layer replacement possible. For the PC, this

would most likely require the installation of specialized communications boards with on-board processors like the Frontier Technologies OSI products [85].

Few wide area store-and-forward networks for the communication of health care data employ a fully compliant OSI approach [7, 86]. Even some of the most sophisticated systems, such as EUCLIDES [26, 63] and LIFELINE [1], provide a mix of standard and non-standard protocols. Most of the networks are non-compliant at the final point of message acquisition or delivery.

4.7 Message Processing

Health Link relies on a sophisticated procedure to prepare a message for sending. Each message passes through stages which load the message into the message base, and then address, compress, encrypt and sign the message before it is ready for transmission. These processing stages are distinct and are separated by *processing states*. Figure 6 is a state diagram which shows the message processing stages for both incoming and outgoing messages. There are ten different states which relate to message processing. Some of the states are visited both by incoming messages (those being received) and outgoing messages (those being sent).

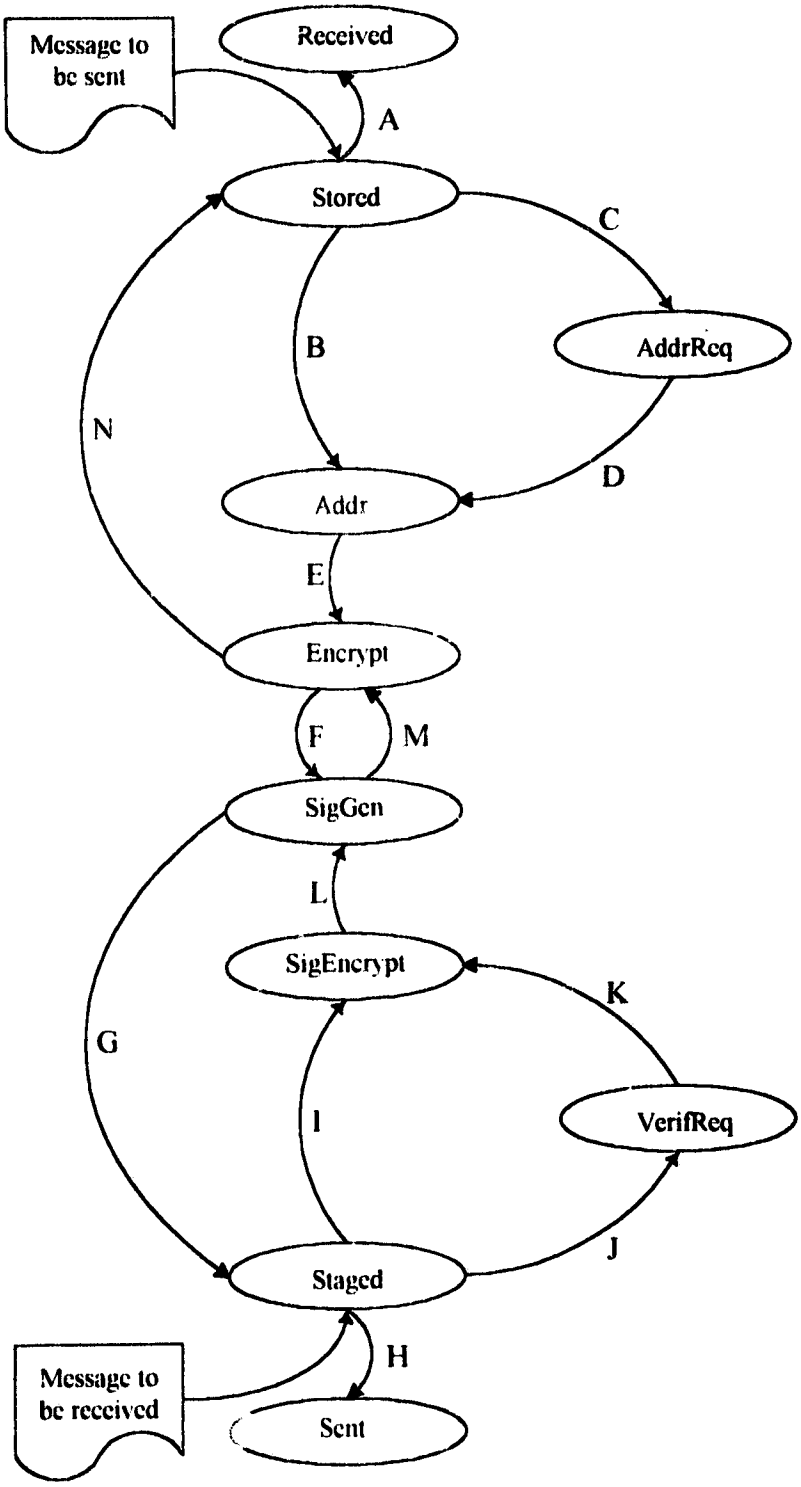
Received An incoming message is transferred from *Health Link* to the application host, the message header is archived and the message is deleted. Similarly, an incoming message reaches this state if the destination user's profile indicates that a manual review of messages is desired and the user reads the message.

Stored For an outgoing message, this is the initial processing state where the message is stored in clear text form in the message base. Subsequent operations encrypt the message and prepare it for delivery to the destination adapter. For an incoming message, this is the state in which the message is stored while awaiting transfer to the application host or while awaiting manual review, depending on the destination user's profile.

AddrReq An outgoing message is held while awaiting destination user and adapter information. This state is reached only if this information is not already on file at the local adapter.

Addr	An outgoing message is successfully addressed.
Encrypt	The message, either outgoing or incoming, is in a compressed and encrypted state. For an outgoing message, the clear text message is compressed using an adaptive Huffman compression routine [87]. For an incoming message, the signature is deciphered and the message-specific data contained in the signature is verified. As the message size of the encrypted message is different from that of its clear text equivalent, the message is stored as two instances in the message base (with two unique message identification numbers). Thus the message ID of the message in the "Encrypt" state differs from its ID in the "Stored" state.
SigGen	An unencrypted signature is generated. For an outgoing message, an RSA [3] private key encryption is applied to message-specific data which can be verified by the destination. For an incoming message, an RSA private key decryption is applied to the encrypted signature on the message.
SigEncrypt	The address of an incoming message is successfully verified: the source adapter's public encryption keys are on file and the destination user exists on the adapter.
VerifReq	An incoming message is held while awaiting a source address verification. This state is reached only if the source address is not on file at the local adapter.
Staged	An outgoing message is held until it can be transmitted. For an incoming message, this is the initial state in a processing sequence where the message is stored in a fully encrypted state. Subsequent operations decrypt the message and prepare it for transfer to the application host.
Sent	An outgoing message is successfully transmitted, the message header is archived and the message is deleted.

Figure 6. Message Processing State Diagram



Tables 5 and 6 show processing times associated with most of the arcs shown on the state diagram for a 1,000, 10,000 and 100,000 character clear text message on both an Intel 8088 and an Intel 30386SX computer. Times for processing along arcs (A) and (H) are dependent on a wide range of factors which include the physical communications medium, the response times of the DCE and the time needed to execute the connect and disconnect procedures.

Interstate wait time, currently set to two seconds, is used to perform other tasks, such as responding to interactive keyboard entries or maintaining communication protocol. Thus, for a 10,000 character message using a 256 character alphabet uniformly distributed, the total message processing time from "Stored" to "Staged" is approximately 14 seconds using an 80386SX25 processor because eight seconds of accumulated interstate wait time are added to the net processing time of six seconds.

Table 5. Message Processing Times (seconds) by Length (characters) for Messages Using an 80 Character Alphabet

	8088 processor Message Length			80386SX25 processor Message Length		
Arc	1,000	10,000	100,000	1,000	10,000	100,000
B	4	4	4	0	0	0
E	9	42	362	1	3	29
F	20	20	20	1	1	1
G	16	16	16	1	1	1
Total	49	84	432	3	5	31
I	5	4	5	0	0	0
L	21	21	21	2	2	2
M	12	12	12	1	1	1
N	10	50	450	1	4	33
Total	48	86	498	4	7	36

Message integrity, authenticity and non-repudiation is ensured through use of signatures generated by the RSA public key crypto-system. Section 4.8 provides a more detailed description of this and other *Health Link* security features.

Table 6. Message Processing Times (seconds) by Length (characters) for Messages Using a 256 Character Alphabet

Arc	8088 processor Message Length 1,000	10,000	100,000	80386SX25 processor Message Length 1,000	10,000	100,000
B	4	4	4	0	0	0
E	9	42	369	1	3	27
F	20	20	20	2	2	1
G	15	15	16	1	1	1
Total	48	81	409	4	6	29
I	5	4	5	0	0	0
L	21	21	21	2	2	2
M	12	12	12	1	1	1
N	11	49	428	1	4	31
Total	49	86	465	4	7	34

4.8 Security

Health Link provides message encryption, authentication, non-repudiation, user authorization and registered delivery. All of these features are kept in force for all messages except for registered delivery which is user-selectable.

Message encryption is achieved in three ways: 1) through direct encryption, 2) through data compression, and 3) through character encoding. Two encryption methods have been implemented: a polyalphabetic substitution cipher [69] and the DES cipher [69, 88]. The polyalphabetic substitution cipher is currently installed because its performance exceeds that of the DES cipher by a ratio of 26:1 (see Table 7). The DES cipher could be implemented in hardware to obtain improved performance but to do so would either require an in-line black box on the communications channel or an on-board DES read only memory (ROM). Either approach would raise the cost of an installation and, in the case of the on-board ROM, make software portability more of an issue.

Table 7. Timing Benchmarks of Security-Related Processes for a Message of 1,000 Characters

Routine	Execution time (seconds) 8088 processor	80386SX25 processor
Polyalphabetic substitution	0.243	0.015
DES encryption	6.376	0.443
Huffman data compression	2.013	0.152
Character encoding	0.161	0.011
Digest calculation	0.288	0.021
CRC-CCITT	0.067	0.005
RSA signature generation	14.121	1.035
RSA key generation	189.965	14.885

The polyalphabetic substitution cipher used is a variation of Porta's cipher [69:501] which depends on character pairs to determine a mapping from a matrix of substitution alphabets. In *Health Link*, the coordinates of the preceding substitution are taken with the value of the character to be enciphered to determine the substitution mapping. The order of the substitution matrix is determined by the cipher key.

Data compression also has a mild enciphering effect. Typically message data is printable. However, as the character data is passed through the compression filter, it becomes a bit stream. Furthermore, an adaptive Huffman compression algorithm [87] starts with an initialized Huffman trie which changes dynamically with each character processed. (Decryption could be further complicated by using the encryption key to initialize the trie.) The *Health Link* implementation uses an adaptive Huffman compression with built-in run length encoding. A fixed initialization of the Huffman trie is used.

Experimentation with arithmetic compression [89] has also been done. The arithmetic and Huffman compression techniques proved to be almost identical in terms of performance and compression ratio. The Huffman compression has the advantage that special bit codes could be used in the output bit stream to denote end of string conditions and run length sequences.

Character encoding is performed to make the final bit stream ready for transmission over the data link. Binary characters, which might be interpreted as flow control characters or data link escapes, are mapped into printable ones. Two methods

have been implemented and compared. The first uses bit stuffing [69] to shift non-printable octets into the printable range. Although the technique is highly efficient for random bit streams (resulting in an increase of approximately 22 percent in length), it performs poorly for bit streams already in near-printable format. A second technique encodes blocks of four characters each. The block is preceded by an encoding control character which indicates the range from which each of the subsequent four characters has been shifted. As well, there is an escape sequence which enables and disables the encoding so that a stream of printable characters is not encoded unless it is efficient to do so. The worst case effect of this encoding process is an approximate 38 percent increase in output length. For random bit streams, it performs better with an increase of approximately 25 percent in string length (see the outcome of one benchmark shown in Table 8). For bit streams containing a random mix of only printable characters, it fairs better with an approximate 4 percent increase in length. The second encoding technique has been incorporated in the current version of *Health Link* because of its low execution overhead and its high efficiency with strings of mostly printable characters.

Table 8. Data Encryption and Character Encoding Efficiency

Routine	Data Type	Actual Efficiency	Optimum Efficiency
Huffman data compression	text	0.618	0.559
	binary	0.668	0.649
Character encoding	text	1.046	1.000
	binary	1.241	1.214

4.8.1 Authentication and Non-Repudiation

The RSA signature is used for message authentication [3, 90, 91]. The RSA public key crypto-system is one example of a set of asymmetric or "trapdoor" encryption algorithms which use one key for encrypting data and an inverse key for decrypting data. The one key can not be easily computed from the other because of the numerical complexity of factoring a very large number into its two constituent primes.

Two sets of keys are used by the RSA algorithm: one pair is generated by the message sender, the other by the receiver. The sender keeps one member of the pair private and makes its inverse publicly available. The receiver does the same with its

pair of keys. The sender generates a signature using its private key which is unavailable to the receiver and which the receiver is unable to calculate. It encrypts the signature using the receiver's public key. The encrypted data can not be decrypted by anyone but the receiver because no one can calculate the receiver's private key. Similarly, because only the sender knows the sender's private key, the receiver is able to authenticate the sender if the receiver can decode (verify) the signature with the sender's public key.

The RSA algorithm [3] uses exponentiation for encryption and decryption. For example, to generate a signature, Equation 1 applies. To encrypt the signature, Equation 2 applies.

$$S_1^e \pmod m = S_2 \qquad \text{Eq. 1}$$

$$S_2^f \pmod n = S_3 \qquad \text{Eq. 2}$$

where S_1 is the signature data,
S_2 is the signature,
S_3 is the encrypted signature,
m is a modulo associated with the sender's keys,
n is a modulo associated with the receiver's keys,
e is the sender's private key, and
f is the receiver's public key.

Similar computations are performed for decryption and signature verification. A modulo is the product of two large prime (or probable prime [92]) numbers, p and q. An encryption key is relatively prime to the product $(p-1)\cdot(q-1)$. The corresponding decryption key is the multiplicative inverse of the encryption key, modulo $(p-1)\cdot(q-1)$.

The signature data, S_1, is encrypted as a block (or single binary number) by choosing a value of the modulo, m, which exceeds the largest possible value of S_1,. Because m is greater than the maximum value of S_1, S_2 may also be greater than S_1. The second stage of the process requires a modulo, n, which exceeds the largest possible value of S_2. Thus, n must be greater than m which must be greater than S_1. This observation impacts the implementation of the algorithm. Each node must

maintain two sets of keys: one for signature generation/verification and one for signature encryption/decryption. For *Health Link*, the modulo for signature generation/verification is 50 decimal digits. The modulo for signature encryption/decryption is 52 decimal digits.

A MAC, which is a hashed digest of the message, is carried within the signature. In other networks, the MAC is often generated using a DES-based algorithm [91, 93] although there is still wide discussion regarding the security of such hash codes [93, 94]. *Health Link* uses, in the interest of simplicity and performance, a hashing function which is based on the CRC-CCITT. Sixty-four bits are hashed into a digest by making four parallel computations of the CRC over the message. Each computation is performed with a different bit rotation applied to the characters in the message. No assertions can be made about the security of the algorithm beyond the predictability of detectable bit errors using the CRC [69]. However, with respect to authentication, these assertions should be unnecessary since the digest and the message encryption key are secured by the RSA public key crypto-system algorithm. If it is assumed that the RSA encryption keys have not been discovered for a particular message and it is assumed that the encryption of the message body has not been broken, then any alteration of the encrypted message text has a high likelihood of being detected with the CRC hashing function.

The message header is authenticated by embedding the message ID, message length and user ID in the signature. After the signature is decrypted, the corresponding, redundant fields in the header are compared to the signature data. If the data elements differ, the message is rejected as being invalid.

Message non-repudiation is achievable by showing that the clear text message could not be altered without changing the hashed digest of that message. The digest contains 64 bits whereas the message can be many times longer. As the size of the digest increases, the degree of its data redundancy increases and its vulnerability to repudiation decreases. Although we can make no assertions about the security of a CRC-based digest with respect to a DES-based digest, we do know that they both suffer from the same fundamental problem of *birthday attacks* [91:19, 93:56]. (A birthday attack refers to where a given message is altered yet causes the original digest to be generated, thereby making the alteration undetectable by comparing the digest with the message data.) From a practical viewpoint, the CRC-based hashing function is far

easier to implement and likely to provide the same level of non-repudiation as do slower, more complex implementations.

4.8.2 User Authorization

All users having access to *Health Link* are registered in at least two ways. The subnet gateway maintains a list of authorized users for the subnet. This list is maintained by a subnet administrator. The user is also registered at a specific "home" adapter by the system manager of the adapter. Access to the network requires that the user log on to an adapter at which the user is registered and be in good standing at the gateway. A user may maintain aliases at other adapters in the same subnet. An alias logon mnemonic may or may not be the same mnemonic as the "home" user mnemonic, depending on naming conflicts at the specific adapter. Passwords are retained at the local adapters.

User registrations are maintained through a technique of automatic expiry and renewal. The gateway is contacted to recertify a user whose registration has expired. If the user's profile at the gateway allows automatic renewal, then the adapter can reactivate the user's registration. A user can be manually removed from a particular adapter by the adapter's system manager. This, however, does not guarantee removal of the user from the subnet.

There are two ways in which user authorization can be breached. If the adapter is configured to pick up messages automatically from a directory on the application host, a user of the application host can mimic an authorized *Health Link* user by generating a message with a valid header under the name of the authorized user. This type of violation can be prevented by specifying appropriate directory protections on the application host.

An unauthorized user can also breach security by learning the password of an authorized user. This applies to the *Health Link* adapter and, potentially, to the application host.

4.8.3 Registered Delivery

Although *Health Link* uses a communications protocol which provides reliable communications from point-to-point, an end-to-end acknowledgment is required to

ensure non-repudiation. A field carried in each message header specifies whether or not a message acknowledgment should be generated by the destination. *Health Link* does not generate the acknowledgment until either the message is transferred from the adapter to the application or the user has reviewed the message manually.

4.8.4 Performance Statistics

Table 7, above, provides the performance statistics for each of the routines used in securing the messages. In cases where a module provides both a procedure and its inverse, the performance of the two have been averaged. For example, the time for character encoding has been averaged with the time for character decoding. Although DES encryption is not currently part of the *Health Link* implementation, timings are given in the table. The RSA signature and RSA key generation statistics shown in the table are independent of message size. Benchmark tests have also provided information about changes in message sizes due to data compression and character encoding. Table 8 shows the results for several cases. The efficiency is calculated by dividing the length of the output string by the length of the input string. The optimum efficiency shown is the efficiency which would have resulted had an optimal routine acted on the same test data.

4.9 Message Scheduling

This section describes the different modes of message delivery, the basic message delivery cycle, variations of the basic delivery cycle and, finally, a summary of the timing parameters used by the delivery cycle. The objective of the section is to 1) provide an overview of the delivery algorithm of *Health Link* and 2) present some of the modeling parameters which are referenced in Chapters 5 and 6.

Health Link adapter and gateway software provides a message store-and-forward service to a multiplicity of nodes in the network. Two independent variables play a role in determining the message schedule at a particular node: 1) the priority of the message and 2) the availability of the designated receiver. Control messages which are used to manage the network are usually given a high priority whereas user messages are typically given a medium or low priority. Accordingly, those messages with high priority are queued for sending ahead of messages with lower priority.

The availability of the designated receiver, (which may be an intermediate node with respect to the message's final destination), is complicated by the following:

- the receiver may be configured to not accept incoming calls,
- the receiver may be unavailable or inoperative,
- the sender's ports may not be available at the desired time, or
- the sender may be inoperative.

Like a postal system, *Health Link* strives to provide an automatic mail delivery service. Unlike the post office, which assumes only a few classes of users, *Health Link* must accommodate a range of individual operating characteristics.

4.9.1 Basic Message Delivery Cycle

In the basic message delivery cycle, messages are queued for delivery according to their time of instantiation and their priority. The term, *time of instantiation*, refers to the time at which a message has been 1) created at an originating node or 2) received by an intermediate node on a route to a final destination. Equation 3 is used to calculate a *trigger* time which determines when the message should be sent to its immediate destination.

$$\tau_t = \tau_i + \delta(p) \qquad \text{Eq. 3}$$

where τ_t is the trigger time,
τ_i is the time of instantiation, and
$\delta(p)$ is the delay associated with a particular priority, p.

The probability that more than one message is being held for delivery to a destination increases with δ. Similarly, the probability that a message addressed to the source is being held by the destination also increases with δ. When a connection is made, all pending messages are delivered, even if their trigger time is in the future. This feature is referred to as *mailbagging*. The practicality of mailbagging is based on the assumption that the time to transfer a message is significantly smaller than the time to make (and break) the communications link. Mailbagging can be subverted by high

priority messages because then the only message holding delay experienced is that introduced by the queue review procedure.

The queue review procedure is performed periodically whenever communications ports are available at a node. The message with the earliest trigger time, which is either current or past, is selected. If no communications link is established for the message's destination, the link connection procedure is initiated and, if the connection is successful, the message is sent. Once a link is established, the queues at both the destination and source are reviewed to locate other pending messages. When no further messages are pending, the link is released.

The queue is a variation of the FCFS queue. Entries are queued and serviced according to their trigger times. However, once a link is established, the queue is viewed as a set of subqueues, one subqueue for each destination. Entries within the subqueue are then serviced according to their trigger times.

4.9.2 Variations of the Basic Delivery Cycle

There are three classes of variation of the basic delivery cycle. The first class relates to the sender's and receiver's configuration. The second class relates to exception behaviour whenever a link cannot be established or fails once it is established. The final class relates to preemptive operations which are intended for local interactive applications requiring real time response.

Each node has a set of physical ports, each of which is configured according to its type of communications medium and its call acceptance. For example, an adapter could be configured with three ports, two for the Public Switched Telephone Network (PSTN) and one for a Packet Switched Network (PSN). Suppose one of the telephone lines is shared and can be used only to originate calls. The other telephone line and the PSN can accept and originate calls. A message destined for an adapter having only a telephone connection causes a connection to be initiated at either of the two ports connected to the telephone system.

Port configurations are known publicly within the subnet. The gateway has a complete set of routings (which are based on port configurations) for all nodes within its subnet. An adapter lacking the route to another adapter within the same subnet queries the gateway which then supplies and certifies the requested information (see Section 4.11.3).

The cases which follow describe different variations of the delivery cycle brought about by configuration differences.

Case 1 - Polling

Consider the adapter which has only a single shared telephone line. The configuration specifies that the adapter can only originate calls, yet, the adapter must be able to receive incoming messages even if it has no messages to send. In order to initiate the connection, the adapter can be configured to periodically construct *polling messages* (fictitious messages which have no data but which can be queued for sending) to be sent to one or more destinations. Once a link is established, all polling messages in the queue are removed except for those with a trigger time in the future. (The polling message uses the Noop operation described in Section 4.6.4.)

Case 2 - Scheduled Delivery

Not only can an adapter be configured for polling, but it can also have pre-established message delivery times. Suppose an adapter is configured to accept and originate calls at its single port. However, it has a large volume of outgoing traffic which is fairly time-insensitive. Schedules can be defined for the destination addresses which determine the trigger times of outgoing messages. As a message is instantiated, the schedule is consulted to determine the next trigger time for a specific destination. The message is held until that time unless an incoming call is accepted from the destination.

Case 3 - Scheduled Delivery with Polling, Both Adapters Accept Calls

A combination of scheduled delivery with polling is possible. The sending adapter can maintain a schedule to determine the holding time for a message and the receiving adapter can maintain a schedule to determine the time and frequency of polling. If a destination adapter fails to poll, a call is initiated and the mail bag is delivered. This mode of delivery is convenient if the resource cost of call connection is too high for the sender. Consider the example where adapter A has a schedule to deliver any waiting messages to adapter B at 4:30 pm and adapter B has a schedule to poll adapter A at 9:30 am and 4:00 pm. If adapter B is able to poll according to schedule, adapter A

initiates a connection only if messages have been created between 4:00 pm and 4:30 pm. Adapter B assumes nearly all of the call initiation overhead.

Case 4 - Scheduled Delivery with Polling, Destination Adapter Refuses Calls

In Case 3, if the destination adapter is configured to refuse incoming calls, the source holds messages according to its schedule or until the destination polls. If the destination fails to poll, then the source sends the mail bag to the gateway.

The gateway has the same polling and delivery features as does any adapter. However, the gateway, because it has a complete set of route specifications for the subnet, may know of less stringent connection conditions for a specific destination than those known by the adapter's peers. Furthermore, if a message is undeliverable because of the destination's configuration, the gateway retains the message and keeps attempting to deliver it.

Variations in the basic delivery cycle can also be caused by problems occurring during message delivery. The receiver may not be ready to accept an incoming call or there may be a communications link failure. Not only is the message rescheduled but so are others which are pending. An abbreviated version of this procedure is shown in Figure 7. For the gateway, rescheduling can occur indefinitely. Polls, unlike actual messages are rescheduled for MaxNumberRetries and are then discarded.

The third class of variation is populated by a single case: interactive requests for message transfer, or, for that matter, any type of operation described in Section 4.6.4. Interactive requests are made in real time and require immediate response. An attempt is made to gain access to a physical port immediately without any queuing. If a port is available and the destination is reachable, the message is transferred immediately. If the port is not available, the destination is unreachable, or a maximum number of operations are already established on an existing link to the designated destination, the message is queued with a trigger time of zero (the earliest possible time). Once the message is queued, the message is treated the same as any other message.

Figure 7. Procedure for Rescheduling Messages After Transmission Failure

```
procedure RescheduleMessages(var Msg: Message;
        RetryWait,MaxNumberRetries: integer);

        var NewTriggerTime, CurrentTime: Time;
        var Dest: Destination;

        begin
                CurrentTime := GetTime();
                Dest := Msg.Dest;                          {save message destination}
                NewTriggerTime := CurrentTime +
                        CalcRetryWait(Msg.NumberRetries,RetryWait);
                repeat
                        Msg.TriggerTime := NewTriggerTime;
                        Msg.NumberRetries := Msg.NumberRetries + 1;
                        if (Msg.NumberRetries > MaxNumberRetries) then begin
                                Msg.TriggerTime := CurrentTime +
                                        CalcRetryWait(0,RetryWait);
                                ReaddressToGateway(Msg)
                        end else
                                Msg.TriggerTime := NewTriggerTime;
                        UpdateQueue(Msg)
                until not (ReadNextMessage(Msg,Dest)) or
                        (Msg.triggerTime >= NewTriggerTime)
        end. {RescheduleMessages}
```

4.9.3 Timing Parameters

Table 9 shows the some of the timing parameters related to Message Scheduling. Unlike the other parameters, the RetryWait and MaxNumberRetries are both configurable parameters. Typically these parameters are set in the range of 120 to 300 seconds and ten to 20 retries, respectively. The research reported in this

document, unless otherwise stipulated, uses 180 seconds and 20 retries, respectively. It should be noted that the function, CalcRetryWait in Figure 7, uses a randomized linear backoff (see Equation 4) for all messages except those directed to the gateway. Equation 4 also applies to messages directed to the gateway, however, n is fixed at 0.

$$\delta = w \cdot [(n+1) \cdot r + 1] \qquad \text{Eq. 4}$$

where δ is the retry time delay in seconds,
w is the configured RetryWait time in seconds,
n is Msg.NumberRetries, and
r is a random real number in [0,1).

Two parameters in Table 9, ScanDir and SchedReview, require some further explanation. ScanDir is a built-in time constant which causes FilDir operations to be queued perpetually for all users configured for automatic message pickup. The table of users is reviewed once every 120 seconds. If a user is found which requires message pickup, a FilDir operation is queued for that user. Then, there is a wait of 120 seconds before the FilDir operation is queued for the next user in the table configured for automatic message pickup. The ScanDir parameter adds an average delay of one minute to the message end-to-end delivery time for those messages requiring automatic pickup.

Table 9. Timing Parameters (seconds) Used in Message Scheduling

Parameter	Description	Value
ScanDir	wait between successive scans for outgoing messages being picked up from directories on the application host system	120
IdlePorts	wait between successive reviews of idle ports which, if one is found, causes a review of the queue for pending messages	13
HiPrioDelay	holding time for high priority messages	0
MedPrioDelay	holding time for medium priority messages	300
LowPrioDelay	holding time for low priority messages	600
SchedReview	wait between successive reviews of schedule tables in order to queue polls	3600

SchedReview is a parameter which determines how often the delivery schedules are reviewed to generate polls to other adapters in the subnet. The algorithm is initiated once hourly and does a complete review of all schedules. If a schedule indicates that a poll is to occur within the hour, an entry is written to the queue. Consequently, this parameter has no direct bearing on the timing of the delivery cycle itself, simply on the hysteresis of the schedule review procedure.

Except for SchedReview, the parameters listed in Table 9 are used in the simulation studies presented in Chapters 5 and 6.

4.10 System Integrity and Auditing

A number of precautions have been taken to make *Health Link* programs resistant to failure. However, both hardware and software failure do occur. Provisions for failure recovery and system auditability have been built in. As well, performance analysis has been implemented for the purposes of benchmark and field testing.

4.10.1 Failure Recovery

The message base is the most critical and dynamic of any data store used by a mail system. As messages can be of any length, they are broken into blocks for storage on disk. Should a failure occur mid-message, integrity of the message base is at risk. The message base is vulnerable even if the failure should occur after a message is stored but file buffers remain unflushed in memory. The message base can also be corrupted by a faulty disk drive or controller. Data bases besides the message base are vulnerable to failure as well but they are usually small and fairly static.

The message base in *Health Link* has two components: 1) a flat file which contains the contents of each message and its header and 2) an indexed file which contains the header of each message with an appropriate set of keys. The indexed file uses a record address to cross-reference the flat file. The flat file is written in 512 byte blocks and closed between write operations. If a failure should occur, the indexed file is rebuilt automatically from the flat file. If a failure should occur while a message is being written to the flat file, the application protocol is interrupted (resulting in retransmission at some future time) and the partial message remains with an invalid

header. When the system is restarted, either the partial message is ignored because of its status or the message is discarded because of a digest error.

As a message is compressed and encrypted (or decrypted and decompressed), its length changes. To protect the message, it is copied as it passes from the Addr to the Encrypt states (or as it passes from the Encrypt state to the Stored state) (see Figure 6). Once the new version of the message is complete, the old version is deleted. All other message processing steps affect only the message header which is updated with a single logical disk access.

Message deletion in the flat file is accomplished by inserting the freed blocks into a free space chain. If a failure occurs while deleting a message, the message base can be reorganized. Reorganizing the message base is an optional procedure which verifies each message's integrity and recovers any lost space resulting from broken free space chains.

The software uses a *crash flag* to determine if a system failure has occurred. The crash flag is stored in the program configuration file and is updated when the program starts and when it finishes normally. If the program ends abnormally, the crash flag remains in the "Running" state, a state which can be detected when the program is next executed. If the program detects a system failure, it rebuilds the message index and the message delivery queue from the flat file. The program also rebuilds the keys to all other data base files to ensure their integrity. If a data base cannot be rebuilt, it is deleted and an initialized version of the data base is created. Once the crash recovery operation is complete, the program begins normal operation.

If all attempts at data base recovery fail, the data bases can be restored from backup. The system is designed to restart regardless of the state of its data bases. This permits the system manager to use his or her menu to restore the data bases from backup. It should be noted that if the program configuration file is corrupt, the software must be reinstalled first.

4.10.2 Audit Trail

The *Health Link* adapter and gateway programs produce an audit trail using three methods. A system audit trail is recorded in a log file, SysLog. A message operations audit trail is kept in the OpsLog file. An archive, MsgArch, of message headers retains information about each message sent and received.

The system log file keeps a record of program startups, program terminations, user logons, user logoffs, program failures, message delivery failures, warnings about lack of disk space, data base backups, data base recoveries and data base reorganizations. The log file can be viewed either by selecting a routine from the program menu or by issuing operating system commands. Whenever the system is restarted, the log file is opened and new entries are appended to the log. This log, which is printable, grows very slowly because only one or two 80-character records are added daily.

The OpsLog keeps track of all message transfer operations which have been completed. When an operation completes, an updated copy of the entry in the queue file is written to the log. The entry includes, among others, the elements listed below:

- operation tag (or ID), type, status and completion code,
- remote address,
- last trigger time and original trigger time,
- operation queuing, start and completion times,
- total number of attempts made to execute the operation,
- total number of application layer SDU's sent and received,
- total length of SDU's sent and received, and
- send object and receive object.

The log is maintained to determine which operations acted on which messages and to analyze network behaviour. The send and receive objects refer to either files or messages, whichever is applicable. Message objects cross-reference the archive, ArchMsg. Records are logged in this file whenever a message is sent or received, either to a remote node or locally. The records are compressed to an average size of approximately 144 bytes each. If a message file is picked up from a local directory and then sent to a remote computer, two records are logged: one for the message pickup and one for the message transfer. In the course of a day, if a physician sends or receives 106 messages, 30 kilobytes of log data are accumulated. This audit control is currently used only for analytic purposes and will be removed once the prototype moves into a production phase.

An archive record supplements a copy of the message header information with the time of message archival. An abbreviated list of the record fields is as follows:

- message ID,
- source and destination mnemonics with their routing addresses,
- priority,
- automatic acknowledgment request flag,
- message category, type and status,
- message subject and application host reference ID,
- time of creation, time of rec[illegible] or time of sending (whichever is applicable),
- time of message expiry (provided by the application),
- version of software which generated the message,
- RSA signature, and
- time of message archival.

Message headers and archive records, for those messages which have been forwarded or which are replies to previous messages, also carry references to their origin. The records stored in this file are 292 bytes each. One record is stored per message sent or received. If a physician sends or receives approximately 106 messages in the course of a day, 30 kilobytes are accumulated in the log per day.

These three files comprise a complete audit trail which indicates when a message is processed, what action is taken and what the outcome of the operation is. It is possible to synchronize the audit trails of more than one adapter to trace a message through the network. The audit key for this type of analysis is the message ID in combination with the source routing address. A précis of the audit trail at each adapter is collected by the gateway to produce accounting information, to troubleshoot subnet problems and to respond to alarm conditions. This function of the gateway is further discussed in Section 4.11.2.

The audit files can be cleared by choosing the *Reorganize Data Base* function on the interactive menu (see Section 4.13 and Figure 8). Furthermore, the archive, MsgArch, can be disabled by changing a parameter in the system configuration (see Section 4.13).

4.10.3 Integrity Issues Relating to Distributed Systems

A subnet possesses some of the same properties exhibited by a distributed system [95]. One issue relates to data base management for routing and user

identification data. *Health Link* maintains partitioned data bases at each adapter with a central master copy at the gateway. Each of the records in the central data base are renewed according to a schedule. When an entry in a local data base expires, the gateway is consulted to obtain a current version. This provides an orderly means to provide gateway-certified updates to each of the partitions. Furthermore, it reduces the load on the network as not all adapters perform updates at the same time; in fact, some adapters may never require an update at all.

A second issue is that of maintaining a distributed clock for the subnet[96]. The gateway provides a clock synchronization service for the adapters within its subnet. These and other subnet control protocols are the subject of Section 4.11.

4.11 Gateway Services

Each subnet is controlled by a gateway providing several services: 1) acting like a message transfer agent (MTA) [97] with other gateways in the network, 2) providing subnet monitoring functions, 3) supplying and certifying user and routing information for the members of the subnet, and 4) synchronizing the clock of each adapter in the subnet. Each of these services is discussed in turn.

4.11.1 Message Transfer

The gateway is capable of transferring messages to gateways in the internet and to adapters in the subnet. An adapter can route messages to the gateway for either purpose in accordance with its local routing table. *Health Link* is designed for point-to-point communication within the subnet (see Figure 2) but this is not an inflexible constraint. For example, an adapter can be supplied with a local routing table which designates the route going to the gateway for all outgoing messages, regardless of actual destination. This might be particularly useful if the adapter has a high volume of widely distributed, outgoing messages which can only be handled effectively using a permanent communications link such as a synchronous leased line. Such an adapter might be located in a laboratory which serves a large population of clinics and physicians.

An adapter routes to the gateway those intersubnet messages which are undeliverable. After the maximum number of attempts has been made, the adapter

redirects a message, unchanged, to the gateway which then assumes the responsibility of forwarding the message to its final destination. The gateway only stores a message until it is delivered. Furthermore, as the gateway is not the designated recipient of the message, it is unable to decrypt the message.

If an adapter in one subnet sends a message to an adapter in another, the message visits two gateways. The gateways are connected through a third-party supplied underlying network. Neither gateway is able to decrypt the message itself. Public encryption keys are accessible to both adapters using a control protocol discussed briefly in Section 4.11.3.

4.11.2 Monitoring

A gateway is used to monitor the adapters within its subnet. Monitoring refers to trapping alarms issued by an adapter and collecting accounting information. An adapter can encounter conditions which should be reported to a human operator. For example, a message could be delivered with an illegal signature, an unauthorized user could repeatedly attempt to logon to an adapter or there could be a critical shortage of storage space. It is probable that no one at the adapter would detect the alarm, even though it is logged, because the operation of the adapter is automatic.

One of the most critical alarms monitored is that of system failure. Since system failure implies immediate shutdown of the adapter, the alarm can only be reported as the adapter restarts. As a matter of course, then, an adapter reports its status to the gateway upon program initialization, providing information about the cause of its shutdown. This has the added benefit of providing the gateway with an established, adapter-initiated communications link which can be used to transfer any undelivered messages stored by the gateway.

4.11.3 Certification of Public Encryption Keys, Routes and Users

Message security (see Section 4.8) relies on the exchange of public encryption keys. Routing information must be changed periodically as the subnet matures. New users must be added and retired users removed. The gateway provides a means of updating this information at each of its subnet adapters. If this information were exchanged among adapters, without using the gateway, there would be no way for the

recipient to certify that the data is correct. An impostor could join the subnet and sabotage it.

As the encryption keys of an adapter α expire, it generates new ones. The public keys are forwarded to the gateway ß for storage and distribution. The private keys are kept secret by the adapter which generated them. Other adapters within the subnet query the gateway for their copy of adapter α's public keys once its current keys expire. The queries and their elicited responses are encrypted and signed to ensure message authentication. The key expiry times are staggered to help prevent traffic surges in the subnet.

Messages sent to remote subnets require special attention because the gateway adapter retains only those public keys for its own subnet. If α is to send a message to a destination σ in a remote subnet, it queries its own gateway, ß, for adapter σ's public encryption keys. Gateway ß processes the query and queries gateway μ, the gateway controlling σ. Gateway μ is able to authenticate gateway ß's query because of ß's signature. Gateway μ replies to gateway ß which, in turn, replies to adapter α. Once the message is received by adapter σ, that adapter must obtain the public signature of adapter α using the same procedure in reverse.

Routing information is retained with the public encryption keys and subject to the same certification procedures. Routing changes, then, are widely installed only when an adapter's public keys are set to expire. The key/route renewal cycle is periodic and automatic unless manual intervention is desired.

As described Section 4.3.2, users also are certified by the gateway. Users, however, present a smaller problem than do encryption keys; remote destinations do not require information about the originating user. Consequently, there is no need to exchange user information among gateways for messages being sent among subnets. To improve subnet performance, user and routing queries and their responses are packaged together.

4.11.4 Clock Synchronization

Health Link is composed of widely distributed, interconnected, autonomous computers. Message security, message auditability and data base maintenance all rely on a synchronized, distributed clock. Even if it were possible to have each adapter and gateway initially set to exactly the same time, the physical clocks on each computer

drift at different rates. (For further discussion on clock synchronization, see [96]) As a result, a weak clock synchronization method has been devised for the subnets. For convenience, until a network-wide clock server is implemented, we assume that the physical clocks of all of the gateways are perfectly synchronized.

To implement clock synchronization in *Health Link*, two factors must be considered: 1) the clocking signal transmission delay and its degree of precision must be known, and 2) the slave clock cannot be set back in time. The first problem is solved by embedding a time stamp in some of the application layer's PDU's. Of all the protocol layers, the application layer is the least standard with respect to the ISO OSI reference model and, accordingly, the time signal has little effect on compatibility between the protocols. The application layer protocol operates in real time, as opposed to control messages, such as the routing control messages described in Section 4.11.3, which must be processed like any other message. The transmission delay of the SDU has been measured at 0.3 seconds over a 2400 bps connection. This delay does not significantly bias the slave's clock which has a granularity of one second.

A second implementation problem is that the slave clock at the adapter cannot be set back in time. If this were done, polling operations might be duplicated and messages might lose their temporal order. A logical clock must be used which is only permitted to advance. The logical clock is weakly synchronized with the adapter's physical clock so that a one second tick on the physical clock causes the logical clock to advance, but, perhaps by less than a second or more than a second depending on its drift. If the logical clock and the physical clock are over 1.25 hours out of synchronization, an alarm is raised. The physical clock is left unchanged because processes other than *Health Link* might rely on it.

4.12 Data Interchange Standards

Health Link is designed to incorporate a high-level data interchange format, to allow multiple heterogeneous applications to join the network with a minimum of customization. Without a standard data interchange format, an application host would be required to interpret the data format of each of the other members of the network. A worst case scenario would see the development of $n(n-1)$ translation routines for a network having n adapters. By assuming a common data standard, the multiplicity of translation routines is reduced to n.

At the outset of the project, the data interchange interface was envisioned to be an integral component of the adapter software itself. A message file would be picked up in the local data format and translated, in accordance with a user-specified dictionary, into a standard data interchange format. Since that time, our orientation has changed to conform to a continuing trend in the market place toward API's. As such, the API is a library of routines which can be called by and linked into application programs to produce a message file in the appropriate data interchange format. Furthermore, an interactive program or scaffold, independent of the application software, can be more easily developed. The program can call the API to permit display and accept entry of transactions when the end-user system has no corresponding application.

The choice of a data interchange standard is not obvious. Several data interchange standards are being developed for the health care field in general. Two of the most prominent in North America are HL7 and IEEE MEDIX. At least two other standards are being developed in Europe: ISO 9735 EDIFACT and EUCLIDES. The standards all consist minimally of a data syntax and a set of object or record definitions. For object-oriented standards such as IEEE MEDIX [98] and EUCLIDES [60], a syntax specification language is used which defines each of the data objects. A transaction template is written in the specification language and provides data descriptors which accompany either one (EUCLIDES) or all of the transactions (MEDIX). For record-oriented standards such as HL7 [99] and EDIFACT [100], the transaction definitions are commonly understood and stored at each of the communicating nodes. The object-oriented standards provide greater extensibility than do record-oriented standards with respect to multi-media applications and transaction composition. HL7, EDIFACT, and other less general standards like ASTM E1238 [11] are intended for transmission of text data only within a fixed format. However, development of object-oriented standards trail behind the record-oriented standards which are themselves incomplete. For example, Harrington, a member of the IEEE P1157 committee, estimates that MEDIX lags behind HL7 by an estimated five years [101].

If the only concern were conformity to the ISO standards, MEDIX is the obvious choice. MEDIX is built on top of the OSI stack and accommodates the ISO 9375 EDIFACT standard, the ISO 8613 Office Document Architecture and

interchange format (ODA), the ISO 8897 Standard Generalized Markup Language (SGML) and the ISO 9595 Common Management Information Service (CMIS) [98]. It makes use of the ISO 8824 ASN.1 to specify its objects. (See [97] for an overview of ASN.1.) However, the very comprehensiveness of the MEDiX standard is a reason for caution. Software tools are needed to provide ASN.1 encoding and decoding of transfer values to the values of the local target system. The processing of data according to Basic Encoding Rules (BER) is CPU-intensive and the messages often carry unnecessary overhead: the Tag, Length and Value (TLV) of each field, rather than simply the value of each field if the format of the data is already known [69:191]. Conformity to one standard often implies conformity with others and possibly all the closely associated standards. With that conformity comes a far more extensive implementation than is needed to solve the basic requirements. Software development complexity and costs rise accordingly. Furthermore, the end product may not be useful for an open system if no other product conforms precisely to the same set of standards [52, 102].

HL7 is already used by a number of hospitals in North America. The University Hospital Consortium, an organization of some 52 institutions, is only accepting tenders from those software vendors providing an HL7 data interchange facility [65]. Installations in Europe are beginning to adopt HL7 as well; at least one hospital in Graz, Austria has chosen HL7 [103].

The *Health Link* application data interchange employs HL7 because of the its widespread acceptance and ease of development. HL7 uses a client-server protocol where the client supplies transactions (HL7 *messages*) to the server and awaits a response. Although HL7 was initially designed for on-line communications, it also permits a batch format for transactions to be transmitted as a file. It is possible within the protocol specifications to ignore the need for an acknowledgment; however the exchange is much more secure if the full protocol is followed. Although *Health Link* software can provide automatic acknowledgment of the batch file itself, it cannot automatically acknowledge the individual transactions in the batch as it is not itself the end-user application program. To provide full acknowledgment capability, the application program must not only send and receive data in HL7 batch files (a facility provided by the *Health Link* API), but it must be able to perform transaction validation and acknowledgment within its processing cycle.

4.13 Interactive Routines

Health Link provides interactive routines for entry and display of text and for performing basic adapter (or gateway) maintenance and monitoring functions. All interactive routines use the same full-screen user interface. The same keys are used universally to navigate from field to field and to edit characters or words within a field. The escape key exits from the screen without retaining any changes. Function key 10 permits immediate update of the screen values without reviewing all items on the screen.

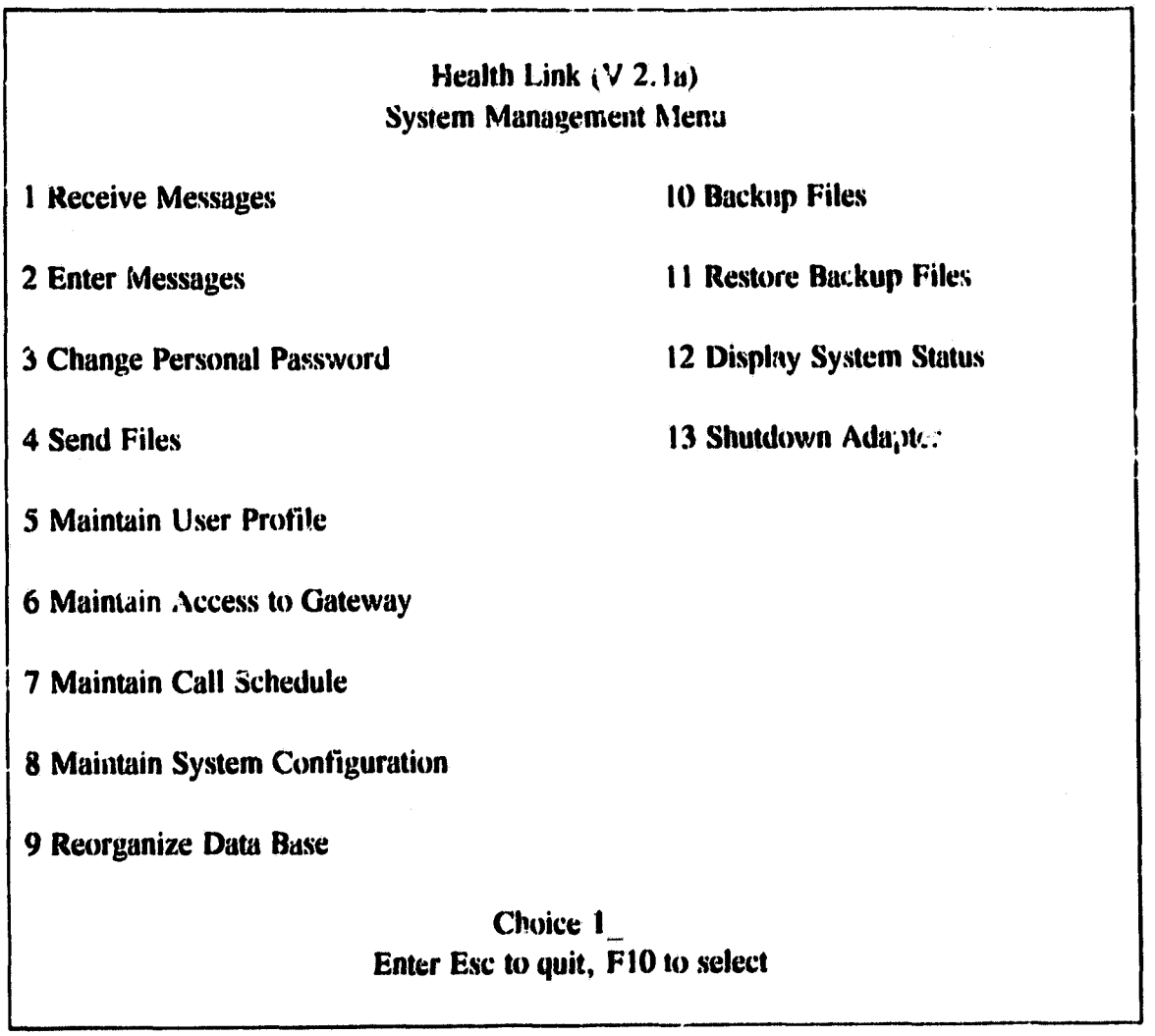
A complete menu for an adapter is shown in Figure 8. A typical user menu has only choices one through four. The menu is rarely used except for system backup and maintaining the user data base. Automatic message pickup and delivery features make it possible for all message-related activities to occur at the application host system without manual intervention.

A brief look at choice eight provides a view of how the program is configured with respect to the network. Two different screens are presented: one which determines global parameters for the node (see Figure 9), the other which determines the parameters for a specific physical port (see Figure 10). The second screen is displayed for each port on the computer.

Discussion of some of the items on the system configuration follows.

- Although it is recommended that message headers be stored in the archive file, MsgArch (see Section 4.10.2), it is possible to disable archiving through the *Archive transactions* parameter.
- The *Activity Timeout* permits the system manager to set a period of keyboard inactivity after which the logon screen is redisplayed.
- An emulation port to an application host can be configured for both the adapter and gateway by specifying the *Port used for emulation* and determining whether or not to *Permit screen emulation*. If screen emulation is turned off but an emulation port is specified, then the screen display and keyboard entry must be done at the adapter only, not at the application host. However, automatic pickup and delivery of message files is still performed at the application host.

Figure 8. Adapter's Menu of Interactive Routines



- The *Line idle wait* (referred to as LineIdle in Section 4.6.6) determines the number of seconds the system waits for characters to be received before deciding that a communications line has been interrupted and issuing a disconnect command sequence.
- The *Number of connection retries* (referred to as MaxNumberRetries in Section 4.9.2) determines the number of attempts that are made to send a message if a communications failure occurs. Each of the retries is attempted according to Equation 4 in Section 4.9.3 where w is the *Time between retries*.

Figure 9. System Configuration Screen Showing Typical Parameter Values

```
                    Maintain System Configuration
          Node mnemonic: his1          Subnet mnemonic: victoria

Name of installation                    Univ of Victoria_____________
Minimum free disk space (bytes)         100000____
Archive transactions (Yes,No)           Y
Auto transfer (No,Send,Receive,Both)    B
Message acknowledgment (No,Yes)         Y
Activity timeout (sec)                  0_
Time between message pickups (sec)      120_
Foreground color                        W
Background color                        B
Port used for emulation (N,1,2)         1
Permit screen emulation (No,Yes)        Y
Line idle wait (sec)                    60_
Number of connection retries            20
Time between retries                    180
                                  Directory Paths
Data          \hl_data_____________________________________________
To Network    \hl_send_____________________________________________
From Network  \hl_recv_____________________________________________
Backup        \hl_back_____________________________________________
              Esc - quit, Ctl-PgDn - next screen, F10 - update
```

- The *Data* directory and *Backup* directories can be specified either for the adapter (or gateway) or for the application system depending on whether or not a port is configured for emulation. The *To network* and *From Network* are default values for directories assigned to individual users which are polled for messages.
- *Auto transfer* and *Message acknowledgment* are also default parameter values.

Figure 10. Port Configuration Screen Showing Typical Parameter Values

Communications table entry 1 of 1

Logical port number (1,2)	2
Connection mode (X.25,dial-Up,Direct)	U
Number of data bits (7,8)	7
Stop bits (1,2)	1
Data parity (No,Odd,Even)	O
Speed (2400,9600,19K,38K,57K,115K)	2400
Flow cntrl (None,Xon-Xoff,Rts-cts)	N
Monitor DCD (Yes,No)	Y
Monitor DSR (Yes,No)	Y
Originate only (Yes,No)	N
Listen script	ATS0=001\r\wK
Connection script	ATS0=000\r\wKATDT
Response to connection script	\r\eY\w
Disconnection script	ATH\r\wK
Reset script	ATZ\r\wKATE0V1&D2\r\wK
Attention (DLE) script	\r\p+++\p3ATE0V1\r\wK

Esc - quit, Ctl-PgUp - prev screen, Ctl-PgDn - next screen
F3 - add port, F9 - delete port, F10 - update

The port configuration screen permits entry of the basic serial port parameters and the connection, disconnection and reset scripts. The first portion of the screen is self-explanatory. The scripts, however, require some further explanation. The scripts shown in Figure 10 are strings of commands conforming to the Hayes command set interspersed with supplementary commands. It is not necessary that the DCE conform to the Hayes command set since the supplementary commands are written without any programmatic interpretation to the physical port at a rate of seven characters per second. The supplementary commands which begin with a backslash ("\") are

programmatically interpreted. These are used to introduce non-printable characters, to pause for a certain period of time, to wait for a specific character in response from the DCE, and to branch out of the script if an illegal response from the DCE is received.

Some of the scripts are used in combination with one another. For example, the *Connection script* is concatenated with the script contained in the destination's routing record (not shown here) and then concatenated with the *Response to connection* script. Thus, the *Connection script*, itself, sets up the connection parameters for its own DCE. The script in the routing record determines the path to the destination, for example, the telephone number. The *Response to connection script* determines the outcome of the connect request.

This routine is the most complex of all the interactive routines. It is usually accessed only by the system manager when the system is installed. All other routines, which are far simpler to understand, use the same user interface. It can be seen that the interface is easily mastered and, for most users, very focused.

4.14 Software Testing and Validation

Section 4.5 describes the software structure of the prototype which is highly modular. This structure permits many of the modules to be tested independently by means of test scaffolds. In some cases dependencies between modules are eliminated through the use of *stubs* which present an inert interface to the module under test. Interactive test scaffolds have been used extensively with all modules. Modules in the lower layers of the structure have batch test scaffolds which subject the module to a battery of pre-determined exercises.

The prototype has been subjected to system tests to validate its behaviour in a network. Both interactive and batch tests have been conducted. In both cases, a standard set of trace statements and logs have been used to record the behaviour of the nodes under test.

Batch tests have been performed under a variety of loading conditions inside the laboratory using a Private Branch Exchange (PBX) and *in situ*, at the medical laboratory and clinics participating in the field test described in Chapter 7 using the actual telephone system. Typically these tests are run for a day at a time over a range of message frequencies and message sizes. Not only have these tests proven to be

valuable for testing but they have provided calibration data for the simulation studies described in Chapters 5 and 6. The batch tests have proven to be critical in ferreting out program failures resulting from interactions in the multi-threaded processing of messages. The high frequency batch tests, or *stress tests*, have highlighted problems with real-time modem control and with processing bottlenecks relating to task scheduling and file management. Stress tests were run over a period of three months prior to the operation of the prototype network described in Chapter 7.

4.15 Summary

This chapter provides a description of *Health Link* and many of its features. *Health Link* is a network consisting of a collection of subnets. Each subnet has a set of nodes or adapters, each of which provides an interface to an application system and a gateway which provides an interface to the internet.

Health Link software is composed of three programs: application host terminal emulator/file copier, adapter and gateway. The software uses the family of IBM-compatible MS-DOS PC's as its main platform. The terminal emulator/file copier program has been ported to the DEC VAX and the IBM RISC System/6000 for the purposes of the field test.

The software is highly modularized in an effort to reduce testing time and to isolate hardware and operating system behaviour. Special care has been taken to make each of the three programs a subset of a single core of software in order to ease maintenance costs and to ensure compatibility. Program structure has been designed around the need to simulate a multi-tasking operating system. Even with an operating system like MS-DOS, *Health Link* can provide adequate interactive services locally while maintaining one or more active communications links.

Health Link follows the ISO OSI reference model. The network architecture has four distinct layers which correspond to the physical layer, data link layer, transport layer and the application layer. This permits other OSI compliant modules to replace the *Health Link* modules with a minimum perturbation. This architecture also enables the *Health Link* protocol to be robust and to provide such features as full-duplex communication and multiplexed transport connections.

Message processing is subdivided into a number of procedures which ultimately provide a guarantee of message integrity, authentication, non-repudiation and

confidentiality. Although message processing is CPU-intensive, the total CPU resource consumed on a 80386SX25 processor is approximately four seconds for a 1,000 character message. When compared to the 13-second message transfer time, the processing time is not significant.

A number of timing factors are associated with different message delivery schemes. The basic delivery cycle queues messages for sending according to their priorities. The longer the message is held for sending, the more likely it will be sent with other messages in the same mail bag. It is possible to specify explicit delivery and pickup times. The gateway stores messages which adapters are unable to deliver.

The software is specifically designed to protect the message base and to generate an audit trail. Critical system conditions and message-related operations are logged. Message headers can be retained in an archive to provide a record of transmission and receipt.

The gateway acts as a communications portal to other subnets and as a subnet administration node. It stores a master data base for all public encryption keys, subnet routes and subnet users. Adapters within the subnet report alarm conditions to the gateway automatically so that problems can be monitored and corrected.

The HL7 standards for data interchange are a component in the *Health Link* network. HL7 encoding and decoding routines are available through an API. To provide HL7 services for those end-users lacking certain applications, a stand-alone interactive display and entry program can be written which calls the API to create and interpret HL7 messages.

The software is designed to provide fully automatic services to end-user application systems. However, it is possible for users to access the network through interactive routines. For example, free-text messages can be entered and sent. The interactive user interface is simple to use.

The design considerations of *Health Link* have been met by providing a low cost, automatic and secure network facility for heterogeneous health care computers.

5 Basic Steady-State Simulation Models

5.1 Introduction

Successful installation of a *Health Link* network is dependent on both the software implementation and the topology of the network. Each subnet should be configured to provide the appropriate level of response. Similarly, the collection of subnets should be configured to minimize operating costs. Although experimentation with different topologies could and would be done with a live network, such adjustments are difficult, inconvenient and time-consuming to make. It is important to consider alternative methods of predicting network behaviour without making actual changes to the configuration of a live network. Simulation studies are such a means to predict behaviour [104, 105, 106]. This chapter examines the steady state behaviour of a cluster of nodes which characterizes a subnet. Although steady state simulation can determine saturation thresholds and highlight areas where the adapter software might be optimized, it cannot predict the behaviour of a large scale network composed of a number of subnets operating under dynamically changing loads. An analysis of a large network is described in Chapter 6.

In this chapter, we examine two basic topologies for a subnet: 1) a fully-connected mesh and 2) a star. These are not the only topologies available to a telephone-based network, but they are representative. The mesh makes use of the distributed routing capabilities present in *Health Link*. Communications within the subnet are all single *hop*, i.e., messages can be passed directly from source to destination by establishing only a single connection. The model is based on the assumption that any node (adapter) in the subnet can initiate a call. In this model, there are no communications with a central hub or gateway.

The star model uses a central hub as a store-and-forward repository. The hub only accepts connection requests from the nodes in the subnet; it does not initiate any connection requests. In this configuration, each node uses only one route: that to the central hub. The hub stores messages for each destination until the destination node establishes a connection. Consequently, this model exhibits a message latency which is inversely proportional to the frequency of connection requests. Two hops are required to transmit a message from its source to its destination.

The mesh topology has been chosen for study because it has the greatest potential for minimizing the message transfer time and resource utilization. The star topology has been chosen because it offers centralized control over message transfer and it is used by a number of already-established networks (for example, [66, 67, 107]).

The models have been developed to describe the behaviour of node *clusters*. A cluster is that set of nodes which form connections with a specific node, including that node itself. Thus, a mesh may consist of one or more node clusters which may or may not overlap. The star cluster of interest is the subnet itself because all nodes in the subnet form connections with the hub.

Simulation studies are intended to predict the performance of a real system. The validity of the simulation model is critical to the credibility of the simulated output. Two types of modeling, discrete event and analytic, are used in this analysis to strengthen the credibility of the results. Both models have high degree of face validity although the discrete event model makes fewer assumptions about the real system. The models have been calibrated against actual or *real* performance measurements taken from a battery of tests conducted with a *Health Link* cluster of four nodes at a subset of the field test sites described in Chapter 7.

Discrete event modeling maps real processing elements into mathematical blocks (or rules) which operate on input events or transactions to produce output events or transactions [104, 106]. The characteristic behaviour of a block is designed to be similar to the real processing element which it represents. The real system is modeled by a system of one or more of these blocks. Inputs to a discrete model are typically either random observations from a probability distribution or a playback from known inputs. Each input event is treated on as a discrete transaction. Accordingly, the statistics gathered from the model are representative of a sampling experiment performed on the real system.

Analytic modeling makes use of statistical relationships to predict the behaviour of a real system [105, 106, 108, 109]. For example, suppose that transactions arrive with an exponentially distributed interarrival time and are queued first-come, first-served (FCFS) at a facility with an exponentially distributed service time. A number of statistical measures can be computed such as the expected queue length and the expected waiting time in the system. Analytic modeling is limited to simple, easily

described systems. For more complex systems, bounds analysis and approximations can be used to predict best- and worst-case performance using the same techniques [105, 110].

This chapter presents discrete-event and analytical models with their simulation results for both the mesh and star topologies operating at a steady state with exponentially distributed message interarrival times and exponentially distributed service times. This type of distribution has been used for several reasons:

1) Real performance data, used to calibrate these models, has been collected from a cluster of four nodes each of which is driven by a memoryless message generator, i.e., it generates messages whose interarrival times are exponentially distributed.

2) Analytic queueing models are markedly easier to formulate for exponentially distributed queueing and service times than they are for general queueing and general service times which are the best suited to the actual operation of a network.

3) Establishing the parameters which would validly describe the actual message arrival and the service time distributions would require intimate knowledge of an actual network, thus obviating the need for exploratory simulations such as those described in this chapter.

The next section focuses on the discrete-event models for the two topologies. It examines the validity of the models and provides an analysis of standard error. The third section provides a discussion of the analytic models used. The final section presents 1) conclusions about the different types of models used in the simulation and 2) a comparative analysis of their results. Appendices A, B, C and D contain computer listings of the simulation programs.

5.2 Discrete-Event Models

The discrete-event models developed for analyzing the steady-state behaviour of the mesh and star topologies are similar in structure. Both models have a node (or adapter) which operates in apposition to a multi-nodal driver. The adapter and the driver are assumed to exhibit the same behaviour. The models are structured analogously to an object (the node) placed in front of a multi-faceted mirror (the driver), as in the case of a person standing in front of a triple mirror in a garment

store. This structure permits the number of nodes in the cluster to be controlled parametrically. The main difference between the mesh and star models is that the latter uses a central hub or gateway in which to store those messages in transit. Abbreviated diagrams which show the message flow are shown in Figures 11 and 12. A legend in Figure 13 accompanies the diagrams.

5.2.1 Implementation

Several aspects of Figures 11 and 12 require further explanation. The diagrams are drawn so that the adapter (or focal node) is at the left and the driver is at the right. The adapter is composed of three sections: 1) the process which picks up the message from a file and prepares it for transmission by encrypting it and by generating its signature; 2) the process which stores the message for transmission, makes the connection and transports the message; and 3) the process which delivers the received message by preparing it for viewing and then deposits the message in a file. The average time delay measured in the first section of the adapter is used directly by the driver as its pickup delay.

Messages are stored at both the adapter and the driver in *user chains* [111, 112] or user-defined queues employing a customized queueing discipline that is described in detail in Figure 7. It is sufficient here to visualize the messages at the adapter being chained together according to their destination ordered by their assigned trigger times. Whenever a message's trigger time is reached, the adapter attempts to connect with the designated destination. If the connection attempt fails, the trigger time of that message and other messages destined to the same node are recomputed and the message is reinserted in the queue. New messages can jump the queue if their trigger time is less than the trigger times of existing messages.

Switches Sw2A and Sw2D remove messages from a chain when their trigger time is reached and reinsert messages in the chain if a connection attempt fails Switches Sw2A and Sw2D are shown as being linked logically. If a connection is established, messages are exchanged in either direction. Thus, if the adapter establishes a connection to a specific destination, Sw2A is set to the destination and Sw2D is set to the source. If the driver establishes a connection from a specific source, Sw2D is set to the source and Sw2A is set to the destination.

Figure 11. Schematic of Discrete-Event Simulation Model for a Mesh Cluster

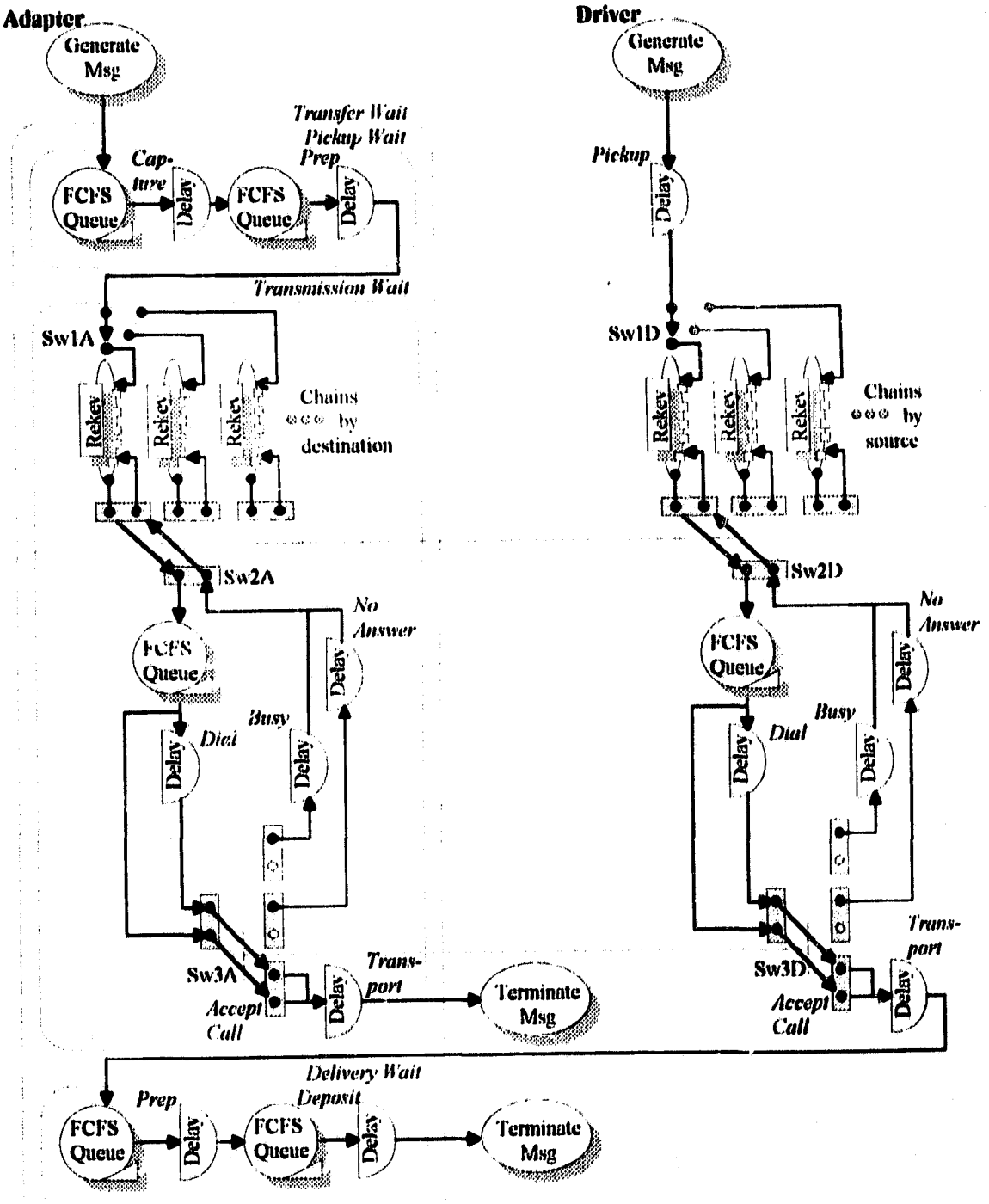


Figure 12. Schematic of Discrete-Event Simulation Model for a Star Cluster

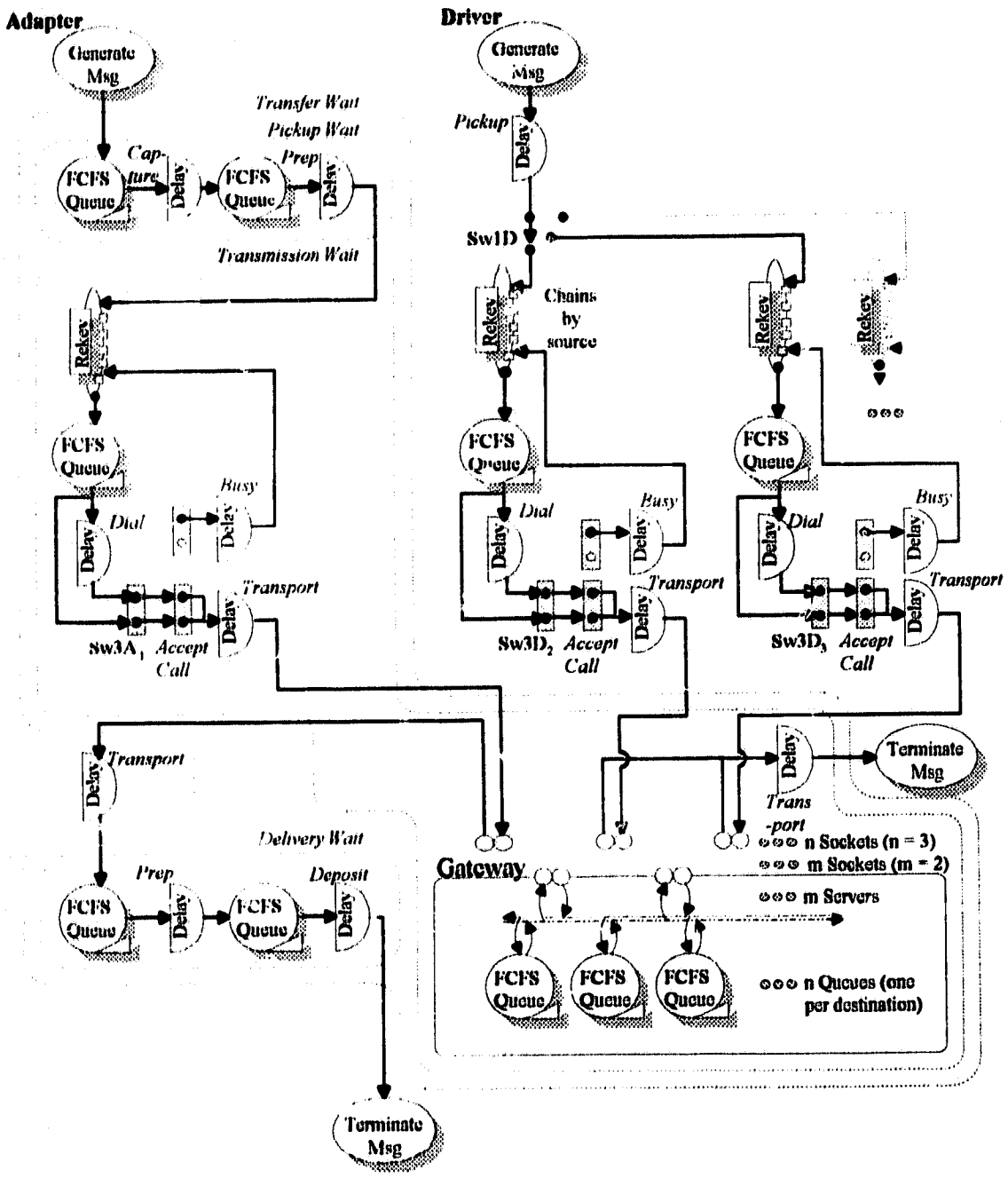
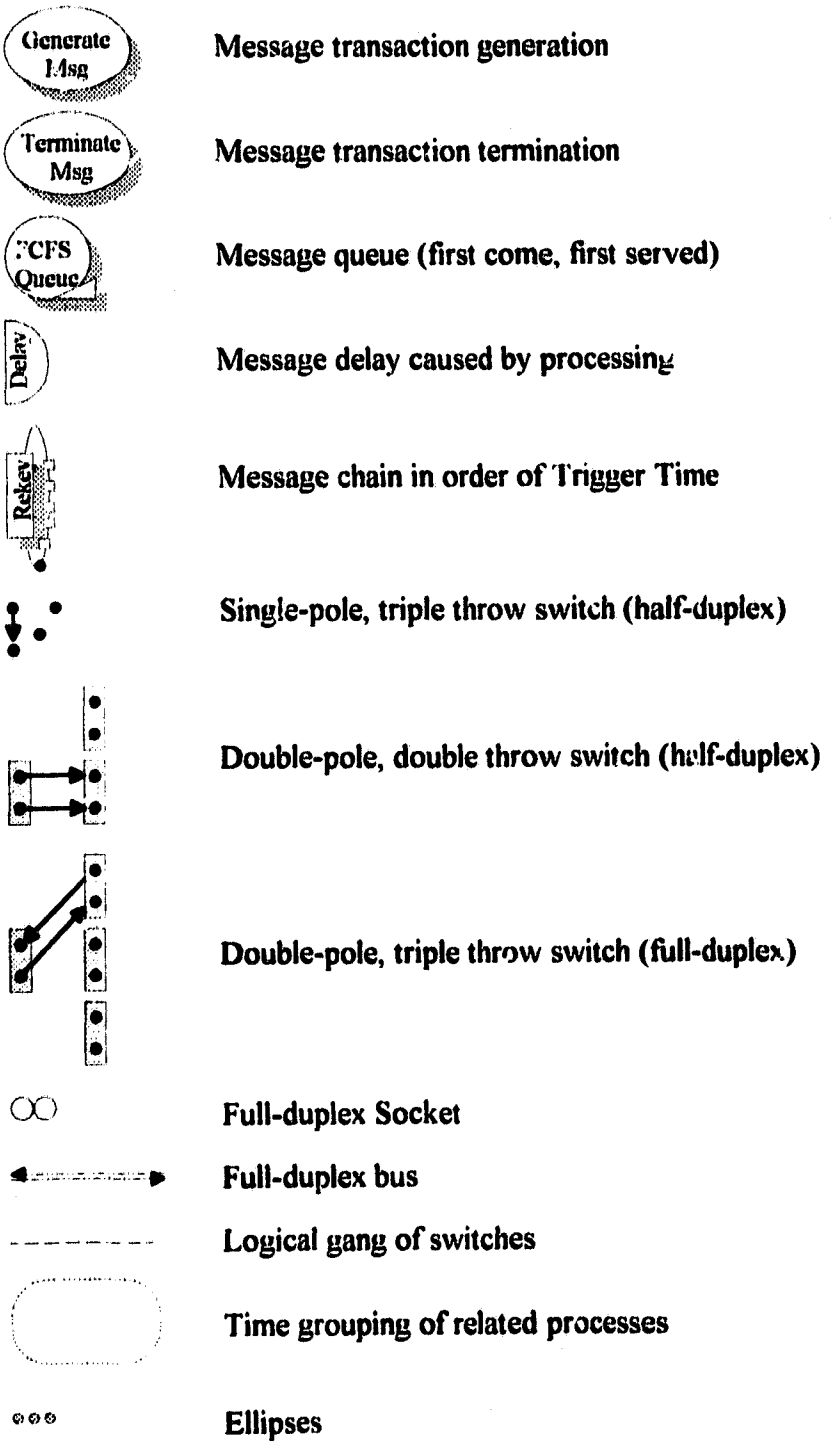


Figure 13. Legend for Discrete-Event Simulation Models



Switch Sw3A is used to direct a message through an established connection or to divert a message back to the chain if the connection attempt is unsuccessful. It shows two poles: one that is used by the message initiating a connection attempt; one that is used by subsequent messages which bypass the dialing delay once a connection is established. Switch Sw3A is a probabilistic switch, i.e., its switch setting is determined randomly from average accessibility ratios measured by the driver while it attempts to establish connections with the adapter. This avoids the need to maintain a complete set of states for every node represented in the model. Switch Sw3D is non-probabilistic: it can determine directly the state of the adapter whenever a connection attempt is made. Like switches Sw2A and Sw2D, switches Sw3A and Sw3D are linked logically.

Except for three differences, the adapter shown in Figure 12 is similar to that shown in Figure 11. The first difference is the existence of only a single user-defined queue. As all messages must be sent to the gateway, only one queue is needed. The second difference is the removal of the "no answer" option on a connection request. The adapter should only rarely encounter a "no answer" condition at the gateway since the gateway does not place its modems in a "no answer" mode (as it would if it originated calls). The third difference is the introduction of a transport delay in the delivery process. This delay is incurred by the second hop in the message transfer.

The driver shown in Figure 12 has two differences from the driver shown in Figure 11. The first difference is the replication of the dialing subsystem in conjunction with the user-defined queues. This is because each node represented by the driver must be able to access the gateway independently. In the mesh where all communication is point-to-point, communication between nodes other than the focal node is irrelevant. In the star, communication with the gateway has an impact on the gateway utilization. The second difference is the removal of the "no answer" option on a connection request (see the explanation above).

Figure 12 shows a simple gateway or hub. The diagram indicates that the messages are stored in queues by destination. The gateway performs no other function and, accordingly, holds the messages until the destination makes a connection request.

The discrete-event models are written in GPSS/H 386 [111, 112] (see Appendices A and B). Two types of transactions are generated by the models: 1) messages and 2) clock ticks having a granularity of one second. The clock is needed

to control the review cycle of the user-defined queues storing the message transactions. The models read a parameter file which controls program execution. A listing file and a summary output file are produced which contain results of the simulation run.

5.2.2 Assumptions

Although the discrete-event models closely follow the logic of the *Health Link* software, there are some assumptions on which the models have been based that are introduced either by approximations to the software functionality or by the sterile operating environment of the model. It is assumed that:

1) the communication links are as noise-free as those used in the live calibration runs,
2) the CPU load resulting from servicing communication interrupts has no significant impact on CPU availability,
3) any communication activity, including the execution of most modem setup scripts, inhibits the preparation of messages being sent or received,
4) the message preparation time for messages being received (relating to decompression, decryption and signature verification) is the same as that for messages being sent (relating to compression, encryption and signature generation) and is independent of message length,
5) at any single node, messages are prepared sequentially,
6) any delays incurred by searching queues, reorganizing data bases, reordering lists and performing directory searches is negligible,
7) only three outcomes can result from a connection request in a mesh cluster: call accept, no answer and destination busy,
8) only two outcomes can result from a connection request in a star cluster: call accept and destination busy,
9) all nodes in the model, both the adapter and the multi-nodal driver, behave uniformly the same,
10) message generation for any individual node is uncorrelated with any other node, and

11) message interarrival times are exponentially distributed at any individual node.

5.2.3 Results

Discrete-event simulations of Intel 80386SX25 platforms have been run for several cases. Table 10 lists the cases and indicates how the runtime parameters differ. Figures corresponding to the cases are referenced by the table. These figures show median elapsed end-to-end transfer time in seconds versus message frequency. End-to-end transfer time is the time taken to read a message from a file, prepare it for sending, send the message, prepare the message for delivery and to write the message to a file. The median is used to measure central tendency because it is less sensitive to outliers than is the mean. The message frequency refers to the frequency at which messages are created at any particular node.

Table 10. Runtime Parameters Used in Discrete-Event Simulations

Topology	Number of Messages	Number of Servers	Priority Delay (sec)	Message Acknow-ledgments	Figure
mesh	circa 3000	N/A	600	no	14
mesh	circa 3000	N/A	300	no	15
mesh	circa 3000	N/A	300	yes	16
star	circa 1000	1	300	no	17
star	circa 1000	2	300	no	18
star	circa 1000	10	300	no	19
star	circa 1000	10	0	no	20
star	circa 1000	10	0	yes	21

Figures 14 through 16 are produced by collecting data from 6,000 message transactions of 1,000 characters each. Since message transactions are generated at both the adapter and the driver the effective number of messages fully processed for both sending and receiving is approximately 3,000. Similarly, Figures 17 through 21 are produced by collecting data from 2,000 *counted* message transactions of 1,000 characters each. The only message transactions which are counted are those which are either sent or received by the adapter. Thus, the effective number of messages fully

processed for both sending and receiving is approximately 1,000. As the Median Elapsed Message Transfer Time, shown on Figures 14 through 21, is a sum of the median pickup, transmission and delivery times; the distribution of messages sent and messages received count as a single additive distribution. Figures 14 through 21 are based on the assumption that the different stages of message pickup, transmission and delivery can be treated as independent events. Figure 22 shows the probability densities (based on time intervals of 50 seconds) of these three components in the end-to-end transfer time for a mesh with cluster size of four operating at 24 messages per hour with a priority delay of 300 seconds.

Although not discussed in Section 5.2.1, acknowledgment messages can be generated by the models. The acknowledgment messages, which consist of only header data, are always sent low priority with a priority delay of 600 seconds.

Inspection of the results for mesh clusters highlights the following:

1) Changing the priority delay from low to medium not only changes the ordinate but also changes the spread of the characteristic curves (see Figures 14 and 15). At an elapsed message transfer time of 2,000 seconds, the low priority nine-node cluster can handle a throughput of 23 messages per hour whereas the medium priority cluster can handle slightly more than 19 messages per hour. This is due to a greater number of low priority messages being collected in the mail bag.

2) Although small clusters perform significantly better than large clusters, there appears to be a worst-case throughput threshold at approximately ten messages per hour. If acknowledgment messages are used, this threshold is approximately halved (see Figures 15 and 16).

3) Use of acknowledgment messages reduces the throughput of a cluster by about 40 percent. This result is not unexpected as the number of connection requests must nearly double if the rate of message arrival is low. The throughput could be improved by significantly increasing the priority delay of the acknowledgments.

Figure 14. Low Priority Message Transfer Times for Discrete-Event Simulation of a Mesh Cluster

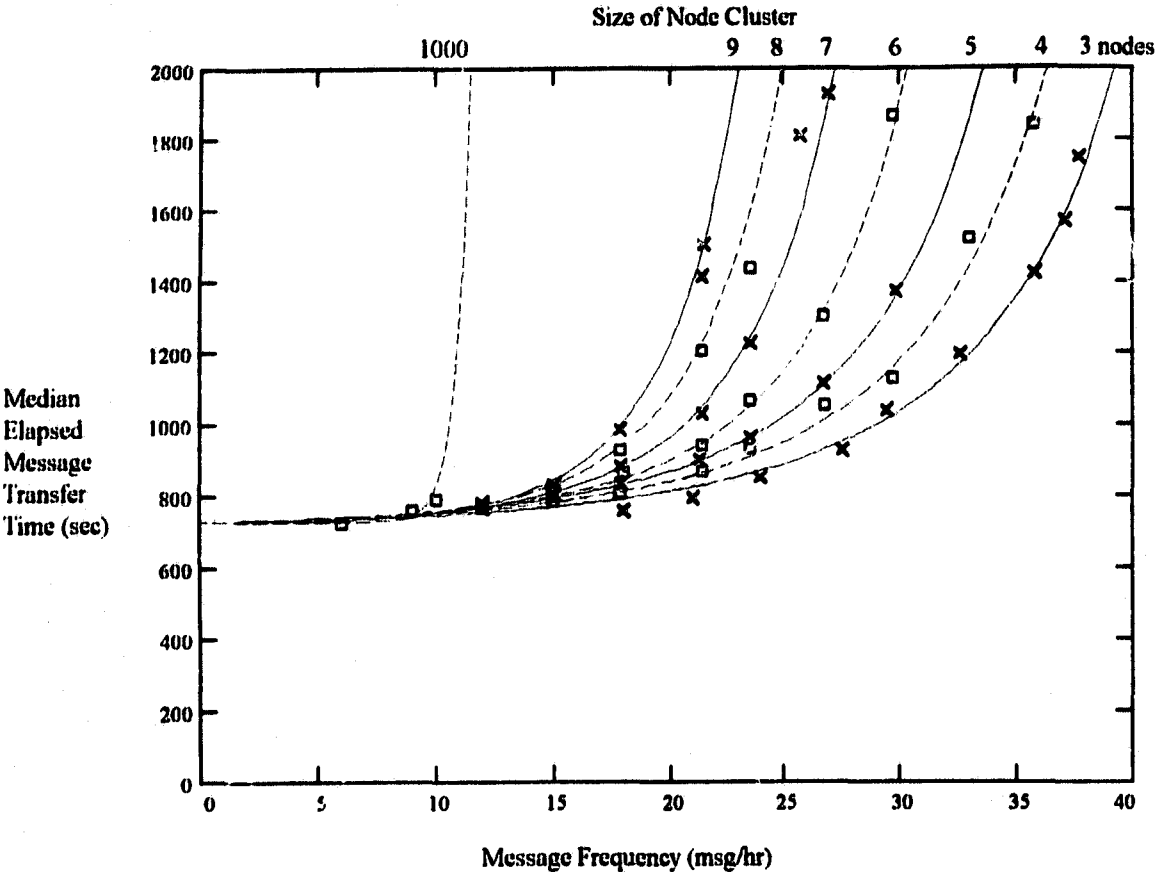


Figure 15. Medium Priority Message Transfer Times for Discrete-Event Simulation of a Mesh Cluster

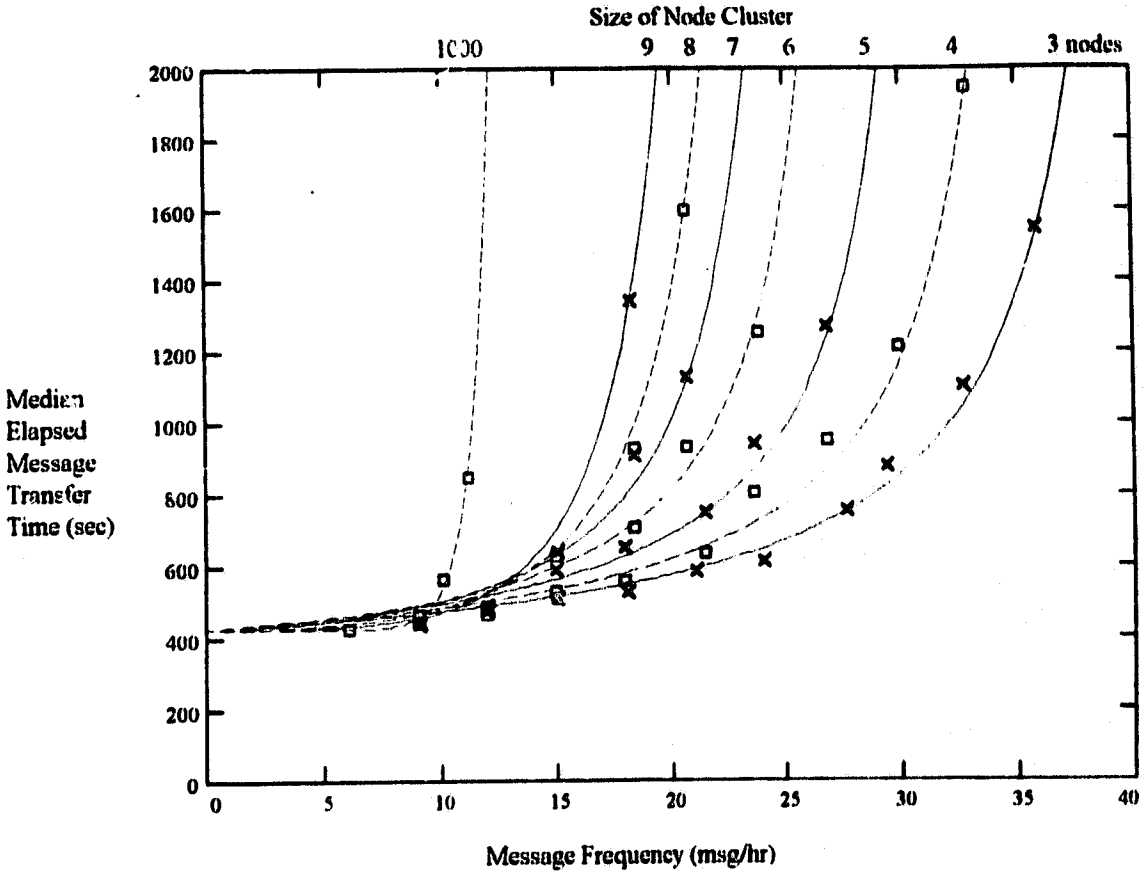


Figure 16. Medium Priority Message Transfer Times for Discrete-Event Simulation of a Mesh Cluster Employing Message Acknowledgments

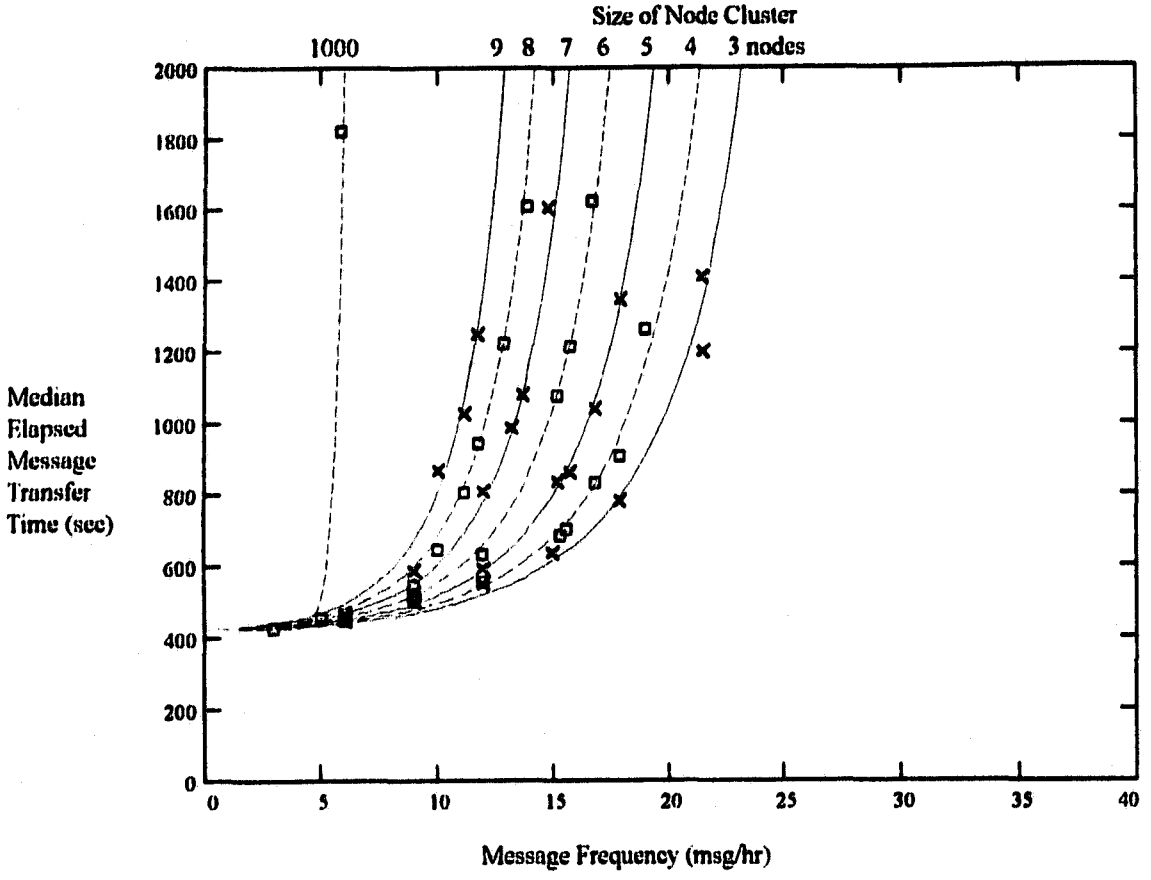


Figure 17. Medium Priority Message Transfer Times for Discrete-Event Simulation of a Single-Server Star Cluster

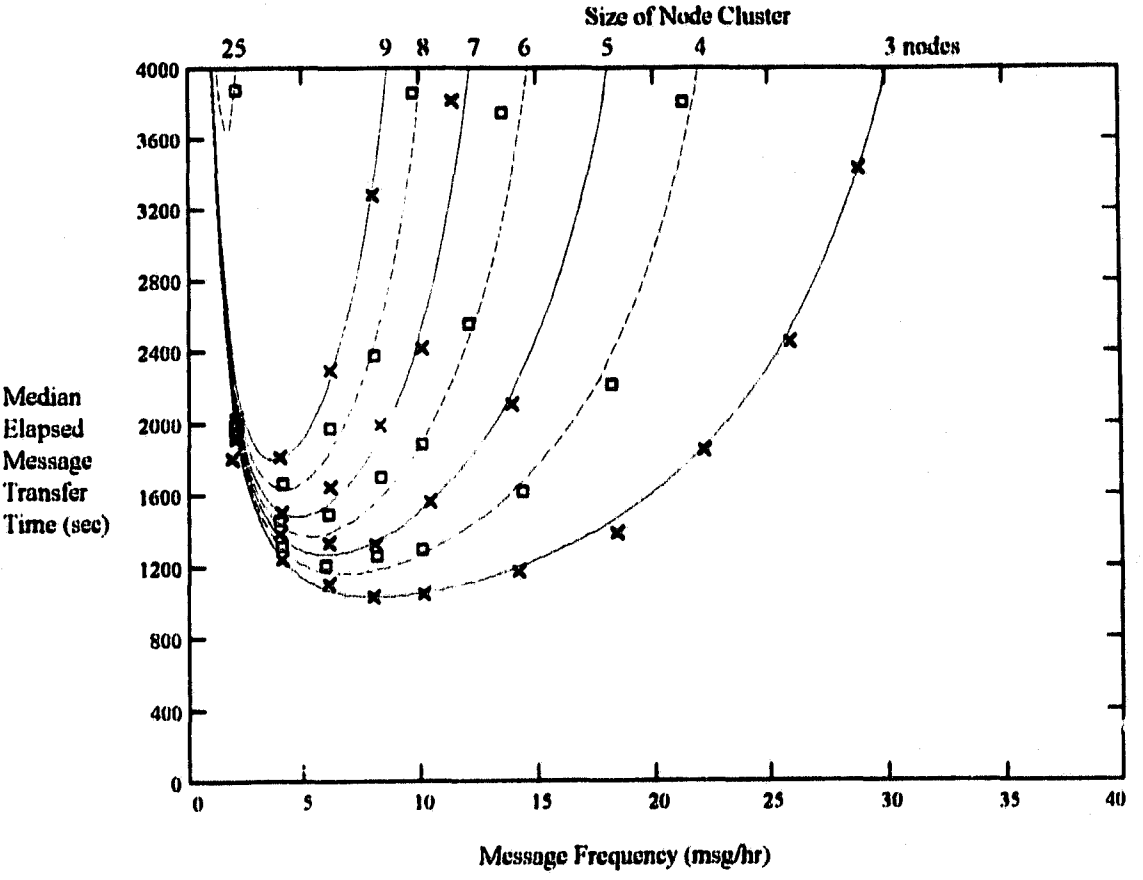


Figure 18. Medium Priority Message Transfer Times for Discrete-Event Simulation of a Two-Server Star Cluster

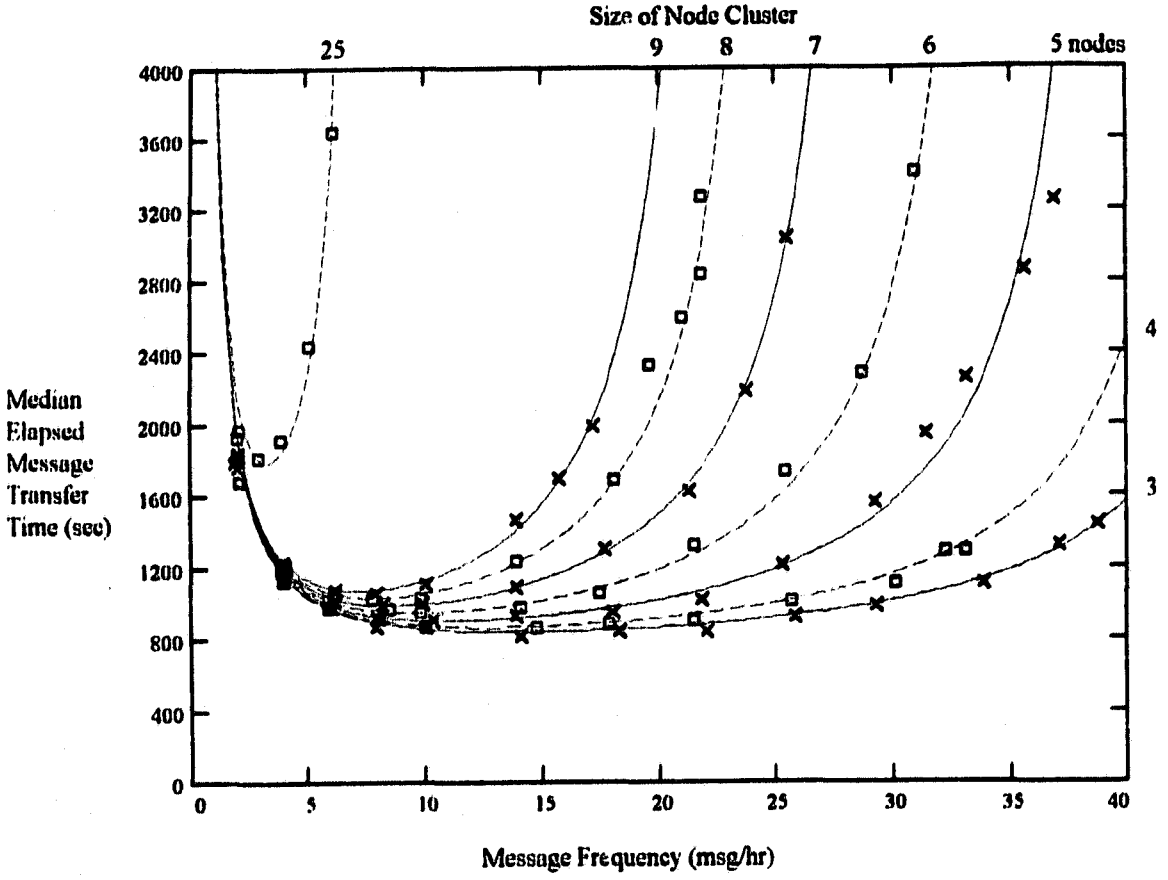


Figure 19. Medium Priority Message Transfer Times for Discrete-Event Simulation of a Ten-Server Star Cluster

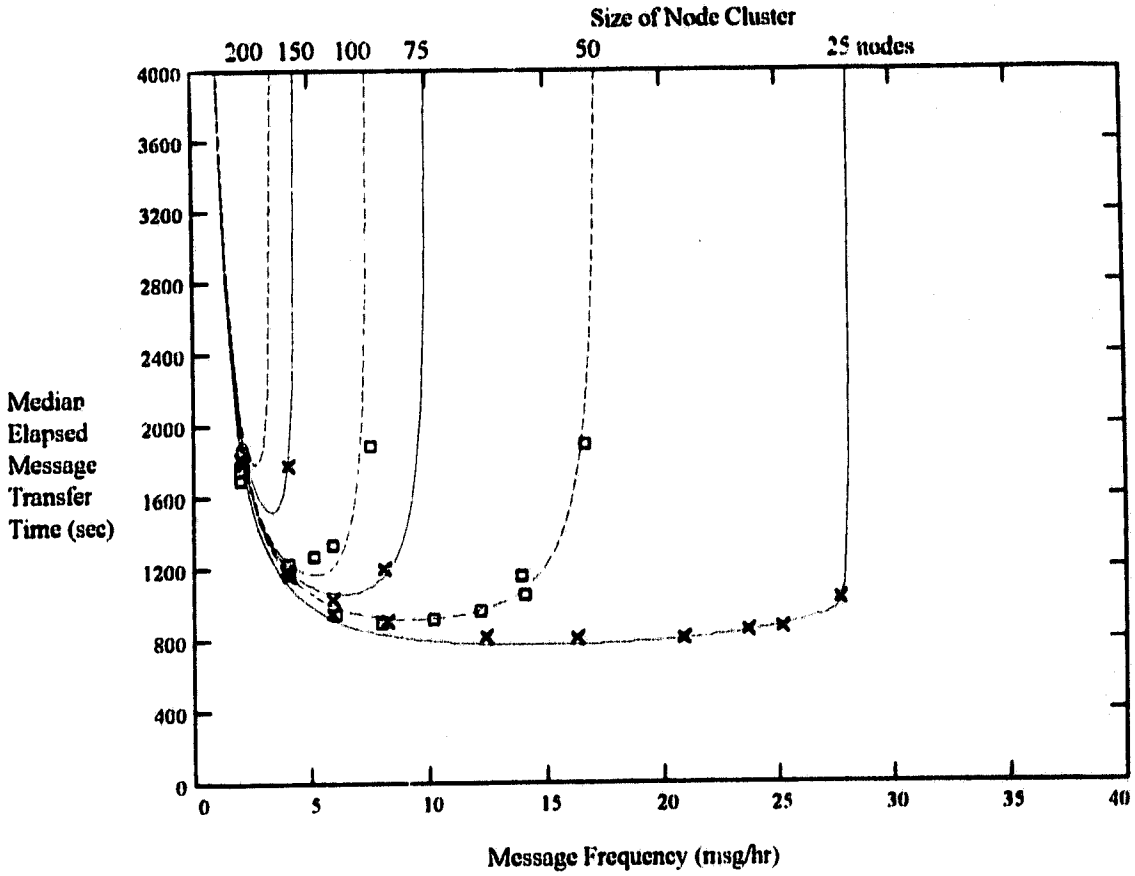


Figure 20. High Priority Message Transfer Times for Discrete-Event Simulation of a Ten-Server Star Cluster

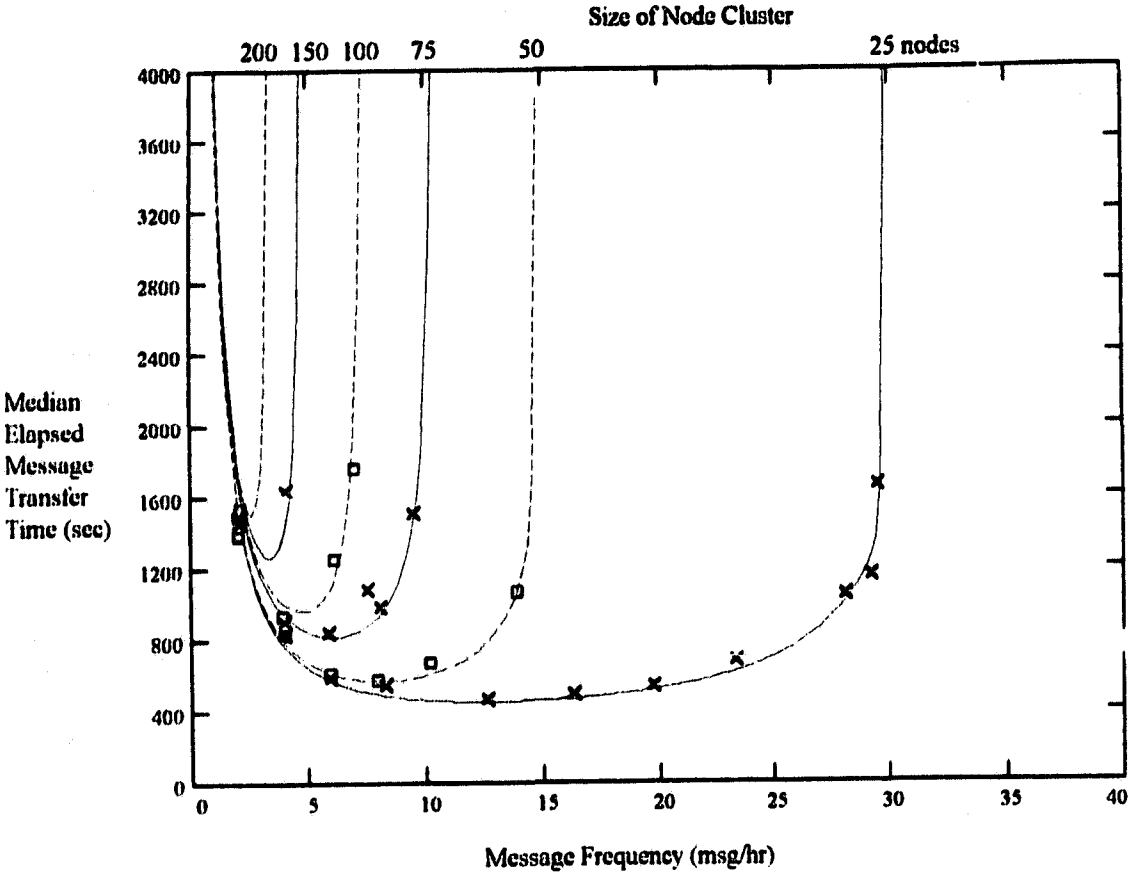


Figure 21. High Priority Message Transfer Times for Discrete-Event Simulation of a Ten-Server Star Cluster Employing Message Acknowledgments

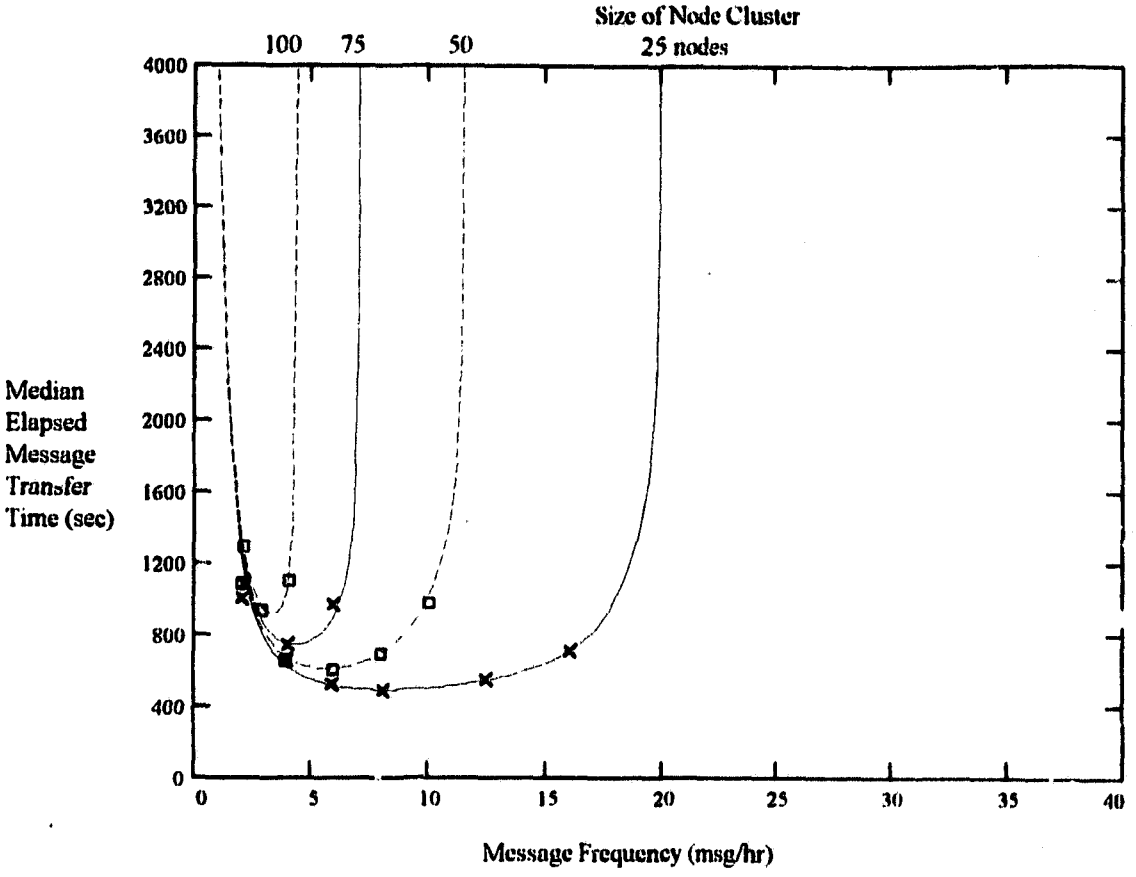
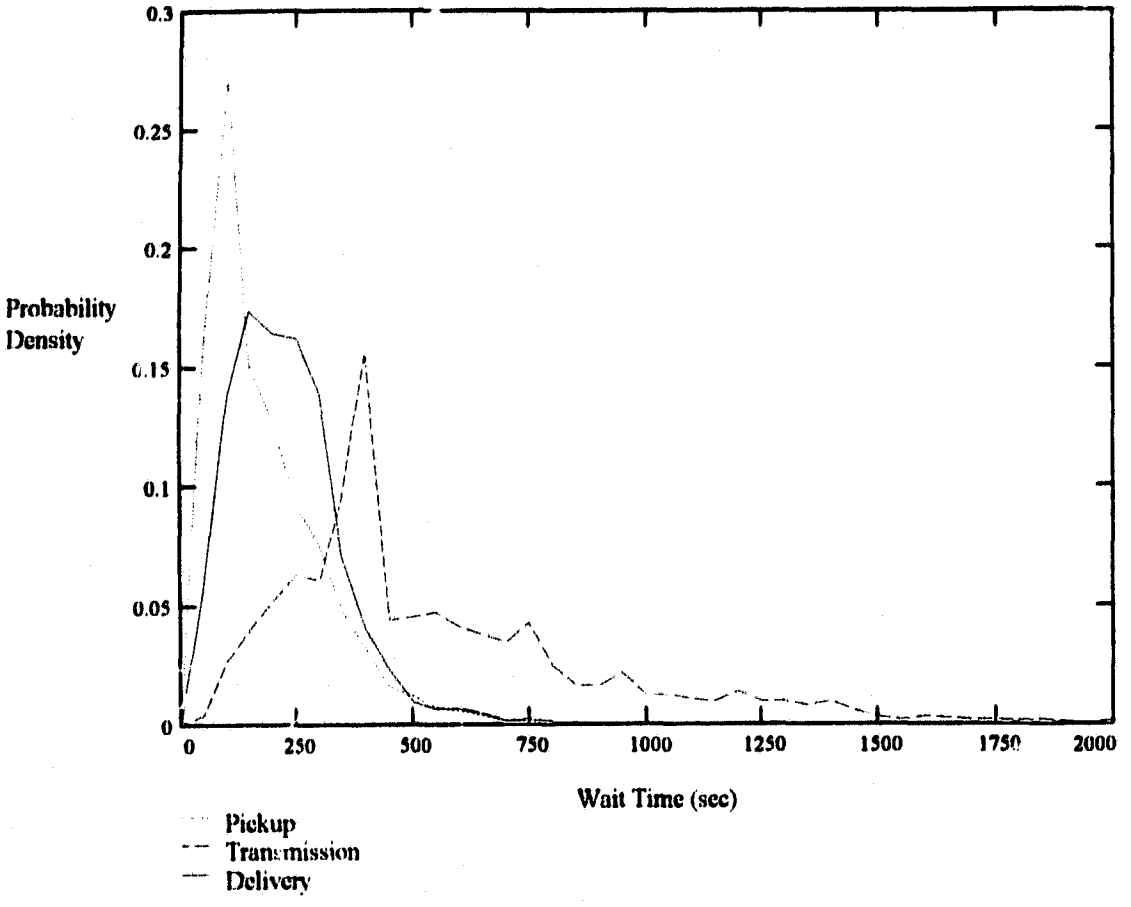


Figure 22. Probability Density Functions for Message Pickup, Transmission and Delivery Times of Medium Priority Messages in a Simulated Four-Node Mesh Cluster Operating at 24 Messages Per Hour



The results of the star clusters indicate:

1) The end-to-end transfer time is very sensitive to the message frequency. At very low frequencies, the transfer time is directly proportional to the mean message interarrival time. At higher frequencies, gateway contention causes significant delays. Only a narrow operating region is available.

2) As the number of servers (ports supported by the gateway) is increased, so does the width of the operating region (see Figures 17, 18 and 19). In this application, the gateway demonstrates a variation of the Lost Calls Cleared (Erlang B) regimen whereby those connection requests for a multi-channel server which are rejected are cleared (and later retried) rather than held in a queue waiting for a free channel [113].

3) Increases in the priority delay have little effect on throughput (see Figures 19 and 20). This is largely due to the low message frequencies associated with the operating region. For a priority delay to have a significant effect, it must be substantially larger than the message interarrival time. As the operating region typically ranges from three to ten messages per hour, the interarrival time is typically 720 to 1,200 seconds, which is approximately the same as the 600 second delay associated with the lowest priority messages.

4) Increases in the priority delay have a negative effect on the end-to-end transfer time (see Figures 19 and 20). At low message frequencies where there is little likelihood of having more than one message per mail bag, this delay directly increases the end-to-end transfer time.

5) The use of acknowledgment messages reduces the upper range of the operating region by about 40 percent but does not appear to affect the lower range of the operating region (see Figures 20 and 21).

5.2.4 Model Calibration and Validity

The mesh and star models closely emulate the *Health Link* software. Benchmarks for the Intel 80386SX25 computer given in Chapter 4 are used to provide some of the constants. As well, other constants are extracted from data recorded during runs of a real, four-node subnet. The real data was collected over a period of

days at sites involved in the field test described in Chapter 7. Each node was provided with an automatic, memoryless message generator which built standard 1,000-character text messages to be sent to other nodes in the subnet. The message generator was run at different frequencies ranging from a nominal rate of 12 messages per hour to 33 messages per hour in increments of three messages per hour. Message transfer times measured in the real runs are shown in Figure 23. A comparison of Figure 15 with Figure 23 shows that for the four-node cluster, there is a close correspondence. Tables 11 and 12 correspond to the data plotted in Figures 15 and 23, respectively. Table 11 is built on the assumption that the pickup, transmission and delivery times are statistically independent, thereby permitting the median, mean and variance of each time component to be subtotaled to calculate the median, mean and variance of the transfer time.

The star model is derived from the mesh. The modeling constants are the same where applicable. An additional structure has been added to the mesh model to represent the gateway. The gateway logic has no embedded time delays: it merely monitors the gateway ports and stores messages for forwarding. The simplicity of the structure may place a small favourable bias on the performance of the model.

A graph showing the probability density functions (based on time intervals of 50 seconds) for end-to-end transfer time and transmission time in a real cluster is shown in Figure 24. These plots are computed from data collected at a message frequency of 24 messages per hour. There is a close correspondence between Figure 22 (the discrete-event model) and Figure 24 (the real network). The plots of the transmission times are directly comparable. The plot of the end-to-end transfer time in Figure 24 fits well with what might be expected if the pickup and delivery times were added to the transmission time in Figure 22. Unfortunately, since the structure of the discrete-event model prevents taking a single measurement of end-to-end transfer time, there is no direct correspondence between simulated and actual plots for this variable.

Figure 23. Medium Priority Message Transfer Times in a Real Four-Node Mesh Cluster

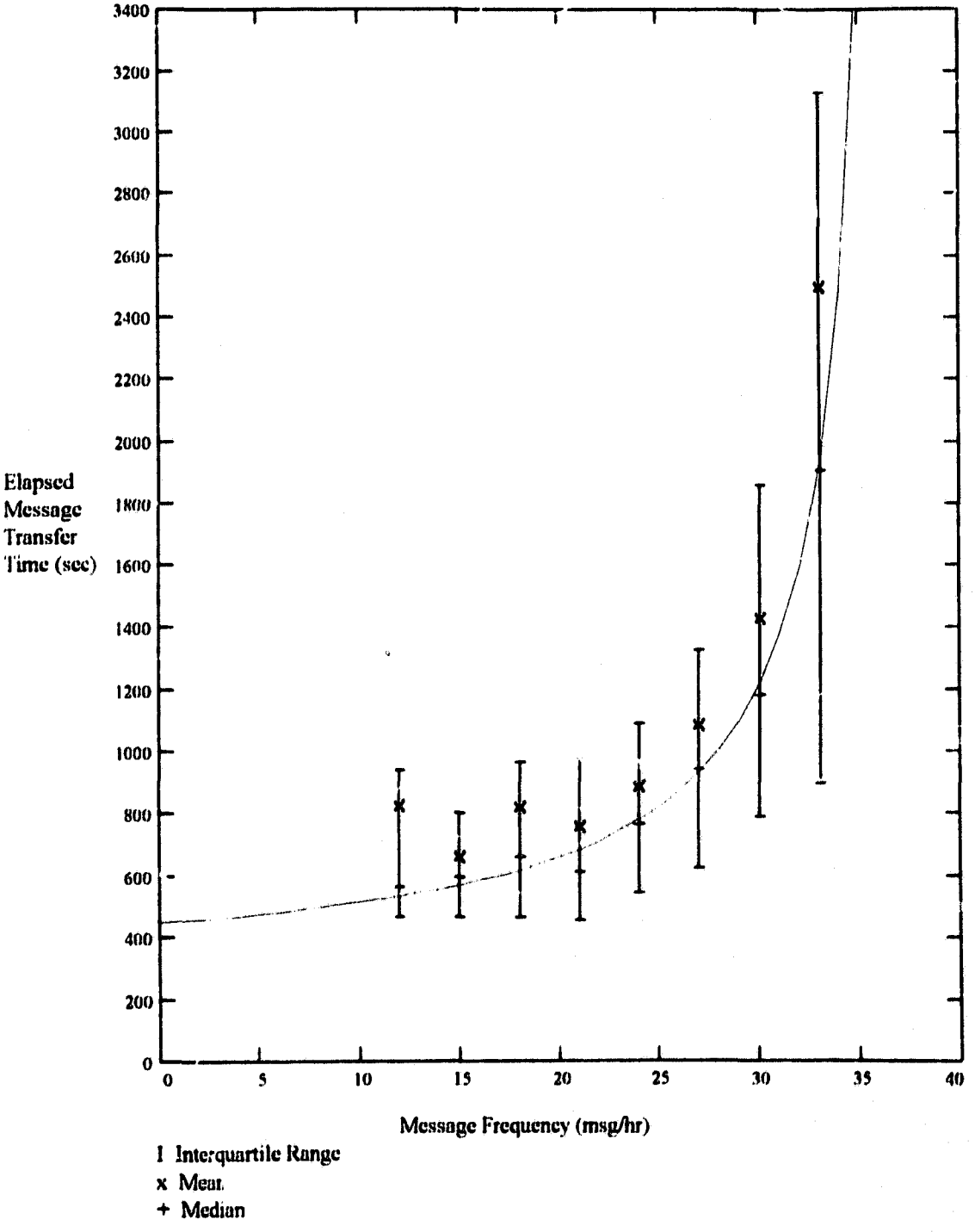


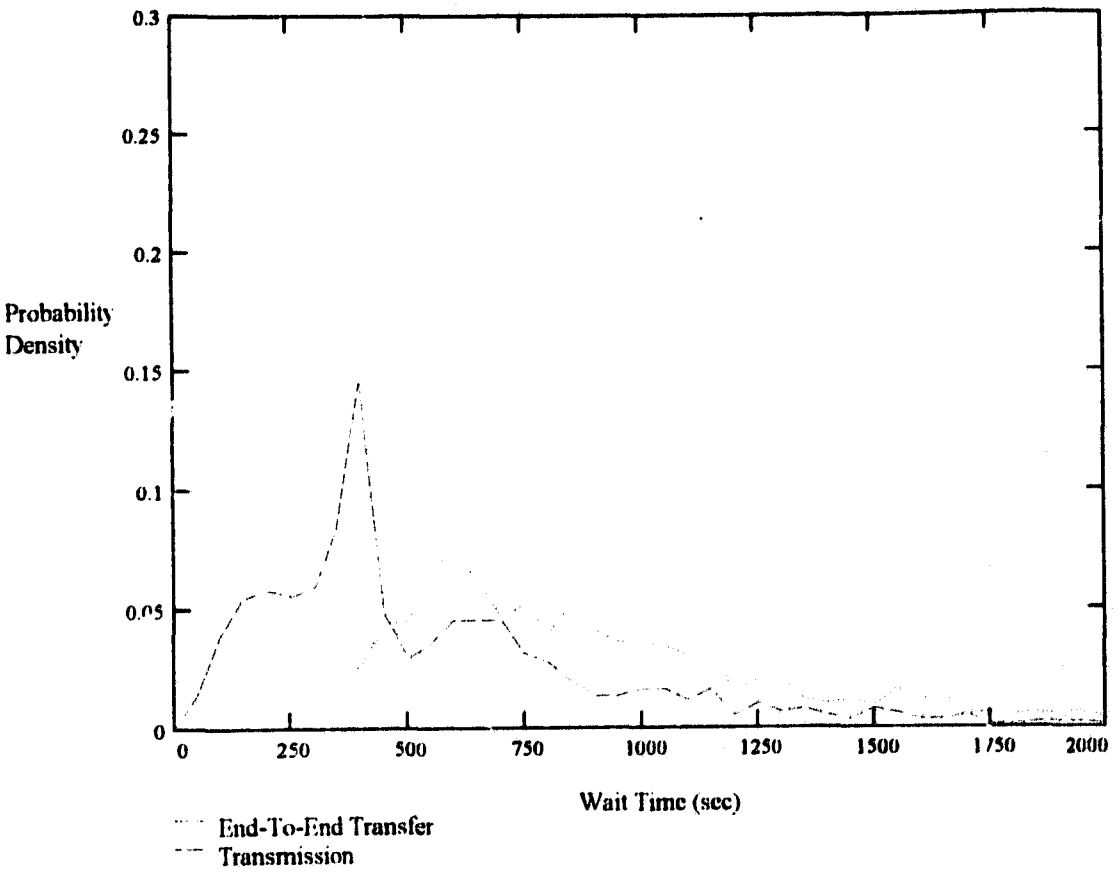
Table 11. Simulated Message Transfer Times in a Four-Node Mesh Cluster Operating at Medium Priority

Nominal Message Frequency (msg/hr)	Sample Size N	Message Transfer Time (sec) Median	Mean	Standard Deviation
12	circa 3000	465	569	384
15	circa 3000	510	634	425
18	circa 3000	560	735	517
21	circa 3000	635	817	582
24	circa 3000	805	1020	720
27	circa 3000	950	1142	809
30	circa 3000	1215	1468	1013
33	circa 3000	1945	2257	1459

Table 12. Message Transfer Times in a Real Four-Node Mesh Cluster Operating at Medium Priority

Nominal Message Frequency (msg/hr)	Sample Size N	Message Transfer Time (sec) 1st Quartile	2nd Quartile	3rd Quartile	Mean	Standard Deviation
12	358	464	559	936	820	662
15	198	468	592	799	659	314
18	234	464	660	964	819	544
21	255	459	611	972	757	445
24	1164	545	765	1086	885	501
27	658	623	942	1326	1081	633
30	721	788	1180	1853	1426	891
33	372	896	1901	3125	2497	2338

Figure 24. Probability Density Functions for Message End-to-End Transfer and Transmission Times of Medium Priority Messages in a Real Four-Node Mesh Cluster Operating at 24 Messages Per Hour



5.2.5 Confidence Intervals

Two aspects of the steady-state simulations must be considered to place an estimate on the size of the confidence intervals associated with their results. First, an estimate or elimination of the initial bias needs to be made [104]. Second, some method must be employed to remove the effects of auto-correlation introduced by the pseudo-random number generator used by GPSS to provide a Poisson-distributed input to the models [104]. Studies reported here are for the four-cluster mesh model generating 1,000-character medium priority messages at a rate of 24 messages per hour. The results of these studies indicate the size of the confidence intervals which might be expected in other simulation runs.

Table 13 displays the results from a study made of initialization bias. The simulation is broken into batches of 20 messages each. Batches are designated only for the purpose of data collection. The simulation and, accordingly, the first batch begins with an empty and idle initial state. The state of the model is maintained throughout the run, i.e., each batch takes up where the previous batch left off. To increase the sample population of each batch, each simulation is replicated 16 times. Each replication generates an independent sample.

Table 13 reports the mean message transfer times for each set of replicated batches. The mean times for the first batches are substantially less than those of the second batches for all frequencies reported. However, the means for the second batches are less only for those frequencies greater than 15 messages per hour. Similar assertions can be made for higher order batches. Although it is difficult to determine at what point steady-state is achieved, two observations seem possible: 1) most of the initialization bias occurs in the first batches, and 2) the number of batches effected by initialization bias increases with increasing message frequency.

Two approaches can be used to reduce the effect of initialization bias: 1) the initializing messages can be deleted, and 2) the initializing messages can be swamped by a very large sample population [104, 111]. The simulation runs presented in Section 5.2.4 employ swamping to compensate for initialization bias. If steady-state is reached after approximately 20 messages, those 20 messages will have an impact of less than one percent on the mean transfer time for a sample population of 3,000 and an impact of less than two percent for a sample population of 1,000.

Table 13. Impact of an Empty and Idle Initial State on Mean Message Transfer Times

Nominal Message Frequency (msg/hr)	Mean Message Transfer Time (sec) Batch 1	2	3	4	5	6
12	501	549	538	606	576	685
15	508	612	602	632	556	677
18	509	650	680	689	735	740
21	571	666	815	805	824	905
24	688	833	890	858	882	987
27	645	951	1013	1121	1122	1135
30	658	1129	1218	1212	1268	1298
33	749	1139	1196	1375	1472	1584

The second bias affecting the level of confidence is that of auto-correlation. If it could be asserted that the observations in a single sampling run were independent, it would be a minor calculation to determine the standard error. Consider n independent observations $\{X_1, X_2, \ldots X_n\}$ with the sample mean estimator, $\hat{\theta}$. The sample variance, S^2, is an unbiased estimator of the population variance where

$$S^2 = \sum_{i=1}^{n} \frac{\left(X_i - \hat{\theta}\right)^2}{n-1} \qquad \text{Eq. 5}$$

An unbiased estimator of the variance of $\hat{\theta}$ is

$$\hat{\sigma}^2\left(\hat{\theta}\right) = \frac{S^2}{n} \qquad \text{Eq. 6}$$

The standard error of the mean estimator, $\hat{\theta}$, then becomes

$$\text{s.e.}(\hat{\theta}) = \hat{\sigma}\left(\hat{\theta}\right) = \frac{S}{\sqrt{n}} \qquad \text{Eq. 7}$$

By assuming sampling independence, we calculate, from Table 11, standard errors ranging from 7.0 at a frequency of 12 messages per hour to 27.0 at a frequency of 33 messages per hour.

This calculation is, however, subject to the assumption of sampling independence. As a random pseudo-number generator is used by GPSS/H to simulate physically random events, this assumption could understate or overstate the confidence intervals depending on whether or not there is a positive or negative auto correlation, respectively, in the random number sequence [104].

If the data are collected from a series of independent, replicated simulations, the assumption of sampling independence can be made where each replication run is a member of the sample population. Each replication, r, has a sample mean

$$\overline{X}_r = \sum_{i=d+1}^{n+d} \frac{X_{ri}}{n} \qquad \text{Eq. 8}$$

where d is the number of initialization observations which are discarded. The grand mean is then

$$\overline{X}_{..} = \sum_{r=1}^{R} \frac{\overline{X}_{r.}}{R} \qquad \text{Eq. 9}$$

where R is the total number of replications. The sample variance is

$$S^2 = \frac{1}{R-1}\left(\sum_{r=1}^{R} \overline{X}_{r.}^2 - R\,\overline{X}_{..}^2\right) \qquad \text{Eq. 10}$$

Accordingly, the standard error for the replication runs is given by

$$\text{s.e.}(\overline{X}_{..}) = \frac{S}{\sqrt{R}} \qquad \text{Eq. 11}$$

Table 14 shows the mean message transfer time and the estimated standard error based on ten replications each consisting of 500 messages of 1,000 characters each sent at medium priority. The initialization bias is reduced in Table 14 by discarding the first 20 messages in each run. The means are directly comparable and in close agreement to those reported in Table 11 in the single replication case.

Table 14. Mean Message Transfer Times and Estimated Standard Error in a Ten-Fold Replication Study

Nominal Message Frequency (msg/hr)	Message Transfer Time (sec)	
	Mean	Standard Error
12	570	5
15	636	5
18	702	12
21	825	10
24	948	22
27	1133	16
30	1350	33
33	2111	207

5.3 Analytic Models

The two analytic models are structurally similar to the discrete-event models. The multi-nodal driver section in both models reflects the behaviour of the adapter section. Message preparation requests at time of pickup and delivery are treated as *customers* in a single server (M/M/1) queue for the CPU. (See Table 15 for a list of standard queueing notations.) A calculation based on local and remote port availability is made to determine the mean waiting time before a connection can be established for message exchange. The mean waiting time is proportional to the mean number of messages exchanged during a connection which, in turn, determines the mean connection time; the local and remote port availability; and the mean end-to-end message transfer time.

For the star cluster, the hub or gateway is accessed by both the source and destination of a message. This behaviour is symmetric; the exchange of the mail bag from the source to the gateway is stochastically the same as the exchange of the mail bag from the gateway to the destination as both processes are conducted in a full-duplex mode.

Table 15. Queueing Notation (A/B/X/Y/Z) [105:9]

Characteristics	Symbol	Explanation
Interarrival time distribution (A) and service time distribution (B)	M	Exponential or Markovian
	D	Deterministic
	E_k	Erlang
	H_k	Hyperexponential
	PH	Phase type
	G	General
Number of parallel servers (X) and restriction on system capacity (Y)	1, 2, 3, ..., ∞	
Queue discipline (Z)	FCFS	First come, first served
	LCFS	Last come, first served
	RSS	Random selection for service
	PR	Priority
	GD	General discipline

5.3.1 Implementation of the Mesh Model

Both the mesh and star analytic models are iterative. This is because the mean connection time has a recursive dependency on the port availability at both the local and remote ports. An increase in connection time due to a higher number of messages being exchanged (to various destinations) results in greater port utilization which, in turn, increases the number of messages waiting in mail bags to be exchanged. Fortunately, a simple univariate method [114] of approximation, which demonstrates linear convergence, can be used to determine the equilibrium. From an initial starting estimate of two messages to be exchanged (one in either direction), the equations described in 5.8 through 5.14 are applied iteratively until the mean time between connection retries, t_{rtry}, changes by less than 10^{-12} seconds.

$$\tau_{conn_{i+1}} = T_{conn} + T_{xfr} \cdot \eta_{xfr_i} \qquad \text{Eq. 12}$$

$$\lambda_{conn\ i+1} = \frac{2\lambda}{\eta_{xfr\ i}} \qquad \text{Eq. 13}$$

$$\rho_{port\ i+1} = \frac{\rho_{port\ i} + \lambda_{conn\ i+1} \cdot (\eta_{node} - 1) \cdot \left[\tau_{conn\ i+1} + T_{full} \cdot \left(\frac{\rho_{remote\ i}}{1 - \rho_{remote\ i}} \right) \right]}{2} \qquad \text{Eq. 14}$$

$$\rho_{remote\ i+1} = \rho_{port\ i+1} \left(\frac{\eta_{node} - 2}{\eta_{node} - 1} \right) \qquad \text{Eq. 15}$$

$$\tau_{rtry\ i+1} = \frac{\tau_{rtry\ i} + \frac{\Phi(\lambda)}{2} \cdot \left(\frac{\rho_{remote\ i+1}}{1 - \rho_{remote\ i+1}} \right)}{2} \qquad \text{Eq. 16}$$

$$\tau_{Q\ i+1} = \frac{\tau_{conn\ i+1} \cdot \rho_{port\ i+1}}{1 - \rho_{port\ i+1}} \qquad \text{Eq. 17}$$

$$\eta_{xfr\ i+1} = 2 \cdot \left[1 + \lambda \cdot \left(\tau_{rtry\ i+1} + T_{prio} + \tau_{Q\ i+1} \right) \right] \qquad \text{Eq. 18}$$

Function, $\Phi(\lambda)$, is defined as:

$1.5 \cdot T_{rtry}$ for $\lambda < \frac{1}{1.5 \cdot T_{rtry}}$,

$\frac{1}{\lambda}$ for $\lambda \geq \frac{1}{1.5 \cdot T_{rtry}}$.

For Equations 12 through 18, tne constants are defined as:

T_{conn}, the time taken to establish a connection, to disconnect and to set the modem to accept incoming calls

T_{xfr}, the time taken to transfer a single message in full-duplex mode,

T_{fail}, the time taken to make a connection attempt, detect connection failure and reset the modem,

T_{prio}, the time a message is held due to a priority delay, and

T_{rtry}, the wait time between connection retries (referred to as RetryWait in Section 4.9.3).

The independent variables are defined as:

λ, the mean rate of message creation per unit time at a particular local node for any individual destination and

n_{node}, the number of nodes in the cluster, including the local node.

The dependent variables are defined as:

τ_{conn}, the mean time that a port is busy due to a successful connection,

η_{xfr}, the mean number of messages transferred during a successful connection in either direction,

λ_{conn}, the mean number of connections made per unit time,

ρ_{port}, the utilization ratio of the local communications port,

ρ_{remote}, the utilization ratio of any individual remote communications port from the perspective of a local node initiating a connection request,

τ_{rtry}, the mean time between connection retries, and

τ_{Q}, the mean queueing time for connection requests.

Once the equilibrium is calculated, the mean end-to-end message transfer time, τ_{ETOE}, is computed (see Equations 19 and 20).

$$\rho_{CPU} = \rho_{port} + 2 \cdot T_{prep} \cdot \lambda \cdot (n_{node} - 1) \qquad \text{Eq. 19}$$

$$\tau_{EToE} = \tau_{retry} + T_{prtw} + \tau_{Q} + \tau_{conn} + \frac{2 \cdot T_{prep}}{1 - \rho_{CPU}} \qquad \text{Eq. 20}$$

In Equations 19 and 20, there is the additional constant,

T_{prep}, the mean time to prepare a message for sending or for delivery,

and the additional dependent variable,

ρ_{CPU}, the utilization of a node's CPU due to a combination of message preparation and communication.

The duration of the connection (see Equation 12) is the sum of the constant time needed to make and break the connection and the time taken to exchange mail bags (one mail bag in either direction). The number of messages in a mail bag is half of the number of messages transferred during the connection. Thus, the number of connections made per unit time, shown in Equation 13, is the rate at which the messages arrive (λ) divided by the number of messages in a mail bag, η_{xfr}.

The utilization of the local port is based on the amount of time the port spends with connection successes and connection failures. Equation 14 is used to calculate port utilization based on a moving average with the previous iteration. After disregarding the terms associated with the moving average calculation, the equation simplifies to two terms: one for connection successes and one for connection failures (see Equation 21).

$$\rho_{port_{i+1}} = \lambda_{conn_{i+1}} \cdot (n_{node} - 1) \cdot \left[\tau_{conn_{i+1}} + T_{full} \cdot \left(\frac{\rho_{remote_i}}{1 - \rho_{remote_i}} \right) \right] \qquad \text{Eq. 21}$$

The term representing connection failures is the sum of the geometric series, $T_{full} \cdot (\rho_{remote} + \rho_{remote}^2 + \rho_{remote}^3 + ...)$, which results from the possibility for repeated failures. As the local node communicates with each of its $(n_{node} - 1)$ peers, this factor is applied to both terms. The factor of λ_{conn} converts the utilization time into a utilization ratio.

It is necessary to estimate the utilization of a remote port in order to determine the likelihood that a connection request is rejected at its destination. Equation 15 expresses this utilization as a proportion of the local port utilization. It is assumed that a busy remote port could only be connected to another remote port, not to the local port. (If the remote port were connected to the local port already, then there would be no cause for a connection request.) Therefore, the remote port could only be busy with $(n_{node} - 2)$ of the $(n_{node} - 1)$ peers of the local port.

Once the utilization of the destination port is determined, it is possible to calculate the time between connection retries. Equation 16, like Equation 14, is a moving average. Further inspection of the equation reveals that it contains a factor, $\left(\frac{\rho_{remote}}{1 - \rho_{remote}}\right)$, the sum of a geometric series, which estimates the mean number of connection retries. The function expressing the time between connection retries, $\Phi(\lambda)$, is a rough approximation to the algorithm described in Sections 4.9.1 and 4.9.2. As most connection attempts are triggered by new messages freshly added to the delivery queue, this approximation has little affect on the output of the model. The time between connection retries is halved because either the sending or receiving node can initiate a connection.

Contention for the local port is also included in the model as there is competition among connection requests to different destinations. The connection requests are served from a M/M/1 queue (see Equation 17). The mean queueing time, the length of time during which the connection request is waits in the queue for service, is part of the delay which the messages experience during transfer. Equation 18 subtotals this time with delays caused by connection retries and priority wait. This total time is multiplied by the message creation rate to determine the number of messages arriving after the first message is captured in a mail bag. Since two mail bags are exchanged when a connection is established (one by the source and one by the destination), the total number of messages exchanged is twice the size of a mail bag.

Once an equilibrium has been calculated for the port utilization, CPU utilization can be computed (see Equation 19). The CPU performs three disjoint operations which affect its utilization: 1) it maintains communication services, 2) it prepares messages as they are sent, and 3) it prepares messages as they are delivered. The port utilization reflects the impact of communication services on CPU utilization. This is supplemented by the utilization of the CPU due to message preparation.

Finally, Equation 20 provides an estimate of the mean end-to-end transfer time by totaling the message delays due to remote port busy, priority wait, local port contention, connection time and two stages of preparation. (The messages wait in a M/M/1 queue for the CPU during message preparation.)

5.3.2 Implementation of the Star Model

The analytic model for the star cluster is similar to the model for the mesh cluster. Many of the equations for the star are similar to those for the mesh model. The star model includes additional equations representing contention for gateway services and message transfer delays introduced by the store-and-forward function of the gateway. The recursive section of the model is expressed in Equations 22 through 25.

$$\lambda_{conn\,i+1} = \frac{2\lambda}{\eta_{xfr\,i}} \qquad \text{Eq. 22}$$

$$\rho_{gateway\,i+1} = \frac{\rho_{gateway\,i} + \left(\begin{matrix} n_{node} - 1 \\ n_{node} \end{matrix}\right) \cdot \left[\frac{n_{node} \cdot \lambda_{conn\,i+1} \cdot \left(T_{reset} + T_{xfr} \cdot \eta_{xfr\,i}\right)}{n_{server}}\right]^{n_{server}}}{2} \qquad \text{Eq. 23}$$

$$\tau_{rtry\,i+1} = \frac{\tau_{rtry\,i} + \Phi(\lambda) \cdot \left(\frac{\rho_{gateway\,i+1}}{1 - \rho_{gateway\,i+1}}\right)}{2} \qquad \text{Eq. 24}$$

$$\eta_{xfr\,i+1} = 2 \cdot \left[1 + \lambda \cdot \left(\tau_{rtry\,i+1} + T_{prio}\right)\right] \qquad \text{Eq. 25}$$

In addition to the constants defined for mesh model, there is

T_{reset}, the time taken to disconnect and to set the modem to accept incoming calls.

There is a newly declared independent variable,

n_{server}, the number of servers (ports) supported at the gateway,

and dependent variable,

$\rho_{gateway}$, the utilization ratio of the gateway.

Once an equilibrium has been reached, the Equations 26 through 29 are used to compute the final end-to-end transfer time.

$$\tau_{conn} = T_{conn} + T_{xfr} \cdot \eta_{xfr} \qquad \text{Eq. 26}$$

$$\rho_{port} = \lambda_{conn} \cdot \left[\tau_{conn} + T_{full} \cdot \left(\frac{\rho_{gateway}}{1 - \rho_{gateway}} \right) \right] \qquad \text{Eq. 27}$$

$$\rho_{CPU} = \rho_{port} + 2 \cdot T_{prep} \cdot \lambda \qquad \text{Eq. 28}$$

$$\tau_{EToE} = \tau_{retry} + T_{prio} + 2 \cdot \tau_{conn} + \frac{2 \cdot T_{prep}}{1 - \rho_{CPU}} + \frac{1}{\lambda_{conn}} \qquad \text{Eq. 29}$$

Like the mesh model, the number of connections per unit time is based on the rate at which messages are created at a node divided by the number of messages in a mail bag (see Equation 22).

As connections are initiated by the nodes only, not the gateway, availability of gateway ports or servers has a major impact on transfer delays. Equation 23 expresses the gateway utilization ratio as a moving average with the previous iteration. Once the terms associated with the averaging are disregarded, Equation 30 results.

$$\rho_{gateway_{i+1}} = \left(\frac{n_{node} - 1}{n_{node}}\right) \cdot \left[\frac{n_{node} \cdot \lambda_{conn_{i+1}} \cdot \left(T_{resp} + T_{xfr} \cdot \eta_{x^{*}_{i}}\right)}{n_{server}}\right]^{n_{server}}$$

Eq. 30

The factor containing the exponent, n_{server}, is the binomial probability that all ports at the gateway are busy. (This expression is somewhat different from the Erlang B regimen because the latter assumes that calls which encounter a busy condition are cleared forever [113] whereas, in our case, calls are resubmitted after a waiting period.) The local node cannot contribute to the utilization ratio as it is attempting to connect with the gateway. Accordingly, the gateway utilization is factored down by $\left(\frac{n_{node} - 1}{n_{node}}\right)$.

If a connection request fails because no gateway ports are available, then a repeated request is made after a delay specified by $\Phi(\lambda)$ in Equation 24. This equation follows the same form as Equation 16 described in Section 5.3.1.

Equation 25, the final equation of the recursive section, is used to calculate the total number of messages exchanged during a connection between a node and the gateway. The number of messages in a mail bag is estimated to be the first message which triggers the connection request plus those messages accumulated in the mail bag while it awaits delivery.

Equations 26 through 29 are used to calculate the mean end-to-end message transfer time. Equation 26 models the mean duration of a connection as the connection and disconnection overheads plus the time spent exchanging mail bags. Equation 27 is used to calculate the port utilization based on the connection time and the connection request failure time. CPU utilization, as defined in Equation 28, is based on communication and message preparation services rendered.

The final result, the mean end-to-end transfer time, consists of three major components: 1) the time spent to pickup and send a message to the gateway, 2) the gateway holding time, and 3) the time spent to pickup a message at the gateway and deliver it to its destination. The first component consists of the priority delay, T_{prio}; the mean time between connection retries, τ_{rtry}; the mean message preparation time,

the mean elapsed time between successive connections with a given destination or $\left(\frac{1}{\lambda_{conn}}\right)$. The third component consists of the mean connection time, τ_{conn}, and the mean message preparation time, $\left(\frac{T_{prep}}{1-\rho_{CPU}}\right)$.

5.3.3 Assumptions

The two analytic models are based on a number of assumptions. Although some of the assumptions are particularly strong ones, it appears that the models are still fairly predictive. The major assumptions are:

1) every message experiences a constant priority delay regardless of its order in the mail bag or when the mail bag is sent,
2) the queueing discipline for CPU service is M/M/1,
3) there is statistical independence among connection requests, priority delays and queueing for CPU services,
4) mail bags contain a minimum of one message and communication is consistently full-duplex,
5) the CPU load resulting from servicing communication interrupts has no significant impact on CPU availability,
6) connection requests are triggered by the arrival of new messages in a mail bag, not by the requeueing of undelivered messages as a result of connection failure,
7) the communication links are as noise-free as those used in the live calibration runs,
8) communication and message preparation are mutually exclusive operations which queue for CPU service,
9) the message preparation time for messages being received (relating to decompression, decryption and signature verification) is the same as that for messages being sent (relating to compression, encryption and signature generation),
10) any delays incurred by searching queues, reorganizing data bases, reordering lists and performing directory searches is negligible,

11) only three outcomes can result from a connection request in a mesh cluster: call accept, no answer, and destination busy (and the ratio of time delays associated with no answer and destination busy is a constant),

12) only two outcomes can result from a connection request in a star cluster: call accept and destination busy,

13) connection requests are never initiated simultaneously at both a node and its destination,

14) all nodes except for the gateway behave uniformly the same,

15) message generation for any individual node is uncorrelated with any other node, and

16) message interarrival times are exponentially distributed at any individual node.

The first four assumptions probably have the greatest effect on the accuracy of the model. Assumption 1) tends to over-estimate the mean message transfer time because messages are often added to a mail bag and then sent without having to wait their full priority delay. As a message is added to a mail bag, it is assigned a trigger time in accordance with its priority. However, the message with the earliest trigger time supersedes the trigger time of all other messages in the mail bag. Furthermore, if a mail bag is being held for future delivery but the destination establishes a connection first, then the mail bag is transmitted immediately, regardless of its trigger time and, consequently, the priority of its messages.

Contrary to assumption 2), the queue for the CPU really follows a general (G/G/1) priority-based queueing discipline rather than a Markovian (M/M/1) one. However, we have chosen to model the Markovian discipline because of its simplicity even though the service time is bi-modal and the interarrival times are non-Markovian. This is because the CPU provides two distinct but related services. First, the CPU performs message preparation which is typically a constant service time and, second, the CPU drives the communication protocol which is most likely to conform best to an Erlang distribution. It is probable that this assumption tends to over-estimate the mean transfer time as the standard deviation of the service time may be less than the mean service time.

Wherever message delay components are summed in the equations, it is assumed that the components are independent. However, assumption 3) is patently false. It is impossible to separate the activities of message preparation, message priority delays, connection management and communication. These activities are stages in the message transfer process; one activity completing frequently has a direct impact on start and duration of the next. The importance of this assumption is difficult to determine without empirical testing.

Empirical observations contradict assumption 4). The performance of the actual protocol indicates that full-duplex communication is significantly slower than half-duplex communication. For example, wall times of 32 seconds have been measured for full-duplex in comparison with 13 seconds for half-duplex transmission of a single 1000-character message. This is largely due to having only a single-bit sliding window protocol which causes a frame to stop and wait for an acknowledgment piggybacked on a full-sized frame. At low message frequencies, it is quite possible that only half-duplex communication will occur, thereby significantly reducing the estimated message transfer time. Even at higher frequencies, there is a mix of messages which are transmitted using both full- and half-duplex modes. A constant transmission time, T_{tr}, of 23 seconds has been used for the simulations.

5.3.4 Results

Figures 25 through 27 show the results of analytic simulations of Intel 80386SX25 platforms in mesh clusters. The mean message transfer time for low, medium and high priority messages is shown for different message frequencies. The results shown in Figures 25 and 26 correspond to the results for the discrete-event model shown in Figures 14 and 15, respectively. As well, the plot for the four-node cluster in Figure 26 corresponds to Figure 23 which describes the performance of the real network. A comparison between the figures for the analytic and discrete-event models indicates that the analytic model over-estimates the levels of saturation for message traffic. The vertical asymptotes occur at greater message frequencies for the analytic than for the discrete model.

The linear portion of the curves have a steeper slope for the analytic model than do those for the discrete-event model. This could be due to graphing the mean message transfer time in the analytic model rather than the median message transfer time as in

the discrete-event model or it could be due to one or more of the assumptions described in Section 5.3.3.

Figures 28 through 32 show the results of analytic simulations for star clusters. Figures 28 through 30, which describe the performance of the cluster for medium priority messages, differ by the number of servers (ports) supported by the gateway. These three figures correspond to Figures 17 through 19 for discrete-event simulations. The single server graph in Figure 28 demonstrates a slope discontinuity at 13.3 messages per hour resulting from the formulation of $\Phi(\lambda)$. As well, all three analytic figures have a higher minimum mean transfer times than the corresponding minimum median transfer times shown in Figures 17 through 19. Although this may be partly due to the difference in the measure of central tendency, it could also be due to assumptions 1) and 4). Unlike the analytic mesh simulations, the star simulations appear to saturate at lower message frequencies than do the discrete-event simulations.

Figures 30 through 32 indicate that raising the message priority has a positive influence on the message delivery time but not on the throughput of the network. This is because the message transfer time of the star cluster at low message frequencies is dependent on the frequency at which a destination node contacts the gateway. At high message frequencies, the contention for gateway resources limits throughput. The lowest message priority is set to 600 seconds which corresponds to a message frequency of six messages per hour. The throughput of the cluster is largely unaffected by the priority delay because the messages wait at the gateway for roughly same amount time, or longer. Accordingly, the messages are delayed by the priority wait with no improvement in throughput. At high message frequencies, the gateway saturates. Under this condition, lower message priority has little effect on throughput because messages are delayed by connection failures.

Figure 25. Low Priority Message Transfer Times for Analytic Simulation of a Mesh Cluster

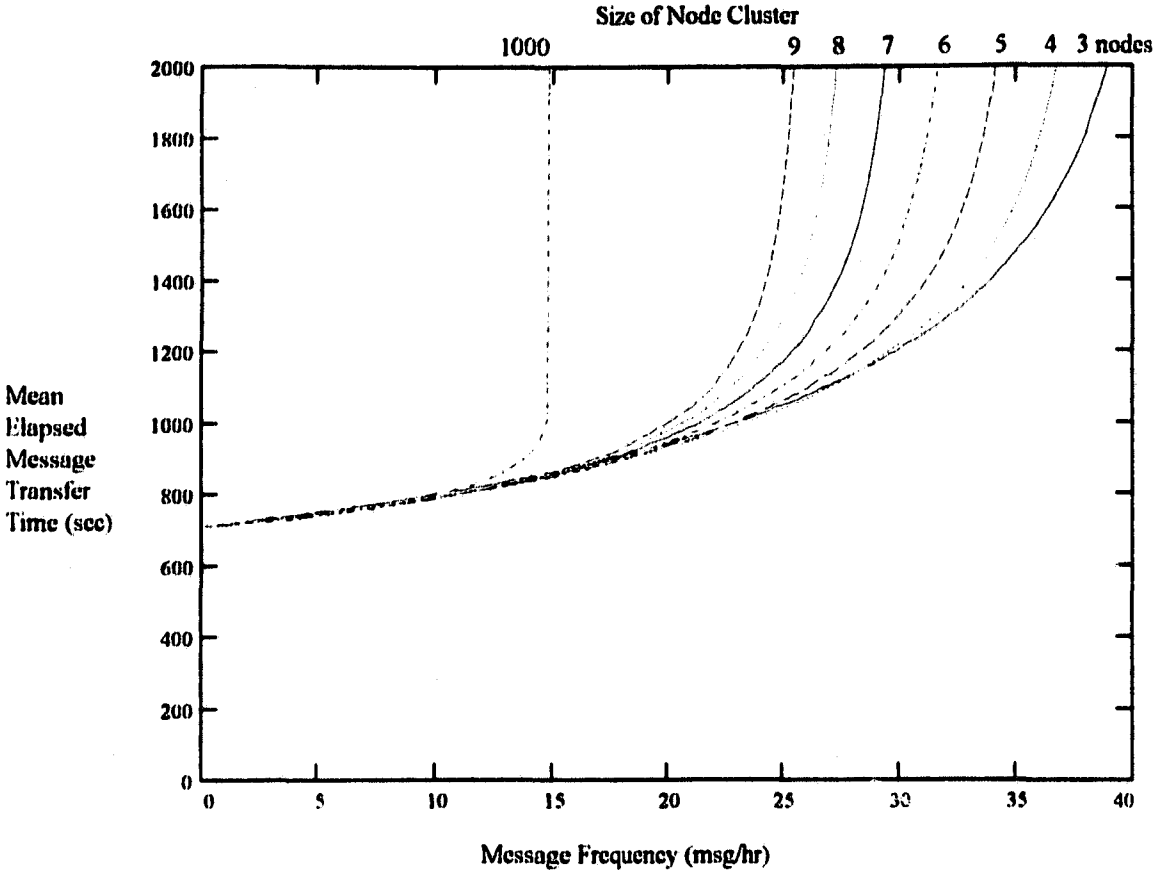


Figure 26. Medium Priority Message Transfer Times for Analytic Simulation of a Mesh Cluster

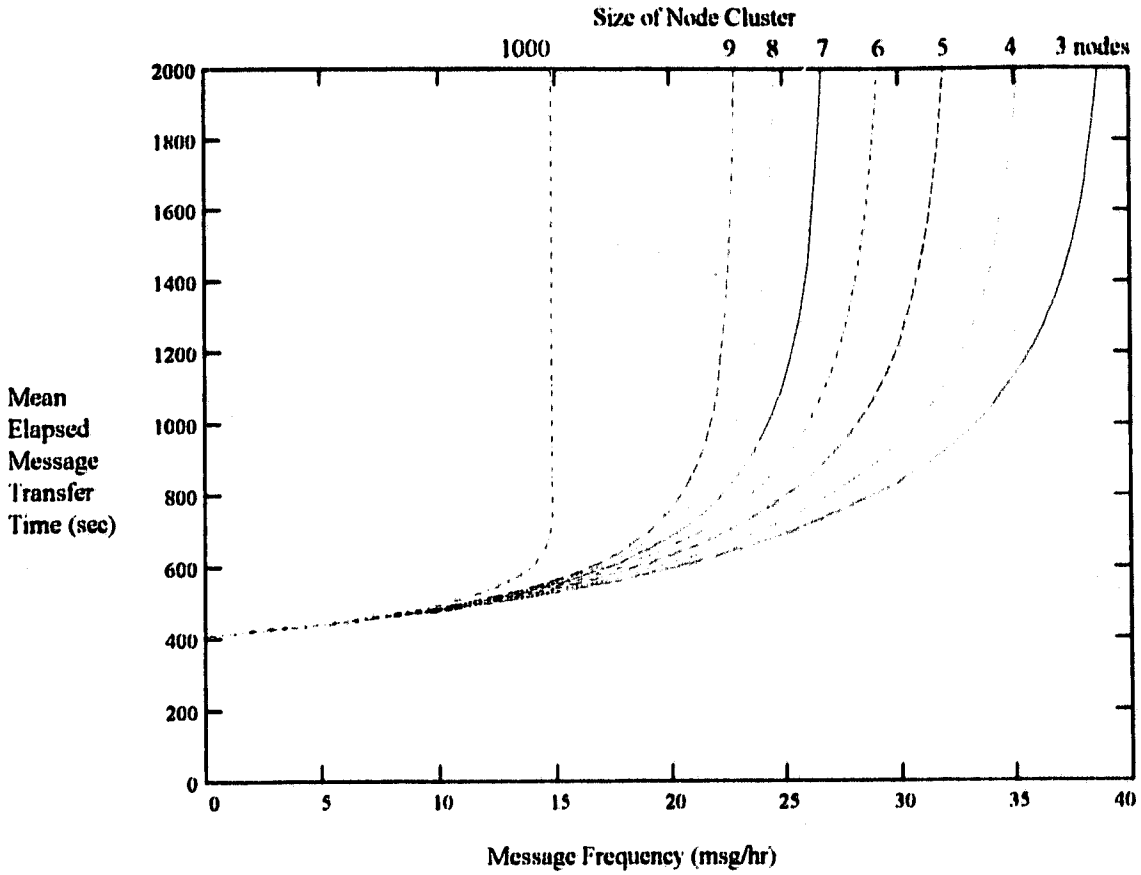


Figure 27. High Priority Message Transfer Times for Analytic Simulation of a Mesh Cluster

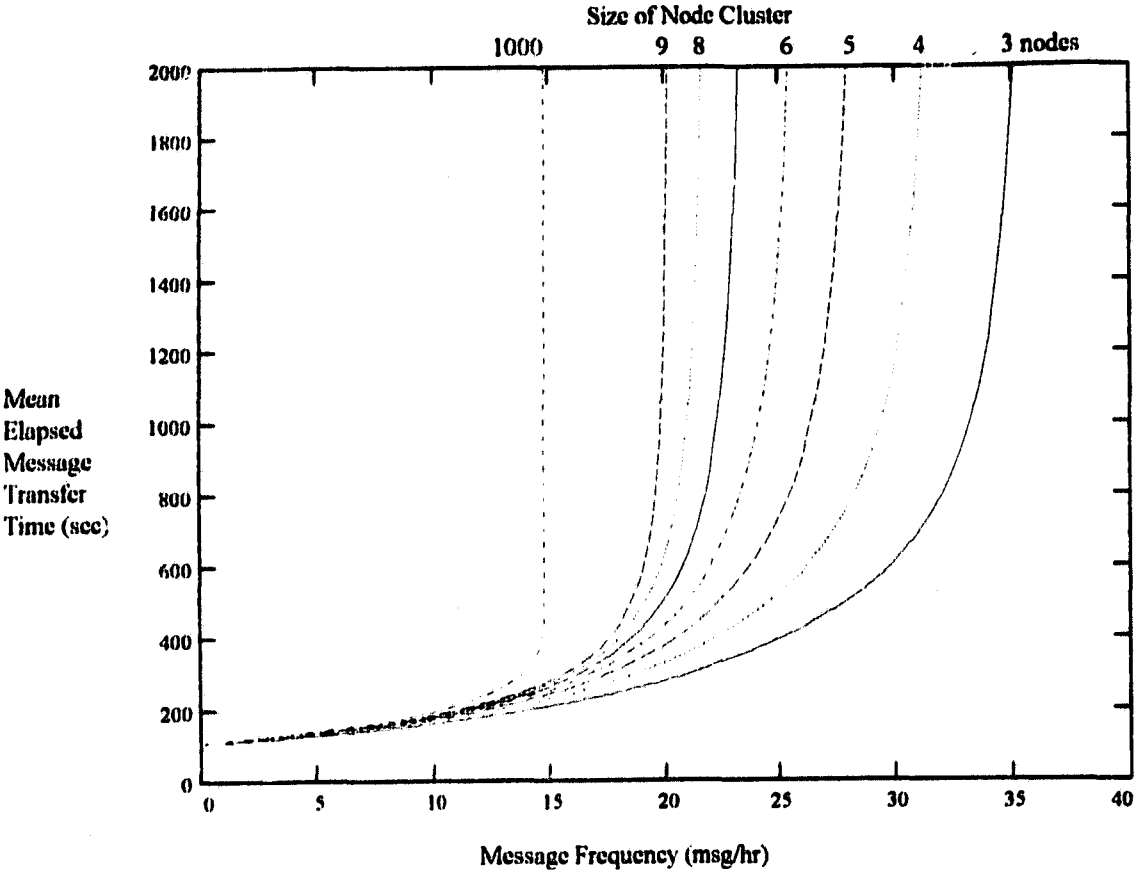


Figure 28. Medium Priority Message Transfer Times for Analytic Simulation of a Single-Server Star Cluster

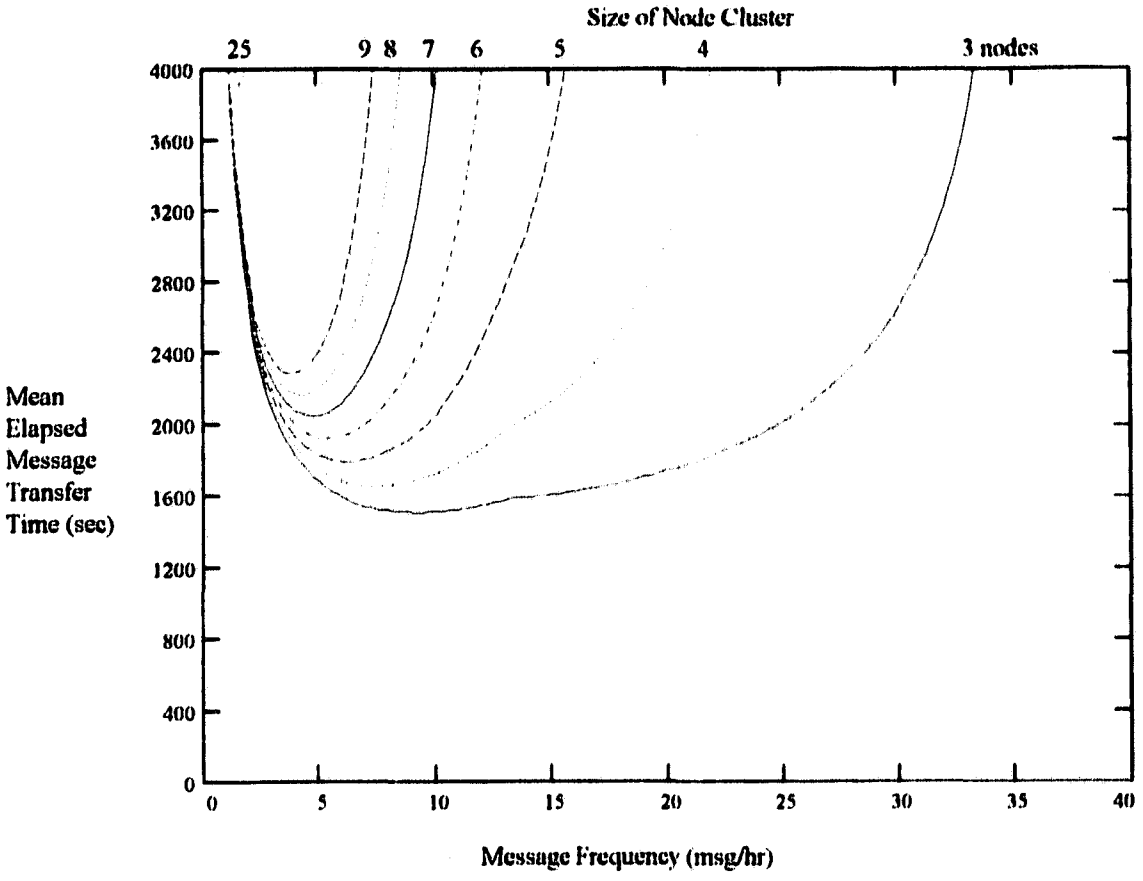


Figure 29. Medium Priority Message Transfer Times for Analytic Simulation of a Two-Server Star Cluster

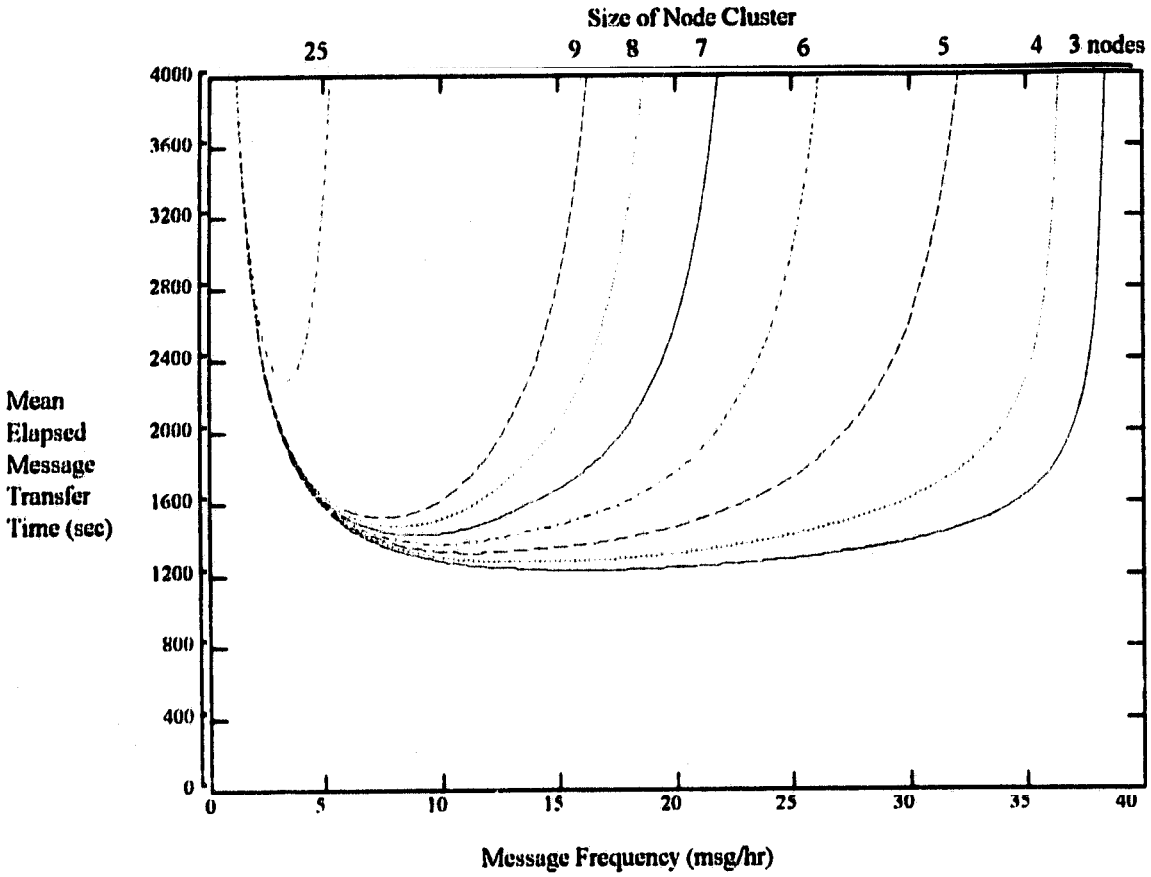


Figure 30. Medium Priority Message Transfer Times for Analytic Simulation of a Ten-Server Star Cluster

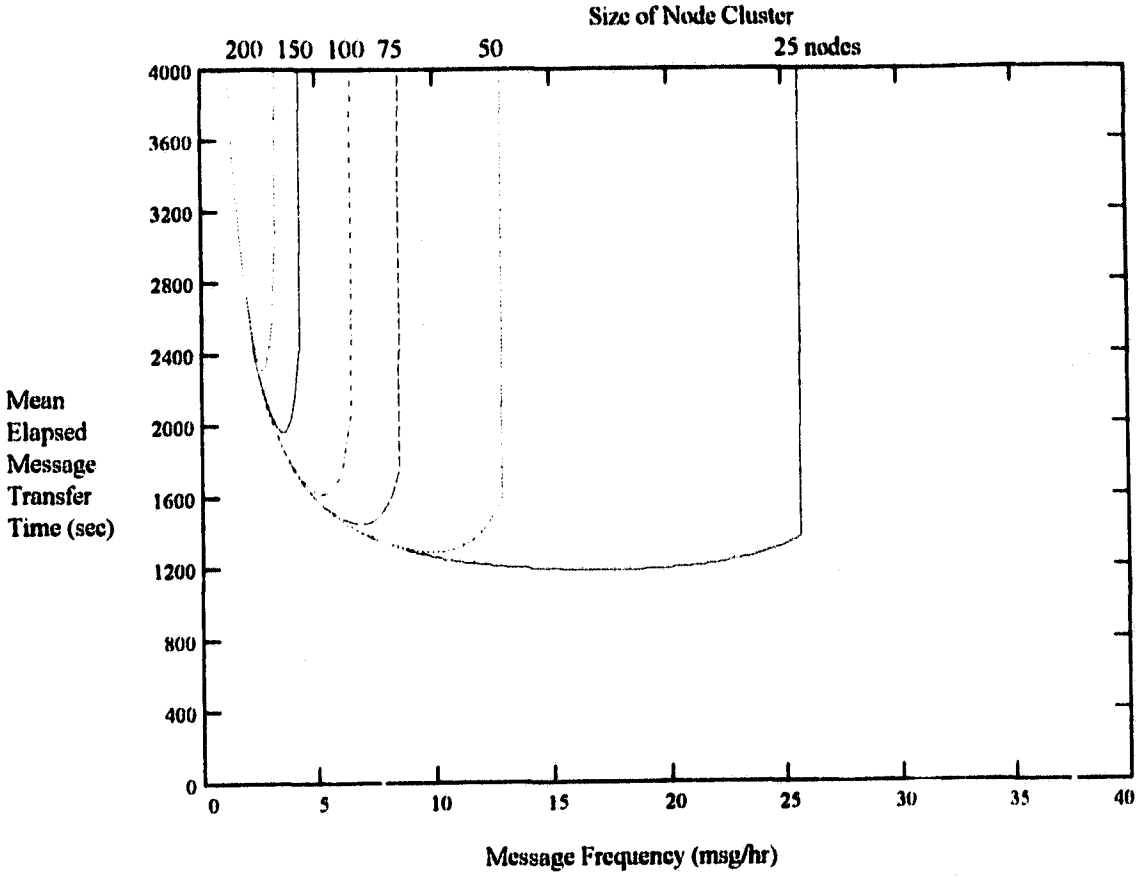


Figure 31. Low Priority Message Transfer Times for Analytic Simulation of a Ten-Server Star Cluster

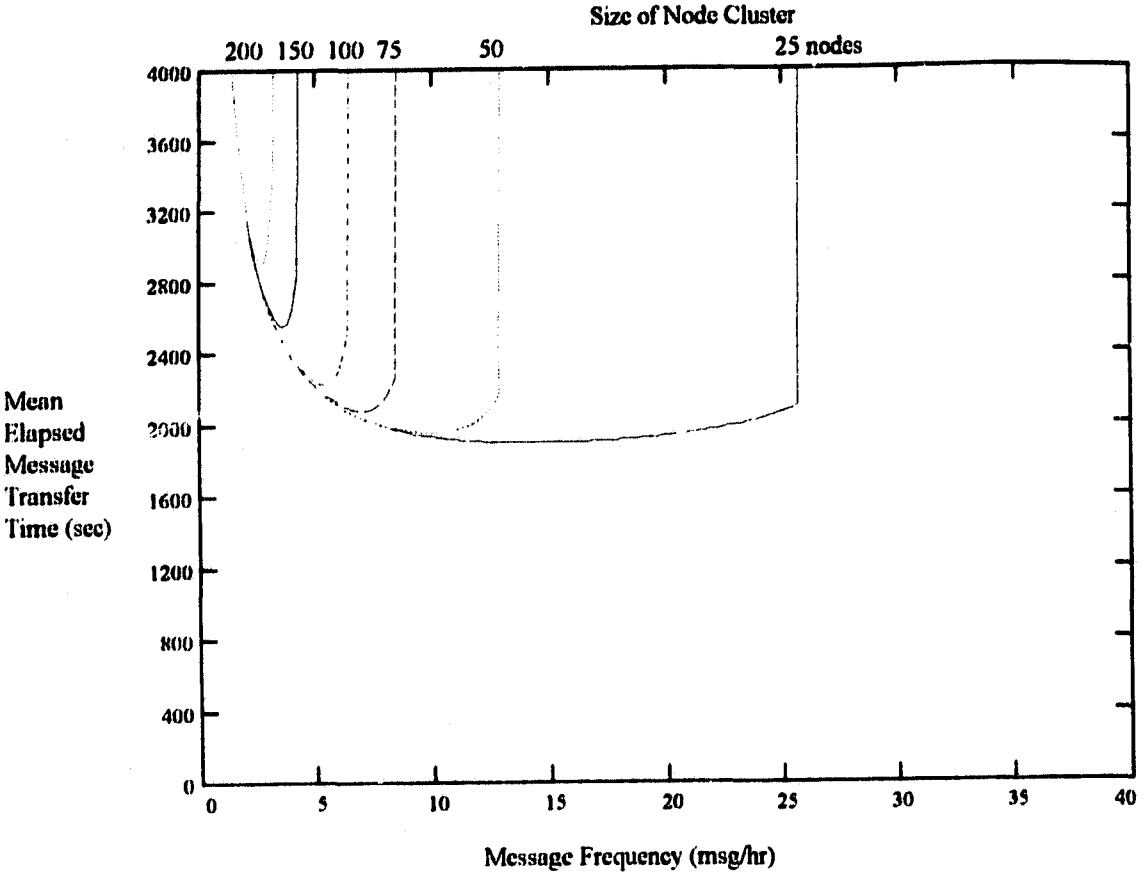
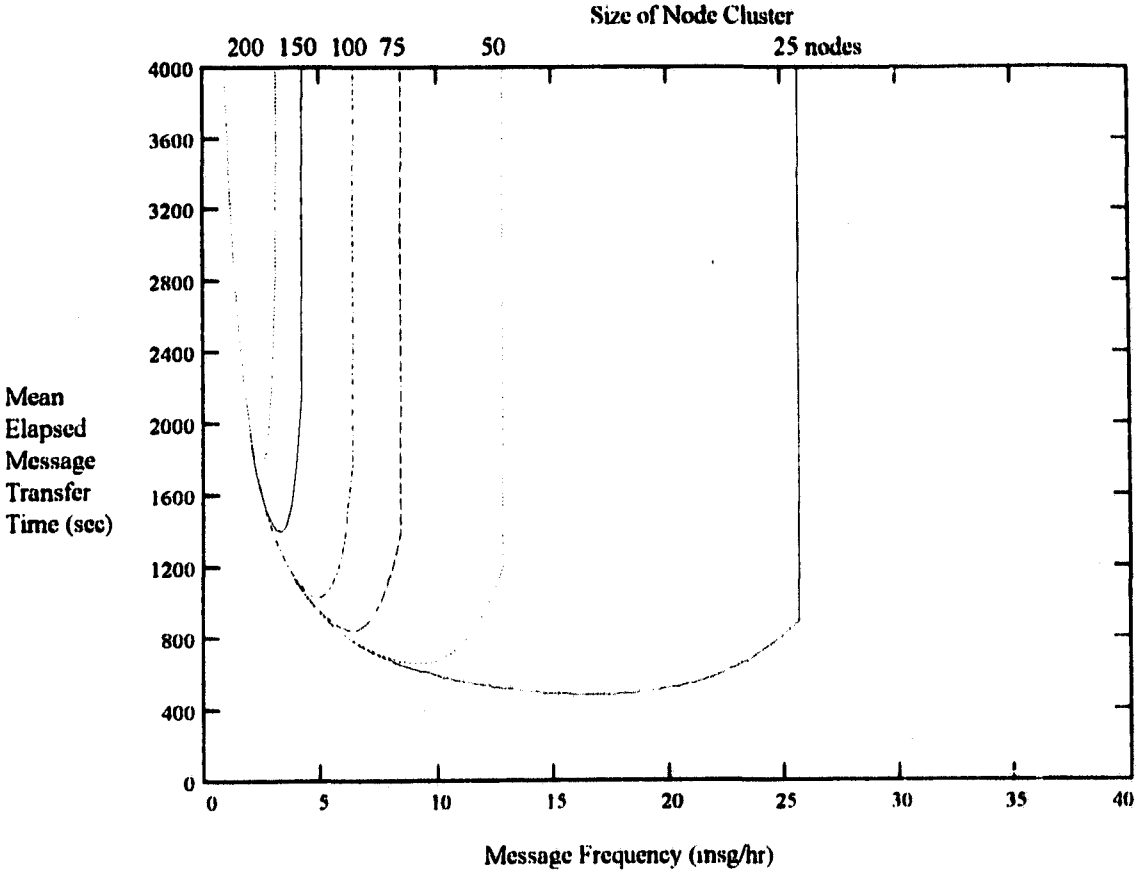


Figure 32. High Priority Message Transfer Times for Analytic Simulation of a Ten-Server Star Cluster



Figures 33, 34 and 35 show the minimum number of gateway servers needed for different sizes of star clusters and different scheduling modes. Figure 33 is based on free-running connections (as are Figures 28 through 32). For each message frequency, 5, 10, 15 or 20, the characteristic curves, like those shown in Figure 32, are analyzed to determine which is the maximum cluster size with a mean message transfer time below one hour. For example, inspection of Figure 32 determines that for ten servers with a message frequency of five, the largest node cluster size is between 100 and 150. Figure 33 can be used to locate this point at approximately 120 nodes.

Figures 34 and 35 are based on a connection schedule. Each node contacts the gateway according to a round robin schedule with fixed time slots. The size of the slot is determined by the mean connection time, τ_{conn}, plus a slack period of five minutes. Figure 34 yields roughly the same mean end-to-end message transfer time as does Figure 33. By using connection scheduling, Figure 34 indicates that gateway utilization has improved substantially for high message frequencies. For example, using scheduling, 100 servers can support 580 nodes in comparison to 300 nodes using unscheduled service at a creation rate of 20 messages per hour. Figure 35 demonstrates a further reduction in server requirements based on an increased mean transfer time of four hours.

Unfortunately, as the gateway utilization increases, so too does the likelihood of slot over-run. For example, if 100 servers are used to support 700 nodes at a frequency of 20 messages per hour given a mean transfer time of four hours, the gateway utilization is 92 percent (see Figure 36). The mean connect time is approximately 61 minutes. The scheduled time slot is then 66 minutes. If the connection times are Poisson-distributed, then only about 75 percent of the connections are of short enough duration to fit within their assigned slots. Assuming that the slot regimen is enforced, then only a portion of the mail bags are completely exchanged. Messages remaining in the mail bag are deferred for another four hours.

Figure 33. Minimum Number of Gateway Servers Required for Delivery of High-Priority Messages Within a Mean End-to-End Transfer Time of One Hour in a Star Cluster

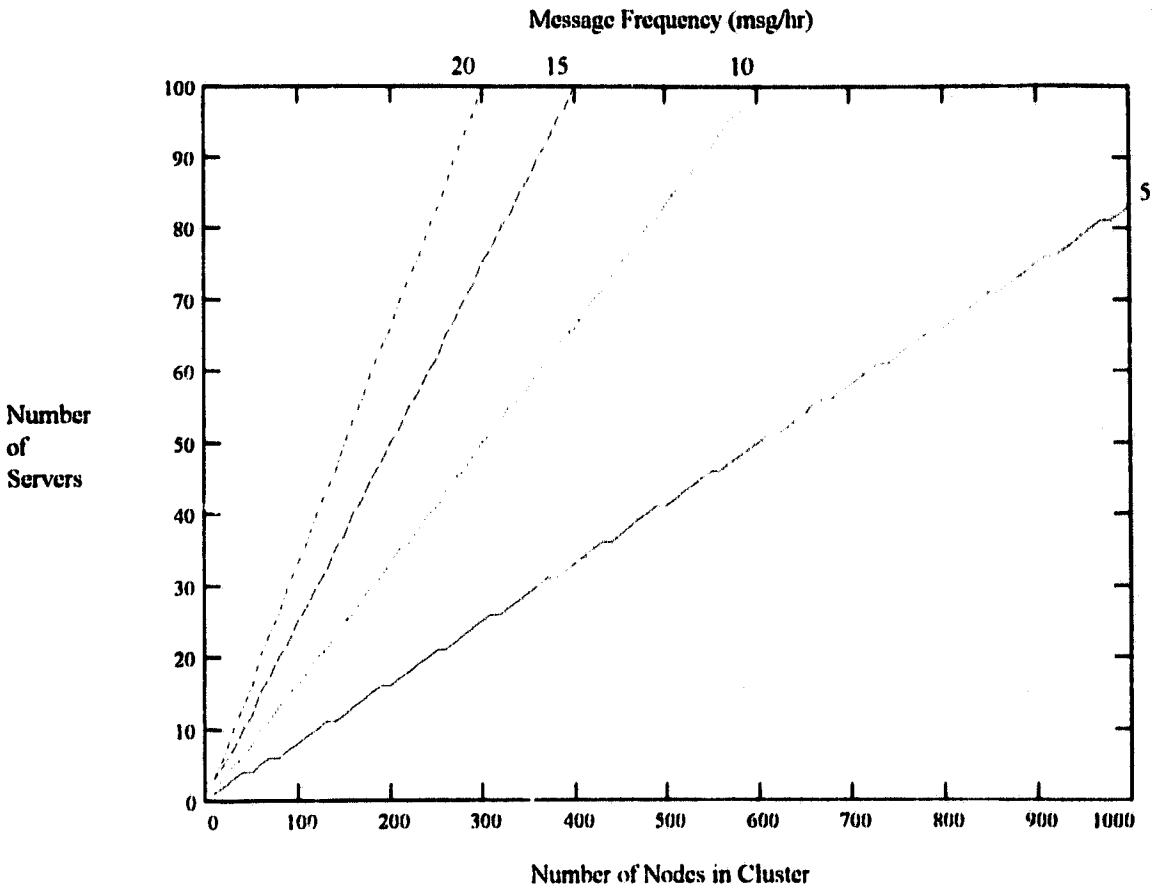


Figure 34. Minimum Number of Gateway Servers Required for Delivery of High-Priority Messages Within a Mean End-to-End Transfer Time of One Hour in a Star Cluster Using Connection Scheduling

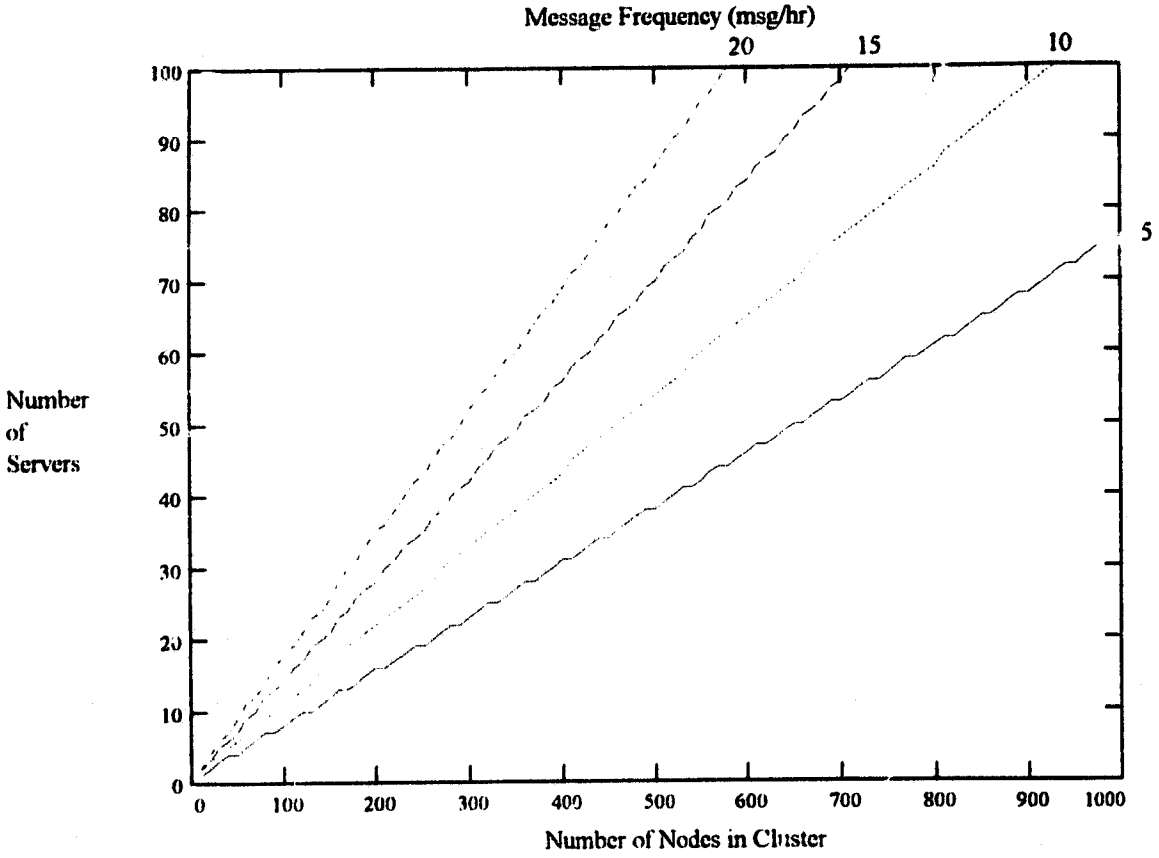


Figure 35. Minimum Number of Gateway Servers Required for Delivery of High-Priority Messages Within a Mean End-to-End Transfer Time of Four Hours in a Star Cluster Using Connection Scheduling

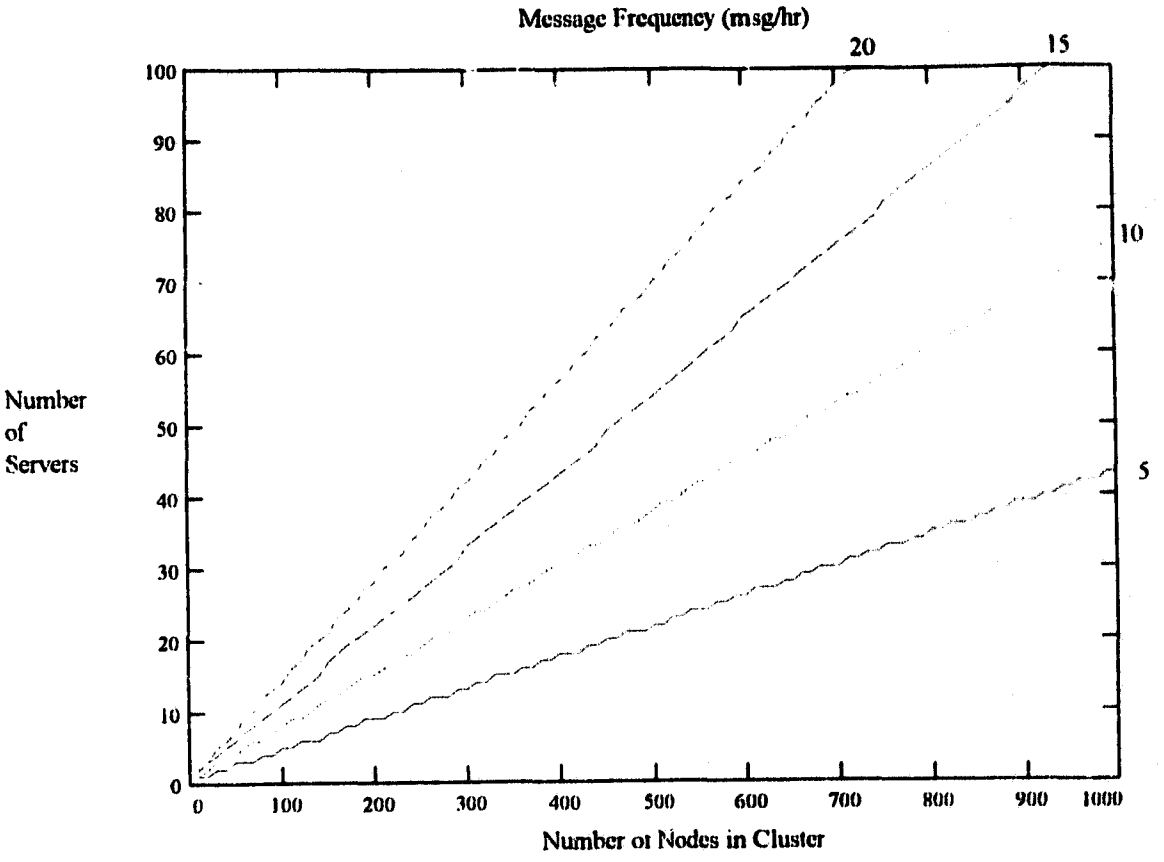
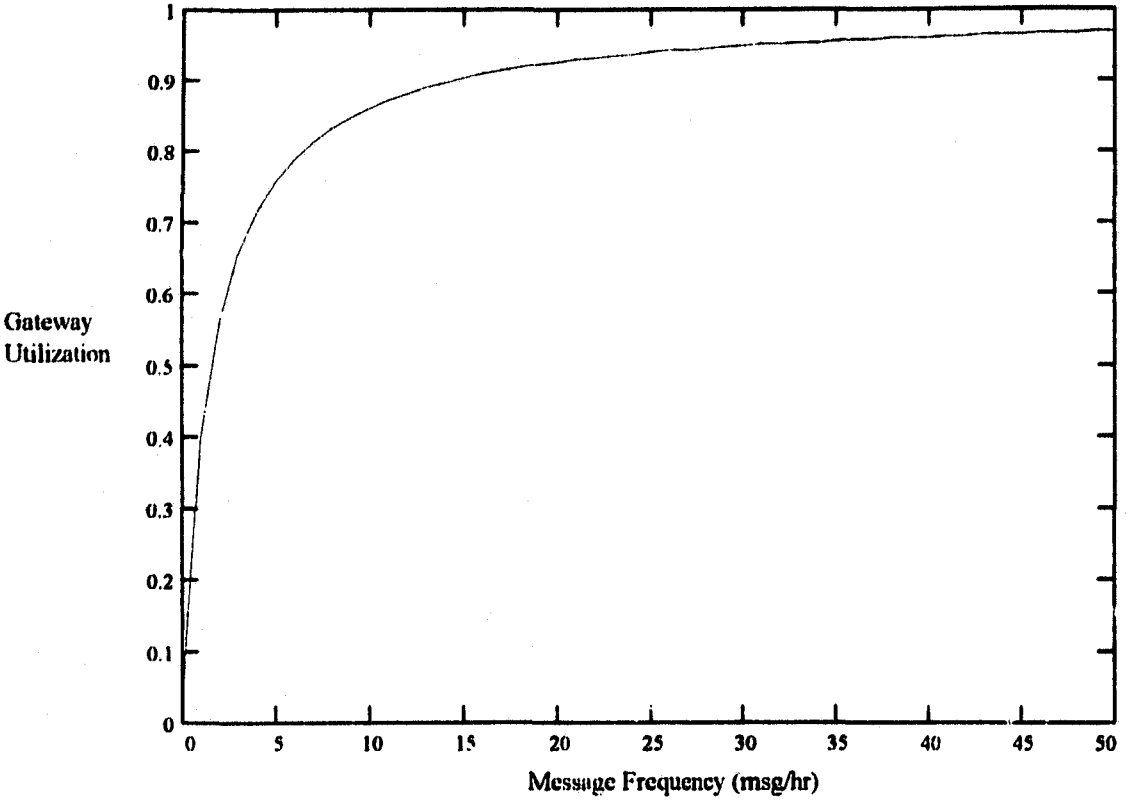


Figure 36. Optimum Utilization of Gateway Servers for Delivery of High-Priority Messages Within a Mean End-to-End Transfer Time of Four Hours in a Star Cluster Using Connection Scheduling with a Five Minute Interconnection Idle Period



5.4 Conclusion

Both the discrete-event and analytic models appear to closely match the performance of the real *Health Link* subnet. The discrete-event model has a high degree of face validity and is capable of simulating not only the exchange of messages but also their acknowledgments. The discrete-event model, however, is prone to long execution times. The computation times for points shown in Figures 14 through 21 range from about ten minutes to several hours on an Intel 80486DX33.

The analytic models are based on a stronger set of assumptions than are the discrete-event models. Nonetheless, the resulting performance curves are fairly predictive of those generated by the discrete-event model. The analytic models appear to predict behaviour with least accuracy at operating extremes such as when a cluster is heavily loaded. It takes only a few seconds to compute the characteristic message transfer curves for an entire figure. A short-coming of the analytic models is that, unlike the discrete-event models, they are able to calculate only a small set of statistics for each configuration.

The mesh topology appears to be better suited for message exchange in a cluster than is the star topology. The size of the mesh cluster has no direct effect on the resources required by the gateway. Furthermore, regardless of the number of nodes in a mesh cluster, the mean message transfer time is relatively independent of the message creation rate up to a limiting threshold of about ten messages per hour.

For a star cluster, the number of servers needed for the gateway can be optimized by using connection scheduling. However, this pre-determines the minimum mean message transfer time without limiting the maximum message transfer time for any given message.

Priority delays have little or no beneficial impact on the performance of a star cluster. A priority delay may affect the amount of time a message is held at its source but not the amount of time the message is stored at the gateway. The gateway is unable to initiate connections to a destination node in order to expedite high priority messages.

This steady state study has revealed several behavioural characteristics of the *Health Link* implementation which should be considered when the prototype is revised for production:

1) The message processing consumes almost the same amount of CPU resource as does the communications. Although data transmission is not a heavy load, communication causes the processing cycle to be halted. Throughput gains can be realized if unnecessary modem control scripts can be removed or accelerated. For example, an answering node has no cause to set its modem to *auto-answer* upon disconnect (a five second operation) since it is already set to auto-answer. (This inefficiency is removed from the simulations described in Chapter 6.) Similarly, rapid detection of a no-answer condition or a line idle condition can reduce CPU utilization.

2) Message processing can be accelerated by removing or reducing the interstate wait time. The delay accumulated for each message processed is eight seconds, which ranges from 16 percent and 23 percent of the time which the CPU is involved in message preparation depending on the CPU load.

3) The communication rates can be improved by changing the link layer protocol to manage a wider sliding window. Under the current implementation the transmission time of a 1,000-character message at 2,400 bps is measured at 13 seconds in half-duplex mode and 30 seconds in full-duplex mode. Under ideal conditions, this could be reduced to about eight seconds for both modes.

Although the models are designed to predict the performance of a *Health Link* subnet, they also predict the performance of any telephone-based network using the same topologies, provided that the constants are adjusted appropriately. It is believed that only radical changes made to the modeling constants would invalidate the general observations and conclusions drawn by this chapter.

6 Simulation of a Health Care Telecommunication Network

6.1 Introduction

There are a number of exploratory projects investigating the use of telecommunications in health care. In the past, projects have typically been focused on a single application or subscribed to by a small number of users. For example, the University of Nebraska Medical Center has established a network, Synapse, for rural physicians at 105 sites using the PSTN [64]. In a second project, COPA, based in The Netherlands, 27 physicians and two regional hospitals are participating in an electronic mail network by exchanging laboratory reports and hospital admission and discharge reports [48, 49]. Smaller efforts, such as these, are leading to more comprehensive implementations. Current initiatives such as the AIM programme are targeted at the European health care sector in general for a wide range of applications [115].

Many researchers seem to assume that the upward scaling of small telecommunication networks presents little or no implementation concern. Although telephone companies have undoubtedly performed extensive analyses on scaling problems associated with various types of telecommunication systems, there appear to be few publications which specifically address the health care sector. Designs of the larger systems often presume that third-party communication services are adequate to the task of supporting the application network [116]. This presumption may be appropriate where operating costs are not a major concern. However, it may not be appropriate where the telecommunication infrastructure must be implemented specifically for telemedicine.

Exploratory projects such as COPA and 3I, use a number of criteria for evaluation of electronic exchange of medical data [2, 45, 48, 49]. These evaluations focus on user satisfaction, network performance, accuracy of data transmission, impact on workload, access to information and operating costs. There are few, if any, evaluations which analyze operating cost with respect to network size, topology and performance.

This chapter presents a preliminary performance analysis for a fictitious health care network for the Province of Saskatchewan, Canada. It makes use of the PSTN in conjunction with a PSN. The behaviour of nodes in the network is modeled after

Health Link. A discrete event simulation program has been developed expressly for analyzing this type of large, wide area network.

Saskatchewan has been chosen as the focus for the simulation study for several reasons. First, Saskatchewan is ranked sixth in its population with respect to the other provinces. Accordingly, it has significantly fewer health care providers than do the larger provinces of Ontario, Quebec and British Columbia. Because there are fewer providers, the simulation model is less complicated. Second, Saskatchewan has a large proportion of its population residing in rural areas. A wide area telephone network is most suited for this context because other network technologies, such as Integrated Services Distributed Network (ISDN), are unlikely to be available in the rural areas. Third, Saskatchewan has been evaluating different health care networks over the past several years. It already possesses a pharmacare network which is likely to be retired shortly in favour of a general purpose health care network. Finally, the health care system of Saskatchewan is well-known in an international forum because of its early entry into universal health insurance in 1962 which resulted in the first doctors' strike in Canada.

The simulation studies presented in this chapter support the hypothesis that such wide area health care networks are practical. The studies are used to produce rough estimates of performance and cost. Two network topologies are investigated: one which makes use of peer-to-peer communication (mesh topology) and one which uses client-server communication within the subnets (star topology).

The steady state simulations presented in Chapter 5 describe the behaviour of a cluster or subnet of nodes under different loads. The conclusions resulting from these simulations are useful in exposing inefficiencies in software implementation and determining critical performance thresholds. For example, it is significant that there is a threshold frequency of approximately ten, 1,000 character messages per hour per node in a heavily populated mesh cluster. A survey of message types, sizes and frequencies conducted at two medical clinics indicates that, over a 24-hour period, these thresholds are not restrictive. (A physician typically sends or receives approximately 50 text-based messages per day of sizes less than 1,000 characters per message.) However, the steady state simulations cannot measure the effect of traffic between subnets or surges in daily activity. A large scale health care network is subject

to both of these conditions. Simulation makes it possible to obtain more accurate estimates of connection times and port utilizations which affect the amount of capital investment and operating costs.

This chapter is organized as follows. The next section presents a brief discussion of one of the few published efforts to estimate costs of a health care network based on its topology. This is followed by a description of the modeling parameters used and the modeling assumptions made. Section 6.4 describes the simulation tool and Section 6.5 presents the results of the preliminary evaluation of the two topologies developed. Conclusions drawn from the analysis are given in the final section.

6.2 Background

One of the few published studies which relates cost to scaling in health care networks is that performed by van Lierop et al. of the GEIN project, The Netherlands [47]. They have estimated the cost of implementing and operating an EDI network in the region of Breda consisting of three hospitals, 180 specialists and 165 GP's. They estimate that one million messages would be exchanged in the network annually. Costs are calculated based on the communication cost of a limited set of EDI transactions and the support organization needed to administer the network.

The GEIN study concludes that the region of Breda is too small to fiscally support an EDI infrastructure and recommends that the region be combined with other regions to increase utilization. Further studies are currently being undertaken by GEIN in this direction.

The GEIN analysis is not applicable to North America where population distribution, patterns of information distribution, and administration of health care are substantially different.

6.3 Modeling Parameters

The simulation model for Saskatchewan uses parameters drawn from a variety of sources describing the distribution of health care practitioners and agencies [4, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129]. Message statistics for

GPs and medical laboratories are drawn from data collected by the Health Link project during a field test conducted at two medical clinics.

Saskatchewan can be viewed as consisting of eight regions as shown in Table 16 and Figure 37. The numbers in the table reflect approximate distribution of health care providers. The table is not comprehensive but does include the majority of primary care providers. To add the less important participants would significantly increase the complexity of the analysis without increasing its predictive accuracy. Some of its omissions are that:

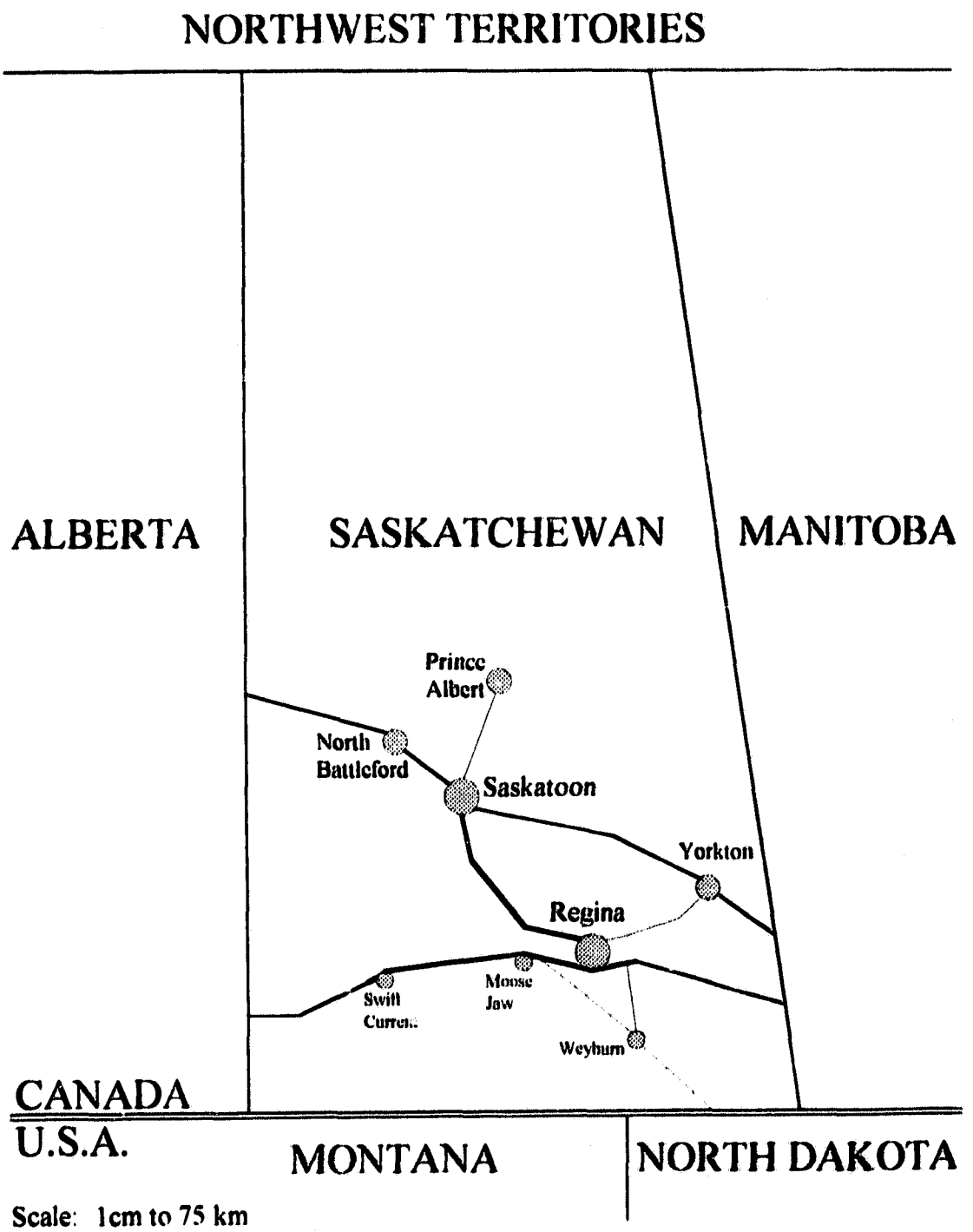
1) It shows only acute care hospitals having 50 beds or more although there are a large number of smaller hospitals distributed throughout the Province.

2) Medical laboratories are under-represented. The major medical laboratories are in Saskatoon and Regina but there are other associated branch laboratories in Moose Jaw, Prince Albert and Yorkton.

Table 16. Distribution of Health Care Providers Saskatchewan by Region

Region	GP's	Specialists	Municipal Hospitals (beds)	Regional Hospitals (beds)	Private Medical Labs
Moose Jaw	48	14	0	2 (332)	0
North Battleford	89	8	0	4 (399)	0
Prince Albert	113	27	0	5 (757)	0
Regina	245	185	4 (1439)	0	2
Saskatoon	338	299	4 (1324)	0	2
Swift Current	40	10	0	1 (164)	0
Weyburn	40	1	0	2 (197)	0
Yorkton	80	8	0	6 (633)	0
Total	993	553	8 (2763)	20 (2482)	4

Figure 37. A Map of Saskatchewan



Scale: 1cm to 75 km

3) It does not show two Province-wide facilities, the Medical Care Insurance Commission and the Provincial Laboratory, which are both located in Regina.

4) It shows only those facilities directly concerned with delivery of orthodox medical care. A more complete inventory would include approximately 381 dentists, 96 optometrists, 907 pharmacists and a wide range of complementary and alternative practitioners.

5) It shows no private medical insurance companies or medical suppliers.

Table 17 gives an estimate of the population catchment of each region with a breakout of urban and rural counts. Inspection of Tables 16 and 17 indicates that a simple simulation model can be constructed by classifying Moose Jaw, Swift Current and Weyburn as regional centres. North Battleford, Prince Albert and Yorkton areas can be similarly classified but they are roughly twice the size of Moose Jaw, Swift Current and Weyburn. Although Regina and Saskatoon are both municipalities of approximately the same size, they are classified uniquely because only Regina has Government facilities. The result, shown in Table 18, is a simplified model which is used for the simulation studies.

Table 17. Saskatchewan 1991 Population Counts by Region

Region	Urban	Rural	Total
Moose Jaw	39,588	18,597	58,185
North Battleford	46,871	81,176	128,047
Prince Albert	60,265	75,917	136,182
Regina	185,885	32,727	218,612
Saskatoon	203,430	31,838	235,268
Swift Current	22,321	33,468	55,789
Weyburn	23,288	33,079	56,367
Yorkton	41,749	58,730	100,479
Total	623,397	365,532	988,929

Table 18. The Saskatchewan Model: Health Care Providers by Region

Region	Subnet Designation	Subnet Scaling Factor	GP's	Spe-cialists	Muni-cipal Hospitals	Regional Hospitals	Private Medical Labs
Moose Jaw	REGIONAL	1	46	8	0	2	0
North Battleford	REGIONAL	2	92	16	0	4	0
Prince Albert	REGIONAL	2	92	16	0	4	0
Regina	REGINA	1	245	185	4	0	2
Saskatoon	SASKTOON	1	338	299	4	0	2
Swift Current	REGIONAL	1	46	8	0	2	0
Weyburn	REGIONAL	1	46	8	0	2	0
Yorkton	REGIONAL	2	92	16	0	4	0
		Total	997	556	8	18	4

The simulation model is based on a restricted set of node classes: 1) GP's, 2) specialists, 3) regional hospitals, 4) municipal hospitals, 5) private medical laboratories, 6) the Provincial medical laboratory, and 7) the Medical Care Insurance Commission. Message traffic among these nodes is based on a survey of two Victoria medical clinics. Unlike traffic to or from physicians, the specification for hospital to hospital messaging is merely a reasonable guess. Some types of messages have been omitted from the model either because they are more appropriately communicated using an alternative method such as the telephone or facsimile, or because they are made to or from a class of nodes not represented in the model. Table 19 is a summary of the message traffic among the nodes. (Refer to Appendices E and F for greater detail regarding the distribution of the messages.)

The assumptions made in the construction of the Saskatchewan model include:

1) All physicians act as independent practitioners. Clinics and partnerships do not significantly alter the behaviour of the system.

2) The traffic requirements for specialists is similar to that for GP's. The traffic requirements for GP's is similar to that measured in the Victoria survey.

Table 19. The Saskatchewan Model: Estimated Message Traffic

Type of Message	Source	Destination	Messages Per Day	Message Size (chars)
Medical lab test report	medical lab	GP	20/GP	175
Provincial Lab test report	Prov. Lab	GP	3/GP	175
Emergency report	hospital	GP	2/GP	200
X-ray report	hospital	GP	2/GP	300
Lab test report	hospital	GP	2/GP	250
Operating room (OR) report	hospital	GP	2/GP	1000
OR booking	GP	hospital	2/GP	175
OR booking confirmation	hospital	GP	2/GP	175
Outpatient clinics	GP	hospital	2/GP	175
Specialist referral	GP	specialist	3/GP	1000
Specialist referral	specialist	specialist	3/source	1000
Specialist report	specialist	GP	5/specialist	200
Health insurance claim	GP	Government	1/GP	2000
Insurance claim confirmation	Government	GP	1/GP	100
Medical record transfer	GP	GP	3/GP	1000
Medical record transfer	hospital	hospital	10/destination	500

3) The number of laboratory test reports made by the hospital is small (two reports per physician) in comparison with the number of reports distributed by the private medical laboratories.

4) The Medical Care Insurance Commission, which pays physicians for patient care services, is the only payer. (In fact there are three payers: two are non-profit intermediaries acting for the Medical Care Insurance Commission [32].)

5) Small hospitals, small laboratories and other unspecified health care providers do not significantly alter the behaviour of the system.

6) All nodes in the system are available throughout the 24-hour day.

7) Messages are created at a stochastically uniform rate over four-hour, nine-hour, or 24-hour periods depending on the schedule chosen.

8) Messages are text documents ranging in length from 100 to 2,000 characters depending on their source. Message lengths vary according to the gamma distribution described in Section 6.4.

9) There is no need for distribution mailings or message forwarding.

10) Only known, existing applications require electronic data exchange.

11) There are no alternative, parallel networks which share the projected load with the model network.

12) Network management messages are incidental and constitute no additional network load.

6.4 Discrete Event Simulation Program

Simulations of large telecommunication networks are difficult to conduct because 1) they usually require a greater number and wider variety of configuration parameters, 2) they are more difficult to tune because of the complex relationships between input parameters and measurable output, and 3) they result in simulation runs having large memory and large execution time requirements. Results from these simulations are difficult to analyze because 1) an enormous volume of data is produced and 2) there is a wide range of potentially interesting dependent variables to study. A discrete event simulation program has been developed which attempts to minimize these problems through classification and simplification.

The steady state simulation studies in Chapter 5 exploit network symmetry to extrapolate the behaviour of one node to a cluster of nodes in one subnet. This type of simulation is appropriate only if all nodes exhibit the same behaviour and only if a steady state analysis is being performed. The studies in Chapter 5 cannot be extended to different classes of nodes and multiple subnets because of the symmetry requirement.

This program is implemented in Microsoft Visual C++. (Even though the steady state simulation program described in Chapter 5 is implemented in GPSS/H 386, the size of the Saskatchewan simulation model makes use of GPSS inappropriate.) The program is designed to run under Microsoft Windows in *386 enhanced mode* which makes use of virtual memory thereby permitting the model to use huge arrays which

require storage in excess of the primary memory. (The Saskatchewan model requires approximately two megabytes of storage for its arrays.) Furthermore, C++ has a much more convenient syntax than GPSS and Visual C++ provides more sophisticated program development tools which include a good symbolic debugger. Unfortunately, programming of simulations in C++ requires the development of queue management and data collection routines which are built into GPSS.

The program is capable of simulating networks in excess of 10,000 nodes. To reduce the complexity of the configuration, the program structures a network as a collection of subnets belonging to a set of subnet classes. Each subnet is in turn a collection of nodes belonging to a set of node classes. Individual subnets and nodes are instances of given subnet and node classes respectively. Each instance of a class inherits the characteristics of its class. This object-oriented approach makes it possible to model a large number of similar nodes and subnets with a minimum of effort. Figures 38 and 39 describe the configuration of a node class and a subnet class respectively. These figures are extracted from the configuration report produced by the simulation program. (See Appendices E and F for complete listings of the two topologies studied.)

The message and call schedules referenced in Figures 38 and 39 are tables which define time slots with respective frequencies. A message schedule determines if and when one or more messages are created. A message is held until either a call from its destination is accepted or it is scheduled to be sent as determined by the appropriate call schedule. Call schedules can also be used to initiate polls even when no messages are waiting to be sent. Each message and call schedule is given a *precision* which determines the random spread of events within a time slot. The precision allows the same schedules to be used for many nodes to introduce asynchronous events. The MSGDAY10 message schedule causes ten messages to be generated in evenly spaced time slots of 54 minutes each starting at 8:30 and ending at 17:30. The ADHOC call schedule has no time slots defined and, consequently, messages are scheduled for delivery as soon as they become available for sending.

Figure 38. Regional Hospital Node Class

Node class:	REGHOSP
Description:	Regional Hospital (125 beds)
Class enabled:	Yes
Accept calls:	Yes
Call schedule to gateway:	ADHOC
Mandatory gateway routing:	No

Destination Node	Message Schedule	Call Schedule	Freq Mult	# Nodes Intra-net	# Nodes Inter-net	Msg Size	Priority	Ack Reqd
GP	MSGDAY10	ADHOC	23	23	0	390	Med	Yes
MAINHOSP	MSGDAY10	ADHOC	8	0	8	500	Med	Yes
REGHOSP	MSGDAY10	ADHOC	3	3	0	500	Med	Yes
SPEC	MSGDAY10	ADHOC	4	4	0	390	Med	Yes

Node and subnet scaling are used to make classes more flexible. Node scaling increases the number of messages generated and the number of destinations at each affected node instance. Subnet scaling permits the number of nodes in a subnet to be increased without altering the attributes of its constituent nodes. Message creation frequencies can also be adjusted through a frequency multiplier.

Figure 39. Regional Subnet Class

Subnet class:	REGIONAL
Description:	Regional Centre (50,000)
Class enabled:	Yes

Node Class	# of Nodes	Node Scaling	Node # Ports	Gateway Call Schedule to Node
GP	46	1	1	ADHOC
REGHOSP	2	1	2	ADHOC
SPEC	8	1	1	ADHOC

Routes are established automatically by the program. Every node within a subnet has a bi-directional route to the gateway node in the subnet. In addition, if a particular node class has several destination node classes, an instance of that class is assigned unidirectional routes to instances of the appropriate destination classes. If, as is often the case, a destination class lists the particular node class among its destination classes, then a reverse route is established. If a node class does not accept incoming calls or if gateway routing is mandatory, messages are directed automatically to the subnet gateway, ignoring possible point-to-point (intranet) connections within the subnet. Three-hop paths are used to communicate between two nodes in different subnets (internet communication): one hop is from the source node to its subnet gateway, one hop is from the source subnet to the destination subnet, and one hop is from the destination subnet gateway to the destination node. If the automatic route assignment is foiled by specifying more destinations than are available, the node is scaled down accordingly.

The average message length is specified for each destination class within a given node class. The user can choose either to send messages with a fixed length (no distribution) or to send messages with a length scaled in accordance with a gamma distribution. The gamma distribution used is given in Equation 31. The distribution function shown has a mean of 1.0 and a standard deviation of 0.5. If a distribution is selected for a simulation run, it is applied to all messages.

$$f(x) = \frac{1}{\Gamma(2)}(2x)e^{-2x}, \text{ for } x \geq 0 \qquad \text{Eq. 31}$$

If a message has no explicit delivery time determined by a call schedule, the priority is used to determine the holding time of the message. For example, if the priority delay for medium priority messages is set to five minutes, then a medium priority message is held for five minutes before a connection is initiated by that message. If a connection to the same destination is established during the intervening time for some other reason, then the message is prematurely transmitted.

The program permits messages to initiate acknowledgment messages upon delivery. Unlike messages whose size and priority can be configured, acknowledgment messages are always set to zero length with low priority.

The number of communication ports are specified for each gateway node and each node class within a given subnet class. The number of ports are used to determine 1) the number of concurrent communications links and 2) the processing power of a node's CPU. CPU power, which determines the length of time needed to prepare a message for sending and delivery, is directly proportional to the number of idle ports. Thus, if a node is configured with three ports and one is active, the node has a CPU power factor of two. The CPU power factor plays no role at a gateway node because no message preparation is performed there.

The program assumes a daily cycle of operation. Message and call schedules are built for a 24 hour day. A simulation can run for up to 48 simulation hours to permit a one day for initialization or *warm-up* before statistics are collected.

Each simulation run requires the user to enter a seed for the random number generator. The random number generator is used to establish routes and to trigger message creation and call schedule times. Replication runs are made possible by changing the seed. Each separately seeded replication constructs a different run time model which has the same set of configuration constraints.

The program is able to simulate approximately 20 messages per second on a 80486DX33 PC. This rate can decrease dramatically if a node in the model becomes backlogged. The program is able to hold on average more than 25 messages per node in a global storage pool. For the Saskatchewan model, although the program allocates storage for up to 41,825 messages at any instant, it usually requires storage for fewer than 20,000 messages.

The simulation program is built around several significant assumptions:

1) Message processing is sequential at any given end-user node.

2) The CPU load resulting from servicing communication interrupts has no significant impact on CPU availability.

3) The effect of CPU power on message processing time is determined only when processing starts for a specific message.

4) Message processing is interrupted if all communication ports are busy.

5) Message processing does not affect concurrent communication activity.

6) Message processing is single-stage and includes not only addressing, compression, encryption and signature generation/verification but message file storage and retrieval.

7) Although message processing is usually FCFS, order is not important if a message should be requeued as a result of a heavy backlog.

8) The effects of data compression, encryption and character encoding offset one another so that a message neither grows nor shrinks in length as a result of these processing steps.

9) Message processing has a fixed overhead, regardless of message length.

10) Communication channels do not permit multiplexing.

11) If a communication link is idle when a message is about to be sent, the message is sent at the half-duplex rate for its entire length.

12) The order in which messages are transmitted once a connection is established is unimportant.

13) The effective data transmission rate is not affected by poor line conditions unless the model builder explicitly reduces the rate.

14) Messages at gateways experience no delays except for priority delays. Gateways experience no processor loading.

15) All end-user nodes have identical platforms except for the number of communications ports, with a corresponding accommodation in CPU power, that each supports. Gateways operate under similar conditions.

16) Communication links between nodes exhibit uniformly the same behaviour. Communication links between gateways exhibit uniformly the same behaviour.

17) Delays incurred by searching queues, reorganizing data bases, reordering lists and performing directory searches are negligible.

18) Three possible outcomes can result from a connection request: call accept, no answer and destination busy.

19) Any communication port at a multi-port node has an equal likelihood of accepting or originating any call.

6.5 Simulation Results

The simulation program has been used to collect statistics for two network topologies. The first, a mesh topology, relies on peer-to-peer communication within each subnet. The second, a star topology, relies on client-server communication within each subnet. The timing parameters are calibrated to *Health Link* timings observed in a battery of actual tests taken in a four-node mesh topology.

6.5.1 Calibration

The data presented in Section 5.2.4 (see Table 12) for a four-node cluster are used to calibrate the timing parameters in the simulations described in this chapter. A series of replicated simulations were conducted for each nominal message frequency. A comparison of mean message transfer times between the simulations and the *real*, or actual, test runs are shown in Table 20 and Figure 40. Each simulation has a warm-up interval during which greater than 60 messages are processed without collecting their statistics. Statistics are gathered for between 900 to 3,200 messages at every replication trial depending on the nominal message frequency and the seed supplied to the random generator.

The timing parameters chosen for these simulations are shown in Table 21. Some of the parameters shown are combinations of parameters used in the steady state simulations described in Chapter 5. For example, the *Modem reset time* is composed of the disconnect time (seven seconds), the reset time (ten seconds), a wait period (five seconds), and the time required to set the modem into a listening state (five seconds). Similarly, the *Net message processing time* is a sum of all the message processing stages (eight seconds) as well as accumulated interstate wait times (six seconds) as described in Section 4.7. Seven additional seconds, which are allocated to help calibrate the simulation model, are partially attributed to processing cycle interruptions and to message file retrieval and storage which are not being simulated.

Figure 40. Calibration Statistics for a Four-Node Mesh Cluster

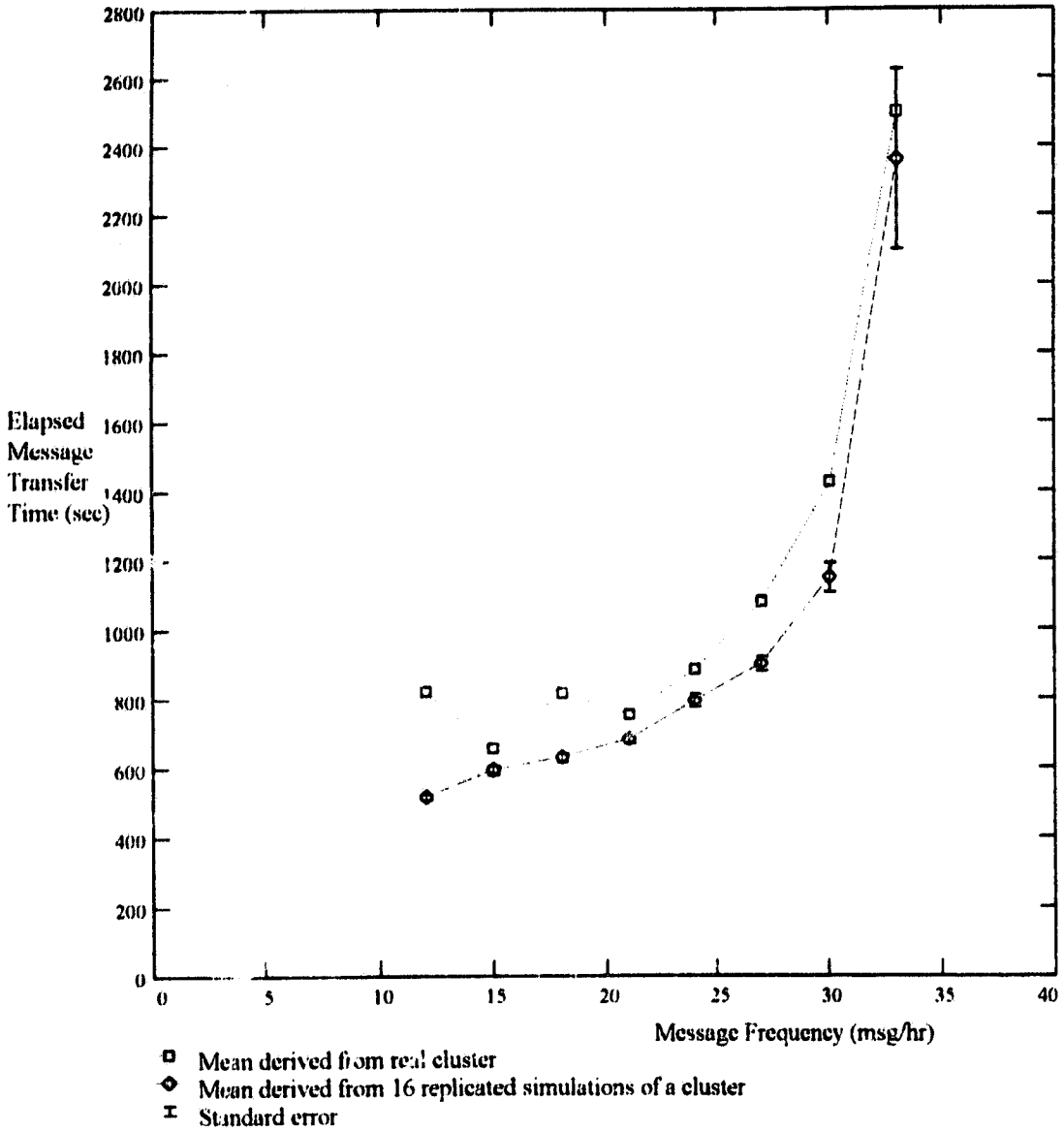


Table 20. Calibration Statistics for a Four Node Mesh Cluster

Nominal Message Frequency (msg/hr)	Real			Simulated		
	Sample Size N	Mean	Standard Deviation	Number of Replications N	Mean	Standard Deviation
12	358	820	662	16	520	20
15	198	659	314	16	594	39
18	234	819	544	16	629	37
21	255	757	445	16	684	42
24	1164	885	501	16	795	67
27	658	1081	633	16	903	77
30	721	1426	891	16	1152	162
33	372	2497	2338	16	2362	1043

Timing parameters used for the peer-to-peer and client-server simulation studies are also shown in Table 21. Although the *Modem reset time* is experienced by both parties to a connection, the *Relisten time* is encountered by only the call originator. In the calibration study, both parties experience both delays and hence the two delays are combined as a single *Modem reset time*. For a full-scale, non-prototypical network, the relisten time would be associated with only the call originator.

Other parameters used in the simulation studies have also been changed. The priority delays have been adjusted to better suit the topologies studied. The *Preparation cycle wait* is a delay which is initiated whenever a connection request interrupts message processing. Although this parameter is currently set to five seconds in the *Health Link* software to facilitate interactive entry, it is thought to be unnecessarily long and has been reduced to one second for the purposes of the simulation studies.

Table 21. Simulation Timing Parameters

Timing Parameter	Calibration Values	Peer-to-Peer Values	Client-Server Values
Dial time (sec)	15	15	15
Ring and connect time (sec)	12	12	12
Relisten time (sec)	0	14	14
Modem reset time (sec)	24	10	10
Line idle wait (sec)	60	60	60
Full-duplex effective data rate (bps)	362	362	362
Half-duplex effective data rate (bps)	770	770	770
Minimum connection retry wait (sec)	180	180	180
Transfer protocol overhead (sec/msg)	2	2	2
Low priority delay (sec)	N/A	14,400	N/A
Medium priority delay (sec)	300	180	N/A
High priority delay (sec)	N/A	N/A	N/A
Net message processing time (sec/msg)	21	21	21
Preparation cycle wait (sec)	5	1	1

6.5.2 Peer-to-Peer Model

Figure 41 is a diagram of part of the peer-to-peer model analyzed. It shows a portion of the REGINA subnet and a portion of one REGIONAL subnet. Routes drawn in the diagram are mostly bi-directional. For reasons of space and clarity only a small percentage of the GP's and specialists are shown in the two subnets illustrated in the figure.

Although peer-to-peer message delivery is achieved wherever the sender and receiver are in the same subnet, different methods of delivery are used. In most cases, including hospital to GP and GP to specialist, messages are transferred after a medium priority delay using *ad hoc* scheduling (no pre-established call schedule). In order to lessen the load on the laboratory nodes, polling is used to pick up laboratory test reports by other nodes in the same subnet. Each GP and specialist polls the medical laboratories and Provincial Laboratory for pending messages every four hours. The

mean delivery delay measured in the simulation for messages sent from the laboratories to the physicians is approximately 2.25 hours. This delivery delay differs from the expected mean of two hours (half of the polling interval) because 1) there is statistical variation, 2) the message processing times increase the end-to-end delivery time and 3) about 30 percent of the messages are not subject to polling because they are internet and, consequently, experience smaller end-to-end delivery delays.

Priority delays are applied to all nodes including the gateway nodes. The delays have been adjusted for medium and low priority messages. Only acknowledgment messages are sent using low priority. (Unless otherwise stated, acknowledgment messages are excluded from message statistics.) Decreasing the medium priority from five minutes to three minutes decreases the transfer time for data messages by approximately two minutes. Non-zero priority delays reduce the number of connection requests on heavily traveled routes such as gateway to gateway and laboratory to gateway, thereby reducing port contention along these routes. Use of priority delays at the gateway nodes decreases the number of internet connection requests by a factor of 4.5.

The low priority delay is set to four hours; the same period as the polling period used to retrieve laboratory test results. This inhibits connection requests made by acknowledgment messages. Under these conditions, acknowledgment messages are transferred once a connection has been established by either a poll or a data message.

Table 22 gives a summary of statistics for the peer-to-peer model simulated over 24 hours after a 24-hour warm-up period. The standard errors shown are computed from ten replications of the same configuration with different seeds for the random number generator. A few pertinent observations should be made:

1) In both the peer-to-peer and the client-server model, messages sent and received by the GP's and specialists constitute approximately 75 percent of the 77,000 data messages (3c) circulated in one 24-hour period. For the peer-to-peer model, only 23 percent or 17,956 data messages (1b) are multi-hop, requiring the assistance of the gateway nodes.

2) The number of unidirectional paths refers to the number of unique end-to-end paths which are constructed in the model. The 26,636 paths (3b) reported are composed of 20,620 unidirectional links.

Figure 41. The Saskatchewan Peer-to-Peer Model: A Partial Topology Diagram

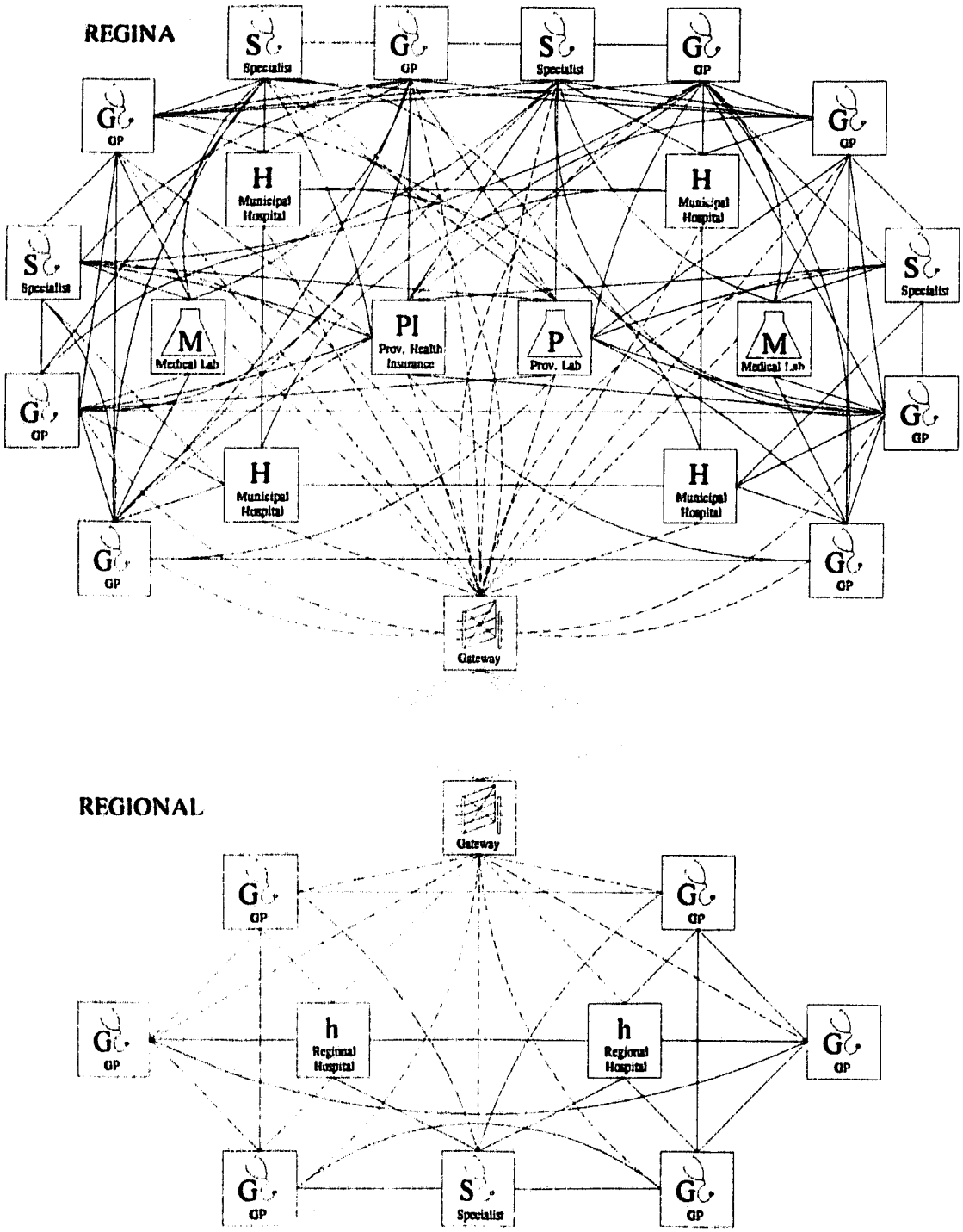


Table 22. The Saskatchewan Peer-to-Peer Model: Summary of Simulation Statistics

Gateway Internet Routes (all gateway nodes)		
1a	Number of nodes	8
1b	Number of data messages transmitted	17,956±338
1c	Number of connections originated	2,806±46
1d	Connection time at point of origin	2,931±33 minutes
1e	Data transmitted	8,394±79 kilobytes
1f	Total port utilization time	6,455±70 minutes
1g	Mean data message transmission time	10 seconds
Gateway Intranet Routes (all gateway nodes)		
2a	Number of nodes	8
2b	Number of data messages transmitted	17,963±338
2c	Number of connections originated	13,709±276
2d	Connection time at point of origin	2,855±101 minutes
2e	Data transmitted	8,402±78 kilobytes
2f	Total port utilization time	22,671±310 minutes
2g	Mean data message transmission time	9 seconds
End-to-end Paths (all non-gateway nodes)		
3a	Number of nodes	1,585
3b	Number of unidirectional paths	26,636
3c	Number of data messages transmitted	77,193±1,332
3d	Number of connections originated	60,205±2,905
3f	Connection time at point of origin	10,026±179 minutes
3g	Data transmitted	24,733±252 kilobytes
3h	Total port utilization time	96,071±686 minutes
3i	Mean data message transmission time	6 seconds
3j	Number of data messages transferred end-to-end	77,209±1,327
3k	Mean end-to-end data message transfer time	4,971 seconds

3) The number of data messages transmitted (3c) differs from the number of data messages transferred end-to-end (3j) because messages are in transit when the gathering of statistics begins and ends. A similar observation is relevant to the discrepancy between (1b) and (2b), and (1e) and (2e).

4) The total port utilization time (1f) is greater than twice the connection time at point of origin (1d) for the gateway internet routes even though both source and destination are gateway nodes. This is due to the slightly different measurements being taken: the connection time is the actual billable time as seen by the network carrier and does not include line idle detection, modem reset and relisten delays which are included in the port utilization time.

5) The mean data message transmission time (1g, 2g and 3i) varies depending on the location of the route. Full-duplex transmission is approximately half the rate of half-duplex transmission. The gateways exhibit more frequent cases of full-duplex transmission because of their high level of traffic.

6) The mean end-to-end data message transfer time (3k) of 4,971 seconds (1:22:51) has a wide range. The standard deviation of this upward-skewed distribution is approximately 10,000 seconds (2:46:40). This result is not surprising as the polling period for lab test reports, which constitute 48 percent of all data messages transmitted, is 14,400 seconds (4:00:00). The mean message transfer time ranges from 387 seconds (0:06:27) at regional hospitals in an unspecified region to 18,228 seconds (5:03:48) at the Medical Care Insurance Commission in Regina. The long delay experienced by messages at the Medical Care Insurance Commission is due largely to a fixed evening call schedule for messages sent to Regina physicians.

Further statistical analysis from one of the trials yields insight into message transfer times. Figure 42 is a plot of the distribution of data message transfer times for medium-priority messages sent among physicians (both GP's and specialists). The probability densities shown are based on 50 second intervals. Those messages delivered point-to-point are transferred with a mean time of 543 seconds (0:09:03) and a median time of 250 seconds (0:04:10) (N=8,845). Messages which are delivered to remote subnets have a mean transfer time of 1,379 seconds (0:22:59) and a median time of 700 seconds (0:11:40) (N=5,361). The third quartile transfer times are 300

seconds (0:05:00) and 920 seconds (0:15:20) for intranet and internet messaging respectively.

Figure 42 can be compared to steady state graph shown in Figure 24 for a four-node cluster. The mode in Figure 42 is far more pronounced because the nodes are lightly loaded with an effective message frequency of about six messages per hour (based on peak loads while treating acknowledgment and data messages alike) as compared to a message frequency of 24 messages per hour for Figure 24. A lighter load results in less CPU and communication port utilization and, consequently, smaller delays. The mode in Figure 42 occurs at 250 seconds (0:04:10) as compared to the mode in Figure 34 which occurs at 550 seconds (0:09:10). This is due mainly to the decreased loading and the reduction in the medium priority delay from 300 (0:05:00) to 180 seconds (0:03:00).

6.5.3 The Client-Server Model

The second set of analyses performed are for the client-server model. The only changes made to the peer-to-peer configuration apply to 1) message routing, 2) call scheduling, and 3) number of gateway ports. Figure 43 is a partial diagram of the topology of the network. All messages are routed through a gateway node or server. Consequently, there are either two- or three-hop paths. Hourly polls are scheduled universally for all client nodes to their respective servers. Messages are held and delivered according to the hourly poll schedule. The servers, in accordance with their server role, never initiate a connection. Although data messages are given medium priority and message acknowledgments are given low priority, this has no effect on their delivery time because of the poll schedule. No priority delays are used in server-to-server communication.

Figure 42. The Saskatchewan Peer-to-Peer Model: Probability Density Functions for End-to-End Transfer Times for Data Messages Sent Among Physicians

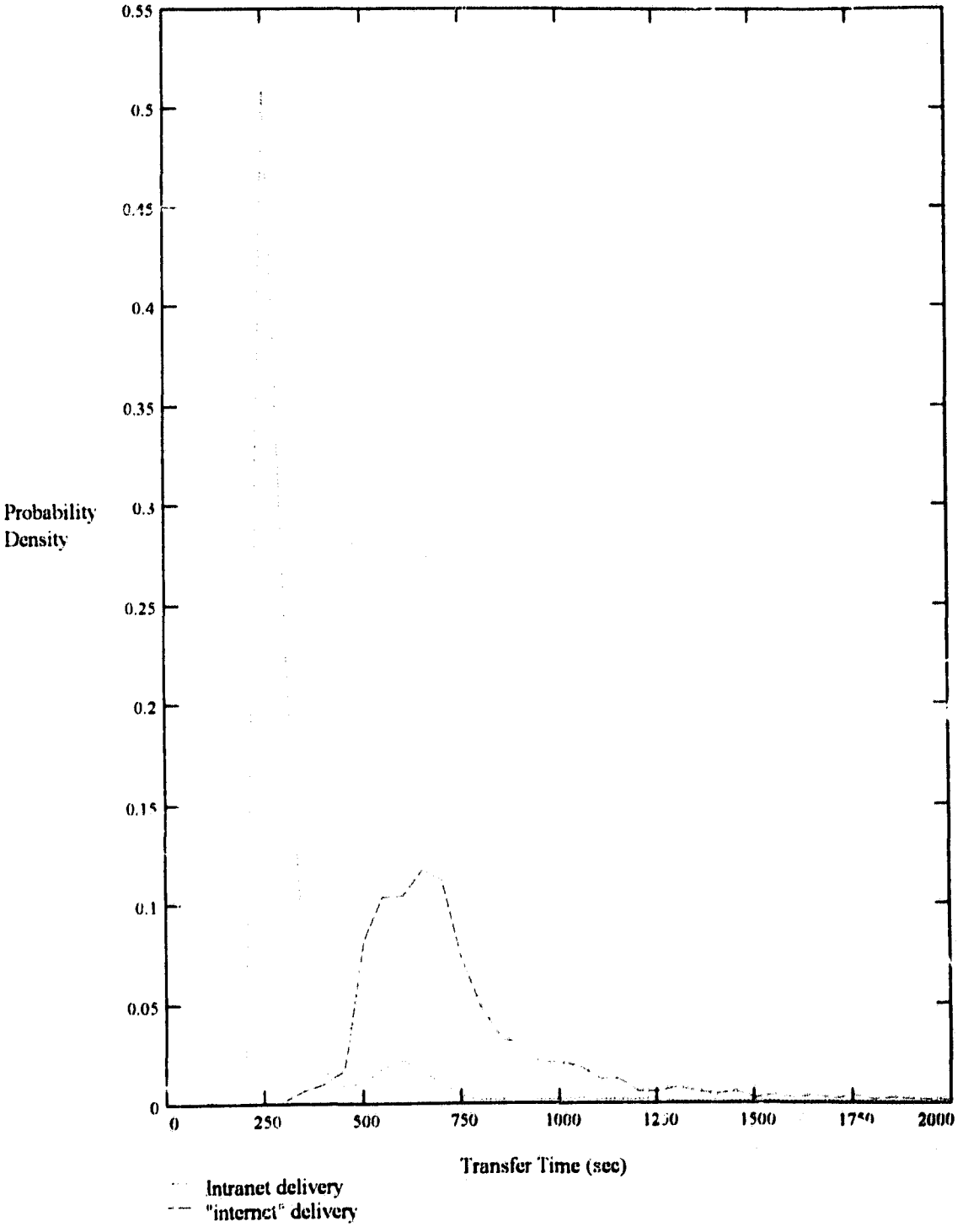
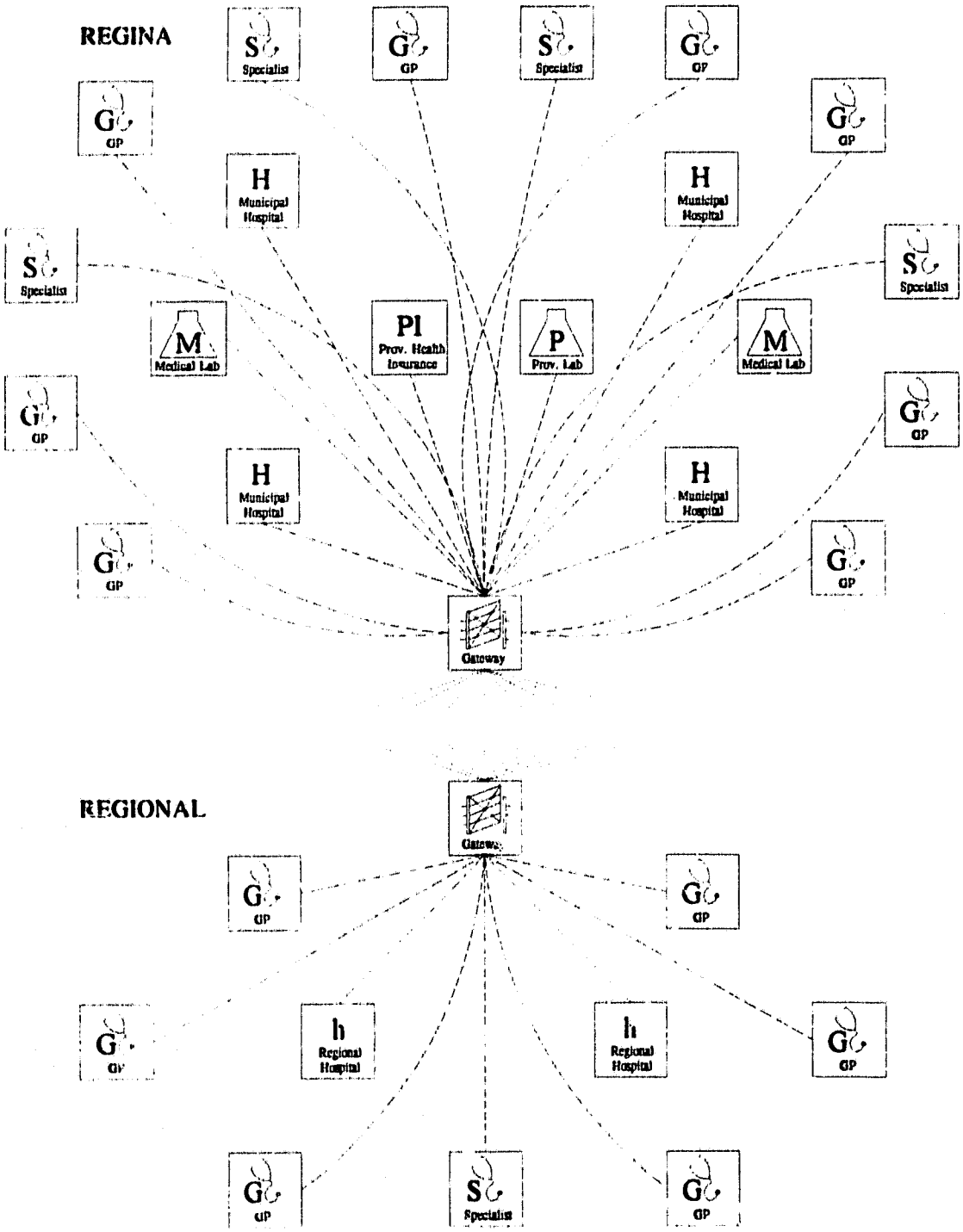


Figure 43. The Saskatchewan Client-Server Model: A Partial Topology Diagram



The number of intranet gateway ports (ports accessible to client nodes but not to other gateways) has been adjusted as shown in Table 23. This change permits the port utilization at the gateways to remain at roughly the same even though there is increased traffic. The number of internet ports remains unchanged at seven since no single gateway has more than seven neighbors. The number of ports at client nodes is unchanged even though only one port at a time can be in use because a client node contacts only its own gateway server. By leaving this parameter unchanged, the CPU power of each client node remains unchanged.

Table 23. The Saskatchewan Model: A Comparison of Gateway Port Configurations

Subnet Designation	Subnet Scaling	Peer-to-Peer Number of Ports	Peer-to-Peer % Utilization	Client-Server Number of Ports	Client-Server % Utilization
REGINA	1	32	14	64	13
REGIONAL	2	12	12	12	12
REGIONAL	1	8	8	8	8
SASKTOON	1	32	15	64	14

Table 24 is a summary of the statistics gathered from the model. As with the peer-to-peer simulations described in Section 6.5.2, means and standard errors resulting from ten replication trials are presented. Each trial has a 24-hour warm-up period before statistics are collected. Observations concerning the table are:

1) 23 percent or 19,038 data messages (1b) require three hops to reach their destinations, requiring the assistance of two gateway nodes. All other data messages require only two hops.

2) The 24,072 unidirectional paths (3b) reported are composed of 3,226 unidirectional links. The number of paths is less than that reported for peer-to-peer communication (26,636) because physicians no longer poll the medical and Provincial laboratories directly.

Table 24. The Saskatchewan Client-Server Model: Summary of Simulation Statistics

Gateway Internet Routes (all gateway nodes)		
1a	Number of nodes	8
1b	Number of data messages transmitted	19,038±557
1c	Number of connections originated	13,259±241
1d	Connection time at point of origin	3,136±58 minutes
1e	Data transmitted	8,637±96 kilobytes
1f	Total port utilization time	9,320±158 minutes
1g	Mean data message transmission time	8 seconds
Gateway Intranet Routes (all gateway nodes)		
2a	Number of nodes	8
2b	Number of data messages transmitted	81,424±1,985
2c	Number of connections originated	0
2d	Connection time at point of origin	0
2e	Data transmitted	25,693±351 kilobytes
2f	Total port utilization time	33,932±374 minutes
2g	Mean data message transmission time	8 seconds
End-to-end Paths (all non-gateway nodes)		
3a	Number of nodes	1,585
3b	Number of unidirectional paths	24,072
3c	Number of data messages transmitted	81,139±1,982
3d	Number of connections originated	38,217±128
3f	Connection time at point of origin	18,913±329 minutes
3g	Data transmitted	25,629±347 kilobytes
3h	Total port utilization time	51,402±427 minutes
3i	Mean data message transmission time	7 seconds
3j	Number of data messages transferred end-to-end	81,423±1,984
3k	Mean end-to-end data message transfer time	5,023 seconds

3) The number of data messages transmitted (3c) differs from the number of data messages transferred end-to-end (3j) because messages are in transit when the gathering of statistics begins and ends. Statistics for (1b) and (2b) and (1e) and (2e) are radically different because all messages are directed to a server, even though they may not be leaving their subnet.

4) The number of connections created (1c) is 4.7 times greater for the client-server model than it is for the peer-to-peer model. This is caused by the removal of priority delays for internet traffic. This increase translates into a 1.4-fold increase in the total port utilization time (1f).

5) The gateways exhibit a notably better mean message transmission time (1g) of eight seconds as compared to the peer-to-peer model which has an average time of ten seconds. Removal of priority delays results in a greater proportion of messages being transferred in half-duplex mode as fewer messages are able to accumulate in mail bags between connection requests. In contrast, the mean message transmission time (3i) at the client nodes is longer (seven seconds) than with the peer-to-peer model (six seconds) because hourly polling of these nodes in the client-server model increases the probability of messages collecting in a mail bag.

6) The mean end-to-end data message transfer time (3k) of 5,023 seconds (1:23:43) compares favourably with that of the peer-to-peer model (1:22:51) . This is in consequence of intentionally setting the polling frequency to one hour. Although such high-frequency polling is probably impractical, it results in grossly similar performance in both models. Mean end-to-end data message transfer times range from 3,668 seconds (1:01:08) to 16,122 seconds (4:28:42) exhibited at the two municipal hospitals in Regina and at the Medical Care Insurance Commission in Regina respectively. The 4:28:42 transfer time is due largely to contention for the single communication link between the corporation and its gateway. Unlike other nodes, the Medical Care Insurance Commission receives its data from all 1,553 physicians in a restricted four-hour period in the evening.

Figure 44, taken from one replication trial, is a distribution of data message transfer times for messages sent among physicians. This graph is directly comparable to the peer-to-peer case in Figure 42 except that the abscissa is scaled by a factor of ten

and probability densities are based on 500 second intervals. There is only a minor difference between the plots for intranet and internet messages. For messages sent within a subnet, the mean time of transfer is 4,619 seconds (1:16:59) and the median is 4,300 seconds (1:11:40) (N=9,064). For messages sent to remote subnets the mean time of transfer is 5,204 seconds (1:26:44) and the median is 4,650 seconds (1:17:30) (N=5,537). The third quartile transfer times are 5,950 seconds (1:39:10) and 6,550 seconds (1:49:10) for intranet and internet messaging, respectively.

Figures 42 and 44 are radically different even though the overall end-to-end transfer times for the two models are similar. The figures illustrate that with the peer-to-peer model, unlike with the client-server model, it is possible to control the transfer times. Physicians typically receive messages in under 15 minutes whereas the average end-to-end transfer time for all messages is 1:22:51. In the client-server model, the mean end-to-end transfer time for physicians is no different from the transfer time for all messages. The transfer times for the client-server model are mainly determined by the polling frequency of the nodes. The shape of the curves in Figure 42 are mainly determined by message processing and transmission, the shape of the curves in Figure 44 are mainly determined by the arrival times of polls at the server.

6.5.4 Network Costs

This section presents two telecommunication cost analyses for the operation of a network. The analyses are based on two trial simulations, one using peer-to-peer communication and one using client-server communication as described in Section 6.5.2 and Section 6.5.3 respectively. A simplified business case is presented for the operation of the less expensive model, the client-server network.

Telecommunication cost comparisons between the peer-to-peer and the client-server models are made even though the results are subject to stochastic variation. The cost is based on two types of third-party supplied network services: 1) public switched telephone and 2) packet switching. An effort has been made to choose the most appropriate but least expensive service for each topology. For the peer-to-peer topology, the telephone is used exclusively within a subnet and Datapac 3000 is used among subnet gateways. For the client-server topology, a mixture of telephone and Datapac 3101 is used within a subnet and Datapac 3000 is used among subnet

gateways. Other types of service, such as Wide Area Telephone Service (WATS) and leased lines, might prove to be slightly more cost-effective. Such optimizations are dependent on geography-specific traffic patterns which are not explicitly addressed in the simulations.

Tables 25 and 26 present the *traffic* and *holding times* during a 22-day month for the peer-to-peer model. Tables 27 and 28 provide the same information for the client-server model. Traffic is defined as the number of protocol and data packets transmitted. This packet count is dependent on 1) the total amount of data sent, 2) the number of messages sent and 3) the number of connections established. The maximum payload for a data packet is 175 bytes (octets), the size used by *Health Link*. The holding time refers to the cumulative elapsed time for established connections. The holding time is dependent on 1) the amount of data transmitted, 2) the number of messages transmitted, 3) the communication mode (full- or half-duplex), and 4) the number of connections established. The holding time of any individual call is rounded up to the nearest minute before it is accumulated. The vast majority of connections are established for one minute or less: 98.6 percent for the peer-to-peer model and 94.5 percent for the client-server model.

The information presented in the tables provides some insight into the behaviour of the two models. The gateway/node traffic for the client-server model is roughly comparable to the gateway/node and the node/node traffic of the peer-to-peer model. Theoretically, it would be the sum of the gateway/node traffic plus twice the node/node traffic. However, the client-server model uses hourly polls rather than calls on-demand. Consequently, the client-server model makes 58 percent of the calls made by the peer-to-peer model which results in a traffic reduction of 34 percent. A similar observation can be made about the holding time.

Figure 44. The Saskatchewan Client-Server Model: Probability Density Functions for End-to-End Transfer Times for Data Messages Sent Among Physicians

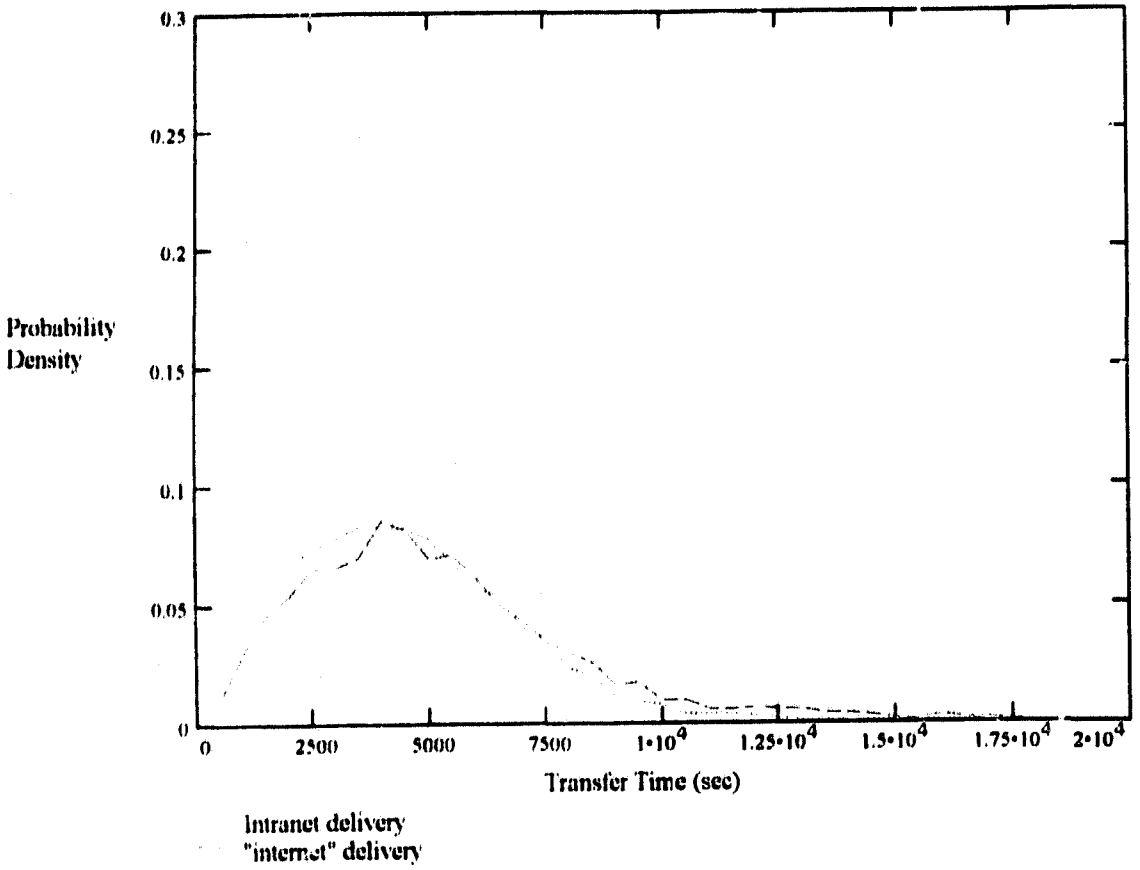


Table 25. The Saskatchewan Peer-to-Peer Model: Monthly Billing Statistics

Subnet	Gateway/ Node Traffic	Node/ Node Traffic	Gateway/Gateway Traffic		
			Moose Jaw	North Battleford	Prince Albert
Moose Jaw					
kilosegments	1,069	921		22	14
holding time (min)	18,964	24,442		418	275
North Battleford					
kilosegments	2,195	1,915	22		33
holding time (min)	36,234	51,436	418		605
Prince Albert					
kilosegments	3,064	1,906	14	33	
holding time (min)	55,506	51,964	275	605	
Regina					
kilosegments	11,114	19,375	188	330	320
holding time (min)	130,218	425,722	2,706	3,938	3,630
Saskatoon					
kilosegments	12,426	18,828	76	164	462
holding time (min)	200,970	423,412	1,133	1,859	4,554
Swift Current					
kilosegments	2,001	931	13	13	11
holding time (min)	41,492	24,882	231	253	220
Weyburn					
kilosegments	1,064	931	8	14	12
holding time (min)	18,216	24,266	187	253	275
Yorkton					
kilosegments	3,495	1,922	14	28	42
holding time (min)	62,194	52,008	286	561	704
Total	36,430	46,729	336	605	893
	563,794	1,078,132	5,236	7,887	10,263

Table 26. The Saskatchewan Peer-to-Peer Model: Monthly Billing Statistics (continued)

	Gateway/Gateway Traffic				
Subnet	Regina	Saskatoon	Swift Current	Weyburn	Yorkton
Moose Jaw					
kilosegments	188	76	13	8	14
holding time (min)	2,706	1,133	231	187	220
North Battleford					
kilosegments	330	164	13	14	28
holding time (min)	3,938	1,859	253	253	561
Prince Albert					
kilosegments	320	462	11	12	42
holding time (min)	3,630	4,554	220	275	704
Regina					
kilosegments		1,511	240	191	654
holding time (min)		11,374	3,245	2,552	5,863
Saskatoon					
kilosegments	1,511		335	128	192
holding time (min)	11,374		3,432	1,507	2,123
Swift Current					
kilosegments	240	335		4	13
holding time (min)	3,245	3,432		77	275
Weyburn					
kilosegments	191	128	4		14
holding time (min)	2,552	1,507	77		297
Yorkton					
kilosegments	654	192	13	14	
holding time (min)	5,863	2,123	275	297	
Total	3,434	2,868	629	370	958
	33,308	25,982	7,733	5,148	10,043

Table 27. The Saskatchewan Client-Server Model: Monthly Billing Statistics

Subnet	Gateway/ Node Traffic	Node/ Node Traffic	Gateway/Gateway Traffic		
			Moose Jaw	North Battleford	Prince Albert
Moose Jaw					
kilosegments	2,021	0		77	22
holding time (min)	36,432	0		1,056	671
North Battleford					
kilosegments	4,189	0	77		63
holding time (min)	70,840	0	1,056		924
Prince Albert					
kilosegments	4,891	0	22	63	
holding time (min)	75,526	0	671	924	
Regina					
kilosegments	33,808	0	358	715	1,665
holding time (min)	368,984	0	9,152	15,994	15,972
Saskatoon					
kilosegments	33,659	0	228	438	849
holding time (min)	459,426	0	5,841	9,702	16,984
Swift Current					
kilosegments	2,319	0	6	16	17
holding time (min)	36,498	0	176	418	484
Weyburn					
kilosegments	2,238	0	7	24	20
holding time (min)	37,532	0	198	682	594
Yorkton					
kilosegments	4,284	0	14	42	67
holding time (min)	72,864	0	374	1,177	1,848
Total	87,409	0	712	1,373	2,703
	1,158,102	0	17,468	29,953	37,477

Table 28. The Saskatchewan Client-Server Model: Monthly Billing Statistics (continued)

	Gateway/Gateway Traffic				
Subnet	Regina	Saskatoon	Swift Current	Weyburn	Yorkton
Moose Jaw					
kilosegments	358	228	6	7	14
holding time (min)	9,152	5,841	176	198	374
North Battleford					
kilosegments	715	438	16	24	42
holding time (min)	15,994	9,702	418	682	1,177
Prince Albert					
kilosegments	1,665	849	17	20	67
holding time (min)	15,972	16,984	484	594	1,848
Regina					
kilosegments		4,164	605	533	644
holding time (min)		23,287	13,200	12,419	15,378
Saskatoon					
kilosegments	4,164		427	167	401
holding time (min)	23,287		5,104	4,268	8,855
Swift Current					
kilosegments	605	427		19	17
holding time (min)	13,200	5,104		264	495
Weyburn					
kilosegments	533	167	19		39
holding time (min)	12,419	4,268	264		561
Yorkton					
kilosegments	644	401	17	39	
holding time (min)	15,378	8,855	495	561	
Total	8,684	6,673	1,108	809	1,223
	105,402	74,041	20,141	18,986	28,688

The gateway/gateway statistics vary between the two models because the peer-to-peer model uses priorities to schedule messages traveling between subnets whereas the client-server model does not. As the internet traffic and holding time is only a fraction of the intranet traffic and holding time, this is unlikely to constitute a major cost consideration. The proportion of internet traffic to intranet traffic is 12.1 percent for the peer-to-peer model as compared to 26.6 percent for the client-server model. The proportion of internet holding time to intranet holding time is 6.4 percent for the peer-to-peer model as compared to 28.7 percent for the client-server model.

Tables 29 and 30 present network telecommunication costs which are based on the two trial simulations. The billing rates used are those supplied by SASKTEL.

In the case of the peer-to-peer model (Table 29), each node requires at least one telephone access. For gateways and larger institutions, such as the hospitals, the number of accesses is determined by the number of ports specified in the configuration. Datapac 3000 is used for internet communication. The number of Datapac 3000 virtual circuits used at each gateway is determined by the gateway loading. All gateways use three virtual circuits except for Saskatoon and Regina which use seven.

The long distance charges in Table 29 are determined by the geographical distribution of the nodes with respect to their respective gateway. It is assumed that either a node is local to its gateway and all other local nodes, thereby incurring no long distance charge, or else it is long distance to its gateway and all other nodes in the subnet, thereby incurring the basic long distance minimum charge for each holding minute. No rate adjustments are made for distance as the minimum charge is usually greater than the distance-sensitive charge assessed for the call. No after-hours rate adjustments are made since the majority of all calls are one minute or under and are billed at the minimum rate.

Table 29. The Saskatchewan Peer-to-Peer Model: Monthly Telecommunication Costs

	Basic Charges ($)		Usage Charges ($)		
Subnet	Phone	Datapac	Long Distance Phone	Datapac Traffic	Total
Moose Jaw	2,061	274	2,457	215	5,007
North Battleford	3,258	274	13,910	455	17,897
Prince Albert	4,456	274	11,607	593	16,930
Regina	17,647	291	9,774	2,782	30,494
Saskatoon	24,089	291	9,248	2,212	35,840
Swift Current	1,727	339	5,854	535	8,455
Weyburn	1,476	339	7,274	243	9,332
Yorkton	3,008	339	17,169	814	21,330
Total ($)	57,722	2,420	77,293	7,849	145,285
	Datapac Volume Discount (9%)				(706)
	Total Monthly Telecommunication Cost				$144,578
	Cost Per Physician				$93.50

Table 30 is an analysis of telecommunication costs for the client-server model. Nodes in the client-server model use polling to pick up messages. Nodes make only outgoing calls and therefore do not need exclusive telephone access. It is safe to assume that all physicians already have telephone access and that there is no need to install additional telephone service. Large institutions, such as hospitals, do require exclusive telephone access because of high network utilization. However, analysis indicates that a single access is adequate for each institution. Those nodes which are distant from their respective gateway use Datapac 3101 which is cheaper than long distance telephone. In most cases, a SASKTEL offering called Universal Datapac Access (UDA) is least expensive. This service incurs the normal Datapac 3101 holding charge ($.05/minute) and traffic charge ($.42/kilosegment) as well as a UDA surcharge of $.15/min which compensates for long distance charges. In two regions, Regina and Weyburn, there are twelve nodes which are able to reach Datapac public access ports without long distance charges even though they are outside the immediate dialing area

of the gateway. These nodes use Datapac 3101 without UDA. Datapac 3101 offers a 25 percent discount which is applied to connections made in off-peak hours (7pm to 7 am).

Table 30. The Saskatchewan Client-Server Model: Monthly Telecommunication Costs

	Basic Charges ($)		Usage Charges ($)			
Subnet	Phone	Datapac	Datapac Traffic	Datapac Holding	Datapac Usage Discount	Total
Moose Jaw	279	309	603	1,218	(43)	2,366
North Battleford	362	505	1,719	6,240	(280)	8,547
Prince Albert	418	507	2,601	4,675	(235)	7,966
Regina	2,550	440	7,564	1,985	(209)	12,329
Saskatoon	2,349	444	5,788	3,773	(266)	12,087
Swift Current	251	394	1,250	2,421	(122)	4,193
Weyburn	139	410	955	3,357	(154)	4,706
Yorkton	306	554	1,738	6,103	(276)	8,486
Total ($)	6,653	3,563	22,218	29,832	(1,585)	60,680
			Datapac Volume Discount (11%)			(2,444)
			Total Monthly Telecommunication Cost			$58,236
			Cost Per Physician			$37.67

Telecommunication cost estimates in Tables 29 and 30 must be viewed with caution for a number of reasons:

1) Not all health care providers who would participate in an actual network are included in the model.

2) Network utilization is based on estimates.

3) Message creation schedules are arbitrary.

4) Call and polling schedules are not optimized with respect to cost.

5) Model topologies are not optimized with respect to cost.

6) Routes are randomly selected by the simulation program.

7) Estimates are based on a single trial simulation, not a replicated series of trials.

8) Long distance tariffs for calls longer than one minute have been charged at the minimum tariff.

9) Nodes which are remote (incurring long distance rates) to their gateway are deemed to be remote to all other nodes in their subnet.

10) Some of the lesser charges such as the cost of telephone hunt groups and corporate Network User Identifiers (NUI) have been omitted from the analysis.

Because of the lack of optimization, it is probable that the cost estimates are overstated. For example, polling can be performed far less frequently at night as long as the bulk of the messages has been received before the following business day. Priority delays can be adjusted to reduce the number of internet connections made, thus reducing the cost of internet communication. Acknowledgment messages may be eliminated for those data messages where no acknowledgment is required, i.e., where there is follow-up documentation or where there is audit control. For the peer-to-peer model, a savings of approximately $4.50 per month per physician can be realized by reducing the number of long distance after-hour calls. For the client-server model, a savings of approximately $6 per month per physician can be realized by setting internet message priorities to the same values as those used with the peer-to-peer model. On the basis of these assumptions and observations, it is not unreasonable to assume a least cost per physician of $30/month for telecommunication.

An estimate of the cost of network operation must also include the cost of capitalization, overhead and administration. Table 31 gives an estimate of the capital investment for the client-server model. Table 32 provides an estimate of the overall operating expenses based on a cost-optimized client-server model as administered by a not-for-profit organization before taxes.

The monthly operating expenses total to $75,760 for a client-server network which serves an actual population of 1,546 physicians, four private medical laboratories, 26 hospitals, the Provincial Medical Laboratory and the Medical Care Insurance Commission. If the cost is divided among the physicians equally, this

represents a subscription price of $49 per month. However, this cost analysis does not include the following items:

- business and real estate taxes,
- Federal and Provincial sales taxes on fixed assets, software and telecommunication services,
- amortization of installation and startup costs,
- purchase and installation of equipment at end-user sites,
- site maintenance at gateway locations, or
- leasehold improvements.

Furthermore, it is assumed that the network is administered by a not-for-profit organization.

Table 31. The Saskatchewan Client-Server Model: Capital Requirements

Capital Account	Amount
Gateway computers (8 installed, 2 spares)	$65,000
Packet assembler/disassembler (8 installed, 1 spare)	$55,000
Hardware tools (testers, break out boxes, connectors, etc.)	$3,000
Office equipment	$5,000
Furniture and fixtures	$5,000
Communications software (purchase)	$250,000
Software tools (compilers, third-party libraries and packages)	$10,000
Total	$393,000

6.6 Conclusion

Keeping in mind that the simulation studies described in this chapter relate to Saskatchewan, their results suggest that:

1) It is practical to build large scale store-and-forward networks to transfer text messages among orthodox health care providers and major health care institutions,
2) Large scale health care networks can be administered for a cost of approximately $50 per month per physician.

3) Operation of the client-server topology costs about half that of the peer-to-peer topology given the current tariff structure for telephone and packet switched services.
4) The client-server topology, unlike the peer-to-peer topology, is insensitive to message priorities. There is no means to provide expedited message delivery and there is little that can be done to improve performance short of increasing the frequency of gateway polling.

Table 32. The Saskatchewan Client-Server Model: Projected Monthly Operating Expenses

Expense Account	Debit Amount
Telecommunication costs (reduced from $58,236 by adjusting internet priority delays)	$49,144
Interest on capital (8% compounded monthly)	$2,553
Equipment depreciation (amortized over five years in a straight line)	$2,136
Furniture and fixtures depreciation (amortized over 25 years in a straight line)	$17
Software depreciation (amortized over three years in a straight line)	$7,222
Leased vehicle	$350
Salaries and benefits (two employees)	$5,958
Hardware maintenance (15% annually of purchase price)	$1,602
Software maintenance (15% annually of purchase price)	$3,125
Office lease (200 square feet at $20 per square foot)	$333
Telephone	$1,000
Utilities	$70
Supplies	$100
Janitorial service	$150
Advertising and documentation	$1,500
Travel	$500
Total Monthly Operating Expenses	$75,760

Some of the proposals for health care networks, like the British Columbia Pharmacy Network Project [38], propose on-line, real-time interactive networks using a

client-server approach. These types of networks are far more expensive to operate. For example, if the Saskatchewan client-server model were to be implemented as an on-line system, every node in the network would pay a Datapac 3000, Datapac 3101 or Datapac 3102 private access fee. For example, Datapac 3101 access costs $230 per month for a 2,400 bps circuit in Moose Jaw, North Battleford, Prince Albert, Regina and Saskatoon. Other areas would pay $287.50 per month. If a node is *off-net*, i.e., not local to the regional centre, additional distance-sensitive charges are incurred to gain *on-net* access. The gateway costs might also be increased as a result of the large number active circuits. (If Datapac 3102, a polled network service, is used the telecommunication costs at the gateway are likely remain approximately the same.) On-line networks, however, have the advantage that the delays in message delivery are inconsequential.

The client-server topology is unattractive with respect to performance. If the user chooses to poll the server manually rather than letting the computer perform the polling automatically, long delivery delays are likely to be experienced. Although the recipient of messages may not be concerned with their timeliness, the sender, who has no control over the recipient's polling frequency, may. Consequently, the delays reduce the attractiveness of electronic data interchange with respect to other modes of communication, like the telephone, facsimile, courier or even the mail, where spontaneous communication can be achieved. End-user applications which rely on electronic data interchange lose their effectiveness and, possibly, their validity if long delays are experienced during communication.

Further optimization of cost with respect to performance could be achieved by simulating hybrid topologies which permit time-sensitive applications to establish point-to-point connections (peer-to-peer communication) with a set of specific institutional nodes while maintaining a client-server relationship with the subnet gateway.

Such large-scale networks are expensive to install and operate. This type of simulation study is worthwhile because it permits the cost and the performance of a proposed network to be estimated in advance. The simulation program developed for these studies is useful for rapid model construction and convenient *what-if* analysis. The program execution time for models like the Saskatchewan network can be lengthy

(six hours of actual time is needed to simulate 48 hours of model time), but these runs are needed only after a series of shorter runs have verified the model configuration.

7 *Health Link* Field Test

7.1 Introduction

A field test is used to evaluate the *Health Link* prototype in an actual operating context. A field test can present an opportunity to discuss and explore fundamental concepts and objectives in telehealth with health care practitioners. The network itself can be assessed with respect to its reliability and ease of installation and operation. Moreover, the impact of the network on its users can be observed. Organizational structure, data exchange traffic, information use, information requirements, information management procedures, staffing, and space utilization are among the characteristics which can be observed and measured at the health care provider sites. Feedback to the development team may result in design improvements, new features, better documentation and more effective installation and maintenance procedures. The field test can provide understanding of the user base which is essential to network growth and increased functionality. Data can be collected for simulation studies which are used to extrapolate the behaviour of a large scale network. Lastly, the field test can provide live reference sites for demonstration to health care providers in an effort to promote a generic wide area network for health care.

A six-node network was installed using the *Health Link* prototype. The field test was conducted in several stages. Surveys of the procedures and information requirements of the participating health care providers were conducted. Stress tests and performance benchmarks were taken for a four node cluster. A subnet was operated for three nodes without a gateway and, later, for all six nodes with a gateway. During the early part of the pilot, the field test participants were two medical clinics and one medical laboratory. Laboratory test reports were sent by the laboratory to the two clinics. Acknowledgment messages confirmed the delivery of each laboratory report. During the later part of the field test, two additional clinics and a hospital were added to the network. The expanded network is currently used to exchange free-text messages and laboratory results. Hospital radiology and pathology reports will be added to the repertory in the near future.

This chapter presents the objectives of the field test, discusses its implementation, and reports the significant findings. Valuable data have been collected

which describe the end-user applications, procedures and information traffic. A wealth of anecdotal findings have been collected as well.

7.2 Objectives

The field test has a number of objectives: 1) it is seen as a means to introduce and demonstrate *Health Link* as an appropriate technology for health care providers, 2) it presents an opportunity to survey some of the applications best suited for electronic messaging, 3) it provides a live context in which to evaluate the reliability of a *Health Link* network, 4) it exposes users to an operational prototype and permits an assessment to be made of user acceptance, and 5) it provides a limited opportunity to assess potential cost benefits.

The field test is a means to install *Health Link* at the sites of health care providers beyond the confines of the laboratory. The current *Health Link* prototype should be demonstrated to, and accepted by, the health care community before it can be taken into production. The field test can be viewed as a live demonstration which will be an installation base for a future production network.

A survey of common information interchange requirements can be conducted as part of the field test. It is particularly important to determine which types of messages are communicated and their frequency, urgency and length. The survey can be used to determine which of several competing technologies are appropriate for each application. The survey can also be used to determine network demands and loads. From this data, the network configuration can be derived and cost estimates can be made for large-scale networks.

The field test can be used to determine the reliability of a *Health Link* network in a live context. Adapters running under live conditions are subject to actual load conditions, non-uniform physical operating environments and a range of system operator procedures and skills. The resilience of *Health Link*, when subjected to difficulties such as poor power conditioning, noisy telephone connections and processing bottlenecks, can be assessed.

Electronic data interchange has an impact on the office procedures of health care providers. Electronic transmittals can replace paper resulting in changes to both computer- and paper-related procedures. Fear of job loss and redefinition of tasks can cause anxiety among staff. This anxiety may be offset by a desire to embrace the new

technology because of its potential to reduce work loads and its high technology appeal. A field test provides a means to assess user acceptance of a telecommunication network in a monitored context.

Once electronic messaging has been incorporated into the procedures of the field test partners, it is possible to estimate the reduction in office operating costs and the tangible benefits resulting from the more timely receipt of accurate information. The costs of electronic messaging can then be compared to these benefits to justify telecommunication quantitatively.

7.3 Implementation

The field test has been conducted over a period of one and one-half years and has grown to a subnet as shown in Figure 45 consisting of six adapters at end-user sites and one gateway at the University of Victoria. In its final form, the subnet serves 27 physicians. The main application supported by the subnet is transmittal of laboratory endocrinology and hematology results. However, it is anticipated that hospital radiology and pathology reports will be distributed by the subnet as well. Free-text messaging is also available to network users.

Table 23 is an abbreviated schedule which shows the six different stages of the field test. The first and fifth stages consist of the collection of message statistics at two medical clinics by project personnel. Subnet stress testing is conducted during Stage Two. It is followed by a two-month period when *in situ* performance tests are conducted in Stage Three. A two node cluster is in operation for the following year (Stage Four). The fifth and sixth stages, when the remaining nodes in the subnet are surveyed and installed, overlap the fourth stage. The author helped to conduct Stages two, three and six; the remaining work has been performed by other members of the project team.

Figure 45. Field Test Geography

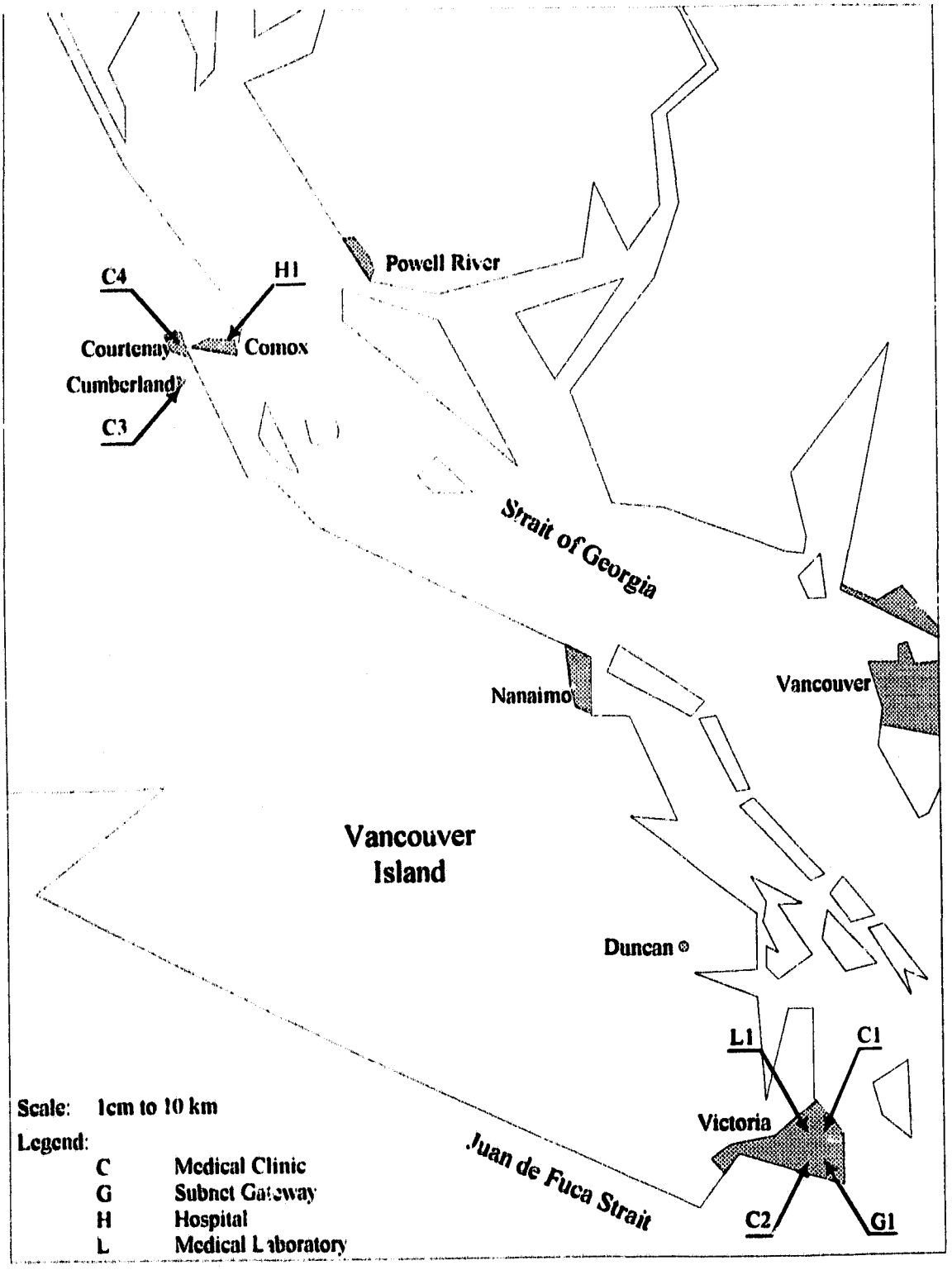


Table 33. Field Test Stages

Duration	Stage	Activity
2 months	1	Survey of procedures and information traffic patterns conducted at medical clinics C1 and C2
3 months	2	Subnet stress tests conducted among sites L1, C1, C2 and University nodes
2 months	3	Performance measurements taken at sites C1, C2 and University laboratory
1 year	4	Live operation of L1 and C1
3 months	5	Survey of procedures and information traffic patterns conducted at medical clinics C3 and C4, and hospital H1
2 months	6	Live operation of all nodes

7.4 Results

The field test has provided data in several areas: 1) message statistics used as input to the simulation studies described in Chapter 6, 2) calibration performance statistics referenced in Chapters 5 and 6, 3) *Health Link* reliability statistics, 4) system reliability statistics which include the performance of the end-user interfaces to *Health Link*, and 5) message characteristics for a range of different applications used by physicians. Perhaps most importantly, the field test has provided a wealth of feedback relating to end-user operating contexts, network appropriateness and acceptance, application interface requirements, and network installation and operating procedures.

As message and performance statistics are described elsewhere, this section reviews only the data relating to reliability statistics (Section 7.4.1) and the message characteristics (Section 7.4.2) and various anecdotal findings (Section 7.4.3). Feedback relating to network installation and operation is reflected in the analysis given in Section 7.5.

7.4.1 Reliability Statistics

During the course of the field test, the *Health Link* network proved to be highly reliable. In the first six months of Stage Four, the software transferred approximately

30 messages and their acknowledgments per week. Midway in Stage Four, service was interrupted with the installation of a new version of the prototype software. The new software required constant supervision for several months until several program errors were eliminated. Now, once again, the software operates without failure. Currently, the subnet transmits approximately 30 messages and their acknowledgments daily.

Equipment failures were extremely rare. In the first six months of operation of Stage Four, there was only one recorded instance where messages failed to be delivered. This failure was attributed to the failure of a modem to reset and assume an auto-answer state. No messages were lost or corrupted by *Health Link* during this period. Except for the one case of equipment failure and situations where reconciliation data were collected, the adapters were left to run without intervention.

The reliability of the end-user interfaces to *Health Link* is not so easily determined. Only the links between L1 and C1 and L1 and C2 have been monitored for application-to-application reliability. A number of data entry errors and application errors at the L1 adapter have lowered the reliability of delivery to at best 93 percent (sample size, N=14). Typically, the reliability is 80 percent (N=171). The L1-C1 link is most problematic of all in the subnet because physicians ordering the tests at clinic C1 often maintain practices outside the clinic. The lab reports directed to C1 need to be manually addressed to send them to their proper destination.

7.4.2 Message Characteristics

The survey conducted in Stage One investigated the structure and information requirements of medical clinics C1 and C2. At these clinics, types of information activities which are most appropriate to electronic patient record systems and telecommunication are:

1) storing, retrieving and distributing documents belonging to patient medical records,
2) forwarding mis-addressed patient records,
3) insurance claims and follow-up on rejected claims,
4) appointment bookings for patients with hospitals, clinics and specialists,
5) medical and other consultation reports to other clinics and specialists,
6) collection and analysis of patient statistics,

7) access to MedLine and other databases, and

8) submission of reports to government agencies, e.g. Workers' Compensation Review Board.

There is a large number of document types falling within the first category. One major type of transaction is the laboratory report produced by private medical laboratories. Other types of reports are generated by the Provincial cytology/virology laboratory, hospital emergency departments, hospital X-ray departments, hospital laboratories, ambulatory clinics, non-physician consultants and specialists. At clinic C1, where only manual systems are in place, electronic storage and retrieval of documents would save 3.5 full-time equivalent (FTE) clerical positions. This is equivalent to 10 minutes of filing time per patient encounter. It is likely that new tasks would be created in consequence to computerization, however, these would likely be absorbed by the existing staff with some net savings.

Telephone communication was also surveyed. Substitution of voice communication by electronic mail would be appropriate only for the follow-up of insurance claims. The telephone remains the most appropriate means of communication for appointment bookings, pharmacy consultations, lab report for stat tests, and external patient contact relating to appointments and treatments.

The survey revealed a need for electronic communication of text data only. It is conceivable that the need for electronic interchange of text overshadows the need for the interchange of digitized image and biometric data.

The second survey conducted at Stage Five provided a synopsis of organization structure and current computer-based applications at the C3 and C4 medical clinics. It also produced a description of radiology and pathology reporting applications at the H1 hospital. This survey was conducted prior to increasing the number of nodes in the subnet and extending the types of electronic messages exchanged.

7.4.3 Anecdotal Findings

The field test has provided understanding of office procedures and of physician information requirements and valuable experience with interface development, adapter installation and network maintenance. The field test has demonstrated some automation problems which are intrinsic to the health care community.

7.4.3.1 Office Procedures

The surveys revealed that the medical clinics receive a large amount of paper communication. If the clinic possesses an electronic patient record system, this information is transcribed into the data base and the paper copy is then archived. In the non-automated clinics, the paper copy is filed. These procedures are time-consuming at both clinic types. The time required for paper storage and retrieval is estimated at 10 minutes per patient encounter in a non-automated clinic. The time for data transcription and paper management at an automated clinic is approximately the same. Electronic data interchange can relieve some of this workload provided that the clinic is automated and there is little consequential administrative overhead. It is estimated that, in the context investigated, the clerical workload would be reduced by up to 50 percent.

The telephone is frequently used for communication among providers. Stat laboratory test results; operating room bookings and confirmations; referral bookings to specialists, outpatient clinics and physiotherapy clinics; medication orders; pharmacy consultations; rejected insurance claims; and communicable disease reports are all examples of verbal transactions. Frequently, these verbal transactions are substantiated by paper documents. Store-and-forward messaging cannot replace verbal communication because of a need for immediate feedback: many of the booking transactions are conducted while the patient is at the clinic.

The pre-field test procedures usually used modes of communication that are most appropriate depending on response time, cost and transaction length. Telephone, facsimile, courier, mail and drop boxes were and still are used. Staffing schedules and procedures are organized around the delivery systems. Electronic data interchange must be able to compete favourably with the existing delivery systems if it is to be adopted. For example, courier deliveries of laboratory test reports occur between two and four times daily at the clinics. Electronic distribution of laboratory test results must provide better response with a granularity of better than two hours at the high volume clinics. Courier service not only permits the delivery of laboratory reports but also the pickup of test specimens. This gives courier service an advantage over electronic data transfer.

PC's in the non-automated clinics and the Medical Records Department of the hospital are used for entering consultation reports as well as for other applications. The

word processing software used could be complemented with electronic communication but the match is less than ideal since 1) there is no compatible word processing package at all sites if the documents were to be transmitted in binary form, 2) there is no convenient, integrated means of addressing the documents, and 3) the destination computers usually have only rudimentary means of classifying, indexing and storing the transmitted documents. Users are unwilling to change their word processing software because they fear that transcription throughput will decrease.

7.4.3.2 Interface Development

In a large network with many nodes the only practical means to facilitate data interchange among the nodes is to employ a common data interchange standard such as EDIFACT, HL7 or MEDIX. In the field test, where there are only a few nodes and a few applications, the field test partners have defined their own interchange formats. For the laboratory reports, two interchange formats are used: one for non-automated clinics and another for clinics with electronic patient record systems. In the first case, the medical laboratory specifies that the format must be the same as it uses to print its reports locally. In the later case, the supplier of the electronic patient record system specifies its own proprietary input data format. It is not possible to use HL7, which is available with *Health Link*, for the first case because the report format requires the use of special form and font controls interspersed with the numeric data. It would be possible to use HL7 to transmit the laboratory results in the second case, however, the format translation process would add complexity to the interfaces.

It was discovered that there are subtle differences between the laboratory reporting system and the electronic patient record system. For example, laboratory reports can be copied to physicians other than the ordering physician. The electronic patient record system carries no corresponding fields and, consequently, no copies are transmitted. In another example, the laboratory usually performs more than the number of hematology tests requisitioned by the clinic for a patient because the laboratory equipment automatically conducts multiple tests on each specimen. When the test reports are delivered, automatic reconciliation is impossible because of the discrepancy between which tests were ordered and which were performed. Only standardization of the complementary end-user applications can resolve these types of

differences; no standardized interchange syntax can resolve differences at a semantic or structural level.

Health Link is capable of returning message acknowledgments to indicate that a data message has been delivered to the end-user computer. Although the communication link provides this level of confirmation, the end-user interfaces do not. There is no easy way to determine that all messages generated by the source system have been submitted to *Health Link* and that all messages delivered by *Health Link* have been accepted and loaded into the destination system. The only means to ensure this level of integrity is by making modifications to the end-user applications. This requires a higher degree of integration than a stand-alone network can provide. In the field test, this type of reconciliation must be performed manually.

7.4.3.3 Adapter Installation

Health Link adapter installation should require only a few minutes but it is complicated by lack of control over the end-user's environment. A supplier of network services is viewed by an end-user as being less important than a primary software supplier. This is particularly true in the context of an experimental endeavour. Although all of the field test partners have proven to be very co-operative, this translates to passive rather than active participation if tasks having higher priorities intervene.

At remote sites, the installer usually has little appreciation of the adapter's operating context. Even such issues as physical placement of equipment and telephone connections present difficulties. At one site, the equipment had to be relocated because of fan noise and the nuisance to staff occasioned by visiting field test personnel. At another site, where the adapter required connection to a local area network, it required several telephone calls and a significant effort to install a compatible Ethernet card in the adapter.

At smaller sites, there are comparatively few end-users who have training in their computer systems' operation; most only have experience in the use of application programs. For example, the clinics in the field test with electronic record systems rely on the application software vendor to provide system support.

Each new installation requires verification which involves another node in the network. During the field test, this requires two people, one at each node. Ideally, the

verification procedure should be automatic using two loop-back techniques: one where the original message is merely turned around, and another where a test message elicits an independent call-back with a reply message.

7.4.3.4 Network Maintenance

Operation of the field test network is complicated by not having a remote software update facility and by being unable to access nodes remotely to perform system maintenance. The prototype is not downward or upward compatible. In the field test, software updates must be synchronized and, if there is a message format change, the message bases at the sites must either be converted to the new format or be empty at the time of the update. In a larger network, ensuring synchronization becomes a major problem without a mechanism for the automatic release of software updates and version compatibility. Because changes in the software often result in changes in one or more of the data bases, data conversion must be part of the update procedure. A distributed version control mechanism is needed to ensure that updates are synchronized and successful.

Although messages can be used to determine the state of the network, these messages are appropriate for functional nodes only. If a node fails, it usually becomes unreachable. The field test, which has only a few nodes in comparison with a production network, is relatively easy to monitor and maintain on site. However, maintenance becomes more difficult with each additional node. If the network is to be supported by a third-party organization, the nodes should permit secure, remote, interactive access to essential maintenance functions.

7.5 Analysis

Perhaps the most important outcome of the field test is an exposure to the problems of network installation. The following recommendations are based on this experience.

1) Promote a small set of end-user applications to facilitate rapid deployment of the network. Only two of the four clinics in the field test have an electronic patient record system. Furthermore, the hospital and the medical laboratory have entirely different computer applications. Although it proved relatively easily to adapt

Health Link to each context, considerable time was invested in either developing an appropriate interface to the existing systems or writing appropriate reporting software for those sites having manual systems.

2) Minimize the number of organizations and people involved in establishing critical links in the network. Delays cascade as activities are often performed either sequentially or in synchronization. Two critical links in the field test were between the medical laboratory and the two clinics with an electronic patient record system. In order to establish a fully-automatic link between the laboratory and the clinics, there had be cooperation among five organizations.

3) Pay early attention to privacy/confidentiality agreements by the network support organization. Inevitably, the individuals installing the network will encounter confidential information at the sites. Delays are incurred while appropriate agreements are drawn up and approved.

4) Prepare installation procedures thoroughly before venturing into the field. No site installation in the field test worked on the first attempt. A number of problems occurred during installation ranging from referencing incorrect telephone numbers to improperly vectored communications interrupts. If the field test sites are any indication of the heterogeneity of organizations and health care computer systems, end-users cannot be expected to perform the installation procedure themselves.

5) Anticipate incompatible end-user applications, even though they appear to be complementary. The field test exposed problems with incompatible audit trails between the laboratory software and the electronic patient record software. Even though there may be close agreement in syntactic interpretation between two communicating systems, there remains a possibility of semantic and structural disagreement.

6) Establish sophisticated version control and software update procedures which can compensate automatically for local platform differences. A more sophisticated, automated procedure is required to accommodate the different platforms. The procedure should permit remote updates and provide fail-safe protection against update failures.

7.6 Conclusion

Although the field test has not met all of its objectives, it has successfully allowed the project team to 1) construct a demonstration network, 2) gather information regarding the types and frequencies of messages appropriate for electronic messaging by GP's, 3) collect reliability statistics for a small subnet, and 4) experience problems which might occur during installation and operation of a larger network.

The field test has fallen short of its objectives in determining the changes the technology has on office procedures. It has been unable to provide data for cost-benefit analyses. Both of these objectives can only be met after electronic data interchange has been fully integrated into the end-user procedures and applications. This level of integration has not yet been achieved given the short duration of the field test and the limited number of computerized, complementary end-user applications.

The software and technology has been successfully demonstrated. ***Health Link*** has proven to be highly reliable despite power fluctuations and communication failures. Experience gained during the field test and during approximately forty formal demonstrations highlight a need to:

- devise a method to update software and data bases automatically and remotely,
- develop a more sophisticated set of software installation procedures and tools,
- coordinate interface development with suppliers of end-user application software in order to promote a single, uniform product offering, and
- transfer the technology to more contemporary windows-based multi-tasking platforms.

These requirements are within easy reach of a production version of ***Health Link***.

8 Conclusion

8.1 Major Findings

The objectives of this research were to:

1) develop prototype messaging software to support wide area telecommunication for health care providers which relies minimally on the public switched telephone system and on existing heterogeneous health care systems,
2) implement techniques to promote data security and confidentiality within the network,
3) implement techniques to provide message tracking, authentication and non-repudiation,
4) examine the impact of network scaling on feasibility and performance,
5) resolve technical problems related to installing and managing the network,
6) investigate the applications appropriate to a messaging network,
7) analyze the behaviour and costs of a messaging network with respect to topology, and
8) test and evaluate a pilot network.

All of these objectives have been met in varying degrees. However, work to achieve some of these objectives has been suggestive of possible further research and areas of new research.

Software has been developed, tested and installed in a pilot network consisting of four clinics, one hospital and one medical laboratory. This field test has operated without any manual intervention at two of the field test sites for periods of time exceeding three months. At one laboratory and one clinic, laboratory tests are transferred automatically from one end-user application system to another. Moreover, participants in the field test have co-operated by responding to surveys used to determine the applications and volumes of inter-provider information exchange.

Simulation analyses of different network topologies are based on observations made in the field test and various benchmark tests. The results of the simulations carry

significance for the prototype as well as for other store-and-forward networks. The simulations are the basis for a cost model of a large scale network.

Health Link has developed over a period of nearly five years. In that time, telecommunication technology has advanced radically. Fibre optic networks have become widely accessible and, with them, high speed data communication protocols. Although communications technology has changed, many of the *Health Link* findings retain their importance for several reasons. First, telephone-based telecommunication will continue to be an important communication modality in rural areas for a few more years. Second, some of the replacement technologies like cellular telephones can make direct use of the *Health Link* implementation. Lastly, results from the simulation studies and the field test are fairly insensitive to the underlying telecommunication technology.

The most significant finding of the research is the success of the prototype. End-user computers can participate in a message store-and-forward network without invasive changes to their existing hardware or software. The network can be made to operate in a fully automatic and transparent manner to users in a way similar to a local area network. *Health Link* can support a generic wide area network which is suitable for use by health care providers, such as doctors, dentists and pharmacists, who have low volume communications requirements. It can serve providers who have disparate communications loads such as doctors and medical laboratories. *Health Link* has proven itself to be reliable by providing automatic messaging for periods of time exceeding a month without any user intervention.

Experience with the field test participants has provided useful information about the traffic patterns among the providers. Surveys with the users have indicated which applications they consider to be most useful. Problems with interface delivery, interface compatibility, site preparation, software version control, and network maintenance and monitoring have led to proposals for improvements in the application software, the *Health Link* software and the installation methodology.

The development of the prototype and the field test installation has provided a valuable understanding of and appreciation for the objectives and conceptual problems associated with telemedicine. With the acceptance of telecommunication in other forums such as education, finance and commerce, it is easy to assume that all networks are equally easy to implement, and that, from a outsider's perspective, the existing

networks have been installed without difficulty. However, this research has uncovered some organizational problems which are associated with health care providers.

The simulation studies and the field test surveys have shown that a network relying on the public switched telephone system is capable of supporting the traffic among physicians, medical laboratories, hospitals and payers. Both a client-server topology and a peer-to-peer topology have excess capacity for the estimated volume of text-oriented messages.

Of the client-server and peer-to-peer topologies, the client-server topology is the cheapest to operate. A large scale client-server network in Saskatchewan would cost approximately $50/month per physician to operate (including telecommunication and administration costs). This cost is found to be less than the administrative cost of distributing laboratory test results to physicians. It is also less than conventional third-party electronic mail services. The network operating cost for Saskatchewan is likely to be close to operating costs in Provinces in Canada or States in the United States with similar telecommunication facilities and tariffs and with similar distributions of population and health care resources. The operating costs are likely to be less in urban areas as compared to rural areas.

Of the client-server and peer-to-peer topologies, the peer-to-peer topology offers the higher degree of control over performance. Message priorities have a significant impact on message end-to-end transfer times. In client-server networks, messages are delayed mainly because the receivers must poll in order to pick up their messages at the server. As there is no way to notify the receiver of pending messages regardless of their priority, priorities have little effect on the respective end-to-end transfer times.

The field test has given rise to some general observations which are consistent with the experiences found in the Dutch 31 project. The end-users most likely to benefit from the network are those with electronic patient record systems. However, if the electronic patient record system does not reduce the workload, the benefits of the network are lost.

At field test sites, where end-user software interfaces with the network, installation times have exceeded their initial estimates by up to eight months. This is mainly due to the number of participants involved in the effort. Typically at each site, installation must be co-ordinated among the software developer, the network supplier and the end-user. Delays experienced by each party to the installation are felt by all.

Furthermore, if, after the interface is installed, it should fail, the end-user has difficulty in determining which party to approach thereby resulting in delays and a possible loss of messages.

The availability of appropriate computerized end-user application software is probably the most critical factor in implementing a network. The inertia of manual systems is difficult, perhaps impossible, to overcome if there is no application with obvious cost-benefits which is designed to employ the network without additional manual effort. The existing communication paths and manual procedures are highly developed and thoroughly ingrained. New technologies are most likely to be accepted if either there are few changes required to existing procedures or there is a great incentive to adapt. For example, the FAX machine is successful because it provides more rapid communication than the postal system without materially altering procedures. Electronic data interchange will be viewed as being a deficient alternative to the FAX as long as there is no appropriate end-user application software.

8.2 Future Research

This research is similar to several other projects which are at various stages of implementation. Only now are these pilot projects being extended to production environments. There is much yet to be developed and learned, both with respect to the area of health care telematics and *Health Link* itself. *Health Link* is a platform on which generic research can be conducted. *Health Link* can also be enhanced, packaged and distributed as a production network.

There is a need to define standards for data interchange, data security, information indexing and access privileges. Although many of the European initiatives have attempted to capitalize on open standards, the use of some of the existing ISO standards has delayed development and increased the degree of artifactual complexity. Some of the standards which have been accepted, such as EDIFACT, are only partially defined for the health care context. Existing standards are sometimes in competition. For example, although EDIFACT is becoming a European standard, HL7 is becoming a North American standard.

Efforts to define standards from the top down such as EUCLIDES and MEDIX seem to take longer. The resulting standards tend to be more complex and less accepted than are standards which grow with each additional application such as

EDIFACT and HL7. Competition among standards is well illustrated with the 15-year conflict between the OSI and the TCP/IP network architectures [102].

Standardization of methods in information indexing and access privileges has barely begun. Without common agreement in these areas, there is little hope for the often referenced *information highway* as there are no road maps and exit ramps.

There is a need to develop network-oriented applications which are useful and cost-beneficial. Without a polished repertory of application software, there is little incentive for the end-user to participate in a network. Many providers believe that information sharing could be a means to improve quality of care and reduce costs and many providers would be willing to invest in mature applications.

Further research should be conducted into the effect of telecommunication on health care management. This research would focus on the individual providers and on the health care system as a whole. For the individual providers, a controlled study would indicate the redistribution of operating costs, changes to office and client management procedures, and changes in information traffic. A study could be used to expose areas where there are gaps in procedures and information and where new applications could be developed.

The impact of data communication on the health care system as a whole is another area of future research. Co-ordination among all health care providers in a specific class or set of classes can potentially reduce costs and improve health care delivery. Applications which are not perceived as beneficial at the level of the provider may have system-wide value. Examples of this type of application are the proposed pharmacare telecommunication projects in Ontario and British Columbia which co-ordinate the drug treatment programs among physicians.

In addition to the generic research described above, *Health Link* itself offers opportunities for further development.

1) Although the *Health Link* gateway is capable of collecting status messages from each adapter, it is difficult to monitor and reconfigure individual adapters remotely. It would make network administration easier if a network manager could access adapter functions interactively from a remote site.

2) There is no facility to update network software or data files remotely. The network would be more manageable if updates could be transmitted electronically and applied automatically without manual intervention.

3) The adapter software could be transported to a wider range of operating environments. It is particularly important that the software should be executable from Microsoft Windows as most application software sold today for PC's operates in this environment. This conversion would remove any need to supply hardware adapters as front-ends for PC's.

4) A convenient application could be developed which promotes network use such as the electronic capture and management medical laboratory results for physicians. Laboratory results constitute approximately 30 percent of all external messaging requirements for physicians. Such an application could be accompanied by other less specialized applications such as electronic mail.

5) A system for information indexing could be developed for *Health Link* which facilitates searches across the network. This research could be based on network index servers, such as those supported by Internet Gopher, or on smart cards carried by the patient or client.

6) *Health Link* takes advantage of a primitive telecommunication technology, the public switched telephone system, which is widely accessible and low cost. Other more advanced telecommunication technologies are becoming accessible. Moreover, some of the cost advantages of the telephone system may vanish with the recent Bell Canada announcement of a new tariff schedule for businesses in Ontario [130]. *Health Link* could be maintained to stay in step with prevalent, cost-effective, telecommunication services.

Bibliography

[1] Nederlof F J. LIFELINE: A New Concept for Message Transfer in Health Care, based on X.400/EDI Protocols. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 41-57.

[2] Hasman A, Arnout P C, Boon P. The 3I Project. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 163-173.

[3] Rivest R L, Shamir A and Adelman L. A Method of Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* 1978,2: 120-126.

[4] Statistics Canada. *Canada Year Book* Catalogue 11-402E/1990. Ottawa: Statistics Canada, 1989.

[5] Health Canada. *Preliminary Estimates of Health Expenditures in Canada: Provincial/Territorial Summary Report 1987-1991*. Ottawa: Health Information Division, Health Canada, February 1994.

[6] Turley J L. *PCs Made Easy*. Berkeley: Osborne McGraw-Hill, 1989.

[7] Science Council of British Columbia. *Sharing Health Information: An Overview of Fifty Projects*. SPARK Health Sector Report. Burnaby: Science Council of British Columbia, 1992.

[8] Duisterhout J S, Hasman A and Salamon R (eds). *Telematics in Medicine*. Amsterdam: North-Holland, 1991.

[9] Pradley D. Telecommunications Standards. *MUMPS User Group Quarterly* 1990, XIX(3): 13-19.

[10] American Medical Informatics Association, Board of Directors. Standards for Medical Identifiers, Codes, and Messages Needed to Create an Efficient Computer-stored Medical Record. *Journal of the American Medical Informatics Association* 1994, January/February 1(1): 1-7.

[11] ASTM Committee E-31. *Designation: E1238 — Standard Specification for Transferring Clinical Observations Between Independent Computer Systems*, Draft Revision 4.2, August 2, 1991. Indianapolis: ASTM, 1991.

[12] Horii S C. The ACR - NEMA Standards: A Tutorial on Their Structure & Use. In: *SCAR 90, Computer Applications to Assist Radiology*, Arenson R and Friedenberg R (eds). Carlsbad, California: Symposia Foundation, 1990: 405-422.

[13] Workgroup for Electronic Data Interchange (WEDI). 1993 WEDI Report. Electronic manuscript available by ftp om cscns.com in directory pub/edi.

[14] The Impact of EDI & ANSI Standards on Administrative Cost Containment. *Healthcare Informatics* 1992, November 9(11): 90-96.

[15] Schmitt W A. Industry Watch. *Computers in Healthcare* 1991 March: 16-18.

[16] Spitzer P and Abelman A. HL7, MEDIX and the Pursuit of the Unified Standard. *Computers in Healthcare* 1990, November: 19-24.

[17] Kinkhorst O M, Hasman A and van Kesteren A C A. Standardization of Data-interchange in Health Care. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 313-326.

[18] Simborg D W. An Emerging Standard For Health Communications: The HL7 Standard. *Healthcare Computing & Communications* 1987, October, reprint.

[19] Weingarten J. Can Confidential Patient Information be Kept Private in High-Tech Medicine? *M.D. Computing* 1992, 9(2): 79-82.

[20] Robinson D M. A Legal Examination of Format, Signature and Confidentiality Aspects of Computerized Health Information. In: *MEDINFO 92*, Lun K C et al. (eds). Amsterdam: North-Holland, 1992: 1554-1560.

[21] Eichinger S and Pernul G. Design Environment for a Hospital Information System: Meeting the Data Security Challange. In: *MEDINFO 92*, Lun K C et al. (eds). Amsterdam: North-Holland, 1992: 1582-1588.

[22] Gritzlis D and Katsikas S. Data Confidentiality and User Access Rights in Medical Information Systems. In: *MEDINFO 92*, Lun K C et al. (eds). Amsterdam: North-Holland, 1992: 1566-1571.

[23] Hopkins R J. The Exeter Care Card: A CP8 based global health care record for the United Kingdom's National Health Service. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 367-381.

[24] Kluge E H W. Advanced Patient Records: some ethical and legal considerations touching Medical Information Space. *Methods in Informatics in Medicine* 1993, 32: 95-103.

[25] Moehr J R. Privacy and Security Requirements of Distributed Computer Based Patient Records. In: *Caring for Health Information Safety, Security and Secrecy* preprints of conference papers , IMIA Working Group 4, Heemskerk, The Netherlands, November 13-16, 1993: 36-41.

[26] Vandenhoute M. EUCLIDES' Security and Message Handling: A Critical Analysis. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 19-28.

[27] Fifield K J. Smartcards Outsmart Computer Crime. *Computers & Security* 1989, (8): 247-255.

[28] Cerf V and Kahn R. Selected ARPANET Maps 1969-1990. *Computer Communications Review* 1990, 20(5): 81-110.

[29] Mills D L INARC Chair. Internet Architecture Workshop: Future of the System Architecture and TCP/IP Protocols. *Computer Communications Review* 1990, 20(1): 6-17.

[30] Comer D E. *Internetworking With TCP/IP: Principles, Protocols, and Architecture*. Englewood Cliffs: Prentice-Hall, 1988.

[31] Ishida H. Internetworking. *Communications of the ACM* 1993, 36(8): 28-30.

[32] Soderstrom L. *The Canadian Health System*. London: Croom Helm, 1978.

[33] Rachlis M and Kushner C. *Second Opinion: What's Wrong with Canada's Health-Care System and How to Fix it*. Toronto: Collins Publishers, 1989.

[34] Williams K J and Donnelly P R. *Medical Care Quality and the Public Trust*. Chicago: Pluribus Press, 1982.

[35] Dickinson H D and Hay D A. The Structure and Cost of Health care in Canada. In: *Sociology of Health Care in Canada*, Bolaria B S and Dickinson H D (eds). Toronto: Harcourt Brace Jovanovich Canada Inc., 1988: 51-73.

[36] Ontario Ministry of Health. ***Request for Qualifying Information: Ontario Drug Programs Systems Requirements.*** Toronto: Ontario Ministry of Health, July 1992.

[37] Rice D (ed). Technical column: Health Information Technology and Communications. *NHIC Newsletter* 1993, 4(2): 7.

[38] British Columbia Ministry of Health, Pharmacare Branch. ***Overview for Prospective Suppliers on the Pharmacy Network Project.*** Victoria: British Columbia Ministry of Health, 1993.

[39] Computer linkage of B.C. pharmacies should cut abuses. Victoria *Times Colonist* 1993, September 4: B5.

[40] B.C. remedies payment plan by putting doctors on-line. *The Financial Post*, Special Report 1988, June 4-6: 39.

[41] Farley J D and Syrja L M. A Communicable Disease Surveillance System for British Columbia. In: *ITCH90 Improving Community Health Through Applied Technology* Coward J H (ed). Victoria: University of Victoria, 1990: 14-16.

[42] Ameritech advertising brochure, Chicago, circa 1993.

[43] Ameritech demonstration held at Langley Memorial Hospital, October 15, 1993.

[44] Moehr J R. Health Link. Health Development Fund - Science Council of B.C. Grant Application, November, 1991.

[45] Ament A and L'Ortye M. An Economic Evaluation of Electronic Communication in Health Care (31-Project). In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 65-78.

[46] Arnou P G et al. A Computer Communication Network for Hospitals and General Practice. In: *MEDINFO 89*, Barber B et al. (eds). Amsterdam: North-Holland, 1989: 1121-1124.

[47] van Lierop D et al. Feasibility of a regional EDI-network in Dutch health care. In: *MEDINFO 92*, Lun K C et al. (eds). Amsterdam: North-Holland, 1992: 89-93.

[48] Branger P J and Duisterhout J S. The Evaluation of the Communication Project Apeldoorn (COPA). In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 29-40.

[49] Branger P J et al. Electronic communication between providers of primary and secondary care. *British Medical Journal* 1992, 305: 1068-1070.

[50] Leguit F. The Case: Leiden University Hospital Information System. In: *Towards New Hospital Information Systems*, Bakker A et al. (eds). Amsterdam: North-Holland, 1988: 59-64.

[51] Koens M L. Experiences with EDI in Dutch Health Care. In: *MEDINFO 92*, Lun K C et al. (eds). Amsterdam: North-Holland, 1992: 94-98.

[52] Stokes A V and Oswin K J. The United Kingdom National Health Service OSI Demonstrator Project. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 287-297.

[53] Palmer G and Willshere J. Exchanging Healthcare Information: A Case Study in Oxford RHA. *The British Journal of Healthcare Computing & Information Management* 1993, 10(8): 29-30.

[54] Oxford RHA. *GP Links Project Newsletter* 1992, August (2).

[55] Oxford RHA. *Oxford RHA's G.P. Systems Links Project* 1992, (bd431).

[56] Athwall P S. Messaging: Open Systems in Practice. In: *MEDINFO* 89, Barber B et al. (eds). Amsterdam: North-Holland, 1989: 1093-1096

[57] Mason S. NHS Family Practitioner Service Data Communications Strategy. In: *MEDINFO 89*, Barber B et al. (eds). Amsterdam: North-Holland, 1989: 1104-1107.

[58] Shining Stars. *Health Informatics Europe* 1993, June 1(2): 18-21.

[59] Europe AIMS at Telemedicine. *Healthcare Informatics* 1993, April 10(4): 62-64.

[60] de Moor G J E. A Standard Solution for the Exchange of Clinical Laboratory Data: EUCLIDES. In: *Telematics in Medicine*, Duisterhout J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 259-280.

[61] EUCLIDES Foundation International brochure, Zellik, Belgium, circa 1992.

[62] Schosser R. *TELEMED II - PROJECT: Short Description*. Overview of project proposal, 1991.

[63] *Developing a platform for medical data interchange*. Information Management Centre brochure, Birmingham, U.K., circa 1992.

[64] Bahensky J. University's Statewide Network Links Rural Doctors. *Healthcare Informatics* 1992, March: 10-12.

[65] Schultz S. Presentation at the IMIA Working Conference on Telematics in Medicine, Rotterdam, The Netherlands, 18-21 November, 1990.

[66] Salkeld S. *Doctors Online Communication System (DOCS) Evaluation Report*. BC Systems Corporation internal report, Victoria, Canada, 1990.

[67] Thomson B, Tomczak M and Salkeld S. GVHS Electronic Transmission of Health Information in the Greater Victoria Area. In: *ITCH'92: Building Partnerships in Community Health Through Applied Technology*, Scott L R (ed). Victoria: University of Victoria: 1992: 148-151.

[68] BC InfoHealth Ltd. Partnership in Networking: First Steps to Province-Wide Communication Network. Corporate newsletter, Winter 1990/1991.

[69] Tanenbaum A S. *Computer Networks*, 2nd Ed. Englewood Cliffs: Prentice Hall, 1988.

[70] Bush R. FidoNet: Technology, Tools, and History. *Communications of the ACM* 1993, 36(8): 31-35.

[71] Parnas D L. On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM* 1972,15(12): 1053-1058.

[72] Parnas D L. Designing Software for Ease of Extension and Contraction. *IEEE Transactions on Software Eng.* 1979,SE-5(2): 128-138.

[73] Telecom Canada. Datapac 3000 Overview Guide. Telecom Canada, 1984.

[74] Sequiter Software Inc. *Code Base 4.1 User's Guide*. Edmonton: Sequiter, 1990.

[75] South Mountain Software, Inc. *Essential Communications Owner's Manual*, Ver 3.0. South Orange, NJ: South Mountain, *circa* 1989.

[76] Symantec Corporation. *Zortech C++ Compiler Guide*. Cupertino: Symantec Corp., 1991.

[77] Dijkstra E W. Structured Programming. In: *Classics in Software Engineering*, Yourdon E N (ed). Yourdon Press, New York, 1979.

[78] Stallings W. *Handbook of Computer Communications Standards, Volume I: The Open Systems Interconnection (OSI) Model and OSI-Related Standards*. New York: Macmillan Pub., 1987.

[79] Shoja G C. Programming technique introduced in class at the University of Victoria, Victoria, 1991.

[80] Chou G T. *dBASE III Plus Handbook*, 2nd ed. Indianapolis: Que Corp., 1986.

[81] Comer D E and Stevens D L. *Internetworking with TCP/IP vol III: Client-Server Programming and Applications*. Englewood Cliffs: Prentice-Hall, 1993.

[82] Digital Equipment Corporation. *VT100 User Guide*, 2nd ed., EK-VT100-UG-002. Maynard: Digital Equipment Corp., 1979.

[83] Campbell J. *C Programmer's Guide to Serial Communications*. Carmel: Howard W. Sams, 1987.

[84] Lee K and Chanson S T. A New File Transfer Protocol for Telephone Lines. In: *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Proceedings, Vol. 2. Washington, DC: *IEEE*, 1993.

[85] Frontier Technologies Corporation. Frontier's OSI/GOSIP Offerings. Product description report. Mequon, WI, USA: Frontier Technologies, 1991.

[86] Duisterhoot J S, Hasman A and Salamon R (eds). *Telematics in Medicine*. Amsterdam: North-Holland, 1991.

[87] Storer J A. *Data Compression: Methods and Theory*. Rockville: Computer Science Press, 1988.

[88] Beckett B. *Introduction to Cryptology*. Palo Alto: Blackwell Scientific Pub., 1988.

[89] Witten I H, Radford M N and Cleary J G. Arithmetic Coding For Data Compression. *Communications of the ACM* 1987 30(6): 520-540.

[90] Newman D B Jr, Omura J K and Pickholtz R L. Public Key Management for Network Security. *IEEE Network Magazine* 1987,1(2): 11-16.

[91] Jueneman R R. Electronic Document Authentication. *IEEE Network Magazine* 1987,1(2): 17-23.

[92] Huthnance E D and Warndof J. On Using Primes for Public Key Encryption Systems. *Applied Mathematics Letters* 1988, 1(3): 225-227.

[93] Mitchell C, Rush D and Walker M. A Remark on Hash Functions for Message Authentication. *Computers & Security* 1989, 8: 55-58.

[94] Highland H J. Random Bits & Bytes. *Computers & Security* 1989, 8: 274-282.

[95] Enslow P H Jr. What is a "Distributed" Data Processing System? *Computer* 1978, 11: 13-21.

[96] Lamport L. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM* 1978, 21(7): 558-565.

[97] Tang A and Scoggins S. *Open Networking with OSI*. Englewood Cliffs: Prentice Hall, 1992.

[98] Harrington, J J. IEEE P1157 Medical Data Interchange (MEDIX): Application of open systems to health care communications. *Topics in Health Record Management* 1991, 11(4): 45-58.

[99] Health Level Seven, Inc. *Health Level Seven: An application protocol for electronic data exchange in healthcare environments*, Version 2.1. Chicago: Health Level Seven, 1990.

[100] Duisterhoot J S and Branger P J. EDIFACT Message Handling and Integration into Applications in the COPA Project. In: *Telematics in Medicine*, Duisterhoot J S, Hasman A and Salamon R (eds). Amsterdam: North-Holland, 1991: 151-162.

[101] The Networking Standards Evolution: Toward a Real Electronic Medical Record. *Computers in Healthcare* 1990, Feb: 18-21.

[102] Rose M T. *The Internet Message: Closing the Book with Electronic Mail.* Englewood Cliffs: P T R Prentice Hall, 1993.

[103] Simonic K M and Gell G. A general approach to the strategy of integrating healthcare applications using HL7. In: *MEDINFO 92: Proceedings of the Seventh World Congress on Medical Informatics*, Lun K.C. et al. (eds). Amsterdam: North-Holland, 1992: 1432-1436.

[104] Banks J and Carson J S II. *Discrete-Event System Simulation*. Englewood Cliffs: Prentice-Hall, 1984.

[105] Gross D and Harris C M. *Fundamentals of Queueing Theory*, 2nd ed. New York: John Wiley & Sons, 1985.

[106] Hillier F S and Lieberman G J. *Introduction to Operations Research*, 3rd ed. Oakland: Holden-Day Inc., 1980.

[107] Culter J. Personal communication. Victoria, June, 1991.

[108] Medhi J. *Stochastic Models in Queueing Theory*. San Diego: Harcourt Brace Jovanovich Pub., 1991

[109] Kleinrock L. *Queueing Systems, Volume I: Theory*. New York: John Wiley & Sons, 1975.

[110] Zahorjan J et al. Balanced Job Bound Analysis of Queueing Networks. *Communications of the ACM* 1982, 2: 134-141.

[111] Banks J, Carson J S II and Sy J N. *Getting Started with GPSS/H*. Annandale: Wolverine Software Corp., 1989.

[112] Henriksen J O and Crain R C. *GPSS/H Reference Manual*. Annandale: Wolverine Software Corp., 1989.

[113] Briley B E. *Introduction to Telephone Switching*. Reading: Addison-Wesley Pub. Co., 1983.

[114] Fox R L. *Optimization Methods for Engineering Design*. Reading: Addison-Wesley Pub. Co., 1971.

[115] Villasante J. Telematic systems for health in Europe: A look towards the future. *Health Informatics Europe* 1993, 1(2): 10-12.

[116] Sarasohn-Kahn J. Europe AIMs at Telemedicine. *Healthcare Informatics* 1993, 10(4): 62-64.

[117] Canadian Hospital Association. *Canadian Hospital Directory*. Ottawa: Canadian Hospital Association, 1992.

[118] Canadian Medical Association. *Thirty-Eighth Annual Canadian Medical Directory*. Don Mills: Southam Business Communications Inc., 1992.

[119] SASKTEL. *Estevan — Weyburn & District*, March 1991. Direct West Yellow Pages, 1991.

[120] SASKTEL. *Melville — Yorkton and District*, June 1992. Direct West Yellow Pages, 1992.

[121] SASKTEL. *North Battleford and District*, November 1992. Direct West Yellow Pages, 1992.

[122] SASKTEL. *Prince Albert*, October 1992. Direct West Yellow Pages, 1992.

[123] SASKTEL. *Regina City*, May 24, 1992. Direct West Yellow Pages, 1992.

[124] SASKTEL. *Regina District*, May 1992. Direct West Yellow Pages, 1992.

[125] SASKTEL. *Saskatoon City*, September 27, 1992. Direct West Yellow Pages, 1992.

[126] SASKTEL. *Saskatoon District*, September 1992. Direct West Yellow Pages, 1992.

[127] SASKTEL. *Swift Current and District*, April 1992. Direct West Yellow Pages, 1992.

[128] SASKTEL. *Moose Jaw & District*, January 1993. Direct West Yellow Pages, 1993.

[129] Statistics Canada. *Urban Areas: Population and Dwelling Counts* Catalogue 93-305. Ottawa: Statistics Canada, 1992: 138-154.

[130] Bell to bill local calls by minute. Victoria *Times Colonist* 1993, December 8: A9.

Appendix A

GPSS/H Source Listing of the Discrete-Event Model for a Mesh Cluster

```
* BALNODE.GPS
* Written by J. McDaniel, February 27, 1993
*       Discrete-Event Model of a balanced Mesh Cluster
*
*       Input file record format (file name must have a ".ctl" extension):
*               LineIdle time,
*               RetryWait time,
*               Emulation or virtual channel used for msg pickup and delivery,
*               Number of adapters in cluster,
*               Length of msg text (characters),
*               Total mean rate of message creation at a given node (for all other nodes in cluster),
*               Message priority (1 = Low, 2 = Medium, 3 = High)
*               Message acknowledgments required (0 = false, 1 = true)
*       Spaces are used to delimit fields on the control record.
*
*       An input file can consist of one or more control records.  The first record starts the model from a zero and empty state.
*       Subsequent records do not re-initialize the model, just the statistics.
*
*       The output file (with a root name the same as the control file and an extension of ".out") consists of the input parameters
*       followed by the ratio of time which the port is busy, the ratio of time which the port does not answer, the mean system
*       time for message pickup, the mean system time for message delivery, the mean system time for message transmission, the
*       mean time a message is prepared, the number of messages sent by the adapter to the driver (not counting message
*       acknowledgments), and the number of messages sent by the driver to the adapter (not counting message acknowledgments).
*
            SIMULATE
            REALLOCATE      COM,20000
            NOXREF
            UNLIST          MACX
*
* reserved facilities
*
FPOLL       EQU             1,F             //polltimer gate throttle
FPORT       EQU             2,F             //porttimer gate throttle
FPREP       EQU             3,F             //preptimer gate throttle
LCHNL       EQU             4,F             //local channel (half-duplex)
AQC03       EQU             1,C             //adapter C03 queue
```

```
DQC03       EQU     2,C     //driver C03 queue
POLLTIMR    EQU     1,L     //poll timer (cyclic)
PORTTIMR    EQU     2,L     //incoming port timer (cyclic)
PREPTIMR    EQU     3,L     //prep wait timer
RLISTIMR    EQU     4,L     //relisten timer
RSETIMR     EQU     5,L     //port reset timer
CPU         EQU     6,L     //CPU preemption
*
* time constants (seconds)
*
SCANDIR     SYN     20      //directory poll cycle (POLLTIMR)
IDLEPORT    SYN     13      //Review idle ports cyc'e (PORTTIMR)
PREPWAIT    SYN     5       //wait after last activity (PREPTIMR)
INTRPREP    SYN     2       //interprocess wait
DIALTIM     SYN     27      //time to dial & get carrier
HUPTIM      SYN     7       //time to hangup
ADDRTIM     SYN     1       //address, address verify
ENCRTIM     SYN     3       //en/decrypt 1000 byte msg
SIGNTIM     SYN     1       //signature generation/encryption
BUSYTIM     SYN     15      //time to wait if connection busy
REACTIM     SYN     10      //time before dest starts FDX comms
RLISTIM     SYN     2       //time to execute relisten script at port
RLISCYCL    SYN     5       //wait before review of port
RSETIM      SYN     10      //reset port time
NANSTIM     SYN     42      //¬&LINEIDLE=time to wait if no answer
CONNOVH     SYN     2       //connection overhead (sec)
INACTIVE    SYN     -1      //set timer to inactive
LOWHOLD     SYN     600     //hold time for low priority msgs
MEDHOLD     SYN     300     //hold time for medium priority msgs
HIHOLD      SYN     0       //hold time for high priority msgs
*
* miscallaneous indices
*
NUMTIMR     SYN     5       //# of timers per node
LOWPRIO     SYN     1       //low priority
MEDPRIO     SYN     2       //medium priority
```

```
HIPRIO      SYN     3           //high priority
EMU         SYN     0           //emulation channel
VIRT        SYN     1           //virtual channel
ISBSY       SYN     1           //ACHNL,DCHNL,CHNL busy
TRUE        SYN     1           //true
FALSE       SYN     0           //false
*
* transaction (msg) parameter indices
*
JDEST       SYN     1           //c03r.remoteAddr
JCONNREQ    SYN     2           //c03r.numConnReq
JPRIO       SYN     3           //priority rating
JMSGLEN     SYN     4           //msg length
JACKREQ     SYN     5           //ack requested
JISACK      SYN     6           //is an ack
JFRST       SYN     7           //first msg on a connection
JKEYTIM     SYN     1           //c03r.keyStmp
JTRIGTIM    SYN     2           //c03r.trigStmp
JTIM        SYN     1           //parm used as timer index
*
* miscelaneous simulation setup constants
*
TESTTRX     SYN     6000        //term counter for test
*
* miscelaneous transmission constants
*
MSGOVH      SYN     400         //msg char overhead (hdr+conn/disconn)
XSPEED      SYN     2400        //modem speed (2400 bps)
VSPEED      SYN     100000      //virtual channel speed (100,000 bps)
ESPEED      SYN     9600        //emulation channel speed (9600 bps)
XMITEFF     SYN     45          //transmission efficiency for HDX (45%)
XMITEFF2    SYN     40          //FDX (incremental) efficiency
*
* ampervariables
*
            REAL    &XEFF,&PORTBUSY,&BEGBUSY,&NOANS,&BEGNOANS
```

```
                REAL            &KEYTIM,&NEWKEY,&BUSYRAT,&NOANSRAT,&XEFF2
                INTEGER         &EORV,&EVXRAT(2),&MSGLEN,&MEANRAT
                INTEGER         &NUMADPTR,&PRIOWAIT(3),&COMMRAT
                INTEGER         &LINEIDLE,&RTRYWAIT,&ACKREQ,&PRIO
                INTEGER         &NUMSNT,&NUMRCV,&ACHNLBSY,&DCHNLBSY,&CHNLBSY
                INTEGER         &NUMTRF
*
* matrices
*
TIMRS           MATRIX          MH,1,NUMTIMR                    //timers (counters)
*
* table definitions
*
QTPICKUP        QTABLE          QPICKUP,20,5,200                //pickup poll+prep times
QTSND           QTABLE          QSND,300,5,200                  //queuing+sending times
QTDELIV         QTABLE          QDELIV,20,5,200                 //recving prep+delivery poll
TCONNREQ        TABLE           PH(JCONNREQ),0,1,20             //number of connection requests
TNUMTRF         TABLE           &NUMTRF,0,1,20                  //# of msgs xferred during connection
*
* calculated constants
*
                LET             &XEFF=XMITEFF/100.0             //transmit efficiency (HDX)
                LET             &XEFF2=XMITEFF2/100.0           //transmit efficiency FDX (incremental)
                LET             &EVXRAT(1)=ESPEED/10            //emu ch rate (9600 bps)
                LET             &EVXRAT(2)=VSPEED/10            //virt ch rate (100,000 bps)
                LET             &COMMRAT=XSPEED/10              //comm line rate (2400 bps)
                LET             &PRIOWAIT(LOWPRIO)=LOWHOLD      //low priority wait
                LET             &PRIOWAIT(MEDPRIO)=MEDHOLD      //medium priority wait
                LET             &PRIOWAIT(HIPRIO)=HIHOLD        //high priority wait
*
**********************************
* macros
**********************************
*
* LGATEONE(LogicSwitch,resourceThrottle)
* latched logic gate which allows only one trx through and
```

```
* resets logic switch after
*
LGATEONE    STARTMACRO
            SEIZE           #B
            GATE LS         #A
            LOGIC R         #A
            RELEASE         #B
            ENDMACRO
*
* PREP(PrepTime)
* Perform one stage of msg prep'ing, then
* reset msg prep timer.
*
PREP        STARTMACRO
            GATE LS         PREPTIMR        //gate holds back msg until ready
            LOGIC R         PREPTIMR        //close gate
            GATE LR         CPU             //gate if CPU is already in use
            LOGIC S         CPU             //prempt CPU
            ADVANCE         #A              //prep msg
            LOGIC R         CPU             //release CPU
            MSAVEVAL        TIMRS,1,PREPTIMR,INTRPREP,MH    //start prep wait timer
            ENDMACRO
*
* SETPTIMR()
* Set the preptimer and return to next statement
*
SETPTIMR    STARTMACRO
            TEST E          &ACHNLBSY,FALSE,*+4     //adapter channel must be free
            TEST E          &DCHNLBSY,FALSE,*+3     //driver channel must be free
            LOGIC R         PREPTIMR        //reset gate
            MSAVEVAL        TIMRS,1,PREPTIMR,PREPWAIT,MH    //start prep wait timer
            ENDMACRO
*
* STPPTIMR()
* Stop the preptimer and return to next statement
*
```

```
STPPTIMR    STARTMACRO
            LOGIC R          PREPTIMR                          //reset gate
            MSAVEVAL         TIMRS.1.PREPTIMR.INACTIVE.MH      //wait indefinitely
            ENDMACRO
*
* AREKEY()
* Connection request failed.  Release adapter port and rekey
* all transactions to same dest which have a key time less
* than &KEYTIM
*
AREKEY      STARTMACRO
            MSAVEVAL         TIMRS.1.RSETIMR.RSETIM+HUPTIM.MH  //hangup and reset port
            ASSIGN           JCONNREQ.PH(JCONNREQ)+1.PH        //increment conn requests
            BLET             &KEYTIM=AC1+FLT(&RTRYWAIT)*_
                             (PH(JCONNREQ)*FRN3+1.0)           //save new keytime for rekeying
            BLET             &NEWKEY=&KEYTIM                   //keep a copy for later changes
            ASSIGN           JKEYTIM.&KEYTIM.PL                //advance key time
            UNLINK E         AQC03.ARKEY.1.JDEST$PH.PH(JDEST)._
                             ARKEYEND                          //requeue other msgs to same dest
            LINK             AQC03.JKEYTIM$PL                  //requeue msg in C03
            ENDMACRO
*
* DREKEY(time)
* Connection request failed.  Release driver port and rekey
* all transactions to same dest which have a key time less
* than &KEYTIM
*
DREKEY      STARTMACRO
            BLET             &DCHNLBSY=FALSE                   //make driver available
            BLET             &CHNLBSY=FALSE                    //make port available
            ADVANCE          #A                                //busy or noans wait
            ASSIGN           JCONNREQ.PH(JCONNREQ)+1.PH        //increment conn requests
            BLET             &KEYTIM=AC1+FLT(&RTRYWAIT)*_
                             (PH(JCONNREQ)*FRN3+1.0)           //save new keytime for rekeying
            BLET             &NEWKEY=&KEYTIM                   //keep a copy for later changes
            ASSIGN           JKEYTIM.&KEYTIM.PL                //advance key time
```

```
                UNLINK E        DQC03,DRKEY,1,JDEST$PH,PH(JDEST)    //requeue other msgs to same dest
                LINK            DQC03,JKEYTIM$PL                    //requeue msg in C03
                ENDMACRO
*
************************************
* timer routine
************************************
*
                GENERATE        1                                   //generate a trx every second
*
* Iterate through all timers.
* If a timer has expired, set then reset logic flag.
* If the expired timer is a cyclic timer, reset the timer.
* Note: if same timer expires and is reset within the same second,
* reduce the second setting by a second due to timer granularity.
*
                ASSIGN          JTIM,NUMTIMR,PH
*
TIMRLP          TEST E          &CHNLBSY,FALSE,L1                   //channel must be available
                UNLINK LE       DQC03,DSND,1,JKEYTIM$PL,AC1,L1      //find entry at driver
                BLET            &CHNLBSY=TRUE                       //port is now busy
                BLET            &DCHNLBSY=TRUE                      //driver is busy
L1              TEST NE         MH(TIMRS,1,PH(JTIM)),0,TIMRLP1      //timer expired?
                TEST G          MH(TIMRS,1,PH(JTIM)),0,TIMRLP2      //timer active?
                MSAVEVAL        TIMRS-,1,PH(JTIM),1,MH              //decrement timer
                TRANSFER        ,TIMRLP6
*
TIMRLP1         MSAVEVAL        TIMRS,1,PH(JTIM),INACTIVE,MH        //timer now inactive
                TEST NE         PH(JTIM),RLISTIMR,TIMRLP4           //relisten procedures
                TEST NE         PH(JTIM),RSETIMR,TIMRLP5            //reset port procedures
                TEST E          PH(JTIM),PREPTIMR,L2                //if preptimer
                TEST E          &CHNLBSY,FALSE,TIMRLP6              //and port busy
                TEST NE         FNU$LCHNL,0,TIMRLP6                 //or local channel busy,don't open gate
L2              LOGIC SPH(JTIM)                                     //set timer flag
                TRANSFER        ,TIMRLP6
*
```

```
* Reset cyclic timers: poll timer and port timer.
*
TIMRLP2     TEST E          PH(JTIM),POLLTIMR,TIMRLP3          //only poll timer here
            LOGIC RPH(JTIM)                                    //reset timer
            MSAVEVAL        TIMRS,1,POLLTIMR,SCANDIR-2,MH      //restart poll cycle for next pull op
            TEST NE         FNU$FPOLL,0,TIMRLP6                //anything pending?
            MSAVEVAL        TIMRS+,1,POLLTIMR,SCANDIR,MH       //wait two cycles: dir and pull ops
            TRANSFER        ,TIMRLP6
*
TIMRLP3     TEST E          PH(JTIM),PORTTIMR,TIMRLP6          //only port timer here
            LOGIC RPH(JTIM)                                    //reset timer
            TEST E          &CHNLBSY,FALSE,L3                  //is port available?
            UNLINK LE       AQC03,ASND,1,JKEYTIM$PL,AC1,L3     //find entry in C03 and send
            BLET            &CHNLBSY=TRUE                      //port is now busy
            BLET            &ACHNLBSY=TRUE                     //adapter is busy
L3          MSAVEVAL        TIMRS,1,PORTTIMR,IDLEPORT-2,MH     //restart incoming port cycle timer
            TRANSFER        ,TIMRLP6
*
* Perform special procedures for resetting and relistening at port
*
TIMRLP4     MSAVEVAL        TIMRS,1,RLISTIMR,RLISCYCL,MH       //set next relisten cycle time
            TEST E          &CHNLBSY,TRUE,TIMRLP6              //port must still be busy
            TEST E          (&ACHNLBSY+&DCHNLBSY),FALSE,TIMRLP6 //adapter and driver channels must be idle
STPPTIMR    MACRO                                              //relisten - stop prep wait timer
            ADVANCE         RLISTIM                            //hangup and relisten
            BLET            &NOANS=&NOANS+AC1-&BEGNOANS        //subtotal noAns time
            BLET            &CHNLBSY=FALSE                     //make port available
SETPTIMR    MACRO                                              //start prep wait timer
            TRANSFER        ,TIMRLP6
*
TIMRLP5     BLET            &ACHNLBSY=FALSE                    //adapter channel available
            BLET            &DCHNLBSY=FALSE                    //driver channel available
            BLET            &PORTBUSY=&PORTBUSY+AC1-&BEGBUSY   //subtotal busy time
            BLET            &BEGNOANS=AC1                      //start timing no answer period
SETPTIMR    MACRO                                              //start prep wait timer
*
```

```
* End of loop
*
TIMRLP6     LOOP            JTIM$PH,TIMRLP                  //decrement timer index
            TERMINATE
*
************************************
* adapter routine
************************************
*
            GENERATE        RVEXPO(2,&MEANRAT).....7PH,2PL  //generate for all adapters
            ASSIGN          JDEST,FIX(FRN4*(&NUMADPTR-1))+1,PH  //assign destination
            ASSIGN          JPRIO,&PRIO.PH                  //medium priority msg
            ASSIGN          JMSGLEN,&MSGLEN.PH              //chosen msg length
            ASSIGN          JACKREQ,&ACKREQ.PH              //ack requested
            ASSIGN          JISACK,FALSE,PH                 //not an ack itself
            ASSIGN          JFRST,FALSE,PH                  //not yet a "first" msg
            BLET            &BUSYRAT=&PORTBUSY/C1           //ratio is busy time/total run time
            BLET            &NOANSRAT=&NOANS/C1             //ratio is noAns time/tot run time
*
* Transfer a msg to the adapter over the emulation or virtual channel...
*
            QUEUE           QPICKUP                         //pickup to staging time
            GATE LS         POLLTIMR                        //hold until polled - leave gate open
            TEST E          &EORV,VIRT,EMUPCKUP             //if virtual channel,
LGATEONE    MACRO           PORTTIMR,FPORT                  //hold each msg until port is reviewed
            TRANSFER        ,L4                             //else
EMUPCKUP    GATE LS         PORTTIMR                        //hold msgs till port is reviewed
L4          SEIZE           LCHNL                           //capture local channel
STPPTIMR    MACRO                                           //stop prep wait timer
            ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                            /(&EVXRAT(&EORV+1)*&XEFF)       //send msg to adapter
            RELEASE         LCHNL                           //free local channel
SETPTIMR    MACRO                                           //start prep wait timer
*
* Message stored in msg base,
* now prep msg...
```

```
*
ARTNACK     SEIZE       FPREP                                  //only prep one msg at a time
PREP        MACRO       ADDRTIM                                //address msg
PREP        MACRO       ENCRTIM                                //encrypt msg
PREP        MACRO       SIGNTIM                                //signature generation
PREP        MACRO       SIGNTIM                                //signature encryption
            RELEASE     FPREP                                  //finished prep'ing
SETPTIMR    MACRO                                              //restart prep timer
            TEST E      PH(JISACK),FALSE,L5                    //ack's are not part of QPICKUP
            DEPART      QPICKUP
*
* Message is prep'ed,
* now put in operations queue
*
L5          ASSIGN      JKEYTIM,AC1+&PRIOWAIT(PH(JPRIO)),PL    //set key stamp
            ASSIGN      JTRIGTIM,PL(JKEYTIM),PL                //save trigger time
            TEST E      PH(JISACK),FALSE,L6                    //collect stats on msg only
            QUEUE       QSND                                   //adapter-to-adapter time
L6          LINK        AQC03,JKEYTIM$PL                       //C03 queue in order of key stamp
*
* Message and port both available,
* now send...
*
ASND        GATE LR     CPU                                    //wait until CPU is free
            BLET        &NUMTRF=0                              //number transferred during connection
STPPTIMR    MACRO                                              //stop prep wait timer
            BLET        &BEGNOANS=AC1                          //no longer listening at port
            ADVANCE     BUSYTIM                                //dial out (min investment)
            BLET        &NOANS=&NOANS+AC1-&BEGNOANS            //port is now busy dialing
            BLET        &BEGBUSY=AC1                           //port goes busy
*
            TEST LE     FRN5,&BUSYRAT,ARINGING                 //destination is busy
AREKEY      MACRO                                              //rekey msgs to same dest
*
ARINGING    TEST LE     FRN6,(&NOANSRAT/(1.0-&BUSYRAT)),_      //destination is originate only
                        ASNDFRST
```

```
                ADVANCE     (NANSTIM-BUSYTIM+&LINEIDLE)      //wait to determine that line's idle
AREKEY          MACRO                                        //rekey msgs to same dest
*
* Rekey the trx going to same destination because of a connection
* failure.  Only rekey those whose trx with key stamps less than
* the future key stamp.  Increment the number of conn requests
* if the trx key time is less than current time.
*
ARKEY           TEST L      PL(JKEYTIM),&KEYTIM,ARKEYEND     //key need changing?
                BLET        &NEWKEY=&NEWKEY+1                //ensure keys remain ascending
                TEST LE     PL(JKEYTIM),AC1,L7               //incr conn requests only for
                ASSIGN      JCONNREQ,PH(JCONNREQ)+1,PH       //those trx older than now
L7              ASSIGN      JKEYTIM,&NEWKEY,PL               //set new key time
                UNLINK E    AQC03,ARKEY,1,JDEST$PH,PH(JDEST) //requeue other msgs to same dest
ARKEYEND        LINK        AQC03,JKEYTIM$PL                 //requeue msg in C03
*
* Connection is complete .. now sending transactions.
*
ASNDFRST        ASSIGN      JFRST,TRUE,PH                    //msg causes connection to be made
                ADVANCE     (DIALTIM-BUSYTIM)                //wait for carrier
                UNLINK E    DQC03,DSNDFDX,1,JDEST$PH,PH(JDEST) //find any entry from same dest
                TRANSFER    ,ASNDNXT
ASNDFDX         BLET        &ACHNLBSY=TRUE                   //adapter is sending
                GATE LR     CPU                              //wait until CPU is free
                ADVANCE     REACTIM                          //pause to search queue
ASNDNXT         TABULATE    TCONNREQ                         //tabulate # conn requests
                TEST E      PH(JISACK),FALSE,L8              //collect stats on msg only
                QUEUE       QTRF
                BLET        &NUMSNT=&NUMSNT+1                //tally number msgs sent
                BLET        &NUMTRF=&NUMTRF+1                //tally number of msgs transferred
L8              ADVANCE     (PH(JMSGLEN)+MSGOVH)_
                            /(&COMMRAT*&XEFF)+CONNOVH        //send msg to remote
                TEST E      &DCHNLBSY,TRUE,L9                //if both channels busy
                ADVANCE     (PH(JMSGLEN)+MSGOVH)_
                            /(&COMMRAT*&XEFF2)+CONNOVH       //xmit time is increased
L9              TEST E      PH(JISACK),FALSE,L10             //collect stats on msg only
```

```
            DEPART      QTRF
            DEPART      QSND
L10         UNLINK E    AQC03.ASNDNXT,1.JDEST$PH.PH(JDEST)._
                        ASNDHUP                          //find entry for same dest & send
            TEST E      PH(JACKREQ).FALSE.DMAKEACK       //acks must be requeued at driver
            TERMINATE   1                                //msg sent, go to next
*
* No msgs left to send to destination.  Check to
* see if link is idle and disconnect if it is.
*
ASNDHUP     TEST E      &DCHNLBSY.TRUE.L11               //if driver is still busy,
            BLET        &ACHNLBSY=FALSE                  //release adapter channel,
            TRANSFER    ,ASNDEND                         //and then terminate msg
L11         BLET        &PORTBUSY=&PORTBUSY+AC1-&BEGBUSY //port on-hook
            BLET        &BEGNOANS=AC1                    //port not listening until reset
            MSAVEVAL    TIMRS.1,RSETIMR.HUPTIM.MH        //hangup port
            TABULATE    TNUMTRF                          //track # msgs xferred in connection
ASNDEND     TEST E      PH(JISACK).FALSE.L12             //if ; ot an ack, then
            TEST E      PH(JACKREQ).FALSE.DMAKEACK       //acks must be requeued at driver
L12         TERMINATE   1                                //msg (or ack) sent
*
**********************************
* ack generator used by driver
**********************************
*
DMAKEACK    ASSIGN      JPRIO,LOWPRIO.PH                 //ack's are low priority
            ASSIGN      JMSGLEN,0.PH                     //ack's have no msg length
            ASSIGN      JACKREQ.FALSE.PH                 //ack's are not acknowledged
            ASSIGN      JISACK.TRUE.PH                   //is itself an ack
            ADVANCE     QT$DELIV+FT$FPREP                //wait for delivery and prep'ing
            TRANSFER    ,DRTNACK                         //place in return queue
*
**********************************
* driver routine
**********************************
*
```

```
                GENERATE        RVEXPO(2,&MEANRAT),,,,7PH,2PL           //generate for all adapters
                ASSIGN          JDEST,FIX(FRN7*(&NUMADPTR-1))+1,PH      //assign source to the destination field
                ASSIGN          JPRIO,&PRIO,PH                          //medium priority msg
                ASSIGN          JMSGLEN,&MSGLEN,PH                      //chosen msg length
                ASSIGN          JACKREQ,&ACKREQ,PH                      //ack requested (no,yes)
                ASSIGN          JISACK,FALSE,PH                         //not an ack itself
                ASSIGN          JFRST,FALSE,PH                          //not yet a "first" msg
                BLET            &BUSYRAT=&PORTBUSY/C1                   //busy ratio is time busy/total run time
                BLET            &NOANSRAT=&NOANS/C1                     //no answer ratio is time no ans/tot run time
*
* Put msg in operations queue.
*
DRTNACK         ASSIGN          JKEYTIM,AC1+&PRIOWAIT(PH(JPRIO)),PL     //set key stamp
                ASSIGN          JTRIGTIM,PL(JKEYTIM),PL                 //save trigger time
                LINK            DQC03,JKEYTIM$PL                        //C03 queue in order of key stamp
*
* Message and port both available,
* now send...
*
DSND            TEST LE         FRN5,&BUSYRAT,DRINGING                  //adapter is busy
DREKEY          MACRO           BUSYTIM                                 //rekey msgs
*
DRINGING        TEST LE         FRN6,(&NOANSRAT/(1.0-&BUSYRAT)),_
                                DSNDFRST                                //adapter is originate only
DREKEY          MACRO           (NANSTIM+&LINEIDLE)                     //rekey msgs to same dest
*
* Rekey the trx coming from same destination because of a connection
* failure.  Only rekey those whose trx with key stamps less than
* the future key stamp.  Increment the number of conn requests
* if the trx key time is less than current time.
*
DRKEY           TEST L          PL(JKEYTIM),&KEYTIM,DRKEYEND            //key need changing?
                BLET            &NEWKEY=&NEWKEY+1                       //ensure keys remain ascending
                TEST LE         PL(JKEYTIM),AC1,L13                     //incr conn requests only for
                ASSIGN          JCONNREQ,PH(JCONNREQ)+1,PH              //those trx older than now
L13             ASSIGN          JKEYTIM,&NEWKEY,PL                      //set new key time
```

```
            UNLINK E        DQC03,DRKEY,1,JDEST$PH,PH(JDEST)    //requeue other msgs to same dest
DRKEYEND    LINK            DQC03,JKEYTIM$PL                    //requeue msg in C03
*
* Connection is complete .. now send transactions.
* Take connection time afterwards to avoid double-booking the channel.
*
DSNDFRST    ASSIGN          JFRST,TRUE,PH                       //msg causes a connection to be made
            BLET            &BEGBUSY=AC1                        //port goes busy at adapter
            BLET            &NUMTRF=0                           //number transferred during connection
STPPTIMR    MACRO                                               //stop prep wait timer
            UNLINK E        AQC03,ASNDFDX,1,JDEST$PH,PH(JDEST)  //find any entry in adapter's C03 to same dest
            TRANSFER        ,DSNDNXT
DSNDFDX     BLET            &DCHNLBSY=TRUE                      //driver is sending
            ADVANCE         REACTIM,REACTIM/2                   //pause to search queue
DSNDNXT     BLET            &NUMTRF=&NUMTRF+1                   //tally number of msgs transferred
            ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                            /(&COMMRAT*&XEFF)+CONNOVH           //send msg to adapter
            TEST E          &ACHNLBSY,TRUE,L14                  //if both channels busy
            ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                            /(&COMMRAT*&XEFF2)+CONNOVH          //xmit time is increased
L14         UNLINK E        DQC03,DSNDNXT,1,JDEST$PH,PH(JDEST),_
                            DSNDHUP                             //find next entry in C03 from same dest
            TRANSFER        ,RCV                                //recv at adapter and send next
*
* No msgs here left to send to adapter.  Check to
* see if link is idle and disconnect if it is.
*
DSNDHUP     TEST E          &ACHNLBSY,TRUE,L15                  //if adapter is still busy,
            BLET            &DCHNLBSY=FALSE                     //free driver channel,
            TRANSFER        ,RCV                                //and then process msg recv'd at adapter
L15         BLET            &PORTBUSY=&PORTBUSY+AC1-&BEGBUSY    //port on-hook
            BLET            &BEGNOANS=AC1                       //port not listening until reset
            MSAVEVAL        TIMRS,1,RSETIMR,HUPTIM,MH           //hangup port
            TABULATE        TNUMTRF                             //track # msgs xferred in connection
*
**********************************
```

```
* adapter's receiver routine
************************************
*
* Prep msg and then deliver it to the application host.
*
* Note: the pickup and connection times could not be charged at the driver
* end because adapter-initiated transactions would sneak through
* while the ADVANCE blocks are being executed at the driver.
*
RCV         ADVANCE     QT$PICKUP                  //pay for pickup and staging at driver
            TEST E      PH(JFRST),TRUE,L16         //if msg caused connection,
            ADVANCE     DIALTIM                    //pay for dial time at driver
L16         TEST E      PH(JISACK),FALSE,L17       //acks are not part of QDELIV
            QUEUE       QDELIV                     //receive prep+delivery
L17         SEIZE       FPREP                      //only prep one msg at a time
PREP        MACRO       ADDRTIM                    //address msg
PREP        MACRO       SIGNTIM                    //signature decryption
PREP        MACRO       SIGNTIM                    //signature verification
PREP        MACRO       ENCRTIM                    //decrypt msg
            RELEASE     FPREP                      //finished prep'ing
SETPTIMR    MACRO                                  //restart prep timer
            TEST E      &EORV,VIRT,EMUDELIV        //if virt,
LGATEONE    MACRO       PORTTIMR,FPORT             //hold each msg until port is reviewed
            TRANSFER    ,L18                       //else
EMUDELIV    GATE LS     PORTTIMR                   //hold all pending msgs until port is reviewed
L18         SEIZE       LCHNL                      //capture local channel
STPPTIMR    MACRO                                  //stop prep wait timer
            ADVANCE     (PH(JMSGLEN)+MSGOVH)_
                        /(&EVXRAT(&EORV+1)*&XEFF)  //send msg to application host
            RELEASE     LCHNL                      //free local channel
SETPTIMR    MACRO                                  //start prep wait timer
            TEST E      PH(JISACK),FALSE,L19       //acks are not part of QDELIV
            DEPART      QDELIV
            BLET        &NUMRCV=&NUMRCV+1          //tally number msgs received
            TEST E      PH(JACKREQ),FALSE,AMAKEACK //msg's to be acked must be requeued at driver
L19         TERMINATE   1                          //msg sent
```

```
*
************************************
* ack generator used by adapter
************************************
*
AMAKEACK    ASSIGN          JPRIO,LOWPRIO,PH                    //ack's are low priority
            ASSIGN          JMSGLEN,0,PH                        //ack's have no msg length
            ASSIGN          JACKREQ,FALSE,PH                    //ack's are not acknowledged
            ASSIGN          JISACK,TRUE,PH                      //is itself an ack
            TRANSFER        ,ARTNACK                            //place in return queue to driver
*
************************************
* simulation control
************************************
*
            INTEGER         &RECNUM                             //control file record counter
            VCHAR*12        &INAME,&ONAME                       //control and output file names
*
            PUTPIC
Enter input file name (".ctl" extension is assumed):
            GETLIST         &INAME                              //get name of control file
            LET             &ONAME=&INAME||'.out'               //add extension '.out' for output file
            LET             &INAME =&INAME||'.ctl'              //add extension '.ctl' for control file
            PUTPIC          &ONAME
Writing output to ************
IFILE       FILEDEF&INAME
OFILE       FILEDEF&ONAME
CTLLOOP     LET             &RECNUM=&RECNUM+1                   //read each record in control file
            GETLIST         FILE=IFILE,END=FINISH,ERR=NOTIFY,_
                            &LINEIDLE,&RTRYWAIT,&EORV,&NUMADPTR,_
                            &MSGLEN,&MEANRAT,&PRIO,&ACKREQ
*
* Initializations
*
            RESET
            INIT            MHSTIMRS(1,POLLTIMR),FIX(FRN3*(SCANDIR+1))
```

```
                INIT            MH$TIMRS(1,PORTTIMR),FIX(FRN3*(IDLEPORT+1))
                INIT            MH$TIMRS(1,RSETIMR),INACTIVE
                INIT            MH$TIMRS(1,RLISTIMR),RLISCYCL
                LET             &CHNLBSY=FALSE                          //port not busy
                LET             &ACHNLBSY=FALSE                         //adapter channel not busy
                LET             &DCHNLBSY=FALSE                         //driver channel not busy
                LET             &PORTBUSY=0                             //ratio that port gives busy signal
                LET             &NOANS=0                                //ratio that port does not answer
                LET             &NUMSNT=0                               //tot msgs sent (excluding ack's)
                LET             &NUMRCV=0                               //tot msgs rcv'd (excluding ack's)
*
* Test run
*
                START           TESTTRX                                 //data run
                PUTSTRING       'Line Rtry #  Msg Rate _
1-3 Ack PckUp  Delv  Send  Prep  #Msgs'
                PUTSTRING       'Idle Wait E/V Adp  Len spm  _
Pri Req Time   Time  Time  Time  Sent Rcvd'
                PUTPIC          &LINEIDLE,&RTRYWAIT,&EORV,&NUMADPTR,_
                                &MSGLEN,&MEANRAT,&PRIO,&ACKREQ,_
                                TB$QTPICKUP,TB$QTDELIV,TB$QTSND,FT$FPREP,&NUMSNT,&NUMRCV
*** *** * *** **** *** ** ***** ***** ***** ***** **** ****
                PUTSTRING       FILE=OFILE,'Line Rtry #  Msg Rate _
1-3 Ack Busy  NoAns PckUp  Delv  Send  Prep  #Msgs'
                PUTSTRING       FILE=OFILE,'Idle Wait E/V Adp  Len spm  _
Pri Req Ratio Ratio Time   Time  Time  Time  Sent Rcvd'
                PUTPIC          FILE=OFILE,_
                                &LINEIDLE,&RTRYWAIT,&EORV,&NUMADPTR,_
                                &MSGLEN,&MEANRAT,&PRIO,&ACKREQ,_
                                &BUSYRAT,&NOANSRAT,TB$QTPICKUP,_
                                TB$QTDELIV,TB$QTSND,FT$FPREP,&NUMSNT,&NUMRCV
*** *** * *** **** *** ** * *** * *** ***** ***** ***** ***** **** ****
                GOTO            CTLLOOP                                 //loop back for next test
NOTIFY          PUTPIC          &RECNUM,&INAME
Error while reading line #*** of ***********
FINISH          HERE
```

END

Appendix B

GPSS/H Source Listing of the Discrete-Event Model for a Star Cluster

```
* SBALNODE.GPS
* Written by J. McDaniel, February 27, 1993
*       Discrete-Event Model of a balanced star Cluster
*
*       Input file record format (file name must have a ".ctl" extension):
*               Linkidle time,
*               RetryWait time,
*               Emulation or virtual channel used for msg pickup and delivery,
*               Number of servers at gateway
*               Number of adapters in cluster,
*               Length of msg text (characters),
*               Total mean rate of message creation at a given node (for all other nodes in cluster),
*               Message priority (1 = Low, 2 = Medium, 3 = High)
*               Message acknowledgments required (0 = false, 1 = true)
*       Spaces are used to delimit fields on the control record.
*
*       An input file can consist of one or more control records.  The first record starts the model from a zero and empty state.
*       Subsequent records do not re-initialize the model, just the statistics.
*
*       The output file (with a root name the same as the control file and an extension of ".out") consists of the input parameters
*       followed by the mean system time for message pickup, the mean system time for message delivery, the mean system time
*       for message transmission, the mean time a message is prepared, the number of messages sent by the adapter to the gateway
*       (not counting message acknowledgments), and the number of messages received by the adapter from the gateway (not
*       counting message acknowledgments).
*
                SIMULATE
                REALLOCATE COM,20000
                NOXREF
                UNLIST          MACX
*
* reserved facilities
*
FPOLL   EQU     1,F             //polltimer gate throttle
FPORT   EQU     2,F             //porttimer gate throttle
FPREP   EQU     3,F             //preptimer gate throttle
LCHNL   EQU     4,F             //local channel (half-duplex)
```

```
AQC03       EQU     1.C     //adapter C03 queue
DQC03       EQU     2.C     //driver C03 queue
GQC03       EQU     3.C     //gateway C03 queue
POLLTIMR    EQU     1.L     //poll timer (cyclic)
PORTTIMR    EQU     2.L     //incoming port timer (cyclic)
PREPTIMR    EQU     3.L     //prep wait timer
CPU         EQU     6.L     //CPU preemption
*
* time constants (seconds)
*
SCANDIR     SYN     20      //directory poll cycle (POLLTIMR)
IDLEPORT    SYN     13      //Review idle ports cycle (PORTTIMR)
PREPWAIT    SYN     5       //wait after last activity (PREPTIMR)
INTRPREP    SYN     2       //interprocess wait
DIALTIM     SYN     27      //time to dial & get carrier
HUPTIM      SYN     7       //time to hangup
ADDRTIM     SYN     1       //address, address verify
ENCRTIM     SYN     3       //en/decrypt 1000 byte msg
SIGNTIM     SYN     1       //signature generation/encryption
BUSYTIM     SYN     15      //time to wait if connection busy
REACTIM     SYN     10      //time before dest starts FDX comms
RLISTIM     SYN     2       //time to execute relisten script at port
RLISCYCL    SYN     5       //wait before review of port
CONNOVH     SYN     2       //connection overhead (sec)
INACTIVE    SYN     -1      //set timer to inactive
LOWHOLD     SYN     600     //hold time for low priority msgs
MEDHOLD     SYN     300     //hold time for medium priority msgs
HIHOLD      SYN     0       //hold time for high priority msgs
HIVALUE     SYN     99999   //used to indicate port busy condition
*
* miscallaneous indices
*
NUMTIMR     SYN     3       //# of timers
LOWPRIO     SYN     1       //low priority
MEDPRIO     SYN     2       //medium priority
HIPRIO      SYN     3       //high priority
```

```
EMU         SYN         0           //emulation channel
VIRT        SYN         1           //virtual channel
ISBSY       SYN         1           //ACHNL,GCHNL,CHNL busy
TRUE        SYN         1           //true
FALSE       SYN         0           //false
*
* transaction (msg) parameter indices
*
JDEST       SYN         1           //c03r.remoteAddr
JCONNREQ    SYN         2           //c03r.numConnReq
JPRIO       SYN         3           //priority rating
JMSGLEN     SYN         4           //msg length
JACKREQ     SYN         5           //ack requested
JISACK      SYN         6           //is an ack
JSRC        SYN         7           //source address
JKEYTIM     SYN         1           //c03r.keyStmp
JTRIGTIM    SYN         2           //c03r.trigStmp
JTIM1       SYN         1           //parm used as timer index
*
* miscelaneous simulation setup constants
*
TESTTRX     SYN         2000        //term counter for test
*
* miscelaneous transmission constants
*
MSGOVH      SYN         400         //msg char overhead (hdr+conn/disconn)
XSPEED      SYN         2400        //modem speed (2400 bps)
VSPEED      SYN         100000      //virtual channel speed (100,000 bps)
ESPEED      SYN         9600        //emulation channel speed (9600 bps)
XMITEFF     SYN         45          //transmission efficiency for HDX (45%)
XMITEFF2    SYN         40          //FDX (incremental) efficiency
*
* ampervariables
*
            REAL        &XEFF,&KEYTIM,&NEWKEY,&XEFF2
            INTEGER     &EORV,&EVXRAT(2),&MSGLEN,&MEANRAT
```

```
                INTEGER         &NUMADPTR,&PRIOWAIT(3),&COMMRAT
                INTEGER         &LINEIDLE,&RTRYWAIT,&ACKREQ,&PRIO
                INTEGER         &NUMSNT,&NUMRCV,&NUMTRF,&NUMSRVR
                INTEGER         &GSRVRIDX,&DEST
*
* matrices
*
TIMRS           MATRIX          MH,1,NUMTIMR                    //timers (counters)
*
* table definitions
*
QTPICKUP QTABLE                 QPICKUP,20,5,200                //pickup poll+prep times
QTSND           QTABLE          QSND,300,10,400                 //queuing+sending times
QTDELIV         QTABLE          QDELIV,20,5,200                 //recving prep+delivery poll
TCONNREQ        TABLE           PH(JCONNREQ),0,1,20             //number of connection requests
TNUMTRF         TABLE           &NUMTRF,0,1,20                  //# of msgs xferred during connection
*
* computed constants
*
                LET             &XEFF=XMITEFF/100.0             //transmit efficiency (HDX)
                LET             &XEFF2=XMITEFF2/100.0           //transmit efficiency FDX (incremental)
                LET             &EVXRAT(1)=ESPEED/10            //emu ch rate (9600 bps)
                LET             &EVXRAT(2)=VSPEED/10            //virt ch rate (100,000 bps)
                LET             &COMMRAT=XSPEED/10              //comm line rate (2400 bps)
                LET             &PRIOWAIT(LOWPRIO)=LOWHOLD      //low priority wait
                LET             &PRIOWAIT(MEDPRIO)=MEDHOLD      //medium priority wait
                LET             &PRIOWAIT(HIPRIO)=HIHOLD        //high priority wait
*
**********************************
* macros
**********************************
*
* GETSRVR(index)
* searches through gateway servers (or ports) looking for a
* port with no activity and returns an index to that port else
* returns zero
```

```
*
GETSRVR     STARTMACRO
            BLET        #A=&NUMSRVR                    //count down from &NUMSRVR
            TEST GE     ML(GSRVRBSY,1,#A),AC1,*+3      //skip over active ports
            BLET        #A=#A-1                        //decr loop counter
            TEST E      #A,0,*-2                       //loop back to TEST GE
            ENDMACRO
*
* FINDSRVR(destination,index)
* searches through gateway servers for port connected to destination
* and returns value of index if found and active else returns zero
*
FINDSRVR    STARTMACRO
            BLET        #B=&NUMSRVR                    //count down from &NUMSRVR
            TEST E      MH(GSRVRID,1,#B),#A,*+2        //find matching destination
            TEST L      ML(GSRVRBSY,1,#B),AC1,*+3      //skip over inactive ports
            BLET        #B=#B-1                        //decr loop counter
            TEST E      #B,0,*-3                       //loop back to TEST E
            ENDMACRO
*
* LGATEONE(LogicSwitch,resourceThrottle)
* latched logic gate which allows only one trx through and
* resets logic switch after
*
LGATEONE    STARTMACRO
            SEIZE       #B
            GATE LS     #A
            LOGIC R     #A
            RELEASE     #B
            ENDMACRO
*
* PREP(PrepTime)
* Perform one stage of msg prep'ing, then
* reset msg prep timer.
*
PREP        STARTMACRO
```

```
                GATE LS         PREPTIMR                        //gate holds back msg until ready
                LOGIC R         PREPTIMR                        //close gate
                GATE LR         CPU                             //gate if CPU is already in use
                LOGIC S         CPU                             //prempt CPU
                ADVANCE         #A                              //prep msg
                LOGIC R         CPU                             //release CPU
                MSAVEVAL        TIMRS,1,PREPTIMR,INTRPREP,MH    //start prep wait timer
                ENDMACRO
*
* SETPTIMR()
* Set the preptimer and return to next statement
*
SETPTIMR        STARTMACRO
                LOGIC R         PREPTIMR                        //reset gate
                MSAVEVAL        TIMRS,1,PREPTIMR,PREPWAIT,MH    //start prep wait timer
                ENDMACRO
*
* STPPTIMR()
* Stop the preptimer and return to next statement
*
STPPTIMR        STARTMACRO
                LOGIC R         PREPTIMR                        //reset gate
                MSAVEVAL        TIMRS,1,PREPTIMR,INACTIVE,MH    //wait indefinitely
                ENDMACRO
*
**********************************
* timer routine
**********************************
*
                GENERATE        1                               //generate a trx every second
*
* Iterate through all timers.
* If a timer has expired, set then reset logic flag.
* If the expired timer is a cyclic timer, reset the timer.
* Note: if same timer expires and is reset within the same second,
* reduce the second setting by a second due to timer granularity.
```

```
*
            ASSIGN          JTIM,NUMTIMR,PH
*
TIMRLP      UNLINK LE       DQC03,DSND,ALL,JKEYTIM$PL,AC1   //find entry at driver
            TEST NE         MH(TIMRS,1,PH(JTIM)),0,TIMRLP1  //timer expired?
            TEST G          MH(TIMRS,1,PH(JTIM)),0,TIMRLP2  //timer active?
            MSAVEVAL        TIMRS-,1,PH(JTIM),1,MH          //decrement timer
            TRANSFER        ,TIMRLP5
*
TIMRLP1     MSAVEVAL        TIMRS,1,PH(JTIM),INACTIVE,MH    //timer now inactive
            TEST E          PH(JTIM),PREPTIMR,L1            //if preptimer
            TEST LE         ML(CHNLBSY,1,1),AC1,TIMRLP5     //and port not busy
            TEST NE         FNU$LCHNL,0,TIMRLP5             //and local channel not busy,
L1          LOGIC S         PH(JTIM)                        //set timer flag
            TRANSFER        ,TIMRLP5
*
* Reset cyclic timers: poll,port, and prep timer.
*
TIMRLP2     TEST E          PH(JTIM),POLLTIMR,TIMRLP3       //only poll timer here
            LOGIC R         PH(JTIM)                        //reset timer
            MSAVEVAL        TIMRS,1,POLLTIMR,SCANDIR-2,MH   //restart poll cycle for next pull op
            TEST NE         FNU$FPOLL,0,TIMRLP5             //anything pending?
            MSAVEVAL        TIMRS+,1,POLLTIMR,SCANDIR,MH    //wait two cycles: dir and pull ops
            TRANSFER        ,TIMRLP5
*
TIMRLP3     TEST E          PH(JTIM),PORTTIMR,TIMRLP4       //only port timer here
            LOGIC R         PH(JTIM)                        //reset timer
            TEST LE         ML(CHNLBSY,1,1),AC1,L2          //is local port ready?
            UNLINK LE       AQC03,ASND,1,JKEYTIM$PL,AC1,L2  //find entry in C03 and send
            MSAVEVAL        CHNLBSY,1,1,AC1+HIVALUE,ML      //make local port busy
L2          MSAVEVAL        TIMRS,1,PORTTIMR,IDLEPORT-2,MH  //restart incoming port cycle timer
            TRANSFER        ,TIMRLP5
*
TIMRLP4     TEST LE         ML(CHNLBSY,1,1),AC1,TIMRLP5     //prep timer only: port available?
SETPTIMR    MACRO                                           //start prep wait timer
*
```

```
* End of loop
*
TIMRLP5     LOOP          JTIM$PH,TIMRLP                  //decrement timer index
            TERMINATE
*
**********************************
* adapter routines
**********************************
*
            GENERATE      RVEXPO(2,&MEANRAT),,,,,7PH,2PL  //generate for all adapters
            ASSIGN        JDEST,FIX(FRN4*(&NUMADPTR-1))+2,PH  //assign destination
            ASSIGN        JSRC,1,PH                       //assign source
            ASSIGN        JPRIO,&PRIO,PH                  //medium priority msg
            ASSIGN        JMSGLEN,&MSGLEN,PH              //chosen msg length
            ASSIGN        JACKREQ,&ACKREQ,PH              //ack requested
            ASSIGN        JISACK,FALSE,PH                 //not an ack itself
*
* Transfer a msg to the adapter over the emulation or virtual channel...
*
            QUEUE         QPICKUP                         //pickup to staging time
            GATE LS       POLLTIMR                        //hold until polled - leave gate open
            TEST E        &EORV,VIRT,EMUPCKUP             //if virt,
LGATEONE    MACRO         PORTTIMR,FPORT                  //hold each msg until port is reviewed
            TRANSFER      ,L3                             //else
EMUPCKUP    GATE LS       PORTTIMR                        //hold msgs till port is reviewed
L3          SEIZE         LCHNL                           //capture local channel
STPPTIMR    MACRO                                         //stop prep wait timer
            ADVANCE       (PH(JMSGLEN)+MSGOVH)_
                          /(&EVXRAT(&EORV+1)*&XEFF)       //send msg to adapter
            RELEASE       LCHNL                           //free local channel
SETPTIMR    MACRO                                         //start prep wait timer
*
* Message stored in msg base,
* now prep msg...
*
ARTNACK     SEIZE         FPREP                           //only prep one msg at a time
```

```
PREP        MACRO       ADDRTIM                          //address msg
PREP        MACRO       ENCRTIM                          //encrypt msg
PREP        MACRO       SIGNTIM                          //signature generation
PREP        MACRO       SIGNTIM                          //signature encryption
            RELEASE     FPREP                            //finished prep'ing
SETPTIMR    MACRO                                        //restart prep timer
            TEST E      PH(JISACK),FALSE,L4              //ack's are not part of QPICKUP
            DEPART      QPICKUP
*
* Message is prep'ed,
* now put in operations queue
*
L4          ASSIGN      JKEYTIM,AC1+&PRIOWAIT(PH(JPRIO)),PL  //set key stamp
            ASSIGN      JTRIGTIM,PL(JKEYTIM),PL          //save trigger time
            TEST E      PH(JISACK),FALSE,L5              //collect stats on msg only
            QUEUE       QSND                             //adapter-to-adapter time
L5          LINK        AQC03,JKEYTIM$PL                 //C03 queue in order of key stamp
*
* Message and port both available,
* now send...
*
ASND        GATE LR     CPU                              //wait until CPU is free
            BLET        &NUMTRF=0                        //number transferred during connection
STPPTIMR    MACRO                                        //stop prep wait timer
            ADVANCE     BUSYTIM                          //wait to detect line is busy
*
GETSRVR     MACRO       &GSRVRIDX                        //find inactive gateway port
            TEST E      &GSRVRIDX,0,ASNDFRST             //gateway is busy
*
* Connection request fails due to a busy line.  The gateway
* disconnect script has been reconfigured to prevent situations of
* no answer.  The adapter port is released and the transactions are
* rekeyed.  Those trx with key stamps less than the future key stamp
* are rekeyed.  The number of connection requests, PH(JCONNREQ),
* are incremented if the key stamp is less than the current time.
*
```

```
                MSAVEVAL        CHNLBSY,1,1,_
                                AC1+HUPTIM+RLISTIM+RLISCYCL,ML      //set relisten time at adapter
                ASSIGN          JCONNREQ,PH(JCONNREQ)+1,PH          //increment conn requests
                BLET            &KEYTIM=AC1+FLT(&RTRYWAIT)*_
                                (PH(JCONNREQ)*FRN3+1.0)             //save new keytime for rekeying
                BLET            &NEWKEY=&KEYTIM                     //keep a copy for later changes
                ASSIGN          JKEYTIM,&KEYTIM,PL                  //advance key time
                UNLINK L        AQC03,ARKEY,1,JKEYTIM$PL,&KEYTIM    //requeue other msgs
                LINK            AQC03,JKEYTIM$PL                    //requeue msg in C03
ARKEY           BLET            &NEWKEY=&NEWKEY+1                   //ensure keys remain ascending
                TEST LE         PL(JKEYTIM),AC1,L6                  //incr conn requests only for
                ASSIGN          JCONNREQ,PH(JCONNREQ)+1,PH          //those trx older than now
L6              ASSIGN          JKEYTIM,&NEWKEY,PL                  //set new key time
                UNLINK L        AQC03,ARKEY,1,JKEYTIM$PL,&KEYTIM    //requeue other msgs to same dest
                LINK            AQC03,JKEYTIM$PL                    //requeue msg in C03
*
* Connection is complete .. now sending transactions.
*
ASNDFRST        MSAVEVAL        CHNLBSY,1,1,AC1+HIVALUE,ML          //adapter port goes busy
                MSAVEVAL        FDX,1,1,1,MH                        //HDX channel
                MSAVEVAL        GSRVRBSY,1,&GSRVRIDX,AC1+HIVALUE,ML //gateway port busy until released
                MSAVEVAL        GSRVRID,1,&GSRVRIDX,1,MH            //source is registered
                ADVANCE         (DIALTIM-BUSYTIM)                   //wait to answer and detect carrier
                UNLINK          GQC03,ARCV,1,JDEST$PH,1             //check gateway for msgs pending
ASNDNXT         TABULATE        TCONNREQ                            //tabulate # conn requests
                TEST E          PH(JISACK),FALSE,L7                 //collect stats on msg only
                QUEUE           QTRF
                BLET            &NUMSNT=&NUMSNT+1                   //tally number msgs sent
                BLET            &NUMTRF=&NUMTRF+1                   //tally number of msgs transferred
L7              ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                                /(&COMMRAT*&XEFF)+CONNOVH           //send msg to remote
                TEST E          MH(FDX,1,1),2,L8                    //if full duplex
                ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                                /(&COMMRAT*&XEFF2)+CONNOVH          //xmit time is increased
L8              TEST E          PH(JISACK),FALSE,L9                 //collect stats on msg only
                DEPART          QTRF
```

```
                ASSIGN          JTRIGTIM.AC1.PL                 //set trigger time at gateway
                ASSIGN          JKEYTIM.AC1.PL                  //set new keytime at gateway
L9              UNLINK          AQC03.ASNDNXT.1...ASNDHUP       //find next msg
                LINK            GQC03.JKEYTIM$PL                //store msg at gateway
*
* No msgs left to send to gateway.  Check to
* see if link is idle and disconnect if it is.
*
ASNDHUP         TEST E          MH(FDX.1.1).2.L10               //if channel still busy.
                MSAVEVAL        FDX.1.1.1.MH                    //make HDX
                LINK            GQC03.JKEYTIM$PL                //store msg at gateway
L10             MSAVEVAL        FDX.1.1.0.MH                    //port is cleared
FINDSRVR        MACRO           1.&GSRVRIDX                     //search for gateway port
                MSAVEVAL        GSRVRBSY.1.&GSRVRIDX.AC1+HUPTIM+_
                                RLISTIM+RLISCYCL.ML             //gateway port released after delay
                MSAVEVAL        CHNLBSY.1.1._
                                AC1+HUPTIM+RLISTIM+RLISCYCL.ML  //set relisten time at adapter
                TABULATE        TNUMTRF                         //track # msgs xferred in connection
                LINK            GQC03.JKEYTIM$PL                //store msg at gateway
*
**********************************
* adapter's receiver routine
**********************************
*
* Receive msg from gateway, prep it, then store it.
*
ARCV            MSAVEVAL        FDX.1.1.2.MH                    //FDX port at adapter
                ADVANCE         REACTIM                         //pause at gateway to search queue
ARCVNXT         TEST E          PH(JISACK).FALSE.L11            //collect stats on msg only
                QUEUE           QTRF
                BLET            &NUMTRF=&NUMTRF+1               //tally number of msgs transferred
L11             ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                                /(&COMMRAT*&XEFF)+CONNOVH       //recv msg from gateway
                TEST E          MH(FDX.1.1).2.L12               //if full duplex
                ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                                /(&COMMRAT*&XEFF2)+CONNOVH      //xmit time is increased
```

```
L12             TEST E          PH(JISACK).FALSE.L13                    //collect stats on msg only
                DEPART          QTRF
L13             UNLINK          GQC03.ARCVNXT.1.JDEST$PH.1.ARCVHUP      //find next msg
                TRANSFER        .ARCVPRP                                //prep and store msg
*
* No msgs left to receive from gateway.  Check to
* see if link is idle and disconnect if it is.
*
ARCVHUP         TEST E          MH(FDX,1.1).2.L14                       //if channel still busy,
                MSAVEVAL        FDX.1,1.1.MH                            //make HDX
                TRANSFER        .ARCVPRP                                //prep and store msg
L14             MSAVEVAL        FDX.1,1,0.MH                            //port is cleared
FINDSRVR        MACRO           1,&GSRVRIDX                             //search for gateway port
                MSAVEVAL        GSRVRBSY,1,&GSRVRIDX.AC1+HUPTIM+_
                                RLISTIM+RLISCYCL.ML                     //gateway port released after delay
                MSAVEVAL        CHNLBSY,1.1,_
                                AC1+HUPTIM+RLISTIM+RLISCYCL.ML          //set relisten time
                TABULATE        TNUMTRF                                 //track # msgs xferred in connection
ARCVPRP         TEST E          PH(JISACK).FALSE.L15                    //collect stats on msg only
                QUEUE           QDELIV                                  //receive prep+delivery
L15             SEIZE           FPREP                                   //only prep one msg at a time
PREP            MACRO           ADDRTIM                                 //address msg
PREP            MACRO           SIGNTIM                                 //signature decryption
PREP            MACRO           SIGNTIM                                 //signature verification
PREP            MACRO           ENCRTIM                                 //decrypt msg
                RELEASE         FPREP                                   //finished prep'ing
SETPTIMR        MACRO                                                   //restart prep timer
                TEST E          &EORV,VIRT,EMUDELIV                     //if virt,
LGATEONE        MACRO           PORTTIMR,FPORT                          //hold each msg until port is reviewed
                TRANSFER        ,L16                                    //else
EMUDELIV        GATE LS         PORTTIMR                                //hold all pending msgs until port is reviewed
L16             SEIZE           LCHNL                                   //capture local channel
STPPTIMR        MACRO                                                   //stop prep wait timer
                ADVANCE         (PH(JMSGLEN)+MSGOVH)_
                                /(&EVXRAT(&EORV+1)*&XEFF)               //send msg to application host
                RELEASE         LCHNL                                   //free local channel
```

```
SETPTIMR    MACRO                                                   //start prep wait timer
            TEST E      PH(JISACK),FALSE.L17                        //acks are not part of QDELIV
            DEPART      QDELIV
            BLET        &NUMRCV=&NUMRCV+1                           //tally number msgs received
            TEST E      PH(JACKREQ),FALSE.AMAKEACK                  //acks must be requeued at driver
L17         TERMINATE   1                                           //msg received
*
**********************************
* adapter's ack generator
**********************************
*
AMAKEACK ASSIGN         JPRIO,LOWPRIO.PH                            //ack's are low priority
            ASSIGN      JMSGLEN,0.PH                                //ack's have no msg length
            ASSIGN      JACKREQ,FALSE.PH                            //ack's are not acknowledged
            ASSIGN      JISACK,TRUE.PH                              //is itself an ack
            ASSIGN      JDEST,PH(JSRC).PH                           //source is now destination
            ASSIGN      JSRC,1.PH                                   //adapter is now source
            TRANSFER    ,ARTNACK                                    //place in return queue to driver
*
**********************************
* driver routines
**********************************
*
            GENERATE    RVEXPO(2,&MEANRAT/(&NUMADPTR-1)),_
                        ,,,,7PH,2PL                                 //generate for all other adapters
            ASSIGN      JDEST,FIX(FRN7*(&NUMADPTR-1))+1.PH          //assign destination
            ASSIGN      JSRC,FIX(FRN5*(&NUMADPTR-1))+2,PH           //assign src
            TEST E      PH(JDEST),PH(JSRC),L18                      //if src and dest are same,
            ASSIGN      JDEST,&NUMADPTR,PH                          //choose alternative dest
L18         ASSIGN      JPRIO,&PRIO.PH                              //medium priority msg
            ASSIGN      JMSGLEN,&MSGLEN.PH                          //chosen msg length
            ASSIGN      JACKREQ,&ACKREQ.PH                          //ack requested (no,yes)
            ASSIGN      JISACK,FALSE.PH                             //not an ack itself
*
* Put msg in operations queue.
*
```

```
                ADVANCE         QT$PICKUP                       //pickup and staging times
DRTNACK         ASSIGN          JKEYTIM,AC1+&PRIOWAIT(PH(JPRIO))+_
                                DIALTIM,PL
                                                                //set key stamp (the dialling
                                                                //time adds to the holding time)
*
                ASSIGN          JTRIGTIM,PL(JKEYTIM),PL         //save trigger time
                LINK            DQC03,JKEYTIM$PL                //C03 queue in order of key stamp
*
* Message and port both available,
* now send...
*
DSND            TEST LE         ML(CHNLBSY,1,PH(JSRC)),AC1,DREQUEUE   //check for busy port
GETSRVR         MACRO           &GSRVRIDX                       //find inactive gateway port
                TEST E          &GSRVRIDX,0,DSNDFRST            //gateway is busy
*
* Connection request fails due to a busy line.  The gateway
* disconnect script has been reconfigured to prevent situations of
* no answer.  The driver port is released and the transactions are
* rekeyed.  Those trx with key stamps less than the future key stamp
* are rekeyed.  The number of connection requests, PH(JCONNREQ),
* are incremented if the key stamp is less than the current time.
*
                MSAVEVAL        CHNLBSY,1,PH(JSRC),AC1+BUSYTIM,ML   //port goes busy
                ADVANCE         BUSYTIM                         //wait to detect line is busy
                ASSIGN          JCONNREQ,PH(JCONNREQ)+1,PH      //increment conn requests
                BLET            &KEYTIM=AC1+FLT(&RTRYWAIT)*_
                                (PH(JCONNREQ)*FRN3+1.0)
                                                                //save new keytime for rekeying
                BLET            &NEWKEY=&KEYTIM                 //keep a copy for later changes
                ASSIGN          JKEYTIM,&KEYTIM,PL              //advance key time
                UNLINK          DQC03,DRKEY,1,JSRC$PH,PH(JSRC)  //requeue other msgs from same src
                LINK            DQC03,JKEYTIM$PL                //requeue msg in C03
DRKEY           TEST L          PL(JKEYTIM),&KEYTIM,DREQUEUE    //key need changing?
                BLET            &NEWKEY=&NEWKEY+1               //ensure keys remain ascending
                TEST LE         PL(JKEYTIM),AC1,L19             //incr conn requests only for
                ASSIGN          JCONNREQ,PH(JCONNREQ)+1,PH      //those trx older than now
L19             ASSIGN          JKEYTIM,&NEWKEY,PL              //set new key time
                UNLINK E        DQC03,DRKEY,1,JSRC$PH,PH(JSRC)  //requeue other msgs from same src
```

```
DREQUEUE    LINK          DQC03,JKEYTIM$PL                    //requeue msg in C03
*
* Connection is complete .. now send transactions.
*
DSNDFRST    MSAVEVAL      CHNLBSY,1,PH(JSRC),AC1+HIVALUE,ML   //adapter port goes busy
            MSAVEVAL      FDX,1,PH(JSRC),1,MH                 //HDX channel
            MSAVEVAL      GSRVRBSY,1,&GSRVRIDX,AC1+HIVALUE,ML //gateway port goes busy
            MSAVEVAL      GSRVRID,1,&GSRVRIDX,PH(JSRC),MH     //source is registered
            ADVANCE       DIALTIM                             //wait to dial and detect carrier
            UNLINK E      GQC03,DRCV,1,JDEST$PH,PH(JSRC)      //search gateway for return msgs
DSNDNXT     ADVANCE       (PH(JMSGLEN)+MSGOVH)_
                          /(&COMMRAT*&XEFF)+CONNOVH           //send msg to gateway
            TEST E        MH(FDX,1,PH(JSRC)),2,L20            //check if FDX transfer
            ADVANCE       (PH(JMSGLEN)+MSGOVH)_
                          /(&COMMRAT*&XEFF2)+CONNOVH          //xmit time is increased
            ASSIGN        JTRIGTIM,AC1,PL                     //set trigger time at gateway
            ASSIGN        JKEYTIM,AC1,PL                      //set new keytime at gateway
L20         UNLINK E      DQC03,DSNDNXT,1,JSRC$PH,PH(JSRC),_
                          DSNDHUP                             //find next entry in C03 from same dest
            LINK          GQC03,JKEYTIM$PL                    //store msg at gateway
*
* No msgs here left to send to adapter.  Check to
* see if link is idle and disconnect if it is.
*
DSNDHUP     TEST E        MH(FDX,1,PH(JSRC)),2,L21            //if gateway is still busy,
            MSAVEVAL      FDX,1,PH(JSRC),1,MH                 //release adapter's channel
            LINK          GQC03,JKEYTIM$PL                    //store msg at gateway
L21         MSAVEVAL      FDX,1,PH(JSRC),0,MH                 //port is cleared
FINDSRVR    MACRO         PH(JSRC),&GSRVRIDX                  //search for gateway port
            MSAVEVAL      GSRVRBSY,1,&GSRVRIDX,AC1+HUPTIM+_
                          RLISTIM+RLISCYCL,ML                 //gateway port released after delay
            MSAVEVAL      CHNLBSY,1,PH(JSRC),_
                          AC1+HUPTIM+RLISTIM+RLISCYCL,ML      //set relisten time at adapter
            LINK          GQC03,JKEYTIM$PL                    //store msg at gateway
*
**********************************
```

```
* driver's receiver routine
**********  ********************
*
* Receive msg from gateway.
*
DRCV        MSAVEVAL    FDX,1,PH(JDEST),2,MH                    //FDX port at adapter
            ADVANCE     REACTIM                                 //pause at gateway to search queue
DRCVNXT     ADVANCE     (PH(JMSGLEN)+MSGOVH)_
                        /(&COMMRAT*&XEFF)+CONNOVH               //recv msg from gateway
            TEST E      MH(FDX,1,PH(JDEST)),2,L22               //if full duplex
            ADVANCE     (PH(JMSGLEN)+MSGOVH)_
                        /(&COMMRAT*&XEFF2)+CONNOVH              //xmit time is increased
L22         UNLINK E    GQC03,DRCVNXT,1,JDEST$PH,PH(JDEST),_
                        DRCVHUP                                 //find next msg
            TRANSFER    ,DRCVPRP                                //prep and store msg
*
* No msgs left to receive from gateway.  Check to
* see if link is idle and disconnect if it is.
*
DRCVHUP     TEST E      MH(FDX,1,PH(JDEST)),2,L23               //if channel still busy,
            MSAVEVAL    FDX,1,PH(JDEST),1,MH                    //make HDX
            TRANSFER    ,DRCVPRP                                //prep and store msg
L23         MSAVEVAL    FDX,1,PH(JDEST),0,MH                    //port is cleared
FINDSRVR    MACRO       PH(JDEST),&GSRVRIDX                     //search for gateway port
            MSAVEVAL    GSRVRBSY,1,&GSRVRIDX,AC1+HUPTIM+_
                        RLISTIM+RLISCYCL,ML                     //gateway port released after delay
            MSAVEVAL    CHNLBSY,1,PH(JDEST),_
                        AC1+HUPTIM+RLISTIM+RLISCYCL,ML          //set relisten time
DRCVPRP     TEST E      PH(JISACK),FALSE,L24                    //collect stats on msg only
            TEST E      PH(JSRC),1,L24                          //and adapter '1' msg only
            DEPART      QSND
L24         ADVANCE     QT$DELIV                                //deliver msg
            TEST E      PH(JACKREQ),FALSE,DMAKEACK              //msg's to be acked must be requeued at driver
            TERMINATE   1                                       //msg received
*
**********************************
```

```
* driver's ack generator
************************************
*
DMAKEACK    ASSIGN          JPRIO,LOWPRIO,PH          //ack's are low priority
            ASSIGN          JMSGLEN,0,PH              //ack's have no msg length
            ASSIGN          JACKREQ,FALSE,PH          //ack's are not acknowledged
            ASSIGN          JISACK,TRUE,PH            //is itself an ack
            BLET            &DEST=PH(JDEST)           //save destination
            ASSIGN          JDEST,PH(JSRC),PH         //source is now destination
            ASSIGN          JSRC,&DEST,PH             //destination is now source
            ADVANCE         FT$FPREP                  //prep acknowledgement
            TRANSFER        ,DRTNACK                  //place in return queue
*
************************************
* simulation control
************************************
*
            INTEGER         &RECNUM                   //control file record counter
            VCHAR*12        &INAME,&ONAME             //control and output file names
*
            PUTPIC
Enter input file name (".ctl" extension is assumed):
            GETLIST         &INAME                    //get name of control file
            LET             &ONAME=&INAME||'.out'     //add extension '.out' for output file
            LET             &INAME=&INAME||'.ctl'     //add extension '.ctl' for control file
            PUTPIC          &ONAME
Writing output to ************
IFILE       FILEDEF         &INAME
OFILE       FILEDEF         &ONAME
CTLLOOP     LET             &RECNUM=&RECNUM+1         //read each record in control file
            GETLIST         FILE=IFILE,END=FINISH,ERR=NOTIFY,_
                            &LINEIDLE,&RTRYWAIT,&EORV,&NUMSRVR,_
                            &NUMADPTR,&MSGLEN,&MEANRAT,&PRIO,&ACKREQ
*
* Initializations
*
```

```
            CLEAR               //clear queues. etc.
CHNLBSY     MATRIX              ML,1,&NUMADPTR                                  //adapter busy clocks
FDX         MATRIX              MH,1,&NUMADPTR                                  //FDX flags
GSRVRBSY    MATRIX              ML,1,&NUMSRVR                                   //gateway busy clocks
GSRVRID     MATRIX              MH,1,&NUMSRVR                                   //gateway destination
            INIT                MH$TIMRS(1,POLLTIMR),FIX(FRN3*(SCANDIR+1))
            INIT                MH$TIMRS(1,PORTTIMR),FIX(FRN3*(IDLEPORT+1))
            LET                 &NUMSNT=0                                       //tot msgs sent (excluding ack's)
            LET                 &NUMRCV=0                                       //tot msgs rcv'd (excluding ack's)
*
* Test run
*
            START               TESTTRX                                         //data run
            PUTSTRING           'Line Rtry    # #  Msg Rate _
1-3 Ack PckUp  Delv  Send  Prep  #Msgs'
            PUTSTRING           'Idle Wait E/V Srv Adp  Len spm  _
Pri Req Time   Time  Time  Time  Time Sent Rcvd'
            PUTPIC              &LINEIDLE,&RTRYWAIT,&EORV,&NUMSRVR,_
                                &NUMADPTR,&MSGLEN,&MEANRAT,&PRIO,_
                                &ACKREQ,TB$QTPICKUP,TB$QTDELIV,TB$QTSND,_
                                FT$FPREP,&NUMSNT,&NUMRCV
*** *** * *** *** **** *** * * ***** ***** _
** ** ***** **** ****
            PUTSTRING           FILE=OFILE,'Line Rtry    # #  Msg Rate _
1-3 Ack PckUp  Delv  Send  Prep  #Msgs'
            PUTSTRING           FILE=OFILE,'Idle Wait E/V Srv Adp  Len spm  _
Pri Req Time   Time  Time  Time  Time Sent Rcvd'
            PUTPIC              FILE=OFILE,_
                                &LINEIDLE,&RTRYWAIT,&EORV,&NUMSRVR,_
                                &NUMADPTR,&MSGLEN,&MEANRAT,&PRIO,_
                                &ACKREQ,TB$QTPICKUP,TB$QTDELIV,TB$QTSND,_
                                FT$FPREP,&NUMSNT,&NUMRCV
*** *** * *** *** **** *** * * ***** ***** _
***** ***** **** ****
            GOTO                CTLLOOP                                         //loop back for next test
NOTIFY      PUTPIC              &RECNUM,&INAME
```

Error while reading line #* of **********
FINISH HERE
END**

Appendix C

C Source Listing of the Analytic Model for a Mesh Cluster

```
/* mesh.cpp                                                              */
/* Written by J. McDaniel, March 24, 1993                                */
/*      Analytically computes the message transfer times of Health Link for */
/*      medium-priority msgs in a fully-connected mesh.                  */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define TPRIO           300.0       /* MedPrioDelay (sec)               */
#define TDIAL           27.0        /* time to dial & connect(sec)      */
#define TBUSY           15.0        /* time to dial & find busy(sec)    */
#define TXFR            23.0        /* FDX transfer time (sec/msg)      */
#define THUP            7.0         /* time to hang up (sec)            */
#define TRTRY           180.0       /* retry wait (sec)                 */
#define TRSET           10.0        /* reset modem (sec)                */
#define TLIDL           60.0        /* LineIdle wait (sec)              */
#define TRLISTN         7.0         /* time before relistening (sec)    */
#define TPREP           17.0        /* prep+InterPrep+PrepWait(sec)     */
#define SPH             3600.0      /* seconds per hour                 */
#define TRTRYEPSILON    1.0e-12     /* convergence epsilon for tRtry    */
#define MAXLOOP         1000        /* max count for main loop          */
#define MAXTETOE        9999.0      /* max end-to-end xfer time(sec)    */
#define FREQINCR        0.2         /* rate msg freq increases          */
#define TRTRYMAX        5000.0      /* overflow test on tRtry           */
#define OUTFILE         "mesh.dat"

int nNodes[] = {3,4,5,6,7,8,9,1000};    /* # nodes in each trial cluster */

/* FILE *openResultsFile(in:fileName)                                    */
/* Opens output file for results and then prints heading.                */
/* Returns file pointer.                                                 */

static FILE *openResultsFile(char *fileName)
{
        FILE *fp;
        int n;
```

```
        fp = fopen(fileName,"w+b");

        if (!fp) {
                printf("Unable to open %s for output\n",fileName);
                exit(0);
        }
        fprintf(fp,"%2d ",0);
        for (n = 0; n < sizeof(nNodes)/sizeof(int); n++)
                fprintf(fp,"%5d ",nNodes[n]);
        fprintf(fp,"\r\n");
        return fp;

} /*openResultsFile*/

/*****mainline*****/

void main() {

        FILE *fp;                       /* pointer to output file          */
        double w;                       /* freq msgs generated (msg/hr)    */
        double lambda;                  /* msg creation rate (msg/sec)     */
        double nXfr;                    /* # msgs xferred per connection   */
        double nConn;                   /* # connections/sec/node          */
        double tConn;                   /* time per connection (sec)       */
        double tEToE;                   /* end-to-end transfer time(sec)   */
        double tQ;                      /* local queuing time (sec)        */
        double pRemote;                 /* utilization of remote port      */
        double pPort;                   /* utiliziation of local port      */
        double p_Port;                  /* previous value of pPort         */
        double pCPU;                    /* utilization of  CPU is busy     */
        double tRtry;                   /* holding time due to retries     */
        double ptRtry;                  /* previous value of tRtry         */
        double nNode;                   /* # nodes in cluster              */
        int loopCnt;                    /* convergence control counter     */
        int n;                          /* iterator over nNode vector      */
        int testsFinished;              /* termination control counter     */

        fp = openResultsFile(OUTFILE);
```

```
w = FREQINCR;

while (w < 40.0) {

        printf("Number of msgs: %.1f\n",w);
        fprintf(fp,"%4.1f ",w);
        testsFinished = 0;

        for (n = 0; n < sizeof(nNodes)/sizeof(int); n++) {

                /* set starting conditions                              */

                nNode = nNodes[n];
                nXfr = 2.0;
                tConn = 0.0;
                loopCnt = 0;
                ppPort = pPort = pRemote = 0.0;
                tRtry = ptRtry = 0.0;
                lambda = w/SPH/(nNode-1);

                do {

                        /* The number of connections made is determined */
                        /* by the rate of msg creation and the number of */
                        /* msgs xferred during the connection.          */

                        nConn = 2.0*lambda/nXfr;

                        ptRtry = tRtry;

                        /* Connection time is based on 50% of the calls */
                        /* originating locally, 50% remotely.  Thus,    */
                        /* the dialling time is divided by two (on average). */

                        tConn = THUP+TRLISTN+TDIAL/2.0+TXFR*nXfr;

                        /* The local port is made busy by each connection */
```

```
/* request, whether successful or not.                    */
/* Approximately 25% of those requests which              */
/* fail are due to "no answer."                           */

pPort = nConn*(nNode-1)*(tConn+
        (TBUSY+ TRLISTN+THUP+
        0.25*(TDIAL-TBUSY+TLIDL+TRSET))*
        pRemote/(1-pRemote));

/* A moving avg dampens swings in pPort. */

pPort = (pPort+ppPort)/2.0;
if (pPort >= 1.0)
        pPort = 0.99;
ppPort = pPort;

/* Remote ports have one fewer node in their              */
/* clusters as viewed from the perspective of             */
/* the local node.  pPort is prorated accordingly         */

pRemote = pPort*(nNode-2.0)/(nNode-1.0);

/* There is a fallback upon connection failure            */
/* for all msgs in the mailbag.  The fallback             */
/* time is approximated as 1.5*RetryWait                  */
/* (eq. 4.4) or the msg interarrival time,                */
/* whichever is less.  The cumulative effect of           */
/* the fallback is a geometric series.  As both           */
/* local and remote ports can initiate                    */
/* connections, the delay is halved.                      */

if (1.5*TRTRY < 1/lambda)
        tRtry = (1.5*TRTRY)*pRemote/(1.0-
                pRemote)/2.0;
else
        tRtry = (1/lambda)*pRemote/(1.0-pRemote)/2.0;

/* A moving average is used to dampen swings in  */
```

```
        /* the computation of tRtry.                    */

        tRtry = (tRtry+ptRtry)/2.0;

        /* A check for non-convergence is performed.    */

        loopCnt++;
        if (loopCnt >= MAXLOOP || tRtry > TRTRYMAX) {
                loopCnt = MAXLOOP;
                break;
        }

        /* There is contention for the node's own port. */
        /* This results in a queueing delay which is    */
        /* accumulated with the priority wait and retry */
        /* time to determine the number of msgs in a    */
        /* mailbag.  Assume that the queueing delay is  */
        /* based on connection requests which are       */
        /* Poisson-distributed.                         */

        tQ = tConn*pPort/(1.0-pPort);

        /* The number of msgs in a mailbag is based on  */
        /* the msg arrival rate, the holding time of    */
        /* the mailbag, and the minimum initial         */
        /* requirement of one existing msg.  Since a    */
        /* node, on average, sends as many msgs as it   */
        /* receives, the total number of msgs xferred   */
        /* is doubled.                                  */

        nXfr = 2.0*(1.0+lambda*(tRtry+TPRIO+tQ));

} while (tRtry-ptRtry > TRTRYEPSILON ||
        ptRtry-tRtry > TRTRYEPSILON);

/* The utilization of the communication port places a   */
/* minimum requirement on the CPU utilization (as msg   */
/* preparation cannot occur simultaneously with msg     */
```

```
                /* transfer).  The time that CPU is in use is the time      */
                /* that it is busy managing communications plus the time  */
                /* it is preparing msgs (both incoming and outgoing).      */

                pCPU = pPort+2.0*TPREP*lambda*(nNode-1);
                if (pCPU >= 1.0)
                        pCPU = 0.99;

                /* End-to-End msg xfer time is the sum of the net node-   */
                /* to-node communications time and the preparation time. */

                tEToE = tRtry+TPRIO+tQ+tConn+2.0*TPREP/(1.0-pCPU);

                if (tEToE > MAXTETOE) {
                        tEToE = MAXTETOE;
                        testsFinished++;
                }
                fprintf(fp,"%5.0f ",tEToE);

        } /*for*/

        fprintf(fp,"\r\n");
        if (w > 1 && testsFinished == sizeof(nNodes)/sizeof(int))
                break;
        w += FREQINCR;

} /* while */

fclose(fp);
printf("output written to %s\n",OUTFILE);

} /* main */
```

Appendix D

C Source Listing of the Analytic Model for a Two-Server Star Cluster

```
/* star.cpp                                                              */
/* Written by J. McDaniel, March 24, 1993                                */
/*      Analytically computes the message transfer times of Health Link for */
/*      medium-priority msgs in a two-server star cluster.  Only nodes originate */
/*      calls; the hub (gateway) only accepts calls.                     */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define NSERVER         2.0         /* number of servers at gateway      */
#define TPRIO           300.0       /* MedPrioDelay (sec)                */
#define TDIAL           27.0        /* time to dial & connect(sec)       */
#define TBUSY           15.0        /* time to dial & find busy(sec)     */
#define TXFR            23.0        /* FDX transfer time (sec/msg)       */
#define THUP            7.0         /* time to hang up (sec)             */
#define TRTRY           180.0       /* retry wait (sec)                  */
#define TRLISTN         7.0         /* time before relistening (sec)     */
#define TPREP           17.0        /* prep+IntoPrep+PrepWait(sec)       */
#define SPH             3600.0      /* seconds per hour                  */
#define TRTRYEPSILON    1.0e-12     /* convergence epsilon for tRtry     */
#define MAXLOOP         1000        /* max count for main loop           */
#define MAXTETOE        9999.0      /* max end-to-end xfer time(sec)     */
#define FREQINCR        0.2         /* rate msg freq increases           */
#define TRTRYMAX        5000.0      /* overflow test on tRtry            */

int nNodes[] = {3,4,5,6,7,8,9,25};      /* # nodes in each trial cluster */

/* FILE *openResultsFile(in:fileName)                                    */
/* Opens output file for results and then prints heading.                */
/* Returns file pointer.                                                 */

static FILE *openResultsFile(char *fileName)
{
        FILE *fp;
        int n;

        fp = fopen(fileName,"w+b");
        if (!fp) {
```

```
            printf("Unable to open %s for output\n",fileName);
            exit(0);
        }
        fprintf(fp,"%2d ",0);
        for (n = 0; n < sizeof(nNodes)/sizeof(int); n++)
            fprintf(fp,"%5d ",nNodes[n]);
        fprintf(fp,"\r\n");
        return fp;

} /*openResultsFile*/

/*****mainline*****/

void main() {

        FILE *fp;                       /* pointer to output file          */
        double w;                       /* freq msgs generated (msg/hr)    */
        double lambda;                  /* msg creation rate (msg/sec)     */
        double nXfr;                    /* # msgs xferred per connection   */
        double nConn;                   /* # connections/sec/node          */
        double tConn;                   /* time per connection (sec)       */
        double tEToE;                   /* end-to-end transfer time (sec)  */
        double pPort;                   /* utilization of local port       */
        double pCPU;                    /* utilization of CPU              */
        double pGateway;                /* utilization of gateway          */
        double ppGateway;               /* previous pGateway               */
        double tRtry;                   /* holding time due to retries     */
        double ptRtry;                  /* previous value of tRtry         */
        double nNode;                   /* # nodes in cluster              */
        int loopCnt;                    /* convergence control counter     */
        int n;                          /* iterator over nNode vector      */
        int testsFinished;              /* termination control counter     */
        char outFile[13];               /* output file name                */

        sprintf(outFile,"star%.0fs.dat",NSERVER);
        fp = openResultsFile(outFile);

        w = FREQINCR;
        while (w < 40.0) {
```

```
printf("Number of msgs: %.1f\n",w);
fprintf(fp,"%4.1f ",w);
testsFinished = 0;

for (n = 0; n < sizeof(nNodes)/sizeof(int); n++) {

        /* set starting conditions                                  */

        nNode = nNodes[n];
        nXfr = 2.0;
        tConn = 0.0;
        loopCnt = 0;
        ppGateway = pGateway = 0.0;
        tRtry = ptRtry = 0.0;
        lambda = w/SPH;

        do {

                /* The number of connections made is determined */
                /* by the rate of msg arrival and the number of */
                /* msgs xferred during the connection.          */

                nConn = 2.0*lambda/nXfr;

                ptRtry = tRtry;

                /* The gateway is busy during msg xfer, hangup, */
                /* and relisten. The gateway does not originate */
                /* any calls. The probability that the gateway is */
                /* busy is a binomial distribution. The probability */
                /* is prorated since the caller is not currently */
                /* connected.                                   */

                pGateway = (nNode-1.0)*pow(nNode*nConn*
                        (THUP+TRLISTN+nXfr*TXFR)/NSERVER,
                        NSERVER)/nNode;

                /* A moving avg dampens swings in pGateway      */
```

```
pGateway = (pGateway+ppGateway)/2.0;
if (pGateway >= 1.0)
        pGateway = 0.99;
ppGateway = pGateway;

/* There is a fallback upon connection failure      */
/* for all msgs in the mailbag. The fallback        */
/* time is approximated as 1.5*RetryWait            */
/* (eq. 4.4) or the msg interarrival time,          */
/* whichever is less. The cumulative effect of the  */
/* fallback is a geometric series.                  */

if (1.5*TRTRY < 1/lambda)
        tRtry = 1.5*TRTRY*(pGateway/(1.0-pGateway));
else
        tRtry = (1/lambda)*(pGateway/(1.0-pGateway));

/* A moving average is used to dampen swings in     */
/* the computation of tRtry.                        */

tRtry = (tRtry+ptRtry)/2.0;

/* A check for non-convergence is performed.        */

loopCnt++;
if (loopCnt >= MAXLOOP || tRtry > TRTRYMAX) {
        loopCnt = MAXLOOP;
        break;
}

/* The number of msgs in a mailbag is based on      */
/* the msg arrival rate, the holding time of the    */
/* mailbag, and the minimum initial requirement     */
/* of one existing msg. Since a node, on average,   */
/* sends as many msgs as it receives, the total     */
/* number of msgs xferred is doubled.               */

nXfr = 2.0*(1.0+lambda*(tRtry+TPRIO));
```

```
} while (tRtry-ptRtry > TRTRYEPSILON ||
        ptRtry-tRtry > TRTRYEPSILON);

/* All calls originate locally. The connection time is the  */
/*    .ie to dial, xfer the msgs, hangup and relisten.      */
/* Assume that either the gateway answers or gives a busy   */
/* signal.                                                  */

tConn = THUP+TRLISTN+TDIAL+TXFR*nXfr;

/* The local port is made busy by each connection           */
/* request. Since lost calls are cleared at the gateway,    */
/* calls receiving a busy signal are retried. The           */
/* probability that the port is busy is a geometric series  */
/* based on the probability that gateway is busy.           */

pPort = nConn*(tConn+(TBUSY+TRLISTN+THUP)*
        pGateway/(1.0-pGateway));

/* The utilization of the communication port places a       */
/* minimum requirement on the CPU utilization (as msg       */
/* preparation cannot occur simultaneously with msg         */
/* transfer). The time that CPU is in use is the time that  */
/* it spends managing communications and the time it        */
/* spends preparing incoming and outgoing msgs.             */

pCPU = pPort+2.0*TPREP*lambda;
if (pCPU >= 1.0)
        pCPU = 0.99;

/* End-to-End msg xfer time is the sum of the net node-     */
/* to-node communications time and the preparation time.    */

tEToE = tRtry+TPRIO+2.0*tConn+
        2.0*TPREP/(1.0-pCPU)+1.0/nConn;

if (tEToE > MAXTETOE) {
        tEToE = MAXTETOE;
```

```
                    testsFinished++;
                }
                fprintf(fp,"%5.0f ",tEToE);

            } /*for*/

            fprintf(fp,"\r\n");
            if (w > 1 && testsFinished == sizeof(nNodes)/sizeof(int))
                break;
            w += FREQINCR;

        } /* while */

        fclose(fp);
        printf("output written to %s\n",outFile);

} /* main */
```

Appendix E

The Saskatchewan Peer-to-Peer Simulation Model: Configuration and Output from a Single Simulation Trial

Network configuration

Description: Saskatchewan Health Care System
Subdescription: Peer-to-Peer (Mesh) Model

Qty	Subnet	Scaling	# PORTS Intranet	# PORTS Internet
1	REGINA	1	32	7
3	REGIONAL	2	12	7
3	REGIONAL	1	8	7
1	SASKTOON	1	32	7

Timing parameters

	Node	Gateway
Dial time:	15	0
Ring and connect time:	12	3
Relisten time:	14	0
Reset time:	10	3
Line idle time:	60	10
FDX effective data rate:	362	362
HDX effective data rate:	770	770
Minimum retry wait:	180	20
Msg transfer time base:	2	2
Low priority delay:	14400	14400
Medium priority delay:	180	180
High priority delay:	0	0
Net processing time:	21	
Preparation cycle wait:	1	

Subnet classes

Subnet class: REGIONAL
Description: Regional Centre (60,000)
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	46	1	1	ADHOC
REGHOSP	2	1	2	ADHOC
SPEC	8	1	1	ADHOC

Subnet class: SASKTOON
Description: 235,268 pop.
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	338	1	1	ADHOC
MAINHOSP	4	1	8	ADHOC
MEDLAB	2	1	8	ADHOC
SPEC	299	1	1	ADHOC

Subnet class: REGINA
Description: 218,929 pop.
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	245	1	1	ADHOC
MAINHOSP	4	1	8	ADHOC
MEDLAB	2	1	8	ADHOC
PROV	1	1	8	ADHOC
PROVLAB	1	1	8	ADHOC
SPEC	185	1	1	ADHOC

Node classes

Node class: MAINHOSP
Description: Municipal Hospital (350 beds)
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY10	ADHOC	73	73	0	390	Med	Yes
MAINHOSP	MSGDAY10	ADHOC	3	3	0	500	Med	Yes
REGHOSP	MSGDAY10	ADHOC	20	0	20	500	Med	Yes
SPEC	MSGDAY10	ADHOC	61	61	0	390	Med	Yes

Node class: REGHOSP
Description: Regional Hospital (125 beds)
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY10	ADHOC	23	23	0	390	Med	Yes

MAINHOSP	MSGDAY10	ADHOC	8	0	8	500	Med	Yes
REGHOSP	MSGDAY10	ADHOC	3	3	0	500	Med	Yes
SPEC	MSGDAY10	ADHOC	4	4	0	390	Med	Yes

Node class: MEDLAB
Description: Medical Laboratory
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Prio-rity	Ack Req
GP	MSG10	DELVEVE1	497	200	48	175	Med	Yes
SPEC	MSG10	DELVEVE1	277	128	10	175	Med	Yes

Node class: PROVLAB
Description: Prov Lab: Vir/Cystology
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Prio-rity	Ack Req
GP	MSG10	DELVEVE1	298	245	748	175	Med	Yes
SPEC	MSG10	DELVEVE1	166	185	368	175	Med	Yes

Node class: SPEC
Description: Specialist
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Prio-rity	Ack Req
GP	MSGDAY1	ADHOC	5	4	1	200	Med	Yes
MAINHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
MEDLAB	NONE	POLL6	1	2	0	0	Low	No
PROV	MSGEVE1	ADHOC	2	1	1	2000	Med	Yes
PROVLAB	NONE	POLL6	1	1	0	0	Low	No
REGHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
SPEC	MSGDAY1	ADHOC	3	1	2	1000	Med	Yes

Node class: PROV
Description: Provincial Medical Services
Class enabled: Yes

Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Prio-rity	Ack Reqd
GP	MSG10	DELVEVE1	99	245	748	100	Med	No
SPEC	MSG10	DELVEVE1	55	185	368	100	Med	No

Node class: GP
Description: General Practitioner
Class enabled: Yes
Accept calls: Yes
Call sched to gateway: ADHOC
Gateway routing: No

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Prio-rity	Ack Reqd
GP	MSGDAY1	ADHOC	5	4	1	200	Med	Yes
MAINHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
MEDLAB	NONE	POLL6	1	2	0	0	Low	No
PROV	MSGEVE1	ADHOC	2	1	1	2000	Med	Yes
PROVLAB	NONE	POLL6	1	1	0	0	Low	No
REGHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
SPEC	MSGDAY1	ADHOC	5	3	2	1000	Med	Yes

Call and polling schedules

Schedule name: POLL6
Description: 6 Polls Per Day
Polling enabled: Yes
Precision: 120

Start	End	#calls	Start	End	#calls	Start	End	#calls
0:00	3:59	1	4:00	7:59	1	8:00	11:59	1
12:00	15:59	1	16:00	19:59	1	20:00	23:59	1

Schedule name: POLLDAY2
Description: 2 Polls Per Day
Polling enabled: Yes
Precision: 135

Start	End	#calls	Start	End	#calls	Start	End	#calls
8:30	12:59	1	13:00	17:29	1			

```
Schedule name:          DELVEVE1
Description:            1 Delivery Per Evening
Polling enabled:        No
Precision:                      60

Start    End   #calls    Start    End   #calls    Start    End   #calls
-----  -----  ------    -----  -----  ------    -----  -----  ------
20:00  21:59       1

Schedule name:          ADHOC
Description:            Delivery As Required
Polling enabled:        No
Precision:                       0
```

Message generation schedules

```
Schedule name:          MSGDAY10
Description:            10 Msgs Per Day
Enabled:                Yes
Precision:                      27

Start    End   #msgs    Start    End   #msgs    Start    End   #msgs
-----  -----  ------    -----  -----  ------    -----  -----  ------
 8:30   9:23       1     9:24  10:17       1    10:18  11:11       1
11:12  12:05       1    12:06  12:59       1    13:00  13:53       1
13:54  14:47       1    14:48  15:41       1    15:42  16:35       1
16:36  17:29       1

Schedule name:          MSGDAY1
Description:            One Msg Per Day
Enabled:                Yes
Precision:                     270

Start    End   #msgs    Start    End   #msgs    Start    End   #msgs
-----  -----  ------    -----  -----  ------    -----  -----  ------
 8:30  17:29       1

Schedule name:          MSGEVE1
Description:            One Msg Per Evening
Enabled:                Yes
Precision:                     120

Start    End   #msgs    Start    End   #msgs    Start    End   #msgs
-----  -----  ------    -----  -----  ------    -----  -----  ------
18:00  21:59       1

Schedule name:          MSG10
Description:            One Msg Every 144 Min
Enabled:                Yes
Precision:                      72
```

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
0:00	2:23	1	2:24	4:47	1	4:48	7:11	1
7:12	9:35	1	9:36	11:59	1	12:00	14:23	1
14:24	16:47	1	16:48	19:11	1	19:12	21:35	1
21:36	23:59	1						

Schedule name: MSG1
Description: One Msg Every 24 Hr
Enabled: Yes
Precision: 720

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
0:00	23:59	1						

Schedule name: NONE
Description: No Msgs
Enabled: Yes
Precision: 0

Warnings

subnet REGIONAL, node GP has no polling schedule enabled
subnet REGIONAL, node REGHOSP has no polling schedule enabled
subnet REGIONAL, node SPEC has no polling schedule enabled
subnet SASKTOON, node GP has no polling schedule enabled
subnet SASKTOON, node MAINHOSP has no polling schedule enabled
subnet SASKTOON, node MEDLAB has no polling schedule enabled
subnet SASKTOON, node SPEC has no polling schedule enabled
subnet REGINA, node GP has no polling schedule enabled
subnet REGINA, node MAINHOSP has no polling schedule enabled
subnet REGINA, node MEDLAB has no polling schedule enabled
subnet REGINA, node PROV has no polling schedule enabled
subnet REGINA, node PROVLAB has no polling schedule enabled
subnet REGINA, node SPEC has no polling schedule enabled
13 warnings reported

Simulation statistics

Start time:	24:00:00		
End time:	47:59:59		
Random number seed:	25253		
Generate log file:	Yes		
Gamma msg lengths:	Yes		

REGINA

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	9194	Av xmit time ±s.d.:	7±8
# conn originated:	1088	Orig conn time(min):	1063
Data sent(KB):	2506	Tot conn time(min):	2461
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	5164	Av xmit time ±s.d.:	13±14
# conn originated:	2875	Orig conn time(min):	1331
Data sent(KB):	4256	Tot conn time(min):	6173
Tot # msgs waiting:	423	# msgs wait send:	422
Tot # acks waiting:	120	# acks wait send:	113
Node class:	G	Number of nodes:	245
# msgs transmitted:	3669	Av xmit time ±s.d.:	9±11
# conn originated:	8169	Orig conn time(min):	1436
Data sent(KB):	2150	Tot conn time(min):	12528
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	1100	# acks wait send:	1094
# msgs delivered:	3669	Av dlvry time ±s.d.:	992±1582
Node class:	MAINHOSP	Number of nodes:	4
# msgs transmitted:	5764	Av xmit time ±s.d.:	7±6
# conn originated:	5054	Orig conn time(min):	763
Data sent(KB):	2223	Tot conn time(min):	6636
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	1	# acks wait send:	1
# msgs delivered:	5764	Av dlvry time ±s.d.:	637±793
Node class:	MEDLAB	Number of nodes:	2
# msgs transmitted:	15957	Av xmit time ±s.d.:	5±3
# conn originated:	933	Orig conn time(min):	395
Data sent(KB):	2721	Tot conn time(min):	4137
Tot # msgs waiting:	1206	# msgs wait send:	1187
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	15946	Av dlvry time ±s.d.:	9319±13679
Node class:	PROV	Number of nodes:	1
# msgs transmitted:	1737	Av xmit time ±s.d.:	3±1
# conn originated:	428	Orig conn time(min):	140
Data sent(KB):	196	Tot conn time(min):	1443
Tot # msgs waiting:	355	# msgs wait send:	355

Tot # acks waiting:	874	# acks wait send:	874
# msgs delivered:	1737	Av dlvry time ±s.d.:	15246±20830
Node class:	PROVLAB	Number of nodes:	1
# msgs transmitted:	5083	Av xmit time ±s.d.:	5±3
# conn originated:	328	Orig conn time(min):	364
Data sent(KB):	990	Tot conn time(min):	1474
Tot # msgs waiting:	110	# msgs wait send:	109
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	5064	Av dlvry time ±s.d.:	4199±9855
Node class:	SPEC	Number of nodes:	185
# msgs transmitted:	2391	Av xmit time ±s.d.:	8±10
# conn originated:	5627	Orig conn time(min):	880
Data sent(KB):	1147	Tot conn time(min):	8672
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	675	# acks wait send:	667
# msgs delivered:	2391	Av dlvry time ±s.d.:	898±1491
*Subnet total:	REGINA	Number of nodes:	438
# msgs transmitted:	34601	Av xmit time ±s.d.:	6±6
# conn originated:	20539	Orig conn time(min):	3981
Data sent(KB):	9427	Tot conn time(min):	34892
Tot # msgs waiting:	1671	# msgs wait send:	1651
Tot # acks waiting:	2650	# acks wait send:	2636
# msgs delivered:	24571	Av dlvry time ±s.d.:	5953±11914
REGIONAL			
Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	819	Av xmit time ±s.d.:	17±19
# conn originated:	176	Orig conn time(min):	201
Data sent(KB):	631	Tot conn time(min):	793
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	3797	Av xmit time ±s.d.:	6±6
# conn originated:	2921	Orig conn time(min):	453
Data sent(KB):	1092	Tot conn time(min):	3563
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	49	# acks wait send:	49
Node class:	GP	Number of nodes:	92
# msgs transmitted:	1353	Av xmit time ±s.d.:	10±14
# conn originated:	1477	Orig conn time(min):	248
Data sent(KB):	802	Tot conn time(min):	3789
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	64	# acks wait send:	64
# msgs delivered:	1340	Av dlvry time ±s.d.:	1793±2862
Node class:	REGHOSP	Number of nodes:	4
# msgs transmitted:	1668	Av xmit time ±s.d.:	7±5
# conn originated:	1397	Orig conn time(min):	213

Data sent(KB):	553	Tot conn time(min):	1975
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	1668	Av dlvry time ±s.d.:	1540±2197
Node class:	SPEC	Number of nodes:	16
# msgs transmitted:	190	Av xmit time ±s.d.:	9±12
# conn originated:	205	Orig conn time(min):	35
Data sent(KB):	101	Tot conn time(min):	810
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	12	# acks wait send:	12
# msgs delivered:	188	Av dlvry time ±s.d.:	2532±3547
*Subnet total:	REGIONAL	Number of nodes:	112
# msgs transmitted:	3211	Av xmit time ±s.d.:	8±10
# conn originated:	3079	Orig conn time(min):	496
Data sent(KB):	1455	Tot conn time(min):	6575
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	76	# acks wait send:	76
# msgs delivered:	3196	Av dlvry time ±s.d.:	1705±2597

REGIONAL(1)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	838	Av xmit time ±s.d.:	16±16
# conn originated:	219	Orig conn time(min):	199
Data sent(KB):	656	Tot conn time(min):	462
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	1213	Av xmit time ±s.d.:	8±9
# conn originated:	992	Orig conn time(min):	208
Data sent(KB):	507	Tot conn time(min):	1634
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	28	# acks wait send:	28
Node class:	GP	Number of nodes:	92
# msgs transmitted:	1357	Av xmit time ±s.d.:	10±12
# conn originated:	1483	Orig conn time(min):	252
Data sent(KB):	817	Tot conn time(min):	2729
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	16	# acks wait send:	16
# msgs delivered:	1354	Av dlvry time ±s.d.:	1401±3092
Node class:	REGHOSP	Number of nodes:	4
# msgs transmitted:	1408	Av xmit time ±s.d.:	8±6
# conn originated:	1178	Orig conn time(min):	210
Data sent(KB):	603	Tot conn time(min):	1724
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	1408	Av dlvry time ±s.d.:	799±1123
Node class:	SPEC	Number of nodes:	16

# msgs transmitted:	201	Av xmit time ±s.d.:	9±11
# conn originated:	212	Orig conn time(min):	38
Data sent(KB):	99	Tot conn time(min):	664
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	3	# acks wait send:	3
# msgs delivered:	202	Av dlvry time ±s.d.:	1572±3696

*Subnet total:	REGIONAL(1)	Number of nodes:	112
# msgs transmitted:	2966	Av xmit time ±s.d.:	9±9
# conn originated:	2873	Orig conn time(min):	501
Data sent(KB):	1520	Tot conn time(min):	5118
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	19	# acks wait send:	19
# msgs delivered:	2964	Av dlvry time ±s.d.:	1127±2447

REGIONAL(2)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	764	Av xmit time ±s.d.:	17±18
# conn originated:	225	Orig conn time(min):	178
Data sent(KB):	611	Tot conn time(min):	439
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	1133	Av xmit time ±s.d.:	9±10
# conn originated:	994	Orig conn time(min):	136
Data sent(KB):	488	Tot conn time(min):	1596
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	20	# acks wait send:	20
Node class:	GP	Number of nodes:	92
# msgs transmitted:	1408	Av xmit time ±s.d.:	10±11
# conn originated:	1565	Orig conn time(min):	266
Data sent(KB):	845	Tot conn time(min):	2953
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	25	# acks wait send:	24
# msgs delivered:	1399	Av dlvry time ±s.d.:	1504±2918
Node class:	REGHOSP	Number of nodes:	4
# msgs transmitted:	1484	Av xmit time ±s.d.:	8±5
# conn originated:	1296	Orig conn time(min):	231
Data sent(KB):	608	Tot conn time(min):	1871
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	1484	Av dlvry time ±s.d.:	1024±1286
Node class:	SPEC	Number of nodes:	16
# msgs transmitted:	207	Av xmit time ±s.d.:	9±13
# conn originated:	219	Orig conn time(min):	43
Data sent(KB):	111	Tot conn time(min):	700
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	6	# acks wait send:	6
# msgs delivered:	206	Av dlvry time ±s.d.:	1729±3085

*Subnet total:	REGIONAL(2)	Number of nodes:	112
# msgs transmitted:	3099	Av xmit time ±s.d.:	9±9
# conn originated:	3080	Orig conn time(min):	540
Data sent(KB):	1564	Tot conn time(min):	5525
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	31	# acks wait send:	30
# msgs delivered:	3089	Av dlvry time ±s.d.:	1288±2341

REGIONAL

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	429	Av xmit time ±s.d.:	17±16
# conn originated:	157	Orig conn time(min):	96
Data sent(KB):	349	Tot conn time(min):	256
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	655	Av xmit time ±s.d.:	8±10
# conn originated:	559	Orig conn time(min):	103
Data sent(KB):	277	Tot conn time(min):	888
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	14	# acks wait send:	14
Node class:	GP	Number of nodes:	46
# msgs transmitted:	715	Av xmit time ±s.d.:	10±11
# conn originated:	794	Orig conn time(min):	138
Data sent(KB):	440	Tot conn time(min):	1415
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	12	# acks wait send:	12
# msgs delivered:	709	Av dlvry time ±s.d.:	1319±3491
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	658	Av xmit time ±s.d.:	8±5
# conn originated:	564	Orig conn time(min):	103
Data sent(KB):	301	Tot conn time(min):	791
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	658	Av dlvry time ±s.d.:	599±846
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	91	Av xmit time ±s.d.:	8±12
# conn originated:	101	Orig conn time(min):	15
Data sent(KB):	41	Tot conn time(min):	354
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	4	# acks wait send:	4
# msgs delivered:	91	Av dlvry time ±s.d.:	1422±3797
*Subnet total:	REGIONAL	Number of nodes:	56
# msgs transmitted:	1464	Av xmit time ±s.d.:	9±9
# conn originated:	1459	Orig conn time(min):	257
Data sent(KB):	783	Tot conn time(min):	2561
Tot # msgs waiting:	0	# msgs wait send:	0

Tot # acks waiting:	16	# acks wait send:	16
# msgs delivered:	1458	Av dlvry time ±s.d.:	1000±2696

REGIONAL(1)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	373	Av xmit time ±s.d.:	17±21
# conn originated:	146	Orig conn time(min):	80
Data sent(KB):	285	Tot conn time(min):	247
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	727	Av xmit time ±s.d.:	6±7
# conn originated:	633	Orig conn time(min):	98
Data sent(KB):	226	Tot conn time(min):	894
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	12	# acks wait send:	12
Node class:	GP	Number of nodes:	46
# msgs transmitted:	678	Av xmit time ±s.d.:	9±11
# conn originated:	742	Orig conn time(min):	118
Data sent(KB):	367	Tot conn time(min):	1387
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	13	# acks wait send:	13
# msgs delivered:	676	Av dlvry time ±s.d.:	1114±3116
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	718	Av xmit time ±s.d.:	8±7
# conn originated:	627	Orig conn time(min):	101
Data sent(KB):	291	Tot conn time(min):	834
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	718	Av dlvry time ±s.d.:	715±1436
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	96	Av xmit time ±s.d.:	7±7
# conn originated:	103	Orig conn time(min):	15
Data sent(KB):	42	Tot conn time(min):	373
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	8	# acks wait send:	8
# msgs delivered:	96	Av dlvry time ±s.d.:	954±2040
*Subnet total:	REGIONAL(1)	Number of nodes:	56
# msgs transmitted:	1492	Av xmit time ±s.d.:	8±9
# conn originated:	1472	Orig conn time(min):	236
Data sent(KB):	699	Tot conn time(min):	2595
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	21	# acks wait send:	21
# msgs delivered:	1490	Av dlvry time ±s.d.:	911±2387

REGIONAL(2)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	391	Av xmit time ±s.d.:	17±18

# conn originated:	155	Orig conn time(min):	89
Data sent(KB):	330	Tot conn time(min):	247
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	656	Av xmit time ±s.d.:	8±7
# conn originated:	556	Orig conn time(min):	98
Data sent(KB):	268	Tot conn time(min):	859
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	20	# acks wait send:	20
Node class:	GP	Number of nodes:	46
# msgs transmitted:	668	Av xmit time ±s.d.:	9±11
# conn originated:	743	Orig conn time(min):	123
Data sent(KB):	390	Tot conn time(min):	1395
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	14	# acks wait send:	14
# msgs delivered:	664	Av dlvry time ±s.d.:	1202±2684
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	776	Av xmit time ±s.d.:	7±5
# conn originated:	683	Orig conn time(min):	100
Data sent(KB):	268	Tot conn time(min):	901
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	776	Av dlvry time ±s.d.:	629±1357
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	97	Av xmit time ±s.d.:	9±10
# conn originated:	96	Orig conn time(min):	17
Data sent(KB):	50	Tot conn time(min):	308
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	2	# acks wait send:	2
# msgs delivered:	96	Av dlvry time ±s.d.:	790±1577
*Subnet total:	REGIONAL(2)	Number of nodes:	56
# msgs transmitted:	1541	Av xmit time ±s.d.:	8±9
# conn originated:	1522	Orig conn time(min):	242
Data sent(KB):	709	Tot conn time(min):	2605
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	16	# acks wait send:	16
# msgs delivered:	1536	Av dlvry time ±s.d.:	886±2067
SASKTOON			
Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	6093	Av xmit time ±s.d.:	11±13
# conn originated:	737	Orig conn time(min):	1190
Data sent(KB):	3515	Tot conn time(min):	1904
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	5549	Av xmit time ±s.d.:	7±7
# conn originated:	5149	Orig conn time(min):	820

Data sent(KB):	1770	Tot conn time(min):	8169
Tot # msgs waiting:	47	# msgs wait send:	37
Tot # acks waiting:	559	# acks wait send:	557
Node class:	GP	Number of nodes:	338
# msgs transmitted:	5048	Av xmit time ±s.d.:	9±11
# conn originated:	9187	Orig conn time(min):	1649
Data sent(KB):	2916	Tot conn time(min):	14310
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	1095	# acks wait send:	1093
# msgs delivered:	5074	Av dlvry time ±s.d.:	2763±4283
Node class:	MAINHOST	Number of nodes:	4
# msgs transmitted:	6188	Av xmit time ±s.d.:	7±6
# conn originated:	5301	Orig conn time(min):	829
Data sent(KB):	2324	Tot conn time(min):	7249
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	4	# acks wait send:	4
# msgs delivered:	6188	Av dlvry time ±s.d.:	1638±2814
Node class:	MEDLAB	Number of nodes:	2
# msgs transmitted:	16091	Av xmit time ±s.d.:	5±3
# conn originated:	981	Orig conn time(min):	371
Data sent(KB):	2812	Tot conn time(min):	4687
Tot # msgs waiting:	1206	# msgs wait send:	1196
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	16175	Av dlvry time ±s.d.:	8528±12291
Node class:	SPEC	Number of nodes:	299
# msgs transmitted:	3818	Av xmit time ±s.d.:	8±10
# conn originated:	6916	Orig conn time(min):	1082
Data sent(KB):	1871	Tot conn time(min):	10456
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	711	# acks wait send:	711
# msgs delivered:	3847	Av dlvry time ±s.d.:	2818±4482
*Subnet total:	SASKTOON	Number of nodes:	643
# msgs transmitted:	31145	Av xmit time ±s.d.:	6±7
# conn originated:	22385	Orig conn time(min):	3932
Data sent(KB):	9922	Tot conn time(min):	36705
Tot # msgs waiting:	1206	# msgs wait send:	1196
Tot # acks waiting:	1810	# acks wait send:	1808
# msgs delivered:	31284	Av dlvry time ±s.d.:	5527±9742
Internet total		Number of nodes:	N/A
# msgs transmitted:	18901	Av xmit time ±s.d.:	10±13
# conn originated:	2903	Orig conn time(min):	3100
Data sent(KB):	8883	Tot conn time(min):	6812
Gateway total		Number of nodes:	8
# msgs transmitted:	18894	Av xmit time ±s.d.:	9±10

# conn originated:	14679	Orig conn time(min):	3302
Data sent(KB):	6885	Tot conn time(min):	23780
Tot # msgs waiting:	470	# msgs wait send:	459
Tot # acks waiting:	822	# acks wait send:	813
Node class total		Number of nodes:	1585
# msgs transmitted:	79519	Av xmit time ±s.d.:	7±7
# conn originated:	56409	Orig conn time(min):	10190
Data sent(KB):	26079	Tot conn time(min):	96578
Tot # msgs waiting:	2877	# msgs wait send:	2847
Tot # acks waiting:	4639	# acks wait send:	4622
# msgs delivered:	79568	Av dlvry time ±s.d.:	497±10147

Appendix F

The Saskatchewan Client-Server Simulation Model: Configuration and Output from a Single Simulation Trial

Network configuration

Description: Saskatchewan Health Care System
Subdescription: Client-Server (Star) Model

Qty	Subnet	Scaling	# PORTS Intranet	# PORTS Internet
1	REGINA	1	64	7
3	REGIONAL	2	12	7
3	REGIONAL	1	8	7
1	SASKTOON	1	64	7

Timing parameters

	Node	Gateway
Dial time:	15	0
Ring and connect time:	12	3
Relisten time:	14	0
Reset time:	10	3
Line idle time:	60	10
FDX effective data rate:	362	362
HDX effective data rate:	770	770
Minimum retry wait:	180	20
Msg transfer time base:	2	2
Low priority delay:	14400	0
Medium priority delay:	180	0
High priority delay:	0	0
Net processing time:	21	
Preparation cycle wait:	1	

Subnet classes

Subnet class: REGIONAL
Description: Regional Centre (60,000)
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	46	1	1	HOLD
REGHOSP	2	1	2	HOLD
SPEC	8	1	1	HOLD

Subnet class: SASKTOON
Description: 235,268 pop.
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	338	1	1	HOLD
MAINHOSP	4	1	8	HOLD
MEDLAB	2	1	8	HOLD
SPEC	299	1	1	HOLD

Subnet class: REGINA
Description: 218,929 pop.
Class enabled: Yes

Node class	# of nodes	Node scaling	Node #ports	Gateway call sched to node
GP	245	1	1	HOLD
MAINHOSP	4	1	8	HOLD
MEDLAB	2	1	8	HOLD
PROV	1	1	8	HOLD
PROVLAB	1	1	8	HOLD
SPEC	185	1	1	HOLD

Node classes

Node class: MAINHOSP
Description: Municipal Hospital (350 beds)
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY10	ADHOC	73	73	0	390	Med	Yes
MAINHOSP	MSGDAY10	ADHOC	3	3	0	500	Med	Yes
REGHOSP	MSGDAY10	ADHOC	20	0	20	500	Med	Yes
SPEC	MSGDAY10	ADHOC	61	61	0	390	Med	Yes

Node class: REGHOSP
Description: Regional Hospital (125 beds)
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY10	ADHOC	23	23	0	390	Med	Yes

MAINHOSP	MSGDAY10	ADHOC	8	0	8	500	Med	Yes
REGHOSP	MSGDAY10	ADHOC	3	3	0	500	Med	Yes
SPEC	MSGDAY10	ADHOC	4	4	0	390	Med	Yes

Node class: MEDLAB
Description: Medical Laboratory
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSG10	ADHOC	497	200	48	175	Med	Yes
SPEC	MSG10	ADHOC	277	128	10	175	Med	Yes

Node class: PROVLAB
Description: Prov Lab: Vir/Cystology
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSG10	ADHOC	298	245	748	175	Med	Yes
SPEC	MSG10	ADHOC	166	185	368	175	Med	Yes

Node class: SPEC
Description: Specialist
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY1	ADHOC	5	4	1	200	Med	Yes
MAINHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
PROV	MSGEVE1	ADHOC	2	1	1	2000	Med	Yes
REGHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
SPEC	MSGDAY1	ADHOC	3	1	2	1000	Med	Yes

Node class: PROV
Description: Provincial Medical Services
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24

Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSG10	ADHOC	99	245	748	100	Med	No
SPEC	MSG10	ADHOC	55	185	368	100	Med	No

Node class: GP
Description: General Practitioner
Class enabled: Yes
Accept calls: No
Call sched to gateway: POLL24
Gateway routing: Yes

Dest Node	Message Schedule	Call Schedule	Freq mult	#nodes intra	#nodes inter	Msg Size	Priority	Ack Reqd
GP	MSGDAY1	ADHOC	5	4	1	200	Med	Yes
MAINHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
PROV	MSGEVE1	ADHOC	2	1	1	2000	Med	Yes
REGHOSP	MSGDAY1	ADHOC	4	1	0	175	Med	Yes
SPEC	MSGDAY1	ADHOC	5	3	2	1000	Med	Yes

Call and polling schedules

Schedule name: POLL24
Description: One Poll Hourly
Polling enabled: Yes
Precision: 29

Start	End	#calls	Start	End	#calls	Start	End	#calls
0:00	2:59	3	3:00	5:59	3	6:00	8:59	3
9:00	11:59	3	12:00	14:59	3	15:00	17:59	3
18:00	20:59	3	21:00	23:59	3			

Schedule name: HOLD
Description: Hold Until Polled
Polling enabled: No
Precision: 0

Start	End	#calls	Start	End	#calls	Start	End	#calls
0:00	0:00	1						

Schedule name: ADHOC
Description: Delivery As Required
Polling enabled: No
Precision: 0

Message generation schedules

Schedule name: MSGDAY10
Description: 10 Msgs Per Day
Enabled: Yes
Precision: 27

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
8:30	9:23	1	9:24	10:17	1	10:18	11:11	1
11:12	12:05	1	12:06	12:59	1	13:00	13:53	1
13:54	14:47	1	14:48	15:41	1	15:42	16:35	1
16:36	17:29	1						

Schedule name: MSGDAY1
Description: One Msg Per Day
Enabled: Yes
Precision: 270

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
8:30	17:29	1						

Schedule name: MSGEVE1
Description: One Msg Per Evening
Enabled: Yes
Precision: 120

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
18:00	21:59	1						

Schedule name: MSG10
Description: One Msg Every 144 Min
Enabled: Yes
Precision: 72

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
0:00	2:23	1	2:24	4:47	1	4:48	7:11	1
7:12	9:35	1	9:36	11:59	1	12:00	14:23	1
14:24	16:47	1	16:48	19:11	1	19:12	21:35	1
21:36	23:59	1						

Schedule name: MSG1
Description: One Msg Every 24 Hr
Enabled: Yes
Precision: 720

Start	End	#msgs	Start	End	#msgs	Start	End	#msgs
0:00	23:59	1						

Schedule name: NONE
Description: No Msgs
Enabled: Yes
Precision: 0

Warnings

subnet REGIONAL, node GP has no polling schedule enabled
subnet REGIONAL, node REGHOSP has no polling schedule enabled
subnet REGIONAL, node SPEC has no polling schedule enabled
subnet SASKTOON, node GP has no polling sch le enabled
subnet SASKTOON, node MAINHOSP has no polling schedule enabled
subnet SASKTOON, node MEDLAB has no polling schedule enabled
subnet SASKTOON, node SPEC has no polling schedule enabled
subnet REGINA, node GP has no polling schedule enabled
subnet REGINA, node MAINHOSP has no polling schedule enabled
subnet REGINA, node MEDLAB has no polling schedule enabled
subnet REGINA, node PROV has no polling schedule enabled
subnet REGINA, node PROVLAB has no polling schedule enabled
subnet REGINA, node SPEC has no polling schedule enabled
13 warnings reported

Simulation statistics

Start time:	24:00:00		
End time:	47:59:59		
Random number seed:	8705		
Generate log file:	Yes		
Gamma msg lengths:	Yes		

REGINA

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	8194	Av xmit time ±s.d.:	6±7
# conn originated:	4256	Orig conn time(min):	1169
Data sent(KB):	2137	Tot conn time(min):	3198
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	30002	Av xmit time ±s.d.:	8±9
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	10954	Tot conn time(min):	11601
Tot # msgs waiting:	1140	# msgs wait send:	1130
Tot # acks waiting:	656	# acks wait send:	458
Node class:	GP	Number of nodes:	245
# msgs transmitted:	3762	Av xmit time ±s.d.:	12±14
# conn originated:	6256	Orig conn time(min):	2067
Data sent(KB):	2158	Tot conn time(min):	7385
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	288	# acks wait send:	275
# msgs delivered:	3773	Av dlvry time ±s.d.:	5473±4391
Node class:	MAINHOSP	Number of nodes:	4
# msgs transmitted:	6196	Av xmit time ±s.d.:	9±6
# conn originated:	77	Orig conn time(min):	1067
Data sent(KB):	2338	Tot conn time(min):	1133
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	6196	Av dlvry time ±s.d.:	3961±2311
Node class:	MEDLAB	Number of nodes:	2
# msgs transmitted:	13768	Av xmit time ±s.d.:	4±3
# conn originated:	43	Orig conn time(min):	1140
Data sent(KB):	2286	Tot conn time(min):	1177
Tot # msgs waiting:	629	# msgs wait send:	586
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	13840	Av dlvry time ±s.d.:	4797±3151
Node class:	PROV	Number of nodes:	1
# msgs transmitted:	1516	Av xmit time ±s.d.:	3±1
# conn originated:	15	Orig conn time(min):	844
Data sent(KB):	142	Tot conn time(min):	857
Tot # msgs waiting:	376	# msgs wait send:	376

Tot # acks waiting:	745	# acks wait send:	744
# msgs delivered:	1517	Av dlvry time ±s.d.:	15364±15316
Node class:	PROVLAB	Number of nodes:	1
# msgs transmitted:	4279	Av xmit time ±s.d.:	4±2
# conn originated:	20	Orig conn time(min):	342
Data sent(KB):	650	Tot conn time(min):	359
Tot # msgs waiting:	184	# msgs wait send:	153
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	4548	Av dlvry time ±s.d.:	5964±4644
Node class:	SPEC	Number of nodes:	185
# msgs transmitted:	2388	Av xmit time ±s.d.:	10±12
# conn originated:	4522	Orig conn time(min):	1860
Data sent(KB):	1177	Tot conn time(min):	5703
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	179	# acks wait send:	177
# msgs delivered:	2391	Av dlvry time ±s.d.:	5461±4611
*Subnet total:	REGINA	Number of nodes:	438
# msgs transmitted:	31909	Av xmit time ±s.d.:	6±7
# conn originated:	10933	Orig conn time(min):	7324
Data sent(KB):	8750	Tot conn time(min):	16616
Tot # msgs waiting:	1189	# msgs wait send:	1115
Tot # acks waiting:	1212	# acks wait send:	1196
# msgs delivered:	32265	Av dlvry time ±s.d.:	5426±5333
REGIONAL			
Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	834	Av xmit time ±s.d.:	12±12
# conn originated:	1578	Orig conn time(min):	306
Data sent(KB):	588	Tot conn time(min):	1053
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	6015	Av xmit time ±s.d.:	8±8
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	1937	Tot conn time(min):	2228
Tot # msgs waiting:	61	# msgs wait send:	61
Tot # acks waiting:	6	# acks wait send:	6
Node class:	GP	Number of nodes:	92
# msgs transmitted:	1398	Av xmit time ±s.d.:	11±14
# conn originated:	2232	Orig conn time(min):	574
Data sent(KB):	775	Tot conn time(min):	2471
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	41	# acks wait send:	41
# msgs delivered:	1402	Av dlvry time ±s.d.:	5317±4682
Node class:	REGHOSP	Number of nodes:	4
# msgs transmitted:	1626	Av xmit time ±s.d.:	9±7
# conn originated:	83	Orig conn time(min):	305

Data sent(KB):		586	Tot conn time(min):	376
Tot # msgs waiting:		0	# msgs wait send:	0
Tot # acks waiting:		0	# acks wait send:	0
# msgs delivered:		1629	Av dlvry time ±s.d.:	4236±2475
Node class:	SPEC		Number of nodes:	16
# msgs transmitted:		201	Av xmit time ±s.d.:	11±14
# conn originated:		346	Orig conn time(min):	304
Data sent(KB):		98	Tot conn time(min):	598
Tot # msgs waiting:		0	# msgs wait send:	0
Tot # acks waiting:		17	# acks wait send:	17
# msgs delivered:		200	Av dlvry time ±s.d.:	5137±3300
*Subnet total:	REGIONAL		Number of nodes:	112
# msgs transmitted:		3225	Av xmit time ±s.d.:	10±11
# conn originated:		2661	Orig conn time(min):	1184
Data sent(KB):		1460	Tot conn time(min):	3446
Tot # msgs waiting:		0	# msgs wait send:	0
Tot # acks waiting:		58	# acks wait send:	58
# msgs delivered:		3231	Av dlvry time ±s.d.:	4761±3681
REGIONAL(1)				
Node class:	Internet		Number of nodes:	N/A
# msgs transmitted:		712	Av xmit time ±s.d.:	12±12
# conn originated:		1078	Orig conn time(min):	204
Data sent(KB):		543	Tot conn time(min):	616
Node class:	Gateway		Number of nodes:	1
# msgs transmitted:		3522	Av xmit time ±s.d.:	9±8
# conn originated:		0	Orig conn time(min):	0
Data sent(KB):		1301	Tot conn time(min):	1888
Tot # msgs waiting:		11	# msgs wait send:	11
Tot # acks waiting:		0	# acks wait send:	0
Node class:	GP		Number of nodes:	92
# msgs transmitted:		1375	Av xmit time ±s.d.:	10±12
# conn originated:		2127	Orig conn time(min):	434
Data sent(KB):		747	Tot conn time(min):	2242
Tot # msgs waiting:		0	# msgs wait send:	0
Tot # acks waiting:		23	# acks wait send:	23
# msgs delivered:		1377	Av dlvry time ±s.d.:	5151±4203
Node class:	REGHOSP		Number of nodes:	4
# msgs transmitted:		1512	Av xmit time ±s.d.:	9±7
# conn originated:		103	Orig conn time(min):	288
Data sent(KB):		579	Tot conn time(min):	376
Tot # msgs waiting:		0	# msgs wait send:	0
Tot # acks waiting:		0	# acks wait send:	0
# msgs delivered:		1512	Av dlvry time ±s.d.:	4009±2043
Node class:	SPEC		Number of nodes:	16

# msgs transmitted:	206	Av xmit time ±s.d.:	10±12
# conn originated:	308	Orig conn time(min):	190
Data sent(KB):	104	Tot conn time(min):	452
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	1	# acks wait send:	1
# msgs delivered:	204	Av dlvry time ±s.d.:	5521±4546

*Subnet total:	REGIONAL(1)	Number of nodes:	112
# msgs transmitted:	3093	Av xmit time ±s.d.:	10±10
# conn originated:	2538	Orig conn time(min):	913
Data sent(KB):	1430	Tot conn time(min):	3070
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	24	# acks wait send:	24
# msgs delivered:	3093	Av dlvry time ±s.d.:	4617±3409

REGIONAL(2)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	861	Av xmit time ±s.d.:	12±14
# conn originated:	1137	Orig conn time(min):	246
Data sent(KB):	668	Tot conn time(min):	679

Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	3405	Av xmit time ±s.d.:	10±9
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	1340	Tot conn time(min):	1930
Tot # msgs waiting:	4	# msgs wait send:	4
Tot # acks waiting:	4	# acks wait send:	4

Node class:	GP	Number of nodes:	92
# msgs transmitted:	1375	Av xmit time ±s.d.:	11±13
# conn originated:	2106	Orig conn time(min):	437
Data sent(KB):	782	Tot conn time(min):	2228
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	12	# acks wait send:	12
# msgs delivered:	1365	Av dlvry time ±s.d.:	5096±4234

Node class:	REGHOSP	Number of nodes:	4
# msgs transmitted:	1656	Av xmit time ±s.d.:	10±7
# conn originated:	87	Orig conn time(min):	320
Data sent(KB):	663	Tot conn time(min):	394
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	1656	Av dlvry time ±s.d.:	4197±2169

Node class:	SPEC	Number of nodes:	16
# msgs transmitted:	215	Av xmit time ±s.d.:	11±14
# conn originated:	333	Orig conn time(min):	206
Data sent(KB):	125	Tot conn time(min):	490
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	2	# acks wait send:	2
# msgs delivered:	218	Av dlvry time ±s.d.:	5390±4735

*Subnet total:	REGIONAL(2)	Number of nodes:	112
# msgs transmitted:	3246	Av xmit time ±s.d.:	10±10
# conn originated:	2526	Orig conn time(min):	965
Data sent(KB):	1570	Tot conn time(min):	3112
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	14	# acks wait send:	14
# msgs delivered:	3239	Av dlvry time ±s.d.:	4656±3418

REGIONAL

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	410	Av xmit time ±s.d.:	12±11
# conn originated:	671	Orig conn time(min):	124
Data sent(KB):	313	Tot conn time(min):	362
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	1701	Av xmit time ±s.d.:	9±9
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	649	Tot conn time(min):	967
Tot # msgs waiting:	6	# msgs wait send:	6
Tot # acks waiting:	1	# acks wait send:	1
Node class:	GP	Number of nodes:	46
# msgs transmitted:	721	Av xmit time ±s.d.:	11±12
# conn originated:	1112	Orig conn time(min):	227
Data sent(KB):	407	Tot conn time(min):	1172
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	9	# acks wait send:	9
# msgs delivered:	729	Av dlvry time ±s.d.:	5285±4506
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	730	Av xmit time ±s.d.:	9±5
# conn originated:	46	Orig conn time(min):	139
Data sent(KB):	275	Tot conn time(min):	178
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	730	Av dlvry time ±s.d.:	4667±2283
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	116	Av xmit time ±s.d.:	10±12
# conn originated:	148	Orig conn time(min):	102
Data sent(KB):	62	Tot conn time(min):	228
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	2	# acks wait send:	2
# msgs delivered:	118	Av dlvry time ±s.d.:	5446±4271

*Subnet total:	REGIONAL	Number of nodes:	56
# msgs transmitted:	1567	Av xmit time ±s.d.:	10±10
# conn originated:	1306	Orig conn time(min):	469
Data sent(KB):	744	Tot conn time(min):	1579
Tot # msgs waiting:	0	# msgs wait send:	0

Tot # acks waiting:	11	# acks wait send:	11
# msgs delivered:	1577	Av dlvry time ±s.d.:	5011±3640

REGIONAL(1)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	487	Av xmit time ±s.d.:	11±10
# conn originated:	652	Orig conn time(min):	118
Data sent(KB):	326	Tot conn time(min):	346
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	1515	Av xmit time ±s.d.:	9±9
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	566	Tot conn time(min):	903
Tot # msgs waiting:	6	# msgs wait send:	6
Tot # acks waiting:	0	# acks wait send:	0
Node class:	GP	Number of nodes:	46
# msgs transmitted:	687	Av xmit time ±s.d.:	11±12
# conn originated:	1006	Orig conn time(min):	200
Data sent(KB):	386	Tot conn time(min):	1055
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	4	# acks wait send:	4
# msgs delivered:	685	Av dlvry time ±s.d.:	4881±4281
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	712	Av xmit time ±s.d.:	9±6
# conn originated:	42	Orig conn time(min):	131
Data sent(KB):	283	Tot conn time(min):	167
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	712	Av dlvry time ±s.d.:	4339±2085
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	103	Av xmit time ±s.d.:	10±12
# conn originated:	189	Orig conn time(min):	95
Data sent(KB):	52	Tot conn time(min):	256
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	1	# acks wait send:	1
# msgs delivered:	105	Av dlvry time ±s.d.:	5305±4686
*Subnet total:	REGIONAL(1)	Number of nodes:	56
# msgs transmitted:	1502	Av xmit time ±s.d.:	10±10
# conn originated:	1237	Orig conn time(min):	427
Data sent(KB):	721	Tot conn time(min):	1479
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	5	# acks wait send:	5
# msgs delivered:	1502	Av dlvry time ±s.d.:	4654±3468

REGIONAL(2)

Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	391	Av xmit time ±s.d.:	12±12

# conn originated:	674	Orig conn time(min):	129
Data sent(KB):	295	Tot conn time(min):	371
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	1706	Av xmit time ±s.d.:	9±9
# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	633	Tot conn time(min):	936
Tot # msgs waiting:	5	# msgs wait send:	5
Tot # acks waiting:	1	# acks wait send:	1
Node class:	GP	Number of nodes:	46
# msgs transmitted:	688	Av xmit time ±s.d.:	11±13
# conn originated:	1051	Orig conn time(min):	211
Data sent(KB):	394	Tot conn time(min):	1104
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	8	# acks wait send:	8
# msgs delivered:	685	Av dlvry time ±s.d.:	5045±4445
Node class:	REGHOSP	Number of nodes:	2
# msgs transmitted:	688	Av xmit time ±s.d.:	9±7
# conn originated:	48	Orig conn time(min):	132
Data sent(KB):	237	Tot conn time(min):	173
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	688	Av dlvry time ±s.d.:	4418±2254
Node class:	SPEC	Number of nodes:	8
# msgs transmitted:	113	Av xmit time ±s.d.:	10±13
# conn originated:	186	Orig conn time(min):	101
Data sent(KB):	54	Tot conn time(min):	259
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	114	Av dlvry time ±s.d.:	4408±3437
*Subnet total:	REGIONAL(2)	Number of nodes:	56
# msgs transmitted:	1489	Av xmit time ±s.d.:	10±11
# conn originated:	1285	Orig conn time(min):	444
Data sent(KB):	686	Tot conn time(min):	1537
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	8	# acks wait send:	8
# msgs delivered:	1487	Av dlvry time ±s.d.:	4706±3527
SASKTOON			
Node class:	Internet	Number of nodes:	N/A
# msgs transmitted:	5641	Av xmit time ±s.d.:	10±12
# conn originated:	2502	Orig conn time(min):	686
Data sent(KB):	3352	Tot conn time(min):	2229
Node class:	Gateway	Number of nodes:	1
# msgs transmitted:	27750	Av xmit time ±s.d.:	7±7
# conn originated:	0	Orig conn time(min):	0

Data sent(KB):	7388	Tot conn time(min):	12353
Tot # msgs waiting:	410	# msgs wait send:	408
Tot # acks waiting:	838	# acks wait send:	819
Node class:	GP	Number of nodes:	338
# msgs transmitted:	5017	Av xmit time ±s.d.:	11±13
# conn originated:	8252	Orig conn time(min):	2106
Data sent(KB):	2710	Tot conn time(min):	9120
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	157	# acks wait send:	156
# msgs delivered:	5011	Av dlvry time ±s.d.:	5252±4218
Node class:	MAINHOSP	Number of nodes:	4
# msgs transmitted:	5926	Av xmit time ±s.d.:	9±7
# conn originated:	81	Orig conn time(min):	1123
Data sent(KB):	2285	Tot conn time(min):	1192
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	5926	Av dlvry time ±s.d.:	3717±2090
Node class:	MEDLAB	Number of nodes:	2
# msgs transmitted:	14205	Av xmit time ±s.d.:	4±3
# conn originated:	37	Orig conn time(min):	1169
Data sent(KB):	2329	Tot conn time(min):	1200
Tot # msgs waiting:	670	# msgs wait send:	657
Tot # acks waiting:	0	# acks wait send:	0
# msgs delivered:	14304	Av dlvry time ±s.d.:	4795±3354
Node class:	SPEC	Number of nodes:	299
# msgs transmitted:	3985	Av xmit time ±s.d.:	10±13
# conn originated:	6911	Orig conn time(min):	1926
Data sent(KB):	2018	Tot conn time(min):	7800
Tot # msgs waiting:	0	# msgs wait send:	0
Tot # acks waiting:	107	# acks wait send:	103
# msgs delivered:	3986	Av dlvry time ±s.d.:	5326±4686
*Subnet total:	SASKTOON	Number of nodes:	643
# msgs transmitted:	29133	Av xmit time ±s.d.:	7±8
# conn originated:	15281	Orig conn time(min):	6324
Data sent(KB):	9342	Tot conn time(min):	19313
Tot # msgs waiting:	670	# msgs wait send:	657
Tot # acks waiting:	264	# acks wait send:	259
# msgs delivered:	29227	Av dlvry time ±s.d.:	4727±3569
Internet total		Number of nodes:	N/A
# msgs transmitted:	17530	Av xmit time ±s.d.:	8±10
# conn originated:	12548	Orig conn time(min):	2985
Data sent(KB):	8224	Tot conn time(min):	8857
Gateway total		Number of nodes:	8
# msgs transmitted:	75616	Av xmit time ±s.d.:	8±8

# conn originated:	0	Orig conn time(min):	0
Data sent(KB):	24769	Tot conn time(min):	32811
Tot # msgs waiting:	1643	# msgs wait send:	1631
Tot # acks waiting:	1506	# acks wait send:	1289
Node class total		Number of nodes:	1585
# msgs transmitted:	75164	Av xmit time ±s.d.:	7±8
# conn originated:	37767	Orig conn time(min):	18054
Data sent(KB):	24703	Tot conn time(min):	50155
Tot # msgs waiting:	1859	# msgs wait send:	1772
Tot # acks waiting:	1596	# acks wait send:	1575
# msgs delivered:	75621	Av dlvry time ±s.d.:	5023±4415

Appendix G

Glossary of Acronyms

3I	Dutch EDI project: Inter-Institutional Information Exchange
ACR/NEMA	American College of Radiologists/National Electronic Manufacturers Association
ACSE	Association Control Service Element
AI	Artificial Intelligence
AIM	European Advanced Informatics in Medicine
ANSI	American National Standards Institute
API	Application Programming Interface
ASC	accredited ANSI committee for standards development
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
ASTM	American Society of Testing and Materials
BAZIS	Dutch consortium of hospitals maintaining a common hospital information system
BER	Basic Encoding Rules
BSD	Berkeley Standard Distribution
CASE	Common Application Service Element
CCITT	International Consultative Committee for Telephony and Telegraphy
CMIS	Common Management Information Service
COPA	Dutch EDI project: Communications Project Apeldorn
COSIT	NHS Central OSI Team
CPU	Central Processing Unit
CR	Carriage Return character (ASCII 13)
CRC	Cyclical Redundancy Check
CSNET	NSF Computer Science Network
DARPA	U.S. Defence Advanced Projects Research Agency
DCE	Data Communicating Equipment
DEB	U.K. Dental Estimates Board
DEC	Digital Equipment Corporation
DES	Data Encryption Standard
DIV	Dutch networking corporation: Dienst Informatie Verwerking
DOS	Disk Operating System developed for IBM PC's

ECL	Essential Communications Library
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
ELIAS	Dutch project to develop an electronic patient management system
EUCLIDES	European standard for Clinical Laboratory Data Exchange
EUREKA	Consortium of vendors and EUCLIDES researchers
FAX	Facsimile transmission
FCFS	First-Come, First-Served
FidoNet	Electronic mail network based on telephone dial-up
FTAM	File Transfer, Access and Management
FTE	Full-Time Employee
FTP	File Transfer Protocol
GEIN	Dutch feasibility study on a large scale EDI network in Breda
GNP	Gross National Product
GOSIP	U.S. Government OSI Profile
GP	General Practitioner
GPSS	General Purpose Systems Simulator
GUI	Graphical User Interface
HL7	Health Level 7 data interchange standard
IBM	International Business Machines Corporation
ICI	Interface Control Information
IEEE	Institute of Electrical and Electronic Engineers
IMIA	International Medical Informatics Association
IMP	Interface Message Processor
ISAAC	Integrated System Architecture in Advanced Primary Care
ISDN	Integrated Services Distributed Network
ISO	International Standards Organization
LLC	Logical Link Control
MAC	Message Authenticity Code
MEDIX	Medical Data Interchange
MEMOCOM	Electronic mail system supplied by the Dutch public telephone system

MIRAGE	Dutch EDI project conducted by DIV and BAZIS in co-operation with 3I
MRI	Magnetic Resonance Imaging
MSP	Medical Services Plan
MTA	CCITT X.400 Message Transfer Agent
NHS	U.K. National Health Service
NHSCR	NHS Central Registry
NSF	U.S. National Science Foundation
NUI	Network User Identification
ODA	Office Document Architecture and interchange format
OS/2	Operating System developed by IBM Corporation
OSI	Open System Interconnect developed by ISO
PBX	Private Branch Exchange
PC	Personal Computer
PDU	Protocol Data Unit
PHONENET	Electronic mail and news network based on telephone dial-up
POSINET	Prototype OSI Network developed by COSIT
PPA	U.K. Prescription Pricing Authority
PROM	Programmable Read-Only Memory
PSN	Packet Switched Network
PSTN	Public Switch Telephone Network
RAM	Random Access Memory
ROM	Read-Only Memory
RSA	Rivest Shamir Adleman public key crypto-system
SAP	Service Access Point
SDU	Service Data Unit
SHINE	Strategic Health Informatics Network for Europe
SOH	Start Of Header character (ASCII 1)
TCP/IP	Transmission Control Protocol/Internet Protocol
TELEMED	German communications and workstation project
TLV	Tag, Length and Value
TSR	Terminate-Stay-Resident

UA	CCITT X.400 User Agent
UDA	Universal Datapac Access
UNIX	Operating system developed by Bell Laboratories
UVAMC	University of Virginia Medical Centre
VAN	Value Added Network
VT	Virtual Terminal
WATS	Wide Area Telephone Service
WEDI	Workgroup for Electronic Data Interchange
WHIN	Wisconsin Health Information Network
Windows NT	Operating system developed by Microsoft Corporation