Periodic Data Structures for Bandwidth-intensive Applications

by

Ilijc Albanese
B.Eng., University of Roma Tre, 2005
M.Eng., University of Roma Tre, 2008

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

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Periodic Data Structures for Bandwidth-intensive Applications

by

Ilijc Albanese
B.Eng., University of Roma Tre, 2005
M.Eng., University of Roma Tre, 2008

Supervisory Committee

Dr. Thomas Edward Darcie, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Stephen W. Neville, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Sudhakar Ganti, Outside Member
(Department of Computer Science)

Supervisory Committee

Dr. Thomas Edward Darcie, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Stephen W. Neville, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Sudhakar Ganti, Outside Member
(Department of Computer Science)

ABSTRACT

Current telecommunication infrastructure is undergoing significant changes. Such changes involve the type of traffic traveling through the network as well as the requirements imposed by the new traffic mix (e.g. strict delay control and low end-to-end delay). In this new networking scenario, the current infrastructure, which remained almost unchanged for the last several decades, is struggling to adapt, and its limitations in terms of power consumption, scalability, and economical viability have become more evident.

In this dissertation we explore the potential advantages of using periodic data structures to handle efficiently bandwidth-intensive transactions, which constitute a significant portion of today's network traffic.

We start by implementing an approach that can work as a standalone system aiming to provide the same advantages promised by all-optical approaches such as OBS and OFS. We show that our approach is able to provide similar advantages (e.g. energy efficiency, link utilization, and low computational load for the network hardware) while avoiding the drawbacks (e.g. use of optical buffers, inefficient resource utilization, and costly deployment), using commercially available hardware.

Aware of the issues of large scale hardware redeployment, we adapt our approach to work within the current transport network architecture, reusing most of the hard-

ware and protocols that are already in place, offering a more gradual evolutionary path, while retaining the advantages of our standalone system.

We then apply our approach to Data Center Networks (DCNs), showing its ability to achieve significant improvements in terms of network performance stability, predictability, performance isolation, agility, and goodput with respect to popular DCN approaches. We also show our approach is able to work in concert with many proposed and deployed DCN architectures, providing DCNs with a simple, efficient, and versatile protocol to handle bandwidth-intensive applications within the DCs.

Contents

List of Figures

DEDICATION

To my Mother and Father, for making all this possible.

Chapter 1

Introduction

1.1 New scenario, old infrastructure

Over the last decade the current network infrastructure has seen a dramatic increase in traffic. Most of this traffic increase is due to the rapid proliferation of bandwidth-intensive applications (streaming, peer-to-peer, Video on Demand, Content Delivery Networks and so on) [1]. These applications require large amounts of data to be transferred over the network, often with stringent delay requirements. This type of traffic has been steadily increasing (Figure 1.1) becoming the dominant type of traffic in todays network [1].

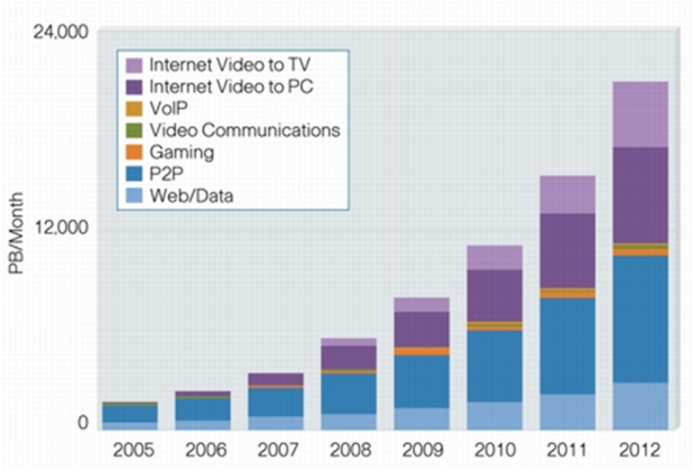


Figure 1.1: Significant changes occurred in the type of traffic traveling through the network. [Source: [1]]

In addition to the radical change in the traffic type, network providers also witnessed a dramatic reduction of the bandwidth costs over the last two decades (Figure 1.2). The combination of increased traffic load and reduced bandwidth costs will soon lead to a point in which the costs for network operators will surpass the revenues [2], posing serious limitation to the continued growth of Internet-based applications.

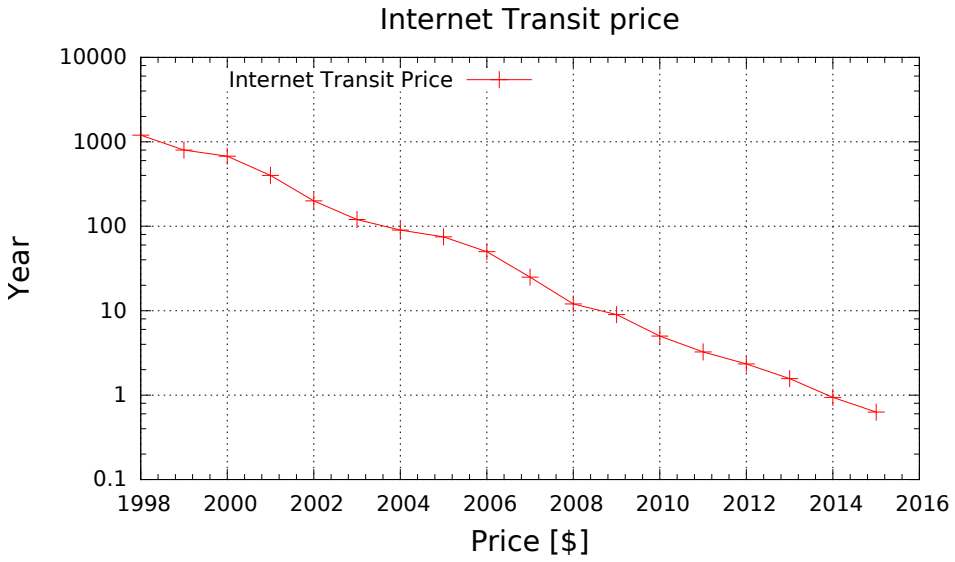


Figure 1.2: Internet transit price has reduced by several orders of magnitude over the last two decades. [Source: [3]]

In order to keep up with the rapidly changing network environment in an economically viable manner, ISPs should reduce the cost per bit transported and increase the capacity of their infrastructure. However, increasing the capacity of the infrastructure is expensive, and it is sensible to deploy new capacity only if the potential of the current infrastructure has been fully harnessed.

1.2 Internet power consumption problem

In the past the average user would access the Internet mainly for web browsing, generating a bursty kind of traffic that doesn't generally require a tight control of the delay, and enables ISPs to heavily oversubscribe their networks. In such a scenario an ISP could afford oversubscription rates of 24 or more [4] (the *Oversubscription rate* is defined as $(M - n)/n$, where M is the number of users connected and n is

the number of users that the network can support simultaneously at the peak access rate). Today, with the advent of multimedia applications such as P2P, IPTV,Video on Demand and so on, the bandwidth demand has changed: in order to support this new set of applications, a much higher and relatively constant bandwidth needs to be provided to each user. This will pose severe limitations on the ability to oversubscribe the network since such practice will result in poor performance.

Furthermore, in recent years, the rate at which IP traffic grows every year is around 50% or higher [5, 6]. This increase of traffic means a corresponding increase in the amount of equipment to be deployed in order to support such growth and of the power consumption of the network infrastructure which, at this rate, may soon become unmanageable [7, 8, 9]. Another aspect is that as access rates increase, more and more power will be consumed by core network routers [10, 11] which are usually concentrated in few buildings. Cooling all this hardware will be not only a challenging task itself but will also further increase the overall power consumption if it is assumed that for every *Watt* of power consumed, about the same amount is needed for cooling [7, 12]. Lastly, although not less important, is the carbon footprint of the network's infrastructure that this increased power consumption will produce together with the costs that all this energy will have for the companies and countries who operate the infrastructure [13, 14, 15].

1.3 A new player: Data Centers

Many bandwidth-intensive, Internet-based applications often rely on the services offered by data centers (DC). These computing facilities, host a large portion of the Internet-based applications available today, and handle a significant slice of the Internet traffic. As it will be discussed in some detail in Section 2.4, DCs are assembled by a large number of "cheap" servers interconnected by a high-capacity network infrastructure. The data center network (DCN) only accounts for a small portion of the entire cost ($10-15\%$ [16]), however, it plays a key role in determining the overall performance of the DC.

DCNs differ significantly from transport networks (e.g. WAN), and protocols that may work in one environment may not work as well in the other. As a result a plethora of protocols and architectures specifically designed for DCN have been proposed [17, 18, 19, 20, 21, 22, 23, 24]. Some of these protocols propose radical changes in the DCN architecture or topology-specific solutions (e.g. [25, 27, 26]), while

others (e.g. [28]) aim at modifying the dynamics of the underlying transport protocols to adapt them to the DC environment. The main differences between WAN and DCN can be summarized as follows: (1) DCNs have much smaller geographical extension (RTTs can be orders of magnitude lower), (2) the degree of statistical multiplexing is much higher in a WAN, while in DCNs is not uncommon for a single flow to dominate a particular path, (3) DCNs are usually under a single administrative domain, and as a result their physical topology is well know and relatively stable, and lastly (4) a DCN connects to the external network using load balancers and application proxies, making backward compatibility less of an issue in DCNs with respect to WANs.

In DCs a new set of issues has to be faced, issues which are either not present or less pressing in a WAN environment. One of such issues is the variability of network performance, which can result in higher costs for the DC tenants and can become a severe performance bottleneck (this is particularly true in map-reduce [29] clusters). Since, in a DC, tenants may be running their application(s) on the same physical machine with several other tenants, another important issue is assuring that the DC tenant are able to share the computing resources of the DC with other tenants with minimal or no impact on each others performance (*performance isolation* [26]). Furthermore, in order to optimize the utilization of the DC computing resources, it is also important that every service (e.g. tenant application) can be assigned to any machine, a property referred to as *agility* [30, 27]. With standard TCP, migrating a service to another physical machine may result in interrupting the ongoing connections, and can lead to significant service degradation. Ideally, a DCN should be able to: (1) provide the agility to optimize its computing resources and move tenant applications from one machine to another as the workload changes (unpredictably), (2) provide stable and predictable network performance with a highly unpredictable traffic load [30, 27], (3) guarantee performance isolation [26, 31] between users.

1.4 Scope and outline of the thesis

Many alternative solutions have been investigated to cope with the current networking scenario and better handle the new traffic mix (e.g. [32, 33, 34, 35, 36, 37]). However, most of these proposals advocate for major architectural redesigns of the current network infrastructure. This is likely to require massive investments and none of the proposed approaches have yet found their way into commercially deployed networks. Furthermore, many of these proposals make extensive use of optical components (e.g.

optical buffers [32]) which, to date, are not commercially viable [38] and don't offer clear advantages over their electronic counterparts [39] [40].

This work aims at developing a networking approach that can work as a complement to the existing network infrastructure. Our idea is to devise a methodology which is specifically designed to efficiently handle large transactions (a significant portion of the new traffic mix [1]), offering significant advantages with respect to traditional IP protocols, such as TCP [41] or UDP [42], in terms of link utilization, delay and computational load, without requiring radical changes in the currently deployed network infrastructure. The approach presented in this work also opens the possibility for a significant reduction of the power consumed by the network hardware by handling large transactions at lower network layers.

Since large-scale hardware redeployment is impractical (especially in the WAN space), one of our design goals is to reuse as much as possible hardware and protocols already deployed, reducing the deployment impact of the proposed methodology, while offering advantages, in terms of network performances and power consumption, similar to those promised by more radical approaches (e.g. [32],[33]) such as high link utilization and low latency. In other words we are trying to provide the current network infrastructure with a means to more efficiently handle a specific portion of current network traffic for which the present infrastructure was not designed for, without rethinking the entire ICT architecture and thereby offering a more gradual evolutionary path.

We also apply our approach to Data Centers (DCs), showing that - in addition to a more efficient utilization of the available resources- significant performance advantages can also be achieved in terms of network performance stability and predictability.

Given the publication-based organization of this dissertation, a general description of the problems faced together with some relevant background information is provided in the main body of the dissertation. A summary of the contributions of the author to the main body of work relative to the use of periodic data structures for data transmission can be found in Section 1.5, while a brief summary of the specific content of the various papers is provided in Section 5. The details of the research can be found in the appendices. The rest of the dissertation is organized as follows: Chapter 2 provides a general overview of the current IP-WDM network infrastructure and a brief analysis of the main causes of power consumption. ITU-T G.709, Optical Transport Network (OTN) and supporting hardware are also described in some detail in this section. Chapter 3 provides details on two of the Next Generation

Network proposals that preceded the present work: Optical Burst Switching and Optical Flow Switching. Their functioning, and implementation details are presented, advantages and drawbacks of OBS and OFS can also be found in Appendix A and B. In Chapter 4 the proposed approach is described in detail. Chapter 5 summarizes the results obtained for the various application cases of periodic data structures to bandwidth-intensive applications, while details on the results and the various studies and methods can be found the Appendices.

1.5 Contributions

1.5.1 Power-efficient Electronic Burst Switching for Large File Transactions

The potential advantages of using various framing approaches for data-intensive transactions were investigated by I.Albanese. I. Albanese wrote a software simulation and conducted a performance study on a simple bottleneck topology. Software simulation of standard UDP/IP protocol was also implemented by I. Albanese. Potential gains in terms of power consumption of the proposed approaches were studied and compared it to that of a standard IP router by I. Albanese. The manuscript was written by I. Albanese and reviewed and edited by all the authors.

1.5.2 Electronic implementation of optical burst switching techniques

A data transfer protocol targeted specifically at handling large transactions was designed by I. Albanese. Two software simulators were developed by I. Albanese, one implementing the proposed protocol and another implementing Tag-OBS. A performance study comparing Tag-OBS to the proposed protocol was conducted by I. Albanese. A simple mathematical demonstration comparing the proposed approach to OFS systems was provided by I. Albanese. The manuscript was written by I. Albanese and reviewed and edited by all the authors.

1.5.3 Big file protocol (BFP): A traffic shaping approach for efficient transport of large files

I. Albanese designed a data transfer protocol (named Big File Protocol or BFP) able to efficiently handle bandwidth-intensive transactions at lower network layers, over currently deployed transport networks. A BFP simulator was designed and implemented by I. Albanese. A TCP simulator was implemented by Dr. Y. O. Yazir. A performance study comparing BFP to TCP/IP was conducted by I. Albanese. A deployment strategy for BFP, potentially not requiring any hardware redeployment was devised by I. Albanese and Dr. Y. O. Yazir. The manuscript was written by I. Albanese with the contribution of Dr. Y. O. Yazir, review and editing of the manuscript was done by all the authors.

1.5.4 Big File Protocol (BFP) for OTN and Ethernet Transport Systems

I. Albanese designed a detailed integration strategy of BFP within ITU-T G.709 OTN hierarchy. Details of mapping procedures and hardware integration of BFP within OTN-enable networks were provided by I. Albanese. The manuscript was written by I. Albanese and reviewed and edited by all the authors.

1.5.5 Big File Protocol (BFP) for efficient quasi-deterministic data transfer in data centers

BFP was adapted to work in a data center environment by I. Albanese (the resulting variant of BFP was named DC-BFP). A simple load balancing protocol specific for DC-BFP was designed by I. Albanese. Implementation and testing of the load balancing approach was done by I. Albanese and Dr. Y. O. Yazir. An optimized software implementation of BFP was designed by I. Albanese. Dr. Y.O. Yazir helped with the implementation and testing of the BFP simulator. A detailed performance study was conducted by I. Albanese comparing DC-BFP to some of the most popular data center network architectures and proposals. An integration strategy for BFP within the data center network was devised by I. Albanese. The manuscript was written by I. Albanese and reviewed and edited by all the authors.

Chapter 2

State of the art

In this chapter a description of current networking architecture is given in some detail in order to identify those elements most responsible for power consumption. Some of the most interesting proposals for future network architecture will also be presented.

Section 2.3.1 provides a brief overview of the transport architecture of interest in this work, namely the recently standardized ITU-T G.709 Optical Transport Network (OTN). Section 2.4 concludes this chapter by presenting a brief overview of the most common Data Center network architectures and the relative issues.

2.1 Overview of Communication Networks

Let's start from the beginning: what is a communication network?

Due to its complexity, there is no clear and simple definition. We can think of a network, however, as a series of interconnected elements which exchange information using various media, generally called *links*. A link can be a wireless link, a coaxial cable, or an optical fiber, just to name a few. Each link type has its own characteristics and performance. In this work we consider optical fiber links, which can achieve very high bitrates (in the order of magnitude of Tb/S), and have the advantage to be completely uninfluenced by electromagnetic interference and ground currents. Elements connected by these links can be of many types and are generally called *end-systems*. This name refers to anything that is connected on the edge of a network: a personal computer, a mobile phone, TV, or even a security system. Usually end-systems are not directly connected to each other but are connected through intermediate devices called switches, which can be classified based on what entity they are able to

switch (i.e. packet switch, burst switch, etc.). Going deeper into the network we meet the network nodes (routers), which together with the physical links provide the end-systems with the infrastructure for their communication. This infrastructure is a network of its own. Network elements can be interconnected according to various topologies. Some of the most common are shown in Fig. 2.1:

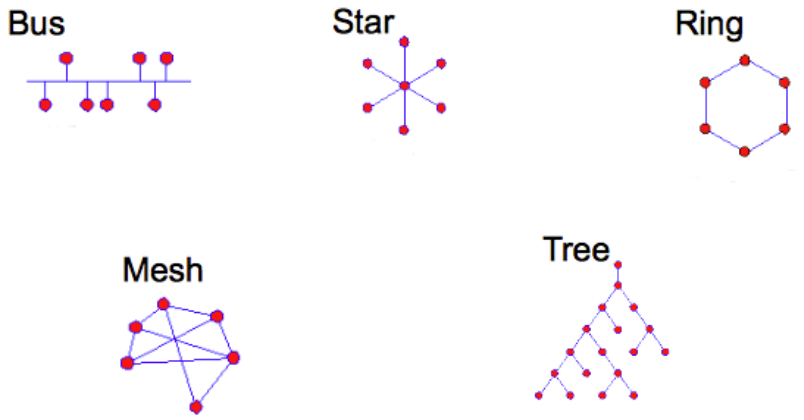


Figure 2.1: Common topologies

In the picture, above red dots represent the nodes while the blue lines represent the links. Each entity in a communication network behaves according to a protocol. The protocols in a network can be thought of as the software structure of a network and are as important as their physical counterpart (i.e. the network hardware). A protocol defines certain rules that are to be followed by the elements of the network in order to keep communicating. Specific messages have to be sent in order to establish (or release) communication, and specific responses are awaited from the communicating entities.

A rigorous definition of *protocol* can be taken from [43]:

"*A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the action taken on the transmission and/or receipt of a message or another event.*"

If we accept the notion of protocol as a series of rules, we can go on and divide these rules into three main categories:

- *Algorithms*: specifies how entities accessing the network should behave as well as the internal network's behaviour.

- *Timing*: determines when the algorithms (implementing the protocols) should be executed.
- *Formats*: describes how information exchanged should be codified and/or structured.

In communication networks, protocols define how users should access network services and how these services are provided to them. Network protocols can be quite complex, and to make their design easier we need a structured and clear model of the network, i.e., a way to *think* about the network. One possible way of looking at a network is to divide protocols into layers, each one implementing simpler protocols, which combined together implement more complex functions. Each layer provides the upper layers some service, starting from the service provided to it from its lower layer. How a layer manages to provide its services to the upper layer is of no concern for the layers above it. This principle, often encountered when dealing with information processing, is called *encapsulation*. Each layer can communicate only with its adjacent layers and therefore can request a service only to its lower layer, and provide its services only to its upper layer. Communication between layers is done using *interface primitives* (functions built to enable interaction between two layers). One model, which has the aforementioned structure, and is a good means to form an idea on how a network is organized, is the OSI model (Open Systems Interconnection) [44], developed by the International Standards Organization (ISO). Although limits of the OSI model are well known and easy to find, its usefulness is also well known and in fact is widely used in the networking community.

Let's start with the name"Open Systems Interconnection", that is, a system which is *open*, and therefore able to exchange information through a network. Each end system is modeled by seven layers while intermediate systems (e.g. switches) may have fewer layers to model their functioning and structure. The lower layer of the OSI model is the *Physical Layer*. It interfaces directly with the transmission medium and defines modulation standards, signal coding, signal power levels to be used for transmission, mechanical structure of connectors used to connect with the physical layer, and so on.

The second layer of the stack is the *Data Link Layer* and is split into two sublayers: *Media Access Control* (the MAC layer) and *Logical Link Control* (LLC layer). The MAC sublayer defines the standards to access and share the communication medium (which is usually shared among many users) while the LLC sublayer pro-

vides a communication free from transmission errors, controls the data flow in order to match the transmission and reception speeds between sender and receiver, and regulates segmentation and delimitation of data flows. In the data link layer the fundamental data unit is the bit string.

The third layer is called *Network Layer*. This layer sees the network as a series of links and nodes and mainly deals with routing and congestion control (i.e. traffic engineering, network engineering, network planning, etc...). The fundamental unit for this layer is the packet.

The fourth layer is called *Transport Layer* and is the first layer in the stack dealing with direct communication between two end systems (i.e. from source to destination). This layer implements functionalities in charge of checking messages for errors and performing flow control between two end-users. Moreover, it fragments messages within packets at the source and reassembles them at destination. Its fundamental unit is the message.

The *Session Layer* manages communications between two users, defining the structure and synchronization of the dialogue between them. Going up in the stack, we find the second to last layer called *Presentation Layer* which deals with data representation format, that implements functionalities such as cryptography and data compression. The last layer of the OSI model is the *Application Layer*. This layer interfaces directly with the user. It implements services like file exchange, e-mail exchange, access to online resources, and so on. A visual representation of the OSI model is depicted in Figure 2.2.

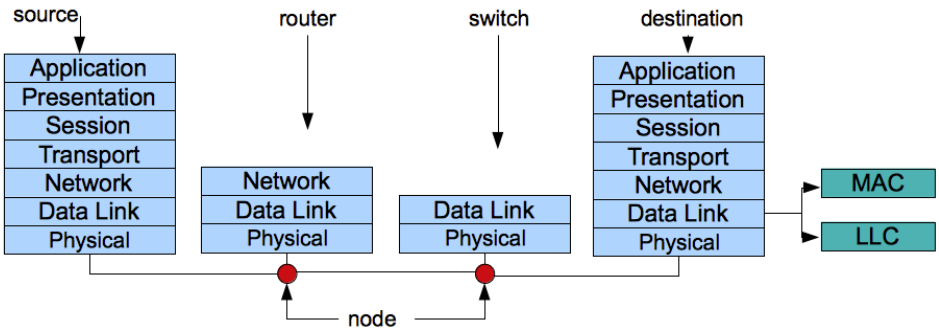


Figure 2.2: The OSI Model

In a network, end systems communicate using the infrastructure provided by the core network. Core networks provide two main kinds of services:

1 *Connection-oriented*: Reliable service. Control information is exchanged between communicating entities to prepare the network to receive the data flow. Data delivery is guaranteed.

2 *Connectionless*: Unreliable service. When one entity has to send some data it sends the data through the network and "hopes" it will be delivered correctly (*Best Effort*).

Networks can be classified based on their geographical extent, capacity, and number of customers serviced.

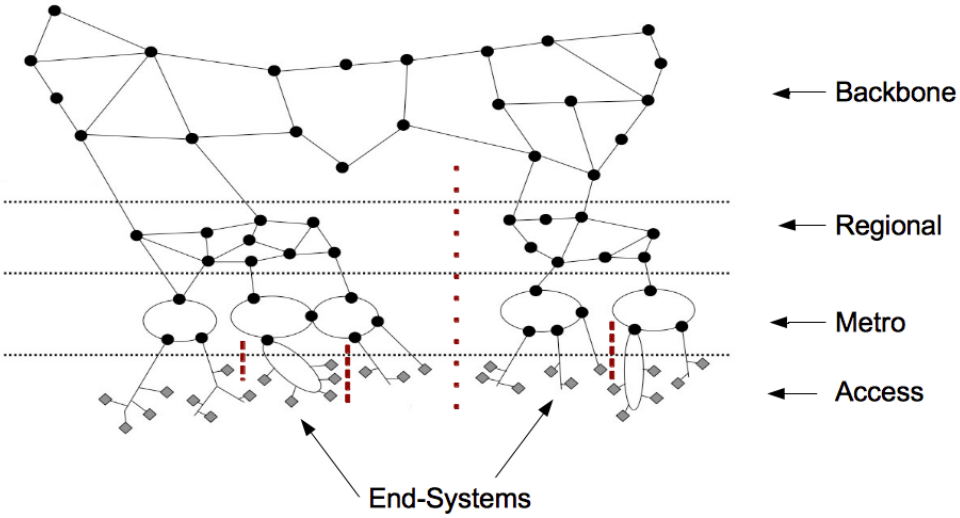


Figure 2.3: General network's topological organization [45].

At the lowest layer of Fig. 2.3 is the *Access Network*. Here are the end-systems. In this part of the network two main operations take place: data is collected from the end systems and is given an appropriate format in order to be sent through the next layer of the network (data aggregation); and from the downstream point of view, data is distributed and routed among the end-systems. This layer of the network typically extends for a few kilometers and can serve (roughly) hundreds of end-systems. A wide range of access technologies are available and others are being developed [46].

One step higher from the access network (See Figure 2.3), we find the *Metro/Edge Network* followed by the *Regional Network* (connecting various metro networks to each other). Similarly to the access network, this layer aggregates traffic coming from the layer beneath (i.e. the access or metro networks) and prepares it for transmission through the higher layers of the network. The geographical extension of these

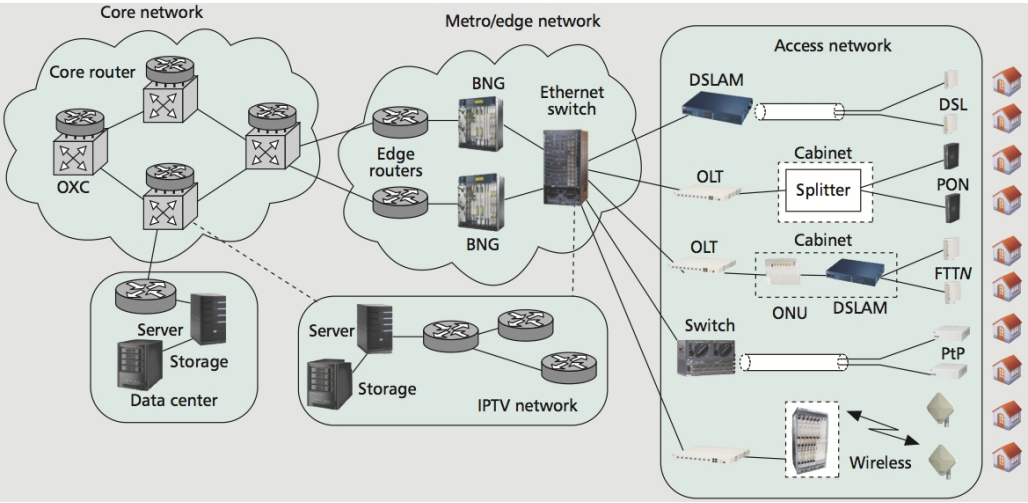


Figure 2.4: High-level network structure [10].

two layers ranges from hundreds of kilometres to thousands of kilometres, and the customers served vary from thousands to hundreds of thousands. At this layer of the network, we find specialized routers in charge of controlling the access to different services within the network by controlling the access rates, providing authentication, security services, and eventually compiling statistics for billing purposes. An Ethernet switch (see Figure 2.4) then connects to a Broadband Network Gateway router (or BNG), which performs authentication and access control functions, and is in turn connected to a Provider Edge router connected to the core network. Ethernet switch, BNG, and the Provider Edge are grouped to form an edge node.

At the very top layer of Figure 2.3 we find the Core Network, which is usually comprised of a small number of large interconnected routers often located in major cities. This layer of the network extends over planetary dimensions and serves millions of end systems. Core routers perform routing functions and also serve as gateways to neighboring core routers. High capacity Wavelength Division Multiplexed (WDM) fiber links are used to interconnect core routers and networks belonging to various operators. Typical speeds of a core link is $40Gb/s$ (per wavelength) with $100Gb/s$ bitrates well underway. In order to give an idea of the order of magnitude of the power consumed by this kind of hardware, we take as an example one rack of the Cisco CRS-1 with a switching capacity of $640Gb/s$ (full-duplex) and consuming roughly $\sim 10kW$ [47].

2.2 WDM Networks

WDM networks constitute the optical backbone infrastructure for the world's telecommunications, due to their capacity to exploit the full extent of the BW provided by the optical medium without requiring the connected hardware to work at impractical speeds. The multiplexing of wavelengths into a single fiber allows the network's hardware to work, at most, at the maximum speed supported by electronic equipment. By multiplexing the signals into a single fiber it is possible to achieve speeds approaching $50Tb/S$, many orders of magnitude higher than the maximum speed achievable with electronic equipment (few tens of Gb/S), hence the idea of introducing some concurrency between end-systems accessing the optical media.

WDM is not the only technology that was proposed. Other possible solutions like Optical Time Division Multiplexing(OTDM) [48] or Optical Code Division Multiplexing (OCDM) [49] have been studied. WDM, however, seems the only practically feasible solution (at least with today's technology), since both OTDM and OCDM require the transmitting node to operate at impractical speeds for an electronic device of any kind. In WDM networks, instead, the various flows are aggregated (multiplexed) before transmission and the resulting bit-rate is the aggregated bitrate of all the sources, which can now work at more feasible speeds. WDM also allows for the use of wavelengths (or a set of wavelengths) to setup logical circuits within the same physical infrastructure, effectively establishing many logical topologies over the same physical network. Such logical topologies can be reconfigured as needed. The possibility for the application layer to directly access the optical layer, bypassing all the intermediate layers, provides a protocol and data format independent transport service - yet another advantage of WDM networks.

The typical architectural implementation requires that an infrastructure, together with a distributed intelligence, is used to control the optical network and provide reconfigurability in an automated fashion, for example, in order to adapt dynamically the network's configuration to different traffic conditions (e.g. changing the logical topology). This infrastructure is referred to as the *control plane* of the network. The control plane is usually implemented with electronic components due to the complexity of the functionalities found at this layer of the network.

Various international organizations such as International Telecommunication Union (ITU) [50] and Internet Engineering Task Force (IETF) [51] tried to develop standards to support this electronic control plane, such as Automatically Switched Opti-

cal Network (ASON, developed by ITU) [52] and Generalized Multi-Protocol Label Switching (GMPLS, developed by IETF) [53], including within these standards signaling protocols to implement functions (i.e. automated control of optical networks, discovery of network topology and resources from the entities within the network).

2.2.1 WDM Node Architecture

A WDM node must support two main functions: wavelength routing, implemented by Optical Cross-Connects (OXC), and channel multiplexing/demultiplexing, implemented by Optical Add/Drop Multiplexers (OADM). In this section a generic IP over WDM node architecture is described. This type of hardware can be thought as functionally as made of two parts: a Wavelength Routing Switch (WRS) and an Access Station (AS). The AS deals with adding/dropping local traffic and grooming lower speed traffic. Traffic grooming in an IP over WDM architecture is done using a TDM-based multiplexing technique. The data fluxes are then aggregated onto the same channel. The AS, equipped with an IP/MPLS router, performs multiplexing of lower speed traffic onto high capacity channels. Within the AS are transmitters and receivers which can or cannot be tunable depending on the implementation. The WRS consists of an OXC, a Network Control and Management unit (NC&M) and an OADM. The OXC implements wavelength switching functionalities. The NC&M, further subdivided into a Network to Network Interface (NNI) and a Network to User Interface (NUI), is in charge of configuring the OXC and exchanging control information with the User to Network Interface (UNI) placed within the AS and constituting its control component.

2.2.2 Where does the power go?

In section 2.1 a high-level description of a communication network was given. In the this section, we'll try to understand where the power is consumed and the relationships between the various layers of the network in terms of power consumption.

At the present moment, the access network is the most power-hungry segment of the communication infrastructure [55]. This situation, however, will not last much longer. Figure 2.6 shows that as the access rate increases beyond $\sim 100Mb/s$, the power consumption [1] of the core network starts to dominate, and will increase with

[1] Expressed as the amount of power (W) required by the infrastructure to support each customer.

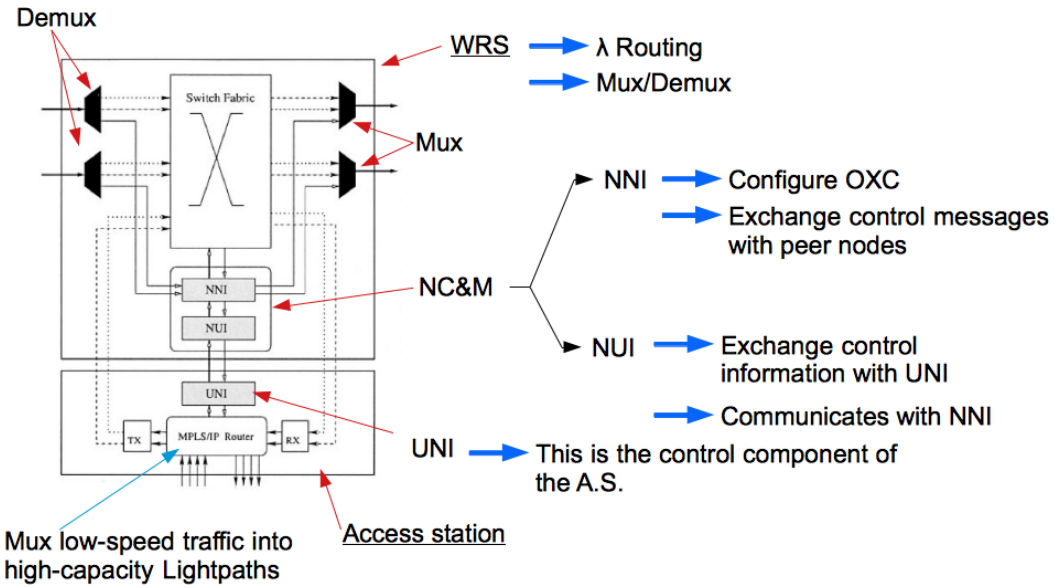


Figure 2.5: WDM node architecture [54].

the access rate more than linearly. It is clear then that a more efficient approach is needed in order to fulfill the demand for an ever increasing access rate driven by the rapid proliferation of bandwidth-intensive applications accessed by more and more users (such as NetFlix [56] or YouTube [57], offering HD video streaming services to a large user-base).

Let's now take a closer look at the network hardware. An interesting approach to model a generic network node is provided in [58]. This model (Figure 2.7) distinguishes eight high level entities, which are present in most of the intermediate systems (IS) of interest in this work. Various types of systems will have each part implemented with different technologies and carrying out different tasks, but the basic functional blocks present in the model would still be there in some form.

The functions mapped by each block of the model present in Figure 2.7 are explained below:

- *Optics*: In this block are the optical modules necessary for transmission/reception of optical signals, and a serializer/deserializer providing support for multiple encoding schemes, and allowing presentation of those encoding schemes to the upper layers. The hardware here is typically implemented with CMOS and photonics technology.

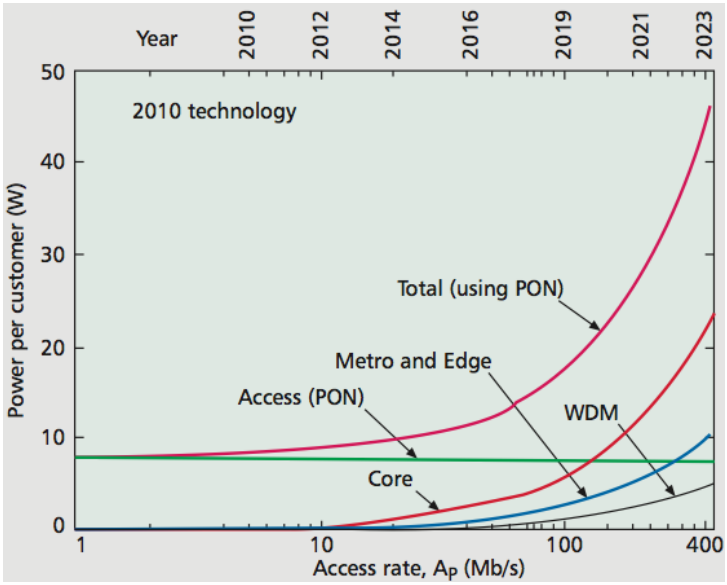


Figure 2.6: Power consumption of the various segments of the communication infrastructure [10].

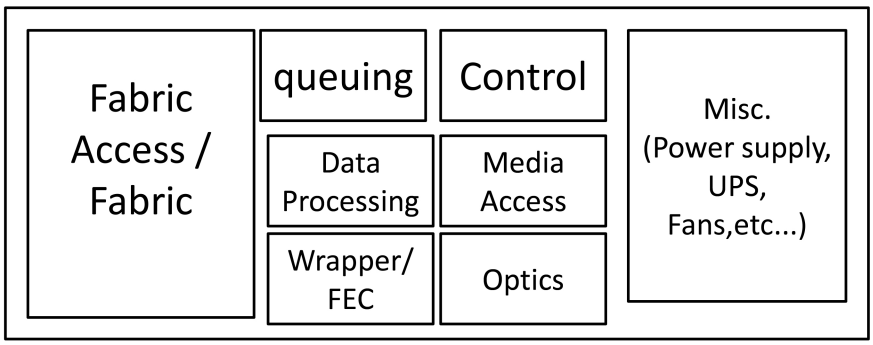


Figure 2.7: A generic abstract model for transmission systems.

- *Wrapper and FEC*: A digital wrapper is present in most of today's Optical Transport Network (OTN) systems or in system having at least one OTN interface. This block is in charge of wrapping incoming data frames into Optical Data Units (ODU). Forward Error Correction (FEC) functions are also located in this functional block. Both functionalities are implemented using CMOS technology.

- *Medium Access*: Here reside the functions described by the MAC layer of the OSI pile for an IP router (electronic terminations, client monitoring, etc.). In the case of a $SONET/SDH$ cross-connect, this block groups the framer, High

Order/Low Order (HO/LO) multiplexer, and some primitives such as fault management and performance monitoring.

- *Traffic/Data Processing*: This functional block includes functions such as packet forwarding, header processing, packet classification, metering and policing. For example, in IP routers it includes functions like Deep Packet Inspection (DPI) or, in the case of a TDM cross connect, it can perform data adaptation and pointer processing functions. Components in this entity are usually realized using Ternary Content Addressable Memory (TCAM), used to store, retrieve, and match strings during algorithmic searches (some protocols require such operations, for example, when searching for a match in an IP address of a packet for routing purposes).

- *Queueing and traffic management*: Only IP routers actually implement this functional block. This entity deals with the user's traffic but doesn't directly control the switch fabric. Among the various functions implemented in this block we find performance monitoring and statistics gathering, just to name a few.

- *Fabric Access and Fabric*: Here are grouped functions relative to the access, operation, and protection of the switch fabrics. The structure of this block depends on the entity to be switched (packet, cell, frame, etc...). In an all-optical device, this is usually implemented with a MEMS switch, in which case the fabric access need not be implemented. In IP routers the fabric access device is in charge of balancing the load offered to the fabric and queuing incoming data that must cross the fabric. Arbitration and fabric control are demanded to a subsystem also placed within this functional block. Since service availability is a key factor in these systems, some form of protection is usually provided: typically $m : n$ redundancy for IP routers, or $1 + 1$ protection for TDM and OTN systems.

- *Control*: Under this block is an external management interface, a command interpreter, and some translation functions. These functions are present in basically every system. Route processing also takes place here. Processors here are usually replicated to provide increased reliability against failures and ensure enough computational power to carry out the required tasks (protocols like Border Gateway Protocol -BGP-, Incremental Shortest Path First -ISPF-

or spanning tree protocol are executed in this block. These protocols, especially for IP routers, may be computationally intensive).

- *Miscellaneous*: In this functional block is included all the equipment for the cooling and power supply of the system (both standard and Uninterruptible Power Supply (UPS)).

Now let's consider the power consumption of each functional block. This aspect was also studied in [58] and the results are shown in Figure 2.8.

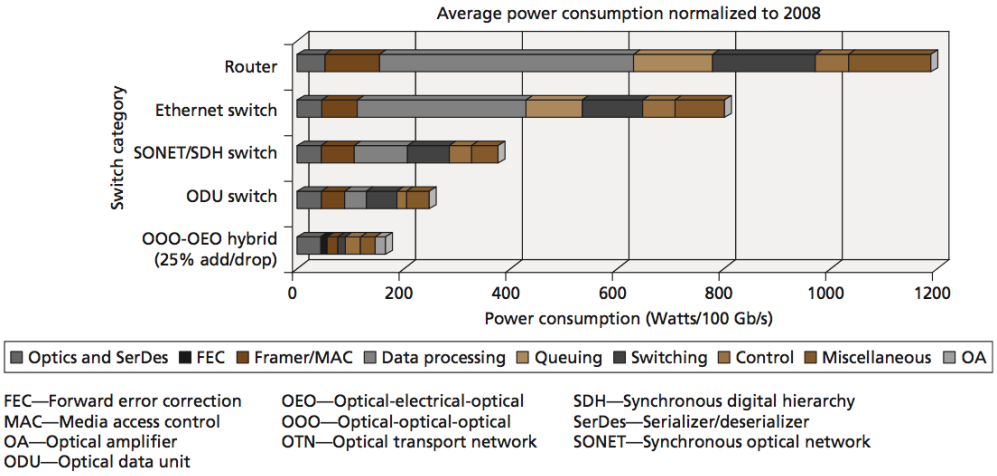


Figure 2.8: Power consumption for different system types, $100Gb/s$ Ethernet interfaces are assumed and values are normalized to power figures of 2008 technologies [58].

From this picture it is clear that IP routers consume much more power than all the other technologies considered. Another thing that can be seen is that three functions dominate the power consumption: data processing, switching, and queuing.

This work aims at optimizing the transmission of large files which, at the present moment, are an important portion of the IP traffic (see Figure 1.1) and are expected to grow even further [1]. It is therefore important to see specifically where the power is consumed in an IP router. An example of a high-end electronic router together with the relative power consumption of its various functional blocks is shown in Fig. 2.9.

From this picture it can be noticed that the forwarding engine, the power supply, and cooling block alone take up to 67% of the total power used by the router, hence a technology enabling transmission of large amounts of data with minimal header processing should result in significant power savings for the overall system.

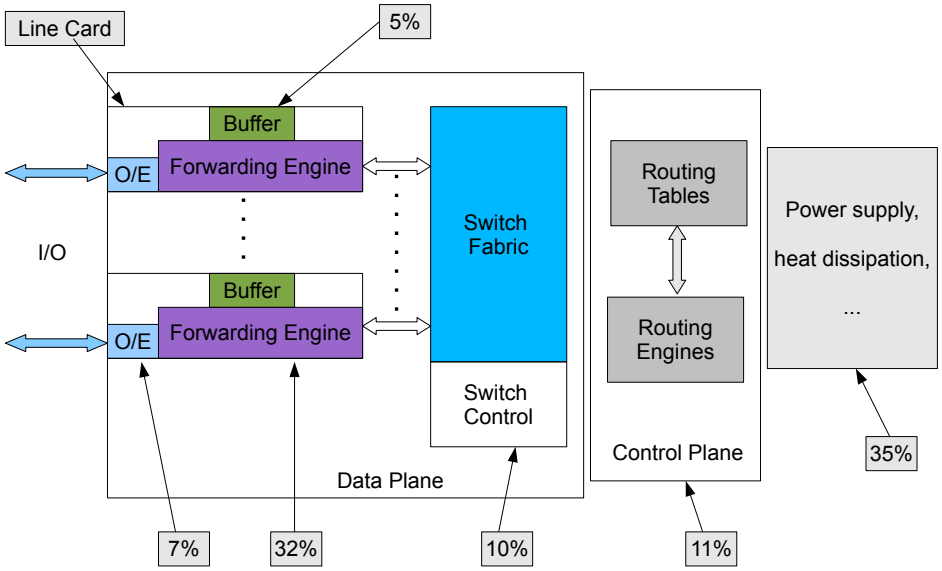


Figure 2.9: Functional blocks of a high-end router and their relative power consumption [59].

2.3 Transport Architecture

In this section we will provide an overview of the recently standardized ITU-T G.709 Optical Transport Network (OTN) architecture [60]. Besides being relevant to the present work, G.709 is rapidly becoming the deployment of choice for transport networks, slowly substituting itself to legacy architectures such as SONET/SDH [61]. A brief discussion on the supporting hardware for OTN will conclude this section.

2.3.1 ITU-T G.709 - Optical Transport Network (OTN)

The shift towards a data-centric, packet-based traffic exposed the limitations of the SONET/SDH infrastructure in terms of scalability, flexibility, and OAM&P capabilities.

The fine-grained TDM structure of SONET/SDH requires a minimum switching granularity of $51.84Mb/s$ to be supported by each intermediate node, making it hard to scale to bitrates in the order of Tb/s and beyond. On the other hand, OTN multiplexing bandwidth granularity is much higher than that of SONET/SDH (roughly one order of magnitude higher), making it much easier for OTN to scale to higher bitrates.

SONET/SDH, besides lacking standard payload containers above $40Gb/s$, is somewhat rigid in subdividing the available bandwidth, often forcing operators to use complex virtual concatenation (VC) techniques. This increases management complexity and is likely to require extensive buffering at the ending points of the network in order to compensate for differential delays.

Another important limitation of SONET/SDH is its Forward Error Correction (FEC). SONET/SDH FEC is relatively limited [62], and consequently so is the maximum distance allowable between regenerators. This requires network operators using SONET/SDH to deploy more hardware, increasing the overall cost of their network. OTN offers a much stronger FEC, providing up to $6.3dB$ coding gain and allowing longer spans to be covered without having to regenerate the signal. This allows for reducing the amount of hardware necessary and lowers the overall cost per bit. This is a compelling reason for the adoption of OTN.

ITU-T G.709 (OTN) [60, 63] provides bit and timing transparent transport services for both CBR and Packet-based clients. Standard containers for any client signal available today are defined together with their relative mapping procedures. Flexible containers (i.e. ODUflex [60]) are also defined in the G.709 transport hierarchy, providing support for packet-based clients with a wide range of bitrates.

OTN flexibility in bandwidth allocation also enables it to achieve higher per-wavelength utilization while supporting a wide range of client signals, including SONET/SDH (which becomes just another client signal for OTN).

Although VC is fully supported by OTN, OTN also provides a much simpler multiplexing structure with respect to SONET/SDH, making operation, administration, maintenance, and provisioning (OAM&P) procedures less complex.

As a result of the many advantages provided by OTN over SONET/SDH (i.e. reduced technology complexity, higher channel utilization, higher flexibility, and higher scalability), there is general agreement in the industry to consider OTN as a requirement for the next generation network infrastructure; OTN is now supported by most of the available network hardware.

In the following paragraphs we'll provide some detail about the OTN architecture, mapping procedures, and hardware needed to support OTN.

OTN Architecture

When a set of client signals are to be carried over a WDM system, one option is to send each client signal in its native format and pair it with a separated OAM channel (e.g. a separate wavelength). This, however, is not an ideal solution, as each client would require twice as many channels. Furthermore, client and OAM channels may not experience the same impairments and (eventual) faults conditions, making this approach difficult to manage.

Another approach is to add the OAM channel to the client signal using sub-carrier multiplexing, removing it at the end point by filtering the signal with a low-pass filter. This approach also turned out to be quite complex to handle besides negatively impacting the jitter performance of the system.

A third option is also available, in which the client and OAM signals are carried over the same channel. Following this approach, the client signal is handled as the payload of a digital frame, the overhead of which carries the necessary information for OAM operations relative to the client signal. This approach is referred to as *digital wrapper* approach.

In OTN, a digital wrapper approach is used to encapsulate the client signal and its associated OAM overhead. The resulting signal is then either mapped directly onto a wavelength or multiplexed via Time Division Multiplexing (TDM) with other client signals into a higher-rate signal. The resulting signal is then mapped onto a wavelength (e.g. when multiple client signals sharing the same wavelength). A set of wavelengths carrying the client signals, are then grouped together and a separate wavelength, carrying the optical network overhead relative to the group, is added to it.

Signal Architecture - OTN digital layer comprises 3 hierarchical data containers: Optical Payload Unit (OPU), Optical Data Unit (ODU), and Optical Transport Unit (OTU, which is the fully formatted OTN digital signal). Moving on to the optical layer, OTN defines 3 more hierarchical transport entities, namely: Optical Channel (OCh), Optical Multiplexing Section (OMS), and Optical Transport Section (OTS). In this section we'll give a brief overview of the OTN signal architecture and show how OTN containers are assembled as well as their hierarchical relationships. Details of the OTN frame structure and multiplexing hierarchy will be provided in Section 2.3.1 and 2.3.1, respectively.

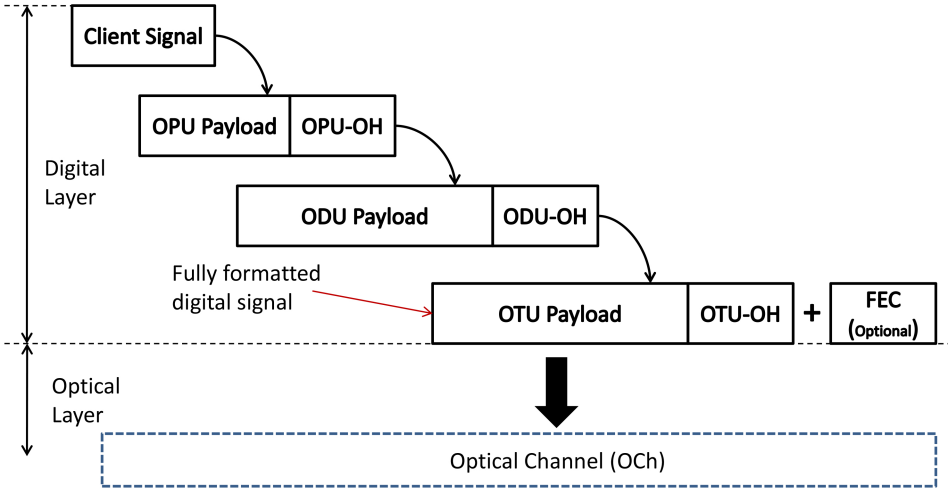


Figure 2.10: OTN digital signal hierarchy

Figure 2.10 shows the hierarchical relationship between the OTN containers at the digital layer. The client signal is first mapped onto the payload area of the OPU, then the relative OAM channel is added to form the OPU. The OPU can be thought as the analogous to SONET/SDH *Path* [61]. The OPU is then mapped onto the payload area of the ODU. After the relative ODU overhead (ODU-OH) is added, the ODU is fully formed. ODU is functionally equivalent to the SONET *line* (*multiplex section*, if we follow the SDH terminology). At this point the OTU overhead (OTU-OH) is added. The OTU-OH adds frame alignment overhead (i.e. FAS and MFAS, See Section 2.3.1) as well as an optional OH used for Forward Error Correction. The OTU frame is a fully formatted digital signal for OTN, it is functionally analogous to the SONET *Section* (equivalent to SDH *Regenerator section*), and is mapped directly onto the Optical Channel (OCh).

The first transport entity of the OTN optical layer architecture [64] is the Optical Channel (OCh) which is a wavelength in a WDM system. A group of OChs is wavelength division multiplexed, and a supervisory channel is the added (Optical Supervisory Channel or OSC) to form the Optical Multiplexed Section (OMS). A group of n OMS each with its own OSC forms an OMS of order n. By adding one overhead channel to an OMS of *order* n an Optical Transmission Section of *order* n is formed. The supervisory channels of OTS, OMS, and OCh are used to assess the quality of the transmission channel, implement defect detection, and connectivity verification functionalities.

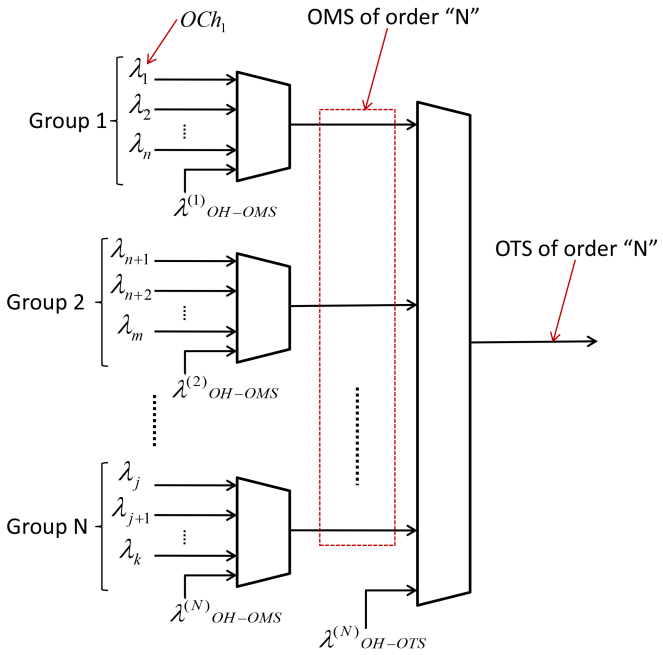


Figure 2.11: OTN optical signal hierarchy

The hierarchical relationships between the transport entities at the optical layer for OTN are shown in Figure 2.11, while the scope of the various OTN layers is shown in Figure 2.12.

OTN Frame Structure

A fully formed OTN frame (i.e. an OTU), is made of 4 rows $\times$ 4080 columns (including the optional FEC bytes of the OTU-OH). Its maximum size is therefore equal to 16320 bytes (or 15296 bytes if FEC is not used). The OTN frame size is fixed, regardless of the data rate. As the data rate changes the OTN frame period changes accordingly (Table 2.1). Figure 2.13 shows an OTN frame divided in areas (A to G), each representing a field-specific portion of the overhead.

The innermost area of the OTN frame (area F of Figure 2.13) is the OPU payload area. Here is where the client signal is mapped. The relative overhead area is further subdivided into two areas (areas D and E of Figure 2.13). The OPU-OH covers the OPU from the client signal mapping point to its extraction, and handles its mapping and demapping. This portion of the overhead implements justification control functionalities (*area D*) and provides support for Virtual Concatenation (*area E*). Except for the Payload Structure Indicator byte (PSI, Figure 2.14), the use of the

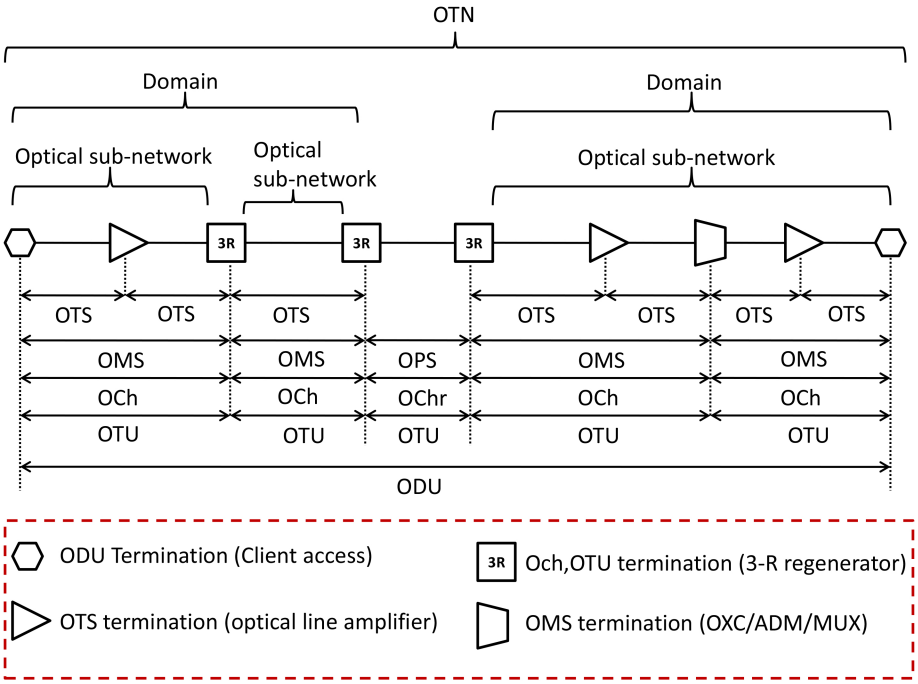


Figure 2.12: Scope of the OTN layers

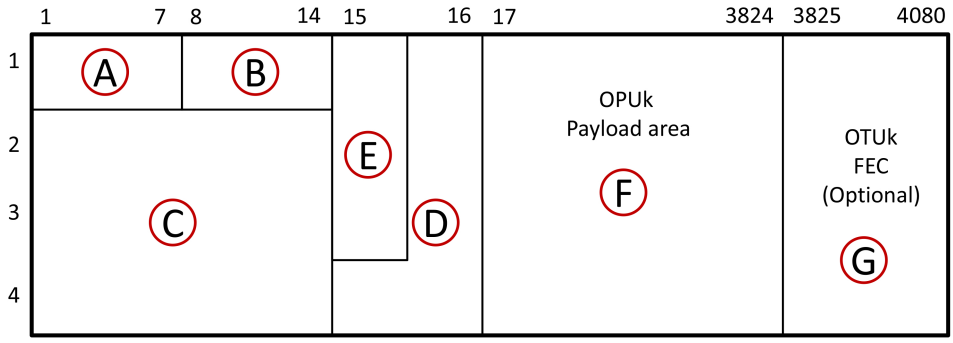


Figure 2.13: OTN Frame structure

remainder of the bytes of the OPU-OH depends on which client is being serviced as well as on which mapping procedure is used [60]. The PSI byte is part of a 256-byte signal associated with an *ODU multiframe* (i.e. 256-frames multiframe). The first byte of the PSI signal ($PSI[0]$, found in the first frame of the ODU multiframe, i.e. position 0000 0000) is referred to as the Payload Type field (PT), and indicates the composition of the OPU payload. The remaining bytes of the PSI signal ($PSI[1]$ to $PSI[255]$) are mapping and concatenation specific. Error correction functionalities for the OPU area are implemented in the ODU-OH area.

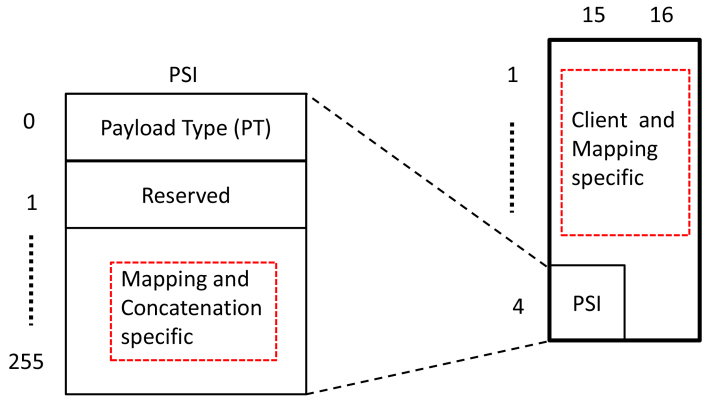


Figure 2.14: OPU Overhead (areas D and E of Figure 2.13)

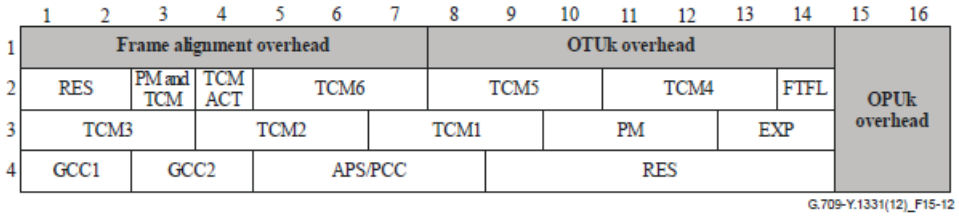
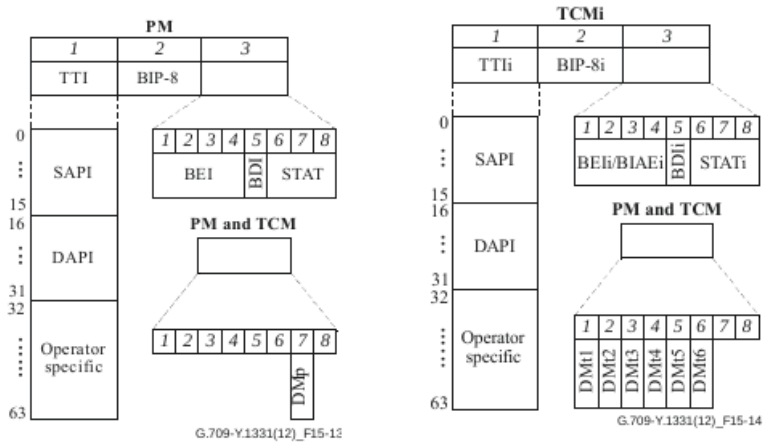


Figure 2.15: ODU Overhead (area C of Figure 2.13) [60]

Moving to the next (lower) layer of the OTN digital hierarchy is the ODU overhead area (area C of Figure 2.13). ODU-OH implements functionalities relative to performance monitoring at the path level (PM area in Figure 2.15), plus up to 6 levels of tandem connection monitoring (TCM1 to TCM6), two generic communication channels (GCC1 and GCC2), 8 bytes reserved for future standardization (RES bytes of Figure 2.15 divided into two areas of 2 and 6 bytes, respectively and set to all 0s), and 3 bytes for experimental purposes (EXP bytes). Fault/error handling functions are found at each TCM level and in the PM area (BEI/BIAE and BDI fields of Figure 2.16(b) and 2.16(a), respectively), as well as at the Fault Type / Fault Location byte (FTFL byte at line 2, column 14). Automatic Protection Switching and Protection Communication Channels (APS/PCC) functionalities are also placed in the ODU-OH. Lastly, a delay measurement function was added later on to the standard and placed in the PM&TCM byte (line 2 column 3).

For a more thorough description of the various functionalities implemented by the ODU-OH, the interested reader is referred to [60].

At the lowest level of the OTN digital hierarchy is the OTU-OH (areas A and B of Figure 2.13). In the first 6 bytes of area A (Figure 2.13, row 1 column 1 to 6) is a



(a) PM Overhead fields (b) TCM Overhead fields

Figure 2.16: PM and TCM$_i$ overhead [60].

bit pattern utilized for frame alignment (Frame Alignment Signal or FAS). The last byte of the frame alignment overhead (area A) is the Multi-Frame Alignment Signal (MFAS), and is used in combination with other multiframe fields of the OTN frame to determine their specific meaning in a specific frame of the multiframe. Bytes from 8 to 14 of row 1 (i.e. area B of Figure 2.13) implement section monitoring functionalities (SM, 3 bytes), a generic communication channel (GCC0, 2 bytes), and the last 2 bytes are reserved for future standardization.

With the exception of the OTU framing bits, all other bits of the OTU are scrambled before transmission onto an optical channel in order to provide the receiver with a high enough transaction density to perform clock recovery on the signal.

OTN Signal rates, Mapping and Multiplexing

ITU-T G.709 defines four fixed signal rates for the OTU, five for the OPU/ODU, and ODUflex signals, which are further subdivided into ODUflex(CBR) and ODUflex(GFP) for CBR and packet-based clients, respectively. ODUflex signals were designed to allow mapping client signals onto OTN using a flexible container to accommodate a wide range of client bitrates. The rate of ODUflex signals can be changed without re-establishing the connection [65].

The specific rate of OTU, ODU, and OPU is indicated with a subscript (e.g. ODU$_k$, where k varies from 0 to 4 given the currently standardize signal rates). Table 2.1 below illustrates the various signal rates and their relative OTN frame

period. In addition to the standard signal rates shown in Table 2.1, other rates were added in order to provide a simplified mapping procedure for Ethernet client signals that didn't fit within the already defined OTN containers, namely: ODU2e, ODU3e1, and ODU3e2 [66].

Table 2.1: Currently standardized OTN signal rates (ODUflex is not indicated as signal rate is not fixed)

k	OTU_k signal rate	OPU_k payload area rate	$OTU_k/ODU_k/OPU_k$ frame period
0	N/A	1.238954 Gb/s	98.354 μs
1	2.666057 Gb/s	2.488320 Gb/s	48.971 μs
2	10.709225 Gb/s	9.995277 Gb/s	12.191 μs
3	43.018414 Gb/s	40.150519 Gb/s	3.035 μs
4	111.809974 Gb/s	104.355975 Gb/s	1.168 μs

OTN Payload Mapping - When mapping a client signal (CBR or packet/cell-based) onto OTN, the client signal is mapped directly into the OPU payload area of an OTN frame.

There are three main mapping procedures defined by OTN: Asynchronous Mapping Procedure (AMP), Bit-Synchronous mapping procedure (BMP), Generic Mapping Procedure (GMP) [60].

When performing asynchronous mapping (AMP) the client signal rate is adapted to the OTN signal rate. Rate adaptation between the client signal rate and the OPU payload rate is done using the justification control bytes available at the OPU-OH. Up to $\pm 45 ppm$ of client frequency variation can be accommodated using AMP. In this case frequency justification is necessary since the OPU clock is generated locally and is not related to the client signal.

When using bit-synchronous mapping (BMP), the OPU clock is derived from the client signal clock. This way the phase and rate of the OPU signal are locked to those of the client signal and no justification control is necessary. In other words, BMP wraps the client frame with the OTN overhead.

OTN also provides a mapping procedure for clients with rates that differ significantly from the defined OPUk rates (Table 2.1). These clients can be mapped using GMP, which maps the client signal onto the OPU payload area, alternating data words with stuffing words using a Sigma/Delta algorithm to decide which word of the OPU payload area is to contain data and which is to be filled with stuffing. Information

about the amount and position of data words in the $i-th$ frame is communicated to the receiving node by the $(i-1)-th$ frame so the receiver can distinguish which word contains data and which contains stuffing when recovering the client signal. Using this technique, frequency ranges much wider than $\pm 45ppm$ can be accommodated. A similar technique can also be used for multiplexing LO-ODUs into a HO-ODU.

For cell or packet based clients OTN uses GFP [67] to encapsulate data packets and generate a continuous stream of GFP frames which is then mapped directly in an octet-aligned directly onto the OPU payload area. In this case, rate adaptation is done using GFP idle frames, which are transmitted anytime there is no data to send.

Note that there are also other mapping procedures, as well as client specific mappings (e.g. for Ethernet signals), which are not described in detail here. The interested reader can refer directly to [60] for a complete description of the mapping procedures available for OTN.

Before moving on to illustrate OTN multiplexing principles, it is important to add some terminology relative to ODUs, specifically what is meant for *Low Order* and *High Order* ODUs. ODU signals are referred to as High Order ODU (HO-ODU) or Low Order ODU (LO-ODU) depending on their role as clients or servers of another ODU signal. Specifically, an ODU signal servicing (carrying) another signal is always referred to as the HO-ODU, while the one that is being serviced (carried) is referred to as the LO-ODU. A similar terminology is used for OPUs. Note that ODUflex signals (i.e. ODUflex(CBR) and ODUflex(GFP) [60]) are always LO, meaning that they are always mapped into higher-rate ODUs before transport.

OTN Multiplexing - In order to multiplex LO signals into HO signals, OTN uses TDM. The HO-OPU payload area is divided into Tributary Slots (TS) and an integer number of TS is assigned to each LO-ODU.

Since the OTN frame size is always the same regardless of the data rate, a LO-ODU is multiplexed into a set of HO-OPU (i.e. a multiframe). The size of the multiframe, that is, the number of frames in a multiframe, equals the number of TS in the payload area of the HO-OPU. The smallest TS supported by OTN is 1.25G. 2.5G TS, however, are also allowed to provide support for legacy mappings (this is the case for ODU_2 and ODU_3, supporting both 1.25G and 2.5G TS). The number of TS in each ODU_k type (with $k = 1...4$) is illustrated in Table 2.2.

The OTN multiplexing procedure is reminiscent of the mapping one, both AMP and GMP are used also for multiplexing. The difference with the mapping procedure

is in the use of an intermediate structure, called Optical Channel Data Tributary Unit (ODTU). The ODTU is essentially a portion of the HO-OPU which is reserved for a specific LO-ODU that is being multiplexed onto it, and contains both the justification overhead used in the LO-ODU and the LO-ODU itsef. A LO-ODU is asynchronously mapped into the OTDU, and then is multiplexed onto an integer number of TS in the HO-OPU payload area using, in some cases, GMP [60]. The OTN multiplexing hierarchy together with the various mapping and multiplexing passages is shown in Figure 2.17.

Table 2.2: Number of Time Slots (TS) and relative size in HO-OPUk

HO-OPU type	Number of TS ($\equiv$ frames per multiframe)	TS size
ODU_1	2	1.25G
ODU_2	4	2.5G
ODU_2	8	1.25G
ODU_3	16	2.5G
ODU_3	32	1.25G
ODU_4	80	1.25G

OTN Hardware [P-OTP, μ-OTP - hardware overwiew]

The dramatic growth in packet traffic is pushing network operators to deploy network elements (NE) that can handle switching packet, OTN, and Optical domain traffic. Some operators are oriented toward deploying a set of single-function network elements, each dealing with one type of traffic, e.g., using Reconfigurable Add/Drop Multiplexers (ROADM, eventually equipped with OTN framers) to handle traffic in the optical domain, and using Carrier-Ethernet Switching Routers (CESR) to handle packet traffic in the electrical domain. Other Operators are more oriented towards using a single platform able to handle WDM, TDM-based (e.g. OTN), and packet traffic. Such a platform is referred to as Packet-Optical Transport Platform (P-OTP), and its architecture will be the focus of the following section. Using a single platform to handle all types of traffic has its drawbacks, such as the difficulty to upgrade the system. Using a single network element able to handle a wide variety of traffic types in various domains simplifies network operations and can reduce the operating expenses (OPEX) for the operators.

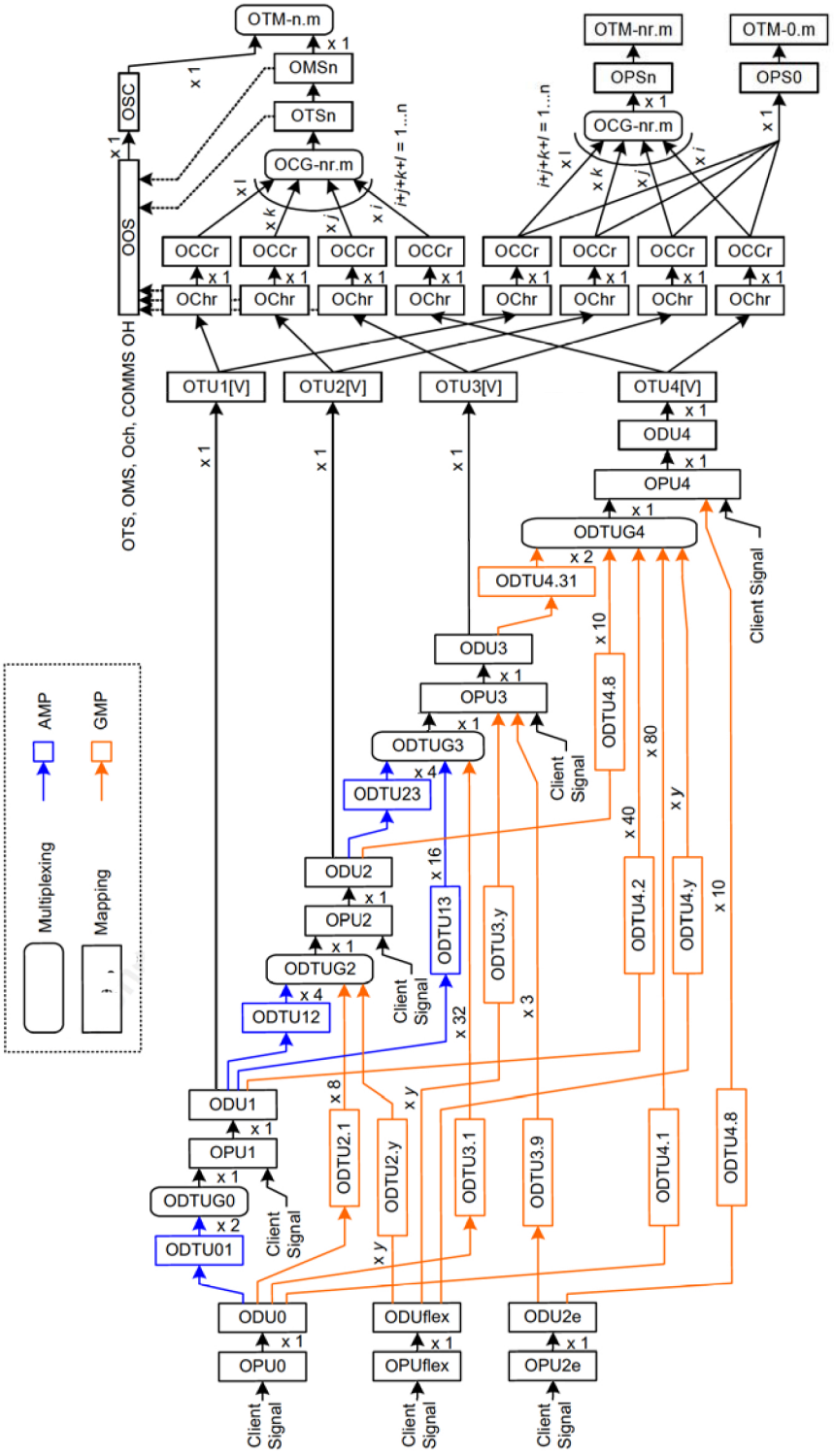


Figure 2.17: OTN multiplexing hierarchy [63]

A P-OTP (Figure 2.18) is designed to offer maximum flexibility and should be able to take as an input any kind of traffic and multiplex it onto any lambda(s). A P-OTP contains, at least, the following elements:

- ROADM
- System-wide traffic grooming for TDM-based traffic (OTN)
- Support for Connection-Oriented Ethernet [68], with system-wide L2 aggregation and switching
- A centralized fabric able to switch both packet and OTN (i.e. ODU_k) traffic in the electrical domain concurrently

The ability to switch concurrently TDM and packet traffic can be achieved in various ways. Two separate fabrics can be deployed, one for TDM traffic and one for packet traffic. The drawback of this approach is that if one of the two fabrics hits its scalability limit, the whole platform would be limited by that. Whereas scaling both fabrics would still require building two distinct fabrics and would increase the overall cost and power consumption of the P-OTP. Furthermore, a backplane able to handle both fabrics concurrently should be used, and this kind of equipment can be quite complex.

Another option is to use a packet or cell based fabric, and adapt either to handle TDM traffic as well, mimicking the behavior of a TDM fabric [69]. Such a device is able to transmit ODU_k and its timing over a packet fabric, enabling transmission of ODU_k without disrupting the timing characteristics of the OTN signal.

Further details on P-OTPs can be found in Appendix D.

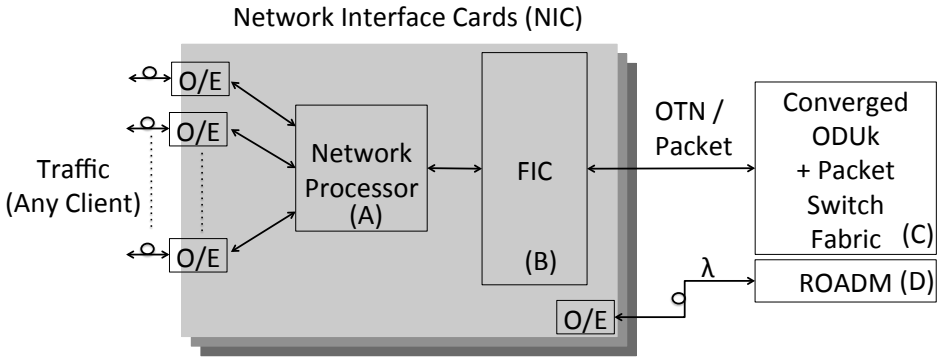


Figure 2.18: Packet - Optical Transport Network (P-OTP)

2.4 Data Centers

In recent years a significant shift towards server-side computing (i.e. "cloud computing") is taking place. Services that used to be hosted in client machines such as data storage, office applications, and computing services are offered more and more as Internet-based services. In addition to such services, a plethora of other Internet-based applications is becoming more and more reliant on cloud computing infrastructures. These services include web search, web mail, and social networking, just to name a few.

As a result, significant research effort is being made to improve the performance of cloud computing facilities, which should be able to continue supporting the rapid growth of the aforementioned services while remaining profitable for the operators [70, 71, 72].

The advantages offered to both clients and operators by cloud computing facilities are several, including: (1) improved user experience (users can now access extensive computing resources); (2) ease of software updates for software vendors, which can deploy their software upgrades only inside their DCs on a small number of well-tested hardware configurations, instead of having to integrate with a wide range of different machines; and (3) the ability to share data center (DC) resources among many users, improving resource utilization and leading to significant costs savings for the DC operators.

A DC is assembled with a large number of *commodity-off-the-shelf* (COTS) components (i.e. servers), constituting the computing resources of the DC. The choice of using COTS instead of high-end machines comes from the need to use cost-efficient hardware configurations for the DC. The difference in performance between high-end servers and low-end servers is relatively small in a DC scenario, where there are tens of thousands machines and the workload is highly parallelized [73]. Using a large number of low-end server platforms also benefits from an economy of scale with respect to using a smaller number of high-end systems. There is a limit, however, to how "low-end" the components should be chosen when designing a data center. The reasons are several. As more (cheaper) servers are used, the software needs to be optimized for handling this highly parallelized environment. This can result in increased software development and optimization costs, negating the savings achieved with the use low-cost hardware. Furthermore, as more machines are connected to the data center network (DCN), a higher number of ports will be needed, increasing the

overall hardware cost for the DC. Another issue of using large numbers of cheaper servers is that these servers may be able to fit in their hard drives more applications than their CPUs are able to handle, leading to inefficient resource utilization (i.e. a portion of their hard drives may be left empty). Lastly, faults and failures are more likely to occur in low-end servers, negatively impacting performance of the data center[2].

The availability required by most Internet-based services is in the order of $\sim$ 99.99%, making fault tolerance a major concern in DCs. In DCs with tens of thousands machines (or more), failures can occur on an almost hourly basis [73]. The software infrastructure of a DC must be designed to account for this and provide stable performances even in the presence of failures.

In a DC, servers are *packaged* into racks mounting (roughly) 20 to 40 machines, all connected to a Top-of-Rack switch (ToR), in turn connecting the servers to one or more cluster-level switch, depending on the DCN architecture. Link capacity from server ToR is usually in the $1Gb/s$ range, while from to ToR to the higher layers it typically ranges from 1 to $10Gb/s$. There are several alternatives for interconnecting racks and implementing DCNs, depending on what type of workload and applications the DC has to support. We refer to [17, 18] for a summary of the most popular DCN architectures (both implemented and proposed ones).

Disk drives can be directly attached to the servers (Directly Attached Storage or DAS) and managed by a distributed file system (e.g. [75]). Alternatively, the storage facility for servers can be its own storage system (Network Attached Storage or NAS), connected to the servers via a cluster-level switching fabric. NAS are usually simpler to deploy; fault handling, data management, and integrity is left to the NAS vendor. In the NAS case, this is usually achieved by replication and error correction capabilities within the NAS. DAS are more complex to deploy as fault-handling is left to the cluster-level file system. The higher system complexity is traded for a lower overall hardware cost, as the storage resources (disks) leverage on the existing server enclosure and network ports at each server are effectively shared between the computing task and file system. In DAS systems, reliability is achieved by replicating content across the disk drives of different machines, resulting in higher use of the network bandwidth and higher aggregate bandwidth as data can be read from multiple sources simultaneously. In a DAS system, each server has a local DRAM (shared) and directly attached disk drives, which are accessed through the ToR switch. Cluster-

[2]A classification of DCs based on the service availability provided can be found in [74]

level switches can access all storage resources in all racks. The trade-offs in DAS systems is between higher write overheads and higher availability, and lower costs as well as the possibility to exploit data locality. Larger DCs usually deploy DAS, while smaller clusters usually opt for NAS systems.

The software infrastructure of a DC is typically comprised of three layers:

1. *Platform-level software*, providing server-level services and including servers' operating systems, and all the libraries to provide hardware abstractions for the servers.

2. *Cluster-level software*, usually a distributed software infrastructure providing resource management and cluster-level services such as distributed file system, remote procedure calls, and schedulers. This portion of the DC software architecture provides the means to handle resources at the DC scale. Applications such as Hadoop [76], map-reduce [77] and Chubby [78] are examples of the software applications running at this layer.

3. *Application-level software*, implementing specific services such as web-search, e-mail services, and so forth. In this category are found applications performing both online and offline computations (e.g. large datasets processing or Web-index building).

Several studies on characteristics of DC traffic showed that DC traffic patterns are hard to predict and are strongly dependent on the type of DC and applications hosted [27, 79, 80]. Some commonalities, however, do exist: (1) most flows are small (e.g.$\leq 10kB$) and last short amounts of time. (2) The number of active flows per rack is very rarely over 10000 (due to work consolidation which optimizes the distribution of workloads and minimizes resource segmentation). (3) A high degree of oversubscription exists and it is not uncommon that uplinks from ToR show oversubscription rates ranging from 5 to 20 [27]. (4) Losses don't occur predominantly in highly utilized links, suggesting that unpredictable traffic bursts are a primary cause of losses in DCNs. (5) Although most flows are small, most bits ($\geq 90\%$) belong to large flows, showing that a significant amount of the bandwidth is used by large, long lasting flows (e.g. $\geq 100MB$). (6) Most data transactions are handled by protocols like TCP [41] and its variants [28].

The high variability and unpredictability of DC traffic is a major issue for DC applications and can heavily impact the DC performance [72, 81, 82, 83]. The highly

variable network performance in DC can result in wasting significant amounts of computational resources and the DCN ultimately becoming a computational bottleneck for the DC. Lastly, a well known issue of DCNs is *TCP Incast* [84, 85, 86]. Resulting directly from the TCP transmission dynamics, when multiple users attempt to access the same resources, TCP incast can lead to severe throughput collapse and can further worsen the network performance variability in DCNs.

Further details on current DCN issues and some of the proposed solutions can be found in Appendix E

Chapter 3

Next Generation Networks

In this chapter we will present some of the NGN proposals relevant to our study, namely, Optical Burst Switching (OBS) and Optical Flow Switching (OFS). Both of these approaches promise to improve significantly the energy efficiency of current network architecture as well as to provide more efficient ways to handle the the ever changing networking scenario and the rapid proliferation of bandwidth-intensive applications. Although potentially advantageous in terms of energy efficiency and resource utilization, both such approaches have their own drawbacks (Appendices A and B). An important drawback of both OBS and OFS is that they both require major architectural changes to the current networking architecture which, so far, no network operator has shown the will to undertake.

OBS uses large data bursts (e.g. $\geq 1Mb$) assembled at the network edge with traffic coming from multiple sources. Resources for the bursts are reserved over the data path using a control packet. The use of large OBS data bursts saves the energy that a standard IP network would spend to process the headers of the single packets assembled into each burst. OBS, however, makes extensive use of optical buffers [87], which to date do not offer a commercially viable alternative to their electronic counterparts [38]. Even when electronic buffering is used in burst switching [88], the energy spent to assemble the bursts at the edge of the network may negate the advantages achieved in the core. Another drawback of OBS is that given the amount of control traffic needed to operate OBS networks, the OBS control plane is prone to congestion [89] if the data bursts used are too small.

OFS gives access to the full network bandwidth to users able to occupy it for more than a certain time threshold. This can be very effective in terms of delay and bandwidth utilization when dealing with very large transactions (e.g. several tens of

GB), but the scalability of OFS to a large number of users is questionable. Moreover, any end user willing to access OFS services is required to install costly long-haul transmission equipment at their premises, limiting the application of OFS to a few specialized nodes.

To date no OBS nor OFS networks are commercially deployed and I doubt if anything has been published in recent years.

3.1 Optical Burst Switching

Optical Burst Switching (OBS) [90, 91] refers to a broad class of sub-wavelength switching architectures which share the same basic operations.In OBS data packets are assembled into bigger chunks called *bursts* at the network edge, in the electronic domain, and then the data bursts are forwarded through the network after first converting them in the optical domain. Each burst is then switched entirely in the optical domain, node by node, throughout its path from source to destination and is disassembled once it reaches the egress node of the OBS network. The nodes in each path are notified of the pending burst arrival by a control packet (called *Burst Header Cell* or BHC) which is sent through the data path before the burst leaves the source node (BHC and burst are spaced in time domain by an Offset-Time). Each of these control packets is associated with a burst and carries all the information needed by the nodes to handle the bursts without having to process the burst itself (i.e source/destination address, burst length, offset time just to name a few). Different implementations of the signaling or scheduling algorithms for OBS may require different kinds of information to be carried by the control packet. Based on such information the various nodes can configure themselves before the burst reaches them. When the burst arrives it is switched and buffered entirely in the optical domain.

3.1.1 Burst Assembly Techniques

Various techniques are used to assemble the bursts at the ingress of the network and the signalling process, in charge of configuring the network for the incoming bursts, starts only after assembly process is completed - at least in standard OBS implementations. Three main approaches are possible for burst assembly in OBS networks: *Timer-based*, *Threshold-based*, and *Mixed* approach.

3.1.1.1 Timer-Based Assembly Mechanism

The Timer-based approach defines a timer parameter (T_a) which is used at the edge node as a time limit for the burst aggregation process. All packets arriving within T_a are assembled into a burst and when T_a is reached the burst is transmitted. The advantage of this technique is that bursts are released at well defined time intervals which may simplify scheduling and signaling processes. When using this technique, however, the burst length depends on the traffic conditions: under high traffic load longer bursts are generated than under low traffic load. An important issue of timer-based assembly techniques is the choice of the timer value (T_a). If T_a is too small a great number of short bursts may flood the network and the amount of control information to be exchanged and processed will increase accordingly, overloading the control plane [89]. On the other side, if T_a is too big, the data will experience unacceptable assembly delay. The timer value must be chosen wisely considering the trade-off between the amount of processing required to the network and the amount of delay allowed.

3.1.1.2 Threshold-Based Assembly Mechanism

Another assembly mechanism used in OBS networks is referred to as *threshold-based*. In this approach a threshold value (T_h) is defined and used to limit the size of the burst that is being assembled. Once the size of the burst reaches the threshold value the burst is released by the ingress node through the network. During the burst assembly process the burst is stored in the electronic domain at the ingress node. The advantage of this technique is that the dimensions of the transmitted bursts are fixed and are not dependent on the network load. The drawback, however, is that the timing of the burst transmission and the delay of the burst assembly are not directly controllable, as bursts are released only when the threshold is reached, which may take a long time if the network is lightly loaded. In such conditions the assembly delay may become intolerable. Like in timer-based mechanisms, a similar tradeoff exists for the choice of T_h: if too small, the number of bursts will increase and so will the overhead. If the threshold is too big the assembly delay will become unacceptable.

3.1.1.3 Mixed Approach

A mixed approach takes advantage of the positive aspects of both timer and threshold based mechanisms, though at the cost of a higher operational complexity. Here an

optimal threshold is determined using the minimum burst length, while a timer value is used to define the maximum delay tolerance for the network. Once either one of these two conditions is reached the burst is transmitted.

3.1.2 OBS Signalling

Signaling is generally defined as that series of procedures and signal exchanges which controls a communication system. These procedures deal with establishing the path that the data bursts must follow and define those mechanisms to maintain and release paths and network resources associated to each burst. OBS signaling protocols use control packets traveling through the network to reserve the necessary resources for the bursts. Two main approaches are available and can be combined together (hybrid signaling):

1. *One-way Reservation*: also referred to as *source-initiated* reservation protocol. A control packet leaves the source node and reserves resources for the burst on the path from source to destination. If resources are not available at some node along the path, the burst will be dropped as it reaches the node that lacks the resources. This technique is generally flexible, relatively efficient, and provides a low latency for the overall transmission process.

2. *Two-way Reservation:* also referred to as *destination-initiated* reservation protocol. In this approach a control packet is sent from the source node to the destination node to collect information on the network state, topology, and resource availability without making any reservation. Once the control packet reaches the destination node, the information carried by the control packet is used to form an ACK packet which is in turn sent towards the source node along the reverse path to reserve resources for the burst. Upon receiving the ACK the source node transmits the burst along the reserved path. This approach offers a lower burst drop probability compared to the one-way approach, at the cost of a higher latency coming from the more complex reservation procedure. A time diagram of a two-way reservation protocol is shown in Fig. 3.1.

3. *Hybrid Reservation Protocol*: also referred to as *intermediate node initiated* reservation protocol. A control packet leaves the source to collect resource information until it reaches a certain node in the path. Upon receiving the control packet, the node starts a bidirectional reservation process which confirms

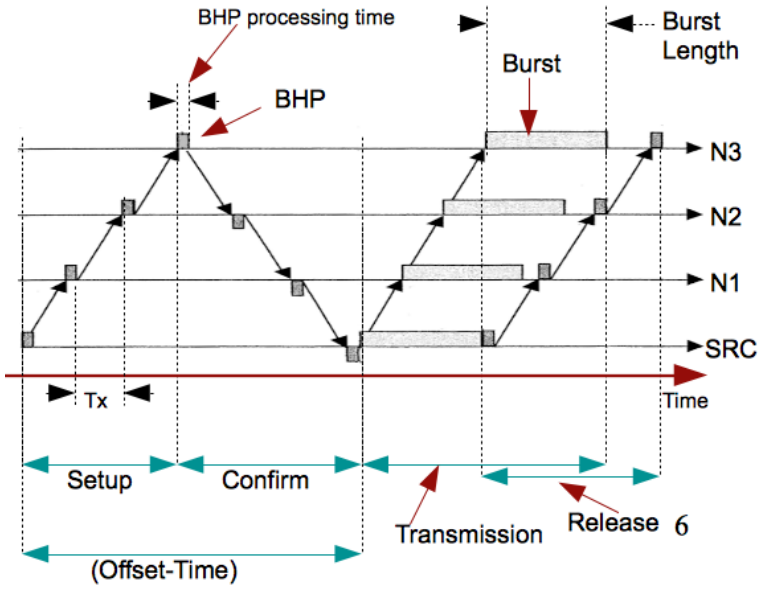


Figure 3.1: Two-way reservation protocol

the path reservation from this initiating node to the source node, and continues the reservation up to the destination node without notifying the source node the result of this part of the reservation process. A hybrid reservation procedure is shown in Fig. 3.2.

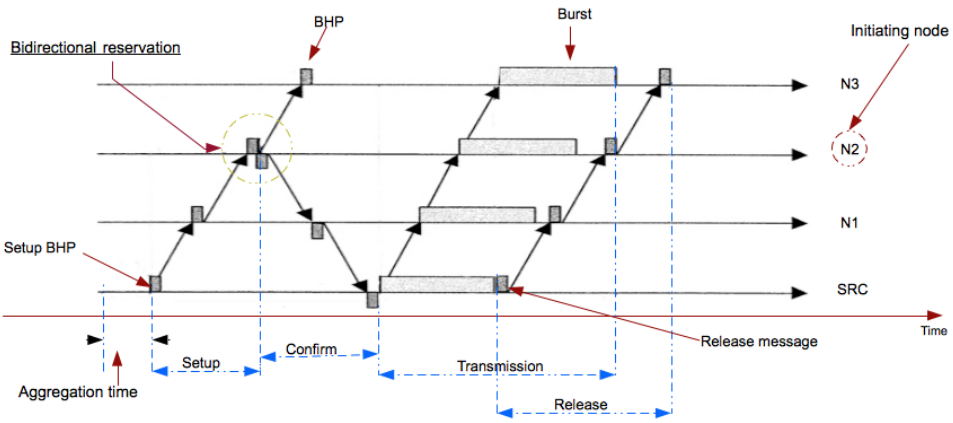


Figure 3.2: Hybrid reservation protocol

Usually one-way signaling protocols are used in OBS (e.g. Tell-and-Go or Tell-and-Wait OBS [92, 93]). In the remainder of this section, we will focus on this class of protocols. Before proceeding, some definitions are in order.

- A ***centralized signaling protocol*** is a signaling protocol which requires a centralized server to be in charge of setting up the routes and assigning the wavelengths for each data burst between each source-destination node pair within the network. This approach is more efficient when the network is small and the traffic not bursty.

- A ***distributed signaling protocol*** relies on the individual nodes to perform scheduling operations. Each node has a scheduler in charge of assigning wavelengths for each incoming BHC (and relative burst), and there is no need for any centralized request server. This approach is suitable for large networks with bursty data traffic.

The first signaling protocol we'll study is called *Just Enough Time* (JET) signaling [94] (see Figure 3.3). This is a distributed protocol, requiring no optical buffer or delay at intermediate nodes.

A control packet (BHC) is sent from the source node along the path on a specific control channel towards the destination node. The BHC is processed at each node, and the path for the associated burst, along with the necessary network resources, is reserved before forwarding the BHC to the next node in the path. The BHC carries the offset time information, which is updated at each node, enabling resource reservation at the *expected arrival time* of the burst. The expected arrival time is calculated with a *prediction algorithm*, which is one of the weak points of this technique due to possible mistakes in the prediction mechanism that may result in burst drop. The BHC also carries the information about the length of the burst. This way, automatic release of the resources is enabled and resource usage is optimized. A more accurate technique that doesn't rely on a prediction mechanism to calculate the expected arrival time is called *Time-Space Label Switching Protocol* and can be found in [95].

The next signaling protocol is called *Just In Time* (JIT) signaling [96]. This is a centralized protocol which requires network synchronization and a central scheduler in charge of managing the requests from the source nodes, processing the BHCs, and reserving the necessary resources for the data bursts. The centralized scheduler also informs the requesting nodes on which path to follow as well as the exact time to start transmitting each burst. Even in JIT signaling the nodes will be configured by the time the burst reaches them, but the reservation starts when the nodes are informed by the central scheduler. Resources are released when an explicit release message is received from the central scheduler. Therefore, the bandwidth utilization

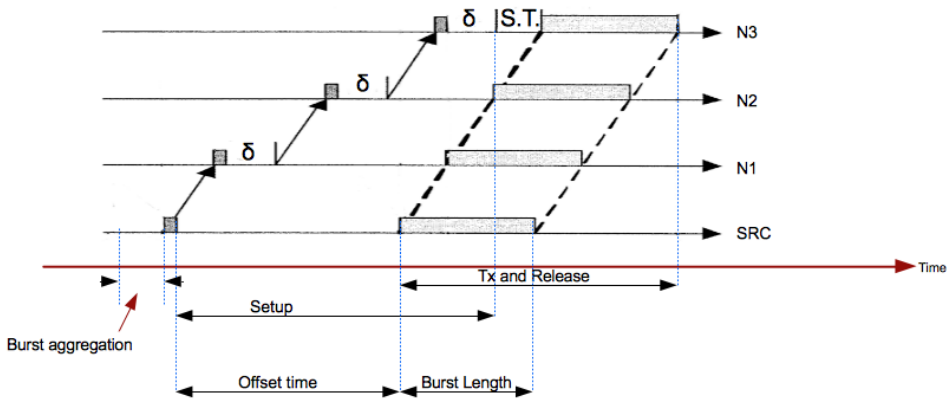


Figure 3.3: Just Enough Time signalling

is not optimized as in JET signaling (i.e. no delayed reservation and no automatic release). The advantage of JIT signaling is its simplicity: no information on the burst length is needed and the architecture is simpler to implement even if the centralized nature of this scheme makes it not very scalable. A schematic representation of JIT signaling is provided in picture 3.4.

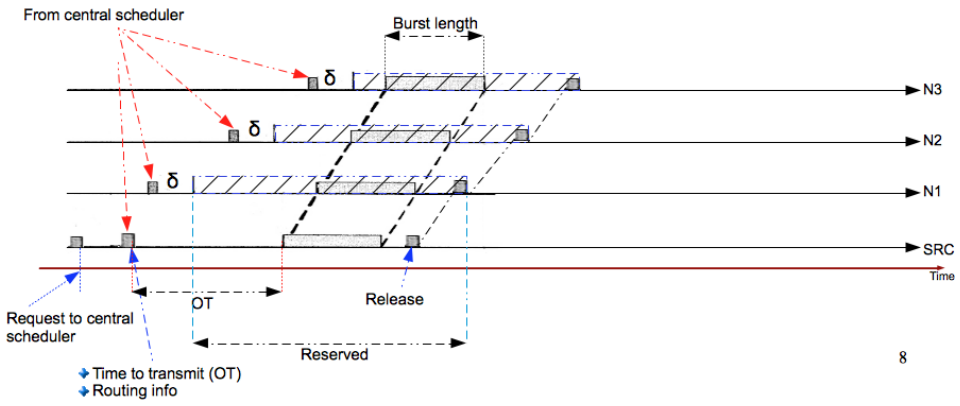


Figure 3.4: Just In Time signalling

3.1.3 OBS Scheduling

Scheduling aims at avoiding collisions, achieving optimal bandwidth utilization by scheduling the transmission of the various bursts, and solving the routing and wavelength assignment problem (RWA) for the network. Before going into the details

of the scheduling algorithms available for OBS networks, some definition are needed (refer to Figure 3.5).

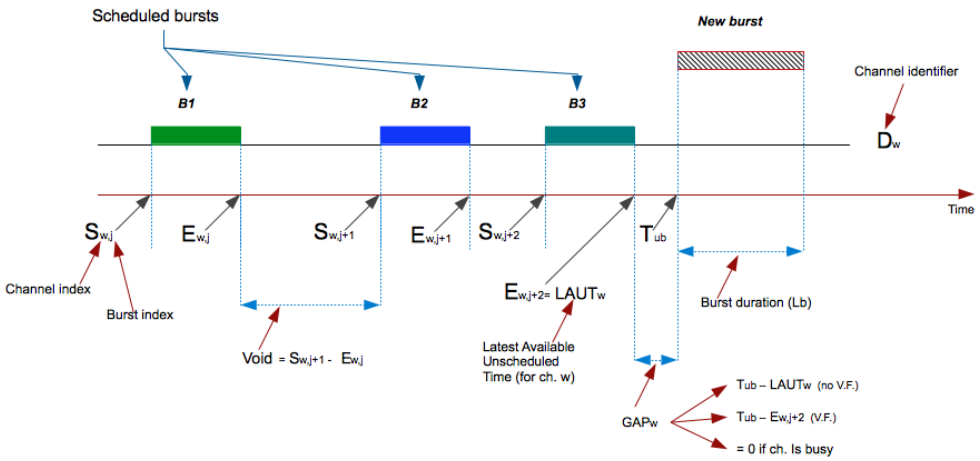


Figure 3.5: OBS scheduling: quantities definitions

Usually the channel scheduler obtains the burst arrival time and its duration from the information carried by the BHC. Other information that should be maintained is listed below:

- *Latest Available Unscheduled Time* (LAUT) for a specific channel. This quantity, also referred to as the *horizon* of the channel, is the earliest time at which the data channel is available for an unscheduled data burst to be scheduled.
- A *Gap* is the time difference between a previously scheduled burst and the arrival time of an unscheduled burst.
- A *Void* is the duration of the idle time between two scheduled bursts. Scheduling algorithms can be distinguished into two categories depending on if the voids in a channel are used to accommodate new arriving bursts (i.e. *Void-Filling* or *non-Void-Filling* scheduling).
- *"S"* and *"E"* (Figure 3.5) are the starting and the ending times for a scheduled burst.
- T_{ub} is the expected arrival time for the unscheduled burst.

3.1.3.1 First Fit (FF)

The various scheduling algorithms also differ from each other based on which data they keep track of. The first scheduling algorithm (which is also the simplest) is called First-Fit (FF). Here the only information that needs to be maintained by the scheduler is LAUT for each channel. Channels are scanned sequentially and, as soon as a suitable channel is found, the scan stops and the incoming burst is scheduled. If other channels could accommodate the burst, possibly leaving a smaller void, they are simply ignored since the channel scan stops as soon as a suitable channel is found, and no void size is computed. The strength of this algorithm is its simplicity and, as a consequence, its speed. From the resource utilization point of view, however, high resource wastage may occur. Since the channel voids are not considered for scheduling the bursts, the burst drop probability is higher with respect to other techniques where voids are considered. A schematic example of FF functioning is shown in Figure 3.6.

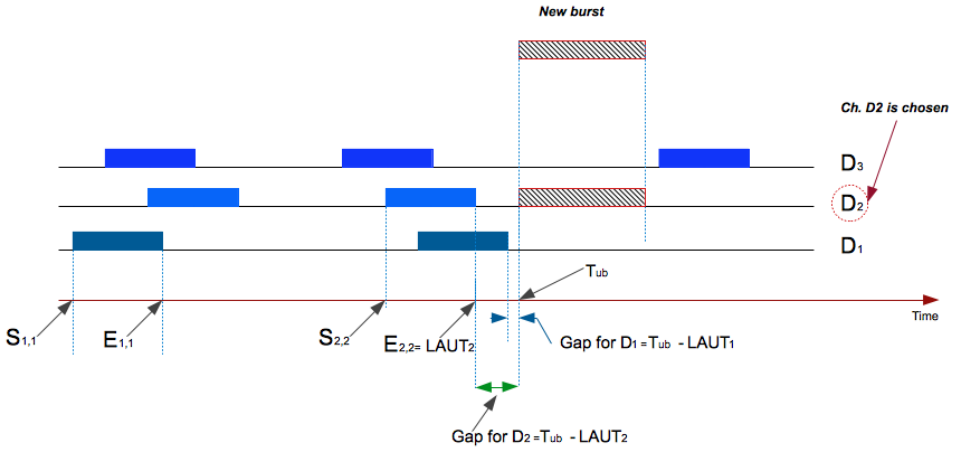


Figure 3.6: First Fit scheduling

3.1.3.2 Latest Available Unused Channel (LAUC or Horizon scheduling)

In this scheduling algorithm [97] the LAUT for each channel is maintained, and the channel with the horizon minimizing the gap between the last scheduled burst on the channel and the incoming unscheduled burst is chosen. In this algorithm all channels must be scanned. Therefore it is computationally more complex but nonetheless enables a better bandwidth utilization. An example is shown in Figure 3.7.

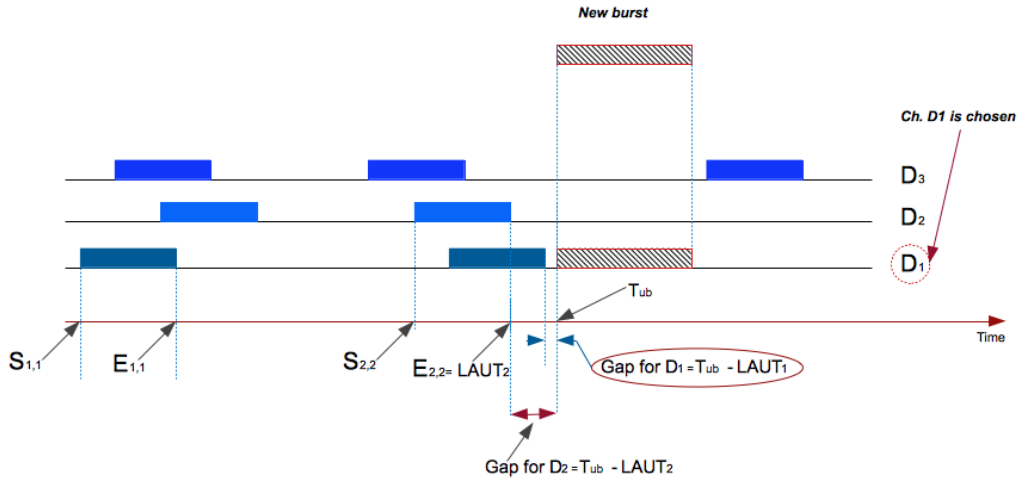


Figure 3.7: Horizon scheduling

3.1.3.3 LAUC with Void Filling (LAUC-VF)

In this algorithm [98], all channels are scanned in depth and more channel information needs to be maintained. Starting and ending times for each burst in each channel must be known in order to scan every void in each channel and to select the void minimizing the gap between already scheduled bursts and the new incoming burst. Figure 3.8 shows an example of LAUC-VF).

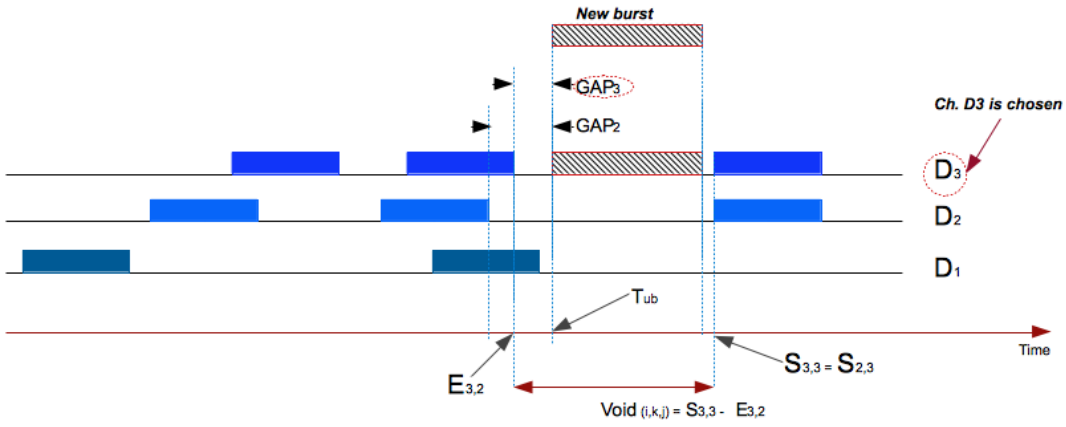


Figure 3.8: LAUC-VF scheduling

3.1.3.4 Non-Preemptive Minimum Overlap Channel with Void Filling (NP-MOC-VF)

In order to implement this algorithm another parameter needs to be defined (and maintained by the scheduler): the "overlap" parameter identifying which portion of an incoming burst overlaps with an already scheduled burst. This algorithm allows for burst segmentation, that is, discarding part of a burst in order for the burst to fit in a void smaller than its size if no suitable void (i.e. a void that minimizes the gap but is non-overlapping) is found. All channels (and their voids) are scanned and the amount of overlap is calculated (together with the gap) for each candidate void. The void minimizing the overlap (or the gap, if available) is chosen. The incoming burst is segmented (if necessary) and accommodated in the chosen void. Since the segmented burst has changed its length, the BHC information needs to be updated before it is sent to the next nodes in the path to avoid virtual contentions. The algorithm is shown schematically in Figure 3.9.

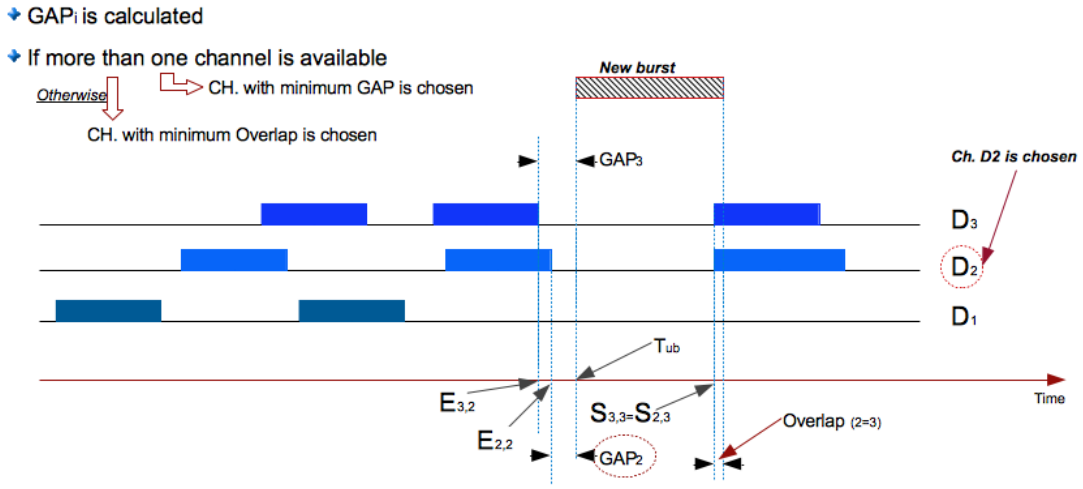


Figure 3.9: NP-MOC-VF scheduling.

The functioning of the algorithm when no void filling is used (NP-MOC) is shown in the figure below 3.10:

3.1.4 Contention Resolution

Due to the one-way reservation protocols usually implemented in OBS networks, the probability that two bursts will compete to use the same resources at the same time

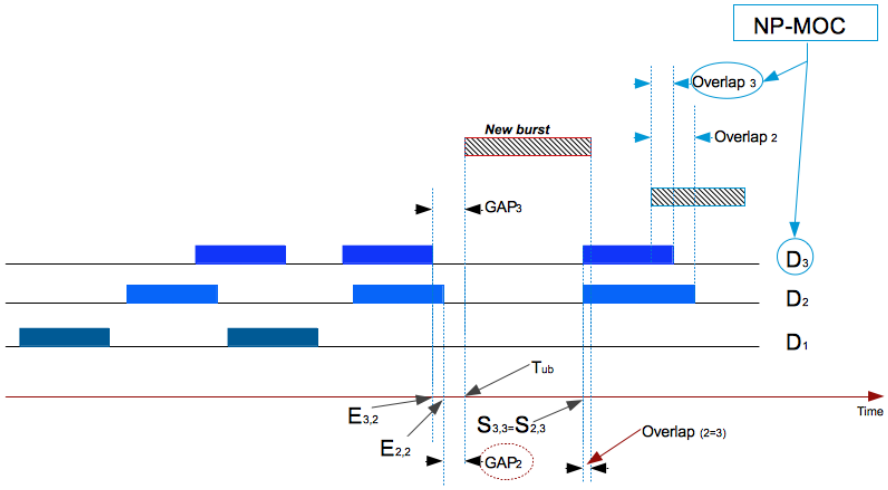


Figure 3.10: NP-MOC scheduling. In this case channel 3 is chosen for the incoming burst

is not zero. It is therefore necessary to provide the network with some contention resolution mechanism.

In an optical network at least three dimensions are available in order to resolve contentions: time, space, and wavelength. In case all other techniques fail and we are ready to accept some data loss, the burst segmentation technique used for scheduling can be adapted to resolve contentions. To resolve contentions in the time domain in an all-optical network, the only feasible solution at the moment is using Fiber Delay Lines (FDL). When two bursts contend for resources at a certain node one of them is routed towards a FDL in order to be delayed. Once a burst is inside a FDL it cannot get out before it reaches the end of the FDL, this being the first drawback of using FDL - delay is not precisely variable. Another issue arising from the use of FDL is the signal impairment the burst will experience by traveling in the FDL. This may require optical amplifiers or expensive signal regenerators. Besides the cost, another negative aspect of using FDL is the space occupancy which may need hundreds or even thousands of kilometers of optical fiber (depending on the line speed); not always an easy case to accommodate.

When using the space dimension for contention resolution a possible solution is called *deflection routing*. One of the two contending bursts is routed toward another path or through a circular path that will let it come back to the node after a certain amount time. The obvious advantage of this technique over the use of FDL is that, in this case, no expensive optical buffers are required. The drawback, however, is

that deflecting a burst in the network will increase the traffic, occupy resources, and therefore decrease the network efficiency both in terms of resource usage and delay experienced by the burst (the alternative path may be too long or the burst may find itself trapped in a loop).

To resolve a contention in the wavelength domain, we need to use wavelength converters to convert one of the two contending bursts into another wavelength so that they can share the same resources (port). With this technique no additional delay is added, but wavelength converters are expensive, especially if they have to be fast and tunable. Burst segmentation may be a good solution for applications with stringent delay requirements if data loss is acceptable. Moreover, in certain cases, it may be more efficient to loose part of a burst than having to retransmit it entirely from the source. When using segmentation the burst is divided into segments (see Figure 3.11), each with a unique header. If preemption is enabled among the contending bursts, it is also possible to resolve the contention in a prioritized manner, giving the possibility to high priority bursts to preempt low priority ones, protecting high priority traffic from loss due to segmentation. Various segmentation mechanisms are available depending on which part of the burst is dropped: *head dropping* when packets on the initial part of the burst are dropped, *tail dropping* when packets on the final part of the burst are dropped, or a combination of the two (in this case the segmented burst loses packets from both head and tail). Figure 3.11 shows the burst structure needed in order to use segmentation.

When using segmentation each segment of a burst needs to be identified (ID, length) increasing the overhead. The choice of the segment length is also important; the number of packets lost per contention is proportional to the segment length while the overhead is proportional to the number of segments per burst. A tradeoff exists between these two parameters. Generally, contention resolution mechanisms are constrained by the specific signaling and routing protocol implemented in the network. Sometimes it is more convenient to reduce contention probability by wisely designing signaling and routing protocols to avoid contentions as much as possible [95].

3.1.5 Architectural Overview

An OBS network is a series of OBS nodes interconnected by WDM links. The functions of the ingress are described below:

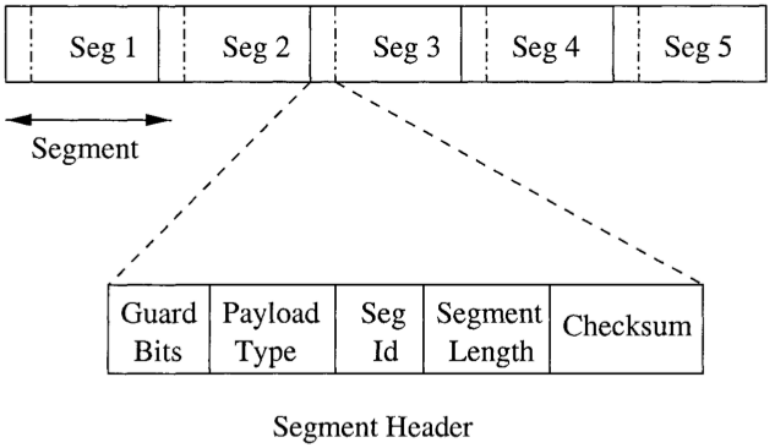


Figure 3.11: Burst Segmentation: burst structure [99].

1. *Packet pre-sorting:* in order to support *Quality-of-Service* (QoS) various incoming packets can be placed in various queues at the ingress node according to their class of traffic and QoS requirements. They'll be assembled within bursts accordingly.

2. *Packet buffering.*

3. *Burst Assembly* according to the particular assembly mechanism used in the network (see section 3.1.1).

4. *Routing and Wavelength Assignment* (RWA).

While the egress node implements the following functions (note that the distinction between *ingress* and *egress* node is relative to the data flow direction):

1. *Burst disassembling:* upon receiving a burst the egress node disassembles it into it packets before routing them to the higher layers of the network.

2. *Packet forwarding.*

A scheme showing the architecture of an OBS edge node is shown in Fig. 3.12

The scheduler assembles the bursts according to the particular assembly policy in use and schedules the bursts (locally). The routing module selects an appropriate queue of the burst assembler module for each packet (pre-sorting). The queues are used to sort the packet before assembling them into bursts.

Some of the key functions implemented by core nodes follow:

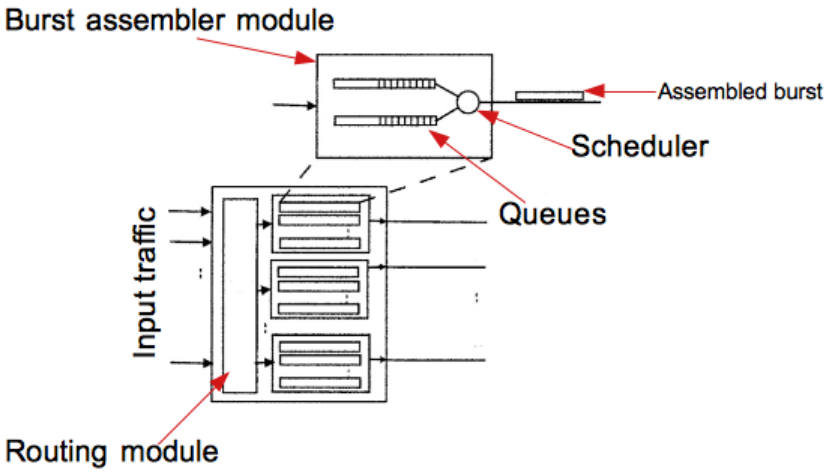


Figure 3.12: OBS: edge node architecture [99].

1. *Burst scheduling* on core links.
2. *Burst switching* according to BHC's informations and scheduling policy (see Section 3.1.3).
3. *Contention resolution.*
4. *Signaling.*

The architecture of an OBS core node is shown schematically in Figure 3.13.

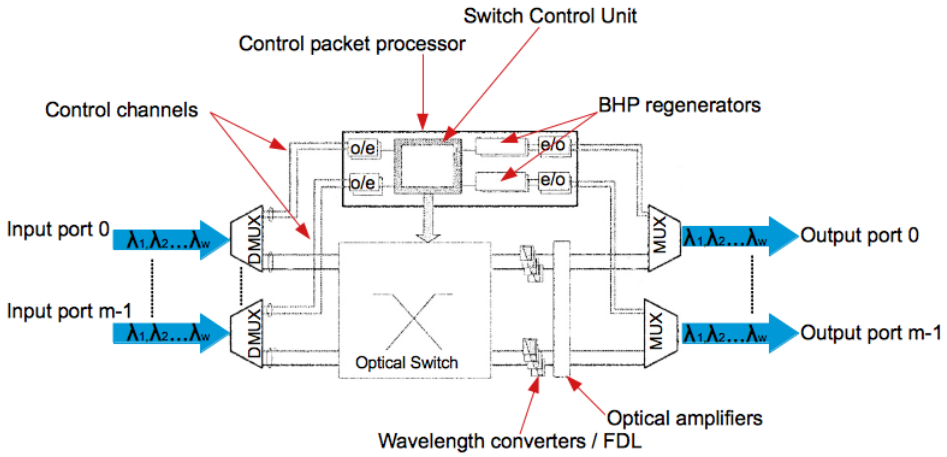


Figure 3.13: OBS: core node architecture [99].

A WDM signal enters the node and the various wavelengths are demultiplexed by the input demultiplexers. Then the control wavelengths are routed towards a Control

Packet Processor to be processed, while the data wavelengths are routed to the optical switch. The optical switch switches the various wavelengths according to the control plane information before routing them to the output multiplexers, which reassemble all the wavelengths, reforming the WDM signal before routing it to the next node in the path. From the control plane point of view, there is a Control Packet Processor made of O-E converters and BHP regenerators to regenerate the BHC according to the information coming from the Switch Control Unit (SCU). The functions of the SCU are listed below:

- Create/maintain a forwarding table
- Forwarding table lookup
- BHC processing
- BHC rewriting
- Wavelength conversion control
- OXC configuration

3.1.6 Reliable OBS: techniques to increase OBS networks reliability

In this section we will present the various techniques available to increase the reliability of an OBS network. Some of these techniques, such as contention resolution techniques, have been previously treated in the former sections, whereas others like admission control and load balancing are common among many kinds of networks and need not to be treated here extensively. Instead we will focus on loss recovery mechanisms which can be used in OBS networks.

Loss recovery mechanisms can be divided into two main categories: *Reactive mechanisms* and *Proactive mechanisms*. Reactive mechanisms use explicit failure messages to trigger the recovery process. One commonly used technique is retransmission. When a burst is dropped at one intermediate node, this node notifies the source node which keeps a copy of the burst and, upon receiving the failure notification, simply retransmits the burst, if necessary, over another path. Proactive mechanisms try to manage fault situations by including additional information within the data flows in order to let the failure be locally managed without need to involve the source node

and/or retransmissions. The first proactive technique is widely used in many networks (including IP networks) and is called Forwarding Error Control (FEC). This procedure adds check bits to the data burst which enable the receiver to correct (within certain limits) the errors due to transmission within a data burst. The main drawback of this technique is that part of the bandwidth (BW) has to be used to transmit the check bits, which could be a problem in networks with BW shortage, though this is not a big issue in optical networks where the BW is usually abundant and the retransmission delay may be considerable (for example, long haul connections). If the check bits are redundant and placed properly in the data burst, then FEC can be successfully combined with segmentation. This is particularly true when we have complete information on which kind of segmentation is used within the network and we can place the check bits accordingly(see picture 3.14).

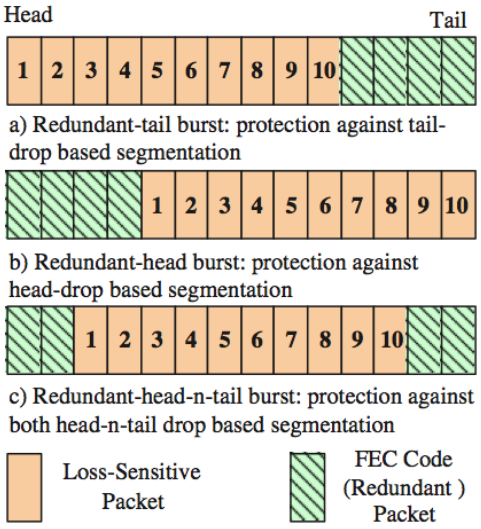


Figure 3.14: Reliable OBS: Burst Drop protection techniques [90]

.

Another proactive mechanism which is possible to use in OBS networks is called $1+1$ protection (see Fig. 3.15), this method is widely used even in other networking technologies such as GMPLS, MPLS, and so on. Here the high priority traffic is duplicated at the ingress node of the network and routed over two disjoint paths in the network. The main drawback of this technique is that the amount of resources used is at least doubled (only in the case when the protection path has the same length of the original one), and the overall amount of traffic in the network is obviously increased.

Furthermore, a mechanism to recognize redundant data has to be implemented at the egress node in order to not forward repeated data to the higher layers of the network.

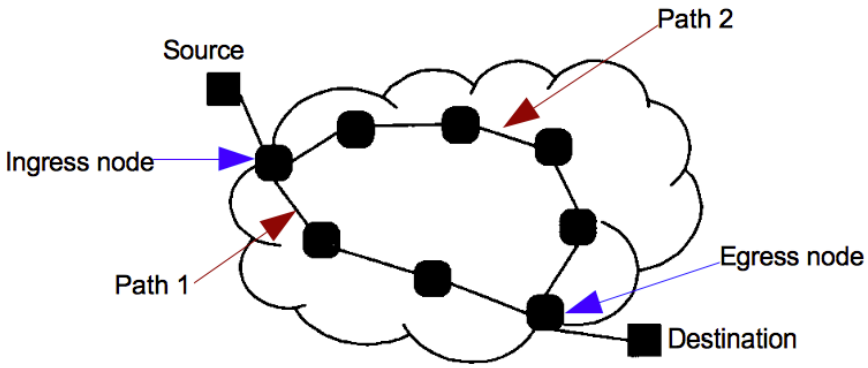


Figure 3.15: Reliable OBS: Burst protection techniques.

The third proactive technique presented is called *Composite Burst Assembly* and is implemented at the source node where burst assembly takes place. Burst are assembled by placing the less loss-sensitive traffic at the boundary (beginning, end or both) of the burst. In this way if a burst undergoes segmentation, only the less loss-sensitive data is lost while the loss-sensitive data is preserved. A good aspect of this method is that no additional overhead has to be added to the data burst, but this comes at the expense of a slightly longer burst assembly delay.

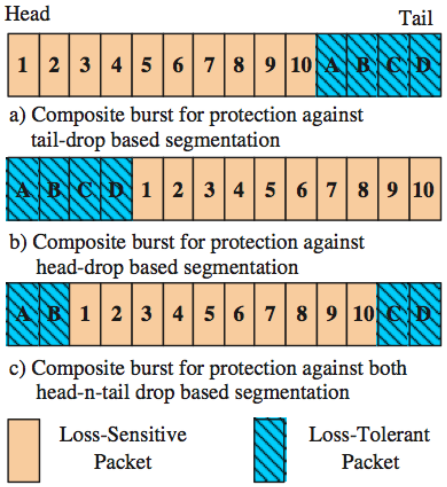


Figure 3.16: Reliable OBS: Burst Drop protection techniques [90]

.

The last mechanism is similar to 1+1 protection with the difference that duplicated bursts don't necessary follow disjoint paths, bursts are also not necessarily duplicated at the ingress of the network, and more than one copy of the burst can be done. This technique goes under the name of *Burst Cloning* and is schematically shown in Figure 3.17.

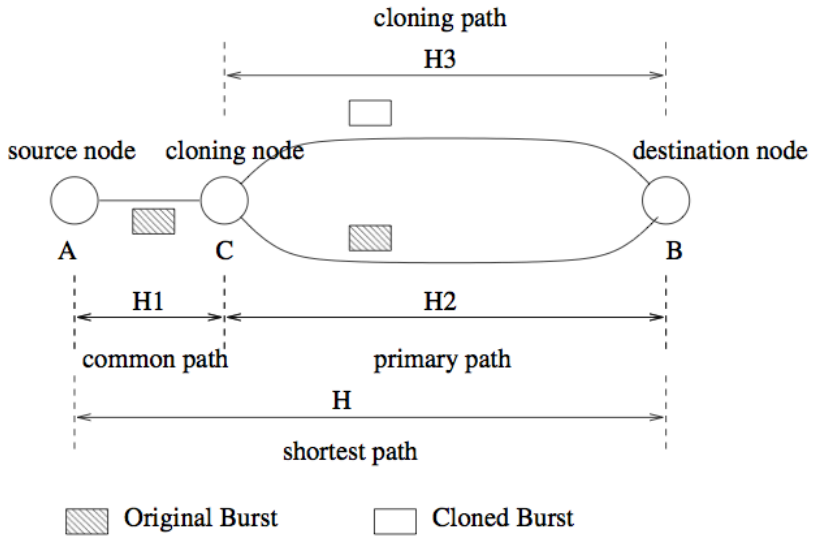


Figure 3.17: Reliable OBS: Burst Drop protection techniques [100]

.

Having many clones of a burst in the network increases the probability that the data is correctly received but also increases traffic and usage of network resources. Another issue with burst cloning is that since the various clones are not bound to follow disjoint paths, it is possible that the "original" burst may find itself in a situation where it has to compete with its own clones for the same network resources. This can be avoided by implementing a preemptive traffic isolation mechanisms which give preemptive priority to the original burst over its clones at every node. Another problem, however, analogous to the case of $1+1$ protection, involves implementation of a method to distinguish original data from cloned data at the destination node to avoid forwarding redundant data to higher layers of the network. To achieve this, a packet ID for each packet in each burst must be kept at the destination node, increasing the amount of buffer space needed. The network node in charge of cloning the burst much be chosen wisely. While a shorter common path (H_1 in the picture above) will increase the probability that a burst is not lost before being cloned, the cloning path will be longer thereby increasing the probability that the clones are lost.

A good policy is to choose the cloning node on the shortest path between source and destination node ($H_3 - H_2 \geq 0$, See Figure 3.17).

3.2 Optical Flow Switching

Optical Flow Switching [33] is a transport technology enabling end-to-end transparent transport service using backbone bandwidth for short time durations. OFS is best suited for those users who can fully utilize the bandwidth provided by a backbone wavelength for time durations greater than roughly $100mS$. Such users are expected to contribute significantly to future network traffic. OFS users are given direct access to the core bandwidth. For small transactions, the burden of the necessary network management and the cost of the equipment that end users are required to have, such as (at least) one long-haul tunable transceiver (for data transfer) and a short-haul tunable transceiver (to communicate with the scheduler), will exceed the benefits of the OFS approach. Two positive aspects of OFS are that it is possible to implement using technologies commercially available today as well as the absence of core buffering. The drawback is that considering the expense (both computational and economical) required to use OFS services and the nature of OFS-based transport (i.e. use of large amount of dedicated bandwidth for a single transaction), makes it suitable only for those applications needing to transfer consistent amounts of data in a relatively short time (such as distributed storage services, data gatherers, on demand transactions with large bandwidth requirements, and grid/cloud computing and so on). A picture showing the possible organization of an OFS network is shown in Figure 3.18

Each Metropolitan Area Network (MAN) is assigned a scheduling node to which the end users send their requests to use OFS services via an Electronic Packet Switched (EPS) network, supporting the OFS control plane (as in OBS, also here control plane and data plane are kept separate). Flows are aggregated at the end user or, in case many users wants to use OFS services, it is possible for them to concatenate their data before transmitting, in order to fully utilize the bandwidth provided by the OFS network. To minimize the network management, switch control, and overhead the flows are treated as indivisible entities all along their path. It is often assumed that there is no wavelength conversion along the path (i.e. *wavelength continuity*). The control plane works based on two kinds of processes executed

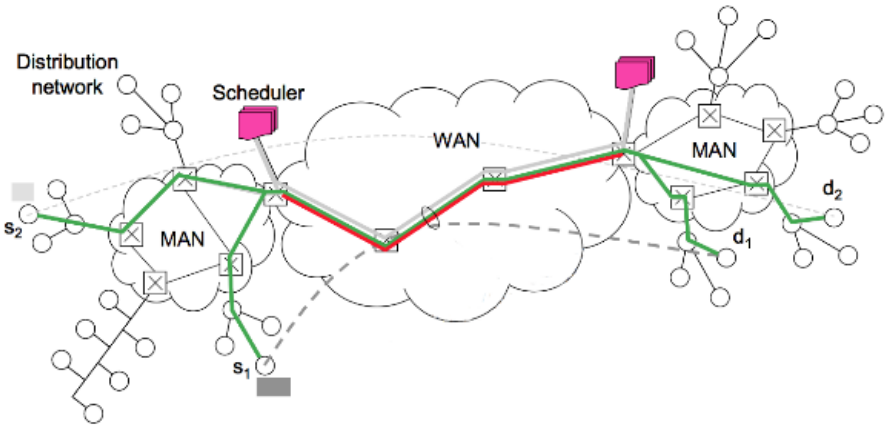


Figure 3.18: OFS network [101].

at different layers of the network. In the Wide Area Network (WAN) centralized processes are in charge of the following functions:

- Gathering network state information.
- Computing the routes for each connection. The assumption here is that the traffic in the core network is efficiently aggregated and large and substantial enough that these paths can be considered as varying only on coarse time scales (in the order of seconds or minutes or more, i.e. a quasi static logical topology is assumed).
- Disseminating the computed routes together with the network state information to all nodes of the network.
- Reconfiguring the WAN (according to the traffic assumption made above, also WAN reconfigurations take place on coarse time scales).

In the MAN, distributed processes are in charge of scheduling individual transactions and exchanging information for the MAN physical layer reconfiguration (this will occur in time scales in the order of mS and requires a distributed process in the network).

3.2.1 Physical Layer Organization

Figure 3.19 shows the physical layer's hierarchical organization of an OFS network.

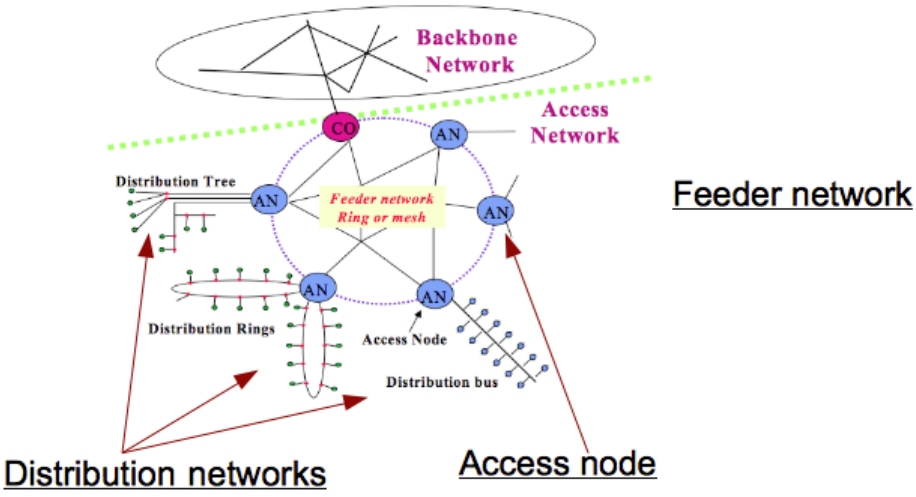


Figure 3.19: OFS network [33].

Core nodes are equipped with bufferless OXCs. This fact represents a significant scalability advantage over electronic packed switched (EPS) networks since queuing systems are the main bottleneck of EPS systems as the number of lines and ports increase. The drawback is that the absence of buffers in the core nodes together with the assumption of indivisibility of the flows makes it hard to efficiently utilize the resources of the network (Appendix B). The use of preemptive mechanisms for transactions with different QoS requirements may help to solve this issue. The feeder network may be implemented using various topologies (ring or mesh usually). At this layer of the OFS network, the most expensive optical devices are placed (such as OXCs, tunable filters, optical amplifiers and OADMs). This part of the network serves more users than the distribution network. It is therefore more cost efficient to place the expensive hardware here and leave a passive structure for the distribution network. Other functions of the feeder network involve monitoring network status (to detect and recover failures for example), and network control and management functions. The access node runs a media access control (MAC) protocol to regulate the access to the physical media and network services, carries on routing functions within the access network, and functions as a scheduler for data flows. Distribution networks simply connect the end systems to the OFS infrastructure using mainly passive components, and usually have a broadcast structure.

3.2.2 OFS Scheduling

The scheduling algorithm is presented referring to Figure 3.20. Some assumptions on the structure of the network have to be made. The first of such assumptions is that within the graph shown in Figure 3.20, there is an embedded tree which is used to transport inter-MAN traffic under normal working conditions (in failure conditions OFS traffic can be carried out of the embedded tree). The embedded tree is shown in Fig. 3.21.

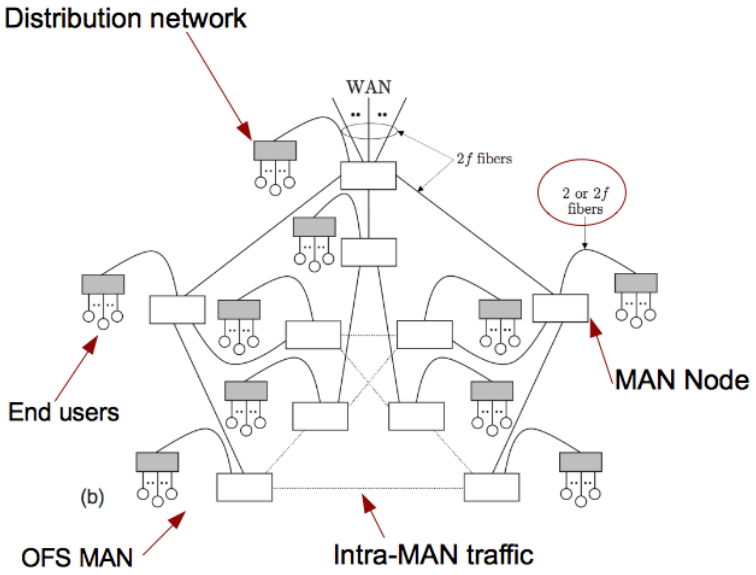


Figure 3.20: OFS topology [102].

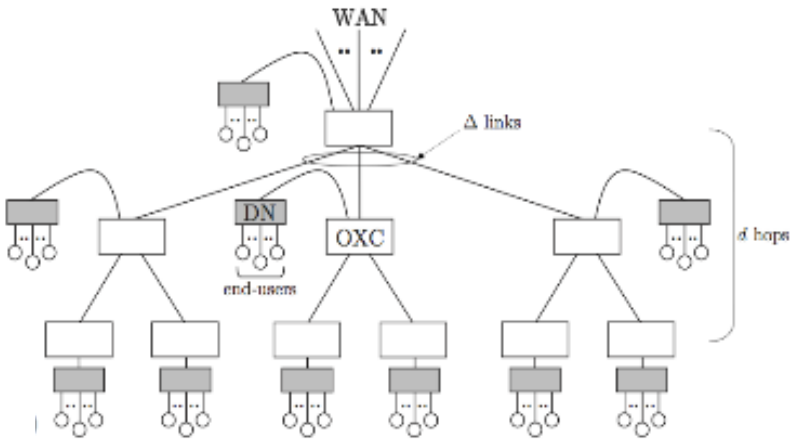


Figure 3.21: OFS topology: embedded tree [102].

The grey boxes in pictures 3.20 and 3.21 above are the *distribution networks* (DN) which connect the users to the OFS infrastructure. DN can connect to MAN nodes using two types of configurations:

1. Using 2 fibers: this choice will require less hardware and proves to have good scalability since no modification in the physical connections are needed in case of network expansion.

2. Using $2F$ fibers: F in each direction (upstream/downstream), hosting within them all the wavelength channels provisioned for inter MAN communications and carrying various kinds of traffic types besides OFS traffic (Note that OFS networks, being *application-oriented* technology, need to coexist with other transport technologies). Obviously this choice requires more hardware and proves to be less scalable with respect to the 2 fibers solution.

Another assumption that greatly reduces the computational effort of the scheduling algorithm is that for each WAN channel (i.e. wavelength) reserved for inter-MAN communications there exists a dedicated wavelength in each link of the embedded tree in both source and destination MAN. This avoids resources being reserved in the WAN if no resources are available in the MAN, which would result in wasting precious backbone capacity. It is also assumed that aggregated traffic (i.e. optical flows) is generated at each end user machine. This allows the further assumption that the traffic generated for a particular source-destination pair arrives according to a Poisson process. The last assumption is that a significant multiplexing of flows occurs in each MAN, resulting in a quasi-static WAN logical topology with changes occurring on time scales in the order of many flows (this allows considering as *static* the wavelengths provisioned for inter MAN communications). With reference to Figure 3.22 , let's now see an example of an OFS scheduling algorithm [102].

The scheduling algorithm is implemented through the following steps:

1. Data flows are formed at the end-user's machine in the source DN (D_s)

2. The user, willing to transfer his data, sends a primary request ($R\omega$) to the scheduling node relative to its MAN (i.e.: the source MAN, M_s- of Figure 3.22). At the M_s scheduling node there is one queue (primary queue) for each possible destination MAN (M_d), and these queues are modeled as $M/G/\omega_m$ queues systems, where ω_m equals the number of dedicated wavelengths for inter MAN communication.

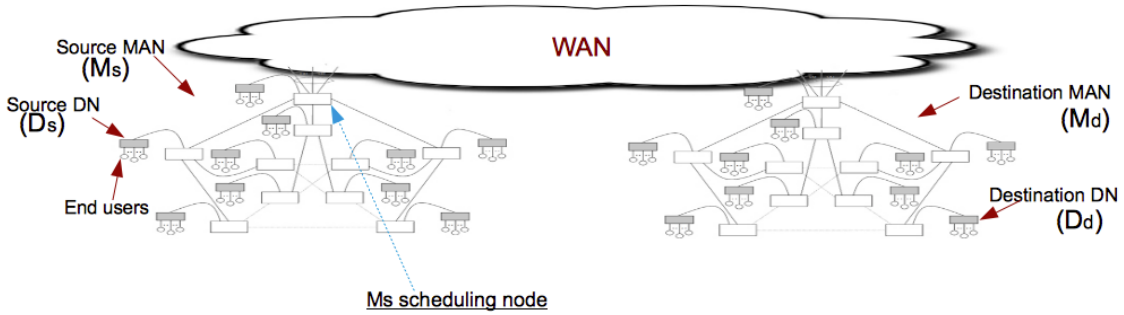


Figure 3.22: OFS Scheduling

.

3. Once $R\omega$ reaches the head of the of the primary queue, the flow is assigned a wavelength among those reserved for inter MAN communications.

4. An all-optical path is established from the edge of the source DN (D_s) to the edge of the destination DN (D_d). Note that due to the assumptions previously made, this path is guaranteed to exist.

5. Two secondary requests (R_s and R_d) are sent and stored into secondary queues (located at the M_s and M_d scheduling nodes respectively).

6. Once R_s and R_d reach the heads of the their respective queues, an outgoing channel on D_s and an incoming channel on D_d are reserved.

7. M_s and M_d scheduling nodes are notified.

8. End users are instructed to begin transmission.

9. Once transmission is finished the various requests made during the whole process (that is: $R\omega$,R_s and R_d) leave the head of their queues.

Figure 3.23 shows the results of the performance analysis of the proposed scheduling algorithm:

The algorithm presented above was tested for two working conditions:

- *Optimistic:* in this case the secondary requests are sent simultaneously after the primary request reach the head of its queue.

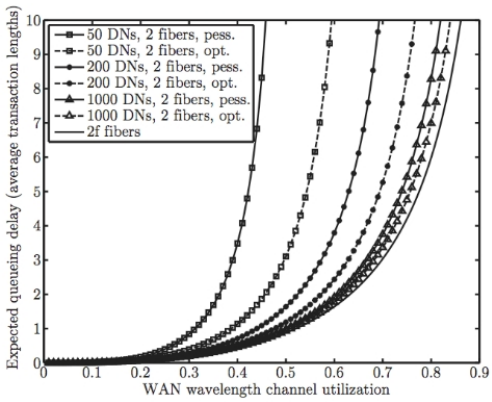


Figure 3.23: OFS Scheduling: performances [102]

.

- *Pessimistic:* in this case the secondary request for D_d is sent only after the request for D_s has reached the head of its queue.

OFS path setup mechanisms are two-way mechanisms involving a combined use of centralized and distributed processes occurring at different time scales. This technique tries to achieve path setup in times that are only a little longer than one round trip time (RTT) plus hardware configuration and control information processing times, by using a combination of slow centralized processes and fast localized processes together with a one-way reservation mechanism. A Central Network Manager (CNM) gathers, updates, and periodically broadcasts link state information (i.e. slow, centralized processes are executed in the CNM with changes at coarse time scales). This information is used by the CNM to compute routes to be used for data transmission by the various nodes in the network. These paths are used by the source node to form a path request packet, which is sent over the network from source to destination node to reserve resources for the flow node by node (fast localized processes). The various nodes on the path may "suggest" any of the free wavelengths available in the path, and if resources are available an ACK message is forwarded to the next node in the path and resources are reserved for the flow. Otherwise no message is forwarded and the resource reservation fails. If more than one path request reaches one node at the same time, claiming the same resources, the node operates its choice based on the request priority and the number of nodes where reservation was successful, thus far: the longest survivor has higher priority. When multiple requests reach the destination node (each one with a different path reserved), the destination node chooses the proper path and notifies the source, which then begins transmission while another

message is sent by the destination node to release unused resources. The CNM is also notified in order to update the list of available resources.

Since OFS is still in its development phase many aspects of its implementation remain unexplored. A lightweight transport layer mechanism (UDP-like) has not yet been developed for OFS networks. Another important unsolved issue is the implementation of a flow error check mechanism. Since OFS is not suitable for all users, it has to be integrated in networks supporting other kinds of transport mechanisms. An implementation that enables the coexistence of OFS with other kinds of networks has not been developed. Likewise, algorithms and protocols capable of computing and disseminating network resources and information within time frames required for resource configuration over commercially deployed networks (at the architecture level) are yet to be implemented.

Chapter 4

Big File Protocol

4.1 General concept and functioning principle

As discussed in Section 1.1, network traffic is dominated by bandwidth-intensive applications, often requiring strict delay control [1]. Nowadays most bandwidth-intensive transactions are handled by protocols such as TCP [41] or UDP [42] which were not designed for this type of traffic. As a result, network traffic is bursty and hard to predict[1], forcing network operators to factor in significant bandwidth headroom to accommodate bandwidth peaks.

In current IP networks, large files are segmented into a large number of small packets which are routed and processed individually at each node in their path to destination. Dropped packets are also processed up to the dropping node, using up network resources (bandwidth, buffer space, and computational) and increasing network load regardless of their effective delivery. As the number of bandwidth-intensive applications continues to grow [1], the computational load placed on network routers by large transactions also increases, ultimately constraining their evolution.

While many alternative networking approaches and architectures have been proposed (Chapter 3), the traditional resistance of network operators to changing their infrastructure, as well as the investment required by large scale architectural redesigns, limited their applicability, and none of them has found their way into commercially deployed networks.

[1]Note that this is not due to the nature of TCP itself but to the fact that, due to congestion, TCP packets can be dropped at any point in the network in an unpredictable manner.

Our idea is to devise a protocol that can work in concert with the current infrastructure and provide it with the means to handle efficiently bandwidth-intensive transactions, without requiring major architectural redesigns.

The key idea is that with using transport structures specifically designed for large transactions, it is possible to handle such transactions at lower layers (e.g. L1/ L2), thereby skipping most of the per-packet processing necessary when using traditional IP protocols, and opening the possibility for considerable power savings and resource usage optimization (Appendix A).

In our approach, large transactions are handled separately from the rest of traffic over an overlay network (Appendices A and B) that is built by reserving a portion of the available bandwidth on each link of the pre-existing network. This overlay network is used for large file transactions only, avoiding any potential issue coming from coexistence of the proposed approach with legacy protocols [103]. The capacity of each link of the BFP overlay network can be planned offline and based on the particular needs of each network (or subnetwork). Assuming that ODUflex [60] channels are used to setup the aforementioned overlay network, any capacity adjustment can be done hitlessly using currently available OTN [60] functionalities [65]. From this point on we will refer to our approach as *Big File Protocol* or BFP.

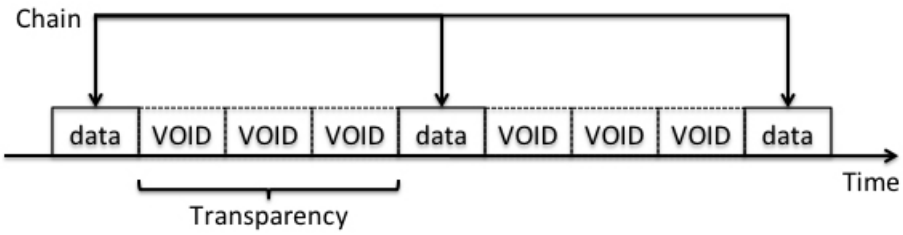


Figure 4.1: BFP Data Chain

.

4.1.1 BFP transport structure (*Chain*)

In BFP large files are organized into periodic, semi-transparent transport structures referred to as *chains* (Figure 4.1). A chain is assembled by segmenting each file into a set of fixed-size *data frames*. At transmission time, data frames are interspersed with *void frames*, each the same size and structure as data frames but carrying only stuffing bits. The number of void frames between two consecutive data frames is

fixed and remains constant for the entire chain. The resulting transport container used for a transaction is a periodic structure, alternating between data frames (carrying payload bits) and a defined number of consecutive void frames (carrying stuffing bits). Void frames are used only to intersperse data frames to make the chain periodic, and are always preempted by data frames, never processed or buffered, but discarded upon reception, and added in between data frames again at transmission time. Both void and data frames are assembled using (smaller) atomic frames, whose size can be decided on a case-by-case basis and left as a design parameter. We refer to these atomic frames as *Basic Payload Frames* and *Basic Void Frames*, or BPF/BVF. These frames are sized according to the underlying transport hierarchy (e.g. OTN or Carrier Ethernet [104, 105, 106]) to ease mapping of BFP chains and to avoid further frame fragmentation (Appendix C). Both BPF and data frames are numbered independently using a binary counter, which allows selective retransmission of single BPF in the event of transmission error. A data frame assembled using BPF is shown in Figure 4.2.

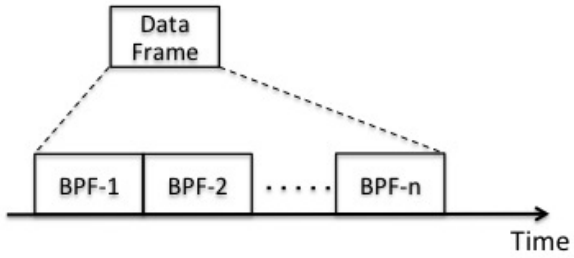


Figure 4.2: Data frame assembled with multiple atomic data frames (BPF)

.

The capacity of each (outgoing) channel is modeled in the control plane by a set of timeslots, each of the same duration of a BPF at the channel bitrate. This channel model is implemented only in the control plane and is used by the resource reservation algorithm to schedule incoming transactions (Section 4.1.2). Time Division Multiplexing does not need to be supported by the underlying hardware.

As a result of the use of chains as the BFP transport container, information about the configuration of an entire transaction can be condensed into a few parameters: (1) the size of the data/void frames (*frame size*) expressed as the number of BPF/BVF in each data/void frame; (2) the period of the chain, defined as the number of voids between two consecutive data frames *plus* 1 (this is equivalent to the number of chains that can be interleaved onto the same channel), referred to as the *Transparency Degree*

(TD) of the chain; and (3) the size of the transaction (or *chain length*), expressed as the number of data frames per chain.

As all the frames of a chain go through the same path, and require the same processing in terms of buffering and switching, the configuration of a chain also determines its average bandwidth occupancy onto an outgoing channel (Equation 4.1).

$$\overline{B}_c = \frac{f_b * L_{Ch}}{f_s * [TD * (L_{Ch} - 1) + 1]} \tag{4.1}$$

where:

f_b size (in bits) of a data frame (including any control/assembly overhead)

L_{Ch} number of data frames per chain (i.e. chain *length*)

f_s duration (in seconds) of a data frame at the (output) line rate

TD Transparency Degree

Given a transaction size of "T" Bytes, the chain length (L_{Ch}), can be computed using Equation 4.2, below:

$$L_{Ch} = \left\lceil \frac{T}{F_s} \right\rceil \tag{4.2}$$

where:

T Transaction size [B]

F_s Data frame (payload) size [B]

One of the keys to avoid unnecessary processing of individual data packets is to embed in each transaction as much information as possible about its configuration. In BFP this information is made implicit with the use of chains (fixed periodicity and frame size), thereby avoiding most of the need for per-frame processing - otherwise necessary if non-periodic structures are used (e.g. [107]). Chains can be delineated by a receiving node solely using the information about their configuration, which is communicated in advance to each node in the path by a Control Packet (CP). From this perspective, using MPLS terminology [108], the TD of a chain can be thought as an *embedded forwarding equivalence class* for the frames of each chain, and the distribution of chain configuration information over a path by means of a CP loosely resembles MPLS Label Distribution Protocol [109].

A chain maps an entire transaction into a predictable data structure which can be signaled through the network using a small CP, and switched with minimal or no per-frame processing (See Appendix B and C). This enables the handling of chains using almost exclusively the functionalities available at the lower layers of the network (L1/L2), where the cost per bit is the lowest, while also skipping most of the per-packet/per-frame processing necessary when using traditional IP protocols. As described in more detail in Section 4.1.2, a CP carries information about configuration and timing of a chain. Whenever a chain is ready for transmission, the CP is sent over the network to the destination node to reserve resources for the chain. Using chain timing and configuration information carried by the CP, a receiving node schedules in advance the resources necessary for the chain and locks on to a stream of data frames (i.e. the chain) to switch the frames from their input port to an output port without performing any header-related operation on the single data frames. This saves a significant amount of processing to the higher network layers, effectively handling the transaction at the lower layers. This feature greatly relaxes the computational requirements on the network equipment supporting BFP, which can handle the same amount of traffic with a computational load that is orders of magnitude smaller than when using legacy IP protocols (Appendices A and B). Furthermore, as discussed in more detail in Section 4.1.2, the functionalities required to handle BFP transmission are "lean" enough that virtually any device able to perform store and forward of data frames, and with enough computing power to execute the BFP resource reservation protocol, can potentially support BFP.

Periodicity also allows chain interleaving when more than one flow competes for the same output channel. As shown in Figure 4.3, buffering (i.e. *Buffering Time* or BT) is used to align incoming chains in time. Once aligned, chains will naturally interleave at the output ports. This buffering technique also makes the amount of buffer to be used for BFP predictable and dependent on the configuration of chains more than on the traffic load. This way, the amount of buffer space to reserve for BFP can be decided beforehand, depending on the desired performance and expected traffic load.

A resource reservation protocol designed to handle periodic data structures is used to perform end-to-end reservation for each chain (Section 4.1.2). Once successfully reserved, the entire transaction will be delivered without incurring any additional delay, providing stable delay performance and predictable resource occupancy. This

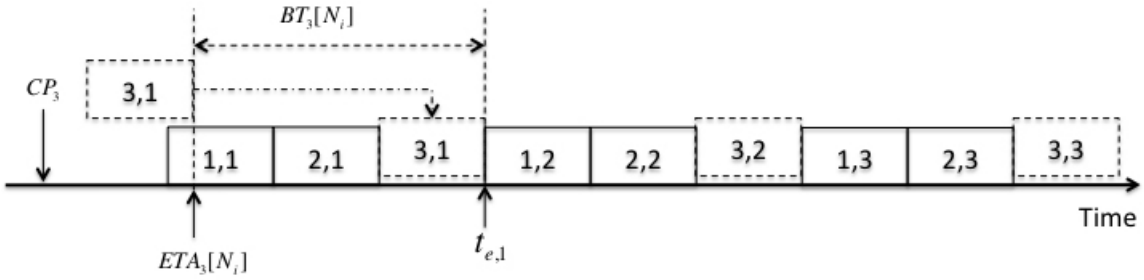


Figure 4.3: Chain interleaving at node N_i. Chains 1 and 2 are already scheduled, chain 3 (incoming) is buffered ($BT_3[N_i]$) and interleaved with the other chains

.

feature may be particularly desirable for applications where strict delay control is required (e.g. video streaming).

Using per-transaction resource reservation could become cumbersome if transactions are too small. For large files (e.g. $\geq 100MB$), however, the benefits of using reservation protocols are well known. The file size threshold for which using BFP becomes beneficial can vary from network to network and from application to application, and is left as a design parameter to be evaluated on a case-by-case basis.

The functionalities required by BFP are already present in the current OTN network hierarchy, so no new hardware deployment would be needed. All the necessary operations (i.e. chain signaling, delineation, and buffer control) are handled by the control plane, which is the only portion of the network that would need to be updated in order to support BFP (Appendix D). A software/firmware update of currently deployed hardware will suffice.

4.1.2 Resource reservation protocol

There are many ways to perform resource reservation for BFP chains. The reservation procedure can be source-initiated, destination-initiated, or orchestrated by a centralized server, depending on the network in which BFP is deployed. The general assumption is that an estimate for the Round-Trip-Time (RTT) between source and destination is available at the source node when the reservation procedure starts. If the network size and path diversity allow it (i.e. if it is not computationally too cumbersome), BFP may use a centralized route server to compute the path(s) and relative RTT between a given *source-destination* pair, possibly according to some routing/load balancing strategy. This information is then provided by the route server to the source

node. In larger networks where the use of a centralized route server may incur scalability issues, standard ranging protocols can be used to gather RTT information. Currently available OTN delay measurement functionalities [60, 63] can also be used. In a DC environment, where the topology is well-known and the network relatively small, this issue can be solved by using a rough estimate of the maximum possible RTT and adding an offset time before starting the chain transmission (Section 4.1.2). A similar approach can also be used in larger networks. Although this may lead to wasting an amount of bandwidth proportional to the difference between the estimated and effective RTT (RTT_{est} and RTT_{eff}). The high degree of statistical multiplexing and the large number of flows traveling in such networks, however, should keep the wastage small and limited to the links close to the source node[2].

In the following, a source-initiated reservation procedure is described. Extending this procedure to a destination-initiated approach is trivial and boils down to adding a delay equal to $RTT/2$ to the source-initiated procedure in order to account for the time needed by a transmission request (T_{Req}), coming from the destination node to the source node, plus the chain assembly time (δ_a) at the source node.

The (source-initiated) BFP resource reservation procedure (Figure 4.4) starts as soon as a transaction is segmented into BFP data frames at the source node. A CP, associated to the chain though an ID, is generated with the following information included:

- *Source/Destination identifier*: source and destination points for the chain. These could be L2, L3 addresses or any other identifier, depending on the particular BFP implementation and on the capabilities of the underlying hardware.
- *Frame size* (F_s): number of BPFs within each data frame.
- *Chain length* (L_{Ch}): number of data frames within a chain (Equation 4.2).
- *Transparency Degree* (TD): chain period.
- *Expected Time of Arrival* (ETA_i): this parameter communicates to a receiving node "i" the time at which the first bit of the chain will reach it, expressed as

[2]Note that as we move towards the core of the network more and more sources contribute to the overall traffic on a certain link or channel. The large number of contributing flows compensates for any bandwidth wastage resulting from the offset time at the source nodes. In other words, using an offset time before chain transmission a the source may waste access bandwidth close to the source node, but it is highly unlikely to have any influence when a large number of flows coming from different sources are multiplexed.

the amount of time between reception of the CP and arrival of the first bit of the chain at node i. Using this approach frees BFP from the need to rely on global network synchronization as time has only a local significance relative to each node.

- *chain ID*: a unique identifier associating each CP to a specific chain. This must be chosen to be unique only within the source node, as its value is compared with the *source identifier* field by each receiving node to unequivocally identify each chain anywhere in the network. This rids BFP of the need to use a cumbersomely long ID field or procedures to make sure no identical IDs are used network-wide.

The ETA_i parameter is computed by a node for the next node in the data path, i.e. node $i-1$ computes ETA_i (for node i). The first ETA computation occurs at the source node, according to Equation 4.3

$$ETA_{src} = \sum_{i=1}^{N}(\tau_i + p_i) + \tau_{ACK} \equiv RTT \tag{4.3}$$

where:

N number of nodes in the path

τ_i propagation time to reach node i [s]

p_i estimated CP processing time for node i [s]

τ_{ACK} time for the ACK to reach the source node [s]

Using the ETA parameter allows each node in the data path to perform delayed resource reservation for incoming chains, optimizing the utilization of resources by reserving them only for the time strictly necessary to handle a chain. As discussed in Appendix B, this gives an edge to BFP in contrast to approaches like OFS (Section 3.2) in terms of efficient utilization of the network bandwidth.

Once generated, the CP is sent from the source node over the data path to reserve bandwidth, buffering and switching resources for the associated chain. When a node "i" in the path (N_i) receives the CP, the destination identifier is passed onto the routing layer which responds with an outgoing port for the chain. Resource availability for the incoming chain on the outgoing channel is checked:

- **If resources are available**, the buffering time for the incoming chain ($BT_n[N_i]$, Figure 4.3) is computed according to Equation 4.4, the ETA parameter of the CP is updated with the $BT_n[N_i]$ (Equation 4.5) before forwarding it to the next node (i.e. node N_{i+1}). The information necessary to handle the incoming chain, namely: chain ID, $BT_n[N_i]$, ETA and input/output ports is stored on a local table (i.e. the *port-map table*, Table 4.1). This table stores entries relative to every active BFP flow in the node, and when a chain crosses the node the relative entry is deleted.

- **If resources are *not* available**, the CP is blocked and the source is notified with a $NACK$. Upon receiving a $NACK$ the source backs off for a certain amount of time before retrying the reservation procedure. The back-off time can be completely random or loosely based on topological information about the underlying network and traffic characteristics (Appendix E).

Once the CP reaches the destination node, an ACK (also containing the chain ID) is sent to the source node informing it that chain resources are available. Upon receiving the ACK, the source node can start transmitting the associated chain. A confirmation of transmission completion (END_ACK) can be sent from the destination node to the source node, confirming correct reception or indicating which frames (BPF) are to be retransmitted. A time diagram of the source-initiated BFP resource reservation procedure is shown in Figure 4.4

$$BT_i = t_{e,1} - ETA_i \tag{4.4}$$

$$ETA_{i+1} = ETA_i + BT_i - p_i \tag{4.5}$$

where:

$t_{e,1}$ is the ending time of the first available timeslot on the outgoing link (Figure 4.3).

The size of the port-map table at each node depends on the number of active BFP transactions at that node. Although the port-map table can potentially become cumbersome, we assume that the reservation procedure is fast enough and the information to be stored is *lean* enough to keep the size of the table reasonable[3].

[3]This assumption is reasonable for two main reasons: (1) we are only considering a portion of the traffic which is significant in terms of its contribution to the overall bits traveling through the

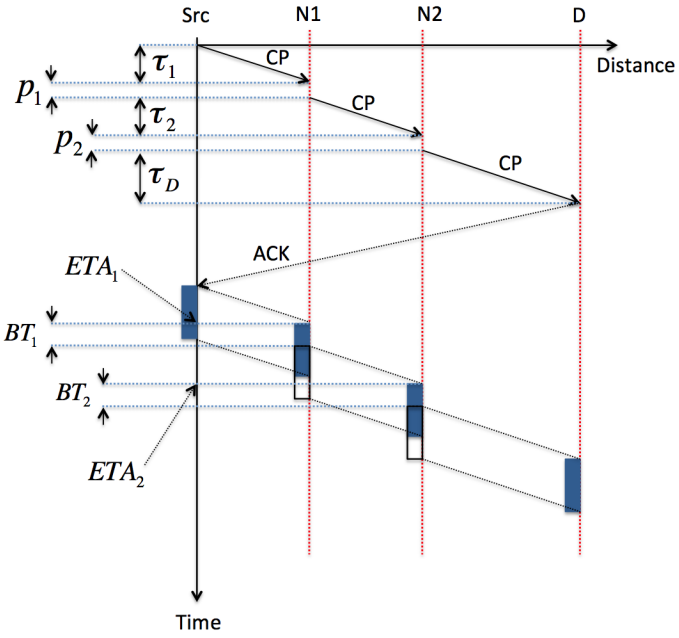


Figure 4.4: *Source-initiated* BFP resource reservation and transmission procedure.

Table 4.1: *port-map table (example entry).*

Chain ID	TD	ETA	Input Port	BT	Output Port
64	8	6.2 mS	7	0.043 mS	18

As mentioned above and discussed in detail in Appendix C, the binding between input and output ports stored in the port-map table can be established using any available routing/load balancing protocol as part of the CP processing, thereby avoiding access to higher layer functions for every single data frame (e.g. power-hungry, header processing functions [110]), and limiting their access to a single packet per transaction (i.e. the CP), regardless of the transaction size. Potential advantages of this approach in terms of energy savings are further discussed in Appendix A.

4.1.2.1 BFP resource reservation for small networks and DC

In this section we discuss a small variation to the BFP resource reservation procedure that can be used in smaller networks (e.g. DCN), where the maximum RTT achievable is small with respect to the duration of a transaction. When deployed in networks

network, but relatively small in terms of number of flows. In other words, the overall number of BFP flows is limited with respect to the total number of flows in the network, and (2) each flow is scheduled within one RTT, which means the maximum permanence of an entry in Table 4.1 is always $T_{MAX} < RTT + RTT/2$

with small RTTs, the resource reservation procedure described above can be modified by adding an offset time (Δ_R) between the reception of the confirmation for the resource reservation and the transmission start for the chain. This offset time is computed upon ACK reception from the difference between the estimated maximum RTT (RTT_{MAX}), and the effective RTT (RTT_{eff}) experienced by the CP during the resource reservation procedure (Equation 4.6). After waiting a time equal to Δ_R the source node can start transmitting the chain. A time diagram for the modified resource reservation procedure is shown in Figure 4.5.

$$\Delta_R = RTT_{MAX} - RTT_{eff} \tag{4.6}$$

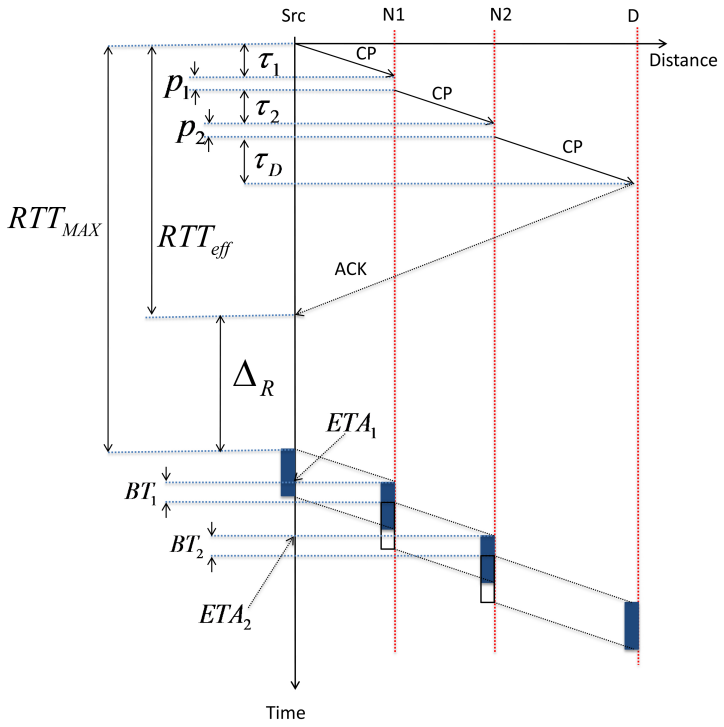


Figure 4.5: *Source-initiated* BFP resource reservation and transmission procedure for networks with small RTT

4.2 Software Implementation and Simulation Details

In this section we will describe the details of the BFP simulator implementated using *Omnet++* [111]. The software structure described here resembles that of the software that can be installed on real routing/switching devices supporting BFP. Whether BFP will be implemented as a software or firmaware upgrade to current devices or as combination of both or even as standalone device (Appendices A and B) depends on the the available hardware, its functionalities as well as other factors (such as the willingness of companies/operators to invest) that, at the moment, are out of our control. Implementation options and their relative details for BFP in *real-world* systems can be found in the appendices.

4.2.1 Channel Model

In order to perform in-advance resource reservation for BFP chains, the available bandwidth of each channel should be modeled in the control plane with an appropriate data structure. This data structure should reflect the chain configuration, allowing the minimizing of the computational load necessary to monitor channel resources (available bandwidth) and schedule of BFP chains.

In our implementation of BFP, the bandwidth of each channel is modeled as a set of timeslots organized in a matrix (referred to as the *channel matrix* or CM) of dimensions $FB \times N_{FB}$, where FB is the number of columns (or *Fundamantal Block*) and N_{FB} is the number of rows (i.e. the *Number of Fundamental Blocks* in the CM). The value of FB is chosen based on the maximum TD allowed on the channel (Equation 4.7).

$$FB \triangleq TD_{MAX} \tag{4.7}$$

where:

FB is the size of the fundamental block

TD_{MAX} is the maximum TD value allowed on the channel

The beginning time ($t_{(0,0)}$) of the first element (slot) of the channel matrix (element $(0,0)$) is initiated with the (local) reception time of the first CP ($t_{(0,0)} = t_{CP\_arrival}$)

and is updated (i.e. shifted in time) as new data frames are booked onto it. The duration of each timeslot is equal to the length ($\overline{f}_s$) -in seconds- of a BPF at the channel bitrate (i.e. $\overline{f}_s = BFP\_size[b]/channel\_bitrate[b/s]$). The beginning time of the other timeslots of the first row of the channel matrix is derived from slot $(0,0)$. All other slots of the CM (i.e. any (n,m) with $n \geq 1$) are shifted versions of the slots in the first row, e.g. beginning time of slot $(1,2)$ is given by $t_{1,2} = (t_{0,0} + 2) * \overline{f}_s * FB$. A general expression to derive the beginning time $t_{(n,m)}$ of a slot (n,m) in the CM is given in Equation 4.8. The *ending time* of each slot is simply its beginning time plus the slot duration i.e. $t_{(n,m)} + \overline{f}_s$. A graphical representation of the CM is shown in Figure 4.6

$$t_{(n,m)} = \Big[n * t_{(0,0)} + m\Big] * \overline{f}_s * FB \tag{4.8}$$

where:

n (≥ 1) is the row index

m is the column index

As time passes and incoming chains are booked, the last row of the CM is eventually reached. When this happens, chain (data frames) booking continues by *rolling over*, back to the beginning of the CM (row 0). In order to be sure that rolling over doesn't cause incoming chains to be booked in timeslots that are still occupied by previously booked chains, the number of rows of the CM (N_{FB}) is selected to be big *enough* such that, as the roll-over point (Figure 4.6) is reached -causing the reservation to roll back to the beginning of the CM-, the timeslots of incoming chains will not coincide with those occupied by previously booked chains. In other words the size of the CM is such that enough time would have passed before roll-over occurs, allowing timeslots at the beginning of the CM to be freed in the meanwhile.

4.2.2 Bandwidth occupancy selection and TD compatibility

The average bandwidth occupied by a chain can be modeled using Equation 4.1 (Section 4.1). Organizing transaction into chains comes at the price of a less flexible bandwidth selection than possible when using protocols like TCP (although it is important to note that TCP does not enable explicit bandwidth selection, but simply tries to get as much as possible of what's available). Methods to make the BFP

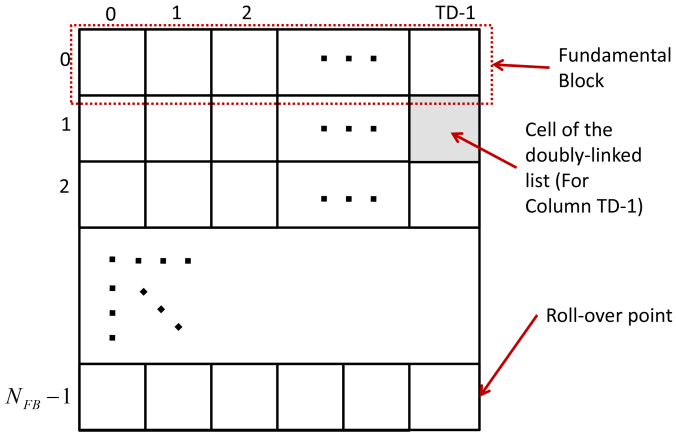


Figure 4.6: BFP Channel Matrix

bandwidth selection more flexible are needed. There are various methods to change the bandwidth occupied by BFP chains: at the source node, or while the chain is *in-flight*. It's best, however, not to change the bandwidth occupation of a chain as it is traveling (unless we are willing to pay a price in terms of buffer space and delay), and limit as much as possible these changes to selecting the appropriate TD at the source node.

The simplest way to select bandwidth when using BFP chains is to change their TD. Smaller TDs correspond to chains occupying a larger portion of the available bandwidth (Equation 4.1), while larger TDs correspond to chains with lower bandwidth occupancy. Another approach is to split a transaction into multiple chains, i.e. using a *multi-chain* which can be signaled on the same path using a single CP carrying multiple $< ETA, TD >$ pairs, one per chain (*non-homogeneous, single-path multi-chain*). Alternatively, multiple chains can be signaled over multiple paths (if multipath routing is available) using multiple CPs. In the case of single-path multi-chains, the resulting bandwidth occupation for the entire transaction would be equal to the sum of the bandwidth occupied by the single chains (Equation 4.9). When using multi-chains, the frames may have to be resequenced at the destination node, especially if each chain travels over a separate path. Another alternative is to transmit multi-chains as a set of back to back chains all with the same TD (*homogeneous, single-path multi-chain*). The effect is equivalent to increasing simultaneously the data frame size while reducing the TD of the resulting multi-chain. This type of multi-chain can be signaled using a single CP, which describes the multi-chain struc-

ture appropriately by selecting the *frame size* and TD fields. Homogeneous and non-homogeneous multi-chains are shown in Figure 4.7 and 4.8, respectively.

$$\overline{B}_{Total} = \sum_{i=1}^{n} \frac{f_{b_i} * L_{Ch_i}}{f_{s_i} * [TD_i * (L_{Ch_i} - 1) + 1]} \tag{4.9}$$

where:

n is the number of chains in the multi-chain

f_{b_i} size (in bits) of a data frame for chain i

L_{Ch_i} chain length for chain i

f_{s_i} duration (in seconds) of a data frame for chain i

TD_i Transparency Degree for chain i

Any combination of the three approaches presented above is also possible, the only constraint is that all the TDs used in a BFP-enabled (sub)network must be compatible. The condition for chain compatibility is expressed by Equation 4.10

$$TD \triangleq k^n, n = 1 \div \overline{n} \tag{4.10}$$

where:

$k, n \in \mathbb{I}$, AND

$TD_{MAX} = k^{\overline{n}}$ is the maximum TD allowed in the BFP (sub)network.

For the sake of completeness, note that when changing the bandwidth occupancy of a chain *in-flight*, a portion of the chain (or even the whole chain, depending on the situation) may have to be buffered. The existing chain reservation from the node that is changing the chain bandwidth until the destination node, would have to be adjusted to account for the new chain (or multi-chain) configuration at the expense of increased control plane traffic, computational load for the network equipment, and delay resulting from any eventual path changes. Furthermore, for when a chain is passing, through a gateway, from a network to another requiring a configuration change, this task can be delegated to the gateway itself, which will change the chain configuration data in the CP during the reservation phase and buffer the incoming chain for the time necessary.

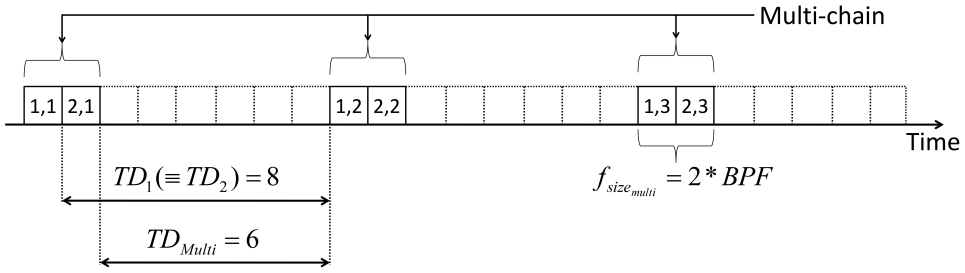


Figure 4.7: Homogeneous multi-chain ($TD_{1,2} = 8$, $f_{size_{1,2}} = 1 * BPF \rightarrow TD_{multi} = 6$, $f_{size_{multi}} = 2 * BPF$)

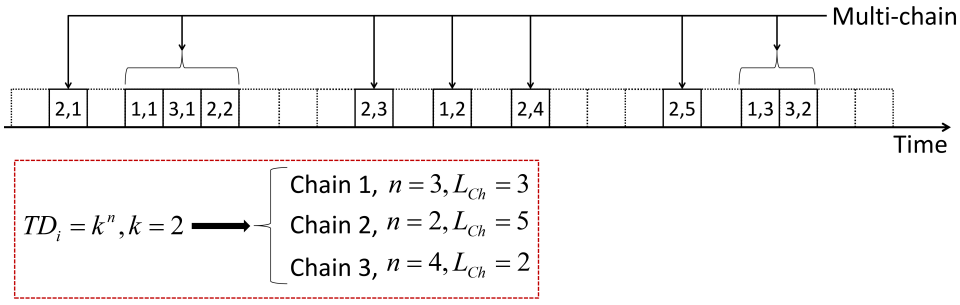


Figure 4.8: Non-homogeneous, single-path multi-chain ($f_{size} = 1 * BPF$ for all chains)

4.2.3 Chain reservation procedure at a local node

In what follows we will describe the booking procedure executed at each node upon reception of a CP. The access to the routing layer to determine the output port/channel for the incoming chain is assumed to be successfully completed at this stage. Furthermore, each column of the CM is coded as a doubly-linked list. Each cell of this linked list is defined by the code shown below (**Step 3**). The chain booking procedure is comprised of the following steps:

Step 1

As a CP reaches a node, the following information is extracted:

- *frame size* (F_{BPF}) $\rightarrow$ number of BPF in each data frame
- $L_{Ch} \rightarrow$ number of data frames per chain
- $TD \rightarrow$ Transparency Degree
- $ETA \rightarrow$ Expected Time of Arrival
- *Chain ID* $\rightarrow$ relative to the incoming chain

Step 2

The number of CM columns needed to book by the incoming chain are computed according to Equation 4.11

$$N_{slots} = FB/TD_i \tag{4.11}$$

Note that this quantity is guaranteed to be an integer if condition expressed by Equation 4.10 is respected.

Step 3

The number of CM rows occupied by the chain data frames in each of the N_{slots} slots computed at **Step 2**, is computed using Equation 4.12. The integer part of this equation ($\overline{\delta_{Res}}$) gives the number of slots needed for each of the N_{slots} slots, while the reminder (r_i) is distributed starting from the first slot until exhaustion of the reminder (*bin sorting* problem).

$$\overline{\Delta R} = \frac{L_{Ch}}{N_{slots}} \rightarrow \overline{\delta_{Res}} + r_i \tag{4.12}$$

At this point the number of CM columns and their *vertical extension* in the CM needed to book the incoming chain is known. This information is saved in a set of N_{slots} cells of a doubly linked lists, each coded as follows:

```
#ifndef CELL_H_
#define CELL_H_

#include <stdlib.h>

class Cell{
    public:
        int slot, start, end;
        Cell *next, *prev;
        Cell(int,int,int);
        virtual ~Cell();
};

#endif /* CELL_H_ */
```

where:

slot $\rightarrow$ is the columns index of the CM corresponding to one of the N_{slots} slots used to book the incoming chain

start $\rightarrow$ is the row index of the CM corresponding to the beginning of the (vertical) reservation interval for one slots occupied by the chain

end $\rightarrow$ is the row index of the CM corresponding to the ending of the (vertical) reservation interval for one slots occupied by the chain

next, previous $\rightarrow$ are the pointers to the *next* and *previous* cells of the doubly linked list used to code the CM columns, respectively

Step 4

The ETA parameter is used to search the CM slots for the first available slot to book the N_{slots} slots of the chain. The search starts from the slot corresponding to the ETA of the incoming chain whose index (n_{ETA}) is given -in $\mathcal{O}(1)$- by Equation 4.13.

$$n_{ETA} = \left\lceil (t_{(0,0)} + ETA) \right\rceil \bmod (FB * N_{FB}) \tag{4.13}$$

Step 5

Once a suitable slot is found, the CM is scanned horizontally ($mod\ TD$), starting from this slot, to check if the $(N_{slots} - 1)$ subsequent slots needed to book the chain are also suitable. If all slots are suitable, that is, none of the (new) N_{slots} cells collide with other cells on the CM columns corresponding to previously booked chains, the Buffering Time (BT) is computed according to Equation 4.4, and the N_{slots} cells are added orderly to each of the suitable CM columns. The incoming chain is now booked. The ETA field of the CP is updated with the BT (Equation 4.5) and the CP is forwarded to the next node. If not all the slots scanned are suitable, that is, one of the columns turns out to be already booked, the search continues column by column until a suitable set of columns is found or a pre-set limit is reached (determined by the maximum buffering time allowed in the system). If no suitable set of columns is found, the CP is blocked, turned into a $NACK$ (by adding, e.g., a *CP_type* field to the CP), source and destination addresses are swapped, and the $NACK$ is sent over the reverse path to free previously booked resources node-by-node.

Roll-over: If the rollover point is reached somewhere in the middle of a cell interval, the cell is split into two cells: one from the beginning of the scheduling

interval to the roll-over point (row $N_{FB} - 1$), and another from the beginning of the CM (row 0) to the ending point of the scheduling interval.

Algorithmic complexity: All above steps are executed in $\mathcal{O}(1)$, except for **Step 3**, which is executed in $\mathcal{O}(n)$, and **Step 5**, also executed in $\mathcal{O}(n)$. The effective computational load of **Step 3** is determined by the value of N_{slot}, while that of **Step 5** is determined by the maximum amount of buffering allowed at each node.

4.2.4 Bitrate upshift/downshift

As a chain passes from a channel at bitrate B_1 to another with bitrate B_2, its configuration naturally changes as its TD and frame duration change. The following two cases may occur:

Case 1: $B_1 < B_2$ **(*upshift*):** The frame *duration* (f_s) shrinks while the incoming TD (TD_{B_1}) expands.

Case 1: $B_1 > B_2$ **(*downshift*):** The frame *duration* (f_s) expands while the incoming TD (TD_{B_1}) shrinks

The resulting TD_{B_2} for both cases is given by Equation 4.14.

$$TD_{B_2} = TD_{B_1} * \frac{B_2}{B_1} \tag{4.14}$$

where:

$B_1 < B_2$ for *upshift*

$B_1 > B_2$ for *downshift*

Note also that the bitrates used in current networks are all (nominally) multiples of each other, e.g., an ODU2 signal rate is 4×ODU1 signal rate. Likewise a 10GbE Ethernet signal is nominally 10 times the rate of a 1GbE signal. Similar rate relationships exist between all other OTN or Ethernet signal rates. This avoids having TD become some fractional value when upshifting or downshifting. Small deviations will obviously be present, but they will be accommodated for during the mapping procedure of a chain on to a channel with a different bitrate, and should not influence the functioning of BFP in any way.

4.3 Packet Processing: BFP vs IP

One of the main advantages of BFP is the reduction of the computational load placed on the network hardware by large transactions. Aside from the advantages in terms of power consumption (See Section 2.2.2), reducing the computational load placed on routers, will also enable using cheaper, less powerful hardware to perform the same tasks. Furthermore, reducing the computational load of currently deployed hardware may result in *spare* computing power that can be used to support unforeseen applications that may come in the future, thereby extending the life cycle of the current network hardware.

In order to have a better idea of the potential savings achievable using BFP, consider the following. IP packet processing functions, such as IP header and MAC table lookups (often implemented with power-hungry TCAM components), header parsing, and packet classification are executed on a per-packet basis. Similar functions are likely to be executed for BFP, but in the BFP case these functions are only executed once per transaction, as the relative CP is processed. A graph showing the difference in number of packets processed per offered load between standard UDP and BFP can be found in Appendix A.

BFP data frames are handled at the lower network layers of the network and, and do not require access to higher layer functionalities such as routing, header parsing or header lookup. BFP data frames are detected and delineated using functionalities available at the lower network layers (i.e. L1/L2). A single access to the port-map table provides the information necessary to handle an entire data flow (e.g. Chain ID, ETA, input port, output port, and buffering time). Once detected, each frame is just buffered and switched onto the appropriate output port, and requires no further processing. In other words BFP avoid all the computationally-intensive functions that are executed on a per-packet basis by standard IP protocols, accessing such functions only once per transaction.

Queue management functions similar to those executed on a per packet basis for IP packets are not necessary for BFP as QoS support can be implemented on a per-flow or per-source basis deciding, e.g., which CPs are serviced and which TD (i.e. how much bandwidth) is assigned to each flow. The only buffer-related functions needed by BFP boil down to computing the amount of buffer space needed by BFP based on the maximum TD allowed, and reserving a portion of the available buffer space on a per-flow basis.

As is the case for IP packets, also BFP data frames have to be written and read to and from a memory. However, the possibility for BFP to use much larger frames with respect to legacy IP protocols such as TCP, allows writing BFP data frames in large, contiguous memory blocks, significantly reducing the number of memory accesses per transaction, and consequently the number of IRQs that the CPU has to handle. This results in further power savings for data storage systems when using BFP.

4.4 OTN and Ethernet Extensions for BFP Support

Various options exist to integrate BFP within currently deployed Ethernet or OTN transport architectures. Details on how to integrate BFP onto Ethernet and OTN are provided in the appendices (C and D). Here we will limit our discussion on a general overview of the extensions necessary to both OTN and Ethernet transport systems to support BFP.

Mapping, framing, and switching functionalities in both OTN and Ethernet are virtually unchanged as BFP is designed to seamlessly integrate with whatever transport technology is available. The ability to establish dedicated bandwidth pipes for BFP traffic (i.e. a BFP overlay network), are already implemented in current transport networks at the OTN layer.

Both OTN and Ethernet need to interact with the port-map table to gather information on the data frame size (i.e. number of BPFs per data frame) and on the output port where the BFP data frames are to be forwarded.

The BFP control plane needs to interact with the OTN or Ethernet layer and provide buffer management functions for BFP data frames. The control plane needs to handle the buffering of data frames according to the value of the buffering time stored in the port-map table. A shim layer interfacing the BFP control plane with functionalities available at the transport layer (e.g. mapping, framing, switching, etc.) can be implemented as a software or firmware upgrade of currently deployed OTN or Ethernet hardware. The form of such shim layer is reminiscent of SDN approaches where some lower layer functionalities are abstracted and made available to higher layers (the BFP control layer in this case). The implementation details of the shim layer depend on the hardware at hand, but the functionalities needed to support BFP

boil down to the ability to intercept BFP control packets and data frames and control the buffering of BFP data frames (See Appendix D).

Chapter 5

Periodic Data Structures For Bandwidth-Intensive Applications

Issues relative to current network protocols and architectures in both IP and data center networks, as well as some of the proposed approaches to solve such issues (e.g. power consumption in IP networks and performance stability and predictability in DCNs), were discussed in previous chapters. In this chapter we will address the issues of current and proposed networking solutions for both IP networks and DCNs, and present the results obtained by the proposed approach, comparing them with current and other proposed approaches.

5.1 Media Frames Networking: an Electronic Burst Switching approach for energy-efficient transport of large files (Appendix A)

As discussed in Section 2.2.2, a significant amount of power in current IP networks is used by header-related processing functions. In current IP network architecture, these functions have to be executed for each packet. The use of small IP packets (e.g. $1500B$), combined with the significant increase in the average file size traveling through the network, generates a large number of small packets that are to be processed individually, pushing the power consumption of the current IP infrastructure to levels that may soon become unmanageable, ultimately limiting the expansion of the network itself. In this paper, we explore the potential advantages of using large data

frames (e.g. $\geq 1Mb$), buffered and processed electronically, to handle large transactions. We refer to these frames as *Media Frames* or MF. MFs are handled as *oversize* UDP packets, and no reservation procedure is employed, avoiding control plane congestion; we refer to this approach as MF-UDP. The aim of this approach is to reduce header-related power consumption. Simple hardware considerations suggest that using large data frames reduces header-related power consumption. Furthermore, some of the requirements placed on standard IP switch fabrics, such as the need for highly dynamic switch fabrics, are relaxed. The drawback of the MF-UDP approach is that a significant amount of buffer space is needed in order to handle the large MFs and to keep frame dropping rates acceptable. To reduce the amount of buffer space needed by MF-UDP, MFs were concatenated into a periodic set of data frames (referred to as *Media Frame Chain* or MFC), and a resource reservation protocol was used to schedule transmission of MFCs. Our simulation study showed that using periodic concatenations of data frames leads to a significant reduction in the amount of buffer space needed with respect to both standard UDP and MF-UDP. The reduction measured in occupied buffer size was between 2 to 5 times that of both other approaches. Better performance of concatenated MF-UDP frames was also achieved in terms of link Utilization and delay per transaction in the high traffic load range ($\sim 9\%$ higher than both UDP and MF-UDP for traffic loads above 74%). The potential advantages of the use of MFCs in terms of power consumption were evaluated, and the schematic of a router able to support MFCs was studied, showing the potential of an $\sim 80\%$ reduction in power consumption with respect to a commercial IP router.

5.2 Advantages of Concatenated Electronic Burst switching over OFS/OBS transport systems (Appendix B)

Many interesting proposals are being researched to solve the energy efficiency and scalability issues of current IP networks. Among the many proposals for next generation networks (NGN), two seem particularly promising: Optical Burst switching (OBS) and Optical Flow Switching (OFS). Both these approaches, and in particular OBS, attracted a significant interest and was the focus of many research efforts. Details of OBS and OFS are discussed in Chapter 3. In spite of their potential advantages in terms of energy efficiency, neither OBS nor OFS were commercially deployed

in large scale networks. This is mostly due to the significant changes to the network architecture required by such approaches as well as to their use (in OBS in particular) of all-optical components, such as optical buffers and all-optical switches, whose commercial viability and advantages in terms of energy efficiency and scalability are still subject to debate. In this paper we attempt to design a networking approach that retains the advantages of OFS and OBS systems (e.g. energy efficiency, low delay, and high link utilization), while avoiding the shortcomings such as extensive use of optical buffers or inefficient burst assembly techniques at the edge of the network (OBS), or the need for end users to install costly long-haul transmission equipment at their premises (OFS). We refer to our approach as Media Frame Networking (MFN) and compare it with TaG-OBS by means of simulation, showing that MFN is able to provide higher link utilization, lower delays, smaller call blocking probability, and functions with smaller buffer sizes with respect to TaG-OBS, while avoiding the use of optical buffers and keeping control plane traffic limited. Using simple mathematical passages, we also show that MFN is inherently better than OFS in terms of bandwidth efficiency. Advantages of MFN in terms of power consumption are also discussed.

5.3 Big File Protocol (BFP) for efficient handling of bandwidth-intensive transactions over current transport architectures (Appendix C)

The current networking scenario is undergoing some significant changes. On one end, bandwidth demand is growing rapidly, while on the other end, the cost per bit transported is decreasing almost at the same pace. This situation will soon lead to a point in which the cost of operating a network will surpass the revenue. In order to cope with this situation, network operators must reduce the cost per bit transported while at the same time increasing the capacity of their networks to meet the increasing bandwidth demand in an economically viable manner. Increasing network capacity is costly, while researching ways to make better use of the deployed capacity is an effort worth taking. Realizing that it is highly unlikely that network operators will be willing to undertake any major hardware redeployment, we designed a method, based on traffic shaping, which is able to better utilize the deployed capacity while at the same time avoiding any major hardware redeployment. The proposed approach, referred to as *Big File Protocol* or BFP, leverages functionalities made available by

the recently standardized ITU-T G.709 (Optical Transport Network), and enables handling bandwidth-intensive transactions at lower network layers (e.g. L1) where the cost per bit is lower. Higher layer functionalities are accessed only when strictly necessary (e.g. to gather routing information or to process control traffic), while the single data frames are handled with minimal or no processing. This approach is able to reduce significantly the computational load placed on current network hardware, while achieving goodput values close to 100%. A simulation study comparing BFP to TCP showed that BFP can offer significant advantages, namely: (1) BFP can handle over 40% more traffic with respect to TCP without incurring congestion or throughput collapse (e.g. TCP Incast); (2) BFP can achieve delay per transaction over 30 times smaller in the same load conditions and topology, with delay variation (especially important in data center networks) much smaller with respect to TCP in all load conditions. (3) Although for light network loads BFP occupies slightly more buffer space with respect to TCP, as the traffic load increases, the buffer space occupied by BFP remains limited within well defined values, while for TCP it jumps up to values several times larger than in those achieved by BFP worst case scenario. Lastly, (4) BFP showed a relative insensitivity to the statistical properties of the traffic. A methodology to integrate BFP within the current architecture and machine OSs is also discussed in some detail, showing that BFP can be deployed potentially without requiring any hardware redeployment.

5.4 Integration of BFP over OTN and Ethernet transport technologies: an easy to integrate approach for bandwidth-intensive transactions (Appendix D)

Extending the work presented in Appendix C, this paper presents a detailed discussion of how the proposed Big File Protocol can be integrated within the underlying OTN and Ethernet layers. With specific focus on OTN integration, details about framing and mapping procedures of BFP onto OTN are provided and discussed. Integration of BFP onto OTN-enabled hardware (e.g. Packet-Optical Transport Platforms) is also discussed, and a possible integration strategy of BFP within currently deployed network architectures that do not require any costly hardware redeployment is ex-

plored. This approach minimizes the deployment impact and offers a more gradual evolutionary path for both IP and data center networks.

5.5 Data Center BFP: a quasi-deterministic data transfer protocol for stable, predictable network performance in Data Center Networks (Appendix E)

Data Centers (DCs) are rapidly becoming an important player in the support and development of Internet-based applications. Although responsible only for a small portion of the overall cost of a DC, the data center network (DCN) plays a key role in determining the overall performance. DCNs are different from Wide-Area Networks in many respects, and protocols that may work in one environment may not work as well in the other. Also issues faced by DCNs are different from those faced in WANs. This is due to the different functionality and performance requirements of the two systems. Key issues in DCNs are performance stability and predictability (these issue are also present in WAN networks but are somehow less pressing), isolation of the tenants performances (in that the presence of a *misbehaving* tenant would not influence the other tenant's performance), and the possibility to (re)assign any service to any machine in a DC (*Agility*), which is also an important requirement in DCNs. All these factors heavily influence the DC user experience and are important issues in DCNs. A sizable volume of research is trying to address such issues. Current DCN architectures, however, still struggle to achieve acceptable performance for all these parameters simultaneously and in a topology-independent manner. In this paper we modify BFP to work in a DC environment (DC-BFP) and show that it is able to provide a wide range of DCN architectures (both proposed and deployed) with the means to manage the available bandwidth more efficiently while providing stable and predictable performance and uniform high-bandwidth between servers, which improves agility and performance isolation in a topology independent manner. DC-BFP does not aim at redesigning the DCN architecture or at making disruptive changes, but is designed to minimize the deployment impact, and is able to integrate with many routing, load balancing and DCN management approaches, substituting traditional IP protocols where they fall short (e.g. bandwidth-intensive applications). In this

paper, we run a BFP simulation in a data center environment over various topologies and compare the results with many popular architectures, showing significant performance advantages of BFP in all cases. An integration strategy of DC-BFP is also discussed. In this case we do require some hardware modification together with a software/firmware upgrade. This, however, is more acceptable in a DCN which is likely to be under the same administrative control and much smaller in size than a WAN. The hardware required to integrate DC-BFP onto current DCNs is commercially available today.

Chapter 6

Conclusions

In this dissertation we considered the application of periodic data structures as a means to handle efficiently bandwidth-intensive transactions. Our first step was to design a standalone system that could work in parallel with the currently deployed communication infrastructure, providing data transport services for large transactions, offloading the current network hardware from transactions it was not designed to handle, resulting potentially in significant energy savings for the network hardware. We compared our approach to both OBS and OFS, showing we can provide similar advantages in terms of energy efficiency, resource utilization, and network performances using commercially available devices.

We then adapted our approach (BFP) to work within the currently deployed communication infrastructure, reusing most of the hardware and protocols deployed, thereby minimizing the deployment impact (a significant hindrance to the adoption of NGN technologies). We provided an integration strategy of our approach within the recently standardized ITU-T G.709 OTN, enabling BFP to use currently available OTN functionalities to handle large transactions at the lower layers of the network, skipping most of the processing required by legacy protocols such as TCP/IP, while still achieving goodput values close to 100% of the available capacity, superior delay performances, and efficient and predictable utilization of network resources. The proposed integration strategy showed that BFP can integrate with currently deployed network architectures using a software/firmware update, significantly limiting the deployment costs associated with other approaches (e.g. OBS, OFS, etc.).

In the last phase of this research we considered applying BFP to DCN to solve some current DCN issues (e.g. performance stability and predictability, performance isolation, agility, efficient usage of the available resources). When adapted to work in a

DC environment, BFP showed significant advantages over popular DCN approaches. BFP was able to achieve quasi-deterministic network performances, perfect fairness, goodput values close to 100% of the available bandwidth, and much smaller delays with respect to other approaches using legacy IP protocols (e.g. TCP). BFP also showed insensitivity to the particular DCN topology, offering all the aforementioned advantages in a topology-independent manner, and easing its integration within many proposed and implemented DCN architectures.

6.1 Future Work

In this work we have demonstrated the potential of BFP to efficiently handle bandwidth-intensive applications. However, BFP is still in its infancy and there are some important steps that need to be taken before a fully functional system can be deployed.

The first step towards a real-world deployment is the implementation of a hardware testbed. Two approaches are available, namely: (1) using existing hardware [112] to implement BFP as a firmware upgrade and integrate it with existing IPs (Intellectual Properties) implementing Ethernet or OTN, or (2) by implementing BFP as a transport protocol for SDN-enabled hardware, using abstractions and functionalities made available by software-defined networking approaches [113]. This second option will give us less flexibility and would most likely require some compromise in terms of performance, but it has the potential to significantly reduce the implementation costs (software-only vs hardware/firmware), and will provide an implementation which is entirely compatible with SDN systems, making BFP attractive for the many DC operators and network providers currently deploying SDN-capable systems.

The size of this testbed can vary from two directly connected machines acting as BFP nodes, in turn connected to a number of BFP sources/destinations (a similar setup to the bottleneck topology used in Appendix C), to a DC-like topology involving several machine interconnected according to various DCN topologies.

Having a hardware testbed will allow us to benchmark the power savings achievable by BFP as well as the computational load effectively needed to handle BFP transmission. Fault tolerance and convergence time of BFP in case of failures can also be determined using a hardware testbed, helping to derive other important performance benchmarks.

Another interesting aspect of BFP in need of further attention is the development of routing and load balancing protocols specifically targeted to handle BFP traffic.

Such approaches are likely to be simpler than those already deployed for non-BFP traffic as routing and load balancing decisions for BFP traffic can rely on the quasi-deterministic behavior of BFP.

The ability to adapt the TD of chains to the available bandwidth and the possibility to change the configuration of a chain while *in flight* are desirable properties for BFP, extending the current implementation with such capabilities would greatly improve BFP flexibility. Studying the potential impact on buffer space, network goodput and delay performance is an important part of our future work.

Lastly, a BFP network management system should be developed and integrated with current network management systems. This will provide BFP with fault recovery procedures, QoS support, and the means to enforce traffic policies in both data center and transport scenarios.

List of Acronyms

AMP – Asynchronous Mapping Procedure
APS – Automatic Protection Switching
AS – Access Station
ASON – Automatically Switched Optical Network
BFP – Big File Protocol
BGP – Border Gateway Protocol
BHC – Burst Header Cell
BMP – Bit-synchronous Mapping Procedure
BPF – Basic Payload Frame
BNG – Broadband Network Gateway
BT – Buffering Time
BVF – Basic Void Frame
CBR – Constant Bit-Rate
CESR – Carrier Ethernet Switching Router
COTS – Commodity-Off-The-Shelf
CP – Control Packet
DAS – Directly Attached Storage
DC – Data Center
DC-BFP – Data Center BFP
DCN – Data Center Network
DPI – Deep Packet Inspection
EPS – Electronic Packet Switching
ETA – Expected Time of Arrival
EXP – Experimental
FAS – Frame Alignment Signal
FB – Fundamental Block
FDL – Fiber Delay Line
FF – First Fit
FEC – Forward Error Correction
GCC – Generic Communication Channel
GFP – Generic Framing Procedure
GMP – Generic Mapping Procedure
GMPLS – Generalized Multi-Protocol Label Switching

HI – High Order
IETF – Internet Engineering Task Force
IP – Internet Protocol
ISO – International Standard Organization
ISPF – Incremental Shortest Path First
ITU-T – International Telecommunication Union - Telecom. Standardization Sec.
JET – Just Enough Time
JIT – Just In Time
LAUC – Latest Available Unused Channel
LAUC-VF – Latest Available Unused Channel with Void Filling
LLC – Logical Link Control
LO – Low Order
MAC – Media Access Control
MAN – Metropolitan Area Network
MEMS – Micro-Electro-Mechanical Systems
MFAS – Multi-Frame Alignment Signal
MPLS – Multi-Protocol Label Switching
μ-OTP – Micro-Optical Transport Platform
MUX – Multiplexer
NAS – Network Attached Storage
NC&M – Network Control and Management
NE – Network Element
NNI – Network-to-Network Interface
NP-MOC – Non-Preemptive Minimum Overlap Channel
NUI – Network-to-User Interface
OADM – Optical Add/Drop Multiplexer
OAM – Operations, Administration, Maintenance
OAM&P – Operations, Administration, Maintenance, and Provisioning.
OBS – Optical Burst Switching
OCDM – Optical Code Division Multiplexing
OCh – Optical Channel
ODU – Optical Data Unit
OFS – Optical Flow Switching
OH – Overhead
OMS – Optical Multiplexing Section

OPEX – OPerational EXpenses
OPU – Optical Payload UnitAS – Frame Alignment Signal
OSI – Open System Interconnection
OTDM – Optical Time Division Multiplexing
OTN – Optical Transport Network
OTS – Optical Transport Section
OTU – Optical Transport Unit
OXC – Optical Cross Connect
P2P – Peer-to-Peer
PCC – Protection Communication Channel
PM – Path Monitoring
P-OTP – Packet-Optical Transport Platform
PSI – Payload Structure Identifier
PT – Payload Type
ROADM – Reconfigurable Optical Add/Drop Multiplexer
RTT – Round Trip Time
RWA – Routing and Wavelength Assignment
SDH – Synchronous Digital Hierarchy
SM – Section Monitoring
SONET – Synchronous Optical NETwork
TCAM – Ternary Content-Addressable Memory
TCM – Tandem Connection Monitoring
TCP – Transmission Control Protocol
TD – Transparency Degree
TDM – Time Division Multiplexing
UDP – User Datagram Protocol
UNI – User-to-Network Interface
VC – Virtual Concatenation
WAN – Wide Area Network
WDM – Wavelength Division Multiplexing
WRS – Wavelength Routing Switch

Bibliography

[1] Cisco Visual Networking Index: Forecast and Methodology, 20122017. CISCO white paper. Retrieved on October 20 2013 from: `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/\\ns705/ns827/whitepaperc11481360.pdf`

[2] Thomas Engel, Dr. Achim Autenrieth , Dr. Stefan Voll. *Quantitative Analysis of Network Architectures for Future National Optic Transport Networks (OTN).* Photonische Netze 03. 04.05.2010 in Leipzig

[3] Internet transit prices, available at `http://drpeering.net/white-papers/Internet-Transit-Pricing-Historical-And-Projected.php`

[4] P.Ferguson and G.Huston, *Quality of Service on the Internet Fact Fiction, or Compromise?*, INET'98,Geneva,Switzerland,1998.

[5] `http://www.telegeography.com`

[6] B. Swanson and G. Gilder, *Estimating the Exaflood: The Impact of Video and Rich Media on the Internet - a Zettabyte by 2015?,* Discovery Institute, Jan. 29, 2008.

[7] J. Baliga, K. Hinton, and R. S. Tucker, *Energy consumption of the Internet*, COIN-ACOFT, Melbourne, Australia, 2007.

[8] J. Baliga, R. Ayre, K. Hinton, and R. S. Tucker, *Photonic switching and the energy bottleneck,* in Proc. Int. Conf. Photonics in Switching 2007, San Francisco, CA, 2007.

[9] R.S. Tucker, R. Parthiban, J. Baliga, K. Hinton, R. W. A. Ayre, and W. V. Sorin, *Evolution of WDM Optical IP Networks: A Cost and Energy Perspective*,

JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 27, NO. 3, FEBRUARY 1, 2009

[10] K. Hinton, J. Baliga, Feng M.Z., R.W.A. Ayre, R.S. Tucker, *Power Consumption and Energy Efficiency in the Internet*, IEEE Network, Vol. 25, March-April 2011, Issue 2, pp. 6-12.

[11] G. Shen, R. S. Tucker, *Energy-Minimized Design for IP Over WDM Networks,* J. OPT. COMMUN. NETW./VOL. 1, NO. 1/JUNE 2009

[12] Koomey, J; Chong, H; Loh, W; Nordman, B; Blazek, M. 2004. *Network electricity use associated with wireless personal digital assistants,* American Society of Civil Engineers Journal of Infrastructure Systems (10): 131137.

[13] J. Baliga et al., *Carbon Footprint of the Internet,* Telecommun. J. Australia, vol. 59, no. 1, 2009, pp. 05.105.14.

[14] O. Tamm, C. Hermsmeyer, and A. Rush, *Eco-Sustainability System and Network Architectures for Future Transport Networks,* Bell Labs. Tech. J., vol. 14, no. 4, 2010, pp. 31128.

[15] T. Asami and S. Namiki, *Energy consumption targets for network systems,* presented at ECOC 2008, Brussels, Belgium, paper Tu.4.A.3.

[16] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2008. The cost of a cloud: research problems in data center networks. SIGCOMM Comput. Commun. Rev. 39, 1 (December 2008), 68-73.

[17] Yan Zhang; Ansari, N., "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," Communications Surveys & Tutorials, IEEE , vol.15, no.1, pp.39,64, First Quarter 2013

[18] Ali Hammadi and Lotfi Mhamdi. 2014. Review: A survey on architectures and energy efficiency in Data Center Networks. Comput. Commun. 40 (March 2014), 1-21

[19] Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Brian Mueller. 2009. Safe and effective fine-grained TCP retransmissions for datacenter communication. In

Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 303-314.

[20] Das, T.; Sivalingam, K.M., "TCP improvements for data center networks," Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on , vol., no., pp.1,10, 7-10 Jan. 2013

[21] Kajita, K.; Osada, S.; Fukushima, Y.; Yokohira, T., "Improvement of a TCP Incast avoidance method for data center networks," ICT Convergence (ICTC), 2013 International Conference on , vol., no., pp.459,464, 14-16 Oct. 2013

[22] Zhengwei Zhao; Zhixiong Jiang; Chunyang Lu; Yushan Cai; Jingping Bi, "A Congestion Control Algorithm for Datacenters," Networking, Architecture and Storage (NAS), 2013 IEEE Eighth International Conference on , vol., no., pp.98,104, 17-19 July 2013

[23] Coudron, M.; Secci, S.; Pujolle, G.; Raad, P.; Gallard, P., "Cross-layer cooperation to boost multipath TCP performance in cloud networks," Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on , vol., no., pp.58,66, 11-13 Nov. 2013

[24] Sahan Gamage, Ardalan Kangarlou, Ramana Rao Kompella, and Dongyan Xu. 2011. Opportunistic flooding to improve TCP transmit performance in virtualized clouds. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11). ACM, New York, NY, USA, , Article 24 , 14 pages

[25] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. 2009. PortLand: a scalable fault-tolerant layer 2 data center network fabric. SIGCOMM Comput. Commun. Rev. 39, 4 (August 2009), 39-50.

[26] Alan Shieh, Srikanth Kandula, Albert Greenberg, Changhoon Kim, and Bikas Saha. 2011. Sharing the data center network. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI'11). USENIX Association, Berkeley, CA, USA, 309-322.

[27] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In Proceedings

of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 51-62.

[28] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). SIGCOMM Comput. Commun. Rev. 40, 4 (August 2010), 63-74.

[29] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

[30] Dennis Abts and Bob Felderman. 2012. A guided tour of data-center networking. Commun. ACM 55, 6 (June 2012), 44-51.

[31] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazires, Balaji Prabhakar, Changhoon Kim, and Albert Greenberg. 2013. EyeQ: practical network performance isolation at the edge. In Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation (nsdi'13), Nick Feamster and Jeff Mogul (Eds.). USENIX Association, Berkeley, CA, USA, 297-312.

[32] Chunming Qiao and Myungsik Yoo. 1999. Optical burst switching (OBS) - a new paradigm for an optical Internet. J. High Speed Netw. 8, 1 (March 1999), 69-84.

[33] V.W. S. Chan, G. Weichenberg and M. Medard, *Optical Flow Switching* (Invited paper).

[34] Bill St. Arnaud, UCLP Roadmap for creating User Controlled and Architected Networks using Service Oriented Architecture, CANARIE., January 2006.

[35] Clapp, G.; Doverspike, R.; Skoog, R.; Strand, J.; Von Lehmen, A., "Lessons learned from CORONET," Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC) , vol., no., pp.1,1, 21-25 March 2010

[36] Ajay Mahimkar, Angela Chiu, Robert Doverspike, Mark D. Feuer, Peter Magill, Emmanuil Mavrogiorgis, Jorge Pastor, Sheryl L. Woodward, and Jennifer Yates. 2011. Bandwidth on demand for inter-data center communication. InProceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages

[37] Baldi, M.; Marchetto, G., *Pipeline Forwarding of Packets Based on a Low-Accuracy Network-Distributed Common Time Reference,* Networking, IEEE/ACM Transactions on, vol.17, no.6, pp.1936,1949, Dec. 2009

[38] R. S. Tucker, *The role of optics and electronics in high-capacity routers,* J. Lightwave Technol., vol. 24, pp. 4655 - 4673, 2006.

[39] Tucker, RodneyS.,"Green Optical CommunicationsPart I: Energy Limitations in Transport" Selected Topics in Quantum Electronics, IEEE Journal of , vol.17, no.2, pp.245,260, March-April 2011

[40] Tucker, RodneyS.,"Green Optical CommunicationsPart II: Energy Limitations in Networks" Selected Topics in Quantum Electronics, IEEE Journal of , vol.17, no.2, pp.261,274, March-April 2011

[41] RFC 793. Available at: `https://www.ietf.org/rfc/rfc793.txt`

[42] RFC 768. Available at: `https://www.ietf.org/rfc/rfc768.txt`

[43] J.K.Kurose, K.W.Ross, *Computer Networking: A Top Down Approach*, 4^{th} Ed. Addison Wesley.

[44] Andrew Tanenbaum. 2010. *Computer Networks* (5th ed.). Prentice Hall Professional Technical Reference.

[45] j.M.Simmons, *Optical Network Design and Planning*, Ed. Springer 2008

[46] Chanclou, P; Gosselin, S; Palacios, J F; Alvarez, V L; Zouganeli, E. 2006. *Overview of optical broadband access evolution,* Institute of Electrical and Electronics Engineers Communications Magazine (August): p.29-35.

[47] CISCO CRS-1 *Data Sheet*. Available at: `http://www.cisco.com/c/en/us/td/docs/\\routers/crs/crs1/8_slot/system/description/sysdesc/hq6345_a.html`

[48] Tucker, RodneyS.; Eisenstein, G.; Korotky, S.K., "Optical time-division multiplexing for very high bit-rate transmission," Lightwave Technology, Journal of , vol.6, no.11, pp.1737,1749, Nov 1988

[49] Hideyuki Sotobayashi, Wataru Chujo, and Ken-ichi Kitayama. 2003. Optical code division multiplexing (OCDM) and its application for peta-bit/s photonic network. Inf. Sci. Inf. Comput. Sci. 149, 1-3 (January 2003), 171-182.

[50] International Telecommunications Union `http://www.itu.int/en/Pages/default.aspx`

[51] Internet Engineering Task Force `https://www.ietf.org/`

[52] ITU-T G.8080 - Automatically Switched Optical Network. Available at: `https://www.itu.int/rec/T-REC-G.8080/en`

[53] RFC 3945 - Generalized Multi-Protocol Label Switching (GMPLS) Architecture. Available at: `http://tools.ietf.org/html/rfc3945`

[54] Biswanath Mukherjee, *Optical WDM Networks,* Ed. Springer;

[55] J. Baliga et al., *Power Consumption in Access Networks,* OFC 2008, paper OThT6, 2008.

[56] NETFLIX `https://www.netflix.com/`

[57] YouTube `https://www.youtube.com/`

[58] O. Tamm, C. Hermsmeyer, and A.M. Rush *Eco-Sustainable System and Network Architectures for Future Transport Networks*, Bell Labs Technical Journal 14(4), 311-328 (2010)

[59] Tucker, RodneyS.; Parthiban, R.; Baliga, J.; Hinton, K.; Ayre, R.W.A.; Sorin, W.V., "Evolution of WDM Optical IP Networks: A Cost and Energy Perspective," Lightwave Technology, Journal of , vol.27, no.3, pp.243,252, Feb.1, 2009

[60] ITU-T G.709 (02/2012) *Interfaces for the optical transport network*. Available at: `http://www.itu.int/rec/T-REC-G.709-201202-I/en`

[61] RFC 1595. Available (online) at: http://tools.ietf.org/pdf/rfc1595.pdf

[62] Gorshe S. *A tutorial on SONET/SDH*. PMC-Sierra white paper. Available at: http://pmcs.com/resources/whitepapers/sonet-sdh/

[63] Steve Gorshe. *A Tutorial on ITU-T G.709 Optical Transport Networks (OTN)*. PMCSierra white paper, *personal communication*.

[64] ITU-T G.872 (10/2012) *Architecture of optical transport networks*. Available at: http://www.itu.int/rec/T-REC-G.872-201210-I/en

[65] ITU-T G.7044. *Hitless Adjustment of ODUflex*. Available at: https://www.itu.int/rec/T-REC-G.7044-201110-I/en

[66] ITU-T G.Sup43. *Transport of IEEE 10GBASE-R in optical transport networks (OTN)*. Available at: http://www.itu.int/rec/T-REC-G.Sup43-201102-I/en

[67] ITU-T G.7041 (04/2011). *Generic Framing Procedure*.

[68] Connection-Oriented Ethernet, On-Ramp Aggregation for Next-Generation Networks. Fijitsu white paper. Retrieved on October 20 2013 from: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/COE.pdf

[69] OTN Over Packet Fabric Protocol (OFP) Implementation Agreement. Available at: http://www.oiforum.com/public/documents/oif-ofp-01.0.pdf

[70] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2008. The cost of a cloud: research problems in data center networks. SIGCOMM Comput. Commun. Rev. 39, 1 (December 2008), 68-73.

[71] Dennis Abts and Bob Felderman. 2012. A guided tour of data-center networking. Commun. ACM 55, 6 (June 2012), 44-51.

[72] Jrg Schad, Jens Dittrich, and Jorge-Arnulfo Quian-Ruiz. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. 3, 1-2 (September 2010), 460-471.

[73] Urs Hoelzle and Luiz Andre Barroso. 2009. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines (1st ed.). Morgan and Claypool Publishers.

[74] Telecommunications Industry Association, Standard TIA-942. Available at: https://global.ihs.com/doc_detail.cfm?&item_s_key=00414811&item_key_date=860905&input_doc_number=TIA-942&input_doc_title=

[75] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. SIGOPS Oper. Syst. Rev. 37, 5 (October 2003), 29-43.

[76] The Hadoop Project. Available at http://hadoop.apache.org

[77] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM, vol. 51, no. 1 (2008), pp. 107113.

[78] M. Burrows, The chubby lock service for loosely-coupled distributed systems, in Proceedings of OSDI06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November 2006.

[79] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10). ACM, New York, NY, USA, 267-280.

[80] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The nature of data center traffic: measurements & analysis. In Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC '09).

[81] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. 2011. Towards predictable datacenter networks. SIGCOMM Comput. Commun. Rev. 41, 4 (August 2011), 242-253.

[82] Iosup, A; Yigitbasi, N.; Epema, D., "On the Performance Variability of Production Cloud Services," Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on , vol., no., pp.104,113, 23-26 May 2011

[83] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. 2008. Improving MapReduce performance in heterogeneous environments. In Proceedings of the 8th USENIX conference on Operating systems design and implementation (OSDI'08). USENIX Association, Berkeley, CA, USA, 29-42.

[84] Amar Phanishayee, Elie Krevat, Vijay Vasudevan, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Srinivasan Seshan. 2008. Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08), Mary Baker and Erik Riedel (Eds.). USENIX Association, Berkeley, CA, USA, , Article 12 , 14 pages.

[85] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. Understanding TCP incast throughput collapse in datacenter networks.

In Proceedings of the 1st ACM workshop on Research on enterprise networking (WREN '09). ACM, New York, NY, USA, 73-82

[86] Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Brian Mueller. 2009. Safe and effective fine-grained TCP retransmissions for datacenter communication. In Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 303-314.

[87] Chandra, P.K.; Turuk, A.K.; Sahoo, B., "Survey on optical burst switching in WDM networks," Industrial and Information Systems (ICIIS), 2009 International Conference on , vol., no., pp.83,88, 28-31 Dec. 2009

[88] Shuping Peng; Hinton, K.; Baliga, J.; Tucker, RodneyS.; Zhengbin Li; Anshi Xu, "Burst switching for energy efficiency in optical networks," Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC) , vol., no., pp.1,3, 21-25 March 2010

[89] Barakat, N., Darcie, T. E., *Control-Plane Congestion in Optical-Burst-Switched Networks*, IEEE J. Selected Areas in Comm. - Optical Comm. and Networking Series, Aug. 2009.

[90] Jue JP, Vokkarane VM. Optical burst switched networks. New York: Springer; 2005.

[91] Chua KC, Gurusamy M, Liu Y, Phung MH. Quality of service in optical burst switched networks. Springer-Verlag; 2007.

[92] Varvarigos, E.A.; Sharma, V., "The ready-to-go virtual circuit protocol: a loss-free protocol for multigigabit networks using FIFO buffers," Networking, IEEE/ACM Transactions on , vol.5, no.5, pp.705,718, Oct 1997

[93] Widjaja, I., "Performance analysis of burst admission-control protocols," Communications, IEEE Proceedings- , vol.142, no.1, pp.7,14, Feb 1995

[94] Lisong Xu; Perros, H.G.; Rouskas, G., "Techniques for optical packet switching and optical burst switching," Communications Magazine, IEEE , vol.39, no.1, pp.136,142, Jan 2001

[95] Huang Anpeng, Xie Linzhen, Li Zhengbin, Xu Ansbi, *Time-Space Label Switching Protocol (TSL-SP): A New Paradigm of Network Resource Assignment*

[96] Wei, J.Y.; McFarland, R.I., Jr., "Just-in-time signaling for WDM optical burst switching networks," Lightwave Technology, Journal of , vol.18, no.12, pp.2019,2037, Dec 2000

[97] J. S. Turner, Terabit Burst Switching, Journal of High Speed Network, vol. 8, no. 1, pp. 316, 1999.

[98] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, Control Architecture in Optical Burst-Switched WDM Networks, IEEE Journal on Selected Areas in Communications, vol. 18, no. 10, pp. 18381851, 2000.

[99] J.P. Jue and V.M. Vokkrane, *Optical Burst Switched Networks*, Ed. Springer 2005

[100] X. Huang, V. M. Vokkarane, and J. P. Jue, *Burst Cloning: A Proactive Scheme to Reduce Data Loss in Optical Burst-Switched Networks*

[101] V. W. S. Chan, A. R. Ganguly adn G. Weichenberg, *Optical Flow Switching with Time Deadlines for High-Performance Applications.*

[102] V.W. S. Chan and G. Weichenberg, *Access Network Design for Optical Flow Switching*, Proceedings of IEEE Global Telecommunications Conference (Globecom 2007), Washington, D.C., Nov. 2007

[103] Divakaran, D.M.; Altman, E.; Post, G.; Noirie, L.; Primet, P.V.-B., "From Packets to XLFrames: Sand and Rocks for Transfer of Mice and Elephants," INFOCOM Workshops 2009, IEEE , vol., no., pp.1,6, 19-25 April 2009

[104] Reid, A.; Willis, P.; Hawkins, I.; Bilton, C., "Carrier ethernet," Communications Magazine, IEEE , vol.46, no.9, pp.96,103, September 2008

[105] *Connection-Oriented Ethernet Completing the Ethernet Revolution.* Fujitsu white paper. Retrieved on October 20 2013 from: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/ConnectionOrien tedEthernetRevolution.pdf

[106] *Connection-Oriented Ethernet, On-Ramp Aggregation for Next-Generation Networks*. Fujitsu white paper. Retrieved on October 20 2013 from: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/COE.pdf

[107] Yong Liu, Kee Chaing Chua, and Gurusamy Mohan, *Achieving High Performance Burst Transmission for Bursty Traffic using Optical Burst Chain Switching in WDM Network*, IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 58, NO. 7, JULY 2010

[108] *Multi-Protocol Label Swtiching Architecture* , IETF - RFC 3031, Available at `https://tools.ietf.org/html/rfc3031`

[109] *Label Distribution Protocol (LDP)*, IETF - RFC 5036, Available at `http://tools.ietf.org/html/rfc5036`

[110] J. Baliga et al., *Energy Consumption in Optical IP Networks,* J. Lightwave Tech., vol. 27, no. 13, 2009, pp. 2391403.

[111] OMNeT++ Discrete Event Simulation Tool, Available at `http://www.omnetpp.org`.

[112] Nallatech PCIe-395 - FPGA Network Processing Card, specifications available at: `http://www.nallatech.com/images/stories/product_briefs/pcie_395pb_v1_6.pdf`

[113] *SDN Architecture*. Issue 1, June 2014. Available at: `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf`

Appendix A

Power-efficient Electronic Burst Switching for Large File Transactions

Ilijc Albanese[1], Sudhakar Ganti[2] and Thomas E. Darcie[1]
[1] *Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada*
[2]*Department of Computer Science, University of Victoria, Victoria, BC, Canada*

[Published in: 2nd International Conference on Smart Grids and Green IT Systems (SMARTGREENS 2013)]

Abstract:

Much of the growth in bandwidth demand and power consumption in todays Internet is driven by the transport of large media files. This work presents a power-efficient overlay network specifically designed using electronic burst switching for these large files. The two approaches are presented in which electronic bursts or media frames (MF) containing $> 1Mb$ are routed in a manner similar to UDP or concatenated into periodic semi-transparent chains and routed using a two-way reservation protocol. Utilization, blocking, delay and buffer size are compared to UDP/IP by means of simulation. Both approaches dramatically reduce header-related power consumption. Concatenation also reduces significantly the amount of buffer space required. A representative router design is evaluated showing a potential energy saving of roughly 80% relative to standard IP routers.

I. INTRODUCTION

Internet routing techniques have enabled the unprecedented growth of a rich diversity of data applications, applications that have become an essential part of business, entertainment and social interaction. Rapid compounded growth in demand for these applications has driven strong advances in router and optical transmission technologies. But despite ongoing improvements in bandwidth capacity and power efficiency, power consumption in router networks continues to be a concern. Estimates suggest that Internet-based communication technologies consume between $2-3\%$ of power generated globally and that this number is increasing at a rate of $16-20\%$ per year [1]. As a result, extensive effort is ongoing to develop new techniques for minimizing power consumption in future Internet technologies.

Of particular interest is power consumed in electronic routers. Numerous studies have dissected the operation of Internet Protocol (IP) routers and shown that a significant portion of power consumed can be attributed to header processing on each packet. With the increasing popularity of bandwidth intensive applications such as streaming video and the sharing of large files, studies support the general observation that file sizes are growing rapidly. Given the large number of IP packets required for these transactions, header related power consumption is a key contributor to the rapid growth of power consumption in routers.

Obviously, energy efficiency improves if larger packets are used for large file transfers. In response, the maximum IP packet size was extended from 1500 bytes (B) using Jumbo and Super Jumbo frames pushing packet sizes to $64KB$. However, these specifications are not in widespread use due to problems related to backwards compatibility [2] and network latency arising from integration of very large packets with smaller packets within the same links. Integration of large (up to $19.5KB$) and standard packets within the same network was considered [3] showing that the use of larger packets may reduce power consumption and computational load required from the network hardware. However, coexistence of large and small packets in the same links results in unfairness and higher drop rates for both types of packets, leading to inefficient bandwidth and computing power utilization.

This raises the question as to the potential value of using a separate overlay network for the traffic associated with large file transfers. Overlay networks have been a fundamental component of the evolution of telecommunications networks wherein generations of switching and routing technologies (voice, Frame Relay, ATM, IP) have evolved on top of a common optical transport layer. SONET and more recently

optical add/drop multiplexers allow convenient partitioning with wavelength or sub-wavelength granularity of provisioned, managed, and restored transmission capacity to whatever service layer intelligence is required. While clearly the introduction of a new overlay network would have to be predicated on compelling value, it would be unwise to suggest that, given power consumption and scaling considerations of IP, a next generation non-IP switching/routing approach cannot exist.

What might be the form of this new overlay network? On one extreme, numerous optical networking approaches offer up to an entire wavelength for some time, through which GigaByte (GB) files can be delivered. Optical burst (OBS) [4] or flow switching (OFS) [5], or user-controlled end-to-end lightpaths (e.g. CAnet4, MONET, CORONET and GRIPhoN[6]) have been explored fully. For example, an OBS approach [7] uses concatenated data bursts where the data units are organized as non-contiguous and non-periodic series of concatenated timeslots (bursts), which are then handled as a whole in an all-optical network infrastructure. These optical approaches are not embraced by industry, in part because the power efficiency of optical switching is questionable [8],[9] and optical buffers, widely used in most OBS proposals, have not yet offered a commercially viable alternative to electronic buffers [9]. Also, while capable of supporting large bandwidth, targeted implementations are in the interconnection of specialized nodes (i.e. campuses) rather than broadly distributed Internet users.

Hybrid architectures have been studied in which both electronic and optical switching are combined [10] to simultaneously handle packets, bursts and TDM circuits. A large reduction in the power consumption is achieved by selecting adaptively which part of the node to activate based on a per-flow evaluation of the data to be routed while the other blocks are put in sleep or low-power mode. While potentially powerful, this approach requires the complex integration of disparate switching and control elements, some of which (like OBS and optical delay lines) have not proven compelling individually.

A more incremental overlay network approach is electronic burst switching (EBS) [11]. Following the OBS model, bursts are assembled at edge burst switches and switched electronically at core switches. It was concluded that using large bursts ($> 1Mb$) may lead to reduction in header-related power consumption in core switches, but the power consumed by burst assembly negates much of the advantage gained in core switching.

In this paper we continue along the path of EBS. Users share the bandwidth of an overlay network, which we presume to be statically provisioned, using electronic switches or routers specifically designed to handle large file transactions. Unlike [11], we eliminate burst assembly at edge switches and consider direct end-to-end delivery of large "media" frames (MF) (roughly $1-10Mb$) to users through an overlay to next-generation optical access networks. Free from the constraints of coexisting with highly granular and dynamic IP traffic, this EBS overlay network can be designed specifically for the efficient delivery of the large data transactions that did not exist when the Internet was conceived. Compared to traditional IP routers, switch reconfiguration can be far less dynamic since only very large packets are supported. Unlike proposed optical alternatives, this can be accomplished using available electronic buffers in a form that is entirely compatible with todays highly efficient cross-point switch arrays.

Our objective is to enable a significant reduction in power consumption of network hardware while optimizing the use of resources. We first explore routing MFs using a standard UDP protocol (MF-UDP). UDP is selected for this study, rather than TCP, as this avoids numerous complexities that add little insight to a comparison with conventional IP and, as discussed later, gives the best case scenario for IP. Based on simple hardware considerations, network performance simulations and comparison with traditional UDP, we arrive at the anticipated conclusion that router power consumption is reduced dramatically, but performance is otherwise unaffected and larger buffers are needed.

We then consider using concatenations of MFs into periodic semi-transparent chains (MFCs) and the scheduled transmission of these MFCs using a two-way reservation mechanism. While such a scheduling mechanism would be inappropriate for traditional IP traffic, the large size of each MFC (e.g. $1GB$) makes scheduling both manageable and worthwhile. Also, the structure of an MFC makes it easy to condense information on its configuration with minimal control plane information, minimizing the amount of information to be processed at each node to schedule the chain and reducing the probability of control plane collisions. Simulation results show increased utilization efficiency and decreasing buffer requirements in comparison to MF-UDP as well as standard UDP. An MFC router is designed based on a commercially available cross-point switch array and power consumption is estimated to be roughly 20% of that of a standard IP router.

The paper is organized as follows: Section II provides an overview of the reference network architecture in the context of transactions of large files. Section III

compares, using *OMNeT++* simulation, traditional UDP to MF-UDP in supporting representative large transactions. In Section IV, chains of media frames (MFCs) are introduced, along with exemplary admission control and scheduling algorithms, and compared to UDP and MF-UDP. Router implementation is discussed in Section V, where implications on power consumption are considered and a representative MFC router is evaluated.

II. NETWORK TOPOLOGY AND LARGE FILE FLOWS

Our discussion is framed by the reference network architectures shown in Figures A.1 and A.2. Figure A.1 shows a hierarchical network representative of the current state-of-the-art comprising various sizes of routers (access, edge, and core) connected to a transport network through various sizes of add-drop multiplexers. In present implementations the add-drop multiplexers would use SONET, but this reference architecture also supports optical add-drop technologies. An important component is interconnection between various carrier networks through peering arrangements, typically facilitated through regional or Metro-area exchange points.

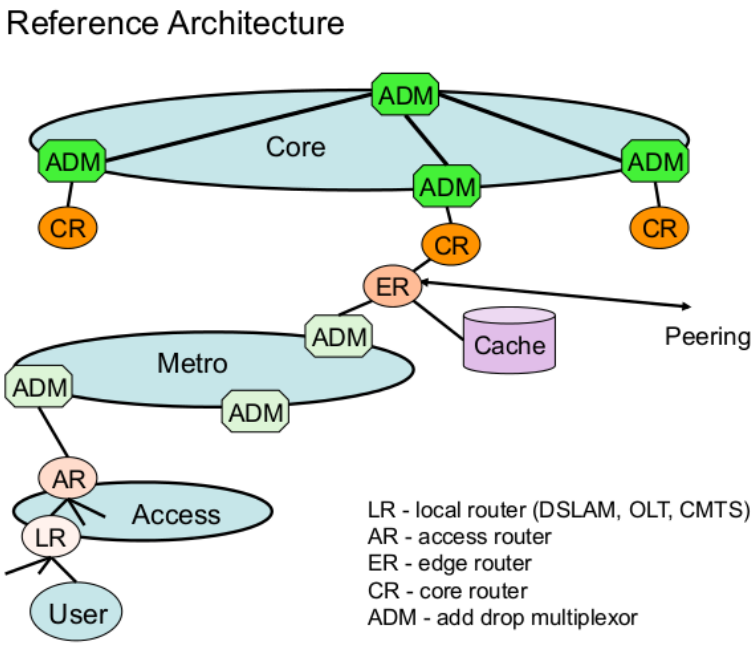


Figure A.1: Reference Architecture

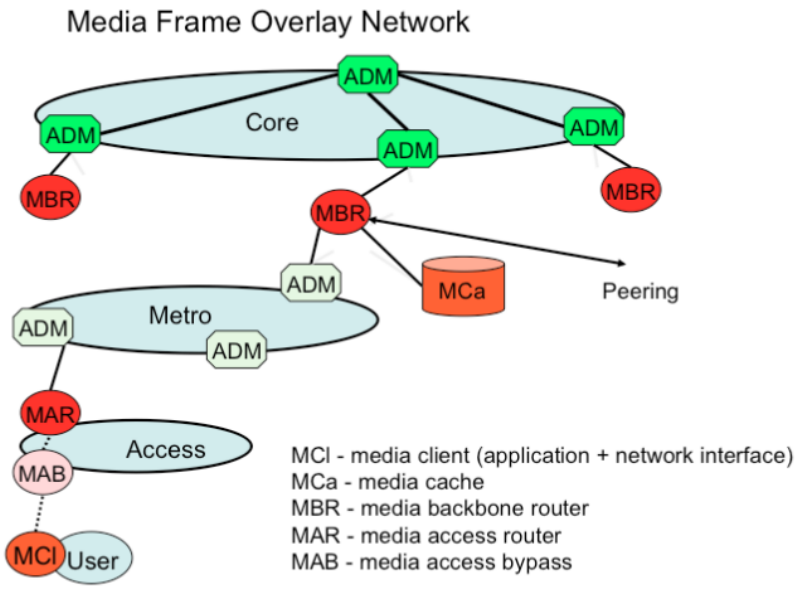


Figure A.2: Media Frame Overlay Architecture

Primary examples of large-file transactions can be super- imposed onto this reference architecture. These examples include: 1) Regionally-cached download: In this case large media files are downloaded from regional cached distribution servers at the end points of Content Delivery Networks (CDN), through Metro and access to end-users. Driven by rapid proliferation of video-related download applications and as evidenced by the popularity of CDN, these downloads represent a significant fraction of traffic growth, and therefore are the focus of this study. Other important transactions include: 2) Source-to-cache distribution: To deliver and update content to CDN servers, large files must be distributed from sources, typically across a core network, to the regional cache; 3) End-to-end file transfers: For peer-to-peer applications, or for files for which widespread distribution is not anticipated, the regional cache is bypassed and files are transacted through Metro and access, possibly across the core network, directly to an end-user.

An overlay network designed specifically for large file transactions might look like Figure A.2. Each of the "media" boxes parallels a present-day IP counterpart and supports the origination or termination (media client interface (MCI)), access bypass (media access bypass (MAB)), admission (media access router (MAR)), and efficient end-to-end routing (media backbone router (MBR)), in accordance with the principles described below for MFs. In addition, it is convenient to consider regional storage of MFs in a media frame cache (MCa).

An obvious challenge to any new overlay network is traversing access, where the large embedded base consisting of a diverse array of existing systems is in place and costly to replace. Since our intention is not to restrict application to specialized nodes (e.g., campuses), but rather to reach broadly distributed consumers, methods must be established to transport MFs through access. Present broadband access systems, including Passive Optical Network (PON) and Hybrid Fiber Coax (HFC), share bandwidth using IP-centric protocols that would require significant change. Two alternative solutions are represented in Figure A.2. A conservative approach involves an application operable between the MCI and the MAR such that using traditional broadband access alone, MFs and MFCs are assembled or disassembled at the MAR. This functionally is parallel to the burst assembly routers proposed for use in OBS and EBS. A preferred approach involves engaging the evolution of optical access standards, where standards for $10Gb/s$ PON have emerged recently, to enable higher dedicated bandwidths perhaps through wavelength overlays. MFs and MFCs would then be assembled at the user end point or client directly.

III. MEDIA FRAMES VERSUS CONVENTIONAL IP

We first explore the issues associated with migrating very large file transfers to a separate overlay network in which the standard unit of bandwidth is a media frame (MF) containing roughly $1-10Mb$ of data plus overhead. In [11] it was concluded that although using large packets would lead to considerable power savings, increasing the frame size beyond $1Mb$ would only marginally increase the energy efficiency of an EBS node. However, using larger frames also reduces the required reconfiguration speed of the switch fabric, minimizing requirements on switching speed and inefficiency introduced during transitions.

Overhead may include address, priority, concatenation details (specifying MFCs, as discussed later), coding, guard time and management information. Given the very large capacity within each MF, considerable header information (e.g. $10KB$) can be included with minimal impact on throughput. Structure may be defined to facilitate error correction, segmentation, security, file compression, and easy assembly from a large numbers of smaller IP packets.

An obvious method for networking with MFs is to adopt the same concepts as used with TCP/UDP, allowing each MF to be routed in accordance with predefined routing tables through a connectionless queuing network. While the dynamics are considerably different than with $< 1500B$ UDP packets, the underlying issues are the same.

To explore this in detail, both UDP and MF-UDP network simulators were built using *OMNeT++* [12] and the performances compared in terms of link utilization, buffer space occupied and delay per GB of data transferred. UDP was simulated using $1500B$ packets and MF-UDP using $1Mb$ packets. Droptail queues were used for both 1Mb and $1500B$ UDP packets. For standard UDP packets the maximum buffer size was set to 1000 packets, corresponding to roughly $1.5MB$. Buffers of the same size were used for MF-UDP.

A dumbbell topology was considered for the simulations (Figure A.3). While more complex topologies could be simulated, this represents the case of our reference network (Figure A.2) with a congested link between multiple source servers and end users.

The capacity of each link is set to 10 Gib/s (i.e. $10 * 2^{30}$ Gb/s according to IEC standard) and each source is offering the network an average load of roughly 1.33 Gb/s. Various load conditions were tested by activating more source-destination pairs and the offered load was made to vary from 25% to 150% of the bottleneck link capacity (corresponding to from 2 to 12 source-destination pairs). Each source transmits data to one destination only and all sources compete for the same bottleneck link.

Results below show that link utilization (Figure A.5), blocking (Figure A.6), and delay (Figure A.7) versus offered load are very similar despite the 3 orders-of-magnitude change in packet size. However, obviously and importantly, the number of headers processed is reduced by the same factor.

Compared to UDP, the router switch fabric becomes far less dynamic. Reconfiguration occurs far less frequently (by 3 orders of magnitude). It remains to be seen if an MF-router can be designed to exploit this less dynamic reconfiguration and negligible header processing with a sufficiently large net increase in power efficiency to justify a separate overlay network. This is addressed further in Section V.

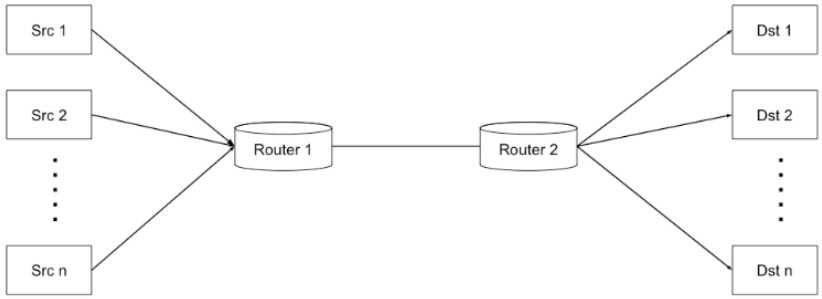


Figure A.3: Topology Used for Simulations

IV. CONCATENATED MEDIA FRAMES

While UDP-like MF routing dramatically eases processing (header and switch reconfiguration) over traditional IP, performance is otherwise essentially the same. We now consider the potential impact of concatenation of MFs into larger structured chains (MFCs). Concatenation creates single entities that would contain an entire large transaction, for example a multi-GB movie download. Discussion centers on two key considerations: scheduling and transparency. Scheduling and admission control become worthwhile for such large transactions and these can be used to increase resource utilization and minimize buffer size. However, serving such large transactions continuously in time introduces significant latency for waiting transactions and is incompatible with the lower end-user client and access network throughputs. Making each MFC partially transparent overcomes both problems. We limit our discussion here to a simple functional description of a representative methodology, including MFC structure, transparency, signaling and scheduling algorithm, then compare performance to conventional UDP and the MF-UDP described in Section III.

IV-A Media Chain Transparency

A MFC with transparency degree 3 (defined as the number of interstices between two consecutive MFs in a chain plus 1) is illustrated in Figure A.4.

Using periodic semi-transparent chained data structures provides five primary functions. First, given the large size of each MF, concatenations of multiple MFs without transparency would introduce substantial latency by occupying network resources for substantial durations ($8000MF = 1GB = 200ms$ @ $40Gb/s$ OC768 transmission). The use of transparency allows servicing multiple MFCs without introducing long delays, albeit at a slower data rate. Secondly, a fixed transparency simplifies scheduling of large amount of data with minimal computational load. Third, the semi-transparent structure of each MFC allows the use of buffering as a means to affect the relative timing of an MFC with respect another in order to interleave MCs competing for the same link, but without buffering entire chains. Our hope is that this use of the buffer will enable significant savings in buffer space occupied relative to the UDP and MF-UDP of Section III.

Fourth, we anticipate bit rates of $10Gb/s$ in access networks at a time when bandwidths in core networks will be $100Gb/s$. Transmission from access must then be up-shifted in rate, resulting in, for this case, 10% time occupancy. This would be accommodated naturally by a core transparency degree 10 times higher than that used in the access. Finally, for file transfer applications one desires to send as much

information as fast as possible. For this transparency is a disadvantage. However, streaming applications have evolved to deliver segments of large files spread out over long time periods. A large transparency degree can be used to emulate streaming delivery while retaining the other worthwhile attributes described above.

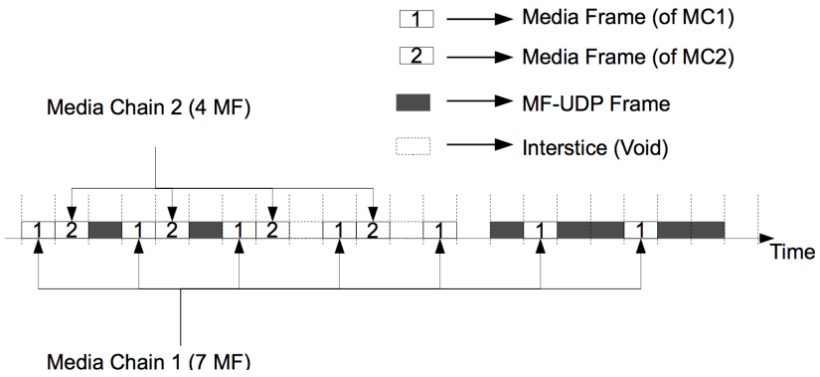


Figure A.4: An Example of how two MFCs, MF-UDP Frames and Voids Fit in a Channel

IV-B Signaling and Control

Signaling is needed to establish and update the network state, schedule MFC transmission, acknowledge receipt, and many other functions. Information about an entire MFC, including length, scheduling information, priority, etc. can be easily contained within each MFC, MF in an MFC, or a small separate control packet. Signaling could be "in-band" using periodic time slots within the MF transport structure, or "out-of-band". Our preference is to exploit the ubiquitous availability of traditional IP networks for out-of-band signaling. In what follows, we assume that each of our media access and backbone routers (MAR, MBR in Figure A.2) are able to signal through a suitable IP network.

Global Control: Since each signaling event corresponds to $1GB$ of data, the number of signaling events is small. It is then reasonable to use a centralized state server to provide each router with global path, timing, and occupancy information. Each router updates status to the state server regularly, and the state server is able to calculate paths and approve initiation of a request for MFC scheduling, as discussed below. The state server must know the topology and is then able to make globally informed decisions to queries from routers. It is also useful to know the propagation time between nodes for efficient scheduling. Numerous methods can be implemented, like the ranging protocol used in PON, to estimate these times and report them to the scheduling server.

Distributed Control: Each router communicates directly with its neighbors and the state server. Each router continually updates the state server of status and load, and MARs request path and approval for MFCs from the state server. Approval does not guarantee success, but suggests high probability. Communication between routers along the path determines ultimate success, as described below. This minimizes latency in each MFC request-grant negotiation.

IV-C Scheduling

The objectives of scheduling are to organize transmission of concatenated chains in such a way as to minimize hardware complexity and power consumption, to maximize link utilization and to minimize buffer requirements. All of these objectives can be addressed through the use of an Expected Arrival Time (EAT_i) of the MFC to the next node in its path. This is computed based on the physical distance between the nodes, which is assumed to be known by each scheduling node, and included in a control packet CP ($< 1Kb$). The CP is used to reserve resources for its associated MFC along its path.

1. MAR queries state server for path, propagation time associated with each hop in path

2. MAR estimates a Time-to-Transmit (TT) parameter. All routers along path use TT to search for available time slots. TT is determined based on transmission and propagation delay from user to the MAR, hop distances along path and processing time for control packets at each node.

3. MAR generates control packet (CP) and sends it to next node along path. CP contains sender/destination address, length of MFC, transparency degree, expected arrival time ($EAT[N_i]$), and ID that associates each CP to an MFC. $EAT[N_i]$ indicates to the node receiving the CP the amount of time, after the reception of the control packet, before the arrival of first bit of the MFC

4. EAT field in CP is updated before forwarding CP to next hop node, in order to account for the additional transmission, buffering and processing delays at each node. This continues until destination (egress MAR) is reached or until the reservation process fails.

5. If reservation succeeds, confirmation packet is sent over IP network directly to source node which starts transmission of MFC. If reservation fails, a $NACK$

packet is sent over reverse path to free resources and source will retry after random back-off time.

Since the estimated EAT for the MFC is computed using the physical hop delay (known globally) and the expected arrival time is carried in the CP, there is no need for network-wide synchronization. A local timer at each node keeps track of the time differences between the reception of the CP and the expected arrival time of its relative MFC. The guard time in each MF compensates for the time uncertainties in this estimation process while buffering is used to align incoming MFCs within the interstices of already scheduled MFCs. Therefore, there is no need to buffer an entire MFC.

IV-D Simulation

The scheduling algorithm was implemented (details to be published) using *OMNeT++* and the same topology used in Section III. Each MF is assumed to contain $1Mb$ of data and each MFC is defined as the concatenation of 8000MF (i.e. $1GB$ of data). A fixed transparency degree of 8 was chosen for all MFC simulations. The performances of the MFC-based transport are then compared to those of UDP sources using MF ($1Mb$) and standard UDP packets ($1500B$).

Given the functioning of the algorithm presented in Section IV-C, the payload bits arrive at the node only if the reservation process was successful. In order to compare this to a connectionless protocol as in Section III (UDP) it was assumed that each source-destination pair would attempt to transmit on average the same amount of data. Hence for the UDP cases (both standard and MF-UDP) the *offered load* is the load physically reaching the bottleneck node. For the MFC case, the offered load is computed based on the number of reservation attempts per second. The maximum buffer size allowed for the bottleneck router (router 1 of Figure A.3) was set to be $1.5MB$ for all cases.

IV-E Simulation Results

Link Utilization: As can be seen in Figure A.5 the link utilization is very close for all 3 approaches tested for load conditions up to 74%. Beyond this point the higher cost of dropping larger frames comes into play and the bandwidth efficiency of MF-UDP is reduced. For load greater than 90% load packet dropping also affects standard UDP and its utilization drops below that of MFC. MFC utilization is 9% higher for higher loads. The highly structured MFC and the scheduling algorithm allow interleaving large amounts of data with link utilization similar to that of time

division multiplexed systems and performance consistently better than both other cases under high load conditions.

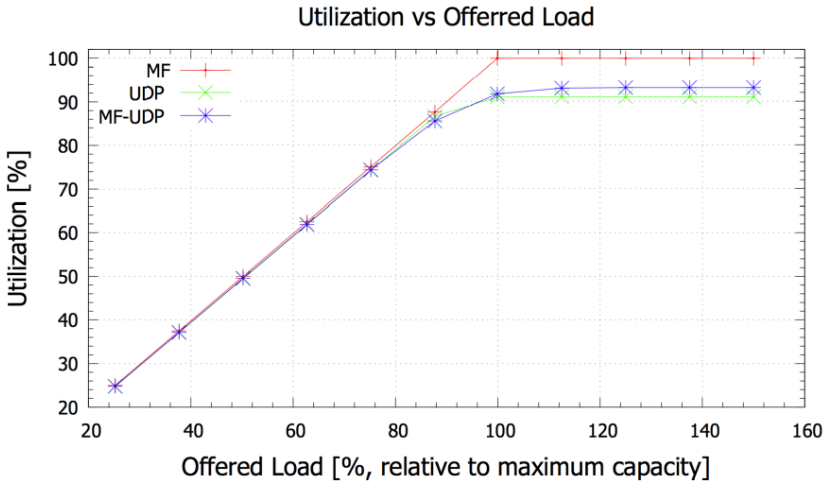


Figure A.5: Link Utilization Vs Offered Load

Packet Dropping and Blocking: Packet dropping (two UDP cases) and blocking (MFC) are presented Figure A.6. The random nature of the arrival for UDP packets allows for the possibility of filling the queue at the bottleneck node even if maximum load has not yet been reached. In the reservation-oriented MFC system call blocking only occur after the maximum bottleneck link capacity has been effectively reached.

It is important to note that when a packet is dropped, payload bits are discarded and these may have already used resources along their path (buffer space, switching power, bandwidth, etc.). When an MFC request is blocked, we are simply rejecting an attempt to transfer the data and not the data itself. This advantage of reservation-based mechanisms in terms of efficiency of resource usage is well known.

Delay: To compare the delay performance a $1GB$ transaction was taken as a reference quantity. For MFCs, upon failing a resource reservation attempt, the source simply backs off for a random amount of time before re-attempting the reservation procedure. The back-off time is exponentially distributed with an average equal to the duration of an entire chain. Re-transmission attempts were also taken into account in the delay performance measurement, as shown in Figure A.7. Up until 75% load, the delay experienced by the MFC system is virtually identical to that of both UDP cases. Beyond this point using MFCs reduces delay. Besides offering equal delay performances to UDP for the majority of the range of operation of the network, MFC

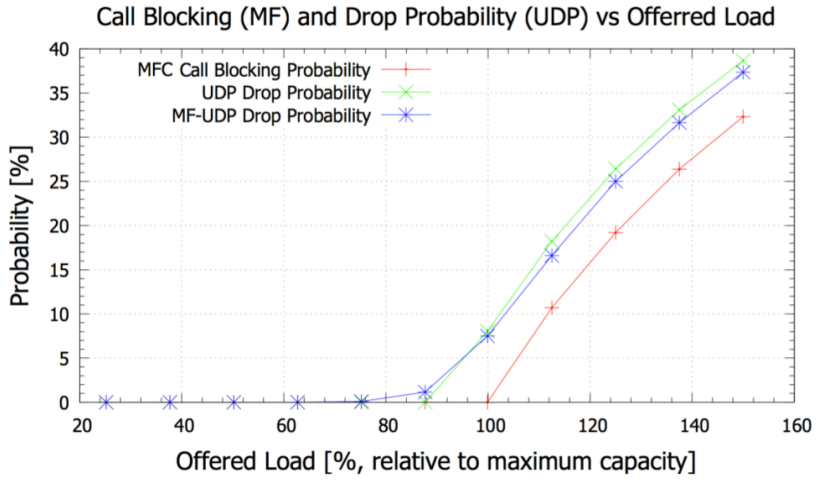


Figure A.6: Packet Dropping and Call Blocking Vs Offered Load

also provides reliable data transfer, which cannot be achieved by UDP due to the statistical nature of arriving packets.

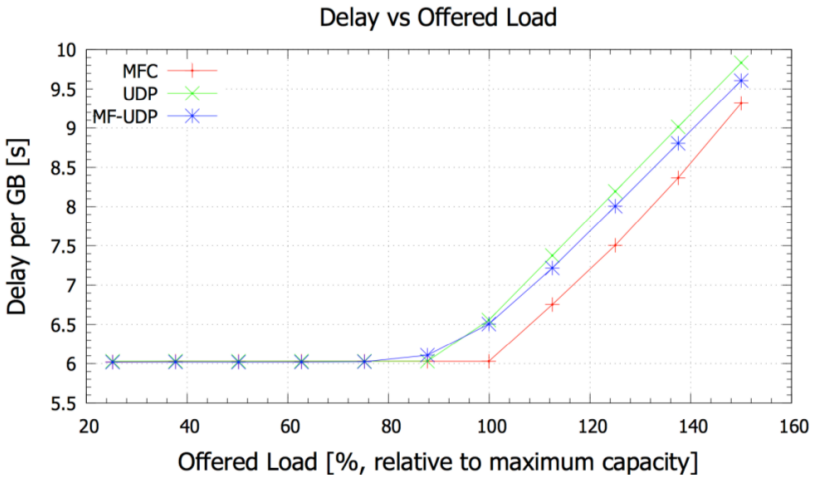


Figure A.7: Delay(per GB of data transferred) Vs Offered Load

Buffer size: As shown in Figure A.8, the buffer size required at low load for MFC and MF-UDP is higher than that occupied by standard UDP (with MF-UDP occupying the largest buffer space). As the load increases ($> 88\%$) MFC requires considerably less buffer space than both MF-UDP and standard UDP, which beyond a certain load quickly start to fill buffers up to the maximum capacity. The periodic data structure of the MFC and the scheduling algorithm bound the required buffer space to about $2 - 5$ times less than that of both UDP approaches. For both UDP

and MF-UDP the buffers are required to avoid dropped packets, while for MFC the buffers are used to align MFCs in time with scheduled slots. This results in a buffer size that depends much more on the size of the MF within the MFC than on the offered load. This dependency is shown in Figure A.9 where various media frame sizes were tested for MFC. Varying frame size results in virtually identical performances in terms of blocking probability and utilization but a significant increase of the required buffer space. Similarly, reducing the MF size for each MFC can reduce the buffer space. Increasing the frame size for the MF-UDP case would result in buffer sizes that would simply become impractical. In the MFC case using frames larger than 1Mb may enable a relaxation of the reconfiguration speed for the switch fabric as well as a reduction of the CPU utilization (see section V-A) while keeping the buffer size limited.

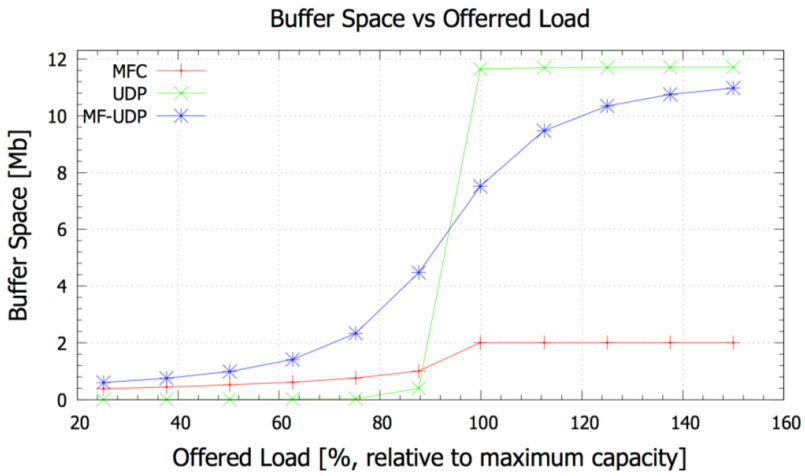


Figure A.8: Buffer Occupancy Vs Offered Load

Packet Processing: The graph in Figure 10 shows the number of packets processed per second for the 3 approaches. Obviously, either approach using MFs dramatically reduces the power consumption of header-related functions in routers to negligible levels.

V. DISCUSSION AND ANTICIPATED BENEFITS

Results for MFC indicate a considerable advantage in terms of buffer size as well as a large reduction in processing power with respect to UDP. A comparison with TCP would have been useful in that, unlike UDP and more like MFC, TCP can guarantee the correct delivery of the file transferred. TCP acknowledgements and retransmissions would have made the delay per GB much higher and bandwidth uti-

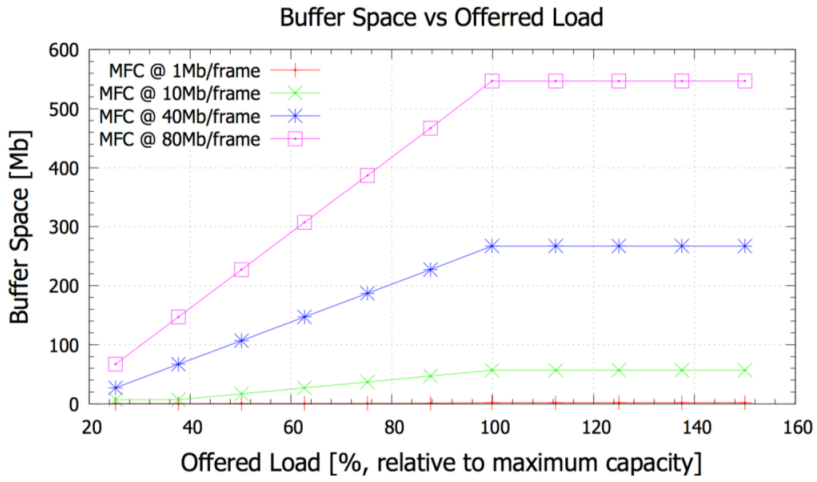


Figure A.9: Buffer Occupancy Vs Offered Load for Various Frame Sizes for MFC

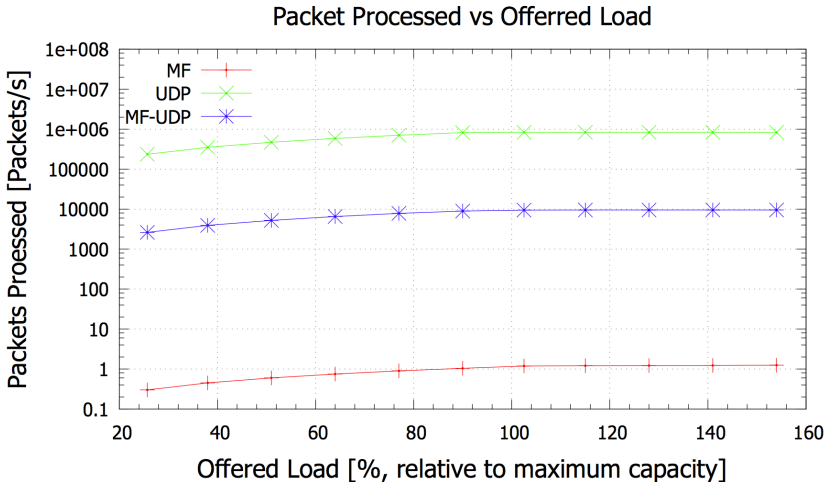


Figure A.10: Packed Processed per Second Vs Offered Load

lization much lower than that achieved with UDP. In other words, for our simulation study UDP is the best-case scenario for IP networking.

V-A Impact on Router Power Consumption

Several studies have described scaling difficulties in IP routers and broken down power consumption into the constituent functional processes. In [13] the issue of power consumption in large scale networks is addressed concluding that power consumption of header related functions is far larger than that consumed by the switching fabric. Apparently, most power is consumed in data processing functions (i.e. header parsing, address lookup, etc.), which must be carried out for each packet traversing a router.

MF-UDP and MFC offer significant reductions in the number of packets to be processed. Consider a state-of-the-art router working at $1Tb/s$. Depending on the manufacturer such a router can consume about $4KW$ [14][15], or an overall energy per bit of roughly $4nJ/b$.

Estimates of the power consumption of the various functional blocks of a similar IP router can be found in [8]. From this study it is clear that, other than power supply and cooling blocks, which are largely dependent on the energy needed by the other blocks, the forwarding engine consumes the most power, using about 32% of the energy supplied to the router.

Assuming an average packet size of $10Kb$ [16] means that a $1Tb/s$ router will have to perform header related operations approximately 10^8 times per second [14]. Organizing data in MFCs carrying roughly $1GB$ of payload can reduce this by many orders of magnitude.

In addition, the required processing speed is reduced so that a much slower processing unit can handle the same data throughput. From the study presented in [17] it is also reasonable to say that the use of large frames, together with organizing the data within the sources in large blocks (i.e. MFs) will allow a significant reduction in CPU utilization in terms of number of memory accesses and IRQs that the CPU has to handle. This may lead to additional power savings in data storage servers.

Given the large size of the MFs, requirements on the reconfiguration speed are significantly relaxed, as well as the amount of control information needed to drive the switch fabric, with further impact on the power consumption. Further reduction in the amount of control information is achieved when using MFCs, as a result of the predictability of the periodic payload.

V-B Power Consumption of Example MFC Router

A schematic of an MFC-based router based on a commercially available cross-point switch array is shown in Figure A.11.

Input data is converted into the electrical domain and data streams from each channel are de-multiplexed into their constituent MFCs. Each MFC is then delayed by the amount indicated in its associated CP. The number of buffer queues needed depends on the number of chains the device is able to handle and is given by *# of dedicated buffer queues = # of input channels * # of MFCs per channel*. The total buffer size also depends on the number of simultaneous flows the device must handle and the transparency degrees of the chains.

At the output of the buffer stage, chains competing for the same output channels are synchronized in order to allow interleaving. MFCs are then passed to the switch fabric, which simply routes each MFC to the appropriate output with no further buffering or processing.

In order to estimate the power consumed by our proposed router shown in Figure A.11, the VSC3144-12 has a switching capacity of $1.2Tb/s$ with a power consumption of about $20W$ [18]. Regarding the buffer stage, values for the power consumption of electronic memories are largely dependent on implementation and size. If we assume our MFC router would use a memory with similar size and structure to that used in [14], using the data on power consumption for a router from [8] gives roughly $200W$ for the buffer stage power consumption. Similarly we can estimate the power consumption of O/E/O blocks (including Tx/Rx equipment) to be about $280W$.

The forwarding and routing engine, using about 32% and 11% of the total energy consumed by a standard IP router, respectively [8] (i.e. a total of $\sim 720W$ [14]), are grouped in the *control* block of Figure A.11 and, as a result of the drastic reduction in the number packet processed (See Figure A.10) will be most likely reduced to negligible levels.

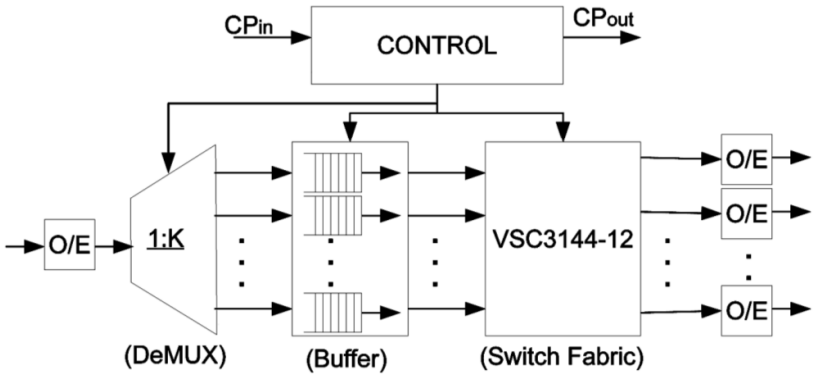


Figure A.11: Concatenated Media Frame Router

The input de-multiplexing stage, which de-multiplexes MFCs from each channel, can be modeled by [19]:

$$E_{DeMUX} = E_0 * Log_2(k) \tag{A.1}$$

Where E_0 is the energy per bit for a $1:2$ de-multiplexer and k is the number of output ports. Assuming 2010 technology, $E_0 = 10pJ$ [18]. We can set $k = 144$ in Equation A.1 obtaining a total energy for the input stage of about $71W$ or $71pJ/b$.

Aside from power supply and cooling equipment, the power consumed by the MFR would be roughly $571W$. Power supply and cooling would consume about 33% of the power consumed by the rest of the device [11], leading to total power consumption for the MFR of roughly $759W$. This is roughly 19% that of a state-of-the-art IP router working in the same throughput range and load conditions [14]. This is dominated by the power consumed by the buffer stage. Recognizing that these buffers function more like slowly reconfigurable delay lines than the buffers used in typical IP routers, further study may reveal even more significant reductions in power consumed.

VI. CONCLUSIONS

In this paper we study the potential advantages of using an overlay network in which only large media frames (MFs) $(1 - 10Mb)$ or concatenated frames (MFCs) are used to efficiently transfer large files. Numerical simulation is used to compare the use of MFs using a traditional UDP routing protocol (MF-UDP) to traditional UDP. Little difference is observed in delay, utilization and throughput, while the large packets dramatically reduce header-related processing load.

In an effort to reduce buffer size and improve resource utilization, a reservation-based networking approach is developed using MFCs and compared to MF-UDP and UDP through simulation. A reservation system is defined for scheduling MFC transmission, eliminating wasted network resources since data leaves the source only if service is guaranteed. Results show that buffer size can be reduced by at least a factor of 2 under high load conditions and the scheduling of large transactions can increase efficiency to close to 100%. Further advantages of using periodic, semi-transparent MFCs is the ability to schedule large amount of data with minimal header processing and to reduce the reconfiguration speed requirements of the switch fabric, ultimately reducing power consumption.

A representative MFC router is designed and power consumption is estimated, under conservative assumptions, to be roughly 20% that of a traditional IP router.

Other ongoing studies include methods to allow coexistence of scheduled MFCs and directly routed MF-UDP within the same routers and links, and extension of the simulations to other network topologies. Our objective is to more definitively articulate the cost-benefit trade-off, where the cost is the rather large barrier associated with the creation of a new overlay network.

In summary, the use of very large packets (MFs) and concatenations of these (MFCs) offers an interesting path to more power-efficient networking for the dominant

and rapidly growing portion of Internet traffic that comprises very large (i.e. $1GB$) transactions.

References:

1. G.Fettweis, E. Zimmermann, ICT Energy Consumption Trends and Challenges, 11th International Symposium on Wireless Personal Multimedia Communications (WPMC 2008)

2. Alteon Networks, white paper, available at: `http://staff.psc.edu/mathis/MTU/\\AlteonExtendedFrames_W0601.pdf`

3. Divakaran, D.M.; Altman, E.; Post, G.; Noirie, L.; Primet, From Packets to XLFrames: Sand and Rocks for Transfer of Mice and Elephants, in IEEE INFOCOM Workshops 2009.

4. J. P. Jue,V. M. Vokkarane, Optical Burst Switched Networks, Ed. Springer 2005.

5. Chan, V.W.S.; , "Optical flow switching", Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC) , vol., no., pp.1-3, 21-25 March 2010.

6. Ajay Mahimkar, Angela Chiu, Robert Doverspike, Mark D. Feuer, Peter Magill, Emmanuil Mavrogiorgis, Jorge Pastor, Sheryl L. Woodward, and Jennifer Yates. 2011. Bandwidth on demand for inter-data center communication. InProceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages

7. Yong Liu, Kee Chaing Chua, Gurusamy Mohan, Achieving High Performance Burst Transmission for Bursty Traffic using Optical Burst Chain Switching in WDM Networks, IEEE Transactions on Communications, Vol.58, Issue 7 pp. 2127-2136, 2010.

8. Tucker, R. S., Parthiban, R., Baliga, J., Hinton, K., Ayre, R. W. A., Sorin, W. V., Evolution of WDM Optical IP Networks: A Cost and Energy Perspective, IEEE J. Lightwave Technology, Vol. 27, Iss. 3, Feb. 2009, pp. 243-252.

9. R. S. Tucker, The role of optics and electronics in high-capacity routers, J. Lightwave Technol., vol. 24, pp. 4655 - 4673, 2006.

10. Slavisa Aleksic, Matteo Fiorani, and Maurizio Casoni, Energy Efficiency of Hybrid Optical Switching, ICTON 2011

11. S.Peng,K.Hinton,J.Baliga,R.S.Tucker et.al, Burst Switching for Energy Efficiency in Optical Networks, OSA/OFC/NFOEC 2010

12. OMNeT++ Discrete Event Simulation Tool, http://www.omnetpp.org

13. Aleksic, S., "Analysis of Power Consumption in Future High-Capacity Network Nodes," Optical Communications and Networking, IEEE/OSA Journal of, vol.1, no.3, pp.245-258, August 2009.

14. CRS-3, single shelf system cisco data sheet, available at: http://www.cisco.com/\\en/US/prod/collateral/routers/ps5763/CRS-3_4-Slot_DS.html

15. T1600 Juniper networks data sheet, available at: http://www.juniper.net/\\techpubs/en_US/release-independent/junos/topics/reference/specifications/\\t1600-specifications-power-requirements.html

16. www.caida.org

17. Wilson Yong Hong Wang, Heng Ngi Yeo, Yao Long Zhu, Tow Chong Chong, Teck Yoong Chai, Luying Zhou, Jit Bitwas, Design and development of Ethernet-based storage area network protocol, Computer Communications, Volume 29, Issue 9, 31 May 2006, Pages 1271-1283.

18. Vitesse VSC3144-11 Data Sheet. Available [Online]: http://www.vitesse.com

19. Tucker, R.S.; , "Scalability and Energy Consumption of Optical and Electronic Packet Switching," Lightwave Technology, Journal of , vol.29, no.16, pp.2410-2421, Aug.15, 2011.

Appendix B

Electronic Implementation of Optical Burst Switching Techniques

Ilijc Albanese[1], Sudhakar Ganti[2] and Thomas E. Darcie[1]
[1] *Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada*
[2]*Department of Computer Science, University of Victoria, Victoria, BC, Canada*

[Published in: Proceedings SPIE 8915, Photonics North 2013 (October 11, 2013)]

***Abstract*:**

Extensive research effort is ongoing in energy-efficient Internet-based communications. Optical Flow Switching (OFS) and Optical Burst Switching (OBS) offer potentially efficient alternatives to IP-router-based networks for large data transactions, but significant challenges remain. OFS requires each user to install expensive core network technology, limiting application to highly specialized nodes. OBS can achieve higher scalability but burst assembly/disassembly procedures reduce power efficiency. Finally both OFS and OBS use all-optical switching technologies for which energy efficiency and flexibility remain subject to debate.

Our study aims at combining the advantages of both OBS and OFS while avoiding their shortcomings. We consider using a two-way resource reservation protocol for periodic concatenations of large (e.g. 1 Mb) packets or Media Frames (MFs). These chains of MFs (MFCs) are semi-transparent with a periodicity referred to as the transparency degree. Each MFC is assembled and stored at an end-user machine

during the resource reservation procedure and is then switched and buffered electronically along its path. The periodic configuration of each MFC enables interleaving of several chains using buffering only to align the MFs in each MFC in time, largely reducing the buffer requirements with respect to OBS. This periodicity also enables a simple scheduling algorithm to schedule large transactions with minimal control plane processing, achieving link utilization approaching 99.9%. In summary, results indicate that implementing optical burst switching techniques in the electronic domain is a compelling path forward to high-throughput power-efficient networking.

I. INTRODUCTION

Current network architecture is facing a massive increase in the volume of traffic as well as a change in the kind of traffic routed through the network [1]. The pace of this growth, roughly $16-20\%$ per year [2], as well as the characteristics of the traffic, are putting under pressure the current network infrastructure which has to cope with much larger traffic volumes and satisfy new requirements in terms of bandwidth guarantees, delay, availability and Quality of Service (QoS). Much of this growth in bandwidth demand is driven by the rapid proliferation of bandwidth intensive applications such as IPTV, Video On Demand (VoD), Content Delivery Networks (CDN), peer-to-peer file transfer and real-time services, which often involve large transactions. In addition to this, the power consumption of the current network infrastructure is a serious concern and will ultimately undermine the future expansion of the Internet itself if nothing is done to improve its energy efficiency [3,4].

Our focus is the power consumed in electronic routers [5]. Several studies have explored the power consumption of IP routers and showed that a key contributor to the overall power consumption of IP routers is that related to packet header processing which alone accounts for over 32% of the power consumed by a standard IP router [6]. Assuming that the average packet size in current Internet is about $10Kb$ [7] this means that large files ($> 1GB$) would require each IP router to process a large number of packet headers anytime one of such files is routed through the network. It is our belief that more efficient framing structures and protocols should be used to handle such transactions, leading to a dramatic reduction in the power consumed by the network infrastructure and securing the continued growth of bandwidth intensive applications.

Header processing is more efficient if larger packets are used, as a result the maximum IP packet size was extended from 1500 bytes (Jumbo Frames) to $64KB$ (Super Jumbo Frames). However, these specifications are not widely used mainly due to

backwards compatibility with legacy equipment [8] still massively deployed in the current infrastructure. Furthermore, besides compatibility issues, integration of packets whose size may differ by orders of magnitude over the same links is not straightforward. A study of the possibility to integrate, over the same links, variously sized packets (from standard IP packets up to $19.5KB$ per packet) was conducted in [9] showing that although using a combination of standard and extra-large frames will enable a reduction of the header related power consumption and of the computational load placed on the network, it will also lead to unfairness between flows using different packet sizes, higher drop rates for both types of packets as well as inefficient use of the bandwidth.

This result about the coexistence of different packet sizes over the same links raise the question of the potential advantage of a separate overlay network, where only large containers are used, and over which large file transfers can be efficiently handled, offloading from the IP network cumbersome transactions it is not optimized to handle.

Many research efforts are being undertaken worldwide to find a suitable evolutionary path for the current network architecture. Some consider providing to the end-user up to an entire wavelength through which large files can be transferred without segmenting them into small IP packets. Approaches like Optical Burst Switching (OBS)[10,11], Optical Flow Switching (OFS)[12] or User Controlled Light Path (e.g. UCLP[13], CAnet4[14,15], CORONET[16] and GRIPhoN[17]) have been explored extensively.

Most of these approaches advocate for a major redesign of the infrastructure in favor of an all-optical transport network where data are transparently transported and processed entirely in the optical domain. Due to the fact that the power efficiency of optical components is still questionable [18] and that such approaches make extensive use of optical buffers, which have not yet offered a commercially viable alternative to their electronic counterparts [18], these optical approaches have not captured the attention of the industry. Furthermore, the capital investment (CAPEX) required to deploy such technologies on a large scale is high and large investments have already been made to deploy the current infrastructure. This makes the economics of an all-optical redesign of the current infrastructure highly challenging.

In this paper we focus on two of such technologies proposed for the future network architecture: OFS and OBS. OFS offers access to backbone capacity to end-users for short time durations; optical flows are scheduled over a dedicated end-to-end path,

which is reserved for the entire duration of the transaction (path setup included). Optical flows are then routed all-optically from end-to-end without core buffering. This approach has two main drawbacks: one is the necessity for the end-users to install long-haul transmission equipment, limiting the access to OFS services to few specialized nodes. The other drawback has to do with the integration of OFS services within the current infrastructure. Each OFS flow is a continuous flow of data transmitted from end-to-end without core buffering, this presents various problems. First the current access rate is completely incompatible with the OFS rate and second, in order to allow end-to-end delivery of OFS flows, the present infrastructure (including access and metro networks) has to be completely redesigned [12] requiring massive investments that most ISP are not ready to make. The absence of commercial OFS systems is a clear sign of this unwillingness by the ISPs.

In OBS data bursts are assembled at the edge of the network and then routed through the network after a Control Packet (CP) is sent to reserve resources for each burst. Tell-and-Go OBS (TaG-OBS), which limits the amount of control information to be processed in the OBS control plane, is limited by its random access nature leading to high burst drop rates and bandwidth inefficiencies unacceptable for long haul transmission. The throughput of TaG-OBS is in fact limited to $0.5/e$ (the same throughput performance of pure ALOHA). Two-way reservation can be used for OBS but this would lead to control plane congestion if bursts are too small or load is too high. Another issue is the amount of buffering needed to handle OBS bursts, which due to the bursts size is either too high to be practical or, if kept small will result in high burst drop probability and low link utilization.

An interesting and more incremental approach is Electronic Burst Switching (EBS) [19]. Along the lines of OBS, in EBS data bursts are assembled at the network edge and electronically processed and switched at each node. In [19] it was concluded that using large bursts ($> 1Mb$) may lead to reduction in header-related power consumption in core switches, but the power consumed by burst assembly at the network edge negates much of the advantage gained in the core.

The aim of our approach is to combine the advantages of OFS and OBS/EBS while avoiding their shortcomings, providing an attractive evolutionary path able to improve energy efficiency relative to the current infrastructure. In our previous work [20] we demonstrated that the use of periodic concatenations of data bursts or Media Frames (MF) into periodic semi-transparent data Media Frame Chains (MFCs) can achieve equal or better performance with respect to

standard IP protocols in terms of delay, throughput and buffer size. Simulations showed that this approach reduces the amount of header-related power consumption to negligible levels and consistently outperforms IP in high-load conditions in terms of delay and bandwidth utilization. An order-of-magnitude estimate of a MFC router was also presented in [20] showing a potential energy savings of roughly 80% with respect to standard IP routers operating in the same range.

We define Transparency Degree (TD) as the number of interstices plus 1 between two consecutive MFs in an MFC. The introduction of transparency makes each MFC easier to push through the access network. The TD of each MFC can adapt to the link rate increasing or decreasing to accommodate rate shifts from one segment of the network to another. It also allows servicing multiple users by interleaving the respective MFCs with one another. Each MFC can transport enough data to justify using a two-way reservation protocol on a per-flow basis. Our scheduling algorithm takes advantage of the highly predictable structure of each MFC to implement a lightweight scheduling protocol which uses buffering only to align in time MFCs competing for the same resources, significantly reducing the amount of buffer needed with respect to standard OBS/EBS.

The paper is organized as follows. Section II provides an overview of the proposed approach and puts it into the context of a modern communication infrastructure. In section III a simple simulation study is presented comparing Tell-and-Go OBS (TaG-OBS) with the reservation-based transfer of MFCs, and performance boundaries of OFS are inferred. Section IV provides a discussion of the simulation results as well as an overview of the potential benefits of the proposed approach with respect to OFS and OBS-based techniques. Section 5 concludes the paper and discusses future studies.

II. MEDIA FRAME NETWORKING

II-A Reference Network

The figure on the left side of Figure 1 (i.e. figure B.1 (A)) shows a hierarchical network representative of a the current state-of-the-art comprising various sizes of routers (access, metro and core) connected to a transport network through various sizes of add-drop multiplexers. We assume this to be our reference architecture for the purpose of this paper.

An overlay network designed specifically for large file transactions might look like the picture of Fig. 1-B). Each of the "overlay" box parallels a present-day IP counterpart and supports the origination or termination of data flows (overlay client inter-

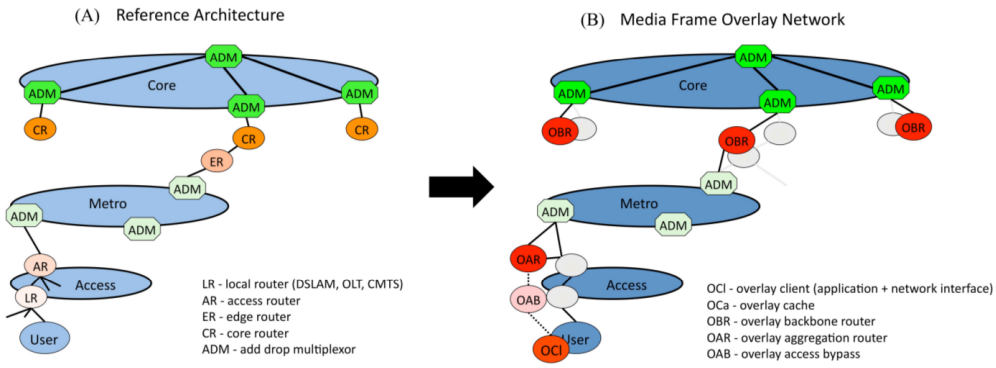


Figure B.1: Reference Architecture (A) and Media Frame Overlay Network (B).

face (OCI)), access bypass (overlay access bypass (OAB)), admission (overlay access router (OAR)), and efficient end-to-end routing (overlay backbone router (OBR)). This simple conceptual architecture may theoretically apply to any overlay network including OFS and OBS/EBS although each particular approach would deploy its own hardware and may use various topologies. Note that figure 1 does not limit the discussion to any specific topology. Topology-dependent aspects of the various approaches considered are beyond the scope of this paper.

An obvious challenge to any new overlay network is traversing access, where the large embedded base consisting of a diverse array of existing systems is in place and costly to replace. Two alternative solutions are represented in the Overlay Architecture presented in Fig. 1-B. A conservative approach may involve an application operable between the OCI and the OAR such that using traditional broadband access alone, data flows (from OBS/EBS, OFS or MFCs) are assembled or disassembled at the OAR. Another approach involves engaging the evolution of optical access standards, where commercially available standards for 10 Gb/s PON enable higher dedicated bandwidths perhaps through wavelength overlays. In this case data flows may be assembled directly at the user end point.

II-B Media Frame Chains And Transparency

Concatenation creates single entities that would contain an entire transaction (e.g. a content update onto a CDN server or a multi-GB movie download, just to name a few). Unlike the case of single data bursts (OBS/EBS) scheduling and admission control is both manageable and worthwhile for large transport entities even if done on a per-flow basis.

When large transactions are serviced continuously in time (this is the case of OFS data flows), channel resources are fully utilized for the service time. For very low network load a continuous stream of data will exhibit the lowest possible latency. However, when the load starts to increase continuous data flows may result in a significant difference in the latency experienced by those flows that are serviced immediately and those waiting in queues to be serviced. Transparency helps here in allowing multiple flows to be serviced simultaneously. Another issue of servicing data transactions continuously in time is the fact that such continuous flows would be completely incompatible with the data rates available in most access networks. Making the flows semi-transparent will enable start servicing several users (i.e. TD users) roughly simultaneously and provides data rates that are compatible with those of the end-users access network.

Latency and fairness are not the only advantages of periodic, semi-transparent data structures. Another advantage of transparency and the TD parameter is that it also implicitly conveys the information about how many MFCs can be serviced in a certain time interval, avoiding unnecessary control packet (CP) processing or queuing (CPs are in fact either serviced immediately or rejected) enabling a quick start of the resource release procedure and avoiding unnecessary occupation of precious network resources. In contrast, in the OFS case, CPs are queued until resources become available (when the queue is full the incoming calls are blocked) leaving previously booked resources booked for longer times without being actually used, reducing the overall bandwidth efficiency. Using a fixed TD also simplifies scheduling of large amount of data with minimal computational load and allows the use of buffering as a means to affect the relative timing of an MFC with respect another in order to interleave MFCs competing for the same link, without buffering entire chains. This reduces dramatically the buffer size needed with respect to OBS/EBS and allows the use of a lightweight scheduling protocol [20] able to achieve, over a packet-switched network, bandwidth efficiencies close to that of TDM systems. From the point of view of file transfer applications, transparency will increase the delay if the network load is low and a high TD is used. To reduce the TD-dependent delay the TD can be changed according to the amount of bandwidth available. When it comes to streaming applications, delay for the entire transaction is less important. Streaming quality requires only that end users receive regular delivery of data frames, and this is accommodated naturally using the appropriate TD.

III. PERFORMANCE STUDY

A simple simulation study was done using a dumbbell topology (Figure B.2) and a network simulator for MFC and TaG-OBS built using Omnet++[21]. This topology offers good insight into the performance of any network operating with performance limited by one congested link.

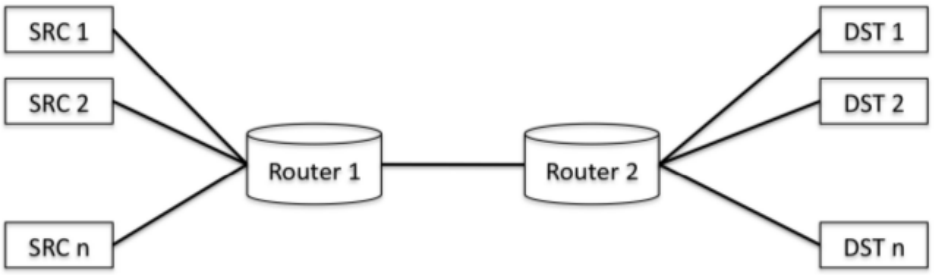


Figure B.2: Topology used for simulation.

The aim of the simulation is to directly compare MFC with OBS/EBS and to infer some performance bounds relative to OFS. An extensive comparison between MFC and OFS is under investigation and will be included in our future work. For the MFC case each MF is assumed to contain $1Mb$ of data and each MFC is defined as the concatenation of 8000MF (i.e. 1GB of data). A fixed transparency degree of 8 was chosen for all MFC and the maximum buffer size allowed for each node was fixed to $1.5MB$, the same value was used for the TaG-OBS buffer. Each OBS/EBS data burst was assumed to bare 1Mb of payload (i.e. equivalent to 1 MF). The data assembly delay is ignored for all approaches. For the comparison with OFS we rely on the following simplifying hypotheses (hp).

1. Both a MFC and an OFS flow carry the same payload (e.g. 1GB).

2. An MFC with $TD = 1$ (i.e. no interstices between two consecutive MFs) is logically equivalent to an OFS flow.

3. OFS flows are not buffered, and reservation requests (CP) are queued until the necessary resources become available (i.e. basic OFS service [12]) or the queue is full. In the latter case incoming calls will be blocked.

4. Each source offers the same average load to the network regardless of the type of source (OFS, EBS/OBS or MFC).

5. Path setup and release times for MFC and OFS are the same.

6. Given n sources, the average delay experienced by the OFS end-users to complete each transaction is roughly TD times smaller than that experienced by MFC end-users for low load conditions, but tends linearly to the same average value experienced by MFC users as the average offered load increases.

7. When channel capacity is reached both MFC and OFS experience the same average delay.

III-A Link Utilization

As shown in Figure B.3, since all sources are offering the same load to the network (hp 4), link utilization is virtually identical for all approaches up until $\sim 74\%$ load. Beyond this point burst dropping starts to degrade the performances of TaG-OBS, which remains consistently lower than that of MFC. As for the OFS case, link utilization is always higher than that of TaG-OBS, as a result of the scheduled nature of OFS, and always lower than that of the MFC case due to the less efficient reservation mechanism of OFS with respect to MFC (further details about bandwidth efficiency of MFC and OFS are discussed in section IV-A)

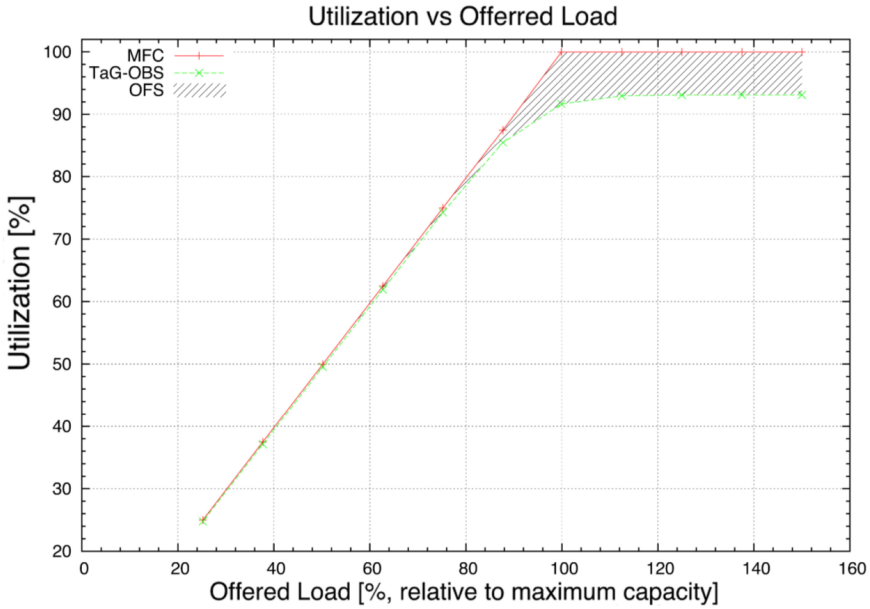


Figure B.3: Link Utilization Vs Offered Load.

III-B Burst Dropping and Call Blocking

The random access of TaG-OBS results in the worst performance in terms of data being dropped, as shown in Figure B.4. Both OFS and MFC are reservation-based

approaches and perform consistently better than OBS. However, due the less efficient reservation mechanism of OFS more resources will be occupied for each transaction with respect to MFC (due to hp 1 both OFS flows and MFC bare the same payload per each transaction) resulting in higher blocking probability with respect to MFC (see also section IV-A). Note that for the purpose of this paper only a single channel was considered in the dumbbell topology used for the simulations. In such a scenario, where no Routing and Wavelength Assignment (RWA) or load balancing algorithm was considered using multiple wavelengths per link would not have added further insights to the study.

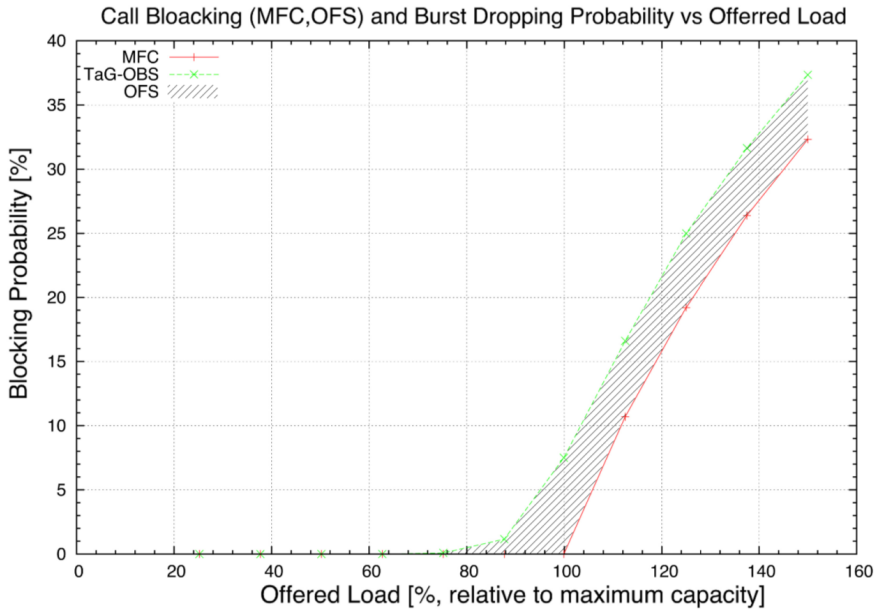


Figure B.4: Link Utilization Vs Offered Load.

III-C Delay

Delay performance of the OFS approach is better than both MFC and TaG-OBS up to almost 100% of the offered load, as seen in Figure B.5. For this preliminary study we assume (hp 6) that delay for OFS increases linearly with the offered load up to the point in which the full capacity is reached, where OFS performances are roughly the same of the MFC approach. This is due to the fact that, assuming the same offered load (hp 4), as the load increases OFS reservations will be queued up and will have to wait until resource become available on the output link. The resulting effect is similar to that of the higher buffering time experienced by MFCs as the load increases (see figure B.6). We are aware that this is just a very rough estimate of the

delay performances of OFS and that many other factors should be considered (e.g. routing algorithm, load balancing, resource reservation protocol, etc.), a more precise comparison of the delay performances of MFC and OFS will be part of our future work.

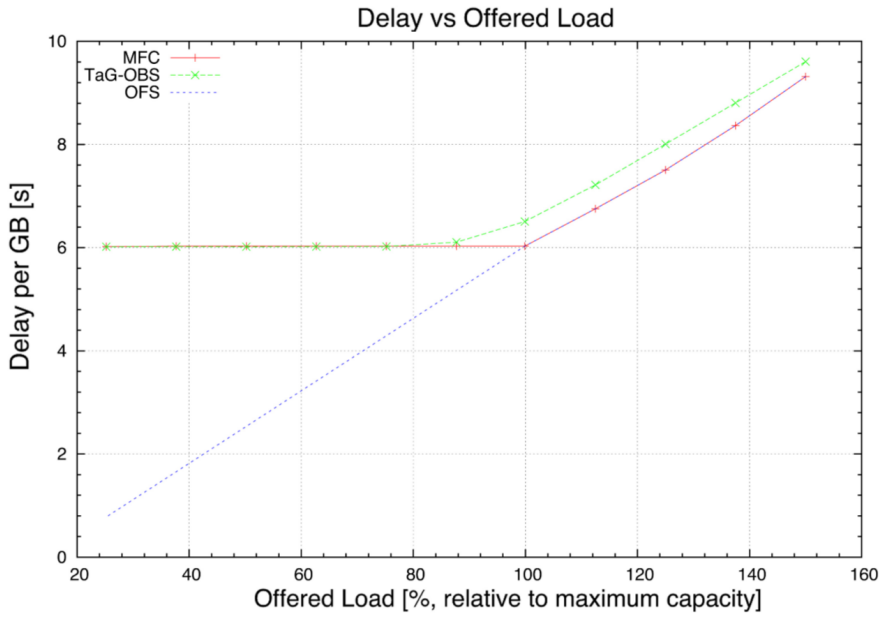


Figure B.5: Delay (per GB of data successfully delivered) Vs Offered Load.

III-D Buffer Size

As shown in Figure B.6, the average buffer space occupied by TaG-OBS is always higher than that occupied in the MFC case. This is due to the efficient buffering technique used for MFC, which uses buffering only to align in time various MFCs. As shown by the simulation result (Figure B.6) this will allow for a large reduction of the amount of buffer space occupied by MFCs (2 to 5 times less than the OBS case) as well as making it tightly controllable. The maximum buffer size occupied with the MFC approach is bounded to $TD*(size\ of\ one\ MF)$ bytes for each channel and will never exceed this quantity. OFS requires no core buffering so we consider its buffer occupancy as 0 in all load conditions.

IV. DISCUSSION AND ANTICIPATED BENEFITS

Simulation results provide many insights about how MFC performs in comparison with OFS and OBS approaches. An MFC is a transport entity that can be thought of as a hybrid between an optical flow (OFS) and an optical burst (OBS). A MFC with a TD of 1 closely resembles an optical flow, while an MFC with length equal to

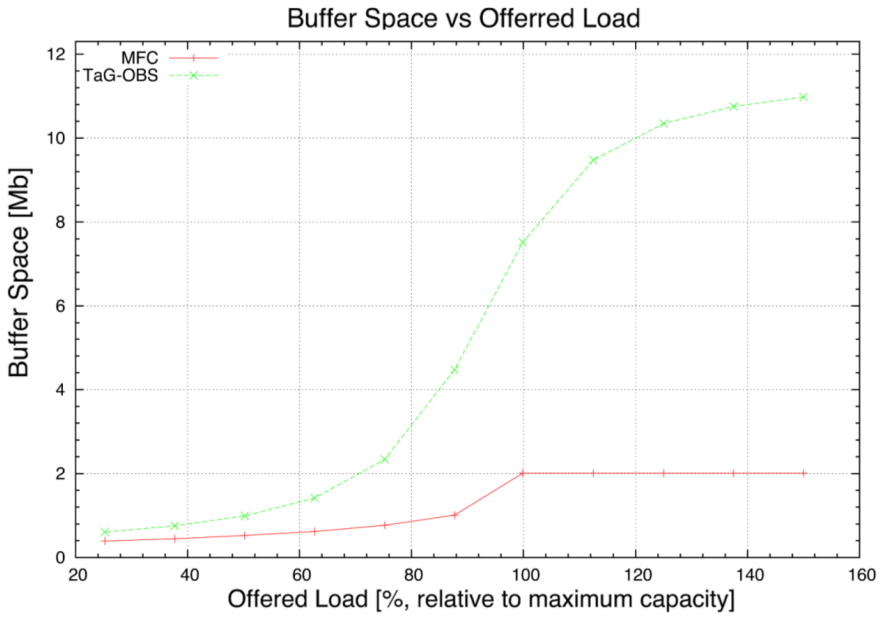


Figure B.6: Buffer Occupancy Vs Offered Load.

1 MF would be logically equivalent to an optical burst the differences being in the scheduling protocol, in the hardware used to handle the data and in data aggregation procedures. Most importantly, each MFC makes use of the electronic buffering and processing capabilities instead of relying on a fully transparent optical network. The scheduling protocols used for all OFS services (i.e. basic per-flow services and ultrafast services) reserve the necessary resources for each flow immediately after accepting each reservation request and such resources are kept booked until the flow actually uses them [12]. This increases the delay in the system by keeping previously booked resources occupied for longer time spans without using them. Furthermore, in OFS the queuing delay for the reservation requests starts increasing much before the 99.9% wavelength utilization is reached [12], while in the MFC approach any eventual queuing delay for the control packets will only start when 99.9% utilization has been reached. For ultrafast OFS services multiple requests are sent over multiple paths and resources are booked in all such paths before the destination node chooses a path and release the resources on discarded paths. Although the path setup and reservation may be faster in this case than when probing only one path at a time, the problem of the sub-optimal timing for the resource booking is worsened with respect to the case of basic OFS services and the overall blocking probability will increase. In our approach [20], the scheduling algorithm books resources using precise timing

information about the expected arrival time and configuration of each MFC. As a result resources are only booked for the time necessary to handle the transaction optimizing resource usage, lowering call blocking probability and reaching very high link utilization (99.9%) before starting to block call reservations. This results in superior performance of MFC with respect to OFS in terms of link utilization and call blocking probability, as discussed in detail in section IV-A. Another advantage of MFC with respect to OFS is that, according to the model presented in [22], MFC networks would have a larger *capacity region* with respect to both OFS and OBS due to the combined use of core buffering, scheduling and wavelength-continuity [12,22]. The OBS/EBS approach, besides having the smallest capacity region of all [22], also suffers from inefficiencies. These include limitations from random access (e.g. TaG-OBS) or control plane congestion [23], when more complex reservation mechanisms are used. Another issue with OBS/EBS is the tradeoff between the amount of buffering needed to guarantee high bandwidth utilization and the high burst blocking probability in the case of bufferless OBS or if buffers are too small. Using MFC and the particular buffering technique used for MFCs it is possible to transfer the same amount of data using concatenations of smaller bursts (or MF) hence reducing the amount of buffer necessary as well as making it tightly controllable. Hence we retain roughly the same performance in terms of delay, blocking probability and link utilization. Figure B.7 show the amount of buffer space occupied when the size of each MF is changed. The amount of buffer is limited to $(TD - 1) * (size\ of\ one\ MF)$. It is then possible to limit maximum buffer size by tuning these two parameters.

IV-A Media Frame Chains and OFS Flows

In this section we will discuss the implications the hypotheses made in Section III. Our goal is to show that regardless of the outcome of a reservation attempt, link utilization of MFC will always be higher than that of OFS and, as a consequence the call blocking probability for MFC will always be lower.

Lets first consider the case of a successful reservation. In this case it is trivial to show that, given both a MFC and an OFS flow bare the same payload (hypothesis 1 of section III) the channel holding time for OFS flows is higher than that of MFCs. This is due to the fact that resources for OFS flows are reserved immediately after the CP is processed as shown in figure B.8 (below).

The grey area in figure B.8 (A) is added to the channel holding time for OFS as a result of the OFS scheduling algorithm [12] hence, if both OFS flows and MFCs carry the same payload, OFS channel holding time will necessarily be longer than MFC for

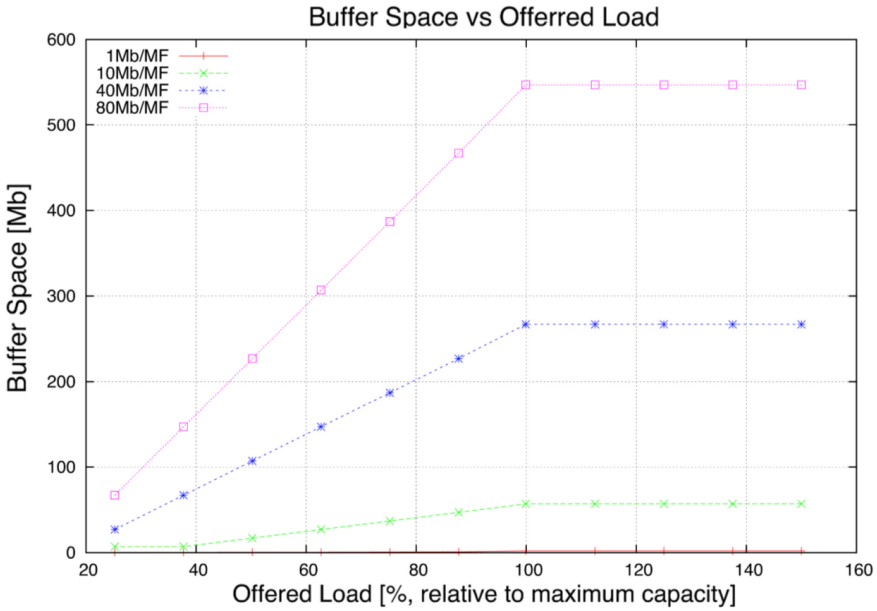


Figure B.7: Buffer Occupancy Vs Offered Load for Various MF sizes.

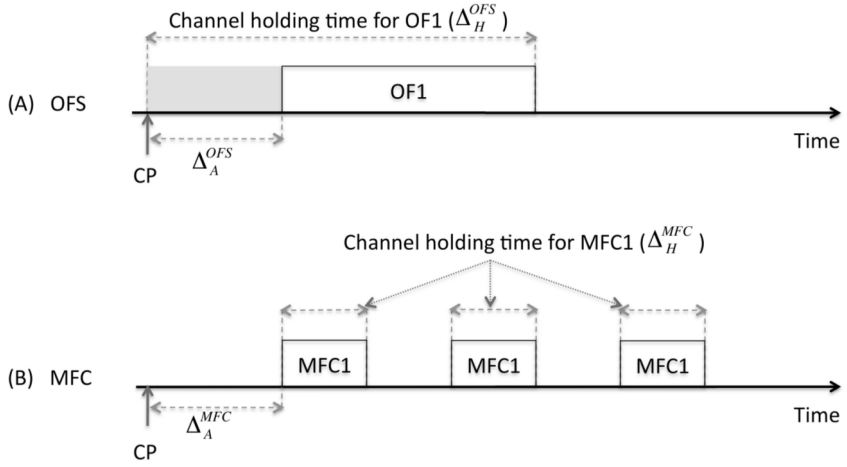


Figure B.8: Channel Holding Times for MFC and OFS for successful reservation.

the successful reservation case (i.e. ${\Delta_H}^{OFS} > {\Delta_H}^{MFC}$). For the failed reservation case we refer to Figure 10. The shaded areas represent the portion of channel being held for each data flow (OFS or MFC) before a $NACK$ comes to free the resources.

The time between arrival of the CP and arrival of the resource release message is indicated with ${\Delta_N}^{OFS/MFC}$ for the OFS case (Figure B.9-(A)) the total channel holding time is:

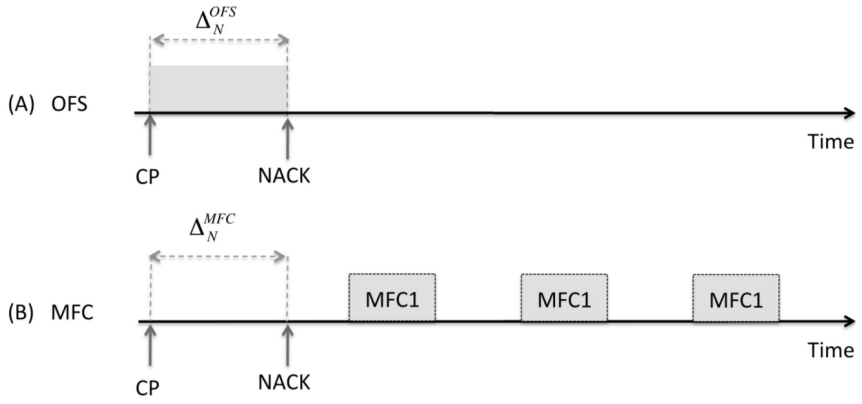


Figure B.9: Channel Holding Times for MFC and OFS for failed reservation.

$$\Delta_H{}^{OFS} = \frac{B_{OFS}}{B} * \Delta_N{}^{OFS} \tag{B.1}$$

where B is the channel bitrate and B_{OFS} is the bandwidth occupied by the OFS flow. For the MFC case the total channel holding time is:

$$\Delta_H{}^{MFC} = \frac{B_{MFC}}{B} * \Delta_N{}^{MFC} \tag{B.2}$$

Since OFS reserves the entire channel bandwidth [12] it is always true that:

$$\frac{B_{OFS}}{B} = 1 \tag{B.3}$$

while for the MFC case, due to the transparency property of the MFCs, we always have that:

$$\frac{B_{MFC}}{B} < 1 \tag{B.4}$$

For the $\Delta_N{}^{OFS/MFC}$ parameter, hypothesis 5 guarantees that $\Delta_N{}^{OFS} \equiv \Delta_N{}^{MFC}$. Therefore, given Equations B.3 and B.4, failed reservation case, the following will always be true (inequality B.5):

$$\Delta_H{}^{OFS} > \Delta_H{}^{MFC} \tag{B.5}$$

It is therefore shown that channel holding time for OFS is always larger than that of the MFC case under hypotheses 1 and 5 of section III. Consequently link utilization

for OFS will necessarily be lower and call blocking probability due to lack of available resources will always be higher for OFS.

Lastly, note that both the time interval between the start of the reservation and the arrival of the OFS flow (i.e. ${\Delta_A}^{OFS}$ in Figure B.8-(A)) and ${\Delta_N}^{OFS}$ depend on the path length. The same is true for MFC (${\Delta_A}^{MFC}$ and ${\Delta_N}^{MFC}$) with the difference that only a fixed amount of resources (i.e. B_{MFC}) are booked in the MFC case during these time intervals. This may result in further degradation of the link utilization and blocking probability for OFS as longer data paths are chosen as the amount of reserved resources will grow with both ${\Delta_A}^{OFS}$ or ${\Delta_N}^{OFS}$ (i.e. successful or failed reservation, respectively). This is not true for the MFC case where only a fixed amount of resources is reserved for each transaction and they are either kept for ${\Delta_N}^{MFC}$ if reservation fails or fully used without any waste if the reservation is successful.

IV-B Impact on Router Power Consumption and Example MFC Router

Extensive literature exists about power consumption of IP routers and the contributing functional processes [3,5,6,24]. Among these studies there is general agreement that a major contribution to the overall power consumption of IP routers is due to header related functions which in the current infrastructure are to be carried out for each packet traversing a node. Organizing large transactions into MFC will enable handling each transaction (e.g. video streaming, CDN content update and distribution, etc.) by processing only one packet per transaction. This will dramatically reduce header-related power consumption as well as the amount of computational load for each router. Besides the periodic structure of the MFCs, the use of large MF to assemble a MFC brings its own set of advantages: storing large files at the source (e.g. a CDN server) in large memory blocks will allow a significant reduction in CPU utilization by reducing the frequency of memory accesses and IRQs that the CPU has to handle [25]. This feature will enable further power savings in data storage servers. Another advantage of using large frames is represented by the tradeoff between the requirements on the switch reconfiguration speed and the buffer size of the same device. Faster reconfiguration speeds are required for switch fabrics handling smaller packets or cells, but this requirements can be relaxed at the cost of an increase in size of the buffer used to synchronize the MFCs in order to support MFCs using larger MFs (see figure 8). A simple schematic of a MFC router (MFR) was presented and analyzed in [20] and is reported here for clarity (Figure B.10).

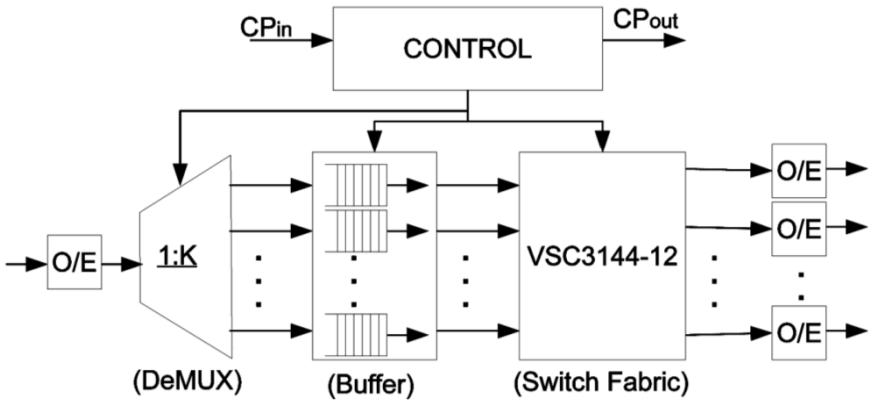


Figure B.10: Concatenated Media Frame Router (MFR).

This device is able to switch data in the Terabits/s range and exhibits a power consumption in the order of $760W$ which is roughly 20% that of a standard IP router operating in a similar range [26,27]. Input data is converted into the electrical domain and data streams from each channel are de-multiplexed into their constituent MFCs. Each MFC is then delayed by the amount indicated in its associated CP. The number of buffer queues needed depends on the number of chains the device is able to handle and is given by # of dedicated buffer queues = # of input channels * # of MFCs per channel. The total buffer size also depends on the number of simultaneous flows the device must handle and the transparency degrees of the chains. At the output of the buffer stage, chains competing for the same output channels are synchronized in order to allow interleaving. MFCs are then passed to the switch fabric, which simply routes each MFC to the appropriate output with no further buffering or processing. This particular architecture, which can be implemented with off-the-shelf components, implements all the functionalities necessary to support MF-based transport (including MFC and unscheduled data bursts). It does not need any internal scheduling or flow control protocol to interface with the switch fabric: once the MFCs are demultiplexed and pass through the buffer stage they are perfectly synchronized with one another and chains directed towards the same output port will interleave naturally without any possibility of collision or contention within the switch fabric. This additional feature will further reduce the complexity and the computational load of the device.

V. CONCLUSIONS AND FUTURE WORK

In this work we compare three different future networking approaches, Optical Burst Switching, Optical Flow Switching, and Media Frame Concatenation, and study their performances in terms of bandwidth efficiency, latency, energy efficiency

and practical feasibility using both computer simulations and simple theoretical and practical considerations. Link utilization achieved by MFC is higher than both OBS and OFS, which are limited mainly by the resource reservation protocol used (OFS) and by random access (TaG-OBS). OFS performs better that both MFC and OBS in terms of latency for the majority of the range of operation of the test network considered, but at the cost of fairness with respect to the MFC approach. Due to lower bandwidth efficiency OFS with respect to MFC, OFS call blocking probability will be higher with respect to that of MFC. A detailed comparison of the delay, utilization and call blocking performance of OFS and MFC, including topology-dependent aspects, will be part of our future studies. Compatibility of OFS and OBS with the current network infrastructure is a critical issue, with key challenges in achieving optical transparency across the network hierarchy (access metro core) and realizing the required optical buffers. The lack of widely deployed commercial systems based on such technologies is evidence of these challenges. We conclude that while OFS and OBS may be attractive technologies for future all-optical networking, their practical feasibility still faces significant challenges. MFC offers similar performance, but with a far more incremental approach.MFC can be implemented with off-the-shelf components and is potentially able to integrate with the current network transport infrastructure with only minor modifications while offering considerable energy savings with respect to currently deployed IP routers (80% less power consumption for MFC routers).

References:

1. Cisco Systems Inc., Entering the Zettabyte Era, CISCO, 30 May2012, available at: `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/\\ns705/ns827/VNI_Hyperconnectivity_WP.html`

2. G.Fettweis, E. Zimmermann, ICT Energy Consumption Trends and Challenges, 11 International Symposium on Wireless Personal Multimedia Communications (WPMC 2008)

3. O. Tamm, C. Hermsmeyer, and A.M. Rush, Eco-Sustainable System and Network Architectures for Future Transport Networks, Bell Labs Technical Journal 14(4), 311328 (2010).

4. Eilenberger, G. J., Bunse, S., Dembeck, L., Gebhard, U., Ilchmann, F., Lautenschlaeger, W. and Milbrandt, J. (2010), Energy-efficient transport for the future internet. Bell Labs Tech. J., 15: 147167. doi: 10.1002/bltj.20446

5. Tucker, RodneyS., "Green Optical CommunicationsPart II: Energy Limitations in Networks," Selected Topics in Quantum Electronics, IEEE Journal of , vol.17, no.2, pp.261,274, March-April 2011doi: 10.1109/JSTQE.2010.205121

6. Tucker, Rodney S.; Parthiban, R.; Baliga, J.; Hinton, K.; Ayre, R.W.A.; Sorin, W.V., "Evolution of WDM Optical IP Networks: A Cost and Energy Perspective," Lightwave Technology, Journal of , vol.27, no.3, pp.243,252, Feb.1, 2009 doi: 10.1109/JLT.2008.2005424

7. `www.caida.org`

8. Alteon Networks, whitepaper .(2012). Extended Frame Sizes for Next Generation Ethernets. Retrieved April 14, 2013, available at: `http://staff.psc.edu/\\mathis/MTU/AlteonExtendedFrames_W0601.pdf`

9. Divakaran, D.M.; Altman, E.; Post, G.; Noirie, L.; Primet, From Packets to XLFrames: Sand and Rocks for Transfer of Mice and Elephants, in IEEE INFOCOM Workshops 2009.

10. J. P. Jue,V. M. Vokkarane, Optical Burst Switched Networks, Ed. Springer 2005.

11. Yong Liu, Kee Chaing Chua, Gurusamy Mohan, Achieving High Performance Burst Transmission for Bursty Traffic using Optical Burst Chain Switching in WDM Networks, IEEE Transactions on Communications, Vol.58, Issue 7 pp. 2127-2136, 2010.

12. Chan, V. W S, "Optical Flow Switching Networks," Proceedings of the IEEE , vol.100, no.5, pp.1079,1091, May 2012 doi: 10.1109/JPROC.2012.2183629.

13. Bill St. Arnaud, UCLP Roadmap for creating User Controlled and Architected Networks using Service Oriented Architecture, CANARIE., January 2006.

14. Yunfeng Peng; Kunjun Jiang, "On user-participated and service-oriented optical networks," Optical Communications and Networks (ICOCN 2010), 9th International Conference on , vol., no., pp.134,138, 24-27 Oct. 2010

15. CANARIE www.canarie.org

16. Clapp, G.; Doverspike, R.; Skoog, R.; Strand, J.; Von Lehmen, A., "Lessons learned from CORONET," Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC) , vol., no., pp.1,1, 21-25 March 2010

17. Ajay Mahimkar, Angela Chiu, Robert Doverspike, Mark D. Feuer, Peter Magill, Emmanuil Mavrogiorgis, Jorge Pastor, Sheryl L. Woodward, and Jennifer Yates. 2011. Bandwidth on demand for inter-data center communication. InProceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages

18. R. S. Tucker, The role of optics and electronics in high-capacity routers, J. Lightwave Technol., vol. 24, pp. 4655 - 4673, 2006.

19. S.Peng,K.Hinton,J.Baliga,R.S.Tucker et.al, Burst Switching for Energy Efficiency in Optical Networks, OSA/OFC/NFOEC 2010

20. I.Albanese, T.Darcie, S.Ganti, Power-efficient Electronic Burst Switching for Large File Transactions, SMARTGREENS 2013

21. OMNeT++ Discrete Event Simulation Tool, http://www.omnetpp.org

22. Weichenberg, G.; Chan, V. W S; Medard, M., "On the capacity of optical networks: A framework for comparing different transport architectures," Selected Areas in Communications, IEEE Journal on , vol.25, no.6, pp.84,101, August 2007 doi: 10.1109/JSAC- OCN.2007.027106

23. Barakat, N.; Darcie, T.E., "Control-Plane Congestion in OBS Networks," Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on , vol., no., pp.1,3, 1-5 Oct. 2006 doi: 10.1109/BROADNETS.2006.4374320

24. Aleksic, S., "Analysis of Power Consumption in Future High-Capacity Network Nodes," Optical Communications and Networking, IEEE/OSA Journal of , vol.1, no.3, pp.245,258, August 2009 doi: 10.1364/JOCN.1.000245

25. Wilson Yong Hong Wang, Heng Ngi Yeo, Yao Long Zhu, Tow Chong Chong, Teck Yoong Chai, Luying Zhou, Jit Bitwas, Design and development of Ethernet-based storage area network protocol, Computer Communications, Volume 29, Issue 9, 31 May 2006, Pages 1271-1283.

26. CRS-3, single shelf system cisco data sheet, available at: http://www.cisco.com\\/en/US/prod/collateral/routers/ps5763/CRS-3_4-Slot_DS.html

27. T1600 Juniper networks data sheet, available at: http://www.juniper.net/\\techpubs/en_US/release-independent/junos/topics/reference/specifications/t1600-specifications-power-requirements.html

Appendix C

Big file protocol (BFP): A traffic shaping approach for efficient transport of large files

Ilijc Albanese*, Yağız Onat Yazır†, Stephen W. Neville*, Sudhakar Ganti† and Thomas E. Darcie, *Fellow*, IEEE*
*Department of Electrical and Computer Engineering
†Department of Computer Science
University of Victoria, Victoria, Canada.

[Published in: Proceedings of IEEE 15th International Conference on High Performance Switching and Routing (HPSR), 2014]

***Abstract*:** Telecommunication networks are under stress due to rapid traffic increase. Since most of this traffic increase is due to large file transfers, this paper proposes a cross-layer transport protocol specifically designed to efficiently handle large transactions. Traffic generated from large transactions is shaped into a periodic succession of fixed-size data frames. Each transaction is then scheduled for transmission using a two-way reservation protocol. Simulation results show that the proposed approach is capable of significantly improving goodput and end-to-end delay relative to TCP, improving efficiency of bandwidth utilization by over 40%.

I. INTRODUCTION

Telecommunication networks are experiencing continued rapid traffic increase mostly driven by the proliferation of bandwidth-intensive applications. Unfortunately, the revenues for network operators are not growing at the same pace [1]. In order to continue supporting the growth of Internet-based applications in an economically viable manner, the industry must reduce the cost per bit transported and increase capacity.

While increasing the deployed capacity is costly, improving traffic grooming and shaping becomes important to better utilize the capacity provided [2,3]. Traffic shaping policies are often used by ISPs to limit bandwidth costs especially when it comes to bulk data transfers. Such policies often lead to significant end-to-end performance losses [3]. Furthermore, constrained by traditional protocols such as TCP and UDP, the increasing proportion of large file transactions (e.g. VoD, IPTV, CDN content update and distribution and so on [4]), places an increasing burden on the network hardware and capacity of current IP networks.

In the increasingly important Data Center (DC) environment, efficient intra-DC and inter-DC networks play a crucial role in minimizing congestion and conserving computational resources. These networks must provide predictable aggregate performance for diverse and unpredictable load conditions [5,6]. Managing this unpredictability generally translates into inefficiency in the use of transport resources such that associated transport costs become appreciable, particularly for wide-area inter-DC networks. In this context, a network that is able to provide predictable network performance for random traffic patterns would be a desirable feature [5,6].

In parallel, recent standardization of transport architectures such as ITU-T G.709 / G.872 Optical Transport Network (OTN) [7,8,9] has provided a key step towards a more flexible and efficient transport infrastructure layer, enabling protocol agnostic transport services, allowing a more efficient use of deployed capacity and simplifying management operations [10,11]. This widely deployed standard provides well-defined mappings of most existing and future higher layer formats into a common underlying transport layer.

In this work, we exploit the observation that bandwidth demand is increasingly dominated by large transactions [4] and propose a networking approach wherein large file transactions are handled separately from the large number of smaller transactions. Traffic shaping can then be used to improve link utilization and reduce the computational load placed on the network. Traffic shaping for the corresponding long hold times can benefit from a far less dynamic traffic mix. In what follows, large transactions are shaped into periodic concatenations of data frames which are scheduled for transmission over an end-to-end path using a lightweight scheduling protocol. This Big File Protocol (BFP) shapes each transaction prior to transmission and executes the scheduling procedure on a per-flow basis. BFP creates a tighter linkage (than TCP/UDP) between the application layer and resource availability at the physical layer (either Ethernet or OTN), made possible by the long push times of long file

transfers. This approach, although reminiscent of TDM systems, (1) does not require network-wide coordination, (2) can be implemented over any packet switched network, and (3) is entirely compatible with the current OTN infrastructure as well as with Ethernet-based networks, commonly used in data-center environments [6]. Similar considerations also apply to pipeline forwarding [12]. In contrast, (1) BFP is designed to handle large files only, and scheduling acts on complete transactions not on individual packets. (2) BFP traffic does not coexist with standard TCP/UDP traffic, and (3) BFP data synchronization is orchestrated locally at each node by control plane operations, without requiring router synchronization or a shared time reference. Our approach is able to overcome the limitations of traffic shaping on end-to-end performance of the network while allowing a more efficient utilization of network resources, and a significant increase in goodput ($> 40\%$), potentially reducing the cost per bit.

Section II provides an overview of BFP. Details of implementation, the underlying processes of scheduling, admission control and mapping are discussed in Section II. Simulation results are presented in Section III.

II. BIG FILE PROTOCOL

BFP is a cross-layer transport protocol designed to handle large transactions (e.g. $\geq 100MB$) in a more efficient manner by shaping traffic and leveraging currently available transport layer functionalities to enable a better use of the deployed capacity and reduce the computational load. The nodes implementing BFP shape traffic from each transaction into periodic, semi-transparent concatenations of fixed-size data frames, referred to as chains henceforth.

Periodicity of each chain is obtained by modeling access to the transport resource as a periodic succession of fixed-length timeslots in which a transaction is allowed access to an output channel at full capacity. Periodicity simplifies scheduling and stabilizes traffic variations resulting in stable network performance. The semi-transparency of each chain, resulting from periodic access to the channel, allows interleaving several flows using buffers to align the flows in time. With reference to Figure C.1, the transparency transparency degree (TD) of a chain is defined as the period (expressed in number of timeslots) of the occupied slots in a periodic frame structure that represents capacity on a transport resource. Note that TD can be dynamically adapted based on the available BW, thus requiring a cross-layer approach. Furthermore, BFP differs from TDM systems in that timeslots are only used to model transport resources and regulate access on a per-flow basis, and need not to be physically implemented in

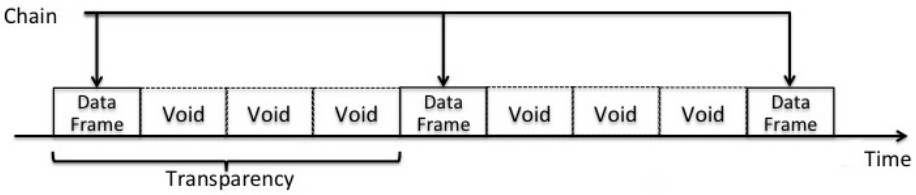


Figure C.1: Data transaction organized into a chain with $TD = 4$.

the underlying hardware. Another difference with TDM systems is that the timeslots in TDM are somehow rigid and each TDM channel occupies a fixed amount of the available bandwidth. In BFP the amount of bandwidth occupied is decided on a per-transaction basis, resulting in a more flexible bandwidth usage. Lastly, the size of the BFP timeslots can be adjusted by varying the size of the data frames. In each chain, the fixed-size data frames are comprised of an integer number of Basic Payload Frames (BPF) as shown in Figure C.2. Once a receiving node delineates a data frame consisting of one or more BPFs, it handles it as a single frame. This allows for the use of data frames of virtually any size. Basic Void Frames (BVF) are also defined. BVF are equal in size to BPFs but carry only stuffing bits and are added at transmission time to fill the gaps between data frames according to the selected TD. Using BPF and BVF effectively allows setting up a TDM-like channel on the fly, targeted to the specific requirements of each transaction, which is automatically released when the transaction is completed.

A network of fixed capacity links is presumed to have been provisioned interconnecting widely separated data centers and this capacity is accessed only by BFP. This deployed capacity may be in the form of dedicated transport capacity, whether this is Ethernet, Ethernet over OTN, or a mapping of BFP directly onto the OTN layer. Frames are selected to fit naturally within the underlying transport capacity.

BFP exploits a tight linkage between application and physical-layer resources, as per Figure C.3. Application programmers would see BFP as just another variant of well-known socket-related actions, like TCP and UDP, as discussed later. But whereas TCP and UDP pass through IP before transport, BFP passes through a parallel path directly to the transport layer. This path manages admission control, scheduling, and mapping into physical-layer resources, functions that may exploit information passed between nodes by TCP/IP.

II-A Admission Control And Scheduling

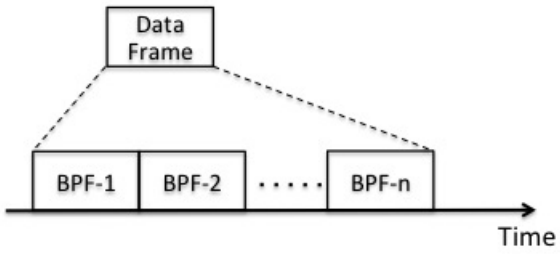


Figure C.2: Data frame assembled by multiple BPF.

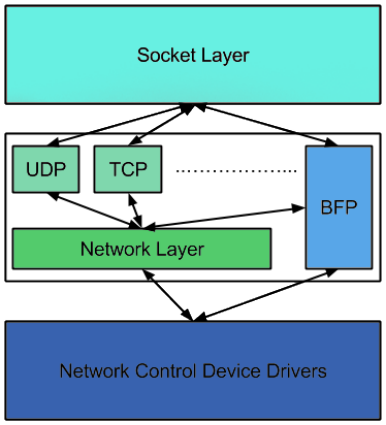


Figure C.3: Integration of the proposed protocol in the current layered architecture.

When a source is ready to transmit a chain, it is associated with a control packet (CP), which is sent over the data path whose propagation time we assume to be known by the source node to reserve resources, such as bandwidth and buffer space, using a two-way reservation protocol. Due to the periodic configuration of the data frames in a chain, a small CP can convey full information on the chain configuration and timing regardless of the amount of data transported by the chain. During the resource reservation phase the data sits at the source machine until confirmation (ACK) of successful reservation is received, avoiding unnecessary occupation of network resources. Many alternative approaches to scheduling are possible. In what follows we describe a representative approach that can be used to schedule chained data structures over an end-to-end path.

When a CP for the selected configuration (i.e. frame size, TD and number of data frames per file) is generated at the source node, an Expected Time of Arrival (ETA_{src}) parameter is computed according to the following Equation C.1 and included in the CP .

$$ETA_{src} = \sum_{i=1}^{N}(\tau_i + p_i) + \tau_{ACK} \tag{C.1}$$

where:

N number of nodes in the path

τ_i propagation time to reach node i

p_i estimated CP processing time for node i

τ_{ACK} time for the ACK to reach the source node

The ETA parameter communicates to each node in the path the delay between the reception of the CP and the arrival of the first bit of its relative chain. Each node uses ETA and chain configuration information to compute the necessary buffering time (BT) to interleave an incoming chain with previously scheduled chains on the output channel, according to Equation C.2. Before being forwarded to the next node, CP is updated with a new ETA information, computed according to Equation C.3. This updated ETA value will be used by the next node in the path to schedule the chain.

$$BT_i = t_{e,1} - ETA_i \tag{C.2}$$

$$ETA_{i+1} = ETA_i + BT_i - p_i \tag{C.3}$$

where:

$t_{e,1}$ is the ending time of the first available timeslot on the outgoing link (Figure 4.3).

Once the CP reaches the destination node, an acknowledgement (ACK) is generated and sent over the reverse path to the source node. Upon receiving the ACK, the source node is assured that resources are available to support the transaction and starts transmitting the data frames separated by a number of void frames (BVF) consistent with the selected TD. Using this procedure, each node in the path can reserve resources only for the time necessary for the chain to traverse it, optimizing resource utilization. We rely on buffering to align frames in time and to overcome any timing uncertainties. Therefore, timing precision required by the scheduling procedure should be manageable using standard ranging procedures.

II-B Mapping Onto Transport Layer

We now describe the mapping of the proposed protocol onto the transport layer after a brief review of Ethernet and OTN framing.

1) *Mapping BFP Onto Ethernet*: Standard Ethernet frames [13] bare a payload varying from a minimum of 42 bytes to 1500 bytes. This limitation is removed when jumbo frames are used, allowing up to 9000 bytes of payload. Each Ethernet frame has a preamble of 7 bytes, 1 byte of Start Frame Delimiter (SFD), 6 bytes for source MAC address and 6 for destination MAC address, a 4 bytes Frame Check Sequence is also appended to the Ethernet frame. Furthermore, after each frame is sent, transmitters are required to transmit a minimum of 12 bytes of Inter Frame Gap (IFG).

BPF and BVF can be mapped directly onto Ethernet frames by setting the size of BPF and BVF equal to the Ethernet frame payload size (e.g.: 9000 B), allowing the proposed protocol to be mapped onto the Ethernet transport layer. In this case, frame delineation is performed by the Ethernet frames and preamble or Inter Frame Gap (IFG) bits can be used to code frame-related information (e.g. frame number, frame type -BPF, BVF or CP- and data frame size). Note that in the 10GE case, only full-duplex mode is allowed, making the preamble bits unnecessary (10GE receivers ignore preamble bits) and allowing the use of these bits for other purposes, e.g. as proprietary OAM channels [9].

2) *Mapping BFP Onto OTN*: ITU-T G.709 (OTN) [7,9] provides bit and timing transparent transport services. Standard containers for any client signal available today and their relative mapping procedures are defined [7,9]. Furthermore, flexible containers (i.e. ODUflex) are also defined to support packet-based clients with a wide range of bitrates. Each OTN frame bares a payload of 15232 bytes over which one or more client signals are mapped. For cell or packet based clients OTN uses GFP [14] to encapsulate data packets and generate a continuous stream of GFP frames which is then mapped in an octet-aligned manner directly onto the OPU payload area. Rate adaptation is done using GFP idle frames, which are transmitted anytime there is no data to send.

For implementation of BFP over OTN, the general idea is to reserve an ODU channel over each link of a network to build an overlay network which is then used exclusively to handle BFP transactions. In certain scenarios it maybe important that the portion of dedicated bandwidth is flexible. In order to achieve this flexibility without using cumbersome control plane operations, setting up ODUflex channels for each link of the overlay network seems the best option as these channels can be resized

without tearing down and re-establishing the connection via the Hitless Adjustment of ODUflex protocol [15]. The configuration of the overlay network (topology and capacities of each link) should be relatively stable (time-wise) and only occasional changes will be needed. Such changes can be planned offline over coarse timescales.

Once the overlay has been established, each chain is built by mapping BPF and BVF onto GFP frames, and GFP Extension Header [14] is used to code frame-related information. The resulting stream of GFP frames is then mapped onto the payload area of the ODUflex (i.e. ODUflex(GFP)) signal. Frame delineation is left to GFP and any eventual rate adaptation is performed using GFP idle frames [7,9,14].

II-C Routing

Since the proposed approach provides a methodology separate from IP for end-to-end data transmission, a way to route chains from source to destination is required. Although routing and load balancing algorithms can be specifically designed, taking into account the periodic configuration of BFP chains, it is also possible to simply re-use protocols that are already in place. In the following a method to reuse deployed routing protocols is discussed.

As the CP travels through each node an association between input port and output port traversed by the CP is built and stored locally in a table at each node. The assumption here is that both CP and data chain will go through the same ports in both the forward and backward direction. This port mapping is based on the particular routing protocol implemented at each specific node and the routing information is only accessed when the CP is processed avoiding any header lookup operations on the data frames of each chain. Data frames of each chain are easily identified using the TD information (periodicity) combined with the ETA parameter carried by the relative CP, and are simply delayed by an amount of time equal to the Buffering Time (which is the same for each frame of the same chain) before being switched to the output port indicated by the aforementioned table without further processing. In this respect the periodicity of each chain can be thought as an "embedded Forwarding Equivalence Class" (e-FEC) relative to all the frames in a chain. Each table entry is relative to a specific chain and is stored in the table until the chain traverses the node. The size of this table depends on both on the number of transactions each node is able to handle and on the average duration of each reservation procedure. Although the size of the table may become cumbersome the assumption here is that large transactions are relatively long lived and the reservation procedure is fast enough to keep the size manageable.

II-D Application Programmer Interface

Usability and deployability of the proposed protocol requires certain important points to be taken into account. First, it is necessary to install the protocol in the network stack in a way that ensures a certain level of familiarity for the application developers. That is, an application developer, should be able to undertake the well-known and common socket related actions such as creating, connecting, listening, sending, receiving, etc. in a way that is not much different from the actions that are involved in programming with TCP or UDP sockets. This requires that the proposed protocol is registered with the OS socket layer, can be easily accessed by just using an indicator that will point to the newly registered protocol upon socket creation, and the necessary functionality is provided with identical function headers. In the same sense, the structural similarities will also have a positive impact on incorporating the proposed protocol with software packages that abstract the socket layer.

Fortunately, the necessary installation options already exist in the Linux Kernel. The proposed BFP can be registered with the socket layer as a new transport layer protocol making it appear to the application developers and the abstraction packages as just another protocol like TCP and UDP, hence, hiding the cross-layer behavior of the proposed protocol at the lower layers (See Figure C.3). Furthermore, implementation of the proposed protocol in the form of a loadable kernel module (LKM) will ensure that the kernel patches, hence possible regressions, are avoided, and the new protocol can be unloaded upon request. This approach also ensures that the deployment is relatively easy from the network administrators point of view. Similar approaches could also be taken with other OSs.

II-E Hardware Implementation

A detailed hardware design of a BFP-enabled router or OTN switch is beyond the scope of this paper. However, preliminary considerations suggest that whether we consider incorporating BFP into a router (BFP over Ethernet) or within OTN directly, the hardware required to extract chains from the optical transport layer, frame, map, buffer, forward, switch through a switch fabric and repackage into an outgoing transport stream does not change substantially. One logical change in function is that buffers that may be associated with ports in a conventional router become associated with specific chains in BFP, but this does not impact hardware. Significant changes do occur however in the control plane with the introduction of scheduling and for BFP over OTN a potentially more dynamic provisioning requirement than for traditional OTN.

III. SIMULATION STUDY

In our simulation study we compare the ability to efficiently handle bulk data transfers of our approach with that of TCP Westwood [16], a high-speed version of TCP currently implemented in the Linux kernel. All simulations were run using Omnet++ [17,18]. The two approaches are compared in terms of goodput, end-to-end delay and average buffer size. A dumbbell topology (See Figure C.4) was selected for our simulation study. More complex topologies and routing-related aspects are out of the scope of this paper and will be part of our future studies. The propagation delay for each link is set to $1mS$ giving a RTT of roughly $6mS$. Bitrate was set to $10Gb/s$ for all links.The size of each data frame (Figure C.2) is set to be one BPF, and the same size is selected for the voids (i.e. one void is comprised of a single BVF) for all BFP cases. We consider transactions varying in size from $500kB$ to $1GB$ and repeat each transmission until statistical stability is reached.

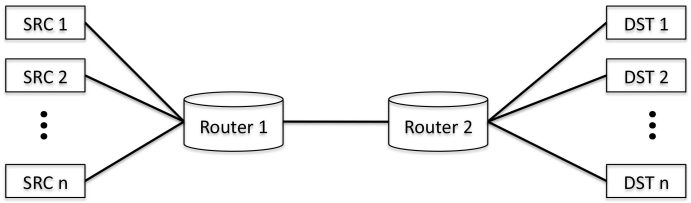


Figure C.4: Simulation topology.

The time interval between two consecutive transactions (for both TCP and BFP) was modeled using both an exponential distribution with mean $\mu = 1mS$ and a generalized Pareto distribution with scale parameter $\sigma = 0.0003$ and shape parameter $\xi = 2$. The latter case was selected to test the protocols with more realistic traffic patterns. However, simulation results did not show significant differences between the two distributions for the performance parameters considered (due to space constraints only results for exponential inter-arrival times are reported).

For the BFP over OTN case, each OTN frame carries 15222B of payload, corresponding to the payload area of the OTN frame minus the GFP overhead including core, payload header, and two bytes of the extended header used as a binary counter to number each frame in a chain, allowing a maximum transaction size of roughly 1GB per chain. An Ethernet jumbo frame with 9000B of payload was selected as the basic payload frame for the BFP over Ethernet case. A TD of 8 was used for all BFP cases and a maximum buffer size of 12 frames ($TD + 4$) was selected for both BFP

cases. This is not a strict requirement and can be relaxed if we are willing to trade buffer space for higher utilization and lower CP blocking probability. For the TCP simulation a MSS of 8960B was selected and the layer 2 MTU was set to 9000B (Ethernet jumbo frame). A RED queuing discipline [19] was used in each router, with: queue weight $q_w = 0.002$, minimum threshold $min_{th} = 5$ packets, maximum threshold $max_{th} = 50$ packets and maximum packet marking probability $max_{p_b} = 0.02$.

III-A Goodput

Instead of considering link utilization (which could be misleading, especially in the TCP case) we consider the goodput normalized to the maximum link capacity. Protocol overhead was excluded since this is not part of the payload. Figures C.5 and C.6 show the average normalized goodput for BFP over Ethernet and TCP respectively. Performance of BFP over OTN were also studied. Results show nearly identical performance of BFP over OTN and over Ethernet. Due to space constraints only the latter is shown here.

We found that for transaction sizes $> 10MB$, in both BFP cases goodput increases linearly up to roughly 99% (reached when the number of sources equals the TD). This is the result of scheduled interleaving of chained data with CP blocking occurring only when normalized goodput is close to its maximum. Beyond this point goodput remains above 70% for both the BFP cases and all transaction sizes considered. Note that with BFP, small transactions ($\leq 10MB$) can achieve high link utilization due to interleaving of flows coming from different sources.

In the TCP case, each source tends to fill the entire bandwidth of their link to the bottleneck router which gives higher goodput with respect to the BFP case for long lived TCP flows ($\geq 300MB$) as long as the number of sources remains limited (≥ 3). For shorter flows ($\geq 100MB$) TCP either does not ramp up fast enough to fill the available bandwidth or, as the number of sources increases, TCP is prone to packet drops which trigger the TCP backoff algorithm, reducing the offered load from the TCP sources and resulting in poor link utilization. Another issue with multiple TCP sources competing for the same resources, common in data center environments, is TCP incast [20], which leads to throughput collapse. While this phenomenon was observed in our simulations for the TCP case it did not occur in the BFP case. Figure C.7 shows a comparison of normalized goodput between BFP and TCP for transaction sizes $\geq 100MB$. The rapid goodput collapse observed for large transactions ($\geq 100MB$) in TCP as the number of sources increase suggest a significant advantage relative to the amount of offered load that BFP can handle

with respect to TCP. BFP can accomodate over 40% more load with respect to TCP without incurring goodput collapse.

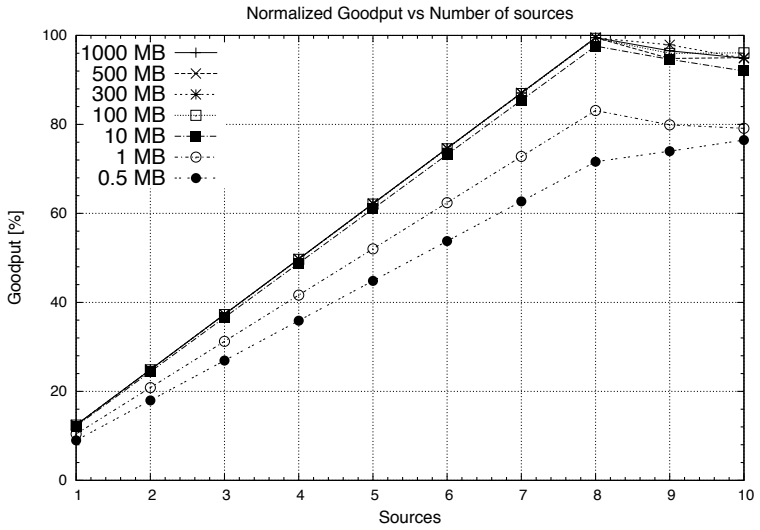


Figure C.5: Normalized Goodput for BFP over Ethernet.

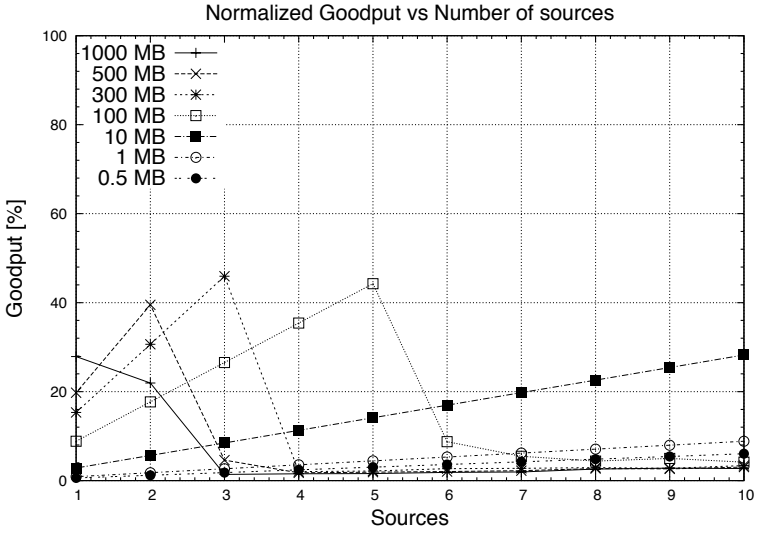


Figure C.6: Normalized Goodput for TCP.

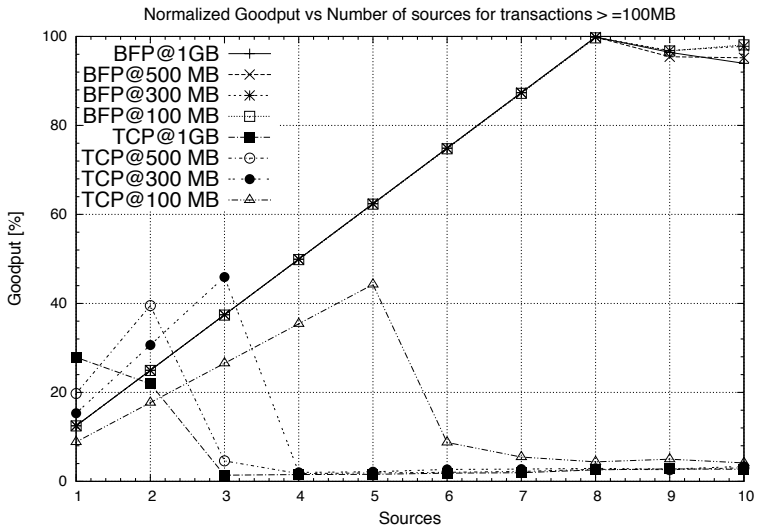


Figure C.7: Normalized Goodput for transactions $\geq 100MB$.

III-B Delay

As a measure of the delay performance we selected the average time needed to successfully complete one transaction, including the time needed to setup the connection and any data retransmission. A new connection is setup for each new transaction. A comparison of the end-to-end delay performance for transactions $\geq 100MB$ is shown in Figure C.8. For transactions $\geq 500MB$ and up to 2 sources TCP tends to fill the available bandwidth and achieves smaller delays. As the number of sources increase, TCP goodput collapses (see Figure C.6) and end-to-end delay rapidly increases to values up to over 30 times larger than BFP. Similar performance can be seen for all other transaction sizes (e.g. $300MB$ and $100MB$), although in these cases BFP achieves better delay performances over the entire range considered. The rapid degradation of delay performance of TCP is due to the high packet drop rate occurring when multiple sources compete for the same resources as well as to the large number of acknowledgements (ACK) used by TCP. Although in the topology studied the Round Trip Time (RTT) is relatively small, the effect of ACKs transmission has a heavy influence on TCP delay performance (each ACK will add to the overall delay an amount proportional to $RTT/2$). Delay will further degrade in the TCP case as the network diameter increases. This would also happen in the BFP case

but to a much smaller degree due to the much smaller amount of control information exchanged between source and destination.

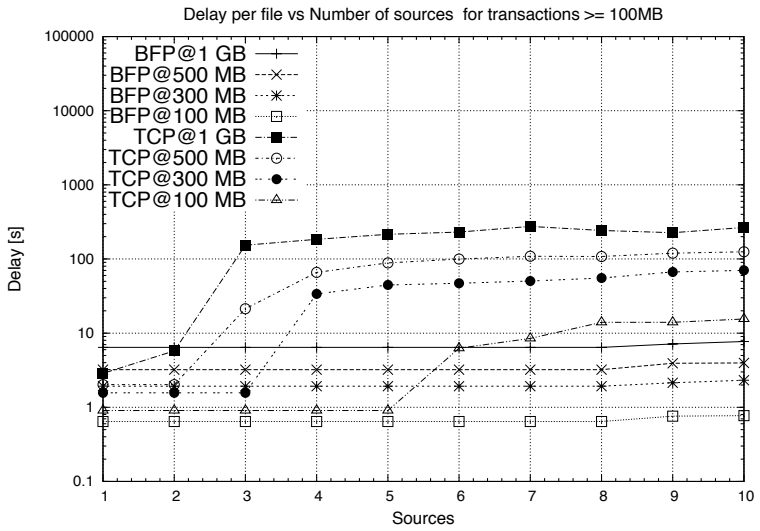


Figure C.8: Delay per transaction for transactions $\geq 100MB$.

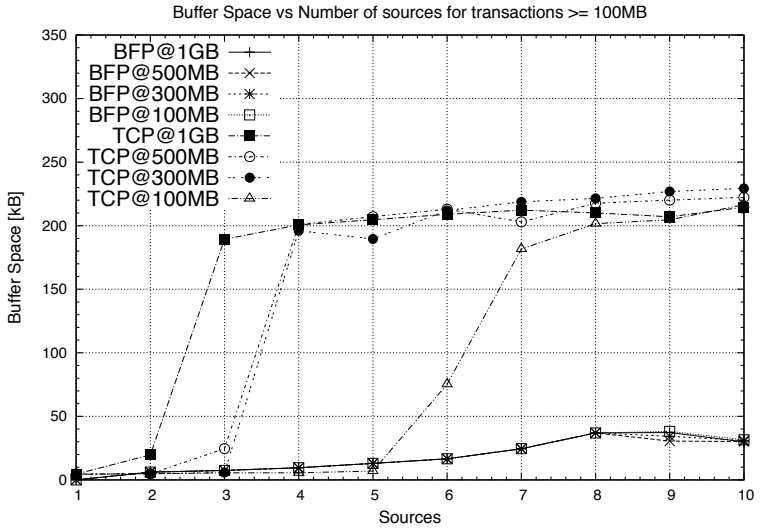


Figure C.9: verage Buffer Size for transactions $\geq 100MB$.

The particular buffering technique used by BFP also improves delay performance with respect to TCP by reducing the queuing delays when the network is congested (See Figure C.9). Lastly, due to the periodic configuration of data frames in each BFP transaction, the variation of the end-to-end delay over the entire range considered is much smaller for BFP, resulting in more stable delay performance of BFP over TCP.

III-C Buffer Size

Figure C.9 shows a comparison of the average buffer space utilized by BFP and TCP for transactions $\geq 100MB$. Average buffer size for BFP is proportional to the BFP frame size and the selected TD, resulting in stable and predictable buffer occupancy in any load condition. The random nature of TCP packet arrival will tend to fill the buffer quickly as congestion approaches. Figure C.9 shows that for a transaction size of $1GB$, buffer size grows rapidly to values several times larger than in the BFP case when more than 2 TCP sources compete for the same bottleneck link. Similar performance is observed for other transaction sizes. Lastly, since BFP uses its buffers to delay each incoming chain in order to interleave it with previously booked chains, the maximum buffer size for BFP is a function of the chains configuration rather than of the network load, and can therefore be tightly controlled.

IV. CONCLUSION

This paper presents BFP, a network protocol designed to efficiently handle large transactions over the existing network infrastructure. Using BFP, large file transactions are handled at lower layers (e.g.$L1$ and $L2$). Higher layers are accessed only when needed with routing functionalities that are only used during connection setup and not for every single data frame as in TCP. Scheduling allows BFP to accommodate network loads over 40% higher with respect to TCP without incurring goodput collapse and achieving goodput values close to 100%. Stable end-to-end delay performance is also achieved, with values over 30 times smaller with respect to TCP. Buffer occupancy of BFP in congested scenarios is predictable and much smaller relative to TCP. The results presented show the potential of BFP to achieve better use of the deployed capacity, potentially resulting in significant cost savings for the network operators in both transport and data center environments.

References:

1. Thomas Engel, Dr. Achim Autenrieth , Dr. Stefan Voll. *Quantitative Analysis of Network Architectures for Future National Optic Transport Networks (OTN).* Photonische Netze 03. 04.05.2010 in Leipzig

2. Tiejun J. Xia, Steven Gringeri and Masahito Tomizawa, *High-Capacity Optical Transport Networks*, IEEE Communication magazine, 201

3. Marcon, M.; Dischinger, M.; Gummadi, K.P.; Vahdat, A. *The local and global effects of traffic shaping in the Internet.* Communication Systems and Networks (COMSNETS), 2011 Third International Conference on , vol., no., pp.1,10, 4-8 Jan. 2011

4. *Cisco Visual Networking Index: Forecast and Methodology.* CISCO white paper. Retrieved on October 20 2013 from: `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/whitepaperc11481360.pdf`

5. Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. *Towards predictable datacenter networks.* SIGCOMM Comput. Commun. Rev. 41, 4 (August 2011), 242-253.

6. Dennis Abts and Bob Felderman. *A guided tour of data-center networking.* Commun. ACM 55, 6 (June 2012), 44-51.

7. ITU-T G.709 (02/2012) *Interfaces for the optical transport networks*

8. ITU-T G.872 (10/2012) *Architecture of optical transport networks*

9. Steve Gorshe. *A Tutorial on ITU-T G.709 Optical Transport Networks (OTN).* PMCSierra white paper, personal communication.

10. Virginia Hutcheon. *OTN to Enable Flexible Networks.* OSA/OFC/NFOEC 2011

11. Ashwin Gumaste, Nalini Krishnaswamy. *Proliferation of the optical transport network: a use case based study.* IEEE Communication magazine, 2010

12. Baldi, M.; Marchetto, G., *Pipeline Forwarding of Packets Based on a Low-Accuracy Network-Distributed Common Time Reference,* Networking, IEEE/ACM Transactions on , vol.17, no.6, pp.1936,1949, Dec. 2009

13. IEEE ETHERNET, IEEE Standard 802.3, 2012

14. ITU-T G.7041 (04/2011). Generic Framing Procedure.

15. ITU-T G.7044 (10/2011). Hitless adjustment of ODUflex(GFP).

16. Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. *TCP westwood: end-to-end congestion control for wired/wireless networks.* Wireless Networks 8, 5 (September 2002), 467-479.

17. *Omnet++*, Discrete Event Simulation System. Available: `http://www.omnetpp.org/`

18. INet Framework for Omnet++. Available: `http://inet.omnetpp.org/`

19. Floyd, Sally; Jacobson, V., *Random early detection gateways for congestion avoidance. Networking*, IEEE/ACM Transactions on , vol.1, no.4, pp.397,413, Aug 1993

20. Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. *Understanding TCP incast throughput collapse in datacenter networks.* In Proceedings of the 1st ACM workshop on Research on enterprise networking (WREN 09). ACM, New York, NY, USA, 73-82.

Appendix D

Big File Protocol (BFP) for OTN and Ethernet Transport Systems

Ilijc Albanese*, Yağız Onat Yazır†, Stephen W. Neville*, Sudhakar Ganti† and Thomas E. Darcie, *Fellow*, IEEE*
*Department of Electrical and Computer Engineering
†Department of Computer Science
University of Victoria, Victoria, Canada.

[IEEE/OSA Journal of Optical Communications and Networking (JOCN). Accepted for publication.]

***Abstract*:**

Demand for high-bandwidth services is rapidly increasing, placing current network infrastructure under significant stress. Many interesting approaches have been studied in order to cope with this growth, but most require major redesigns of current network architectures as well as large-scale hardware redeployments.

Recently Big File Protocol (BFP), a network protocol designed to efficiently handle large file transactions, was proposed to provide improvements over traditional IP protocols (TCP/UDP) in terms of bandwidth utilization and end-to-end delay. This paper presents an integration strategy for BFP within the currently deployed OTN transport infrastructure relying as much as possible on current hardware and functions, thereby minimizing the deployment impact. Simulation results show capacity utilization improvement of over 40% relative to TCP.

I. INTRODUCTION

The rapid traffic increases experienced by the current network infrastructure place it under considerable stress, leading to a significant increase in computational requirements and power consumption. Most of this traffic increase is due to the rapid

proliferation of bandwidth-intensive applications that often involve large transactions with stringent delay requirements (e.g. VoD, CDN content update)[1]

Traditional IP protocols such as TCP or UDP handle large transactions by fragmenting them into a large number of small packets which are routed and processed individually. This approach, so successfully applied to historical traffic, results in a highly dynamic, unpredictable traffic mix and places a significant computational load on current routers, increasing power requirements[2], and ultimately becoming a constraint to IP router evolution. However, as traffic mix shifts to include an increasing number of large transactions, which are more predictable and less dynamic, new protocols may perform better.

Many all-optical solutions to these issues have been proposed [3,4,5,6,7,8,9,10] but their potential value remains limited to specific applications. Most of these approaches advocate for a major architectural redesign of the current infrastructure, requiring large-scale deployment of, for example, optical switches.

Exploring more efficient methods to handle the new traffic mix over the existing infrastructure can provide a more attractive evolutionary path [11]. The Big File Protocol (BFP) approach proposed recently [12] offers advantages similar to those promised by all-optical approaches but without requiring large scale hardware redeployment. BFP can achieve goodput values close to 100% without incurring throughput collapse, and offer much smaller and stable delay performance while using only a fraction of the buffer resources used by TCP [12].

On the transport side, recent standardization of ITU-T G.709/G.872 Optical Transport Network (OTN) [13,14,15] provides a flexible and efficient transport infrastructure supporting protocol agnostic transport services, simplified management operation, and a more efficient use of the deployed capacity [16,17] OTN is currently supported by most of the commercially available transport equipment and provides mappings onto a common underlying transport layer for most existing and future higher layer protocols (e.g. 10GE, 40GE, and so on).

This work explores the integration of BFP within the current OTN transport infrastructure. By handling large transactions at lower layers, and accessing higher layers only when strictly necessary, BFP avoids most of the computations required by higher layer transport protocols (e.g. TCP), and provides most of the benefits promised by all-optical solutions, in terms of bandwidth efficiency and cost per bit transported, while attempting to reuse most of the hardware and protocols already available. Mapping of BFP onto Ethernet is also considered.

The rest of the paper is organized as follows. Section II provides an overview of the proposed approach. Section III provides an overview of the transport layer and BFP mapping and integration procedures. Section IV provides an example of BFP integration within currently available hardware. Section V presents simulations results comparing BFP to TCP. Section VI discusses the benefits of BFP and future research directions are addressed in Section VII.

II. BIG FILE PROTOCOL

BFP [12] is a cross-layer transport protocol designed to handle large transactions (e.g. $\geq 100MB$) by shaping traffic to leverage currently available transport layer functionalities. This enables better use of the deployed capacity and reduces the computational load placed on the network by bandwidth-intensive applications. Here we briefly summarize the approach.

Nodes implementing BFP shape traffic from large transaction into periodic, semi-transparent concatenations of fixed-size data frames. Henceforth such concatenations are referred to as *chains*, while their period is referred to as the *Transparency Degree* (TD) (Figure D.1). In order to provide flexibility in selecting the portions of available bandwidth occupied by each chain, and to minimize the impact of frame retransmissions in case of transmission errors, data frames are assembled using one or more Basic Payload Frames (BPF) constituting the atomic data container for BFP, as shown in Figure D.2.

Regardless of the number of BPF, each data frame is managed and switched as a single entity. A lightweight, distributed scheduling protocol specifically designed for BFP chains is used to perform end-to-end resource reservation on a per-chain basis. Since BFP aims at providing end-to-end transport services, a routing strategy must be devised. As discussed in detail in Section II-B, BFP can reuse any available routing protocol.

In the remainder of this work we assume that an overlay network of fixed-capacity logical links is provisioned within the OTN network for BFP traffic only, such that BFP traffic is completely separate from non-BFP traffic.

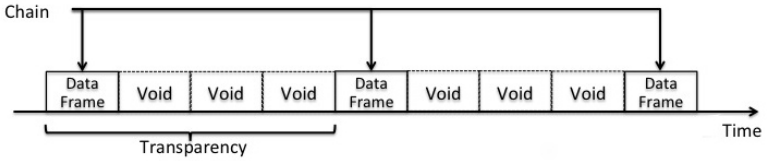


Figure D.1: Data transaction organized into a chain with $TD = 4$.

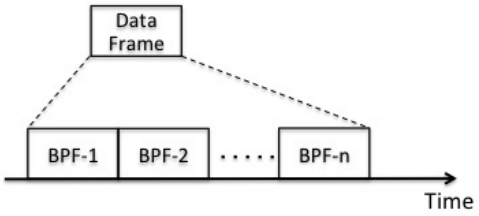


Figure D.2: Data frame assembled by multiple BPF.

II-A Admission Control And Scheduling

Our BFP resource reservation protocol uses a Control Packet (CP) to perform reservation for the BFP chains over an end to end path (Figure D.3). The periodic structure of each chain allows for the tight interleaving of the chains, where buffers are used to align data frames in time and to overcome timing uncertainties.

A BFP source associates each chain (e.g. chain *"i"*) to a CP (CP_i) for each data transaction. The size of each transaction is assumed to be known in advance. As a result, the configuration and timing of each chain are also known in advance, allowing for automatic resource release once the last bit of the chain passes through each intermediate node.

If a reservation fails, i.e. the CP request cannot be accommodated by some node along the path, a $NACK$ is sent from the blocking node over the reverse path to free previously booked resources. Upon receiving a $NACK$ the original source node waits for a random amount of time before retrying.

If reservation is successful, the destination node sends an ACK to the source node (over the reverse path) to inform the source node that the network is ready to handle the BFP transaction. Upon receiving the ACK, the source node starts transmitting the BFP chain.

Based on the Round Trip Time (RTT) for each data path, the source node computes for each chain an Expected Time of Arrival (ETA) according to Equation D.1, and includes it in the CP together with chain configuration and routing information. This includes: (1) source and destination addresses, (2) the chain Expected Time of Arrival (ETA_i), (3) data frame size, i.e. the number of BPF in each data frame, (4) the number of data frames per chain, and (5) chain periodicity (or *Transparency Degree -*TD_i*-*).

The CP_i is then sent over the data path to reserve necessary switching and buffer resources for a BFP chain (with reservation starting at the time indicated by the ETA_i).

$$ETA_{src} = \sum_{i=1}^{N}(\tau_i + p_i) + \tau_{ACK} \tag{D.1}$$

where:

N number of nodes in the path

τ_i propagation time to reach node i

p_i estimated CP processing time for node i

τ_{ACK} time for the ACK to reach the source node

At each node the CP_i is processed, Buffering Time (BT_i) for the chain is computed (Equation D.2) and included in the ETA_i parameter of the CP_i together with the CP processing time (p_i) according to Equation D.3 before routing it towards the next node.

$$BT_i = t_{e,1} - ETA_i \tag{D.2}$$

$$ETA_{i+1} = ETA_i + BT_i - p_i \tag{D.3}$$

where:

$t_{e,1}$ is the ending time of the first available timeslot on the outgoing channel.

A time diagram illustrating the path reservation and transmission procedures for a BFP transaction is shown in Figure D.3.

II-B Routing

Since the proposed BFP approach provides a methodology separate from IP for end-to-end data transmission, a way to route chains from source to destination is required. Although routing and load balancing algorithms can be specifically designed, taking into account the periodic configuration of the BFP transport structure, it is also possible to simply re-use protocols already in place as discussed below. In what follows, we assume that at each node control and data traffic traverses the same ports in both the forward and backward directions.

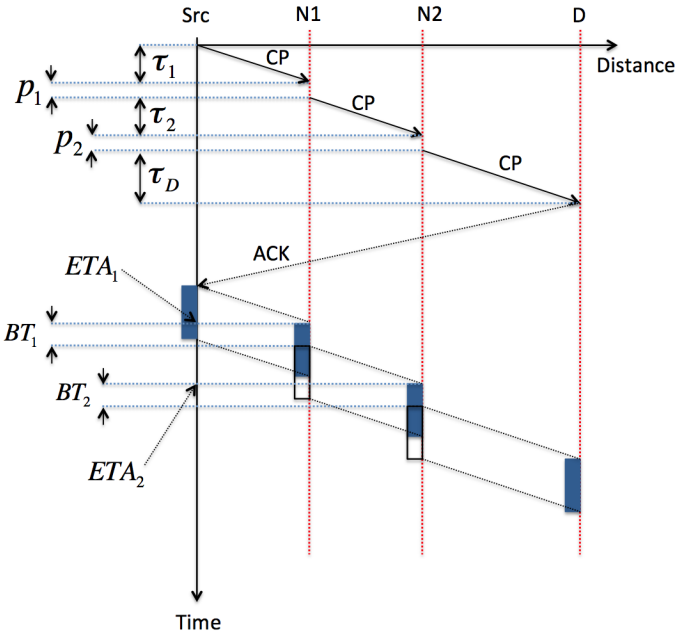


Figure D.3: BFP Path Reservation and Transmission Sequencing.

As a CP reaches a node, information about the source and destination address for its relative chain is passed on to whatever routing protocol is available at that node. The routing layer should provide an output port relative to the chain destination address. Virtually any protocol can be used and there is no requirement for it to be the same at each node in the path.

Routing information received from the routing layer is translated into the following ordered pair $< Input\_port, Output\_port >$ and stored in a locally held table referred to as *port-map* table as illustrated in Table D.1. This table simply binds at each node the input port on which the BFP chain will arrive with its relative output port.

Based on each chain *input port*, ETA, TD, and ID incoming chains are delineated. The port-map table is accessed and all data frames associated with the incoming chain are buffered according to the BT value for the chain as stored in the *port-map*. After a chain traverses the node its relative table entry is deleted.

The use of a *port-map* table avoids having to perform address lookup and routing decisions for each data frame in a chain, significantly reducing the amount of computational load placed on the node and allowing for potentially significant energy savings [18].

The size of the *port-map* table depends on both on the number of BFP transactions each node is able to handle and on the average duration of each reservation and

Table D.1: *port-map table.*

Chain ID	TD	ETA	Input Port	BT	Output Port
64	8	6.2 mS	7	0.043 mS	18

transmission procedure. Although the size of the table may become cumbersome, the assumption is that BFP transactions are sufficiently long-lived, i.e. large, such that the reservation procedure constitutes an acceptably small overhead.

III. TRANSPORT LAYER

The shift towards data-centric, packet-based traffic exposed the limitations of the prior SONET/SDH infrastructures in terms of scalability, flexibility and OAM&P capabilities. In order to overcome such limitations new transport layer technologies were developed.

OTN and Ethernet-based transport architectures are currently the deployments of choice, and are widely supported by most of the available hardware in both IP networks and Data Center (DC) environments.

BFP can be mapped onto both OTN and Ethernet-based transport technologies. Its integration within the current transport layer can take the form of a control plane (software) update, enabling the easy integration of BFP capabilities within widely deployed OTN and Ethernet transport architectures.

In what follows we will give a brief overview of both OTN and Ethernet based transport technologies and discuss how BFP maps onto and integrates with these.

III-A OTN Overview

ITU-T G.709 (OTN) [13,15] is a widely supported, standardized transport architecture providing protocol-agnostic transport services for both CBR and packet-oriented protocols.

Standard containers for any client signal available today are defined together with their relative mapping procedures [13,15]. Flexible containers (i.e. ODUflex) are also defined in order to support CBR and packet-based clients over a wide range of bitrates.

With respect to legacy SONET/SDH architectures, G.709 offers less complex and more flexible OAM&P procedures, more efficient data aggregation for enhanced channel utilization and stronger Forward Error Correction capabilities [13].

Each OTN frame carries a payload of 15232 bytes over which one or more client signals are mapped. For cell or packet based clients, OTN uses GFP [21] to encapsulate data packets and generate a continuous stream of GFP frames which is then

mapped in an octet-aligned manner directly onto the OPU payload area. Rate adaptation is done using GFP idle frames, which are transmitted anytime there is no data to send.

For a complete description of the mapping procedures defined for OTN the interested reader can refer to [13].

III-B BFP Overlay Network

Coexistence of various protocols and packet sizes over the same logical links can be problematic as it can lead to unfairness among data flows and inefficient resource usage [19]. In order to avoid problems deriving from coexistence of BFP with other protocols and to allow BFP to use frames of virtually any size without negatively impacting non-BFP traffic, we leverage the ability of OTN infrastructures to establish flexible data channels (ODUflex [13]). In the rest of this work we assume such link to be provided for BFP-only use.

Without disrupting the service, ODUflex channels can be easily resized [20] to match the particular bandwidth demands. The capacity of the BFP overlay networks can be planned offline over coarse timescales, while Hitless Adjustment of ODUflex [20] can be used to perform online adjustment of the BFP overlay networks.

III-C Ethernet Overview

Another widely supported technology which has become increasingly important at the transport layer is Ethernet [22]. Commonly used for Data Center (DC) networks [23], it is also being deployed as a transport technology [24,25,26,27]. Ethernet-based networks provide a cost-efficient interconnection technology and lower complexity of OAM&P procedures with respect to SONET/SDH. Furthermore, Ethernet signals can be mapped onto the OTN transport layer.

Standard Ethernet frames [22] carry a payload varying from a minimum of 42 bytes to 1500 bytes. This limitation can be removed when jumbo frames are used, allowing up to 9000 byte payloads. Each Ethernet frame has a preamble of 7 bytes: 1 byte of Start Frame Delimiter (SFD), 6 bytes for source MAC address and 6 for destination MAC address, a 4 byte Frame Check Sequence is also appended to the Ethernet frame. Furthermore, after each frame is sent, transmitters are required to transmit a minimum of 12 bytes of Inter Frame Gap (IFG). As BFP focuses on large data transmissions, Ethernet with jumbo frames will be assumed in the discussion below.

III-D Generic Framing Procedure (GFP) Overview

GFP provides a mechanism to adapt higher layer traffic from a client to an underlying transport layer. GFP was originally developed to allow packet traffic to be transported over the SONET/SDH backbone in a more efficient way, and now it allows packet traffic to be carried over the OTN infrastructure in much the same way. GFP can perform adaptation for both Protocol Data Unit (PDU)-oriented clients (i.e. GFP-frame mapped or GFP-F) and block-code-oriented clients (i.e. GFP-transparent or GFP-T). When using GFP-F client packets are mapped directly onto the payload area of the GFP-F frame after all unnecessary header information (e.g. preamble bits for Ethernet frames) is removed.

The issue with this approach is that some protocols (often proprietary implementations of standardized protocols such as Ethernet) may use the bits discarded by the GFP-F mapping procedure to code control information [15]. In this case, using GFP-F will result in loss of important control data.

Another issue to consider is that in order to assemble the GFP-F frame the payload length must be known in order to assign the correct value to the PLI field in the GFP core header (See Figure D.4). This forces the transmitting node to buffer the entire data frame prior to transmission of the GFP-F frame, deteriorating the delay performance of the client protocol and causing problems for delay-sensitive protocols such as Storage Area Networks protocols. In order to remove this drawback a variation of the GFP protocol was developed to assure stable latency and control code transparency for the client signal, namely GFP-T [21].

In GFP-T the client data is transcoded into a new block code (termed "superblock"), which includes all the client signal control information as well as an error check appended to each superblock. An integer number of superblocks is then mapped onto the GFP frame. How many superblocks are mapped into each GPF-T frame depends on the difference between the client data rate and the server layer data rate. As the number of required superblocks is known at transmission time, the need to buffer entire client data frames is removed.

The GFP frame format is shown in Figure D.4 [21]. All GFP frames share the same basic structure comprised of a core header and a payload area. The core header contains a 16-bit Payload Length Indicator (PLI), used to encode the length of the GFP frame plus a CRC-16 (cHEC bytes in Figure D.4) to protect the PLI field. The GFP receiver correlates the bit sequences of the PLI and cHEC fields to identify the

beginning of each GFP frame and uses the PLI information to complete the frame delineation process allowing it to lock on to each stream of GFP frames.

The payload area has its own header, which is divided into a two byte Type header and an optional Extension header (area C of Figure D.4), which varies in size from zero (no Extension header) to 60 bytes. This field is also protected by a CRC-16 error control code (eHEC, area D of Figure D.4). The Type header has the following subfield:

- Payload Type Identifier (PTI, 3-bits): Identifies which type of client signal is mapped onto the GFP frame. Currently only 3 types of GFP frames are specified in the standard, namely user data frames, client management frames and management communication frames.
- Payload FCS Indicator (PFI, 1-bit): Indicates if the optional payload FCS is appended at the end of the GFP frame (PFI = 1) or not (PFI = 0).
- Extension Header Identifier (EXI, 4-bits): Identifies the type of Extension header used, including the limit case in which no extension header is present (EXI = 0000).
- User Payload Identifier (UPI, 8 bits): Identifies the type of payload carried by the GFP payload information field.

The two bytes of the Type header are also protected with a CRC-16 (tHEC, See Figure D.4).

Note also that not all the possible code points for PTI, EXI and UPI are standardized. In fact, at present most of the available codes are reserved for future use. We leverage this to implement our own BFP code points in order to support our approach (See Sec. D).

Once mapped onto the OTN layer, there is no distinction between GFP-F and GFP-T, and the OTN layer handles both GFP frame types in the same manner [63]. This, together with the use of fixed frame size for BFP data and control traffic, opens the possibility to define a third option for the GFP procedure (referred to as Modified GFP or GFP-M) that can be used as an alternative to GFP-T or GFP-F when adapting BFP traffic to the OTN transport layer. This third option combines the advantages of both GFP-F and GFP-T and is discussed in Section III-E.

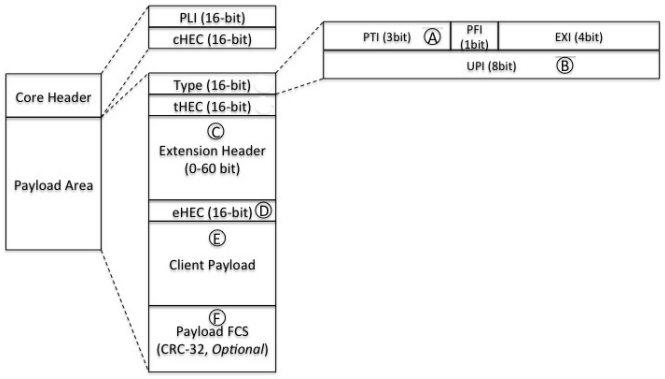


Figure D.4: GFP Frame Structure [21].

III-E Mapping Onto OTN

Many possible variations are possible for implementing BFP. One representative approach defines five different frame types, namely: BPF, CP, ACK, $NACK$ and Basic Void Frames (BVF). This last frame type is used when mapping BFP onto OTN. It is equivalent in size to BPF but carries only stuffing bits. BVF are used to separate the BPF of a chain and give the chain its transparency. They are always preempted by data frames of other chains and discarded by the receiving node.

Each type of BFP frame is mapped directly onto the payload area of a GFP frame (area E of Figure D.4) as shown in Figure D.5 for the case of data frames (i.e. a set of BPF). Frame delineation is left to the GFP layer. BFP traffic is assigned one of the unused code points of the PTI field of the GFP frame Type header (area A of Figure D.4) so the receiving node can distinguish between BFP and non-BFP traffic. Each BFP frame type is then assigned one of the unused code points of the UPI field (GFP Type header, area B of Figure D.4), which is used by the receiving node to distinguish between the various types of BFP frames.

A portion of the GFP Extension header (area C of Figure D.4), with a size depending on both the maximum number of data frames per chain and the maximum size of the data frame allowed in a chain. GFP Extension header is used to number both data frames and the basic payload frames (BPF) of which it is comprised (See Figure D.2). This way the receiving node can identify the data frames and the BPFs of each chain allowing, in case of transmission errors or frame loss, identification and retransmission of single BPFs.

Although BFP can be adapted to the OTN transport infrastructure using either GFP-T or GFP-F, both GFP flavors have their own drawbacks, namely: GFP-F

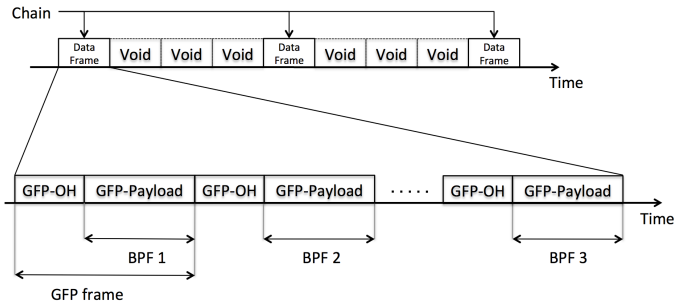


Figure D.5: Mapping of data frames onto GFP frames. Voids have the same structure but only carry stuffing bits in the GFP-Payload area.

needs to buffer the entire frame in order to correctly set the PLI field in the core header, while GFP-T needs to transcode the client signal.

In the BFP case, the sizes of the data and control frames are known in advance and therefore the PLI field of the GFP frame core header can be determined before the GFP frame is fully formed. This frees transmitting nodes from the need to buffer each frame for mapping purposes (GFP-F) or from having to transcode the client signal (GFP-T). BFP frames can then be mapped directly onto GFP-F frames without the need to buffer each frame prior to mapping it onto a GFP frame (note that frame buffering for mapping purposes can negatively impact delay-sensitive applications). We will refer to this use of the GFP as "Modified Generic Framing Procedure" or GFP-M meaning a GFP-F mapping where the PLI field for each frame is known in advance. The use of GFP-M would make no difference for the OTN layer since it does not distinguish between various kinds of GFP frames and the structure of a GFP-M frame is exactly the same of that of GFP-T or GFP-F frames, the only difference being in the way GFP-M frames are assembled.

GFP frames containing BFP data or control traffic are then mapped directly onto the payload area of OTN containers. The OTN layer will handle BFP traffic as a stream of standard GFP frames.

If required, BFP control traffic can also travel on its own dedicated control channel or over a standard IP network, while leaving the functioning of BFP unchanged.

III-F Mapping Onto Ethernet

Alternatively, BFP data and control frames can be mapped directly onto Ethernet frames by setting the size of BPF and BVF equal to the Ethernet frame payload size (e.g.: 9000 B), allowing the proposed protocol to be mapped onto the Ethernet transport layer. In this case, frame delineation is performed by the Ethernet frames

and preamble or Inter Frame Gap (IFG) bits can be used to code frame-related information (e.g. frame number, frame type and data frame size).

BFP admission control and scheduling procedures are implemented in the control plane, and handle data frame buffering and switching by performing delayed reservation of resources (i.e. buffer space and switching resources) and translating routing information into input-port-to-output-port bindings for the chains (See Section II-B). For the Ethernet layer, BFP traffic looks just like standard Ethernet traffic, the only difference being in how long each frame is buffered, which is determined by the BFP functionalities integrated within the control plane. Lastly, note that in the 10GE case, only full-duplex mode is allowed, making the preamble bits unnecessary (10GE receivers ignore preamble bits) and allowing the use of these bits for other purposes, e.g. as proprietary OAM channels [15] (where this also applies to 40GE and to any carrier-grade Ethernet architecture).

IV. IMPLEMENTATION

IV-A Hardware

A detailed hardware design of a BFP-enabled router or OTN switch is beyond the scope of this paper. However, preliminary considerations suggest that whether we consider incorporating BFP into a router (BFP over Ethernet) or within OTN directly, the hardware required to extract chains from the optical transport layer, frame, map, buffer, forward, switch through a switch fabric and repackage into an outgoing transport stream does not change substantially. One logical change in function is that buffers that may be associated with ports in a conventional router become associated with specific chains in BFP, but this does not impact hardware. Significant changes do occur however in the control plane with the introduction of scheduling and for BFP over OTN a potentially more dynamic provisioning requirement than exists for traditional OTN.

BFP functionalities can potentially integrate with a wide variety of existing routing and switching devices. Any device able to perform store and forward of frames and execute the proposed scheduling algorithm can support BFP, given that their control plane can be modified to support the functions required by the proposed protocol.

In order to provide an example of how BFP can integrate with currently available hardware we consider an integrated platform combining OTN and packet switching capabilities in a single platform. Such a device is known as Packet Optical Transport Platform (P-OTP) [28,29,30,31,32] (Figure D.6).

A P-OTP integrates in a single platform functionalities such as: (1) TDM-based traffic (e.g. ODU_k), (2) aggregation and switching of system-wide L2 traffic (e.g. Carrier Ethernet [27,25]), and (3) ROADM functionalities. P-OTP are equipped with electrical switch fabrics able to switch packet traffic as well as OTN traffic, enabling P-OTP platforms to multiplex any type of incoming traffic (i.e. any service) onto any wavelength. A schematic of the system-level architecture of a P-OTP is shown in Figure D.6. The Network Processor (NP, block (A)) represents an OTN processor with integrated framing and mapping capabilities for TDM-based and Carrier Ethernet (CE) transport [29]. The Fabric Interface Chip (FIC, block (B)) serves as an interface between the switch fabric and a network processor, and between a traffic manager and route processor (not shown). A converged ODU_k and packet switch fabric (block (C)) as well as a ROADM (block (D)) are also included in the P-OTP system architecture. A schematic showing how BFP functionalities can be integrated in a P-OTP platform is shown in Figure D.7.

With reference to Figure D.7, incoming optical signals (e.g. OTU_k) are converted into electronic format and processed by the OTN processor (block (A)), which extracts the ODU_k signal and packetizes it before passing it to the Traffic Manager (block (B)) as packets. Here BFP traffic is separated from non-BFP traffic. BFP Control Packets (CP) are processed and information about configuration and timing of their relative chains is used to schedule buffer space and switching resources for the data chains. CP source and destination addresses are passed to the routing layer through the Routing Layer Interface (block (C)) which communicates with whichever routing protocol is available and presents the FIC with an input-port-to-output-port mapping relative to the addresses provided. This information is used by the Traffic Manager (TM) to populate a table (i.e. the port-map table, see Table I in Section II-B), which contains the bindings of input-to-output ports for each chain. Devices integrating FIC, TM and packet processor (PP) in one single chip are currently available on the market [33]. Data frames from each chain are routed through a buffering stage (block (D)) to be delayed by an amount of time consistent with the buffering time (BT) computed for each chain. At the output of the buffering stage, data frames are routed to the FIC where they are adapted to the format required from the Switch Fabric (Block (F)) before being switched to their respective output ports by switch fabric without further processing. Chains going through the same ports will naturally interleave with one another. A description of the functional blocks of the input NIC (block (G) of Figure D.7) is provided below:

- *OTN Processor*, block (A): The OTN processor extracts ODU_k signals from the input OTU_k, packetizes the ODU_k and passes the packetized signal to the FIC (block (B)).

- *Traffic Manager* (TM) + *Packet Processor* (PP), block (B): The Traffic Manager separates BFP traffic (control and data) from standard traffic (i.e. non-BFP traffic). Control packets (CP) are processed in this block and BT is computed for each incoming chain in advance. Based on the source and destination addresses carried by the CP, this block requests routing information to the routing layer through the Routing Layer Interface (block (C)). Upon receiving routing information, the TM schedules switching resources and reserve buffer space (in advance) for incoming chains. The TM also populates/updates the port-map table (See Section II-B) with the newly obtained routing information. When BFP data frames (i.e. chains) arrive at the TM, these are routed to the buffer stage (block (D)) where each data frame is delayed by an amount of time consistent with its relative BT before being routed towards the FIC (block (E)).

- *Fabric Interface Chip* (FIC), block (E):

  This block performs standard FIC functions (i.e. traffic adaptation and congestion control) for both BFP and non-BFP traffic.

- *Routing Layer Interface*, block (C): The Routing Layer Interface can be thought as a universal translator of routing information. It interfaces with various routing protocols and provides the TM with the information necessary to build and update the port-map table.

- *BFP Buffering Stage*, block (D): This stage is controlled by the TM and simply delays frames of an incoming chain by the amount indicated by the BT before passing them to the FIC (block (E)). Note that, although this block is represented as a separate entity, the buffer space needed to support BFP can be implemented by using the buffers already available on the P-OTP, the only difference being in the way this (reserved) portion of the system memory is accessed.

- *Switch Fabric*, block (F): This block is functionally equivalent to blocks (C) and (D) of Figure D.6. It performs switching operations for BFP traffic and non-BFP traffic.

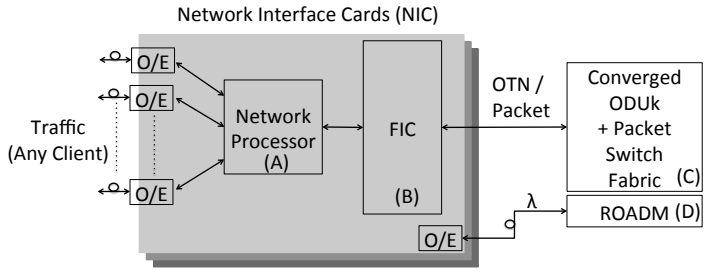


Figure D.6: P-OTP, System Architecture.

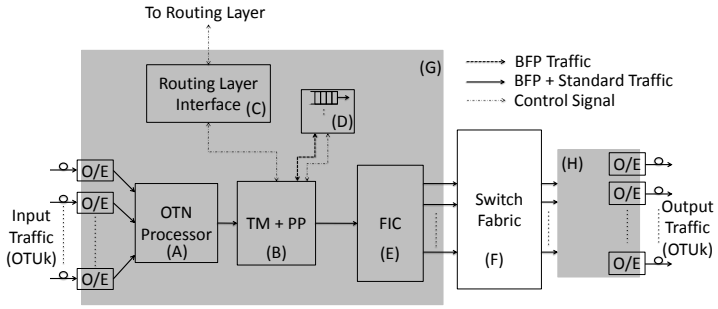


Figure D.7: P-OTP with BFP Functionalities.

- *Output NIC*, block (H): This block is functionally equivalent to the Input NIC (block (G)) and is comprised of the same hardware (for graphical simplicity we indicate it with a grey block). It is in charge of mapping outgoing packets, frames and any other eventual client signal onto OTU_k signals for transmission onto the optical mesh.

IV-B Application Programmer Interface

In order to deploy and use BFP services, the proposed protocol needs to be installed in the network stack in a way that ensures a certain level of familiarity for the clients while masking its cross-layer behavior. Any client willing to use BFP services should be able to access them by undertaking the well-known and common socket related actions – such as creating, connecting, listening, sending, receiving, etc.– in a way that is not much different from the actions that are involved in programming with TCP or UDP sockets. This way applications and application developers will see BFP just as another protocol such as TCP or UDP.

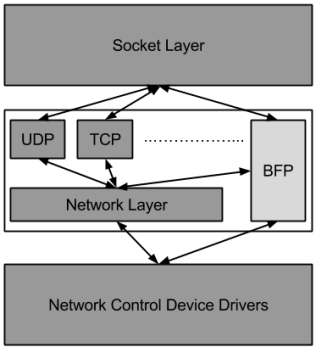


Figure D.8: Integration of the proposed protocol in the current layered architecture.

Registering BFP with the OS socket layer allows easy access to BFP services by just using an indicator pointing to the newly registered protocol upon socket creation; the necessary functionality is provided with identical function headers. In the same sense, the structural similarities will also have a positive impact on incorporating the proposed protocol with software packages that abstract the socket layer.

Fortunately, the necessary installation options already exist in the Linux Kernel. The proposed BFP can be registered with the socket layer as a new transport layer protocol making it appear to the application developers and the abstraction packages as just another protocol like TCP and UDP, hence, hiding the cross-layer behavior of the proposed protocol at the lower layers (See Figure D.8).

Furthermore, implementation of the proposed protocol in the form of a loadable kernel module (LKM) will ensure that the kernel patches, hence possible regressions, are avoided, and the new protocol can be unloaded upon request. This approach also ensures that the deployment is relatively easy from the network administrators' point of view.

A similar approach can used to integrate BFP functionalities within other OSs.

V. SIMULATION STUDY

Using *Omnet++* [35], our simulation study aims at comparing the ability to efficiently handle bulk transfers of BFP versus that of a high-speed version of TCP (TCPWestwood [34]), currently implemented in both the linux kernel and in the simulation software used for this study [36].

A simple dumbbell topology (Figure D.9) was chosen in order to isolate the performance measures of interest (i.e. normalized goodput and delay per transaction) from topology and routing-related aspects, which will be part of our future work.

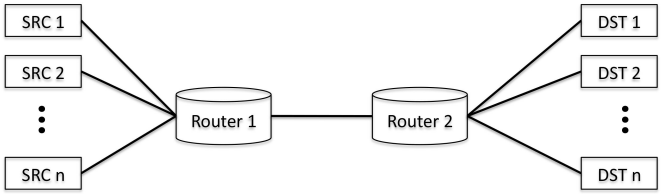


Figure D.9: Simulation Topology.

With reference to Figure D.9, the propagation delay for each link is set to 1mS giving a round-trip time (RTT) of roughly 6mS. Link capacity was set to $10Gb/s$ for all links.

Transaction sizes varying from $500kB$ to $1GB$ were tested, and for each transaction size the connection setup and transmission procedure was repeated until statistical stability for the parameters under study was reached. The size of each data frame (See Figure D.2) was set to one BPF, the same setting was used for the voids (i.e. each void is comprised of a single BVF).

For the BFP over OTN case, the size of each BPF was set to $15222B$, corresponding to the payload area of the OTN frame minus the GFP overhead including core, payload header, and two bytes of the extended header which are used as a binary counter to number each frame in a chain. This two-byte counter allows up to 2^{16} frames per chain, corresponding to a maximum size for the bulk data carried in a chain of roughly $1GB$.

For the BFP over Ethernet case the BPF size was set to be equal to the payload area of an Ethernet jumbo frame, corresponding to $9000B$ of payload per frame.

A TD of 8 was used for all BFP cases and a maximum buffer size of 12 frames $(TD + 4)$ was selected for both BFP cases. This is not a strict requirement and can be relaxed if we are willing to trade buffer space for higher utilization and lower CP blocking probability.

For the TCP simulation a MSS of $8960B$ was selected and the layer 2 MTU was set to $9000B$ (Ethernet jumbo frame). A RED queuing discipline [37] was used in each router, with: queue weight $q_w = 0.002$, minimum threshold $min_{th} = 5$ packets, maximum threshold $max_{th} = 50$ packets and maximum packet marking probability $max_{p_b} = 0.02$.

The time interval between two consecutive transactions was modeled using an exponential distribution with mean $\mu = 1mS$ for both BFP and TCP. In order to study the behavior of the proposed protocol under more realistic traffic patterns,

the study was repeated using a generalized Pareto distribution with scale parameter $\sigma = 0.0003$ and shape parameter $\xi = 2$. However, no significant differences were noticed between Exponentially distributed and Pareto distributed traffic, showing the insensitivity of BFP to the statistical properties of traffic. In the following only results for BFP over OTN with Pareto traffic distribution are shown, performances for BFP over Ethernet show little difference between the two mappings, and can be found in [12].

V-A Goodput

In order to compare the payload delivery capability of BFP and TCP Westwood, we consider the normalized goodput as defined in Equation D.4 below for the purpose of evaluating the portion of the available bandwidth effectively used to deliver payload. Link utilization may be misleading as protocol overhead would be included in its computation. Depending on which protocol is used, protocol overhead may take up a non-negligible portion of the available bandwidth without actually delivering any payload, hence we excluded it from our computations for both TCP and BFP.

Figures D.10 and D.11 show the average normalized goodput for BFP over OTN and TCP respectively.

$$\overline{G} = \frac{P}{C} * 100 \tag{D.4}$$

where:

P Payload bits successfully delivered / Second

C Link Capacity

When using BFP, transaction sizes $> 10MB$ progressively fills the available bandwidth, and goodput increases linearly up to roughly 99% (reached when the number of sources equals the TD). This is the result of scheduled interleaving of chained data, with CP blocking occurring only when normalized goodput is close to its maximum. Beyond this point, goodput remains above 70% for all transaction sizes considered, showing the capability of BFP to provide high goodput even when the requested bandwidth exceed the link capacity and transmission requests (CP) start being blocked.

Note that with BFP, small transactions ($\leq 10MB$) can achieve high link utilization due to interleaving of flows coming from different sources. However, given the increased load on the BFP control plane, we do not advocate BFP for small transactions (e.g. $< 500kB$).

Figure D.11 shows how TCP sources fill the required bandwidth to the bottleneck router, resulting in higher goodput with respect to BFP for long lived TCP flows ($\geq 300MB$) as long as the number of sources remains limited (≤ 3). Beyond this point, TCP is prone to packet drops which trigger the TCP backoff algorithm, dramatically reducing the offered load and resulting in poor link utilization. For shorter flows ($\leq 100MB$), TCP either does not ramp up fast enough to fill the available bandwidth or, as the number of sources increases, the random backoff mechanism kicks in, significantly reducing the goodput, as in the case of longer lived flows.

An issue occurring when multiple TCP sources compete for the same resources is known as *TCP incast* [38,39], common in data center scenarios, and it leads to throughput collapse. While the collapse of the throughput was observed in the TCP case for all transaction sizes $\geq 100MB$ (See Figure D.11), it did not occur in the BFP case.

Figure D.12 shows how BFP compares with TCP Westwood in terms of goodput. Transactions varying in size from $100MB$ to $1GB$ using both BFP and TCP Westwood are shown. The rapid goodput collapse observed in TCP as the number of sources increase suggests a significant advantage relative to the amount of offered load that BFP can handle with respect to TCP. BFP can accommodate over 40% more load with respect to TCP without incurring goodput collapse (See shaded area of Figure D.12).

V-B Delay

Figures D.13 and D.14 show the delay performances of BFP and TCP, as the average time needed to successfully complete one transaction, including connection setup and any additional delay coming from data retransmission (i.e. BFP retrials due to CP blocking -including connection (re)setup- and TCP packet retransmissions). A new connection is setup for each new transaction in both the TCP and BFP case.

Figure D.14 shows that, for transactions $\geq 500MB$ and up to 2 sources, TCP tends to fill the available bandwidth achieving smaller delays. As the number of sources increases, TCP goodput collapses (Figure D.11) and end-to-end delay rapidly increases to values up to over 30 times larger than BFP, as can be seen from Figure D.15. Similar TCP performance is observed for all other transaction sizes (smaller). In these cases BFP achieves better delay performance regardless of the number of competing sources.

The rapid degradation of the delay performance of TCP is due to the high packet drop rate (triggering the random backoff algorithm) occurring when multiple sources

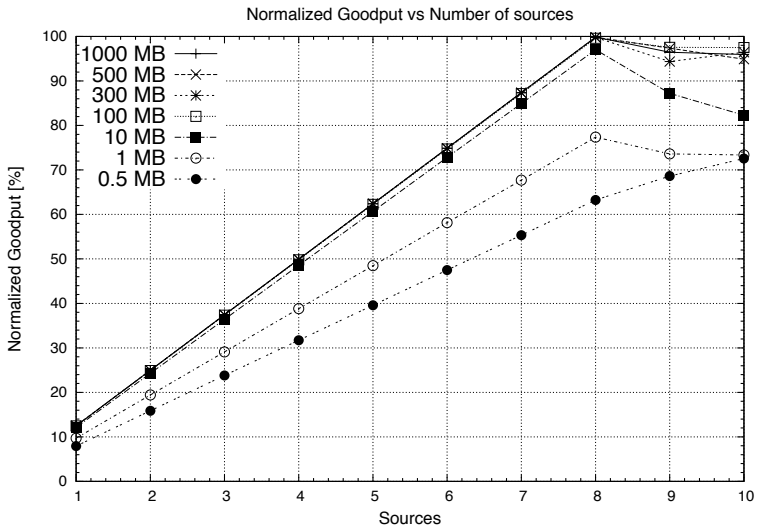


Figure D.10: Normalized Goodput for BFP over OTN (Pareto).

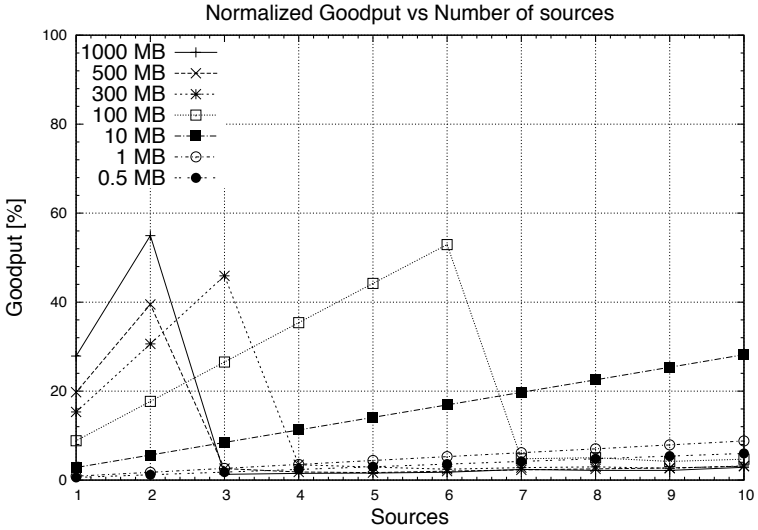


Figure D.11: Normalized Goodput for TCP (Pareto).

compete for the same resources. Another factor negatively affecting TCP delay performance, is the large number of acknowledgments (ACK) used by TCP. Although for the topology studied the Round Trip Time (RTT) is relatively small, the effect of

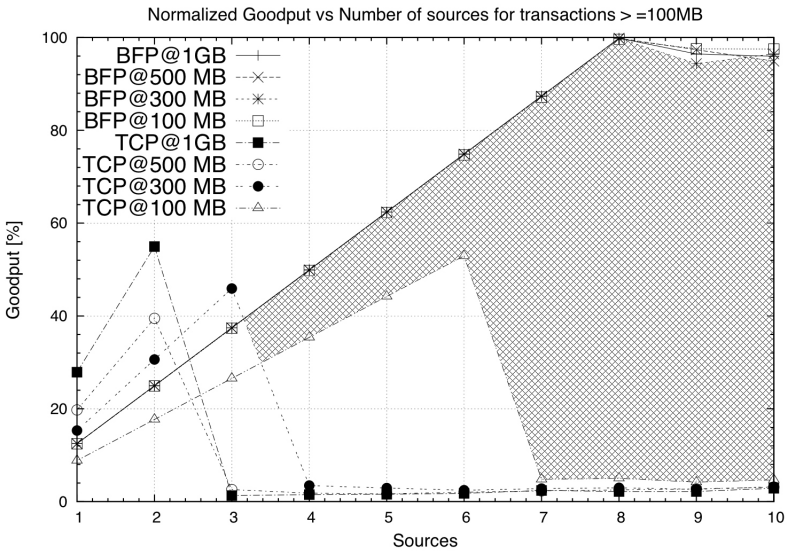


Figure D.12: Normalized Goodput Comparison for Transaction Size $\leq 100MB$.

$ACKs$ transmission has a heavy influence on TCP delay (each ACK will add to the overall delay an amount roughly proportional to $RTT/2$). Obviously this will further degrade TCP delay performance when larger network diameters are considered. Similar considerations also apply to BFP; however, due to the smaller amount of control information exchanged between source and destination, the resulting increase in delay would be much smaller.

The periodic configuration of data frames in each BFP transaction ensures that once a connection setup is successful the delay between the first and last bit of an entire transaction is constant. This result in a much smaller variation of the end-to-end delay (See Figure D.15) of BFP with respect to TCP, making BFP particularly suited for delay-sensitive applications (e.g. video streaming).

VI. DISCUSSION

Delayed periodic reservation is not a new idea [40,41,42,43]. However, to our knowledge, the BFP resource reservation algorithm and its relative buffering technique have not been proposed before in this context.

The particular resource reservation algorithm also differentiates BFP from OFS [5], which can be seen as a particular case of BFP where only chains with $TD = 0$ are used, and the underlying infrastructure is all-optical. In [44] the higher efficiency

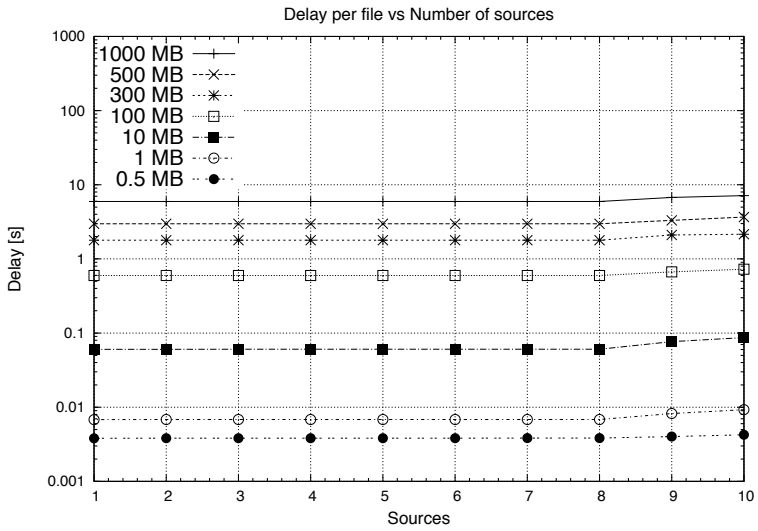


Figure D.13: Delay per transaction for BFP over OTN (Pareto).

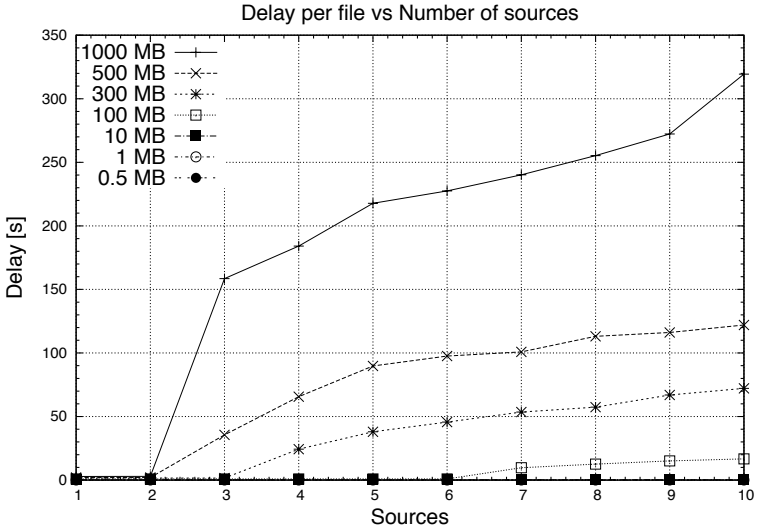


Figure D.14: Delay per transaction for TCP (Pareto).

in terms of link utilization of the BFP resource reservation algorithm with respect to that used in OFS was demonstrated.

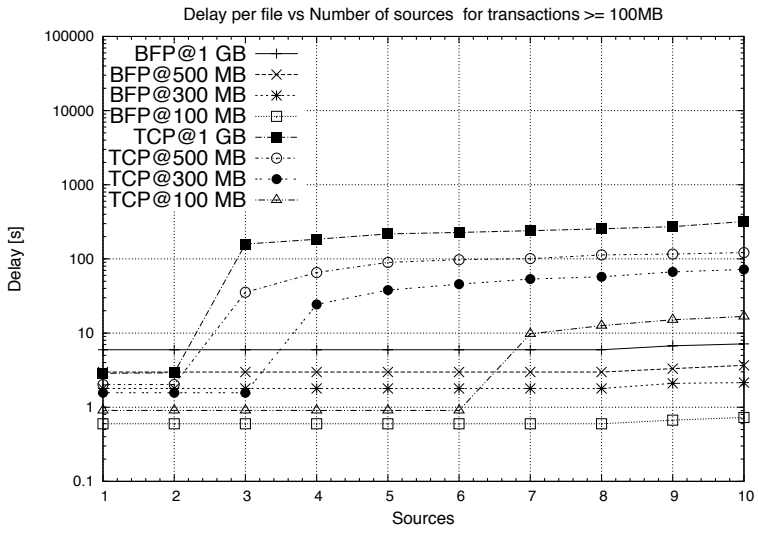


Figure D.15: Delay per transaction Comparison for Transaction Sizes $\geq 100MB$.

Another key difference between BFP and the aforementioned approaches employing delayed periodic reservation is that [40,41,42,43] are all variations of OBS, and require some form of replacement/redesign of the currently deployed transport architecture.

BFP targets large file transfers specifically and does not aim at replacing any of the protocols or architectures currently in place. Our intent is to complement the current infrastructure with a method able to more efficiently handle large file transactions, leveraging currently available functions and minimizing deployment impact.

Another potential weakness of OBS-based approaches in terms of their overall energy efficiency is that burst assembly procedures at the edge of the network may nullify the advantages gained in the core by using large frames [45]. In contrast, BFP does not perform any burst assembly; large files at a source location are simply sliced and packaged, using *whichever* frame size is available, into chains that are then scheduled for transmission and handled by the current architecture as a standard data flow with particular buffering requirements.

BFP also shares some similarities with pipeline forwarding (PF) [46], however most of the above considerations on OBS/OFS also apply to PF. In addition: (1) BFP scheduling acts on complete transactions, not on individual packets. (2) BFP traffic does not coexist with standard TCP/UDP traffic, so no preemption mechanism

of BFP over standard traffic is needed and (3) similarly to PF, in BFP switching is also based on timing. However, in contrast with BFP, the time "reference" (i.e. CP arrival time) is different for each node and has only a local significance, freeing the nodes from the need to coordinate and synchronize with each other. Data synchronization for BFP is achieved by control plane operations and not derived from a common source nor through the use of distributed synchronization protocols. Lastly, (4) partial packet buffering is allowed in BFP resulting in lower queuing delays.

BFP was designed to provide most of the advantages offered by optical layer solutions (high link utilization, low power consumption and efficient resource usage) without requiring major hardware redeployment or architectural redesign of the currently deployed infrastructure. In order to provide ease of integration within the current architecture of BFP functionalities we aim at reusing as much as possible currently available functions, from routing protocols to framing procedures.

Basic functions required to support BFP are extremely simple and ultimately boil down to the ability to store and forward data frames and execute the proposed scheduling algorithm. Thus the only major change required is within the control plane. Given the plethora of protocols and functions currently defined within the network infrastructure, this update can be risky. In this respect, integrating BFP as a LKM will minimize the risks and allow easy evolution of BFP.

Simulations showed that BFP can achieve significantly better performance than TCP in terms of *Goodput*, *Delay* and *average buffer space*. For our specific simulation parameters, BFP can accommodate over 40% more load than TCP without incurring throughput collapse (TCP incast), and delay per transaction can be over 30 times smaller while only using a fraction of the buffer space used by TCP [12].

Although BFP can reuse any available routing protocol (See Section II-B), routing and load balancing protocols specifically designed for BFP traffic can be designed using information about each chain to make load-balancing and routing decisions that take into account the configuration of each chain, expressed as its average bandwidth occupancy (Equation D.5).

$$\overline{B}_c = \frac{f_b * n}{f_s * [TD * (n-1) + 1]} \tag{D.5}$$

where:

f_b size (in bits) of a data frame

n number of data frames per chain

f_s duration (in seconds) of a data frame at the (output) line rate

TD Transparency Degreee of the chain

VII. CONCLUSION

While traditional protocols such as TCP or UDP may struggle to scale in the new networking scenario dominated by large transactions (often with stringent delay requirements), BFP proposes an alternative method, improving on known traffic shaping [47] and pacing [48] techniques.

Most BFP traffic is handled at lower layers, where the cost per bit is lower, while higher layers are accessed only when strictly necessary (e.g. for routing purposes). As a result, most of the computations performed by TCP on a per-packet basis (e.g. header processing, address lookup, etc.) are performed on a per-flow basis by BFP, significantly reducing the computational load placed on the network hardware by large transactions.

While many optical layer approaches have been studied and show the potential to significantly improve performance with respect to current networks, most require major architectural redesign, hardware redeployment, and have not been embraced by industry.

BFP offers similar advantages of optical layer solutions in terms of performance (high goodput, low latency and efficient resource usage) leveraging on currently available network functionalities, potentially offering a less disruptive evolutionary path for both IP and DC networks.

An in-depth study of the topology and routing-related aspects of BFP, considering more complex simulation scenarios for both IP and DC networks is currently under way and will be included in our future work. The tolerance of BFP to errors in the timing estimates will also be addressed. Lastly, using OpenFlow-enabled hardware [49,50] it will be possible to run BFP tests over real-world networks.

References:

1. *Cisco Visual Networking Index: Forecast and Methodology, 20122017.* CISCO white paper. Retrieved on October 20 2013 from:`http://www.cisco.com/en/US/\\solutions/collateral/ns341/ns525/ns537/ns705/ns827/white\_paper\_c11\-481360.pdf`

2. Tucker, RodneyS.; Parthiban, R.; Baliga, J.; Hinton, K.; Ayre, R.W.A; Sorin, W.V., *Evolution of WDM Optical IP Networks: A Cost and Energy Perspective.* Lightwave Technology, Journal of , vol.27, no.3, pp.243,252, Feb.1, 2009

3. J. P . Jue,V . M. V okkarane, Optical Burst Switched Networks, Ed. Springer 2005.

4. Chandra, P.K.; Turuk, A.K.; Sahoo, B., "Survey on optical burst switching in WDM networks," Industrial and Information Systems (ICIIS), 2009 International Conference on , vol., no., pp.83,88, 28-31 Dec. 2009

5. Weichenberg, G.; Chan, V. W S; Medard, M., "Design and Analysis of Optical Flow-Switched Networks," Optical Communications and Networking, IEEE/OSA Journal of , vol.1, no.3, pp.B81,B97, August 2009

6. Yunfeng Peng; Kunjun Jiang, "On user-participated and service-oriented optical networks," Optical Communications and Networks (ICOCN 2010), 9th International Conference on , vol., no., pp.134,138, 24-27 Oct. 2010

7. Grasa, E.; Junyent, G.; Figuerola, S.; Lopez, A.; Savoie, M., "UCLPv2: a network virtualization framework built on web services [web services in telecommunications, part II]," Communications Magazine, IEEE , vol.46, no.3, pp.126,134, March 2008

8. Chiu, A.L.; Choudhury, G.; Clapp, G.; Doverspike, R.; Feuer, M.; Gannett, J.W.; Jackel, J.; Kim, G.; Klincewicz, J.; Kwon, T.; Guangzhi Li; Magill, P.; Simmons, Jane M.; Skoog, R.A.; Strand, J.; Lehmen, A.; Wilson, B.J.; Woodward, S.L.; Dahai Xu, "Architectures and Protocols for Capacity Efficient, Highly Dynamic and Highly Resilient Core Networks [Invited]," Optical Communications and Networking, IEEE/OSA Journal of , vol.4, no.1, pp.1,14, January 2012

9. Ajay Mahimkar, Angela Chiu, Robert Doverspike, Mark D. Feuer, Peter Magill, Emmanuil Mavrogiorgis, Jorge Pastor, Sheryl L. Woodward, and Jennifer Yates. 2011. Bandwidth on demand for inter-data center communication. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages.

10. Mahimkar, A.; Chiu, A.; Doverspike, R.; Feuer, M.D.; Magill, P.; Mavrogiorgis, E.; Pastor, J.; Sethi, V.; Dahai Xu; Woodward, S.; Yates, J., "OutageDetection and dynamic re-provisioning in GRIPhoN A globally reconfigurable intelligent photonic network," Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference , vol., no., pp.1,3, 4-8 March 2012

11. High-Capacity Optical Transport Networks, Tiejun J. Xia, Steven Gringeri and Masahito Tomizawa, IEEE Communication magazine, 2012

12. Ilijc Albanese, Yağız Onat Yazır, Stephen W. Neville, Sudhakar Ganti and Thomas E. Darcie *Big File Protocol (BFP): a Traffic Shaping Approach for Efficient Transport of Large Files*. HPSR 2014

13. ITU-T G.709 (02/2012) *Interfaces for the optical transport network*

14. ITU-T G.872 (10/2012) *Architecture of optical transport networks*

15. Steve Gorshe. *A Tutorial on ITU-T G.709 Optical Transport Networks (OTN)*. PMCSierra white paper, *personal communication*.

16. Virginia Hutcheon. *OTN to Enable Flexible Networks*. OSA/OFC/NFOEC 2011

17. Ashwin Gumaste, Nalini Krishnaswamy. *Proliferation of the optical transport network: a use case based study*. IEEE Communication magazine, 2010

18. K. Hinton, J. Baliga, Feng M.Z., R.W.A. Ayre, R.S. Tucker, *"Power Consumption and Energy Efficiency in the Internet"*, IEEE Network, Vol. 25, March-April 2011, Issue 2, pp. 6-12

19. Divakaran, D.M.; Altman, E.; Post, G.; Noirie, L.; Primet, P.V.-B., "From Packets to XLFrames: Sand and Rocks for Transfer of Mice and Elephants," INFOCOM Workshops 2009, IEEE , vol., no., pp.1,6, 19-25 April 2009

20. ITU-T G.7044 (10/2011). *Hitless adjustment of ODUflex(GFP)*.

21. ITU-T G.7041 (04/2011). *Generic Framing Procedure*.

22. IEEE ETHERNET, IEEE Standard 802.3, 2012

23. Dennis Abts and Bob Felderman. *A guided tour of data-center networking.* Commun. ACM 55, 6 (June 2012), 44-51.

24. Reid, A.; Willis, P.; Hawkins, I.; Bilton, C., "Carrier ethernet," Communications Magazine, IEEE , vol.46, no.9, pp.96,103, September 2008

25. *Connection-Oriented Ethernet Completing the Ethernet Revolution.* Fujitsu white paper. Retrieved on October 20 2013 from: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/ConnectionOrientedEthernetRevolution.pdf

26. *Connection-Oriented Ethernet, On-Ramp Aggregation for Next-Generation Networks.* Fijitsu white paper. Retrieved on October 20 2013 from: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/COE.pdf

27. IEEE 802.1Qay Provider Backbone Bridging Traffic Engineering Standard, Sept. 2009.

28. Mark Orthodoxou. *Solutions for the New Packet Optical Transport Network.* PMC-Sierra White Paper. Retrieved on February 10 2014 from: `https://pmcs.com/myPMC/download.html?res_id=99671\&filename=\\2110752\_OTN\_Convergence\_White\_Paper\_099671.pdf`

29. PMC-Sierra PM5450 HyPHY 20Gflex. Product Datasheet. Retrieved on March 4 2014 from: `http://pmcs.com/products/optical\_networking/otn/\\multi-service\_otn\_processors/pm5450/`

30. Nokia-Siemens hiT 7300. *Product Datasheet.* Retrieved on March 1 2014 from: `http://nsn.com/system/files/document/optical\_transport\_hit7300.pdf`

31. Hitachi AMN6400. *Product Datasheet.* Retrieved on March 3 2014 from: `http://www.hitel.com/pdf/transport/amn6400\_datasheet.pdf`

32. Tejas Networks TJ1600 POTP. *Product Datasheet.* Retrieved on March 1 2014 from: `http://www.tejasnetworks.com/products\_mspp\_TJ1600.html`

33. Broadcom BCM88650 *Data Sheet.*
Available at: http://www.broadcom.com/products/Switching/Carrier-and-Service-Provider/BCM88650-Series.

34. Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. *TCP westwood: end-to-end congestion control for wired/wireless networks.* Wireless Networks 8, 5 (September 2002), 467-479.

35. Omnet++, Discrete Event Simulation System. Available: http://www.omnetpp.org/

36. INet Framework for Omnet++. Available: http://inet.omnetpp.org/

37. Floyd, Sally; Jacobson, V., *Random early detection gateways for congestion avoidance.* Networking, IEEE/ACM Transactions on , vol.1, no.4, pp.397,413, Aug 1993

38. Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. *Understanding TCP incast throughput collapse in datacenter networks.* In Proceedings of the 1st ACM workshop on Research on enterprise networking (WREN '09). ACM, New York, NY, USA, 73-82.

39. Yan Zhang; Ansari, N., *On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers.* Communications Surveys and Tutorials, IEEE , vol.15, no.1, pp.39,64, First Quarter 2013

40. Sheeshia, S.; Chunming Qiao, "Synchronous optical burst switching," Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on , vol., no., pp.4,13, 25-29 Oct. 2004

41. Yu, O.; Ming Liao, "Synchronous stream optical burst switching," Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on , vol., no., pp.1447,1452 Vol. 2, 7-7 Oct. 2005

42. Hernandez, J.A.; Reviriego, P.; Garcia-Dorado, J.L.; Lopez, V.; Larrabeiti, D.; Aracil, J., "Performance Evaluation and Design of Polymorphous OBS Networks With Guaranteed TDM Services," Lightwave Technology, Journal of , vol.27, no.13, pp.2495,2505, July1, 2009

43. Chunming Qiao; Wei Wei; Xin Liu, "Extending generalized multiprotocol label switching (GMPLS) for polymorphous, agile, and transparent optical networks (PATON)," Communications Magazine, IEEE , vol.44, no.12, pp.104,114, Dec. 2006

44. Ilijc Albanese ; Thomas E. Darcie ; Sudhakar Ganti; *Electronic implementation of optical burst switching techniques*. Proc. SPIE 8915, Photonics North 2013, 89150B (October 11, 2013);

45. Shuping Peng; Hinton, K.; Baliga, J.; Tucker, RodneyS.; Zhengbin Li; Anshi Xu, "Burst switching for energy efficiency in optical networks," Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC) , vol., no., pp.1,3, 21-25 March 2010

46. Mario Baldi, Guido Marchetto: Pipeline forwarding of packets based on a low-accuracy network-distributed common time reference. IEEE/ACM Trans. Netw. 17(6): 1936-1949 (2009).

47. Marcon, M.; Dischinger, M.; Gummadi, K.P.; Vahdat, A. *The local and global effects of traffic shaping in the internet*. Communication Systems and Networks (COMSNETS), 2011 Third International Conference on , vol., no., pp.1,10, 4-8 Jan. 2011

48. Aggarwal, A.; Savage, S.; Anderson, T., "Understanding the performance of TCP pacing," INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , vol.3, no., pp.1157,1165 vol.3, 26-30 Mar 2000

49. Lara, A.; Kolasani, A.; Ramamurthy, B., "Network Innovation using OpenFlow: A Survey," Communications Surveys & Tutorials, IEEE , vol.16, no.1, pp.493,512, First Quarter 2014

50. Das, S.; Parulkar, G.; McKeown, N., "Rethinking IP core networks," Optical Communications and Networking, IEEE/OSA Journal of , vol.5, no.12, pp.1431,1442, Dec. 2013

Appendix E

Big File Protocol (BFP) for efficient quasi-deterministic data transfer in data centers

Ilijc Albanese*, Yağız Onat Yazır†, Stephen W. Neville*, Sudhakar Ganti† and Thomas E. Darcie, *Fellow*, IEEE*
*Department of Electrical and Computer Engineering
†Department of Computer Science
University of Victoria, Victoria, Canada.

[Submitted to IEEE Transactios on Cloud Computing on November 24 2014. Pending Review.]

***Abstract*:** Performance variability in data center networks is a major concern, affecting both tenants and operators. Although TCP and its variants are extensively used in data center environments, they nonetheless still suffer from high performance variability and sub-optimal network goodput.

Recently proposed Big File Protocol (BFP) offered significant advantages over legacy data transfer protocols. In this paper we adapt BFP for data center environments and show its ability to deliver superior performance in terms of network variability, goodput, and delay. BFP is largely independent of the underlying network topology and can deliver the same performance over a wide variety of data center topologies and architectures. BFP can integrate with many existing routing and load balancing approaches, offering a versatile, highly-efficient mean to handle bandwidth-intensive applications in data centers.

BFP is able to achieve goodput efficiency within 0.1% of the maximum capacity achievable, with fairness index equal to the theoretical maximum (Jain's fairness index, $J = 1$), and standard deviations below $0.0002Mb/s$

I. INTRODUCTION

In recent years significant investments have gone into expanding and improving data center architectures in order to continue supporting a plethora of hosted services (e.g. web search, scientific computation, distributed file systems, map-reduce and so forth). Data Centers (DC) play increasingly important roles in many sectors of the economy. And as their size and complexity grow, issues related to their sustainability, performance and profitability become more critical [1].

Although only responsible for a small portion of the overall cost of a DC [1], the data center network (DCN) plays a key role in determining the DC performance [2,3]. Many services offered by DCs (e.g. map-reduce [4]) exhibit a high degree of thread-level parallelism, and rely heavily on the network infrastructure. The network infrastructure of the data center can become a serious DC performance bottleneck, significantly impacting both tenants and service providers [2,3,5,6,7].

In order to improve user experience and keep DCs profitable for providers, a data center needs to be able to provide stable and predictable network performance for unpredictable traffic patterns and workloads [2,5,8,9].

Many of the solutions proposed for DCNs [10,11] redesign the DC architecture, creating topology or applications-specific solutions, or requiring some form of hardware modification. In contrast, the aim of this paper is to provide DCNs with a data transmission protocol which is largely insensitive to the underlying network topologies, requires no hardware redeployment, is able to provide quasi-deterministic network performance with goodput efficiency [8] within 0.1% of the maximum capacity, and is able to achieve ideal fairness. Our approach is to overlay an efficient data transfer protocol onto existing DCN architectures to support data-intensive applications (e.g. map-reduce or CDN content update/distribution), giving the DC the means to efficiently utilize the available bandwidth. Our approach is therefore able to work in concert with a wide range of proposed DC architectures, as well as existing routing and load balancing solutions (e.g. [5,8,12,13]).

The rest of this paper is organized as follows: Section II provides a quick overview of the current issues and state-of-the-art in DCNs, Section III describes the proposed approach, and addresses questions relative to its integration within a DCN; Section IV presents the performance of BFP over various DCN topologies. Advantages of

using BFP in a DC, and potential usage scenarios for BFP are discussed in Section V.

II. BACKGROUND

A DC is usually built using a large number of commodity-off-the-shelf (COTS) components interconnected by a high-capacity network providing communications services between servers, which constitute the compute resources of the DC [14,15]. Servers are packaged into racks containing typically between 20 and 40 servers each [8]. Each rack is equipped with a top-of-rack switch (ToR) connecting all the servers to the aggregation layer of the DCN. Typical DCNs, such as that described in [15], implement a two- or three- tier DCN infrastructure depending on the size of the DC. Smaller DCs usually opt for a two-tier infrastructure (capable of supporting several thousands of hosts). Larger data centers, hosting hundreds of thousands of machines, tend to use three-tier architectures [16,17].

There are several different popular DC architectures depending on the size and purpose of the DC. We refer to [10] and [11] for a more complete overview of deployed and proposed DC architectures.

As mentioned in Section I, the DC network infrastructure should provide stable and predictable performance in order to guarantee good user experience. Studies on DC traffic characteristics [8,18] show that DC traffic is unpredictable, and it is strongly dependent on the type of DC. For example, cloud DC, map-reduce clusters, production DC, and so forth, are known to exhibit different workloads and traffic patterns. In this scenario it can be challenging to provide predictable network performance if the underlying data transfer dynamics remain unchanged [2]. A sizable number of research efforts have focused exactly on this aspect [19,20,21,22,23,24,25,26].[1]

In terms of the flow distributions in DCs, several studies seem to agree on the fact that although most flows are small, the vast majority of bits belong to large flows [2,8,18] . TCP [27], which in some DCs handles over 99% of traffic [19] was simply not designed to handle these kind of transactions [28].

Another key issue in DCNs is the variability of network performance [9]. This variability can have severe consequences on the overall performance of the DC [3,5,6,7]. As an example, consider a typical DC application such as map-reduce, which relies heavily on the network to distribute jobs among servers. The servers then carry

[1]Clearly, a limited number of DC scenarios also exists where by all of the incoming workloads are themselves highly predictable and stable. Such DCs do not represent the typical or common case. This work focused on the more typical scenarios whereby well-behaved DCN traffic does not simply arise by default.

out their computations and return a response to the dispatching entity. The overall response time, therefore depends on the job completion time for each machine, as well as the time taken by the slowest server to deliver its results. Even if just one machine experiences network congestion, the whole computation can be slowed or generate less precise results (e.g. web-indexing, web-search). Since DC tenants are often billed based on the time they occupy the DC resources, high variability in DCN performance can result in "hidden costs" for the tenants [5], who may then elect to change providers resulting in revenue losses for the DC operator [29,30].

The ability to assign any service to any server in a DC is referred to as *agility* [2,8]. This key characteristic plays an important role in achieving high utilization of the computational resources, allowing DC operators to meet unexpected workload demands. Agility also eases VM migration, an important feature when it comes to consolidating workloads, avoiding resource fragmentation, and efficiently utilizing DC resources [31]. The lack of agility in current DCNs is due to multiple factors [8] including: (1) the limited server-to-server bandwidth available in common DCN architectures, (2) the high oversubscription ratio often found in DCNs, and (3) the inability of VMs to migrate without changing their current IP address, which may result in the breaking of active connections if the VM is migrated to a different physical machine. Another important problem in DCNs is TCP incast [21], as it can lead to throughput collapse which further degrades the DC network performance and variability [32].

In order to improve DC network performance a wide range of architectures have been proposed and investigated [10,11]. Among them, VL2 [8] was proposed as a solution to improve fault tolerance, oversubsrciption, and DCN agility while providing stable network performance and high bandwidth between servers. VL2 exploits a topology similar to the 3-tier DCN architecture presented in [15] but with a Clos topology between core and aggregation layers, ensuring a rich connectivity at the higher layers and improving path diversity with respect to [15]. A load balancing algorithm based on Valiant Load Balancing (VLB) [33] is used to evenly balance traffic flows over a richly connected network topology. VL2 also uses a flat addressing scheme that separates server names from their physical location, providing agility without the drawback of interrupting ongoing connections during migrations. The mapping between the servers names and their physical location is handled by a centralized directory server. A drawback of VL2 is its reliance on a richly connected topology [12], with associated potential costs and operating issues.

SPAIN is a topology-independent approach presented in [12]. This approach only requires minor software modifications at the end-hosts and, like VL2, can be implemented using COTS components. In SPAIN, an offline network controller precomputes a set of paths which are merged into a set of trees, covering the entire DCN physical topology. SPAIN then exploits the VLAN support of COTS switches to map each tree onto a separate VLAN, achieving multipath forwarding without being tied to a specific physical DCN topology. The SPAIN control and data plane functionalities are implemented on linux end-hosts via a Loadable Kernel Module (i.e. a driver) and a user-level controller. The driver is in charge of host initialization and re-initialization whenever the host migrates to a new ToR as well as being in charge of sending and receiving packets according to the SPAIN algorithm. The user-level controller is in charge of determining the MAC address of the network interface, the ID of the edge switch, as well as contacting a centralized repository with all the precomputed paths and downloading the DCN map (precomputed offline). A "SPAIN virtual Ethernet device" is created and configured by the controller. This virtual device position itself as a master with respect to the physical Ethernet device, intercepts all incoming packets and routes them toward the SPAIN virtual device. Although SPAIN brings the advantage of implementing multipath forwarding for arbitrary topologies using currently deployed hardware, it does not address the network performance variability issue. Furthermore, it assumes that all sources get a fair share of the available bandwidth, which may not be straightforward to achieve in practice [8,19].

As discussed later in more detail, the Big File Protocol (BFP) [28] was developed as an efficient transport protocol for bandwidth-intensive applications. BFP can outperform standard TCP in terms of delay, goodput, buffer space, and delay variation for such high volume traffic.

In this work we adapt BFP [28] to work in data center environments as a generic data transfer protocol for network-intensive applications, and show how BFP is able to work in concert with different DCN architectures (e.g. [5,8,12,13,15]), providing them with a simple data transfer protocol,that is able to provide superior performance in terms of transmission delay and goodput, while also providing stable and predictable network performance[2].

III. DATA CENTER BFP

BFP [28] is a cross-layer transport protocol designed to handle data-intensive transactions through preferentially exploiting lower layer services.

[2]By which we mean that BFP is a (statistically) *well-behaved* system.

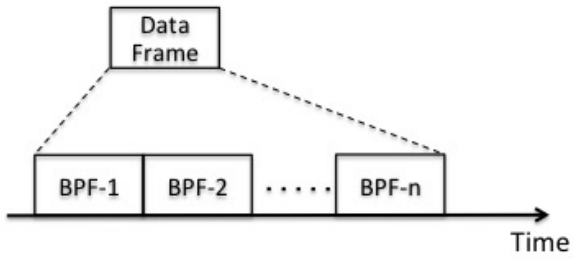


Figure E.1: Data frame assembled with multiple atomic data frames (BPF)

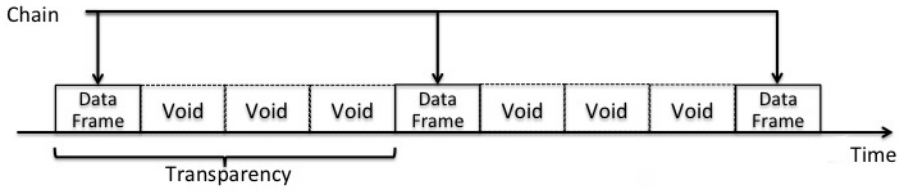


Figure E.2: BFP chain with length $L_{Ch} = 3$ and $TD = 4$

BFP-enabled sources shape traffic coming from large transactions into a set of concatenated fixed-size data frames. These data frames can be sized to fit the frame structure of the underlying lower layer transport technology (e.g. OTN [34] or Ethernet [35]), simplifying the mapping of BFP onto the available underlying transport architecture [28,36]. The data frame assembly mechanism [28] allows for the use of frames of virtually any size by assembling each data frame as a sequence of atomic data frames transmitted back to back (Figure E.1). Such atomic data frames are referred to as Basic Payload Frames (BPF), their size depends on the underlying transport technology.

In BFP, the available bandwidth of each link is modeled as series of data frame sized timeslots. The underlying hardware is not required to implement/support TDM, as the timeslots are only used by the control plane to model the available bandwidth and schedule network resources for incoming transactions [28]. At transmission time, data frames are interspersed with a fixed number of *void frames*, which have the same size and structure of data frames but only carry stuffing bits. Void frames are always preempted by data frames and are never processed, as they are only used to intersperse BFP data frames. Combining data and void frames, large transactions are transmitted as periodical, semi-transparent transport entities referred to as *chains*.

Using chains, the configuration of which is stable and well defined, enables the handling of large transactions at the lowest possible network layer (e.g. L1 for OTN-

mapped BFP or L2 for Ethernet-mapped BFP), skipping most of the per-packet/per-frame processing required by other data transfer protocols (such as TCP, where each packet must be processed individually, as it crosses several layers of the network stack at each node), significantly reducing the computational load placed on the network hardware by bandwidth-intensive transactions [28]. The configuration of each chain is completely defined by: (1) its *frame size*, i.e. the number of BPFs in each data frame, (2) its period, referred to as the *Transparency Degree* (or TD) of the chain, and (3) its length (L_{Ch}) in terms of number of data frames per chain. A BPF chain with $TD = 4$ and $L_{Ch} = 3$ is shown in Figure E.2. A Control Packet (CP) travels node by node and performs delayed, end-to-end resource reservation for each chain. For a generic node "i" in the data path, the reservation starts at the expected time of arrival (relative to that node, ETA_i) of the first bit of the chain [28].

If the BFP resource reservation procedure (Section III-B) is successful the source is notified and chain transmission can start. At each node, buffering is used to align in time the frames of incoming chains and interleave chains competing for the same links, allowing data plane collisions and frame drops to be avoided. The only entity that can be blocked is the CP during the resource reservation phase. When a chain leaves its source, ordered delivery of all its data frames is guaranteed. Such a reservation-based approach is reasonable given BFP's focus on high volume data transmissions.

The mapping of BFP chains onto the underlying transport layer is discussed in detail in [28,36].

When organized into BFP chains, the average bandwidth occupation of each transaction is determined solely by the chain configuration (Equation E.1).

$$\overline{B}_c = \frac{f_b * L_{Ch}}{f_s * [TD * (L_{Ch} - 1) + 1]} \tag{E.1}$$

where:

f_b size (in bits) of a data frame

L_{Ch} number of data frames per chain (i.e. chain *length*)

f_s duration (in seconds) of a data frame at the (output) line rate

TD Transparency Degree

III-A DC-BFP Deployment Strategy

The integration of BFP in a DC environment would be consistent with the current use of large numbers of low-cost commodity servers. Large-scale hardware redeployment would be impractical; Hence a BFP-enabled DC is achieved via software modifications at the control plane and OS level are required. As a sample integration strategy, we consider a network where a BFP network module residing at the ToR is in charge of coordinating BFP traffic for the rack it services. We assume that from the ToR upwards all network equipment supports OTN. Note that this is not a limitation as BFP can also be implemented using Ethernet only [28]. Integration of BFP within the OS network stack and mapping strategies for BFP chains onto the underlying transport layer are discussed in [28]. Here we will discuss the various DC-specific functions needed at each network element (i.e. end-hosts and switches/routers) in order to support BFP. Our discussion is framed by the schematic of Figure E.3, showing a rack unit of servers, each with a BFP software module installed (*BFP Server Software Module* or BSSM) and a ToR switch with a BFP network module (*Ethernet to OTN Bridge* or EOB). This module could be implemented in hardware (e.g. a PCIe network processor [37]) or as a software/firmware upgrade to currently deployed ToR components. An EOB is also assumed to be present in each of the higher layers switch/router supporting BFP transmission. In what follows, the functionalities required at each network element to support BFP are described in some detail, assuming servers to be linux hosts.

1) End-hosts. To support BFP, end-hosts are in charge of the following: organize large files into BFP frames, process the BFP control traffic (i.e. CP, $ACKs$, and $NACKs$) and provide (periodic) access to the communication channel for applications using BFP.

An application (i.e. a BSSM) can be installed at the end-host machine, providing application-level access to BFP transport services. A BSSM provides BFP transport services on a file by file basis (e.g. explicit user selection), automatically (e.g. for file sizes $\geq 100MB$), or by application type (i.e. given the benefits offered by BFP -See Section IV- , applications like map-reduce may be setup to use BFP by default). A discussion on a BFP API allowing application programmers to access BFP transport services is described in detail in [28].

Similarly to the integration approach presented in [12] a LKM (i.e. a "driver") can be installed at the end-host machine to handle transmission and reception of BFP data traffic with the required periodicity, and to process BFP control traffic.

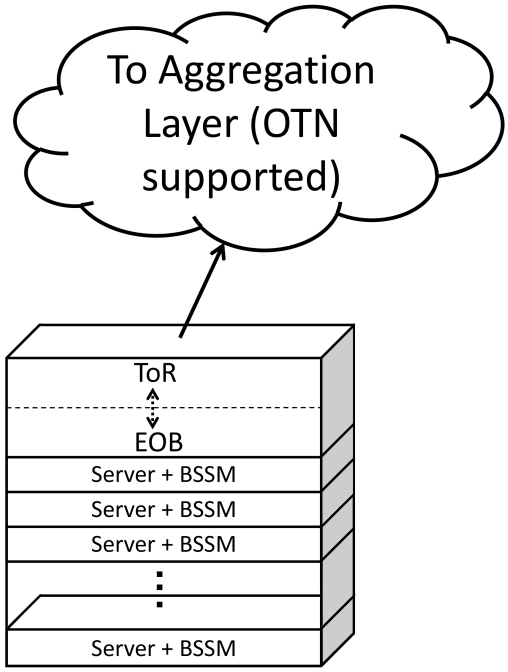


Figure E.3: BFP-enabled DC rack

At transmission time, the driver can intersperse data frames with frames containing stuffing bits (i.e void frames), effectively transmitting periodic data frames sequences (traffic shaping). Deviation from perfect periodicity will be adjusted at the ToR using buffering, while potential data frame collisions can be avoided using guard time. The size of this guard time depends on the achievable timing precision at the end-hosts. In this respect note that in the case of Ethernet-mapped BFP, this guard time is naturally provided by the Inter Frame Gap (IFG), which can be decreased as the line rate increases.

Achieving precise periodic frame transmission for each data flow (chain) at a end-host could be challenging in a virtualized environment where multiple VMs can coexist in the same physical machine. Interrupt reaction times may represent a limitation when it comes to transmitting periodic data frames with the required accuracy (i.e. with deviations from perfect periodicity that can be recovered by buffering at the ToR switch). This issue is mitigated in devices supporting single-root input-output virtualization (SR-IOV) [38], and in natively shareable devices, which are becoming more and more common [39]. In devices where SR-IOV is enabled, upgrading the Virtual Machine Manager (VMM) software can force the Master Driver (MD) to

assign to a VM using BFP transport services, a virtual function able to meet a specific level of service, e.g. guaranteeing 1Gb/s Ethernet performance [39].

The MD, which is specific to a VMM and is responsible for configuring shared resources between various VMs, has higher privileges with respect to the VM drivers and can perform changes that have a global effect, effectively granting a specific level of service to VMs as well as coordinating the network access of all the VMs in a controllable fashion, giving to all the VMs the "illusion" of a dedicated NIC. Network throughput of SR-IOV devices in a virtualized environment is very close to that of non-virtualized ones [38].

Another functionality of the MD is to configure the L2 sorter, switch and classifier, providing additional control on the transmission equipment, and helping reduce timing uncertainties coming from the interrupt delay of the individual VMs. Furthermore, these MD functions can be used to enforce periodic transmissions of BFP data frames coming from different VMs by selecting them in a round robin fashion and providing periodicity to each flow by alternating data frames from different sources and filling in the "blanks" with void frames whenever necessary (i.e. when there is no data to send). Lastly, note that the method described above does not require changes to server-level NICs.

Switches/Routers. Although strongly dependent on the deployed hardware, it is likely that end-hosts do not support OTN framing and mapping. In this case chains will reach the ToR as a set of Ethernet frames. Once at the ToR, BFP frames can either be left as Ethernet frames or the payload can be extracted and mapped onto OTN [28,36], if available at the ToR, for improved protocol efficiency and timing accuracy. Programmable devices sufficient to implement the BFP resource reservation protocol and bridge between Ethernet and OTN (i.e. EOBs) are commercially available today [37,40]. This EOB would be in charge of: (1) executing the BFP admission control and scheduling algorithm, (2) coordinating with BSSMs in the hosted servers, and (3) cleaning up timing in transmissions coming from low cost servers, i.e. via buffering.

Switches, including ToR switches, need to process the BFP CPs, implement the BFP resource reservation protocol, and handle buffering of data frames accordingly. Devices such as [37], can be used for both OTN and Ethernet-mapped BFP. Data frames coming from end-hosts may suffer from poor synchronization and result in chains that deviate significantly from the required periodicity. The buffer space available at the ToR switch can be used to compensate for these deviations. Once the

ToR switch is reached and data frames of incoming chains are properly (re)timed, the underlying network functions available at the NICs found at the higher layers of the DCN hierarchy (i.e. aggregation and core layer) already provide the timing accuracy required to support BFP transmissions.

If BFP is mapped onto OTN directly [36], the OTN layer can provide the timing accuracy needed by BFP, once BFP data frames are mapped onto ODU [34] channels.

Although BFP mappings over OTN and Ethernet are both possible, Ethernet mapping is less efficient as the BFP data frames would have to go through an additional layer of the network stack at each intermediate DCN node (i.e. L2).

Although BFP mappings over OTN and Ethernet are both possible, Ethernet mapping is less efficient as the data frames would have to go through an additional layer at each node (i.e. L2).

III-B DC-BFP Resource Reservation And Signaling Protocol

DC networks differ from wide-area networks (WAN) in many ways, and protocols that may work in one environment may not perform well in the other [19]. The main differences between DCN and WAN can be summarized in terms of DCNs as follows:

1 Much shorter round trip times (RTT)

2 Applications often need both high bandwidth and low latency

3 Lesser degree of statistical multiplexing, that is, a single flow may dominate a particular path

4 DCN is homogeneous, under a single administrative domain and its topology is stable and known

5 Load balancers and application proxies provide connectivity between the DCN and external networks, making backward compatibility with legacy protocols less of a concern

We exploit these characteristics, in particular points 1 and 4, to modify the BFP resource reservation protocol presented in [28] to work in a DC environment.

The main modification required is the introduction of an offset time (Δ_R) between the reception of confirmation of resource availability for a chain and the start time for the chain's transmission.

The resource reservation procedure starts when a chain is assembled at the source. A CP is generated at the source host and associated with the chain via a "transaction

ID" (*chain ID*). Since the topology of the DCN is known (point 4) we can establish a rough maximum RTT (RTT_{MAX}), this information is included in the CP together with the following parameters:

- *Source/Destination identifier*: Source and destination points for the chain. These could be L2, L3 addresses or VLAN tags, depending on the particular addressing scheme deployed in the DCN.
- *Frame size* (F_s): number of BPFs within each data/void frame.
- *Chain length* (L_{Ch}): number of data frames within a chain (Equation E.2).

$$L_{Ch} = \left\lceil \frac{T}{F_s} \right\rceil \tag{E.2}$$

  where:

  T Transaction size [B].

  F_s Data frame size [B].

- *Transparency Degree* (TD): the chain's period.
- *Expected Time of Arrival* (ETA_{i+1}) [28]: time at which the first bit of the chain will reach the next node in the path (node $i+1$), expressed as the amount of time between reception of the CP and arrival of the first bit of the chain at that node. For DC-BFP this parameter is simply set by the source to equal RTT_{MAX} (Equation E.3). This groups, estimates of CP processing times, propagation delays and ACK reception time (τ_{ACK}) together with their relative estimation errors, into a single, precomputed parameter whose value depends on the particular DCN topology and the routing/load balancing algorithm in use.

$$ETA_{src} = \sum_{i=1}^{N} (\tau_i + p_i) + \tau_{ACK} \doteq RTT_{MAX} \tag{E.3}$$

where:

N Number of nodes in the path.

τ_i Propagation time to reach node i [s].

p_i Estimated CP processing time for node i [s].

τ_{ACK} Time for the ACK to reach the source node [s].

The CP is then forwarded from the source host to the ToR switch, where it is processed to reserve resources for the incoming chain before continuing to the next node in the data path. Each node in the data path process the CP and performs delayed resource reservation for the incoming chain, starting at the time indicated by the ETA parameter of the incoming CP. Before routing the CP to the next node (i.e. node $i+1$) the ETA field is updated with the BT_i computed for the chain at the current node (node i), according to Equations E.4 and E.5. As in [28], buffering (BT) is used to time-align and interleave incoming chains onto outgoing links as well as to compensate for any deviations and timing uncertainties that a chain may have incurred travelling to its destination. Lastly, note that each node computes the ETA value for its next node (ETA_{i+1}).

$$BT_i = t_{e,1} - ETA_i \tag{E.4}$$

$$ETA_{i+1} = ETA_i + BT_i - p_i \tag{E.5}$$

where:

$t_{e,1}$ Ending time of the first available timeslot on the outgoing link.

This procedure is repeated at each node in the data path until either the destination is reached or the CP is blocked. If reservation is successful the source is notified with an ACK. Upon ACK reception the source computes the residual time (Δ_R, Figure E.4) before the scheduled transmission start time using Equation E.6, and initiates chain transmission Δ_R seconds after receiving the ACK.

$$\Delta_R = RTT_{MAX} - RTT_{eff} \tag{E.6}$$

where:

RTT_{eff} is the (effective) RTT experienced by the CP.

If reservation is unsuccessful, the source is notified with a $NACK$ and backs off for a random amount of time before retransmitting. In a DC environment where the

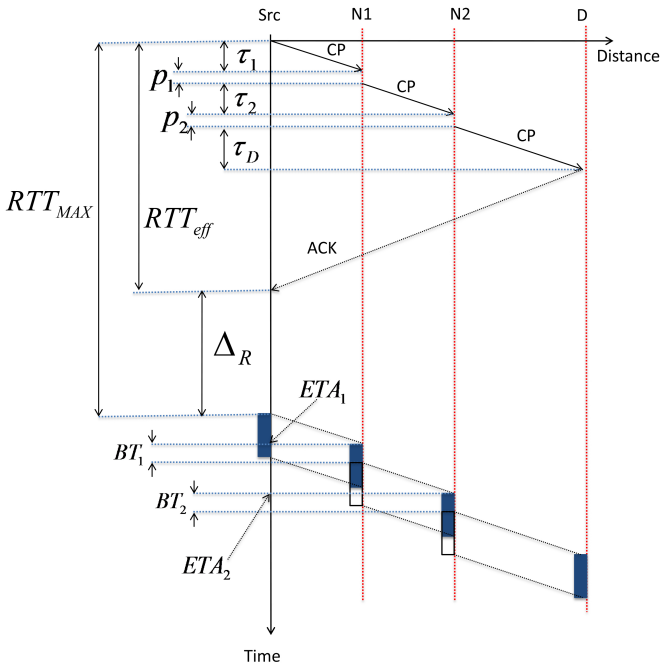


Figure E.4: Path reservation and chain transmission procedure for DC-BFP

topology is stable and known, the backoff time can also be selected considering topological information, chain configuration and RTT_{MAX}. Finding an optimal backoff time for blocked chains is an optimization problem which depends on the particular DCN topology and routing strategy, and its solution is out of the scope of this paper. In this work we generate backoff times using a normally distributed random variable with a narrow standard deviation centered on the chain duration time.

The BFP chain signaling and transmission procedure is illustrated in the time-diagram of Figure E.4. Using an offset time (Δ) to establish chain transmission time at the source, results in wasting some bandwidth per each transaction. The amount of bandwidth wasted is directly proportional to Δ_R. However this wasted bandwidth is minimal, especially for larger files (e.g. $\geq 100MB$), as it will be shown in Section IV.

III-C DC-BFP Routing Algorithm: Weightless Link-Saturation Multipath (WLSM)

Although BFP can inter-operate with any available routing protocol [28], in the present work we assessed BFP under a simple routing strategy based on Dijkstra algorithm [41], to allow BFP flows to be spread over multiple paths, exploiting path

diversity of the topologies studied while avoiding loops. We refer to this algorithm as *Weightless Link-Saturation Multi-path* (WLSM).

An additional parameter named *TTL_threshold* is added to the CP, representing the maximum time a CP is allowed to travel through the network. This parameter is setup at the source node and its value depends on the trade-off between the number of alternative paths to explore and the overall delay per transaction that is acceptable.

At initialization ($t = 0$) each node in the DCN builds a routing table using Dijkstra algorithm [41]. As a CP arrives at a node the routing table is accessed to find the output port relative to the CP destination, the node then attempts to schedule the chain on that port. If this reservation fails, the node then tries one by one all of its available ports excluding the one on which the CP was received. If all the ports are exhausted the CP is blocked and the source is notified ($NACK$), otherwise resources are booked on whatever port was found to be available to service the incoming chain. The CP is then forwarded through that port, towards the next node (original path is now changed). If the *TTL_threshold* is reached or if the CP reachs a node which it has already visited, the CP is blocked and the source is notified. A flow chart for WLSM is shown in Figure E.5.

We are fully aware that this routing strategy is simple-minded and may not work as well as more sophisticated and computationally intensive approaches. Designing an optimal routing protocol for BFP, in a DC environment is out of the scope of this paper. Furthermore, although banal in its functioning, WLSM still allows BFP to outperform other approaches (e.g. those used in [8,12]), suggesting that further improvements are likely achievable by integrating BFP with more sophisticated routing and load balancing protocols.

SIMULATION STUDY

BFP performance in DC environments is evaluated by means of simulation. BFP simulations are run over various topologies, including those presented in VL2 [8], SPAIN [12] as well as over a standard DCN topology (adapted from [15]).

Results for BFP are compared with those obtained in [8] and [12], and show that BFP can: (1) provide uniform high-bandwidth between the servers, (2) provide superior network stability in terms of goodput and delay variation, (3) achieve much shorter delays per transaction, (4) maintain its performance advantages over various topologies, making BFP relatively insensitive to the topology, and lastly (5) provide very high fairness values which, in some cases, reach the theoretical maximum (i.e. Jain's fairness index [42], $J = 1$).

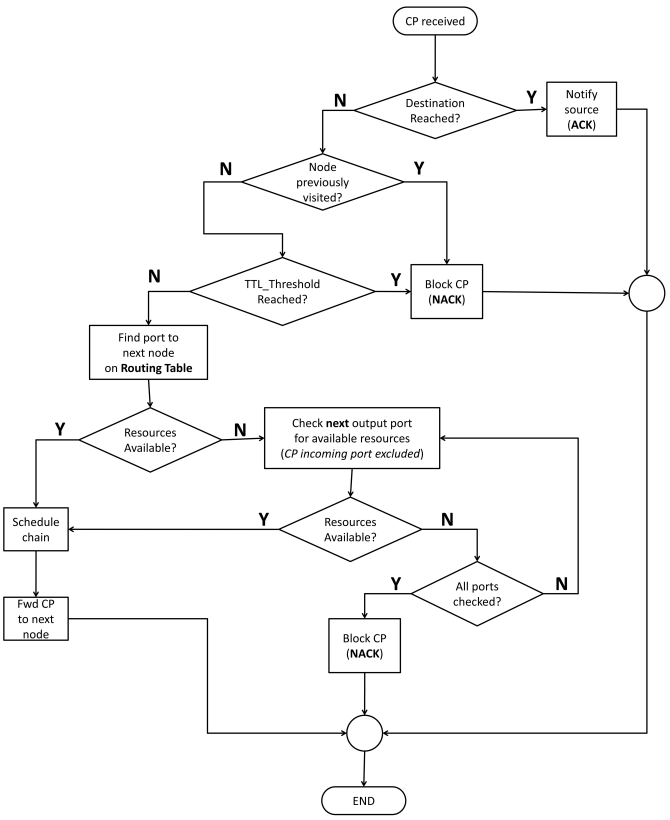


Figure E.5: Weightless Link-Saturation Multi-path routing (flow chart)

In all the topologies tested we use an all-to-all data shuffle stress test, representative of map-reduce-like workloads (e.g. hadoop [43]). Common in many DC environments, this test represents the worst-case scenario for a DCN in which every server tries to transmit to all others simultaneously. This allows us to benchmark BFP performances without focusing on the detail of DC traffic patterns and dynamics, which are complex and may vary from DC to DC [8,18]. The all-to-all data shuffle was also used in [8,12], allowing us to compare our results with those obtained for VL2 and SPAIN, which use TCP as their (reliable) data-transport protocol. No *two-phase* data shuffle was used in the following set of simulations.

The basic set of parameters we measured in all tested scenarios are the following:

- *Shuffle completion time*. The time at which the last bit of the last transaction of the all-to-all data shuffle is delivered.

- *(Mean) Delay per transaction* ($\overline{\Delta}_T$). Completion time for each transaction [44], including resource setup and any eventual retrial due to failure of the resource

reservation procedure. Mean and standard deviation of this parameter are also studied as a measure of the stability of delay performance of BFP.

- *(Mean) Goodput per flow* ($\overline{G}_{Flow}$). Goodput achieved by each flow, together with its standard deviation, highlight the stability of the network goodput under BFP. If compared with Equation E.9, this parameter can give an indication on how well-behaved BFP goodput is. Goodput and delay stability are key to work conservation in DCs, and are important indicators of the stability of network performance [8].

- *Aggregate Goodput* (G_{Agg}). Equivalent to the *goodput efficiency* as defined in [8]. This parameter is the sum of the goodput achieved by all the network interfaces (one per machine) divided by the sum of the interface capacities (Equation E.7).

$$G_{Agg} = \frac{\sum_{i=1}^{N} G_i}{\sum_{i=1}^{N} C_i} * 100 \tag{E.7}$$

where:

G_i is the goodput for interface i

C_i is the capacity for interface i

N is the total number of machines engaged in the all-to-all data shuffle

In all our simulations we use data frames of $15232B$ with $15222B$ of payload, corresponding to the payload area of an OTN frame minus GFP overhead [45] (i.e. BFP over OTN, as described in [28,36]). Alternatively, we could have assembled data frames of roughly the same size using approximately 10 standard Ethernet frames assembled back to back. In this case we would have lost some protocol efficiency due to Ethernet header and IFG, slightly degrading the performances in terms of goodput efficiency and delay per transaction[3]. The maximum buffering allowed for BFP flows was limited to 500 frames per chain for each of the $10Gb/s$ links, and to 100 frames per chain for the $1Gb/s$ links. The average buffer space occupied was computed using Little's law. The maximum average buffer space occupancy was recorded for the case of the VL2 topology with no core nodes active, yielding the following values: $\sim 9MB$ for the $10Gb/s$ links, and $\sim 570kB$ for the $1Gb/s$ links.

[3]Ethernet frames carry less payload than OTN frames, yielding longer chains (Equation E.8) and, as a result, lower goodput efficiency (Equation E.9).

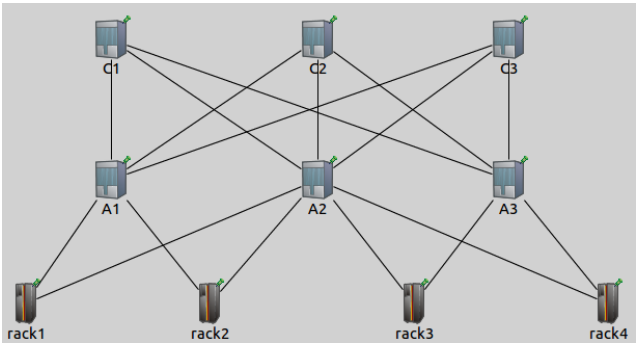


Figure E.6: VL2 Topology

IV-A BFP vs VL2

In our first experiment we compared BFP with VL2 [8] by running BFP over the same topology and with the same configuration used in [8] (Figure E.6): all-to-all data shuffle among 75 servers divided into 4 racks (for a total of $74 * 75 = 5550$ simultaneous flows). We ran the test for transaction sizes varying from a minimum of $500kiB$[4] per file to a maximum of $1GiB$ per file, corresponding to data shuffle varying from $2.775GiB$ to $5.55TiB$. WLSM (See Section III-C) was used for BFP instead of VLB [33]. In [8] only one shuffle size is considered ($2.7TB$).

As shown in Figure E.7, BFP can shuffle roughly the same amount of data as VL2 (i.e. $\sim 2.7TB$), over the same topology in about $310s$, which is over 80 seconds less than VL2 [8]. As expected, BFP shuffle completion time increases linearly with the file size. A behavior observed in all other tests presented in this paper, irrespective of the tested topology.

The average delay per transaction ($\overline{\Delta}_T$) is shown in Figure E.8. As expected, $\overline{\Delta}_T$ also increases linearly with the transaction size, and exhibit a (small) standard deviation ($0.0037s$) which is independent of the file size. Also for Δ_T, the linear increase with transaction size is independent of the topology.

The independence of the standard deviation of $\overline{\Delta}_T$ with file size is due to the fact that all chains retain the same basic structure (i.e. all have the same TD and data frame size). Since, in this experiment no call is blocked ($blocking\_probability = 0$, for all file sizes), the only variability to the transaction delay comes from how much each chain is buffered which is a limited, well defined quantity, dependent only on chain configuration (Equation E.4).

[4]IEC notation.

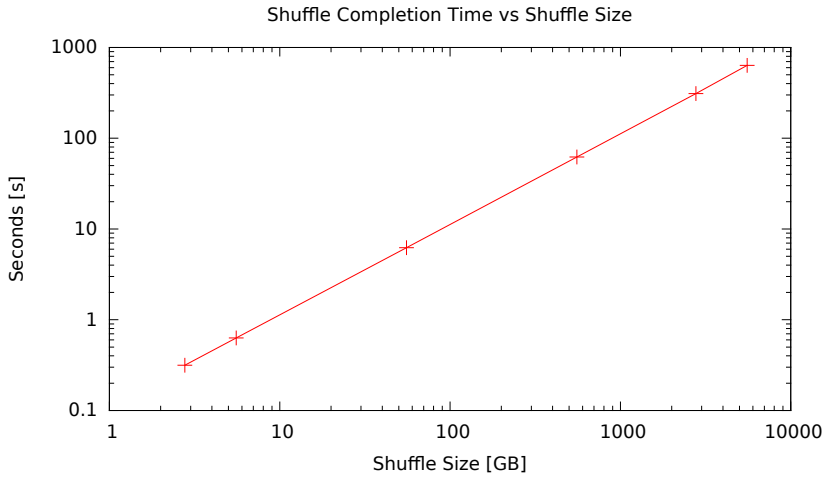


Figure E.7: Shuffle completion time for BFP over the VL2 topology over a range of shuffle sizes.

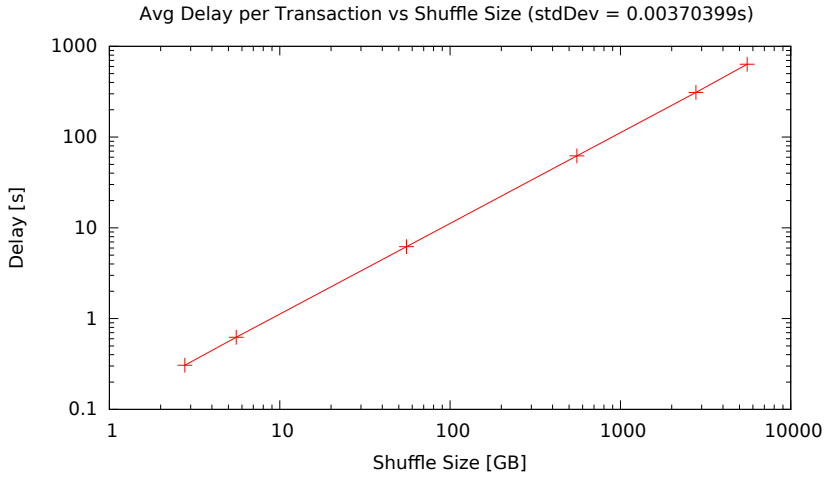


Figure E.8: Average delay per transaction for BFP over VL2 topology.

As BFP is not tied to TCP transmission dynamics, we are able to achieve much higher goodput efficiency with respect to [8]. As shown in Figure E.9, we achieve a goodput efficiency which is 99.9% of the available capacity to servers, versus the 78.4% obtained in [8], thereby showing that BFP is able to handle significantly more traffic with respect to VL2, which uses TCP. Furthermore, in BFP the goodput is perfectly divided among the various flows, yielding a fairness index near to its theoretical maximum ($J = 1$) [42]. In all cases we get $0.999 \leq J \leq 1$, whereas for VL2 [8] $J = 0.995$ (for the $500MB$ files shuffle, where we get $J = 1$). Lastly, the goodput per flow (Figure E.10) obtained by BFP is both higher ($13.5Mb/s$ versus $11.4Mb/s$) and

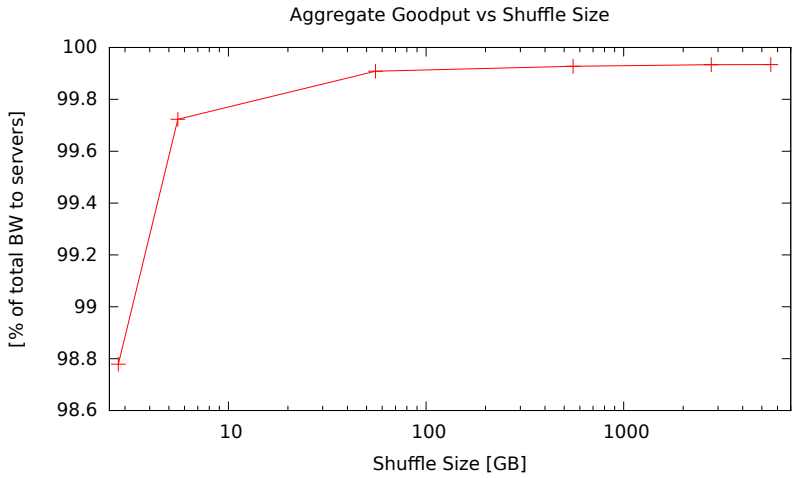


Figure E.9: Goodput efficiency for BFP over VL2 topology.

more stable than that of [8], with a standard deviation ($0.00016Mb/s$) over 3 orders of magnitude smaller than VL2 ($0.75Mb/s$). This small standard deviation highlights BFP's near-deterministic network performance. The performance stability of BFP remains superior to that of VL2 even in the BFP worst case scenario of the smallest shuffle size ($2.775GB$). In this case, BFP still produces a standard deviation of only $0.161Mb/s$, which is over 4 times smaller that that achieved by VL2 (See Figure E.10). Note also that in this case the duration of the resource reservation procedure has the strongest influence on the per-flow goodput, since these chains ($500kiB$) are much shorter than in the $500MiB$ case (i.e. $\sim 2.7TiB$ shuffle).

The detailed simulation data including minimum and maximum values for Δ_T and goodput per flow (G_{Flow}), as well as Jain's fairness index (J) for the various shuffle sizes ($\overline{S}$) tested are shown in Table E.1. Note t hat the overall min/MAX ranges are themselves smaller than VL2 reported standard deviation [8].

Table E.1: *BFP over VL2 topology (additional data).*

$\overline{S}[GB]$	$G_{Flow}(min-max)[Mb/s]$	$\Delta_T(min-max)$	J
2.775	$12.9681-13.7422$	$0.298-0.316s$	0.999
5.55	$13.2845-13.6696$	$0.613-0.631s$	0.999
55.5	$13.4817-13.5203$	$6.204-6.222s$	1
555	$13.5017-13.5056$	$62.111-62.129s$	1
2775	$13.5041-13.5049$	$310.57-310.59s$	1
5550	$13.5044-13.5047$	$636.06-636.08s$	1

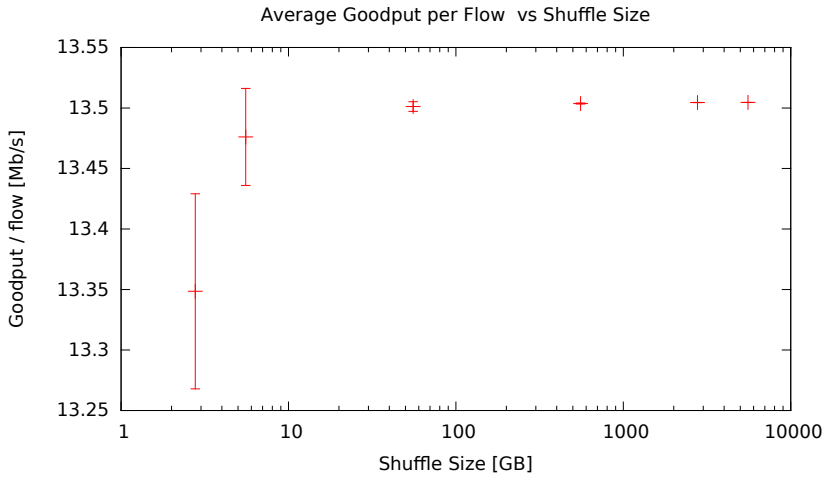


Figure E.10: Goodput per flow for BFP over VL2 topology.

In order to asses BFP's resiliency to failures, we re-ran the experiments by deactivating one-by-one all the core nodes, while monitoring BFP's performance over the constructed topologies. Each deactivated core node can be thought as corresponding to the worst-case scenario of a simultaneous failure of all 3 links attached to the same core node. These tests show, that BFP still retains most of its advantages in terms of network stability, goodput efficiency and fairness even when working in a less connected topologies.

BFP achieves virtually identical performance if one core node is removed (Figures E.11,E.12, E.13 and E.14), showing that BFP can still outperform [8] while using less hardware (one less core switch with its NICs and associated cabling).

BFP performance starts to degrade when two and three core nodes (Figure E.6) are removed, as shown in Table E.2. Note that in keeping with [8], this table only shows results for $2.7TiB$ shuffle. However, BFP's behavior remains similar to that reported in Figures E.7, E.8, E.9 and E.10 for all other tested file sizes.

Under failure conditions, besides the doubled shuffle completion time, BFP still offers stable network performances, higher or equal fairness index, and higher goodput (both aggregate and per flow) with respect to VL2, in all scenarios derived from the VL2 topology.

The doubled shuffle completion time is due to retransmission of BFP chains which, although unnecessary, still retain their configuration (i.e. their TD) when retrying. We solve this issue for map-reduce applications by constructing a *two-step* data shuffle. To our knowledge this approach was never described before in the literature, and it

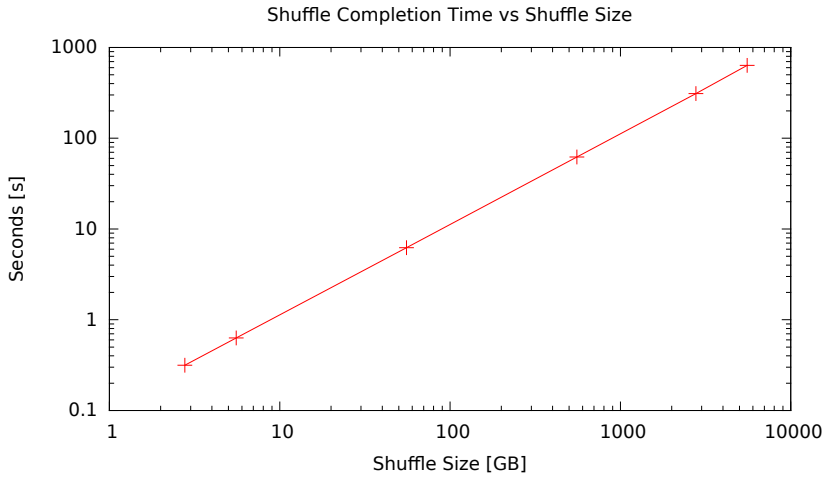


Figure E.11: Shuffle completion time for BFP over VL2 topology (two core nodes active)

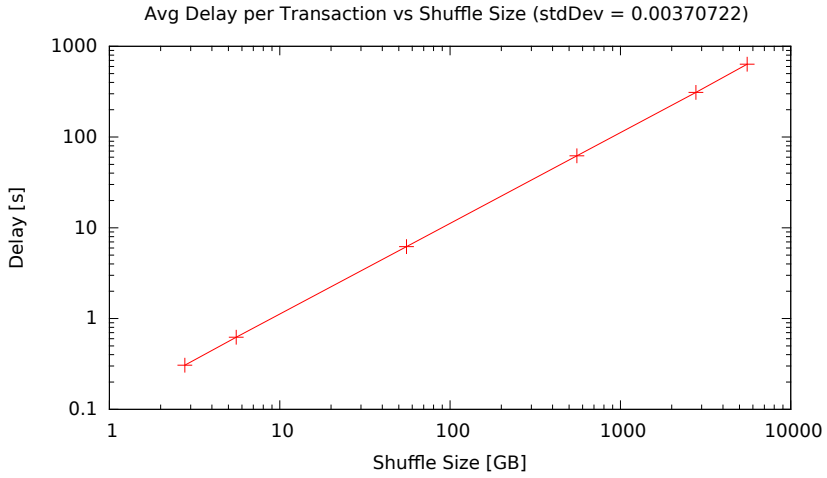


Figure E.12: Average delay per transaction for BFP over VL2 topology (two core nodes active)

is specifically designed for BFP in the context of map-reduce-like workloads (i.e. all-to-all data shuffle).

During *phase 1* of a *two-step* data shuffle all sources start their connection to all other machines simultaneously, as per standard all-to-all data shuffles. In *phase 2*, which is triggered only if some flows are blocked during the first phase of the shuffle, we use a "centralized shuffle controller" (this could just be a application on a VM, e.g. the dispatching entity of the shuffle jobs) in charge of collecting information on how many BFP flows have been blocked from the applications involved in the shuffle

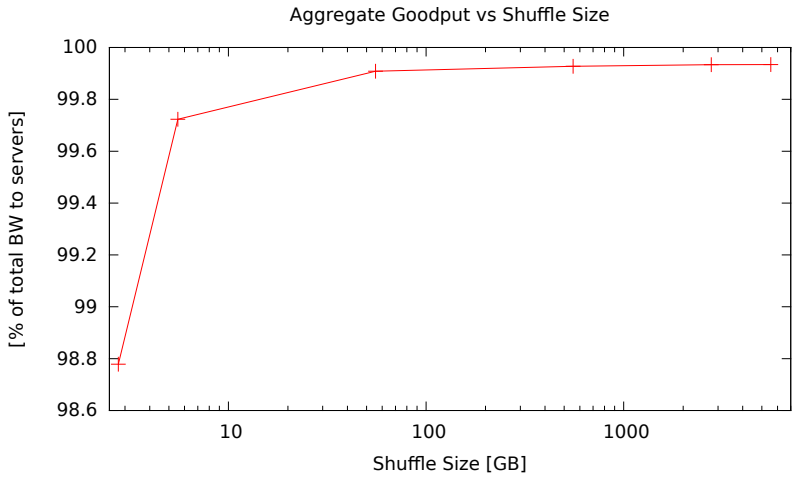


Figure E.13: Goodput efficiency for BFP over VL2 topology (two core nodes active)

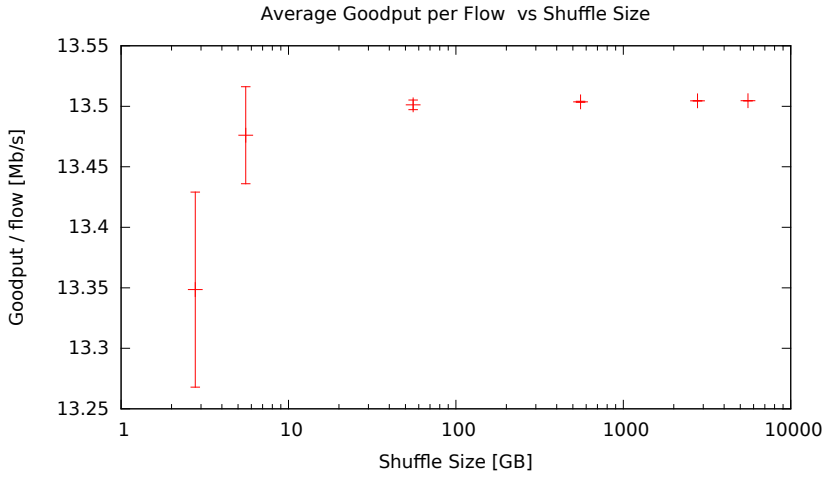


Figure E.14: Goodput per flow for BFP over VL2 topology (two core nodes active)

and to provide them with a TD to be used for flow retransmission during *phase 2* (i.e. TD_{Ph2}). TD_{Ph2} is decided by the shuffle controller based on how many flows were blocked during *phase 1* of the shuffle. Due to the smaller number of flows to be shuffled with respect to *phase 1*, previously blocked flows can now be transmitted, using a smaller TD (i.e. TD_{Ph2}), yielding shorted chains which can be delivered faster than if retransmitted using the same TD used during the first phase of the shuffle.

As an example of a *two-phase* data shuffle, consider performing the shuffle over the topology shown in Figure E.6 in the case of all core nodes inactive (no core nodes). Using the same chain configuration used for the standard VL2 case in the first phase

Table E.2: *BFP over VL2 topology with one and zero core nodes active for* $2.775TB$ *data shuffle.* (σ *is computed over the 5550 flows).*

core nodes active	one active core node	0 active core nodes
Shuffle completion time[s]	$621.377s$	$621.378s$
Aggregate goodput [%]	99.5644%	99.0108%
Average delay per transaction[s]	$312.881s$ ($\sigma = 26.6s$)	$316.282s$ ($\sigma = 41.54s$)
Average goodput per transaction[Mb/s]	13.4546 Mb/s ($\sigma = 0.29$)	13.3798 Mb/s ($\sigma = 0.90$)
Flow blocking probability[%]	0.74%	1.82%
Fairness index (J [42])	0.998	0.995

of the shuffle (i.e. $TD = 74$). Simulation shows that only $\sim 1.8\%$ of the flows are blocked (corresponding to roughly 100 blocked flows over a total of 5550 flows).

Considering that: (1) end-hosts are connected via $1Gb/s$ to their ToR, and the ToR are connected via $10Gb/s$ links to the aggregation layer [8], and that (2) blocked flows are not all directed towards a single destination, we can select $TD_{Ph2} = 10$ for the second phase of the shuffle. Chains with $TD = TD_{Ph2}$, will be upshifted to $TD = 100$ once the flows reach the $10Gb/s$ links, guaranteeing the interleaving of the remaining flows regardless of their path.

Let's now compute the two-step shuffle completion time for three simultaneous core node failures. The duration of a chain as a function of its TD and L_{Ch} (Δ_{Ch}) is expressed by Equation E.8:

$$\Delta_{Ch} = f_s * [TD * (L_{Ch} - 1) + 1] \tag{E.8}$$

where:

f_s Duration (in seconds) of a data frame at the (output) line rate

TD Transparency Degree

With a transaction size of $500MiB$ per transaction ($L_{Ch} = 34443$ frames), $TD_{Ph2} = 10$, and $f_s \approx 0.000122s$ (at the end-host), Δ_{Ch} at phase 2 yields a value of roughly $40s$ which, added to the $310s$, i.e. the time it takes to deliver the first "batch" of

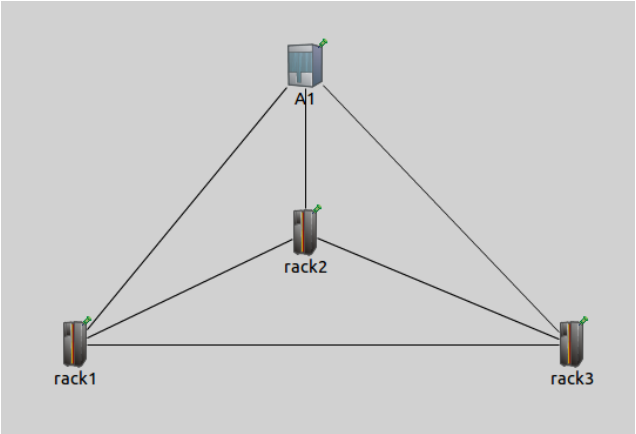


Figure E.15: Topology used in [12].

chains (simulated), gives a two-step shuffle completion time of around $350s$ which is still roughly $40s$ less than in the VL2 best case scenario.

IV-B BFP vs SPAIN

SPAIN [12] was proposed as a COTS-based multipath forwarding approach. Unlike VL2, SPAIN is designed to be topology-independent. We tested BFP against SPAIN by reproducing the same network topology and conditions examined in [12] (Figure E.15). However, instead of using the SPAIN multipath algorithm, for BFP we use WLSM (Section III-C). As in [8], in [12] only a data shuffle with $500MB$ per file is considered (i.e. $\sim 2.7TB$ shuffle size). In the topology used in [12] there are 80 servers distributed over 3 racks with 23, 28 and 29 machines, respectively. We evaluated BFP on this same configuration.

The parameters measured in [12] are the mean goodput per host ($\overline{G}_{Host}$), aggregate goodput (G_{Agg}), the mean host completion time ($\Delta_{T_{Host}}$) and the total shuffle completion time. Fairness and network stability parameters such as standard deviations for goodput per flow and delay per transaction were not addressed in [12].

Performance of BFP in the conditions described in [12] is presented in Figures E.16, E.17, E.18 and E.19, while additional simulation data is presented in Table E.3. Table E.4 shows how BFP compares to SPAIN for the same parameters and shuffle size considered in [12]. As it can be seen from the graphs and Table E.4, in spite of its simple routing algorithm (WLSM), BFP still outperforms SPAIN [12] in terms of network goodput and delay. Furthermore, also for this topology, BFP performance is very close to those achieved over the VL2 topology, showing BFP insensitivity to the particular topology in use. The slight variations with respect to the VL2 topology

that arise due to the different chain configuration used in this case ($TD = 79$, same data frame size), yields slightly longer chains (Equation E.2) to transport the same payload, thereby lowering the bandwidth occupation (Equation E.1).

Table E.3: *BFP over SPAIN topology* (Figure E.15).

$\overline{S}[GB]$	$G_{Flow}(min-max)[Mb/s]$	$\Delta_T(min-max)$	J
3.16	$11.9663 - 12.7807$	$0.320 - 0.342s$	0.999
6.32	$12.3502 - 12.7600$	$0.657 - 0.679s$	0.999
63.2	$12.6187 - 12.6602$	$6.625 - 6.647s$	1
632	$12.6462 - 12.6504$	$66.311 - 66.333s$	1
3160	$12.6492 - 12.6500$	$331.56 - 331.58s$	1
6320	$12.6496 - 12.6500$	$679.04 - 679.06s$	1

Table E.4: *BFP vs SPAIN [12].* Parameter by parameter comparison (*spanning tree results from [12] are also reported*)

	Spanning Tree	SPAIN	BFP
$\overline{G}_{host}[Mb/s]$	449.25	834.51	999.32
$G_{Agg}[GB/s]$	35.60	66.68	79.94
$\Delta_{T_{Host}}[s]$	744.57	397.50	331.57
Shuffle completion time[s]	831.95	431.12	331.58

IV-C BFP Over Standard DCN Topology [15]

To further test BFP sensitivity to the topological configurations of the underlying network, we also ran an experiment over a standard DCN architecture as described in [12]. An all-to-all shuffle between 75 machines distributed over 4 racks (Figure E.20) was performed. We obtained the same behavior observed in all other topologies tested, and the results are still very close to those previously obtained (See Figures E.21, E.22, E.23 and E.24). Additional simulation results are presented in Table E.5. Identical behavior and similar performance are obtained when the all-to-all shuffle is increased to involving 80 machines. The only difference was a slight decrease in the goodput per flow and an increase in the delay per transaction and shuffle completion time as a result of the higher TD used to match the number of machines.

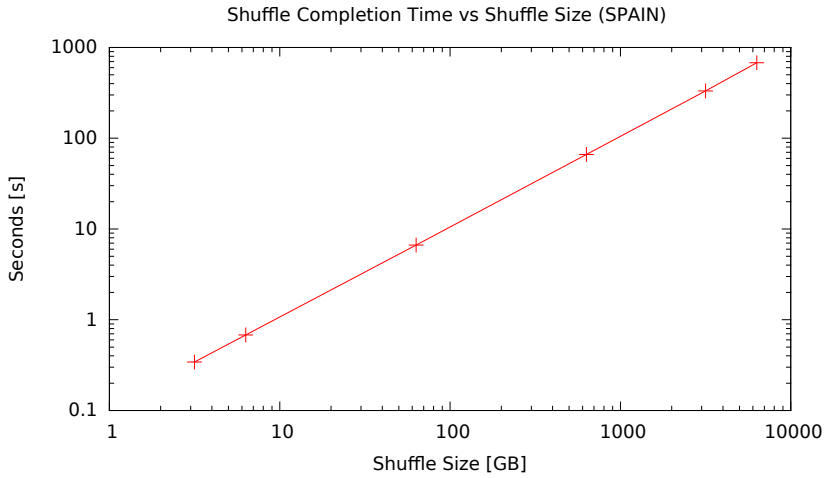


Figure E.16: Shuffle completion time for BFP over SPAIN topology

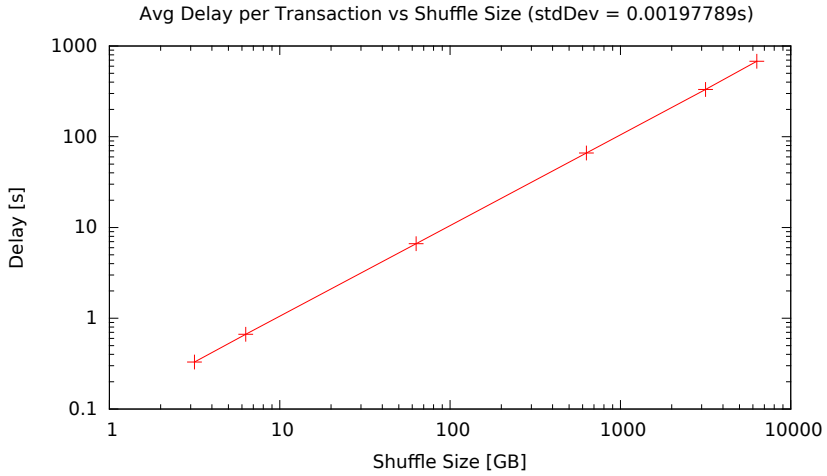


Figure E.17: Average delay per transaction for BFP over the SPAIN topology [12].

Table E.5: *BFP over standard CISCO DCN [15](additional data).*

$\overline{S}[GB]$	$G_{Flow}(min-max)[Mb/s]$	$\Delta_T(min-max)$	J
2.775	$12.9681-13.7422$	$0.298-0.315s$	0.999
5.55	$13.2845-13.6696$	$0.613-0.631s$	0.999
55.5	$13.4817-13.5203$	$6.204-6.222s$	1
555	$13.5017-13.5056$	$62.112-62.129s$	1
2775	$13.5041-13.5049$	$310.57-310.59s$	1
5550	$13.5044-13.5047$	$636.06-636.08s$	1

IV-D BFP Performance Predictability

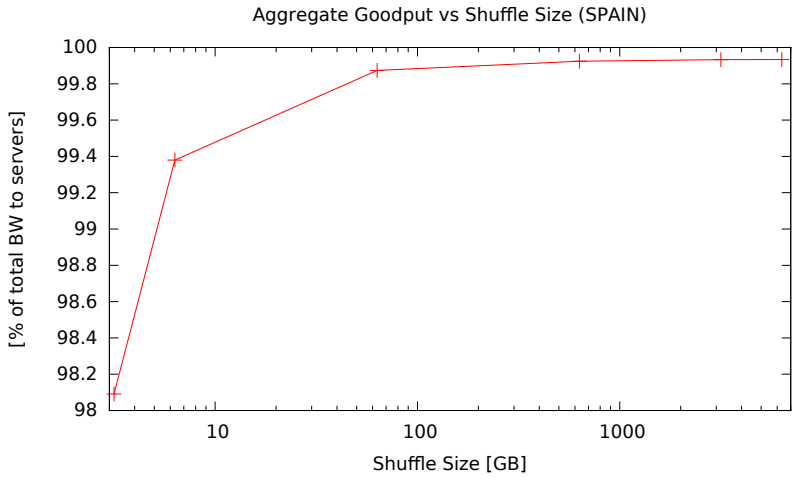


Figure E.18: Aggregate goodput (goodput efficiency) for BFP over SPAIN topology [12].

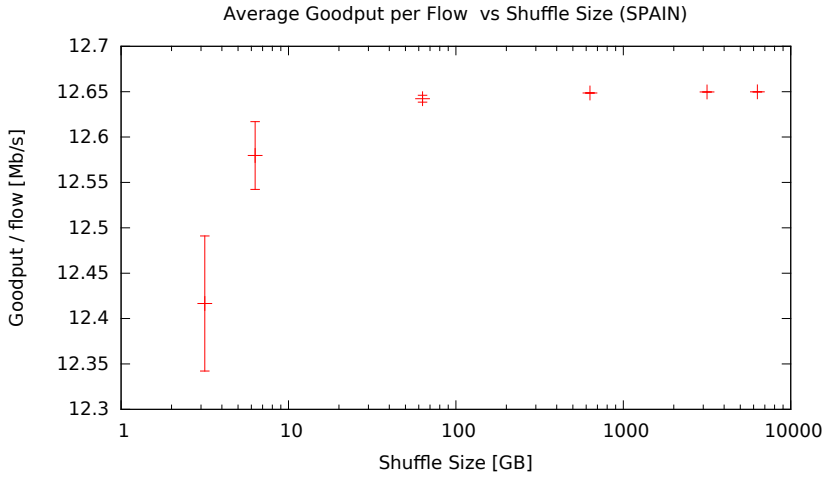


Figure E.19: Goodput per flow for BFP over SPAIN topology [12].

Offering predictable performances is an important issue in DCNs [2,5,9,46]. To show the degree to which BFP is well-behaved, our simulation study concludes by showing how closely measured values of goodput per flow ($\overline{G}_{Flow}$ $[Mb/s]$) match the theoretical values computed via Equation E.9.

$$\hat{G}_{Flow} = \frac{f_b * L_{Ch}}{f_s * [TD * (L_{Ch} - 1) + 1] + RTT_{Max}} \tag{E.9}$$

This equation extends Equation E.1 to include the delay introduced by the reservation procedure (i.e. RTT_{Max}) . Delay due to chain buffering was not included in this

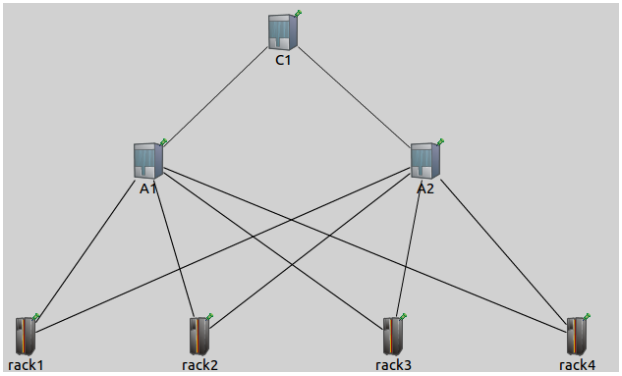


Figure E.20: Example of a standard CISCO DCN topology [15].

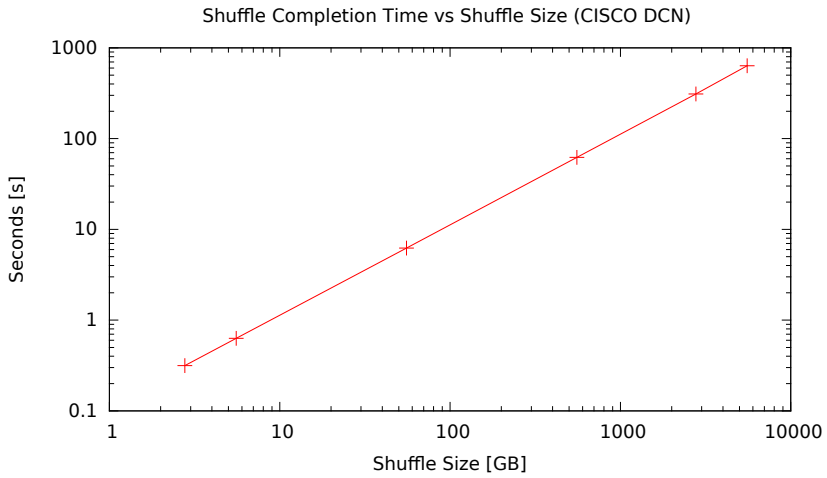


Figure E.21: Shuffle completion time for BFP over standard DCN topolgy [15]

equation, as only the maximum value of this parameter can be computed in advance. Its precise value depends on the topology and the routing/load balancing algorithm used. Table E.6 shows values for all the topologies tested for the $2.775TiB$ shuffle size. Whenever no flow is blocked (i.e. there is enough bandwidth to accommodate the offered load), Equation E.9 is an *unbiased* estimator of the average goodput per flow ($\overline{G}_{Flow}$). In all other cases the goodput per flow performance of BFP are still predictable with reasonable accuracy. A similar degree of predictability was observed for the other shuffle sizes as well as for the delay per transaction (Δ_T, Equation E.2).

V. DISCUSSION

Through our simulation study we have shown that BFP offers significant advantages over other proposed DC networking approaches, particularly those using TCP as their main transport protocol (e.g. [8,12]). BFP can deliver high aggregate good-

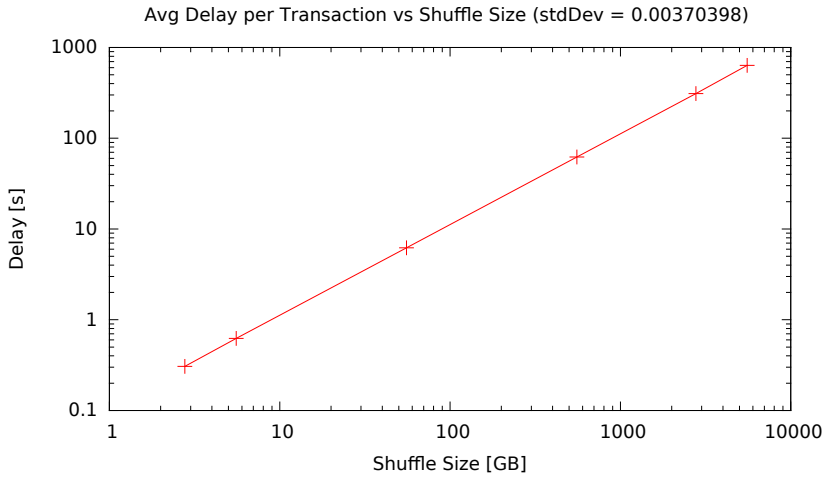


Figure E.22: Average delay per transaction for BFP over standard DCN topolgy [15]

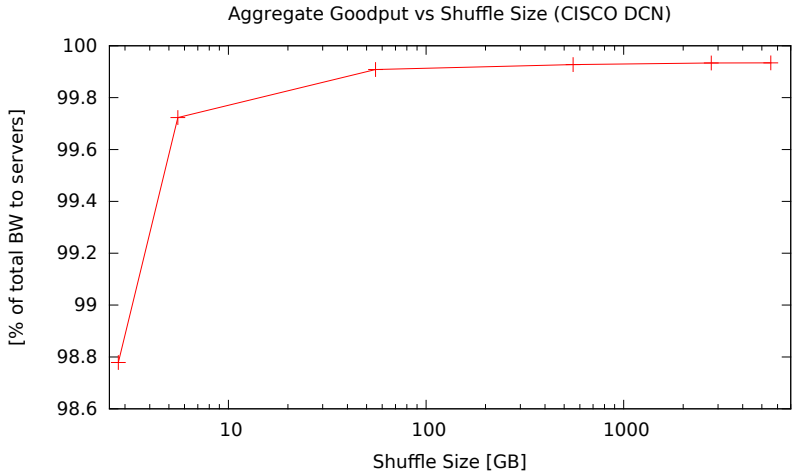


Figure E.23: Goodput efficiency for BFP over standard DCN topolgy [15]

Table E.6: Goodput per flow ($\overline{G}_{Flow}[Mb/s]$), $2.775TB$ shuffle size

Simulation	Theory	Bias	MSE	Topology
13.5045	13.5050	0	$2.335 * 10^{-7}$	VL2
13.5045	13.5050	0	$2.335 * 10^{-7}$	VL2 (2 cor.)
13.4546	13.5050	0.044	0.337085	VL2 (1 cor.)
13.3798	13.5050	0.109	0.830419	VL2 (0 cor.)
12.6497	12.6502	0	$2.37363 * 10^{-7}$	SPAIN
13.5045	13.5050	0	0.729901	CISCO

put ($\geq$ 98%) as well as stable and predictable network performances irrespective of

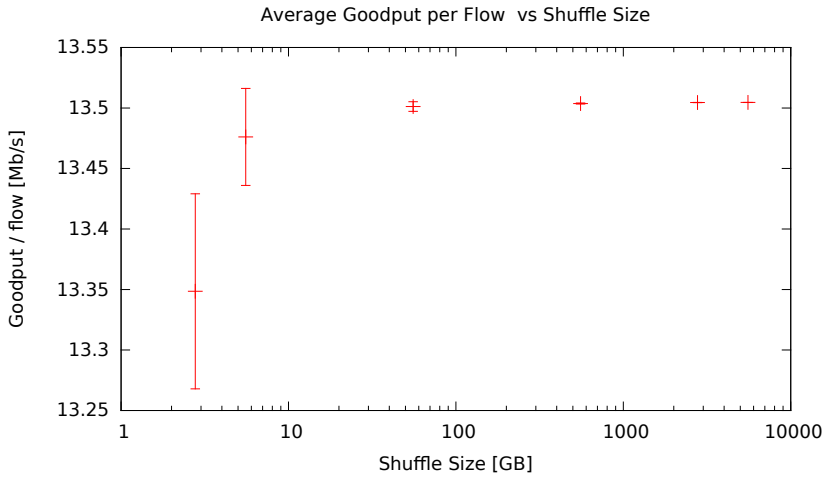


Figure E.24: Goodput per flow for BFP over standard DCN topolgy [15]

the physical DCN topology. By providing uniform high bandwidth between servers, BFP is also able to significantly improve DCN agility [2,8]. The low BFP sensitivity to the DCN topology shows BFP's potential versatility and applicability to a wide variety of DCN architectures.

When it comes to performance isolation, BFP relies on both the effectiveness of the load balancing algorithm deployed in the DCN (similarly to [8]), as well as on the periodic structure of each chain, which naturally isolates data flows, limiting their interaction to a (buffering) delay increase, the maximum of which can be computed in advance. Approaches such as those proposed in [5,8,13,47] used in combination with BFP can provide even stronger performance isolation. This is mostly due to the deterministic nature of BFP, where we can accurately compute the bandwidth occupancy and duration of each flow in advance. This then allows the estimated values of $\overline{G}_{Flow}$ ($\hat{G}_{Flow}$) and $\overline{\Delta}_T$ to be used as a parameters for routing/load balancing algorithms, allowing BFP to make "informed" decisions.

In [28], a strategy to integrate BFP with existing routing protocols is described. The well-behaved nature of the BFP (e.g. bandwidth occupation and flow duration) is a desirable characteristic that can be used by many proposed DCN architectures (e.g. [5,8,12,13]). When used in combination with these architectures, BFP quasi-deterministic network dynamics allow whatever DCN architecture is implemented to make more informed decisions (with respect to bandwidth assignment, routing and load balancing) than possible when dealing with typical TCP dynamics, which tend to be far less predictable. This allows BFP to serve as a simple but effective mean to

control network bandwidth for bandwidth-intensive applications, e.g. those arising with Big Data's emergence (e.g. [43,48,49]).

VI. CONCLUSIONS

BFP offers a mean to efficiently manage large file transmissions and bandwidth-intensive applications using a periodical data structure (BFP chain), which is designed to closely match OTN service characteristics. Chains are scheduled in advance over an end-to-end path using a lightweight, two-way signaling protocol performing delayed resource reservation on a per-chain basis. This work considers the application of BFP to DCNs and presents a possible deployment strategy of BFP using commodity servers augmented with a BFP software module (BSSM). A BFP network module (EOB) at each switch/router implements the BFP resource reservation protocol and act as a bridge between Ethernet and OTN. Performance of BFP was studied by means of simulation and compared with popular DCN architectures. BFP showed superior network performance and stability as well as predictable network performance. These advantages were maintained over various topologies, highlighting BFP's insensitivity to the underlying DCN topology.

BFP simplifies data transmission dynamics with respect to traditional data transfer protocols, offering stable and predictable network performances. BFP behavior is largely deterministic, and it can integrate with many proposed and implemented DCN architectures, providing them with a simple, highly-efficient data transfer protocol to be used where traditional protocols, such as TCP, fall short.

References:

1. Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2008. The cost of a cloud: research problems in data center networks. SIGCOMM Comput. Commun. Rev. 39, 1 (December 2008), 68-73.

2. Dennis Abts and Bob Felderman. 2012. A guided tour of data-center networking. Commun. ACM 55, 6 (June 2012), 44-51.

3. Jrg Schad, Jens Dittrich, and Jorge-Arnulfo Quian-Ruiz. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. 3, 1-2 (September 2010), 460-471.

4. Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

5. Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. 2011. Towards predictable datacenter networks. SIGCOMM Comput. Commun. Rev. 41, 4 (August 2011), 242-253.

6. Iosup, A; Yigitbasi, N.; Epema, D., "On the Performance Variability of Production Cloud Services," Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on , vol., no., pp.104,113, 23-26 May 2011

7. Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. 2008. Improving MapReduce performance in heterogeneous environments. In Proceedings of the 8th USENIX conference on Operating systems design and implementation (OSDI'08). USENIX Association, Berkeley, CA, USA, 29-42.

8. Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09).

9. Jrg Schad, Jens Dittrich, and Jorge-Arnulfo Quian-Ruiz. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. 3, 1-2 (September 2010), 460-471.

10. Yan Zhang; Ansari, N., "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," Communications Surveys & Tutorials, IEEE , vol.15, no.1, pp.39,64, First Quarter 2013

11. Ali Hammadi and Lotfi Mhamdi. 2014. Review: A survey on architectures and energy efficiency in Data Center Networks. Comput. Commun. 40 (March 2014), 1-21

12. Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. 2010. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI'10). USENIX Association, Berkeley, CA, USA, 18-18.

13. Alan Shieh, Srikanth Kandula, Albert Greenberg, Changhoon Kim, and Bikas Saha. 2011. Sharing the data center network. In Proceedings of the 8th

USENIX conference on Networked systems design and implementation (NSDI'11). USENIX Association, Berkeley, CA, USA, 309-322.

14. James Hamilton. 2007. On designing and deploying internet-scale services. In Proceedings of the 21st conference on Large Installation System Administration Conference (LISA'07), Paul Anderson (Ed.). USENIX Association, Berkeley, CA, USA, , Article 18 , 12 pages.

15. Cisco Data Center Infrastructure 2.5 Design Guide. Available at: `http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/\\ccmigration\_09186a008073377d.pdf`

16. Luiz André Barroso, Jeffrey Dean, and Urs Hlzle. 2003. Web Search for a Planet: The Google Cluster Architecture. IEEE Micro 23, 2 (March 2003), 22-28.

17. http://highscalability.com/google-architecture

18. Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10). ACM, New York, NY, USA, 267-280.

19. Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). SIGCOMM Comput. Commun. Rev. 40, 4 (August 2010), 63-74.

20. Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Brian Mueller. 2009. Safe and effective fine-grained TCP retransmissions for datacenter communication. In Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 303-314.

21. Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. Understanding TCP incast throughput collapse in datacenter networks. In Proceedings of the 1st ACM workshop on Research on enterprise networking (WREN '09). ACM, New York, NY, USA, 73-82.

22. Das, T.; Sivalingam, K.M., "TCP improvements for data center networks," Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on , vol., no., pp.1,10, 7-10 Jan. 2013

23. Kajita, K.; Osada, S.; Fukushima, Y.; Yokohira, T., "Improvement of a TCP Incast avoidance method for data center networks," ICT Convergence (ICTC), 2013 International Conference on , vol., no., pp.459,464, 14-16 Oct. 2013

24. Zhengwei Zhao; Zhixiong Jiang; Chunyang Lu; Yushan Cai; Jingping Bi, "A Congestion Control Algorithm for Datacenters," Networking, Architecture and Storage (NAS), 2013 IEEE Eighth International Conference on , vol., no., pp.98,104, 17-19 July 2013

25. Coudron, M.; Secci, S.; Pujolle, G.; Raad, P.; Gallard, P., "Cross-layer cooperation to boost multipath TCP performance in cloud networks," Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on , vol., no., pp.58,66, 11-13 Nov. 2013

26. Sahan Gamage, Ardalan Kangarlou, Ramana Rao Kompella, and Dongyan Xu. 2011. Opportunistic flooding to improve TCP transmit performance in virtualized clouds. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11). ACM, New York, NY, USA, , Article 24 , 14 pages

27. IETF, RFC 793. *Transmission Control Protocol.*
Available at: `https://www.ietf.org/rfc/rfc793.txt`

28. Albanese, Ilijc; Yazir, Yagiz Onat; Neville, Stephen W.; Ganti, Sudhakar; Darcie, Thomas E., "Big file protocol (BFP): A traffic shaping approach for efficient transport of large files," High Performance Switching and Routing (HPSR), 2014 IEEE 15th International Conference on , vol., no., pp.125,130, 1-4 July 2014

29. Bill Claybrook. Comparing cloud risks and virtualization risks for data center apps. Available at: http://searchdatacenter.techtarget.com/tip/Comparing-cloud-risks-and-virtualization-risks-for-data-center-apps

30. H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, L. Zhou. Distributed systems meet economics: pricing in the cloud Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, pp. 6-6

31. Jie Liu; Feng Zhao; Xue Liu; Wenbo He, "Challenges Towards Elastic Power Management in Internet Data Centers," Distributed Computing Systems Workshops, 2009. ICDCS Workshops '09. 29th IEEE International Conference on , vol., no., pp.65,72, 22-26 June 2009

32. Amar Phanishayee, Elie Krevat, Vijay Vasudevan, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Srinivasan Seshan. 2008. Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08), Mary Baker and Erik Riedel (Eds.). USENIX Association, Berkeley, CA, USA, , Article 12 , 14 pages.

33. Rui Zhang-Shen and Nick McKeown. 2005. Designing a predictable internet backbone with valiant load-balancing. In Proceedings of the 13th international conference on Quality of Service (IWQoS'05), Hermann Meer and Nina Bhatti (Eds.). Springer-Verlag, Berlin, Heidelberg, 178-192.

34. ITU-T G.709 *interfaces for Optical Transport Networks.*
Available at: `http://www.itu.int/rec/T-REC-G.709/`

35. IEEE ETHERNET, IEEE Standard 802.3, 2012.
Available at: `http://standards.ieee.org/about/get/802/802.3.html`

36. Albanese, Ilijc; Yazir, Yagiz Onat; Neville, Stephen W.; Ganti, Sudhakar; Darcie, Thomas E., "Big File Protocol (BFP) for OTN and Ethernet Transport Systems". Journal of Optical Communications and Networking, to be published.

37. Nallatech PCIe-395 - FPGA Network Processing Card, specifications available at: `http://www.nallatech.com/images/stories/product_briefs/pcie_395pb_v1_6.pdf`

38. Yaozu Dong; Xiaowei Yang; Xiaoyong Li; Jianhui Li; Kun Tian; Haibing Guan, "High performance network virtualization with SR-IOV," High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on , vol., no., pp.1,10, 9-14 Jan. 2010

39. PCI-SIG SR-IOV Primer. Intel *white paper*.
Available at: `http://www.intel.com/content/www/us/en/pci-express/pci-sig-sr-iov-`
`html`

40. TPOC226: 200G P-OTS Any-Rate Mapper. Intellectual Property (IP). Available at: `http://www.altera.com/literature/po/ss-tpoc226.pdf`

41. E. W. Dijkstra. A note on two problems in connexion with graphs Numerische Mathematik In Numerische Mathematik, Vol. 1, No. 1. (1 December 1959), pp. 269-271

42. R. Jain. The Art of Computer Systems Performance Analysis. John Wiley and Sons, Inc., .

43. Apache Hadoop, in Wikipedia, the Free Encyclopedia [Online], May 13, 2008. Available: `http://en.wikipedia.org/wiki/Apache_Hadoop`

44. Nandita Dukkipati and Nick McKeown. 2006. Why flow-completion time is the right metric for congestion control. SIGCOMM Comput. Commun. Rev. 36, 1 (January 2006), 59-62

45. ITU-T G.7041 (04/2011). *Generic Framing Procedure*.

46. By Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia Communications of the ACM, Vol. 53 No. 4, Pages 50-58

47. Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazires, Balaji Prabhakar, Changhoon Kim, and Albert Greenberg. 2013. EyeQ: practical network performance isolation at the edge. In Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation (nsdi'13), Nick Feamster and Jeff Mogul (Eds.). USENIX Association, Berkeley, CA, USA, 297-312.

48. J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM, vol. 51, no. 1 (2008), pp. 107113.

49. M. Burrows, The chubby lock service for loosely-coupled distributed systems, in Proceedings of OSDI06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November 2006.