Implementation, Evaluation, and Application of Distributed Heuristics for Optimizing SLA Admission

By

Steven John Roy Shelford
B.Sc., University of Victoria, 2001

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

MASTER OF SCIENCE

in the Department of Computer Science

We accept this thesis as conforming
to the required standard

[redacted]

Dr. Eric G. Manning, Supervisor (Department of CSC and ECE)

[redacted]

Dr. G.C. Shoja, Departmental Member (Department of CSC)

[redacted]

Dr. K. Wu, Departmental Member (Department of CSC)

[redacted]

Dr. F. Gebali, External Examiner (Department of ECE)

Supervisor: Dr. Eric G. Manning

ABSTRACT

Reservation of bandwidth is necessary to absolutely guarantee the quality of service (QoS) of multimedia sessions in interconnected networks. We explore the use of the Multidimensional Multiple-choice Multi-knapsack (MMMKP) problem to solve the problem of optimal admission and adaptation control of service level agreements (SLAs) with multiple levels of QoS. The goal of admission and adaptation is to maximize a function of system utility, subject to constraints on resource allocation (link bandwidth) imposed by the need to respect all QoS guarantees. We first propose P-HEU, a new distributed heuristic designed with a parallel algorithm approach. Next, to evaluate several heuristics, we outline the implementation of a simulated distributed optimal SLA admission and adaptation controller, dSLAOpt. Upon evaluating the heuristics, we propose a modification of a previous heuristic, MFA-HEU, which is fast, and yields high system utility. The thesis concludes with the introduction of a unified admission model for the reservation of both network and multimedia server resources for multimedia sessions.

Examiners:

[redacted]

Dr. Eric G. Manning, Supervisor (Department of CSC and ECE)

[redacted]

Dr. G.C. Shoja, Departmental Member (Department of CSC)

[redacted]

Dr. K. Wu, Departmental Member (Department of CSC)

[redacted]

Dr. F. Gebali, External Examiner (Department of ECE)

Table of Contents

List of Tables

List of Figures

Glossary of Terms

A-HEU	Arbitrated Heuristic for solving MMMKPs
CBR	Constant Bit Rate
D-HEU	Distributed Heuristic, the first heuristic for solving MMMKPs
DSC	Distributed SLA Controller
dSLAOpt	Distributed SLA Optimizer: a simulated distributed SLA controller used for testing heuristics
FA-HEU	Fast Arbitrated Heuristic, a modified version of A-HEU
G-HEU	Greedy Heuristic for solving MMMKPs
KP	Knapsack Problem
I-HEU	Iterative Heuristic for solving MMKPs
ISP	Internet Service Provider
MA-HEU	Modified Arbitrated Heuristic, a modified version of A-HEU
MFA-HEU	Modified Fast Arbitrated Heuristic, a modified version of FA-HEU
MKP	Multidimensional Knapsack Problem
MMKP	Multidimensional Multiple-Choice Knapsack Problem
MMMKP	Multidimensional Multiple-Choice Multi-knapsack Problem
P-HEU	Parallel Heuristic for solving MMMKPs
QoS	Quality of Service
SLA	Service Level Agreement
VoD	Video on Demand

Acknowledgements

I would like to thank Dr. Eric Manning for his support and direction over the last 2 years. I am especially appreciative of his understanding during a few extremely difficult periods.

Additionally, I would like to acknowledge the other members of my thesis committee: Dr. Ali Shoja, Dr. Kui Wu, and Dr. Fayez Gebali. Their insights have proved valuable in the completion of this thesis.

Finally, I would like to thank the BC Advanced Systems Institute (ASI) for awarding me an ASI Scholarship, and NSERC, Nortel, and the New Media Innovation Centre (NewMIC), for their joint efforts in awarding me an NSERC Industrial Postgraduate Scholarship.

Dedication

This thesis is dedicated to my little brother, Conley Shelford. You were, and will continue to be, an inspiration to the many whose lives you touched. Thank you.

1 Introduction

The use of the Internet today is not what its designers intended. Not too long ago, the Internet was almost solely used for file transfer, e-mail, and basic web traffic. Now with the advent of broadband at work, and in the home, people are realizing they want, or need, to do much more on-line.

In view of the past uses of the Internet, it is understandable why the Internet is a best-effort system, where all users receive equally poor service. There are no guarantees of bandwidth, latency, or jitter between two points of a network. A router will not discriminate between two connections, giving no special privileges to a particular connection (and thus a particular user) over another. By not limiting users' access to the network, certain network links may become congested, resulting in poor performance observed by the users who are sending or receiving data over those links.

The Internet has been created as a connectionless, packet-driven network, where the packets of a single session may not all travel over the same route. With the type of traffic, such as video, being carried over today's networks, it is being realized that a connection-oriented network would fare better, at least for traffic requiring certain guarantees.

It is conceivable to provide guaranteed quality of service to sessions across the Internet, as explored by the IETF [1]. The guarantee of quality is subject to limiting the number of users allowed to use the network, so that the load on the network is controlled. The aspect of determining which users can use the network is known as *admission control.*

1.1 Motivation

Much money has been spent expanding the backbones of the Internet, but only a small portion of the traffic that is routed across them actually generates revenue. Peering relationships between content and network providers are a key factor for broadband to

succeed [2]. Content and network providers need to work together to form an environment to pay for the investment necessary to install and maintain the infrastructure.

Network providers will make optimal use of their infrastructure and content providers need to offer new revenue generating services; the most obvious include multimedia-broadband applications such as video-on-demand, video-conferencing, and interactive video. These applications, however, require quality of service (QoS) guarantees to be delivered in order that people will be willing to pay for the services [3] [4]. As previously mentioned, the Internet, and most networks in general, are not yet capable of serving these applications fully.

The use of classes of services, such as in Differentiated Services [5], only heightens the likelihood that the quality of a delivered stream will be acceptable, and does not guarantee it. Actually, the likelihood of delivering the streams with acceptable QoS will drop as more and more traffic is delivered over the networks.

It has been suggested that a high performance optical overlay network be used to route traffic requiring QoS guarantees. People would be required to pay to use the high performance optical overlay network, and could use the regular networks as a cheaper alternative for traffic with less stringent, or with no, QoS requirements. The path of session initiation can be seen in Figure 1.1. This simple approach to guarantee network performance to applications requires that we apply admission control to limit the number of users on the network. By carefully admitting sessions onto the network, congestion can be eliminated.

The use of a high performance optical overlay network is possible as a national backbone, for individual enterprise networks, or for a network service provider. The network may actually consist of a small set of interconnected networks, split up for administrative or legal purposes.

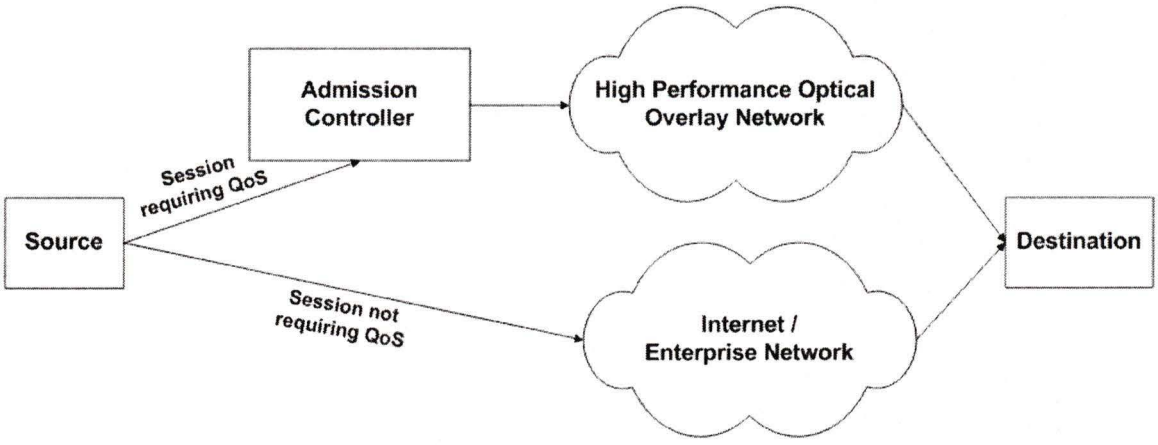

Figure 1.1: Session initiation of multimedia and non-multimedia traffic.

1.2 Problem Definition

The admission controller for a connection-oriented, high performance overlay network must admit sessions to best utilize the networks' resources. Sessions will be admitted on a particular path such that its required end-to-end delay bound will be satisfied, while bandwidth will be reserved along the path so that the capacity requirement of the session will also be met. Since we assumed a small set of interconnected networks, we also assume a small set of interconnected admission controllers: one for each network. The admission controllers must then communicate to admit and reject sessions on the networks, especially since a session may have end-points in different networks, thus utilizing resources in multiple networks.

The interesting problem is not simply admission control, but optimal admission control, where the controllers must coordinate to maximize some function, which we call system utility.

The networks in question must allow for the routing of traffic through explicit paths, either through the explicit formation of connections, or through label-switched paths, such as in MPLS [6]. The network may support resource reservation, such as through RSVP [7], or should at least enforce strict traffic policing.

The use of optimal admission, as described in this thesis, is applicable to either core or edge networks, but not necessarily access networks. Within a backbone, a session can be viewed as an aggregate of flows between two edge networks. Within regional or local ISP networks, a session can either be an aggregate of users' session, or individual user's session. The latter case is the ultimate goal of this research, dependent on the speed of solutions, allowing individual users to place a value on their session, and having admission controllers determine optimally if their session is to be served. The ability to guarantee QoS within these edge networks, and by having content deployed at the edge, allows users to finally utilize, and value, QoS dependent applications.

1.3 Scope and Focus

We limit our scope almost entirely to the work of Akbar [8]. This thesis focuses on the implementation, evaluation, application, and extension of the optimal admission controller as described by Akbar.

1.4 Outline

Chapter 1 outlines the need for quality of service and the need for optimizing traffic over an internet. Chapter 2 provides the background material required for understanding the later chapters in this thesis. Chapter 3 describes a distributed optimal admission and adaptation controller. Chapter 4 introduces a parallel heuristic for solving the optimal admission problem. Chapter 5 describes dSLAOpt, a simulated distributed optimal admission controller, which was used for demonstrating and evaluating the various heuristics for solving the optimal admission problem. Chapter 6 presents the evaluation results for various heuristics. Chapter 7 proposes the Unified Admission Model for an Internet Service Provider (ISP) to optimally provide video-on-demand services. Finally, Chapter 8 summarizes my contributions, and includes recommendations for future work.

2 Background

This chapter provides many of the concepts required for understanding the remainder of this thesis. Fundamental concepts are explained and previous research leading up to, and related to, this thesis is outlined.

2.1 Knapsack Problems

Knapsack Problems (KPs) are combinatorial problems dealing with optimization under various constraints, as will be seen in the following subsections.

2.1.1 Classical 0-1 Knapsack Problem

In the classical, or simple, 0-1 knapsack problem, there exists a conceptual knapsack and n items. Each item has a value, v_i, and a specified weight, w_i, where $0 \leq i \leq n-1$. The objective is to select items to be put into the knapsack in order to maximize the value of the items within the knapsack, V, while respecting the weight constraint of the knapsack, W. Figure 2.1 shows an example knapsack problem with 4 items of varying size.

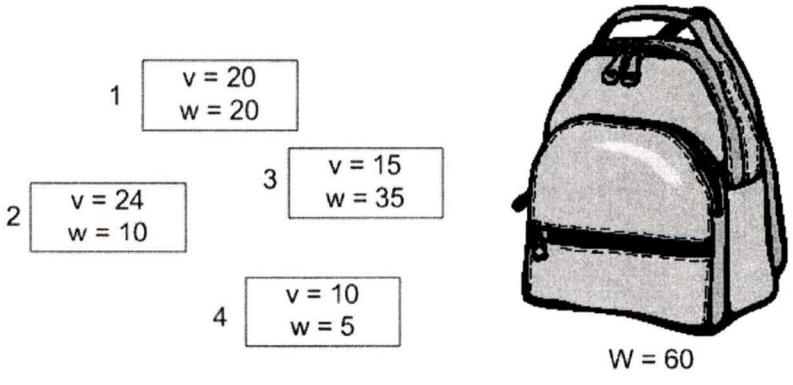

Figure 2.1: 0-1 Knapsack problem with 4 items.

Let x_i be a binary-valued variable representing the decision of whether or not an item should be placed within the knapsack (0 indicates the item is not added, while 1 indicates the item is added), then the problem can be stated concisely as

$$V = MAXIMIZE(\sum_{i=0}^{n-1} v_i x_i), \text{ such that } \sum_{i=0}^{n-1} w_i x_i \leq W, \text{ where } x_i \in \{0,1\}. \quad (2.1)$$

2.1.2 Multidimensional Knapsack Problem

The 0-1 Multidimensional Knapsack Problem (MKP) is similar to the classical 0-1 knapsack problem, except that there exists multiple resource constraints, instead of a single resource constraint W. The 0-1 classical KP had the sole constraint of weight, whereas now a knapsack can have numerous resource constraints, where items can consume some of each resource. Figure 2.2 shows an example MKP with two resource constraints: C1 and C2.

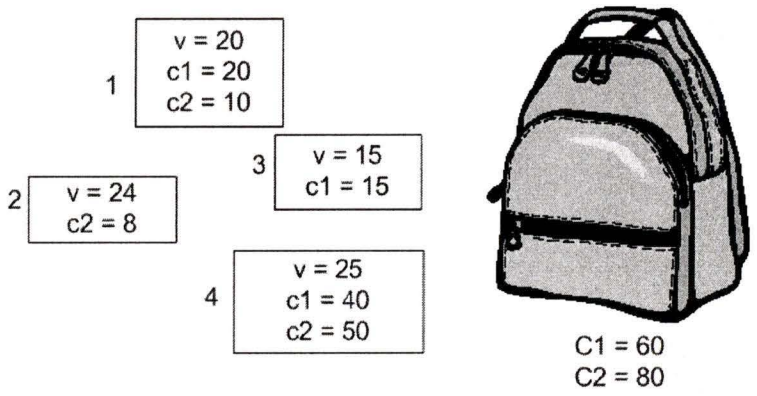

Figure 2.2: Multidimensional knapsack problem with 4 items and 2 resources.

Let n be the number of items, c_{ir} be the amount of resource r item i will consume, and C_r be the total amount of resource r that the knapsack contains, then

$$V = MAXIMIZE(\sum_{i=0}^{n-1} x_i v_i), \text{ such that } \forall r \sum_{i=0}^{n-1} x_i c_{ir} \leq C_r \text{ where } x_i \in \{0,1\}. \quad (2.2)$$

2.1.3 Multidimensional Multiple-Choice Knapsack Problem

The Multidimensional Multiple-Choice Knapsack Problem (MMKP) is an extension of the MKP where items are now clustered into groups, of which one item, and only one item, from each group must be added to the knapsack. For example, Figure 2.3 shows 3

groups of items of which an item in each group has been selected to be placed into the knapsack.

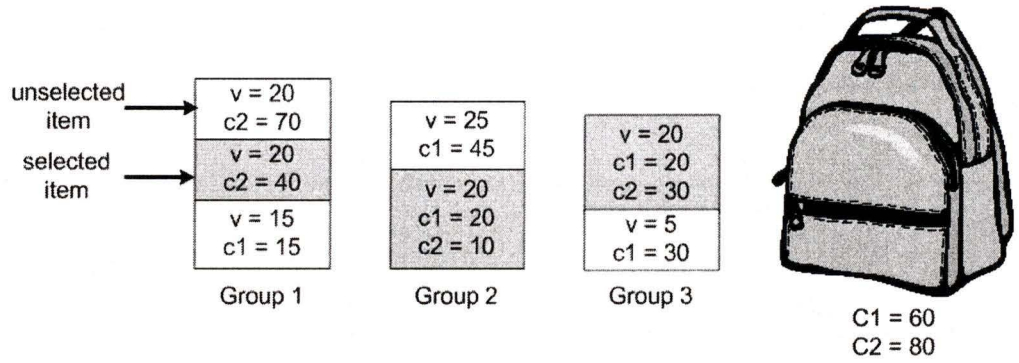

Figure 2.3: MMKP with 3 groups of items.

Let there be n groups and l_i items in group i, where $0 \le i \le n-1$. Now let x_{ij} be the decision of whether or not item j of group i was picked. The MMKP can now be expressed as follows:

$$V = MAXIMIZE(\sum_{i=0}^{n-1}\sum_{j=0}^{l_i-1} x_{ij}v_{ij}), \text{ such that } \forall r \sum_{i=0}^{n-1}\sum_{j=0}^{l_i-1} x_{ij}c_{ijr} \le C_r \text{ where } x_{ij} \in \{0,1\} \text{ and}$$

$$\forall i : \sum_{j=0}^{l_i-1} x_{ij} = 1. \qquad (2.3)$$

2.1.4 Multidimensional Multiple-Choice Multi-Knapsack Problem

The Multidimensional Multiple-Choice Multi-knapsack Problem (MMMKP), defined by Akbar in [8], expands on the concepts of the MMKP by generalizing from 1 to m knapsacks. Each knapsack has resource constraints, and the problem is to add items to the knapsacks so as to maximize the value of the selected items summed over all of the knapsacks, while respecting all resource constraints imposed on the knapsacks. Figure 2.4 is an example of an MMMKP with 3 groups of items and 2 knapsacks.

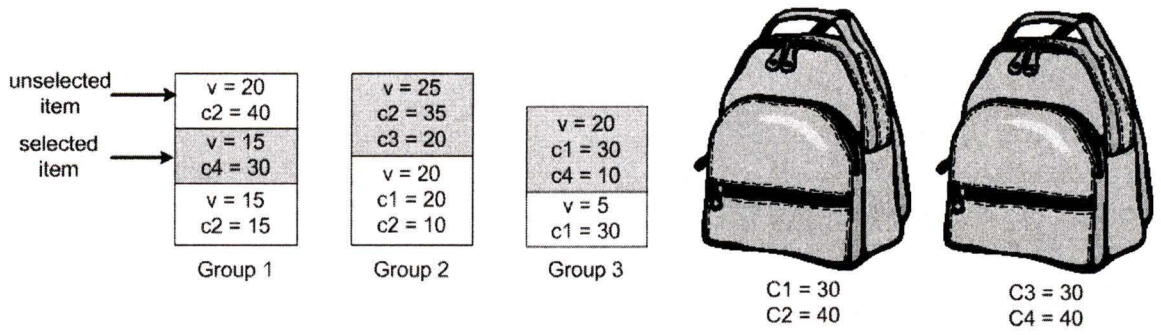

Figure 2.4: MMMKP with 3 groups of items and 2 knapsacks.

The MMMKP fails to meet the usual analogy of packing items into knapsacks, in that an item can be placed into more than one knapsack, in order to consume resources of those knapsacks. Due to this unintuitive property of the MMMKP, it can perhaps better be thought of as a distributed system, where the knapsacks, and thus the resources, are distributed. Mathematically, the MMMKP is similar to the MMKP:

$V = MAXIMIZE(\sum_{i=0}^{n-1}\sum_{j=0}^{l_i-1} x_{ij}v_{ij})$, where the groups are distributed amongst m servers, such that $\forall r \sum_{i=0}^{n-1}\sum_{j=0}^{l_i-1} x_{ij}c_{ijr} \le C_k$, where $x_{ij} \in \{0,1\}$ and $\forall i : \sum_{j=0}^{l_i-1} x_{ij} = 1$. (2.4)

2.2 Optimal SLA Admission and Adaptation

The work in this thesis deals with optimal admission and quality adaptation of SLAs in interconnected networks. We now define Service Level Agreements (SLAs), Optimal Admission Control, Quality Adaptation, and finally, how Optimal Admission and Quality Adaptation are used within this thesis.

2.2.1 Service Level Agreement

In this thesis, a Service Level Agreement (SLA) is a contract between a user and a service provider guaranteeing one of a set of specified qualities of service (QoS) between two points. Hence it can be characterized by the following tuple:

(source, destination, { QoS level})

The *source* and *destination* refer to the two endpoints of the requested connection, and the *QoS levels* specify the possible levels of service the user is willing to accept. Each QoS level is in turn specified by the following tuple:

(bid, data rate, delay, duration).

The *bid* is the price offered by the user for the QoS level per unit of time. The *data rate* is the minimum amount of bandwidth and the *delay* is the maximum end-to-end delay required to satisfy the defined QoS level. *Duration* is the length of time for which the SLA should be honoured if admitted.

The end-to-end delays within the network are considered to be static values, not depending on the traffic load in the network. Although this is not entirely realistic, our admission schemes do avoid many delay inducing problems such as congestion, thus making static delays more plausible. Our schemes could be extended to incorporate sources of time-variable delay, notably queuing delays, although this is not considered in this thesis.

2.2.2 Optimal Admission Control

Admission control is a requirement for the guaranteed QoS of multimedia sessions. Without admission control, a network can become congested by the bandwidth demands of too many sessions, thus making QoS guarantees impossible. Optimal admission control is the admission of SLAs to a network such that the QoS of all admitted SLAs is guaranteed, while optimizing a given utility function. Various entities, or combinations thereof, could be optimized, such as the number of SLAs admitted, the use of network resources, or even network revenue, which is the focus of this thesis.

2.2.3 Quality Adaptation

The concept of quality adaptation is that the quality delivered to a session can be improved as network resources are freed, such as when sessions' SLA contracts expire. For example, imagine a user who prefers viewing a streaming DVD quality movie, but the network only has resources to allow the movie to be streamed in low resolution, with no colour. However, after 5 minutes of viewing the movie, the expiry of many SLAs has freed enough network resources to allow the user to view the movie at DVD quality. More resources should be assigned to this session to improve the QoS for the viewer – referred to as adapting the SLA – if this decision, of the choices available, serves to best optimize the utility function.

Levels of QoS are often ranked in terms of bid prices. If quality adaptation occurs on an SLA, changing its level of QoS to that of a higher bid price, then we refer to that SLA as being *upgraded.* If quality adaptation occurs on an SLA, changing its level of QoS to that of a lower bid price, then we refer to that SLA as being *downgraded.*

2.2.4 Optimal SLA Admission and Adaptation

Optimal SLA admission and adaptation refers to the process of admission and adaptation control of SLAs while maximizing some function affected by the admission process. Adaptation refers not only to quality adaptation, but path adaptation of the SLAs as well. By changing the path an SLA is served on, other SLAs can be admitted, or have their quality adapted.

In this thesis, a function of system utility is maximized. With the inclusion of bid prices in SLAs, as previously described, we can define system utility as the gross or net revenue received by servicing user requests. Therefore, optimal SLA admission and quality adaptation can refer to admission control that respects all QoS constraints imposed by admitted session, while maximizing the system's revenue. This is achieved through the concepts of the Utility Model.

2.3 Utility Model

The Utility Model, designed by Khan, maximizes the efficiency of a multimedia server to stream data to paying clients [9]. The model demonstrates how to perform optimal admission control to maximize revenue for a server by selectively accepting or rejecting sessions depending on the resources each requires and the amount of money, known as the bid price, a user is willing to pay to receive their service. The Utility Model maps the server optimization problem into an MMKP. What is more interesting to us, however, is Watson's application of the Utility Model to enterprise data networks to solve the Optimal SLA Admission and Quality Adaptation problem [10].

The problems of Optimal SLA Admission and Quality Adaptation may seem to be separate problems, but with the Utility Model and the Utility Model Distributed, described in Section 3.1, they are merged into a single problem, and thus can be solved simultaneously. An admission controller considers the admission of all proposed new sessions as well as the upgrading, or downgrading, of all existing sessions to optimize utility. Section 3.4 explains how admission is actually modeled as an upgrade from a logical *null* level of QoS.

2.3.1 Utility Model Applied to Data Networks

Watson & Manning recognized [10] [11] that the Utility Model could be applied to more than just servers. The admission and adaptation of SLAs through a path-switched data network can be optimized by the application of the Utility Model as well. The resources of a data network include all of the links of that network, and the resource constraints consist of the bandwidths of those links. We must not allocate more bandwidth on a link than it possesses. Additionally, the end-to-end delay of the QoS levels must be satisfied.

Khan assumed each level of QoS to be uniquely mapped to the required resources. However, with data networks, each link is a resource, and therefore, since an SLA can be routed between two nodes by more than one path in any practical network, it follows that

a single level of QoS for a session can be mapped to multiple resource configurations. Hence we need a group comprising of interspersed items from different paths, as shown in Figure 2.5.

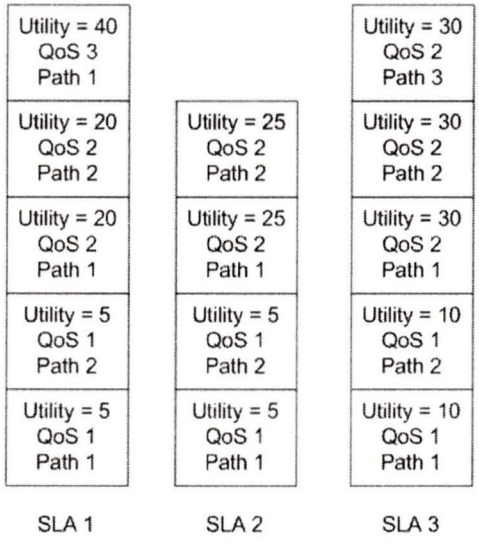

Figure 2.5: Representation of SLAs as stacks of items.

With an admission controller, an SLA is described in terms of its *QoS profile*. An SLA's QoS profile describes the QoS levels of an SLA, as well as the different resource configurations to satisfy each QoS level. The QoS profile for an SLA can be described by a set of *sub-QoS levels*, described by the following tuple:

(QoS level, bid, data rate, duration, resources).

The *bid*, *data rate*, and *duration* are the parameters that were described in the definition of the SLA. The resources are a list of links (resources) that satisfy the defined QoS level. Recall that a QoS level can be satisfied by numerous resource configurations, and thus numerous sub-QOS level entries may exist in the QoS profile for each level of QoS.

An SLA represents a group in an MMKP, where each item is a sub-QoS level for that SLA. Each sub-QoS level has an associated bid price, a required set of resources, and a bandwidth requirement. The goal is to select one sub-QoS level for each SLA such that

the overall value, or revenue in this case, of the sub-QoS levels is maximized without assigning more bandwidth on each link than is available.

To solve an MMKP, Khan assumed that each item in a group (where a group represents a session) could be listed in a fixed order, stacked in order of increasing utility. Due to the multiple resources configuration, the items for a data network cannot be fully ordered in terms of increasing utility alone.

Khan also assumed that if a QoS level could be supported, that any QoS level beneath it, would also be feasible. Using Khan's assumptions, in multimedia servers, an SLA could be upgraded or downgraded by a single QoS level at a time. This is not the case in terms of data networks. Heuristics for data networks should be able to upgrade from sub-QoS n-2 to sub-QoS n if sub-QoS n-1 is not feasible. However, it does imply extra processing, because if sub-QoS n-1 is not currently feasible, we cannot terminate the search for feasible upgrades at sub-QoS n-1. Rather, we must search the entire stack of items.

2.3.2 MMKP Solution Heuristics

The MMKP can be solved optimally, by the popular branch and bound algorithm, or by a near-optimal heuristic. Heuristics have the advantage of fast processing, which is necessary in a real-time system, especially for real-time admission, and perhaps more importantly, real-time quality adaptation. Here we focus on I-HEU, as it is the foundation for many of the MMMKP heuristics described throughout this thesis.

2.3.2.1 I-HEU

I-HEU (Iterative Heuristic), based on Khan's MMKP heuristic [9], was devised by Akbar to solve the MMKP in a non-distributed manner [8]. I-HEU upgrades (and admits) SLAs by trying to maximize the utility (revenue) of the network per unit of aggregate resources required, where aggregate resource is a function of the links required and their current utilization. The uniqueness of I-HEU is that it considers that the MMKP in our

application will be solved repeatedly over a period of time. Over time, some SLAs (groups of items) will be removed, and others will be added to the problem space.

There are 3 steps to I-HEU:

1. Find a feasible solution. Use the solution of the previous time step to determine the selected QoS level for previously existing sessions, and for new sessions, select a logical *null-QoS level.* The null-QoS level is a QoS level that consumes no resources and costs no money, which, if still selected upon completion of the heuristic, means that session has been rejected, as the session is not assigned any resources.
2. Upgrade the session with the highest value added per required resources that is feasible. Loop step 2 until no more value can be added while maintaining all resource constraints. The idea is to upgrade sessions, picking those that provide the most value per unit resource, until no more resources are available.
3. Upgrade the session with the highest value added per required resources (even if it is infeasible, violating one or more resource constraints), then downgrade sessions with low value per required resources until the session is feasible. Revive the previous solution and end the heuristic if the overall value is not increased, otherwise repeat step 3. The idea here is to avoid local maxima, by finding search spaces that offer high utility.

2.3.3 Resource Contention

The Utility Model, as defined by Khan, assures that enough resources will be reserved for the QoS level at which the session is admitted. This promise assumes the non-existence of hardware failure, network reconfiguration, or loss of resources to applications not controlled by the Utility Model.

To deal with such contention, the handling is rather arbitrary and left up to the specific implementation. Ideally, a user's preferences are held (i.e. only downgrading them to their lowest acceptable QoS); however, the handling can be as drastic as terminating an SLA.

3 Distributed Optimal SLA Admission Controller

3.1 Utility Model Distributed Applied to Interconnected Networks

Akbar [8] proposed a distributed version of the Utility Model, the Utility Model Distributed. The Utility Model Distributed is applicable to both interconnected switched data networks and servers.

For a large switched network, optimal admission and quality adaptation is difficult with a single SLA controller. In addition, when applying optimal admission and quality adaptation to SLAs that travel through more than one data network, a distributed algorithm is convenient for coordinating among the SLA controllers of the networks. Figure 3.1 shows the interconnection of 3 networks, each with its own SLA controller.

To solve the problem of controlling a large data network, it is simply partitioned into several sections, each with its own controller. The large data network is therefore treated similarly to the case of multiple small data networks, also each with its own SLA controller. The Utility Model Distributed can then be used to admit SLAs into the network.

For interconnected data networks, the Utility Model Distributed maps the Optimal SLA Admission and Adaptation Problem into the Multidimensional Multiple-choice Multi-Knapsack Problem (MMMKP). Each of the interconnected networks is considered to be a knapsack (multi-knapsack) with the links within each network being the resources of that knapsack. An SLA has multiple QoS levels (or, more specifically, multiple sub-QoS levels), each with an associated bid price, of which one must be chosen (multiple-choice), and each of which requires one or more possible paths – a list of links (multidimensional). The goal is to admit SLAs into the networks such that the overall

value, or revenue in this case, is maximized without assigning more bandwidth on each link than is available.

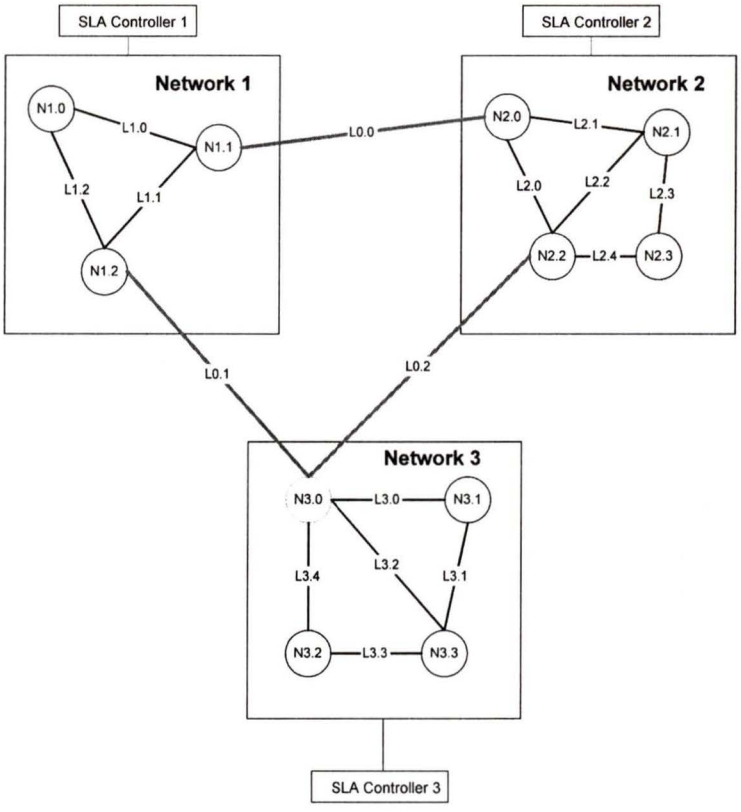

Figure 3.1: Example network with naming scheme.

To fully understand admission, we introduce the concept of the *null* QoS level [10]. The null QoS level is a logical QoS level that consumes no resources and costs no money. Sessions wishing to be admitted have their QoS level initially set to a null QoS level. If, upon completion of the heuristic, the null QoS level is still the chosen QoS level for the session, then that session has been rejected, as the session was not assigned any resources. With the null QoS level, admission can simply be viewed as upgrading an SLA from the null QoS level. Once admitted, the null QoS is removed from the SLA, as an SLA cannot be rejected once admitted. Figure 3.2 shows the representation of SLAs in terms of an MMMKP. Note that SLAs 1 and 2 are awaiting admission, as they still include the null QoS.

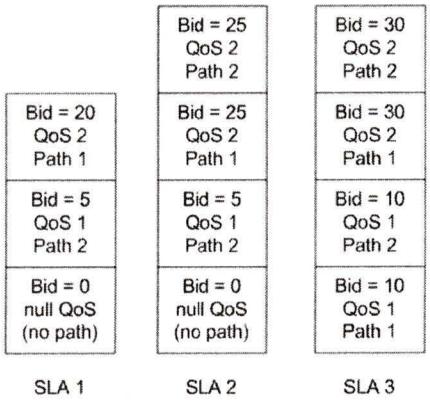

Figure 3.2: SLAs as groups of knapsack items.

3.2 Naming Conventions

Each network is given an integer ID ≥ 0, which we call its network ID. Within a network, each link, node, and path is given a unique ID ≥ 0, called the *link ID*, *node ID*, and *path ID*, respectively. Links, nodes, and paths are named in the format:

- L<*networkID*>.<*linkID*>,
- N<*networkID*>.<*nodeID*>, and
- P<*networkID*>.<*pathID*>.

Example names include: L1.1, N1.2, L0.5, or P3.2. Figure 3.1 shows a sample group of networks using Distributed SLA Controllers (DSCs). Note that we have not defined where in each network a DSC will reside, and thus DSCs are shown as attached to the networks rather than to specific nodes of the networks.

In defining the controllers, it is helpful to introduce the notion of a *global network*. The global network is a network consisting of the individual networks as nodes, with the links between the networks (external links) as the links of the global network. Figure 3.3 shows how the networks of Figure 3.3 map to the global network, which is given the network ID of 0. The networks - the nodes of the global network - are labelled by N0.<*networkID*>, such as N0.1 for Network 1 or N0.2 for Network 2.

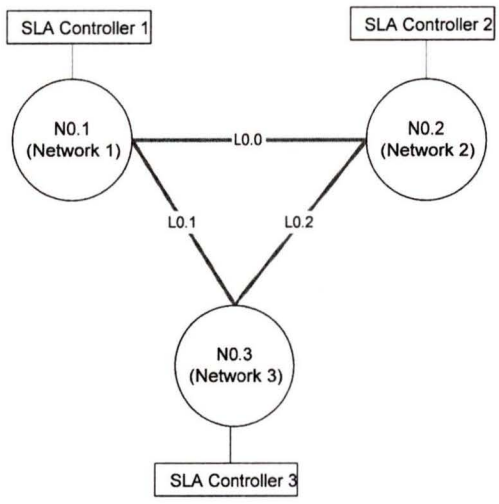

Figure 3.3: Global network view.

A DSC knows only the topology of its local network and of the global network (network 0), including the end nodes of the external links – the gateways.

3.3 Hierarchical k Candidate Paths

The k Candidate Path Heuristic developed by Akbar [8] is designed to quickly calculate the approximate *k* shortest paths for inter-network SLAs. The central idea of the heuristic is to apply Eppstein's *k* shortest paths algorithm [12] hierarchically in two levels: first at the global network level, then at the local network level. The results are then merged to approximate the overall *k* shortest paths. The concept of finding complete paths from localized path segments is also used in OSPF inter-area and inter-autonomous system routing [13].

Since we are not claiming the ability to find paths over an entire internet, we do not restrict ourselves to the Border Gateway Protocol, or distance vector routing in general, for inter-network routing.

3.3.1 Finding Paths in the Global Network

The first step in the *k* candidate path heuristic is to determine which networks a session may be routed through by applying Eppstein's algorithm over the global network. Paths

within the global network are referred to as *global paths*. Figure 3.3 shows the global network corresponding to the 3 interconnected networks, shown in Figure 3.1, as viewed by the *k* candidate path heuristic.

If all external links have a delay of 200ms, then the 2 shortest global paths from N2.1 to N1.2 would be calculated by applying Eppstein's algorithm on the global network, where the source is N0.2 (network 2) and the destination is N0.1 (network 1). Therefore, the 2 shortest global paths for N2.1 to N1.2 are: {L0.0 (200ms)}, and {L0.2 (200ms), L0.1 (200ms)}.

We now introduce an alternative view of the global network that includes the representation of the gateways. Figure 3.4 presents a view of the global network where the gateways, instead of the networks, are the nodes of the network. The dotted lines in the figure represent *dummy links*, the logical paths between two gateways of a network, and, for now, are considered to have a delay of 0.

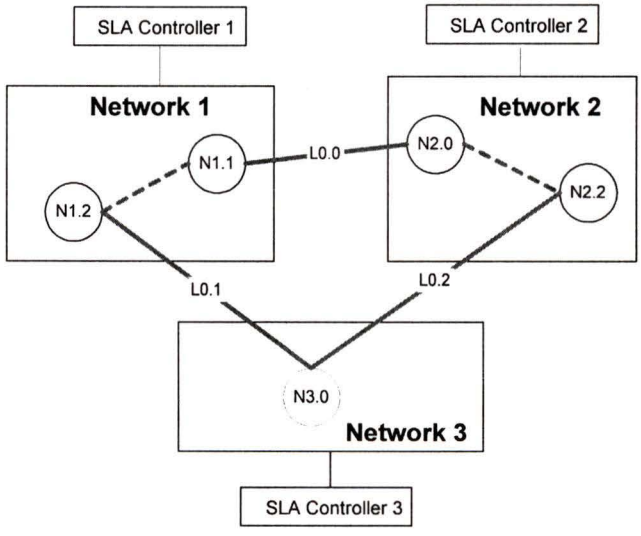

Figure 3.4: Global network represented by gateways as viewed by the k candidate paths heuristic.

The global paths can now be expanded to include dummy links, and use the syntax of the local networks, rather than the global network. For readability, the paths here will be listed in terms of each link's end nodes, and include the dummy links. The global paths from N2.1 to N1.2 can be described as:

{N2.1→N2.0 (0ms), N2.0→N1.1 (200ms), N1.1→N1.2 (0ms)}, and

{N2.1→N2.2 (0ms), N2.2→3.0 (200ms), N3.0→N1.2 (200ms)}.

Notice that a dummy link was added between the source, N2.1, and each of the gateways of the network 2 (N2.0 and N2.2). If the destination node was not a gateway, a similar procedure would have had to be done between the gateway N1.2 in network 1 and the destination node.

3.3.2 Finding Paths Inside the Networks

In this step we descend a level, applying Eppstein's algorithm to find the actual paths for the dummy links, which we call *sub-paths*. For a particular SLA, sub-paths in different networks are calculated by their corresponding DSCs. Each DSC sends path request messages to the other DSCs, requesting they perform the sub-path calculation. Those DSCs then perform Eppstein's k shortest path algorithm between the two end points of the sub-path, and return the results through *path messages* to the requesting DSCs. The path message includes the delay of the path and a path number, not specific links (as the links are known only to the local network).

For the example in the previous section, DSC 2 of network 2 would request path information for N1.1→N1.2 from network 1 and would itself calculate the k shortest paths between N2.1→N2.0 and N2.1→N2.2.

3.3.3 Calculating k Candidate Paths in Interconnected Networks

The paths over the global network, and the sub-path information returned from the various DSCs, are merged to determine the k candidate paths from the source to the destination. For each dummy link, each of the k sub-paths returned can be substituted for the corresponding dummy link in the global path. k^{m+1} combinations are thus possible if all combinations are considered, where m is the number of networks.

Of the k candidate paths calculated, a path will only be considered as a candidate for a particular QoS level of an SLA if its delay is less than the delay constraint of that QoS level.

3.4 Admission and Adaptation Heuristics

Optimal Admission and Adaptation Heuristics are simply heuristics for the MMMKP. Akbar [8] designed several heuristics to solve the MMMKP, three of which we will describe here. Recall that admission of SLAs is logically just an upgrade from a null QoS level, as admission is often not discussed explicitly when describing these heuristics (see Section 2.3.2.1).

3.4.1 D-HEU

The distributed iterative heuristic (D-HEU) is the result of straightforwardly distributing the centralized iterative heuristic (I-HEU). The message passing complexity of D-HEU is $O(mnl)$ where m is the number of DSCs, n is the number of SLAs currently admitted and being admitted, and l is the number of QoS levels per SLA multiplied by k, the number of paths to consider for each QoS level. The time requirements of this heuristic are typically infeasible for most applications, as will be shown in Chapter 6 of this thesis.

Essentially, when an SLA is added to the system, it is submitted to the DSC at each network where the SLA can consume resources. A DSC will pick a single local candidate for upgrading at a time, followed by an all-to-all broadcast of that local candidate between DSCs in order to determine the best overall candidate. This procedure is repeated until no further upgrades can be accomplished. It is easy to see that this is a costly heuristic when there are a large number of SLAs in the system.

3.4.2 G-HEU

One of the simplest heuristics is a greedy heuristic, named G-HEU to keep with previous naming conventions. As the greedy heuristic has yet to be fully described in previous

works, I have included it here[1]. G-HEU upgrades (or admits) SLAs, or more precisely, specific QoS levels of SLAs, starting with the QoS level offering the highest change in utility first. G-HEU will sort the QoS levels of all SLAs in decreasing order of change in utility, then attempt to upgrade the SLAs from the start of the list. A controller will upgrade SLAs in a distributed fashion at first, but upon the failure to upgrade a single SLA, the heuristic will then greedily upgrade only local intra-network SLAs, which requires no inter-controller communication.

The greedy heuristic can be broken up into five main steps:

1. calculate the best QoS level upgrades within a single network,
2. merge best upgrades lists, from each network, and sort,
3. determine the local feasibility, i.e. how many upgrades from the list can be satisfied locally,
4. determine which upgrades all DSCs can perform, and
5. greedily upgrade intra-network SLAs.

We assume there exists a number of SLAs that have already been admitted into the system, S_{old}, and a number of new SLAs to be admitted, S_{new}.

To clarify our terminology, let S be the set of all SLAs in the DSCs: $S = S_{old} \cup S_{new}$. For a given SLA $s \in S$, let K_s be the set of k candidate paths that have been calculated for the SLA from the designated source to the designated destination, where $|K_s| = k$, and let Q_s be the SLA's set of QoS levels. Now let P_s^q be the set of paths from K_s which meet the delay requirement of QoS $q \in Q$. That is,

---

[1] The greedy heuristic, G-HEU, was originally described orally by M.M. Akbar, but had not been implemented or described fully until this thesis. The greedy heuristic noted in [8] was a simpler heuristic, and is not the same as is described here.

$\forall p \in P_s^q$,

$$delay(p) \leq delayBound(q), \text{ where } p \in K_s. \quad (3.1)$$

1. Calculate the Local Best Upgrade

In this step, each controller determines the best upgrades of local SLAs, based on largest increase in revenue. This list of proposed QoS levels is then sent to all controllers.

a) Signal all DSCs to start.
b) For all $s \in S_{new}$, let the initial QoS level be the null-QoS level. The current solution is now feasible. See Figure 3.5 for an example current solution.
c) Set the proposed list of QoS levels for all SLAs to be that of the currently set QoS levels.
d) There exists an upgrade of some SLA that creates the largest possible increase in revenue. If there exists a path $p \in P_s^q$ that contains enough free capacity, assuming all previously proposed QoS changes are admitted, based on knowledge of local resources only, then propose that SLA is upgraded. If no free capacity on a path exists, the change is not feasible, so remove the QoS level from the set of possible upgrades.
e) Repeat step 4 until no increase in revenue is possible.
f) Send the proposed QoS levels to all DSCs, along with the requirements of each QoS level.

For an example of the proposed QoS list, see Figure 3.6. Figure 3.5 shows the SLAs for DSC_1 that were used to determine the list in Figure 3.6a. In the example, QoS level 2 was not proposed for SLA 2 because, for demonstration purposes, it is assumed it is not feasible based on available local resources.

SLA 1	
QoS Level, {Bandwidth, Delay}, Bid	
3, {40Mbps, 20ms}, $40	
2, {15Mbps, 40ms}, $15	
1, {10Mbps, 40ms}, $10	*

SLA 2	
QoS Level, {Bandwidth, Delay}, Bid	
2, {40Mbps, 10ms}, $30	
1, {10Mbps, 100ms}, $15	
Null Level, {0Mbps, 0ms}, $0	*

Figure 3.5: Local SLAs to DSC_1. * represents the current QoS levels. Note that SLA 2 is a new SLA.

a) **Proposed List from DSC 1**

d Bid, {Previous QoS , Proposed QoS}, {Proposed QoS Specification}, SLA
$30, {level 2, level 3}, {40Mbps, 20ms}, SLA 1
$15, {null level, level 1}, {10Mbps, 100ms}, SLA 2

b) **Proposed List from DSC 2**

d Bid, {Previous QoS , Proposed QoS}, {Proposed QoS Specification}, SLA
$25, {level 2, level 3}, {30Mbps, 30ms}, SLA 3
$18, {level 1, level 2}, {20Mbps, 40ms}, SLA 4

c) **Sorted Merged Proposed List**

d Bid, {Previous QoS, Proposed QoS}, {Proposed QoS Specification}, SLA, DSC
$30, {level 1, level 3}, {40Mbps, 20ms}, SLA 1, DSC 1
$25, {level 2, level 3}, {30Mbps, 30ms}, SLA 3, DSC 2
$18, {level 1, level 2}, {20Mbps, 40ms}, SLA 4, DSC 2
$15, {null level, level 1}, {10Mbps, 100ms}, SLA 2, DSC 1

Figure 3.6: Proposed QoS level lists.

2. Merge Local Best Upgrade Lists and Sort

Upon receiving every controller's list of proposed QoS levels (upgrades), a single list is created by merging each of the separate lists.

a) Receive list of SLAs and list of proposed QoS levels from each DSC.
b) Merge the lists of proposed QoS level changes and sort the list in decreasing order by increase in revenue. See Figure 3.6c for an example of a merged list, from the proposed lists shown in Figure 3.6a and Figure 3.6b.

3. Calculate Local Feasibility

Each controller determines their *local feasibility*, which is the number of upgrades they can perform from the start of the proposed QoS list. Their *local feasibility* is then sent to all the controllers.

a) Calculate local_feasibility$_i$, where i is the identity of the local DSC.

local_feasibility$_i$ = 0
do until end of sorted proposed QoS list
if QoS at item local_feasibility$_i$ of the sorted proposed QoS list is feasible by the local network, then:
local_feasibility$_i$ = local_feasibility$_i$ + 1
else:
go to step b)
repeat

b) Send local_feasibility$_i$ to all DSCs. In step a), a QoS level $q_s \in Q_s$, for an SLA $s \in S$, is feasible in a network if the network contains enough resources to commit to q_s along at least one of the paths in P_s^q.

4. Determine Global Upgrades

The number of upgrades all controllers can perform, called the *global feasibility*, is determined, based on the minimum of each controller's local feasibility. Let the global feasibility be g, then all controllers upgrade the first g SLAs from the start of the proposed QoS list.

a) Set *global_feasibility = min(local_feasibility$_0$, .. local_feasibility$_{M-1}$)*, where M is the number of DSCs.

b) Perform upgrades as detailed in the first *global_feasibility* elements of the sorted list of proposed QoS level changes.

5. Determine Global Upgrades

Now greedily upgrade intra-network SLAs only, starting with the feasible upgrade that will produce the largest revenue gain. Continue upgrading until no more feasible upgrades are possible without violating any resource constraints.

3.4.3 A-HEU

A-HEU, proposed by Akbar in [8], is similar to the greedy heuristic, except that step 5 of G-HEU is not performed, and a more time intensive heuristic is used to calculate the local best upgrade of each network (step 1 of G-HEU). The local best upgrades are determined by executing I-HEU, considering the local resources only.

4 P-HEU

Many of the current approaches to solving knapsack, or bin-packing, problems revolve around parallel search techniques (such as those described in [14], [15], [16]). However, they would not be appropriate due to the time required to execute them and the distributed nature of our resources. It was decided that the MMMKP is rather unique – recall that an item may use resources from multiple knapsacks. The distribution of knapsacks, and thus the resources, results in a problem that does not appear to have been studied extensively.

This thesis presents an alternative approach to viewing the problem of optimal admission in interconnected networks that is much simpler than those reviewed, and is described below. This new heuristic, named P-HEU (parallel heuristic), approaches the problem differently than the knapsack heuristics previously studied, by considering a parallel algorithm solution [17].

4.1 Assumptions

In P-HEU, each DSC reserves $1/m$ of the bandwidth on each link, of its associated network, for each of the DSCs to use, where m is the number of networks (knapsacks). Similarly, $1/m$ of the bandwidth on external links is reserved for each DSC to use.

Each DSC then manages a number of SLAs, some of which a DSC admits by using its share of resources over all the interconnected networks. Instead of collaborating for every minor decision, the DSCs now work independently, in parallel, and then share results for the building of the final circuits. The design of P-HEU was based on the following assumptions:

1. Average link bandwidth is high relative to the average capacity of the SLAs.
2. The number of connected networks is relatively low.

3. Session traffic will flow across only one circuit.
4. Demand is uniformly distributed across controllers.

Assumption 1 is very reasonable considering the type of high performance optical network being considered. We expect a single link to be able to support the flows of numerous SLAs at any one time.

Assumption 2 is a result of our target audience: a small set of interconnected networks.

Assumption 3 exists since P-HEU did not explore the possibility of splitting a session across multiple routes in order to achieve a desired bandwidth - i.e. if 4Mbps is required but only two 2Mbps paths exist from the source to destination, then the session will not be admitted. This assumption lets us avoid certain jitter problems and is implicit in other MMMKP heuristics as well.

Assumption 4 states that we assume that the demand for resources by the sets of SLAs processed by DSCs will be uniformly distributed among networks. This limits the environment in which P-HEU can be used; however, Section 4.4 introduces methods to reduce the strictness of this assumption.

4.2 Admission

When a request for a session s is sent to a DSC, the hierarchical k candidate paths algorithm is executed (see Section 3.3). For P-HEU, the algorithm returns the paths as a list of links. From this, a QoS Profile is constructed for the SLA, which is a list of sub-QoS levels, which we represent here as:

(bid, delay, bandwidth, path),

where the *path* is an explicit list of links from the source node to the destination node of the SLA. (See Figure 4.1 for the QoS profiles of 2 SLAs).

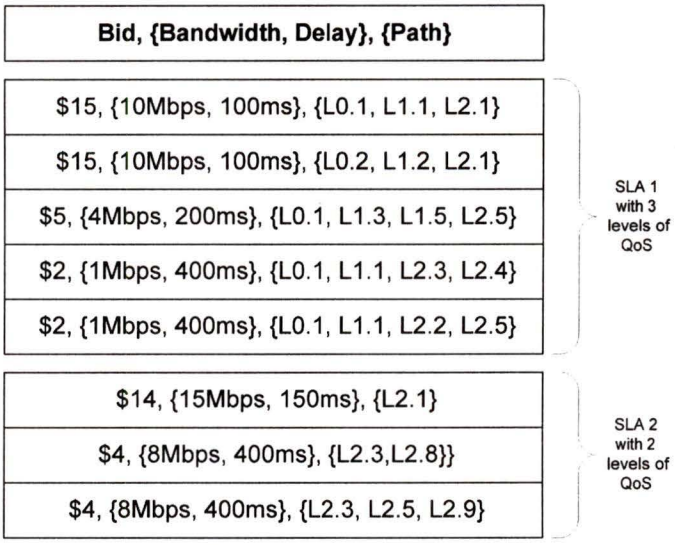

Figure 4.1: View of 2 SLAs in a distrubuted SLA controller.

This SLA can now be added to the rest of the SLAs that the DSC manages, but with a QoS level of null (see Section 3.4). I-HEU, or any MMKP heuristic, can now be run on these SLAs. Note that I-HEU is an MMKP heuristic, not an MMMKP heuristic. This is possible since each link, even if technically another network's resource, can be considered a resource of the local network since bandwidth has already been reserved for the local network on each of the links.

An item from each group must be added to the knapsack. If the null QoS level is chosen for a new SLA, then that item is rejected. If a non-null QoS level is chosen, then the null level is removed and the session is admitted.

Once I-HEU has been executed, an all-to-all broadcast occurs so that each DSC can transmit a list of admitted sessions and required paths to the other DSCs. Once all data has been received, each DSC must assure each admitted session's path is properly setup to assure proper traffic flow. Note that no bandwidth allocation and reservation is required. This is because the bandwidth is already allocated to the DSCs and each DSC is responsible for not over prescribing bandwidth on each of the links.

A disadvantage of this approach is that a DSC from one network can learn of the resources of another network. However, as mentioned earlier, the topology within other

networks is unknown. This is more for network administration purposes rather than security. As a result of a DSC managing more resources, an MMKP heuristic that scales well with respect to the number of resources is preferred.

Also, our number of assumptions may reduce optimality, as will be seen in Chapter 6.

4.3 Complexity

The message passing complexity of P-HEU is *O(m)*, where *m* is the number of interconnected networks, not considering the generation of QoS profiles. The size of the messages in P-HEU are *O(n)*. The reason for these numbers is the fact that once SLAs are admitted, then each DSC can work independently, except for an all-to-all broadcast of the selected QoS level for each local SLA and a description of the required path. This is done so all DSCs can establish the portions of circuits for the SLAs which will travel within their local network.

A-HEU [8] and G-HEU also both require *O(m)* messages, but require more messages than P-HEU. As we will see in Chapter 6, P-HEU achieves much higher utility than A-HEU or G-HEU by using less messages.

In comparison, D-HEU requires the following total number of messages to be sent [8]:

$$2n(l-1)(m-1)+2n(n-1)(l-1)(m-1),$$

where n is the number of SLAs in the system, l is the number of QoS levels for an SLA, and m is the number of networks that are interconnected.

Obviously, computational complexity is also important, as P-HEU will be controlling more resources than the other MMMKP heuristics. For this thesis we will utilize I-HEU as P-HEU's MMKP heuristic, which does not scale well with respect to the number of resources. It is a useful heuristic, however, as we can now compare P-HEU to D-HEU in terms of yielded utility, as both are based on the I-HEU heuristics.

4.4 Closing in on Optimality

Optimality can be affected by the method by which SLAs are assigned to controllers in P-HEU. SLAs can be assigned randomly, by a round-robin approach, or by using prediction algorithms to choose the best controller for an SLA.

There are two other interesting approaches to improve the accuracy (or profitability) of P-HEU:

1. Move SLAs to other DSCs to increase revenue, and
2. Move allocated bandwidth between DSCs to increase revenue.

These approaches will only be discussed briefly, and should be considered as an introduction for future work.

4.4.1 Moving SLAs

SLAs can be moved after being admitted in order to improve revenue. For example, if an SLA is currently managed by DSC_1 on $Network_1$ and is admitted at a Bronze level of QoS for $3, but at a future time DSC_2 could admit the SLA at a Gold level of QoS for $50, then the SLA could be moved from DSC_1 to DSC_2 for management at such a time. It would be interesting to discover when to move SLAs, as well as to determine the best possible DSC to which to move the SLA.

4.4.2 Moving Bandwidth

Besides moving SLAs, revenue could also be increased by moving unused bandwidth between controllers. Consider the same situation as described above. If DSC_2, as well as other DSCs, had unused bandwidth that DSC_1 could use to upgrade the SLA from a Bronze to a Gold level of QoS, then DSC_1 could upgrade the SLA and improve overall revenue. A learning algorithm could be used to balance the network towards an optimal distribution of bandwidth between the DSCs.

This may be a way to address scalability concerns, by adapting the bandwidth to general access patterns. The result would be that instead of each DSC having $1/m$ of the bandwidth of each link (which could be small for a large number of networks), but instead a larger fraction of the link capacity would be held by a smaller number of DSCs. For example, if many sessions run between Network_1 and Network_2, then certain DSCs may accumulate a disproportionate amount of bandwidth for links within those two networks.

5 dSLAOpt

The environment for which we use to demonstrate and test the heuristics for optimal SLA admission across interconnected networks is called dSLAOpt. dSLAOpt is a distributed version of SLAOpt, a simulated optimal SLA admission controller for testing heuristics on a single network [10]. This chapter will describe the setup, interface, and functionalities of the application, in order to give the reader an appreciation of the environment in which the results given in the next chapter were obtained.

5.1 Architecture

Conceptually, dSLAOpt can be described by two main components: a *k* Shortest Path module, and an admission and adaptation heuristic module, as seen in Figure 5.1.

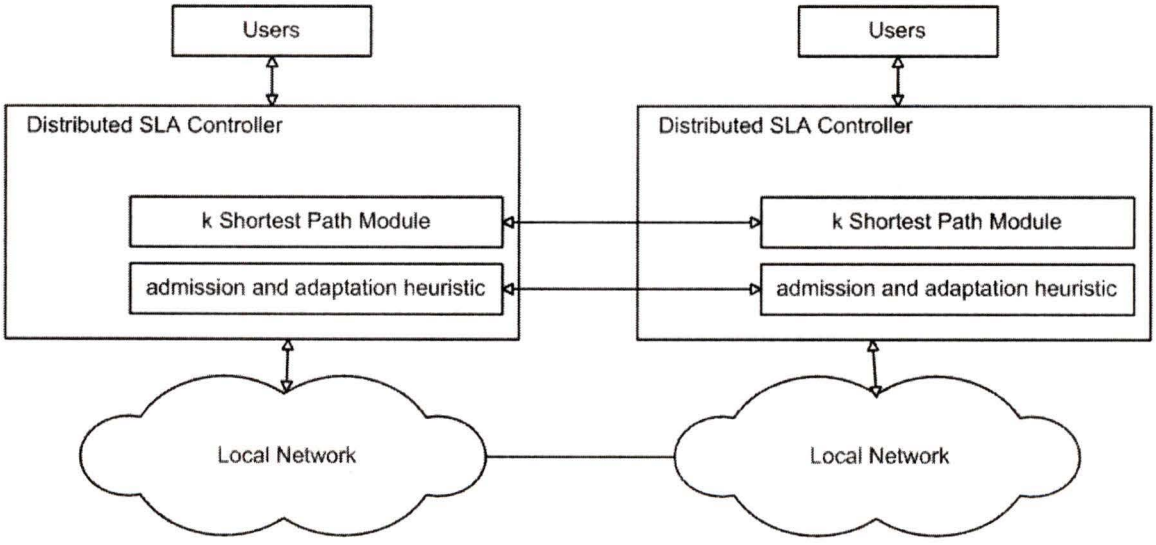

Figure 5.1: The overall architecture of a distributed SLA controller.

dSLAOpt was derived from the code of the non-distributed SLAOpt, greatly impacting the design of dSLAOpt. The main requirements of dSLAOpt, over SLAOpt, included:

1. an addressing scheme for nodes,
2. the ability to perform peer-to-peer communication,
3. the implementation of a hierarchical *k* shortest path module, and
4. the definition of a global network, along with the local network.

5.1.1 An Addressing Scheme

In SLAOpt, all objects (links, nodes, and SLAs) were referred to by textual names. The first step in converting SLAOpt to dSLAOpt involved the creation of an addressing class, *nnAddress*, where addresses were in the form of <network ID>.<unique ID>.

5.1.2 Peer-To-Peer Communication

The concept of communication was obviously not required in SLAOpt, thus one of the largest changes included the addition of threads for communication. The chosen architecture is shown in Figure 5.2, where Message Sender and Message Receiver threads exist for communication to each peer. The receiver thread typically receives a message from a peer controller, performs a small calculation, and returns a result, by sending through the Message Sender thread. Sending and receiving messages are thus non-blocking calls. We assumed reliable communication, simplifying our design.

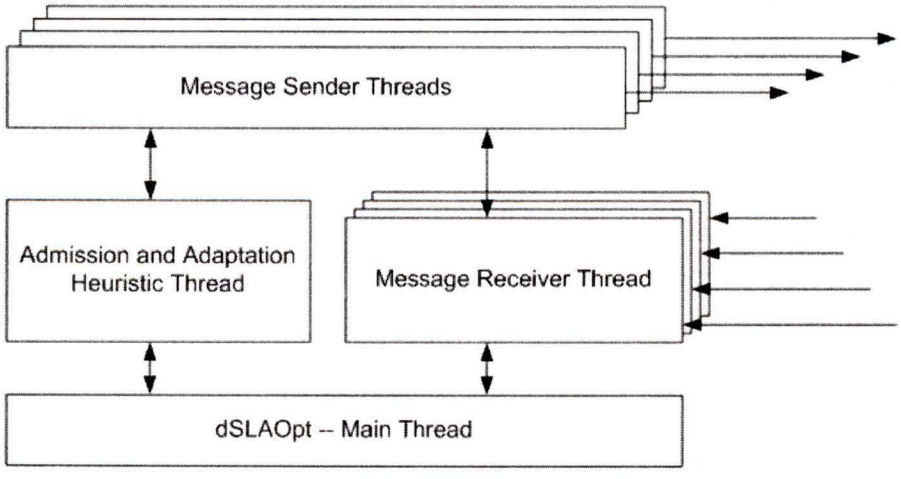

Figure 5.2: The thread architecture of dSLAOpt.

5.1.3 Hierarchical *k* Shortest Path Module

As part of the implementation of the hierarchical *k* shortest path heuristic, a simple cache was used to store previously calculated paths. The cache was implemented as a *pathCache* object, storing *kCompletePath* paths, which themselves consist of source and destination node addresses, along with a series of path descriptions.

Paths are described by the entire list of links which compose the path, not as a list of local links and non-local path identifiers, as was described in Chapter 3. This was done for simplicity of design. The non-local link identifiers in a *kCompletePath* were not used in any calculation, except in P-HEU, which specifically allows the passing of non-local link identifiers.

5.1.4 Global Network

The centralized SLAOpt only understood the concept of a single local network. dSLAOpt required the concept of a global network as well. This required some minor modifications to the network class, *nnNetwork*, of which two instantiations were used: one for the local network, and one for the global network. The addition of the global network also required changes to the GUI, as will be seen later.

5.2 Optimal Admission and Adaptation Heuristics

The optimal admission and adaptation heuristics were defined in the Knapsack class. Although each heuristic could have simply extended the Knapsack class, and been implemented separately, they were implemented together in the same class for simplification of coding. Some of the main methods of the Knapsack are shown in Table 5.1.

Method	Description
KnapSack(int heuristic, int ID,int nServers, nnNetwork lnetwork, nnNetwork gnetwork)	The constructor. Defines the heuristic the controller should use, the knapsacks ID, how many controllers are in the network, and the local and global network topologies.
void load_SLAs(Vector SLAs)	Prepares a number of SLAs for admission.
void start_adapt()	Starts the admission and adaptation heuristic.
int getGlobalRevenue()	Returns the global revenue received from all controllers from the last heuristic execution.
int getLocalRevenue()	Returns revenue received from this local controller from the last heuristic execution.
Vector getRejectedSLAs()	Returns a list of SLAs that were rejected during the execution of the heuristic.
Vector getLocalSLAs()	Returns the local SLAs currently admitted into the system.

Table 5.1: Major methods of the Knapsack class.

5.3 Graphical User Interface

The GUI of dSLAOpt is similar to that of SLAOpt, except for the addition of the Global Network window, as shown in Figure 5.3. The GUI is not required to execute a simulation, but it does allow the user to view and modify the state of the system at run-time. The various windows of the GUI are described below.

The Network Graph window displays the topology of the single network that the simulated controller is controlling. The state of the network is also visually presented through coloured coding of the displayed links. Lightly utilized links are coloured green, moderately utilized links are coloured yellow, and highly utilized links are coloured red.

The Global Network Graph window is similar to the Network Graph window, except that it displays the topology and network state of the global network. As with the Network

Graph window, the links in the Global Network Graph window are coloured; however, since each simulated controller only controls some of the global links (except in P-HEU), only those it controls are colour-coded within this window.

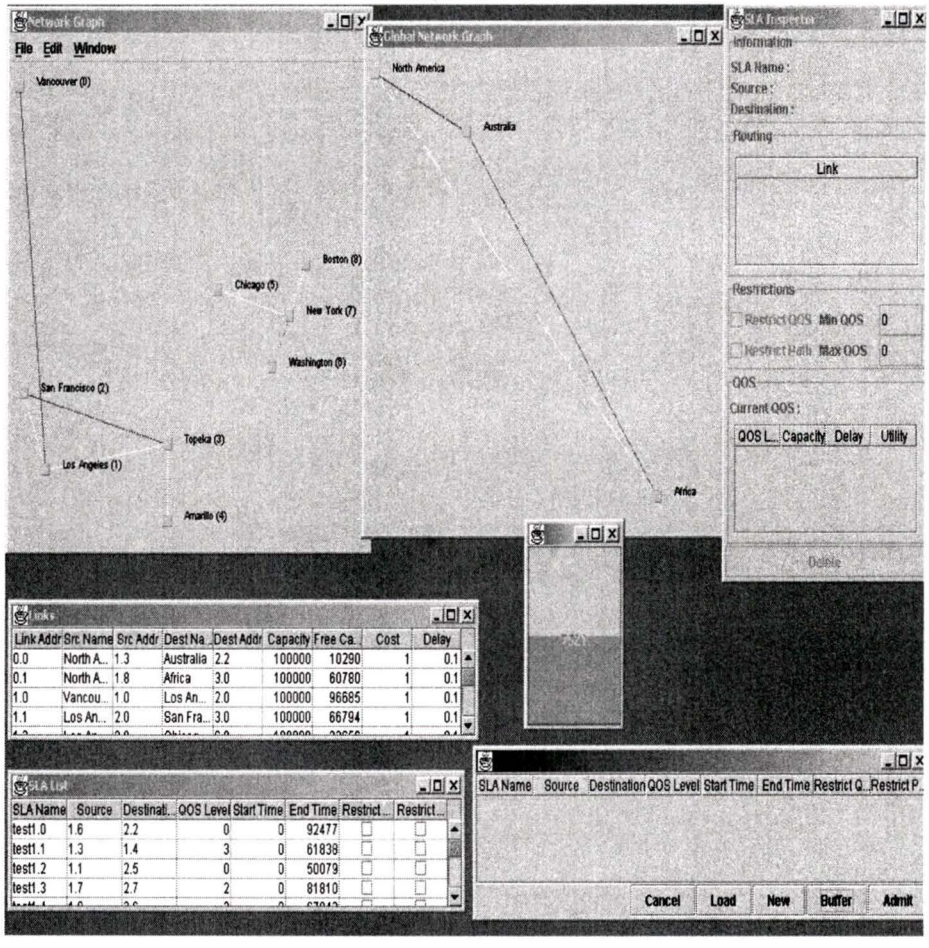

Figure 5.3: View of simulation on one computer.

The Links window lists all the links, both local and global, that are controlled by the simulated controller. The links, listed by their address, are listed along with their attributes and their current state. The attributes include their source and destination addresses, the total capacity of the link, a cost of utilizing the link, and the associated delay. Their state includes the available capacity on the link.

The SLA List window contains a listing of all SLAs that have been admitted and are currently being served by the simulated controller. Each SLA, identified by a textual

name, is listed along with the source and destination nodes of the SLA, the QoS level currently being serviced, the start and end times of the SLA, and the value of the flags for the restricting of downgrading and altering the current path.

The SLA Inspector window displays detailed information for an SLA selected within the SLA List window. The SLA Inspector shows the network resources currently being used by the SLA, presents all the QoS levels of an SLA, and displays the flags for downgrading and switching to new paths. The SLA Inspector also allows the user to delete an SLA, removing it from the system.

The SLA Admission Queue window contains the same information as that of the SLA List window, except all SLAs in the admission queue are awaiting admission, and thus all the SLAs have a current QoS level of -1, representing the null QoS level. The Load button allows the user to read in a number of SLAs from a file, displaying the SLAs in the window. The Buffer button processes the SLAs and prepares them for admission. The Admit button starts the admission process, executing an admission heuristic on the SLAs buffered not only in the current simulated controller, but in each of the interconnected simulated controllers. Cancel simply closes the window, and if some SLAs have been loaded, but not buffered, then they will be removed from system.

6 Performance Evaluation

In this chapter we analyze the performance of A-HEU, D-HEU, G-HEU, P-HEU, and a proposed modification of A-HEU, called *fast* A-HEU (FA-HEU), using the simulated distributed optimal SLA admission controller, dSLAOpt.

Please recall that:

- **D-HEU** simply replicates the computation of I-HEU at each controller;
- **G-HEU** first coordinates the admission of SLAs based on highest bids first, then admits local intra-network SLAs, not requiring confirmation from other controllers;
- **A-HEU** runs I-HEU at each controller, based on local resources only, then performs arbitration among the controllers to determine which SLAs to admit or adapt; and
- **P-HEU** pre-assigns $1/m^{th}$ of the capacity of each link in each network to each controller to reduce the amount of inter-controller negotiation required, where m is the number of interconnected networks.

First we will introduce a definition of contention, which is used in most of the tests to describe the offered load on the network. Then the test bed and the process of creating SLAs are outlined. The variables used in the tests are listed and the results are then presented, together with plausible explanations of some of the more interesting results obtained. Finally, based on analysis of the tests, modifications to improve the performance of A-HEU and FA-HEU are proposed and evaluated.

6.1 Contention

We want to examine heuristic performance as a function of network offered load, as we know that offered load would strongly affect performance. Hence we need to define network load, and control it during our experiments.

First, we need to estimate the load of the network. We call this a measure of *contention*, defined so that:

contention ≥ 1 implies the network is heavily loaded, and

contention < 1 implies the network is lightly loaded.

Roughly, contention is the ratio of the average bandwidth for SLAs admitted and being admitted, to the total bandwidth of the links in the networks. Precisely, contention is defined[2] as:

$$contention = \frac{n \times h_{avg} \times b_{avg}}{b_T}, \tag{6.1}$$

where n is the number of SLAs in the system, including those being admitted, b_{avg} is the average amount of bandwidth required per SLA, h_{avg} is the average hop length of the SLAs, and b_r is the cumulative bandwidth of the networks, the sum of the bandwidths of all links owned by the networks.

b_{avg} is complex to define because an SLA can contain numerous QoS levels, each with a different bandwidth requirement. Therefore, b_{avg} is defined as:

$$b_{avg} = \frac{\sum_{s \in S} bandwidth(s)}{n}, \tag{6.2}$$

where S is the set of all SLAs in the system, including those being admitted, and *bandwidth(s)* returns the average bandwidth requirement of all the QoS levels in SLA s, not including the null QoS level.

---

[2] Eric Gowland, a fellow member of the PANDA lab, helped formulate the definition of contention as used within this thesis.

The average hop length, h_{avg}, is defined as the average hop length of the k candidate paths for SLAs in the system. h_{avg} is determined by considering the k candidate paths for all SLAs in S.

Contention can exceed 1.0 without causing congestion for 2 reasons. First, the definition of contention includes both admitted SLAs, and those awaiting admission. Second, contention uses the average bandwidth of SLAs; however, SLAs have multiple QoS levels, and thus can be downgraded to a QoS level requiring *below average* bandwidth, thus allowing more SLAs to be admitted into the system.

This definition of contention is an estimate; inaccuracies arise because it does not consider the possible bottlenecks due to external links or locality of reference. For example, if the degree of locality of reference is low, a large percentage of traffic is carried over external links, and if the bandwidth of those external links is low, relative to the internal links, then a higher degree of contention will occur than our estimator will indicate.

6.2 Test Bed

To evaluate the heuristics, personal computers were used, each running a copy of dSLAOpt. Each computer contained a Pentium-2 400Mhz CPU, 64MB of RAM, ran Windows XP, and were connected directly through a NetGear 5 Port Fast Ethernet Switch (Model FS105) as 10Mbit Ethernet. Fast links between the machines were not considered an issue, as DSCs are a logical entity, and can be placed in the same room, even though the networks they control are located on separate continents.

6.3 Variables

To help us describe the tests in the following section, we define a number of variables in Table 6.1.

BandwidthRange	Bandwidth of QoS levels are chosen from this range
BidRange	Bid, or value of utility, of QoS levels are chosen from this range
Contention	Estimated amount of contention during the test
kCandidatePaths	The number of candidate paths considered per QoS level
LocalityOfReference	Ratio of the number of intra-network SLAs to the total number of SLAs to be introduced to the system
NumberOfNetworks	Total number of interconnected networks
NodesPerNetwork	Total number of nodes per local network
LinksPerNetwork	Total number of links per local network
ExternalLinks	Total number of external links (links between networks)

Table 6.1: Test variables defined.

The network chosen for most of the analyses has 9 nodes (*NodesPerNetwork*=9) and 14 links (*LinksPerNetwork*=14), as shown in Figure 6.1. It was based on an early configuration of UUNET, as was used in the preliminary evaluation of SLAOpt [10], the centralized predecessor of dSLAOpt.

For most of the tests, this 9-node network was replicated 4 times (*NumberOfNetworks=4*) and interconnected with a total of 12 links (*ExternalLinks=12*). The connection of an external link within a network was done arbitrarily. It should be assumed this configuration was used for a test, unless otherwise specified. The capacity of all links is 100Mbps, unless otherwise specified.

Another set of 4 interconnected networks (NumberOfNetworks=4) was used for the first 3 tests (see Figure 6.2). Each network comprised of 31-node networks

(*NodesPerNetwork=31*) with 38 links each (*LinksPerNetwork=38*). The networks were interconnected with 12 links (*ExternalLinks=12*). All links had a capacity of 100Mbps.

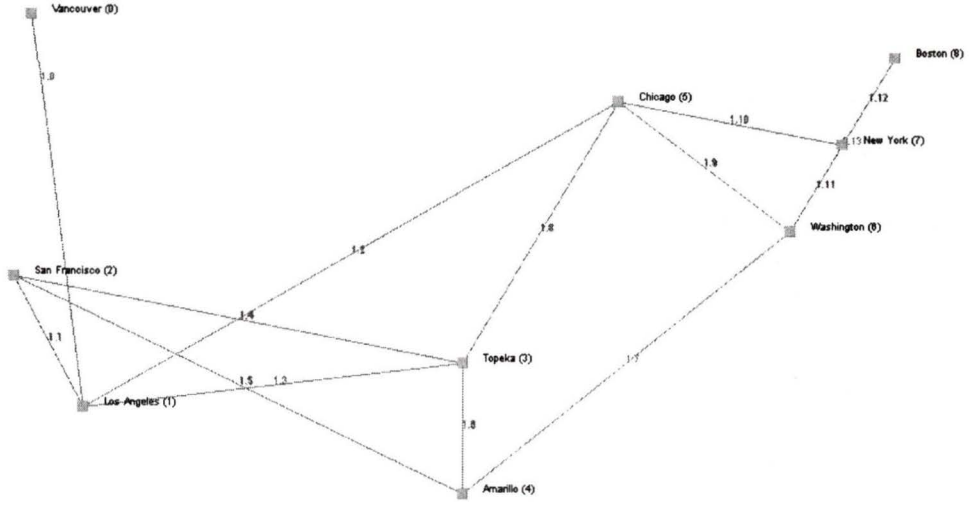

Figure 6.1: A 9-node network used for testing, based on UUNET.

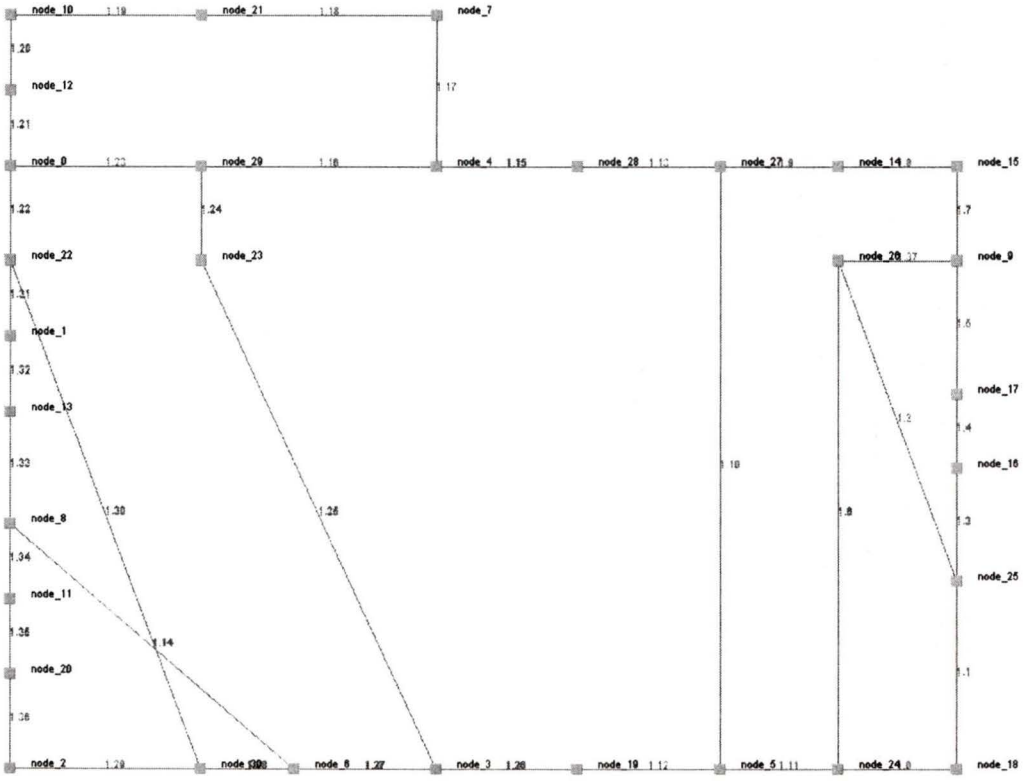

Figure 6.2: A 31-node network used for testing *contention* and *kCandidatePaths*.

6.4 Creating SLAs

The SLAs for a test were created in advance so the exact scenario, with the identical set of SLAs, could be executed by each heuristic. As a result, we cannot guarantee contention over a period of time, as different heuristics will admit more (or fewer) SLAs, causing a greater (or lower) level of contention.

The average bandwidth of SLAs, b_{avg}, used in the definition of contention is estimated to be the middle value of the range *BandwidthRange*. This is required because we cannot know which SLAs will be admitted at any one time, in order to obtain a true value of b_{avg}. To estimate h_{avg}, a large number of SLAs were created (200 in our tests), their shortest paths were calculated, and the average hop length of the $200 \times k$ paths was calculated.

For the initial epoch, enough SLAs are created to cause the required level of contention. At the start of subsequent epochs, 10% of the SLAs that have not yet expired, *numRemovePerRound*, are marked for removal. This models the expiration of SLAs.

The number of SLAs to create per subsequent epoch, *numAddPerRound*, depends on the set level of contention. For $contention \leq 1.0$:

$$numAddPerRound = num\,\mathrm{Re}\,movePerRound\ . \tag{6.3}$$

For $contention > 1.0$: (6.4)

$$numAddPerRound = numSLAs \times (contention - 1.0) + num\,\mathrm{Re}\,movePerRound\ ,$$

where *numSLAs* is the number of SLAs to add to an empty network to cause contention of 1.0. *numSLAs* can be derived from the equation for *contention*.

For $contention \leq 1.0$, we simply assume all SLAs are admitted into the system. Thus, for each epoch, we create the same number of SLAs as the number of SLAs expiring. For

$contention > 1.0$, we assume the network is full at contention of 1.0, thus, to maintain the contention desired, we must add $NumSLAs \times (contention - 1.0)$ SLAs as well as the number of SLAs that have expired.

The results given in the next section are based on averages taken over 50 epochs. A network is loaded with SLAs and 10 subsequent epochs are monitored. This is repeated 5 times to accumulate data for 50 epochs. We do not monitor for more than 10 subsequent epochs because, over time, each heuristic will drift farther from the initial estimation of contention, a value that we are attempting to control as an independent variable.

We randomly assign the number of QoS levels for each SLA to be a number between 1 and 4. Let l_s be the number of QoS levels for SLA s. We then select l_s random bandwidths from *BandwidthRange*, and l_s random delays from the range:

$$(candidatePath_s^1, candidatePath_s^3 + 0.25 \times (candidatePath_s^3 - candidatePath_s^1)),$$

where $candidatePath_s^n$ is the delay of n^{th} fastest candidate path for SLA s. The range for delays was chosen to assure that most, but not all, QoS levels were feasible. The upper value of the delays was selected to be slightly above the 3rd fastest path, since most of the test results use *kShortestPaths* = 3.

We finally select bids randomly from the interval *BidRange* = ($20, $300). The selected bandwidths and delays are then assigned to the QoS levels so that a QoS level with a higher bid has a higher bandwidth requirement and a lower delay requirement compared to the other QoS levels for that SLA.

6.5 Results

"The purpose of computation is to obtain insight, not numbers."

-R.W. Hamming.

Due to the complexity of the MMMKP, it is infeasible to find exact solutions for large test cases, such as those solved in this chapter. According to Akbar [8], I-HEU achieves approximately 96% optimality for smaller data sets. To give the reader an approximation of true optimality, we compared D-HEU against I-HEU using the data in Section 6.5.5, where *LocalityOfReference* = 0.8.

By considering the 4 interconnected networks as a single network, we used the centralized SLAOpt to admit the SLAs using I-HEU. I-HEU resulted in utility of 88,030 at low contention and 145,358 at high contention, compared with D-HEU yielding utility of 83,081 and 145,358 at low and high contention, respectively. Since D-HEU yielded utility between 93.4% and 94.4% of that of I-HEU, and since I-HEU is estimated at being 96% optimal, we therefore estimate D-HEU as being approximately 90% optimal.

6.5.1 Analyzing A-HEU

A-HEU uses I-HEU to determine the order in which SLAs should be admitted. A test was devised to determine the usefulness of step 3 of I-HEU (see Section 2.3.2.1), in which local maxima and minima are avoided. This step was suspected to be time consuming and unlikely to greatly increase utility, especially due to the way DSC's arbitrate over admitting and upgrading SLAs in A-HEU. To test this hypothesis, we removed step 3 from the heuristic, calling this altered heuristic *FA-HEU* (fast A-HEU).

The cost of executing step 3 of I-HEU can be seen in Figure 6.3. Figure 6.4 shows that the extra time cost of A-HEU does not result in improved utility over the simplified FA-HEU. We continue to test both of these heuristics throughout the remainder of the chapter, to further explore the performance of each.

[*kCandidatePaths*=3, *LocalityOfReference*=0.8, *BandwidthRange*=(2Mbps, 5Mbps)]

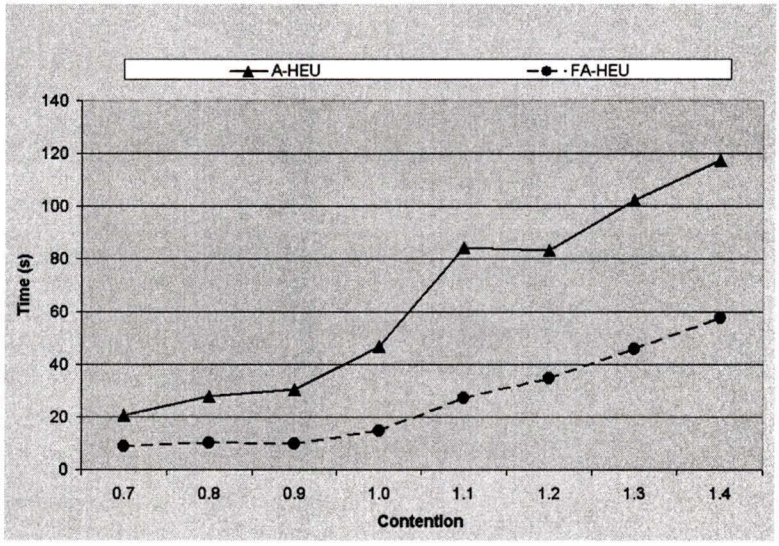

Figure 6.3: Time comparison of A-HEU with and without infeasible upgrade

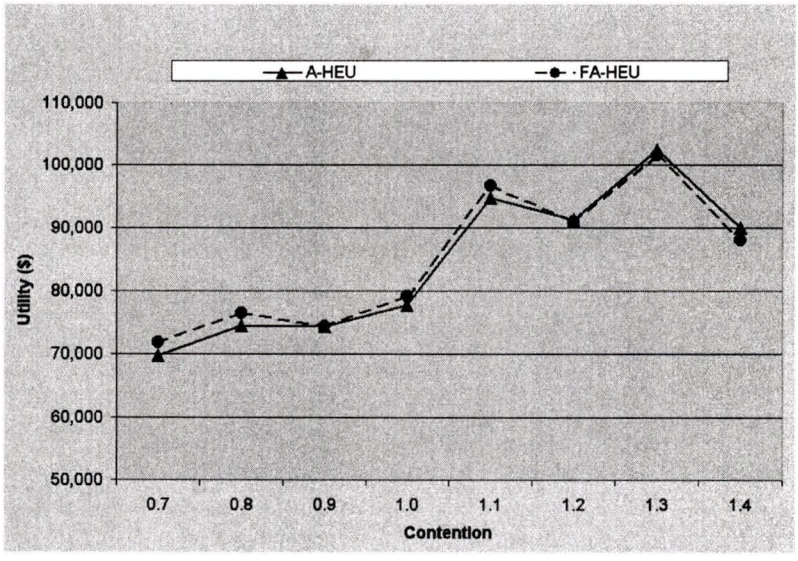

Figure 6.4: Utility comparison of A-HEU with and without infeasible upgrade step

6.5.2 Efficient Number Of Candidate Paths

The number of candidate paths used per QoS level for an SLA drastically affects the running times of all of the heuristics. It is thus important to determine the effect of the number of candidate paths used on the utility achieved by the system.

We tested each heuristic with k = 1,...,5, where k is the number of candidate paths considered for each QoS level of an SLA. Additionally, we tested each heuristic using a network with low contention (0.7), as well as with high contention (1.2).

Figure 6.5 and Figure 6.6 show the average time required to execute the MMMKP heuristics under both low contention and high contention, respectively. The trend for the various values of k is similar under the different levels of contention. D-HEU is affected most by an increase in k, while G-HEU and F-HEU are least affected. Recall that the message passing complexity of D-HEU is dependent on k. It is obvious that reducing the number of candidate paths considered should improve the running time of most of the heuristics.

[*LocalityOfReference*=0.8, *BandwidthRange*=(2Mbps, 5Mbps)]

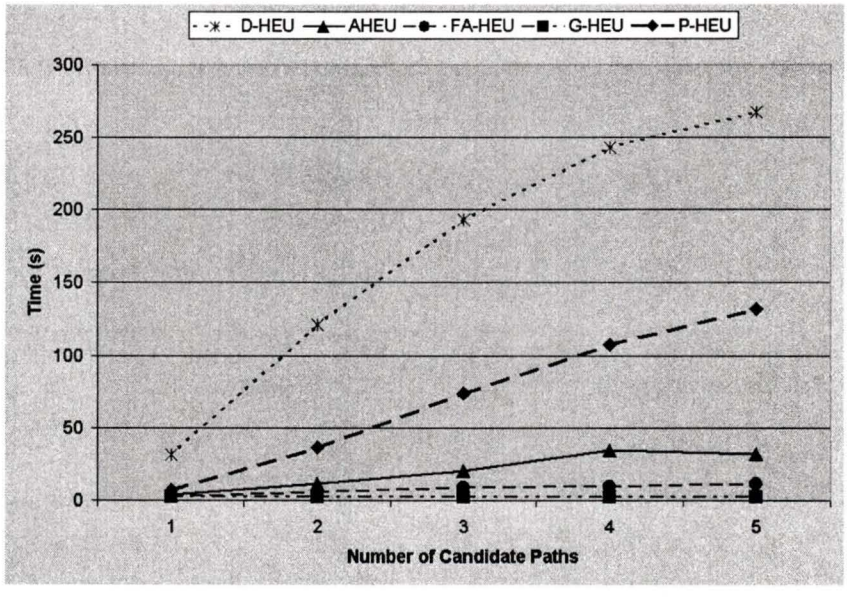

Figure 6.5: Time by k shortest paths with contention of 0.7.

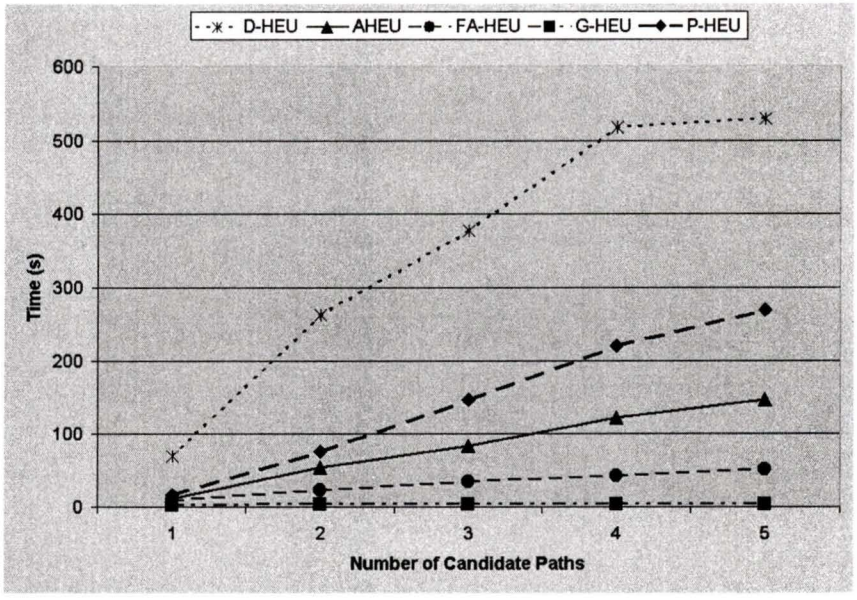

Figure 6.6: Time by k shortest paths with contention of 1.2.

Since we prefer a lower k to reduce heuristic execution time, especially for more complex heuristics, we now investigate the trade-off in terms of utility. Figure 6.7 and Figure 6.8 show the utility achieved by the various heuristics at both low and high contention, respectively. The figures demonstrate that there is little, or no, incentive to use a k greater than 2 or 3, except for A-HEU and FA-HEU.

When there is high contention for network resources, A-HEU and FA-HEU actually lose potential utility as k increases. Recall that in A-HEU and FA-HEU, each controller proposes upgrades for its local SLAs based on local resources only. Inter-network SLAs close to gateways are thus favoured with these heuristics, because even though they may require many links in total, they may require few local links. This bias is increased when more candidate paths are considered. The more candidate paths that are considered, the more inter-network SLAs are considered for admission, as longer complete paths may require fewer local resources. With the bias towards inter-network SLAs increasing as the number of candidate paths are considered, then, due to the bottleneck of the external links and links attached to gateways, fewer SLAs will actually be admitted. This is because the heuristic ceases when an SLA fails to be admitted.

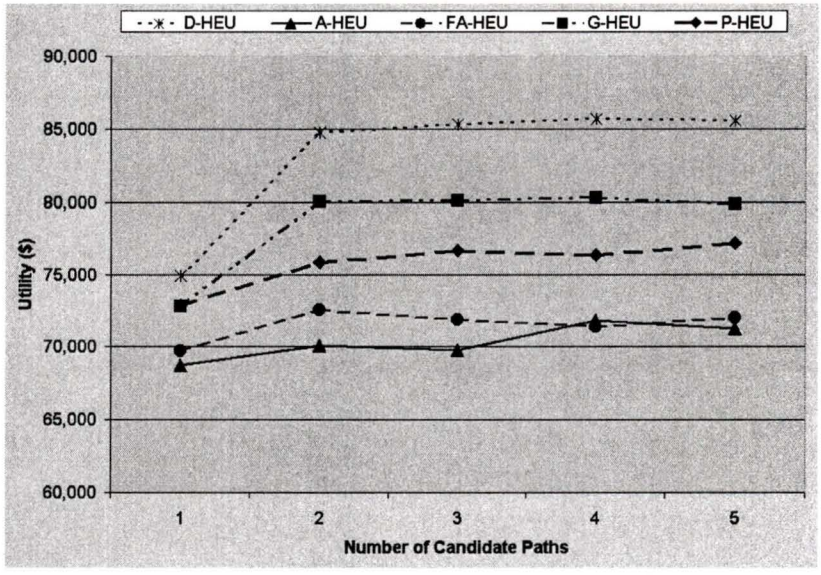

Figure 6.7: Utility by varying k shortest paths with contention of 0.7.

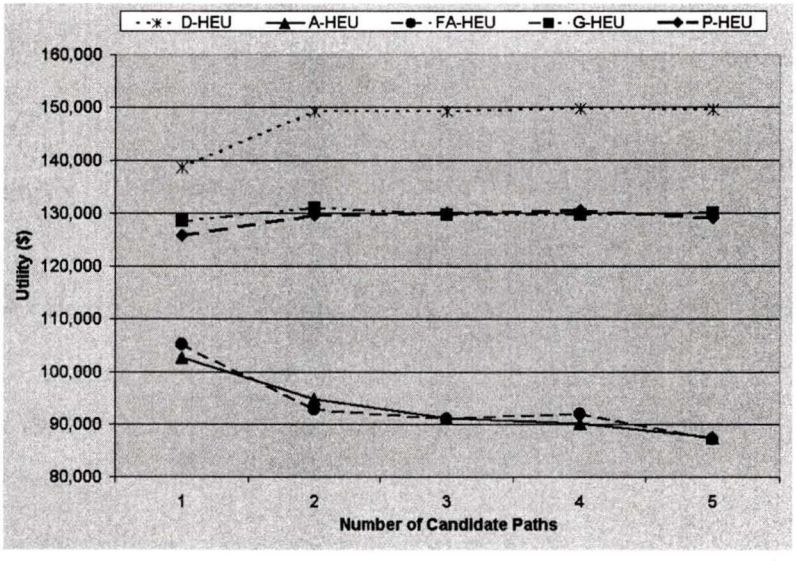

Figure 6.8: Utility by varying k shortest paths with contention of 1.2.

As a result of this test, we will use *kCandidatePaths=3* for the remainder of this chapter.

6.5.3 Varying Contention

The next evaluation compares the MMMKP heuristics with increasing levels of contention. From the definition of contention, it should be understood that an increasing number of SLAs are requesting admission as *contention* increases.

As could be expected, as contention increases, and more SLAs are being considered for admission, both time (Figure 6.9 and Figure 6.10) and utility (Figure 6.11) increase for each of the heuristics. A-HEU and FA-HEU only have a slight upward trend in utility as contention increases, again demonstrating the inefficiencies of these heuristics. A-HEU and FA-HEU perform worse at higher contention due to the fact that the heuristic terminates once an SLA fails to be admitted. At higher contention, such a condition is likely to occur earlier in the processing of SLAs due to the larger number of SLAs, especially the inter-network SLAs. Since A-HEU and FA-HEU have a bias towards inter-network SLAs, it is possible that the heuristic will terminate early, due to the bottleneck of resources for inter-network SLAs.

[*LocalityOfReference*=0.8, *BandwidthRange*=(2Mbps, 5Mbps)]

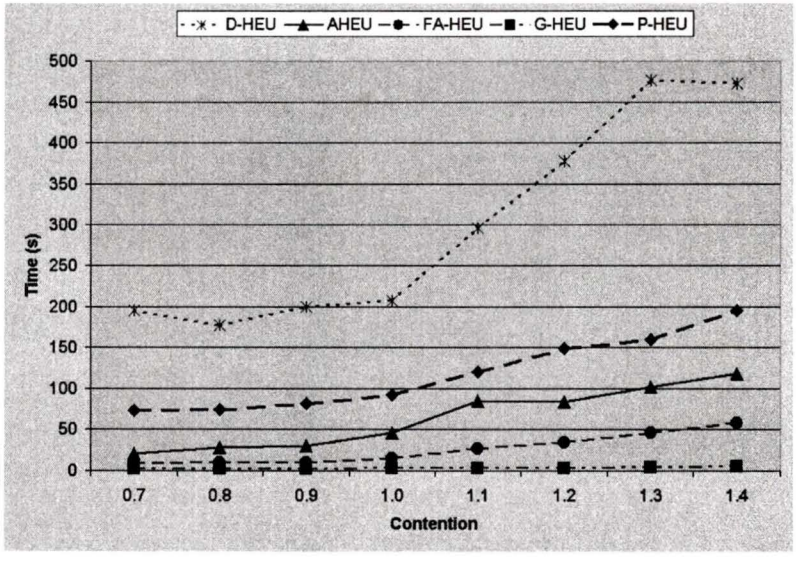

Figure 6.9: Time comparison of heuristics against varying contention.

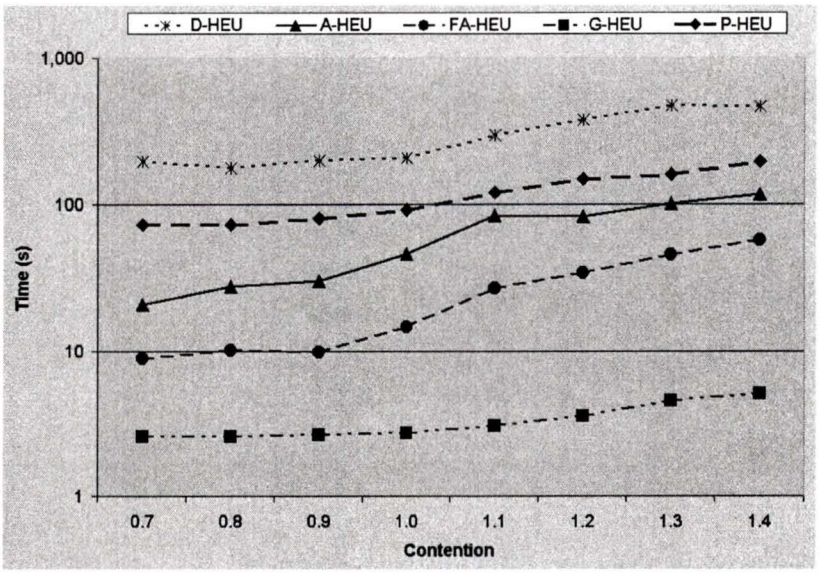

Figure 6.10: Logarithmic time comparison of heuristics against varying contention.

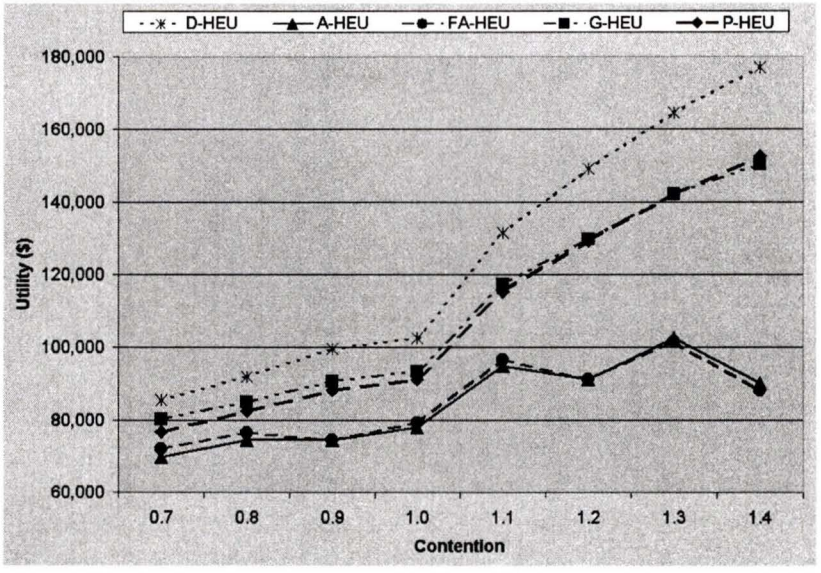

Figure 6.11: Utility comparison of heuristics against varying contention.

P-HEU performs consistently with respect to D-HEU, achieving 89% of the utility of D-HEU at contention of 0.7 down to 86% of the utility of D-HEU at contention of 1.4. G-HEU performed better at low contention, achieving 93% of the utility of D-HEU at contention of 0.7, but achieved only 84% of the utility of D-HEU with contention of 1.4.

Throughout this thesis we are more concerned about the performance of P-HEU with respect to utility than time. Realize that P-HEU can be implemented using any MMKP

heuristic. Since we used I-HEU, we are interested in comparing P-HEU to D-HEU, which is the distributed implementation of I-HEU. Another heuristic that scales well with respect to the number of resources may need to be considered since P-HEU has to solve an MMKP with m times the number of resources, where m is the number of interconnected networks. For example, we could simply use I-HEU without step 3 of that heuristic within P-HEU – as we did for FA-HEU.

Finally, the inefficiency of G-HEU at higher contention is simply due to its simplistic nature of admission.

6.5.4 Varying Average Bandwidth Per SLA

Next we compare the heuristics at low (0.7) and high (1.2) levels of contention by varying the average bandwidth per SLA. The range in bandwidth is 2Mbps, so at 7.5Mbps, *BandwidthRange = (6.5Mbps, 8.5Mbps)*, and at 1.5Mbps, *BandwithRange = (0.5Mbps, 2.5Mbps)*. Recall our definition of contention, and recall that reducing the average bandwidth per SLA at a given level of contention implies that a larger number of SLAs must request admission.

As before, G-HEU outperforms A-HEU and FA-HEU, both in utility earned and time of execution. A-HEU, FA-HEU, and G-HEU also all perform much better, relative to D-HEU, at low contention than at high contention, especially when each SLA consumes a lower percentage of total link bandwidth. This simply demonstrates the inefficiency of these protocols. With an average SLA bandwidth of 1.5Mbps at low contention, FA-HEU achieves 82% of the utility of D-HEU in 1.5% of the time. At high contention, FA-HEU achieves only 71% of the utility of D-HEU in 5% of the time.

Few smooth utility curves, as seen in Figure 6.13 and Figure 6.15, will be shown in this thesis, especially for A-HEU and FA-HEU. When we keep contention constant, but vary the average bandwidth of an SLA, we must increase the number of SLAs in the system. Since the *BandwidthRange* also remains constant, the amount of utility yielded by the system is inversely related to the average bandwidth. As the average bandwidth halves,

the utility yielded by the system approximately doubles for every heuristic, except P-HEU.

[*LocalityOfReference=0.8*]

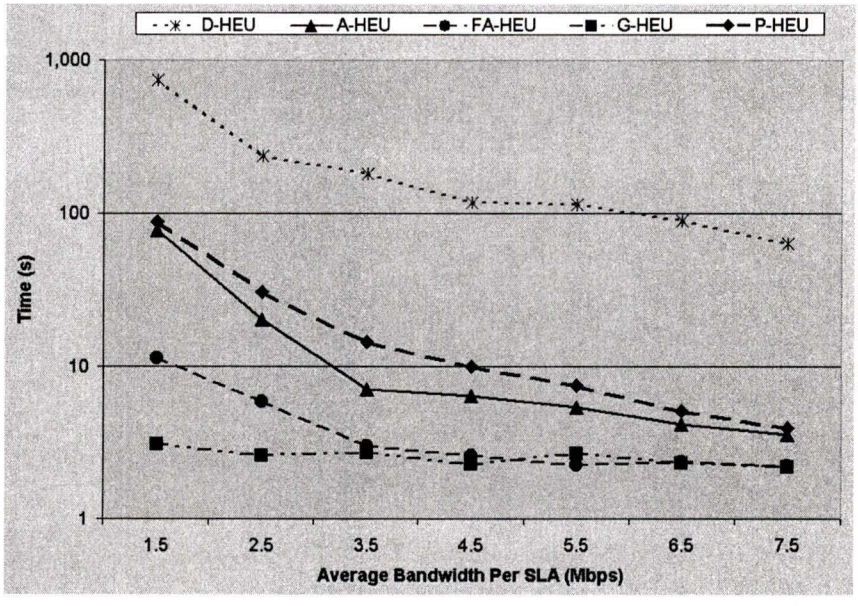

Figure 6.12: Execution time of heuristics for SLAs of increasing bandwidth at low contention.

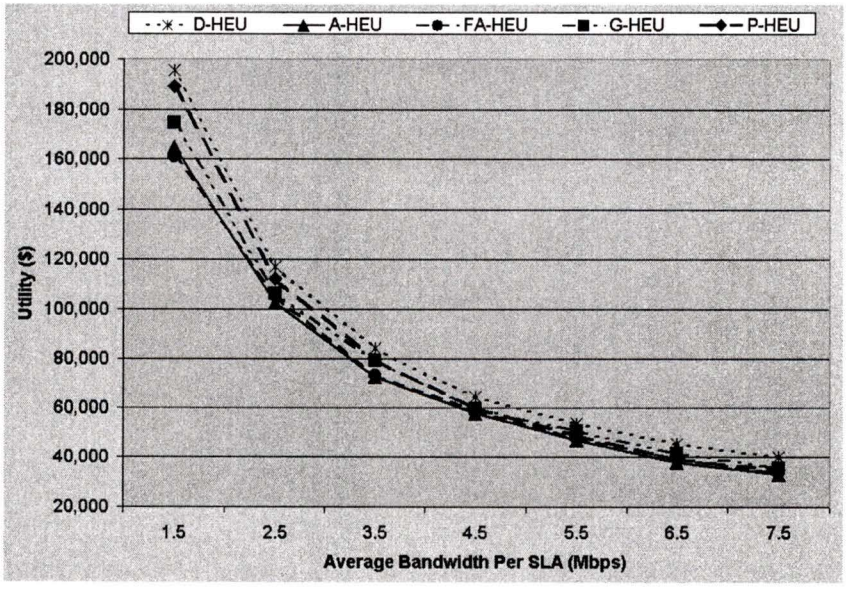

Figure 6.13: Utility of heuristics for SLAs of increasing bandwidth at low contention.

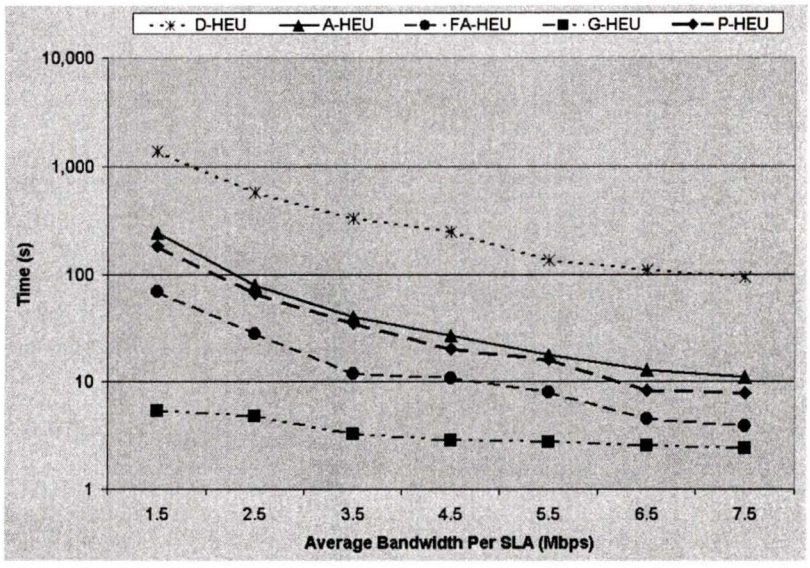

Figure 6.14: Execution time of heuristics for SLAs of increasing bandwidth at high contention.

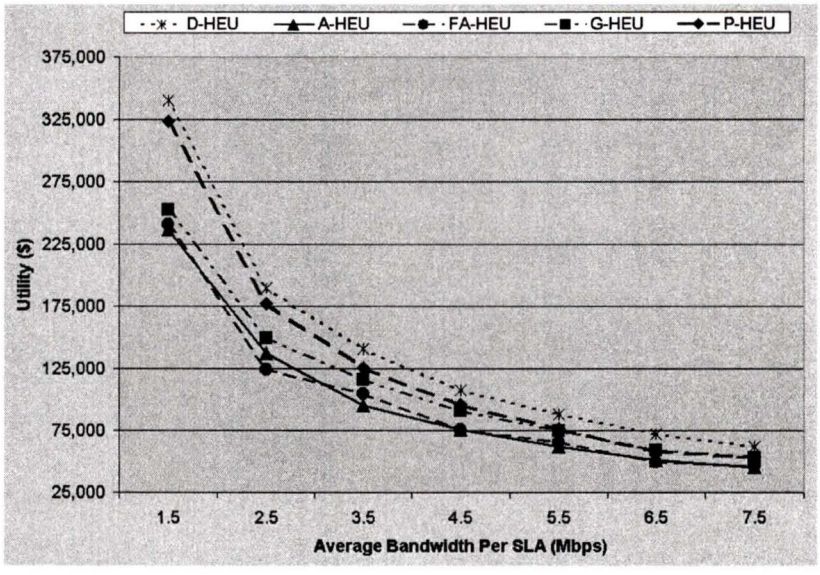

Figure 6.15: Utility of heuristics for SLAs of increasing bandwidth at high contention.

P-HEU performed as expected, improving its performance as the bandwidth of each SLA was reduced. At high contention, we can see P-HEU increasing from 88% of D-HEU to 95% of D-HEU as the bandwidth per SLA was reduced. It was expected, and was the reason for some of our assumptions in Chapter 4, that P-HEU would approach the performance of D-HEU as the average bandwidth of each SLA reduced relative to the link capacity. As the average bandwidth of each SLA reduces, and the number of SLAs increases, there are two factors that affect P-HEU. First, in P-HEU, each controller may

have unused bandwidth on a particular link. If there are 4 controllers, and unallocated bandwidth b on a link per controller, then there is a total of $4b$ unallocated bandwidth on the link. If average bandwidth per SLA is high, then the unallocated bandwidth b could also be high, resulting in inefficiencies in P-HEU. Second, as the number of SLAs per controller increases, the more decisions a controller can make in its effort to reach optimality, and the less chance an upgrade in one controller could be made more efficient by a downgrade in another controller (which will not explicitly occur in P-HEU).

6.5.5 Varying Locality of Reference

The other tests set locality of reference at 0.8. We now examine the heuristics with locality of reference varying from 0.0 – where no SLAs are intra-network SLAs – to 0.8. Locality of reference of 1.0 was not tested, as all SLAs would be intra-network SLAs, which would not require a distributed heuristic for admission.

Again, we compare the results at both low (0.7) and high (1.2) contention.

The only data worth noting is A-HEU's poor performance with low localities of reference and high contention. This is a result of stopping the upgrading of SLAs once one SLA is not upgradeable among all SLAs. Since external link capacity is a bottleneck, and if most SLAs are to use those external links, then A-HEU (and FA-HEU) will respond poorly. Also, as described in Section 6.5.2, A-HEU has a bias towards inter-network SLAs, which further worsens the bottleneck of the external links.

[*BandwidthRange*=(2.5Mbps, 4.5Mbps)]

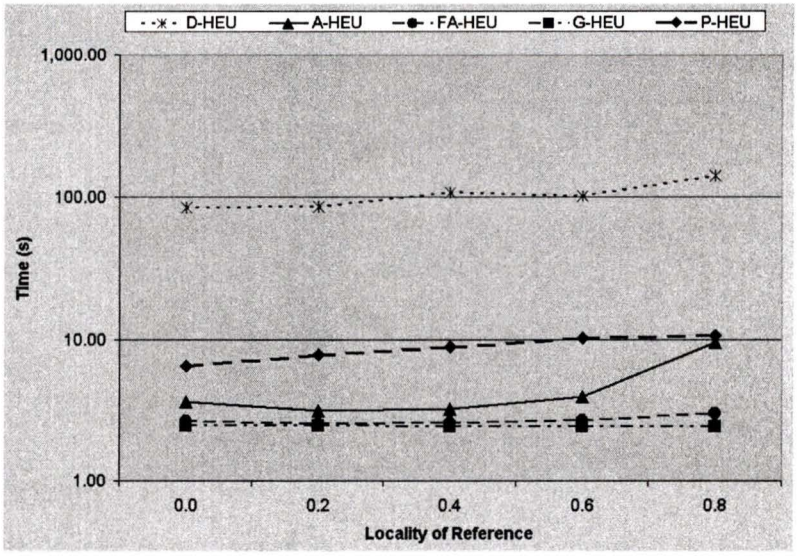

Figure 6.16: Logarithmic time comparison of heuristics against varying locality of reference at low contention.

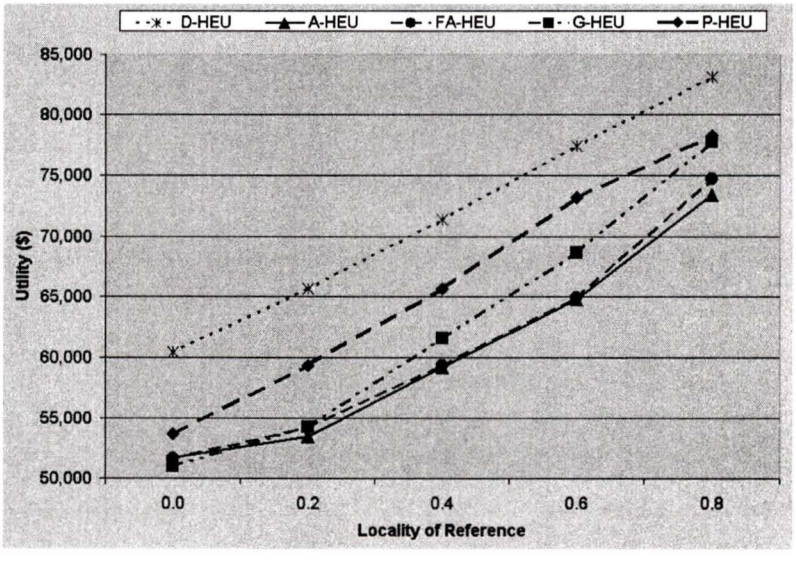

Figure 6.17: Utility comparison of heuristics against varying locality of reference at low contention.

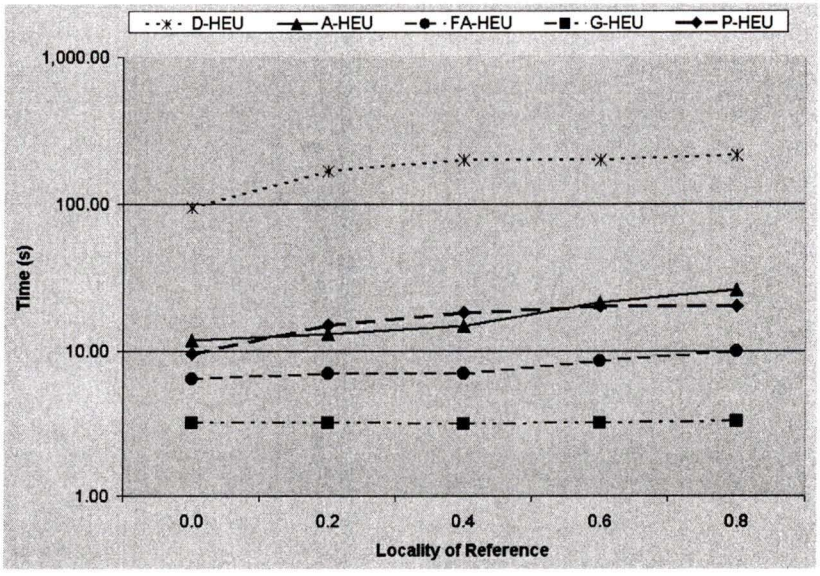

Figure 6.18: Logarithmic time comparison of heuristics against varying locality of reference at high contention.

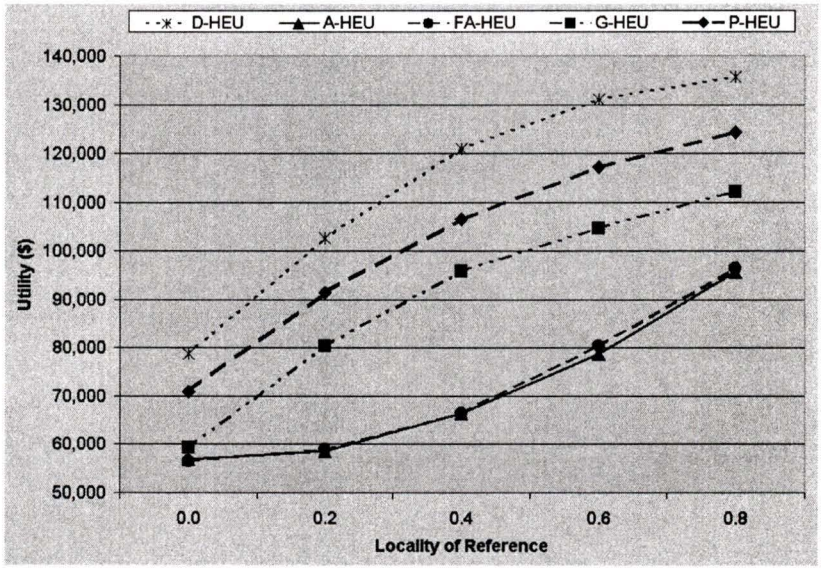

Figure 6.19: Utility comparison of heuristics against varying locality of reference at high contention.

6.5.6 Admission of Inter-Network Versus Intra-Network SLAs

Here the heuristics are tested with respect to fairness; in particular, fairness between the admission rates of inter-network and intra-network SLAs. We show the results of admitting SLAs on a network with *LocalityOfReference* = *0.5*, at both low and high

contention. Table 6.2 shows the percentage of inter-network and intra-network SLAs that are admitted. Since there is such a small number of external links relative to internal links, we increased the bandwidth on the external links by a factor of 1,000 to exclude them as a bottleneck.

[*BandwidthRange*=(2.5Mbps, 4.5Mbps), *LocalityOfReference*=0.5]

	SLA Admitted at Low Contention (%)		**SLAs Admitted at High Contention (%)**	
Heuristic	**Inter-network SLAs**	**Intra-network SLAs**	**Inter-network SLAs**	**Intra-network SLAs**
A-HEU	83	76	25	10
FA-HEU	83	78	24	10
G-HEU	67	95	4	31
D-HEU	92	97	27	31
P-HEU	89	91	29	26

Table 6.2: Bias towards inter/intra-network SLAs of heuristics at low and high contention.

Most notable in the table is the low percentage of inter-network SLAs admitted by G-HEU, especially at high-contention. This occurs because, in G-HEU, at the first failure to admit an SLA, the controllers respond by admitting only local intra-network SLAs. Although bias was expected, the degree of the bias exhibited should be considered before using this heuristic in a real system.

Also worth noting is the higher admission rate for the intra-network SLAs, over the inter-network SLAs, in D-HEU. This is the result of two major factors. First, the internal links directly connected to gateways will be highly utilized and become a bottleneck, especially for inter-network SLAs. Second, D-HEU upgrades SLAs based on value per aggregate resource. Aggregate resource is determined by

$$\sum_{k=1}^{m}(\Delta r_k)\times C_k, \tag{6.5}$$

where m is the number of links in the network, Δr_k is the bandwidth of link k the new QoS requires, and C_k is the total amount of bandwidth currently being utilized in link k.

The calculation does not consider the amount of capacity available on the link, and actually penalizes the upgrading of inter-network SLAs, due to their higher total capacity for this test. The bias towards intra-network SLAs is compounded by the fact that the average hop length for inter-network SLAs in this network is 3.7, while only 2.6 for intra-network SLAs.

P-HEU does not suffer the same bias as D-HEU toward intra-network SLAs. This results from the large average bandwidth per SLA (3.5Mbps) and the low link capacity in P-HEU; each controller views the capacity on each link as 25Mbps. The efficiency of P-HEU is lower than D-HEU with such a high average bandwidth per SLA, resulting in the inability to admit the 31% of intra-network SLAs as D-HEU.

Finally, A-HEU and FA-HEU have a large bias towards inter-network SLAs. This has been mentioned throughout this chapter. The bias is most notable between controllers that control a large number of external links, and controllers that control few external links. Since an external link connecting two networks is controlled by the lower numbered network, DSC_4 in our tests controls no links. Conversely, DSC_1 has control over 5 external links. At high contention, DSC_1 admitted only 9.8% of the inter-network SLAs, whereas DSC_4 admitted 38.2% of them.

6.5.7 Network Size

As many of the results have been completed using a single network configuration, the following tests compare the heuristics using pseudo-randomly created networks of increasing size.

We created networks with 10, 15, 20, 25, and 30 nodes. Each network was created randomly with $LinksPerNetwork = 1.5 \times NodesPerNetwork$. These networks were then replicated 3 more times each, and interconnected with their replicas to form interconnected networks consisting of 4 networks. The networks were interconnected pseudo-randomly with 10 links; however, the bandwidth on the external links was proportional to the number of nodes in each network, as shown in Table 6.3.

Nodes / Network	Links / Network	External Link Bandwidth	Total Bandwidth	Average Hop Length	SLAs per DSC to cause contention (low / high)
10	15	100 Mbps	7 Gb	3.4	181 / 315
15	22	150 Mbps	10.3 Gb	3.5	262 / 444
20	30	200 Mbps	14 Gb	4.1	297 / 515
25	37	250 Mbps	17.3 Gb	4.7	324 / 561
30	45	300 Mbps	21 Gb	5.3	338 / 594

Table 6.3: External link bandwidth various sized interconnected networks.

Table 6.3 also shows the total bandwidth in the system and the average hop length of the SLAs used in the test. The number of SLAs to cause contention is the average number of SLAs required by each controller to cause contention of 0.7 or 1.2. Note that the average hop length affects the number of SLAs required to cause contention. Thus, the interconnection of 30-node network does not require 3 times more SLAs than the interconnection of 10-node networks.

[*LocalityOfReference*=0.8, *BandwidthRange*=(1Mbps, 3Mbps)]

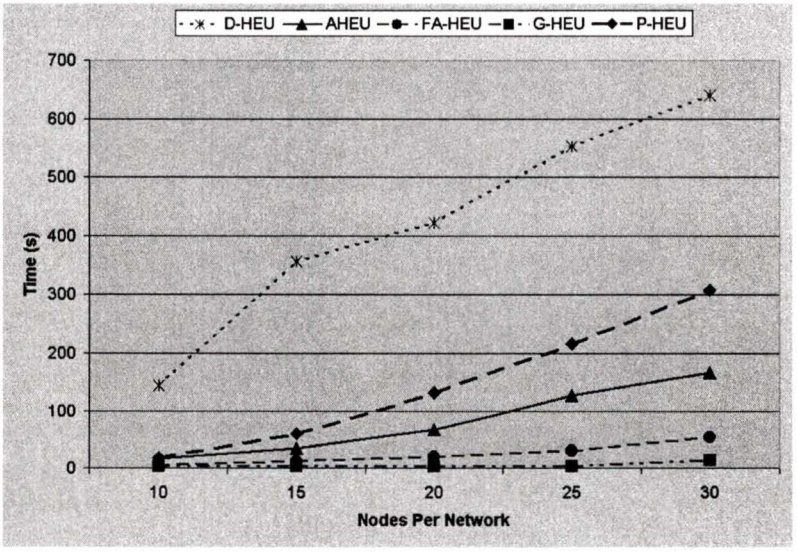

Figure 6.20: Time comparison of heuristics with networks of increasing size at low contention.

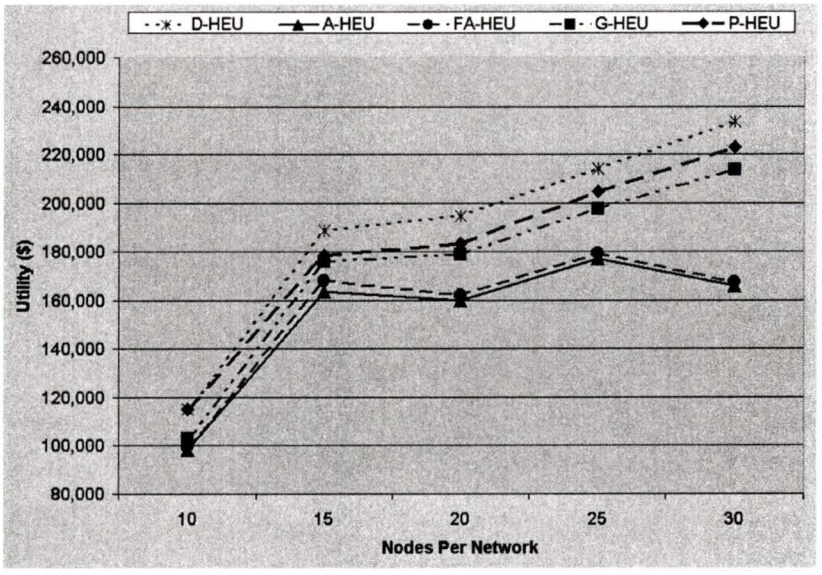

Figure 6.21: Utility of heuristics with networks of increasing size at low contention.

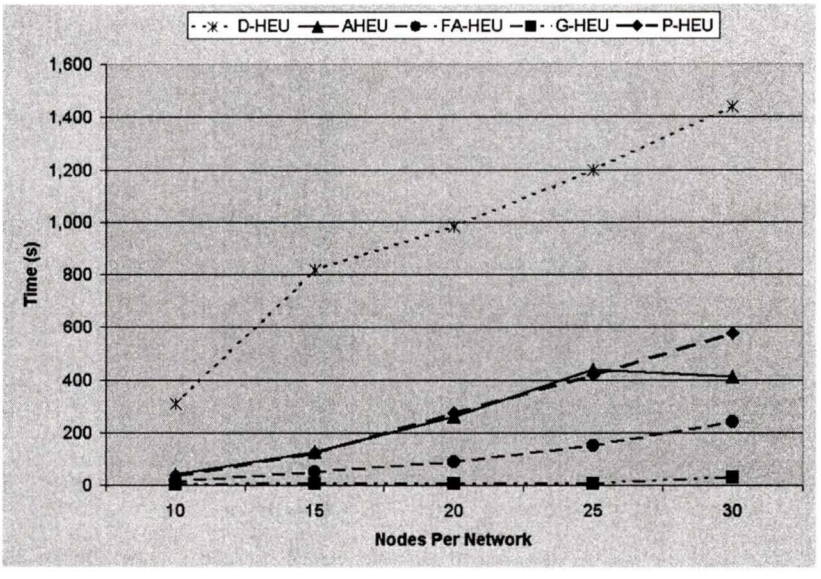

Figure 6.22: Time comparison of heuristics with networks of increasing size at high contention.

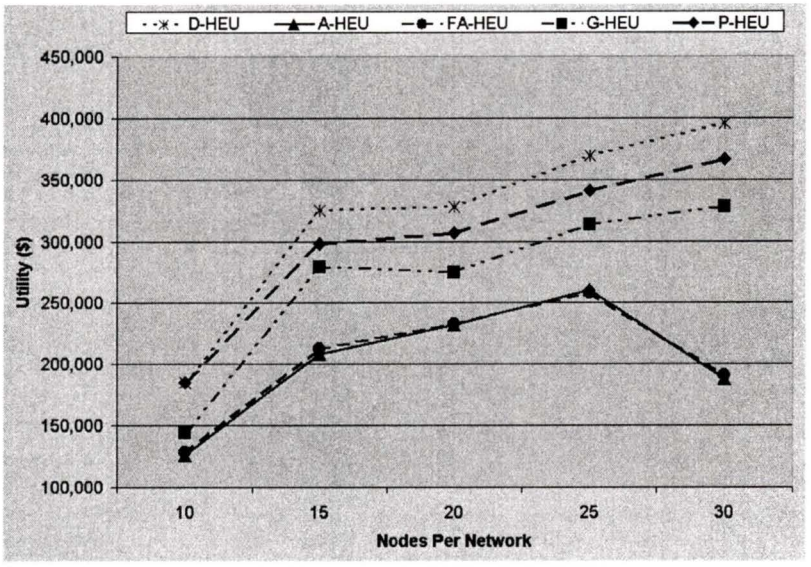

Figure 6.23: Utility of heuristics with networks of increasing size at high contention.

It is difficult to compare the utility derived from each of these networks. As the networks were created randomly, some networks are *better* connected than others, thus allowing more SLAs to be admitted, and more utility to be derived. For example, the 10-node network is a poorly connected network, while the 15-node network is a well-connected network.

At both low and high contention, the time to execute P-HEU quickly increases as the number of nodes per network increases. This is due the number of resources each controller in P-HEU must control. P-HEU uses the I-HEU heuristic to solve the MMKP, which has up to 180 resources in this test – 45 links/network x 4 networks. Remember that P-HEU can be implemented with any MMKP heuristic, and thus the scalability of P-HEU in this test, with respect to the number of resources, could be overcome by utilizing another MMKP heuristic.

6.5.8 Number of Networks

Next we chose to test the scalability of the system, with respect to the number of networks controlled. We used sets of the basic 9-node networks and pseudo-randomly interconnected them with $3 \times m$ links, where m is the number of networks being

interconnected. The capacity on all links was 100Mbps. Admission was tested at both low and high contention.

As seen in Figure 6.25 and Figure 6.27, P-HEU is more sensitive to the number of networks, with respect to utility, than D-HEU. P-HEU, when only 2 interconnected networks are considered, generates within 1% of the utility of D-HEU. When 5 interconnected networks are considered, P-HEU generates 93% of the utility of D-HEU at low contention, and within 89% of the utility of D-HEU at high contention.

This performance was expected, as P-HEU assumes that the average SLA bandwidth is considerably smaller than the link bandwidth. In P-HEU, SLA controllers view the link bandwidth as $1/m^{th}$ of their actual size. At $m = 5$, each SLA controller views the link bandwidth as 20Mbps, which is not considerably greater than the average SLA bandwidth.

[*BandwidthRange*=(1.5Mbps, 3.5Mbps), *LocalityOfReference*=0.8]

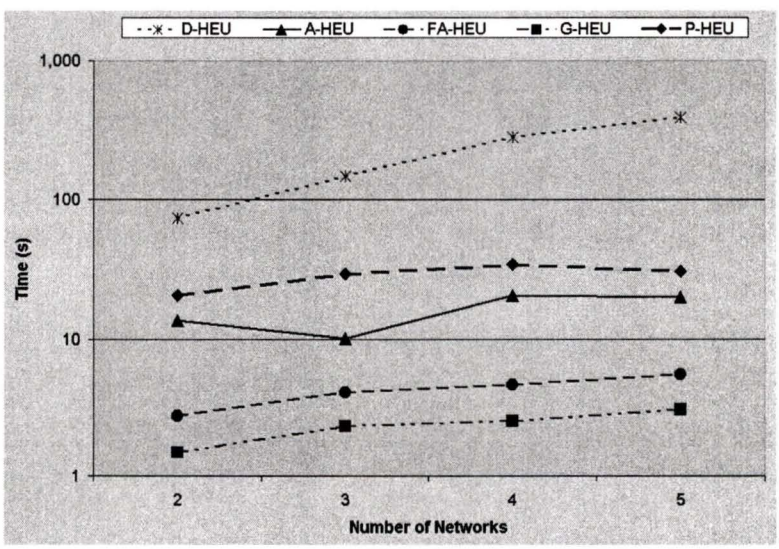

Figure 6.24: Time comparison of heuristics on a varying number of interconnected networks at low contention.

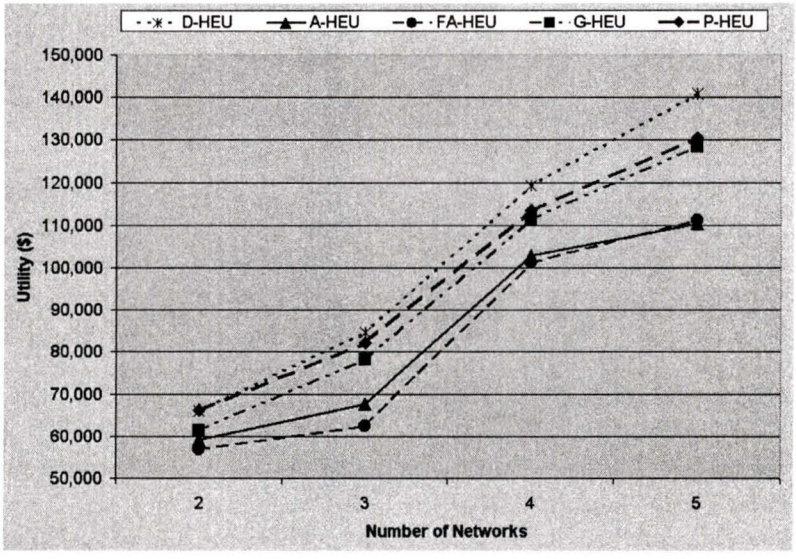

Figure 6.25: Utility of heuristics on a varying number of interconnected networks at low contention.

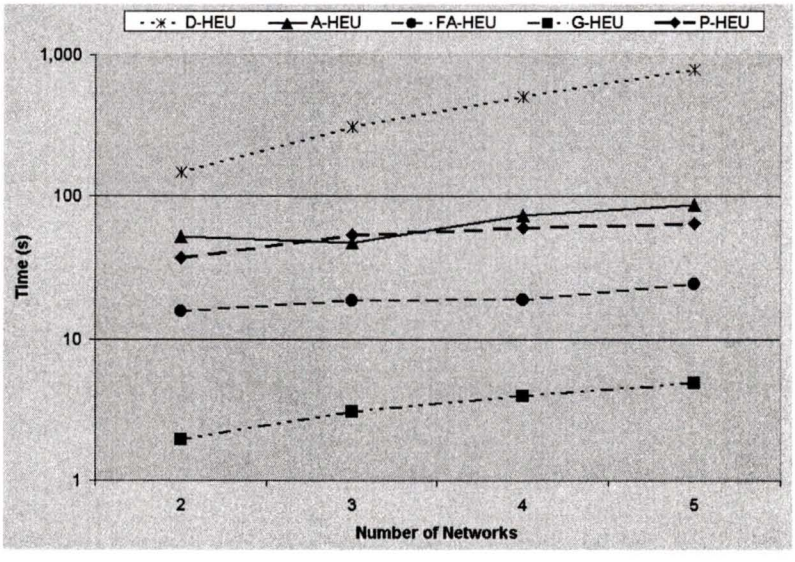

Figure 6.26: Time comparison of heuristics on a varying number of interconnected networks at high contention.

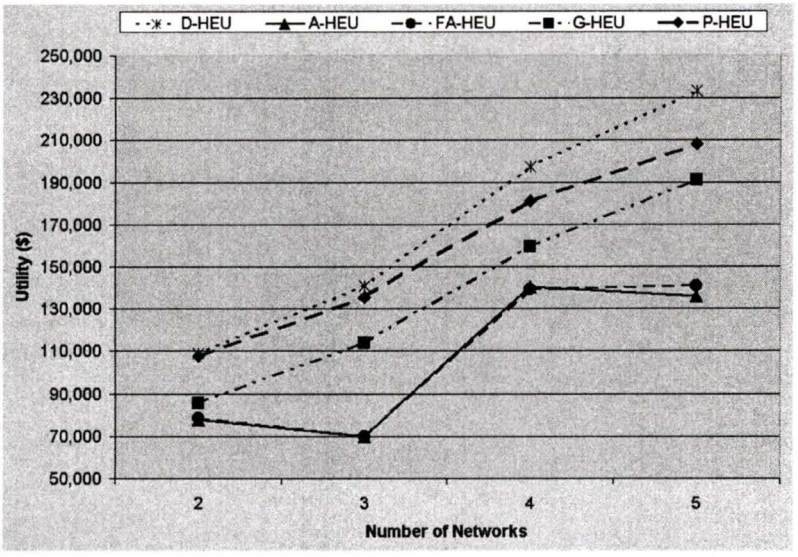

Figure 6.27: Utility of heuristics on a varying number of interconnected networks at high contention.

6.6 Improving the Arbitration Heuristic

As can be seen in the previous results, A-HEU and FA-HEU perform rather arbitrarily. This is due to the nature of the heuristics themselves. Consider a merged list of proposed QoS upgrades in A-HEU (or FA-HEU) (see Figure 6.28).

SLA:	SLA 20	SLA 1	SLA 9	SLA 33	. . .	SLA 4
Current QoS → Proposed QoS:	2→3	4→2	null→3	2→1		1→4
Proposed Path:	Path 1	Path 3	Path 2	Path 1		Path 2

Figure 6.28: Structure of a Proposed QoS Upgrades list.

In A-HEU, all controllers upgrade from the start of the list until one controller cannot reserve the necessary local resources. This early termination of the search is the cause of major inefficiencies, as an SLA that cannot be admitted results in other SLAs being overlooked in the admission process. As an alternative, consider a controller upgrading SLAs from the proposed QoS list; however, when a controller cannot upgrade an SLA, the SLA is simply bypassed, and the controller attempts to upgrade the remaining SLAs. We now show how to do this with minimal message passing.

Each controller creates a *binary local* feasibility list equal in length to the proposed QoS list. A 1 in the list means that the controller can successfully reserve the local resources to upgrade the respective SLA in the proposed QoS list. A 0 in the list means that the controller cannot successfully reserve the local resources to upgrade the respective SLA in the proposed QoS list. We call this binary string the *marking* of the list. An example of a binary local feasibility list created by an SLA controller is shown in Figure 6.29.

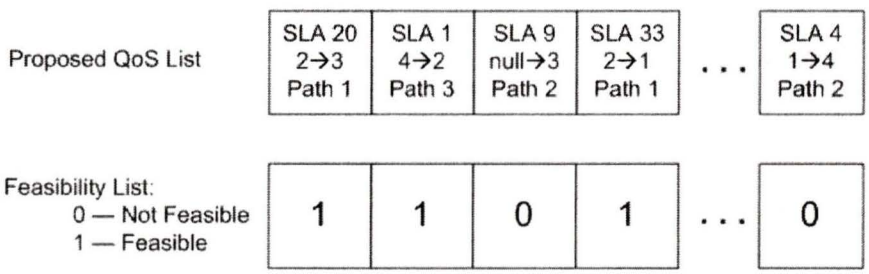

Figure 6.29: Example feasibility list for the given Proposed QoS list.

Upon creating the feasibility list, based only on local resources, the controllers send their list to all other controllers. Each controller can now compare the ability of all the controllers to upgrade each SLA. Each controller now determines which upgrades all the controllers can support, by taking the component-wise AND of the markings in each of the feasibility lists to yield a Global Feasibility List. The selected SLAs are then upgraded. An example Global Feasibility List is shown in Figure 6.30.

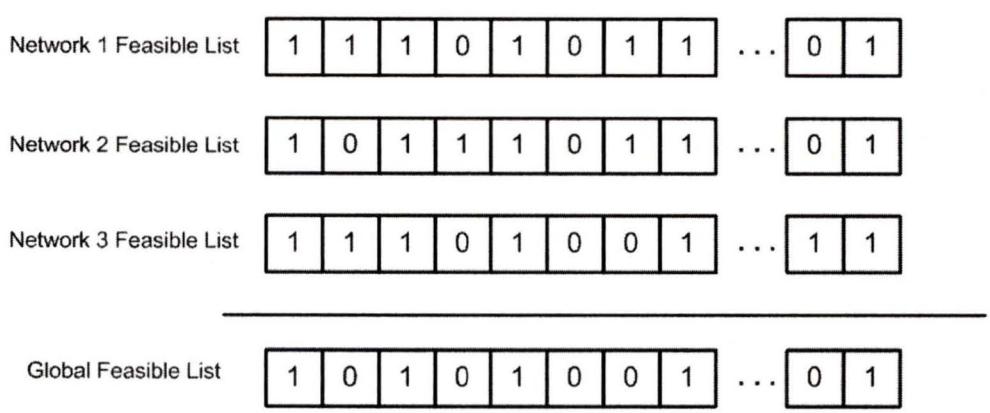

Figure 6.30: Global Feasibility List from individual network feasibility lists.

As with the original A-HEU, this modified version of A-HEU is repeated to improve optimality. The protocol described above can be used to modify either A-HEU or FA-HEU. The changes to these heuristics are referred to as MA-HEU (modified A-HEU) and MFA-HEU (modified FA-HEU). The comparison of this modified arbitration heuristic to the original A-HEU, and even FA-HEU, are shown in Figure 6.31, Figure 6.32, and Figure 6.33.

The modifications made to A-HEU and FA-HEU resulted in more consistent, and higher, values of earned utility. MA-HEU consistently outperforms MFA-HEU, but only by 2.4% at contention of 0.7 to 1.4% at contention of 1.4. The cost for this modest gain in utility is significant, as MA-HEU was between 7 and 10 times slower than MFA-HEU in our tests.

[*LocalityOfReference*=0.8, *BandwidthRange*=(2Mbps, 5Mbps)]

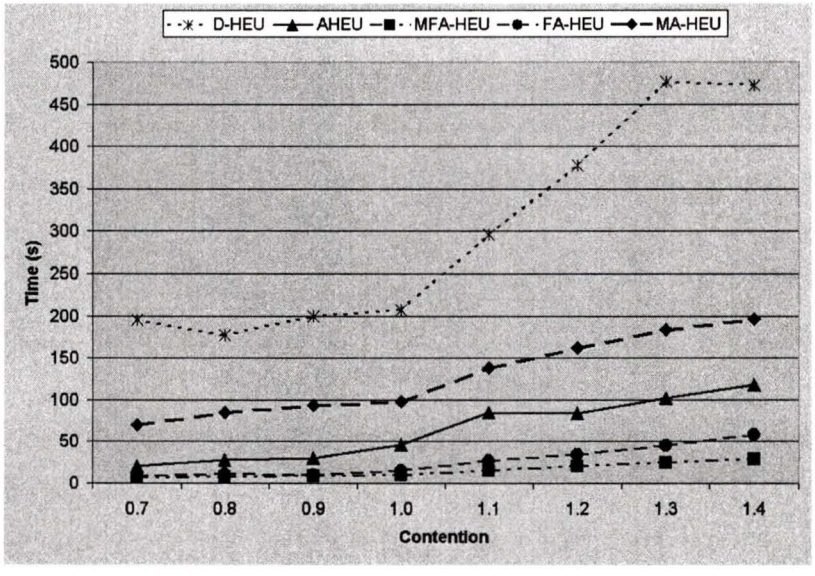

Figure 6.31: Time comparison of modified heuristics against varying contention.

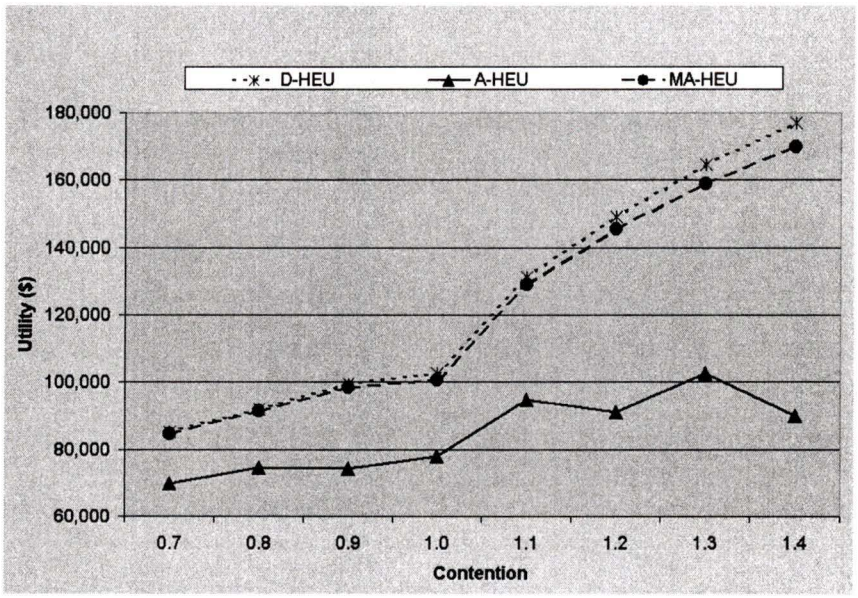

Figure 6.32: Utility comparison of A-HEU and MA-HEU against varying contention.

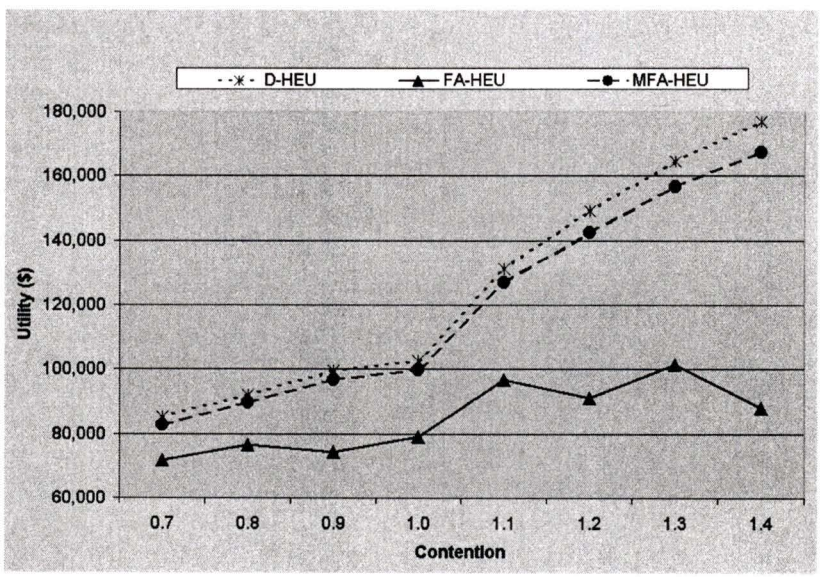

Figure 6.33: Utility comparison of FA-HEU and MFA-HEU against varying contention.

The other interesting result is that MA-HEU was slower than A-HEU, but MFA-HEU was faster than F-HEU. MA-HEU is slower than A-HEU due to a combination of two factors. First, MA-HEU was executed more times per epoch than any of the other heuristics (these four heuristics are repeated until no gain in utility achieved), as shown in Table 6.4. The second factor is the time per epoch required to execute step 3 of I-HEU in MA-HEU.

Heuristic	A-HEU	MA-HEU	FA-HEU	MFA-HEU
Times Executed	2.5	4.2	2.5	3.3

Table 6.4: Times to execute each heuristic.

MFA-HEU was faster than the original F-HEU as a result of the improved earned utility. More SLAs were upgraded per epoch in MFA-HEU, resulting in fewer options for upgrading during the subsequent execution of the heuristic, and thus an increasingly lower execution time per epoch.

7 Application of Optimal SLA Admission

To demonstrate the application of distributed SLA controllers, a real example illustrating the use of distributed SLA controllers is helpful. This chapter first outlines a unified model for optimal admission and adaptation as applied to both interconnected networks and server farms, such that a user is guaranteed both network resources and server resources – both guarantees are necessary to assure the delivery of a multimedia stream with a particular QoS. We then finish by describing how to use the unified model to deliver video on demand over an ISP's network.

7.1 Unified Optimal Admission Model

To guarantee the delivery of a multimedia stream to an end-user, both the network resources and the multimedia servers' resources need to be reserved. The Unified Optimal SLA Admission Model expands the theory of optimal SLA admission to include the joint reservation of both network and server resources for users, in order to maximize the revenue of the service provider.

We assume the service provider is both the network provider and the content provider, at least conceptually. The content provider may simply have a revenue sharing agreement with the network provider. Such a case would exist if Blockbuster Inc. setups a video-on-demand service on a Shaw or Telus network.

7.1.1 Service Level Agreement for the Streaming of Stored Content

Akbar's work [8] demonstrated how to perform optimal admission and adaptation of SLAs by server farms using MMMKP heuristics. In such a context, the server farm is analogous to a knapsack. The server farms may be connected to one another by an interconnection network, which Akbar assumed to have infinite bandwidth and zero

delay, in order to eliminate the influences of the network's resources from the problem. Each server farm contains numerous resources: the memory, the CPU cycles, and the I/O bandwidth of the individual servers of the farm. Note that the resources are not required to be similar, as they are in the case of admission into interconnected networks, where all resources were bandwidth on network links.

The definition of an SLA in the Unified Optimal SLA Admission Model may include the definition of multiple QoS levels, as was done within the context of interconnection networks. However, since more than one server may be capable of serving a specific stream, the SLA is redefined from Section 2.2.1 to

(source, { QoS Level }).

The *source* is the address of the user, the source of the request, and each QoS Level is defined by the tuple

(bid, { streams }).

The *streams* are the set of multimedia streams required to deliver a QoS level. For example, an SLA with three defined levels of QoS to view the movie ET may be defined by the SLA definition in Figure 7.1.

Source	**Bid, {List of Streams}**	
142.123.456	$1, {ET.MonoAudioStream, ET.BlackAndWhiteVideoStream}	QoS 1
	$2, {ET.StereoAudioStream, ET.ColourVideoStream}	QoS 2
	$5, {ET.SurroundAudioStream, ET.HighDefinitionVideoStream}	QoS 3

Figure 7.1: Example SLA to view the movie *ET*.

7.1.2 Unified Distributed Optimal Admission Controllers

A distributed optimal SLA admission controller provides two major functions:

1. Map an SLA request to system resources, and
2. Execute an MMMKP heuristic to perform admission and adaptation.

An MMMKP heuristic does not differentiate between the types of resources being reserved, thus allowing one controller to manage link capacities in a network, while another controller manages resources in a server farm. An area controlled by a distributed SLA controller, conceptualized as a knapsack, can therefore be an entire server farm, or a network. It is possible for a single controller to manage both link capacities and server resources, although that case will not be considered in this thesis.

To unify the treatment of server and network admission problems, there needs to exist two SLA mappings in the controller, one for mapping the SLAs to server resources, and one for the mapping to network resources: *kBestServers()* and *kShortestPaths()*, respectively.

The mapping of the streams of a video to the resources required is done by the admission controller through a function called *kBestServers()*. *kBestServers()* takes the required components as input and returns, for each of the streams, the identities of k servers which contain the stream, the resources required to deliver the stream (bandwidth, cpu cycles, etc), and the delay constraint (if any). Here the k best servers are not determined by server usage, as the heuristic determines scheduling, but are determined by more static parameters, such as proximity to the source.

The architecture of the unified admission controller can be seen in Figure 7.2.

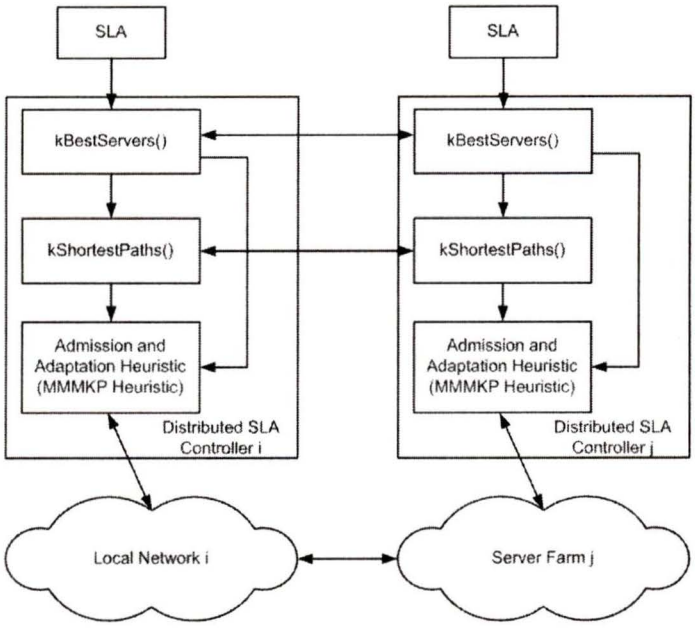

Figure 7.2: Architecture of a Unified Distributed Optimal SLA Admission and Adaptation Controller

An SLA is submitted to a DSC, where it is first passed to the *kBestServers()* function, which determines, by coordinating with other DSCs, which are the best servers to supply each of the requested streams. At this point, server load is not considered in choosing the best servers. *Best* servers may be chosen simply by their proximity to the client. The actual addresses of these servers are forwarded to the *kShortestPaths()* function, along with the requester's address, to determine the *k candidate paths* from the user to each of the possible servers.

Besides identifying the best servers to serve the client, the *kBestServers()* function also provides the admission heuristic with the amounts of resources within each of these servers that are required to deliver each stream component. The *kShortestPaths()* function similarly provides the admission heuristic with the identities of necessary resources (paths) to route the stream from a particular server to the client.

The admission heuristic now has a list of resources, both server and network, required to deliver a multimedia stream to the user. It is now simply a matter of executing the heuristic to determine by which servers, and over which path, the multimedia stream should be delivered. The processing of an SLA is illustrated in Figure 7.3.

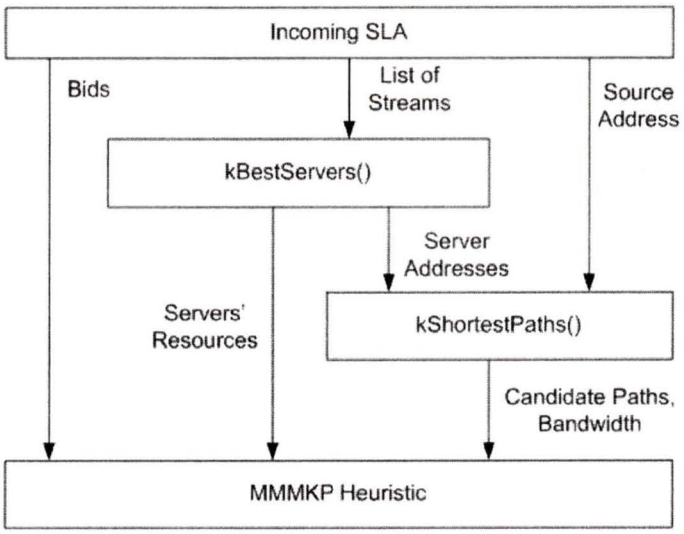

Figure 7.3: Processing of an SLA through a distributed SLA controller.

An example of a QoS Profile created from the SLA in Figure 7.1, after receiving the relevant information from *kBestServers()* and *kShortestPaths()*, can be seen in Figure 7.4.

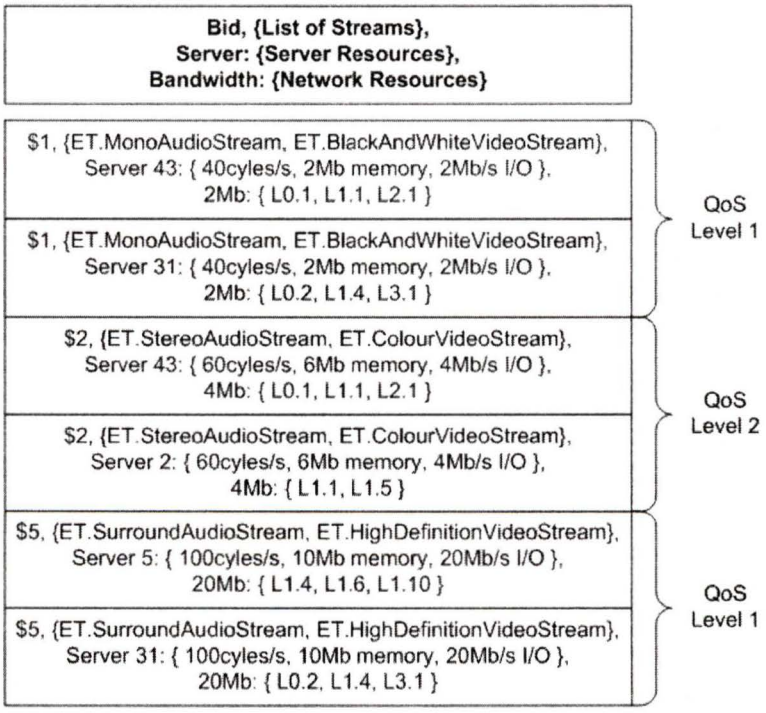

Figure 7.4: QoS profile as viewed by an admission heuristic.

Note that the complexity of a QoS profile has increased dramatically compared to a network-only profile.

The number of sub-QoS levels seen by the heuristic is now $k_h = (k_s)^c k_p q_r$, where k_s is the number of servers returned by *kBestServers()*, c is the number of components in a stream, k_p is the number of paths returned by *kShortestPaths()*, and q_r is the number of QoS levels in the original requested SLA. Thus, if $k_s = k_p = 3$ ($k_p = 3$ was typical in the non-unified model), $q_r = 3$, and $c = 2$, then the total number of sub-QoS levels (K_h) is 81, whereas in the network admission problem, $k_h = k_p q_r = 3 \cdot 3 = 9$. The extra factor comes from selecting 3 servers for each of the 2 components in a multimedia stream.

To control the resulting computational complexity we could restrict ourselves to smaller k_s, k_p, and c values. For example, we could set $k_s = k_p = 2$, and let the user set at most 2 QoS levels ($q_r = 2$). Now $k_h = 2^2 \cdot 2 \cdot 2 = 16$, when $c = 2$, or $k_h = 2^1 \cdot 2 \cdot 2 = 8$ when $c = 1$, which is comparable to the complexity as seen by considering network resources only. Admittedly, more servers and paths may need to be considered to improve optimality to a desired level, but that will depend on the specific network and services in question.

7.2 Applying the Unified Admission Model: Video on Demand

Telephone companies are trying to compete with cable companies by delivering broadcast network television over their DSL lines. They also want to deliver Video on Demand (VoD) [18], to undercut the cable companies and to increase potential revenue. Similarly, the cable companies want to deliver DVD quality VoD, network TV, and telephone services to their customers over their networks.

Our model does not presently incorporate multicast, so we do not model broadcast network TV, but video on demand (VoD) with VCR controls is a natural use of our model. VCR controls allow a user to control the video like a VCR; allowing the pause, rewind, and fast-forward of video [19]. Considering our application, and small local loop

capacities (typically a few Mb/s for DSL and cable), it is reasonable to assume unicast delivery of the video to allow VCR controls [20].

The other potential limitation of our model for delivering VoD is that our model assumes static traffic flows. To guarantee QoS, a user has to reserve the peak bandwidth of their session. We can, however, reduce the peak bandwidth of video with *constant bit rate smoothing*. Although video typically has a highly variable data rate, there has been considerable research on how to smooth its data rate to a constant data rate. Variable bit rate video, such as MPEG, can be transmitted at a constant rate by use of a client buffer [21], by use of various on-line smoothing algorithms [22] [23], by dropping frames at peak periods [24], or by initially compressing the video for constant-bit-rate transmission.

8 Conclusion

The concept of utilizing an admission controller based on a combinatorial problem, such as the knapsack problem, is intriguing from a mathematical point of view. The running times of the heuristics evaluated so far, however, probably do not allow for the on-line admission of non-aggregated users' sessions in an ISP's network. However, as Section 8.2.5 notes, perhaps the model of admission control described within this thesis is not well suited for on-line admission.

It should be emphasized that the running time of the heuristics in Chapter 6 can be greatly improved by using faster processors, utilizing parallel processing, and recoding in a more efficient language than Java. Each of the heuristics, except for D-HEU, used very few messages, and thus communication is not a major bottleneck to performance. With dSLAOpt, using the data for contention of 0.9 in Section 6.6, for 4 9-node networks, MFA-HEU can adapt over 600 SLAs (10% newly admitted) in less than 10 seconds. It is quite conceivable that this calculation could be performed in less than 1 second, utilizing the techniques described above.

This chapter summarizes the major contributions of this thesis and presents directions for future work.

8.1 Contributions

dSLAOpt, a distributed optimal SLA admission and adaptation controller was simulated and the results were presented in this thesis. A new heuristic called P-HEU (parallel-heuristic) was designed using concepts of parallel processing to solve the admission problem, providing a unique solution to the admission problem. dSLAOpt was then utilized to test and compare the various heuristics under a variety of conditions. The analysis of these test results led us to propose MFA-HEU (modified fast A-HEU), an extremely fast heuristic that yields utility second only to D-HEU. Finally, we presented

the Unified Admission Model for reserving both network and multimedia server resources, to guarantee the QoS for admitted multimedia streams in an ISP's network.

The above contributions are summarized in the subsections below.

8.1.1 Simulated Distributed SLA Admission Controller

Admission control is required for the guarantee of absolute QoS for multimedia streams. dSLAOpt, a simulation of a distributed optimal SLA admission controller, was implemented to evaluate numerous heuristics, most of which were developed by Akbar [8], for the admission and adaptation of SLAs for multimedia sessions.

8.1.2 Parallel Heuristic

In the other heuristics, a DSC sees only one of the interconnected networks. P-HEU, however, allows each DSC to see all of the interconnected networks, but each DSC only sees and controls $1/m$ of the bandwidth on each link, where m is the number of networks (knapsacks). Each DSC now manages a selected number of SLAs, which a DSC admits or rejects by using its pre-allocated share of resources over all of the interconnected networks. Instead of collaborating on every decision, the DSCs now work independently, in parallel, and then share results for the building of the final circuits.

8.1.3 Optimal SLA Admission and Adaptation Heuristics

The concept of using the Utility Model-Distributed for optimal SLA admission and adaptation, as used in dSLAOpt, appears promising with the proposal and modification of new heuristics. The greedy heuristic promotes low message passing, allowing for extremely fast admission of heuristics, while still achieving high utility; although at a cost of fairness in admission of inter-network SLAs. Intra-network SLAs are highly favoured by the greedy heuristic, which may or may not be a drawback, depending on the intended application and business environment.

The use of D-HEU as an admission heuristic is limited to off-line processing, and even then its time consumption is so great that it will have limited applicability. P-HEU allows the distributed admission problem to be solved with little message passing, while still achieving utility close to that of D-HEU.

The first modification to A-HEU, named FA-HEU (fast A-HEU) was shown to perform much faster than A-HEU and with little or no utility tradeoff. Both A-HEU and FA-HEU achieved the lowest utility in most scenarios, in comparison to the other heuristics. Although FA-HEU was fast, and proved to be fairer to inter-network SLAs than G-HEU, A-HEU and FA-HEU performed inconsistently, due to their arbitration protocol, which prematurely terminated the search for SLAs to upgrade or admit.

8.1.4 Proposed Modification to A-HEU and FA-HEU

After evaluating the test results, modifications to A-HEU (and FA-HEU) were implemented, which we named modified A-HEU (MA-HEU) and modified FA-HEU (MFA-HEU). MFA-HEU yielded marginally less utility than MA-HEU in only a fraction of the execution time. In our test results, MFA-HEU and MA-HEU generated more utility than all other heuristics, except D-HEU, while MFA-HEU was faster than all heuristics, except G-HEU.

8.1.5 Unified Admission Model

After comparing the heuristics, we explored the possibility of utilizing distributed SLA controllers to deliver video on demand within a telephone or cable company's network. The Unified Optimal SLA Admission Model was proposed, to coordinate and guarantee the reservation of resources of the network, as well as that of multimedia servers, for admission and adaptation.

Such a unified model would have application to a video on demand service. It is valuable because multimedia server resources as well as network resources need to be reserved to guarantee overall QoS as seen by the customer.

8.2 Future Work

8.2.1 Multicasting

The most severe limitation of the current model for Optimal SLA Admission is that it only supports unicasting, and not broadcasting or multicasting. Our model is designed for multimedia traffic, which is often multicast by nature. Multicasting has the ability to reduce required network bandwidth and thus increase network utility. For example, consider an eLearning application, where a teacher multicasts her lecture live to distant learning students. Or consider a new movie release from Blockbuster Inc. distributed over a network. If several people watch a movie at almost the same time, it is an inefficient use of network resources to unicast the movie to each viewer.

8.2.2 Handling Non-CBR Traffic

In the current model a session is considered to be of constant bandwidth. However, multimedia traffic is often bursty by nature. To guarantee QoS, the peak bandwidth for a session has to be reserved. Such a reservation could actually be unnecessarily expensive for a user, and result in a lot of wasted bandwidth.

To deal with this problem, other parameters during the admission process could be included to outline the expected requirements of the multimedia stream over the duration of its lifetime. The DSC could then reserve more link bandwidth than is available, but ensure that, within a specified confidence interval, the total bandwidth of a particular link will not be over-utilized at any point in time. Users could be charged, not necessarily by their peak bandwidth, but by a function which includes their peak, minimum, as well as average expected throughput rates.

8.2.3 Scalability

Scalability is a concern of our model for Optimal SLA Admission. As internets grow, the current model may become infeasible for a given organization. This research also has no

possibility for being used to setup sessions across the entire Internet. A new approach would need to be investigated for such a venture.

8.2.4 Bandwidth on Demand

Bandwidth on demand in optical networks allows a customer to increase its capacity across specific links as required, at a price. It is reasonable to extend the current model of optimal SLA admission and adaptation to include the purchasing/selling of bandwidth as required by the system. An SLA should not be rejected if bandwidth can be immediately bought at a low enough price to yield a net profit from admitting the SLA.

8.2.5 Applicability of Heuristics

An efficient implementation of a simulated SLA controller and its associated heuristics is necessary to get a true understanding of the load of SLAs these heuristics can handle. Also, most of the heuristics, with the exception of D-HEU, should benefit greatly through the use of parallel processing.

Understanding the number of SLAs these heuristics can handle is necessary to better understand the suitability of distributed SLA controllers for core or edge networks, and whether they are only suitable to manage aggregated flows instead of non-aggregated flows, which would be ideal.

Also of interest is the manner in which real users would use the system. We have assumed that users place their true value for a particular level of QoS in an SLA. However, as the length of epochs is reduced, so is the opportunity cost of being rejected. Users have less motivation to advertise their true values of an SLA, as they could simply readmit another SLA with higher values in the next epoch. Overly responsive online admission could therefore reduce the utility yielded by the system, especially for lightly loaded networks. However, it does introduce another interesting, and perhaps beneficial, bidding mechanism.

References

[1] S. Shenker, C. Partridge, R. Guerin. Specification of Guaranteed Quality of Service, *RFC 2212,* September 1997.

[2] *HP Broadband / Content delivery over broadband*, Solution datasheet, HP Intel Solution Centre, http://www.hpintelco.com/pdf/broadband_inktomi_datasheet.pdf.

[3] *End-to-End QoS Solutions*, http://www.gi.com/noflash/ipns_qos.html, Motorola Incorporated, 2002.

[4] *Marconi Introduces Intelligent Packet Networks Solution to Enable New Multimedia Service Offerings*, Marconi news release, June 2000.

[5] *Differentiated Services: Moving towards Quality of Service on the Ethernet*, Intel Corporation, 2002.

[6] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering over MPLS. *RFC 2702*, September1999.

[7] R. Braden, L. Zhang , S. Berson , S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification. *RFC 2205*, September 1997, Proposed Standard.

[8] M.M. Akbar. *The Distributed Utility Model Applied to Optimal Admission Control & QoS Adaptation in Multimedia Systems and Enterprise Networks*. Ph.D. thesis, Department of Computer Science, University of Victoria, 2002.

[9] M.S. Khan. *Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture*. Ph.D. thesis, Department of Computer Science, University of Victoria, 1998.

[10] R.K. Watson. *Applying the Utility Model to IP Networks*. M.S. thesis, Department of Computer Science, University of Victoria, 2001.

[11] M.M. Akbar, E.G. Manning, R. Watson, G.C. Shoja, S. Khan, and K.F. Li. Optimal Admission Controllers for Service Level Agreements in Enterprise Networks. *Sixth World Multiconference on Systemics, Cybernetics, and Informatics*, Orlando, Florida, July 2002.

[12] D. Eppstein. Finding the k Shortest Paths. *35th IEEE Symp. Foundations of Computer Science*, Santa Fe, Nov 1994, pp. 154-165.

[13] J. Moy. OSPF Version 2. *RFC 2328*, April 1998.

[14] S. Niar and A. Freville. A Parallel Tabu Search Algorithm for the 0-1 Multidimensional Knapsack Problem. *11th International Parallel and Distributed Processing Symposium*, April 1997, pp. 512-516.

[15] A. Ferreira and J.M. Robson. Fast and Scalable Parallel Algorithms for Knapsack-Like Problems. *Journal of Parallel and Distributed Computing*, vol. 39, no. 1, Nov 1996, pp. 1-13.

[16] V.N. Alexandrov and G.M. Megson. *Parallel Algorithms for Knapsack type Problems*. World Scientific Publishing, 1999.

[17] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.

[18] Tellicent Incorporated. *Video Over DSL: An Opportunity for Independent Telephone Companies*. White Paper, 2003.

[19] Net to Net Technologies. *Video over IP DSL*. White Paper, October 2002.

[20] C. Lin, J. Zhou, J. Youn, and M. Sun. MPEG Video Streaming with VCR-Functionality. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 11, Issue 3, March 2001, pp. 415-425.

[21] S.G. Rao and S.V. Raghavan. Fast Techniques for the Optimal Smoothing of Stored Video. *ACM/Springer Verlag Multimedia Systems Journal*, Vol 7, May 1999, pp. 222-233.

[22] S. Sen, J.L. Rexford, J.K. Dey, J.F. Kurose, and D.F. Towsley. Online Smoothing of Variable-Bit-Rate Streaming Video. *IEEE Transactions on Multimedia*, Vol 2, No 1, March 2000, pp. 37-48.

[23] G. Cao, W. Feng, and W. Singhal. Online VBR Video Traffic Smoothing. *Eighth International Conference on Computer Communications and Networks*, Boston, USA, 1999, pp. 502-507.

[24] J.K. Ng and S. Song. A Video Smoothing Algorithm for Transmitting MPEG Video Over Limited Bandwidth. *Fourth International Workshop on Real-Time Computing Systems and Applications*, Taipei, Taiwan, October 1997, pp. 229-236.

Vita

Surname: Shelford Given Names: Steven John Roy

Place of Birth: Burns Lake, British Columbia, Canada

Educational Institutions Attended:

University of Victoria	1995 – 2003

Degrees Awarded:

B.Sc.	University of Victoria	2001

Honors and Awards:

NSERC Industrial Postgraduate Scholarship	2001 - 2003
Nortel / NewMIC Scholarship	2001 – 2003
University of Victoria President's Scholarships	2001 – 2003
ASI Graduate Scholarship	2001

Publications:

[1] S. Shelford, M.M. Akbar, E.G. Manning, and G.C. Shoja. Distributed Optimal Admission Controllers for Service Level Agreements in Interconnected Networks, *Proceedings of the 21st IASTED International Conference on Applied Informatics*, February 2003, Innsbruck, Austria.

University of Victoria Partial Copyright License

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

Implementation, Evaluation, and Application of Distributed Heuristics for Optimizing SLA Admission

Author 

Steven John Roy Shelford

August 7th, 2003