NTNU – Trondheim
Norwegian University of
Science and Technology

Fusion Network Performance

An Integrated Packet/Circuit Hybrid Optical Network

Edgar Gerardo Sanchez Gomez

Master in Security and Mobile Computing
Submission date: June 2013
Supervisor: Steinar Bjørnstad, ITEM
Co-supervisor: Raimena Veisllari, ITEM

Norwegian University of Science and Technology
Department of Telematics

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

PROBLEM DESCRIPTION

Student's name:	Edgar Gerardo Sánchez Gómez
Course:	Master Thesis
Project title:	**Fusion Network Performance: An Integrated Packet/Circuit Hybrid Optical Network**

Problem description:

Applications and services such as online gaming, telemedicine and e-health, online banking, cloud computing, high-quality videoconferencing, etcetera are becoming more and more sensitive to timely delivery. Furthermore, traffic volumes in both fixed and mobile networks are expected to keep increasing exponentially over the next few years.

These services and traffic demand are dependent on minimal jitter and delay to run successfully, and they do not tolerate any data loss. And so they need a higher class of Quality of Service (QoS), like the one offered by circuit switching. However, in order to achieve a cost-efficient network with high throughput, packet switching technology is needed. How to achieve them both?

TransPacket is a startup-company that has implemented the novel fusion technology, also called OpMiGua integrated hybrid networks. The fusion concept has the main objective of combining the best properties from both circuit and packet switched networks into a hybrid solution.

It is the objective of this thesis work to perform a network experiment involving TransPacket H1 nodes measuring performance parameters like latency, latency variation (packet delay variation) and packet loss. The experiment shall be performed in the premises of NTNU but remote operation of experimental equipment is available.

Deadline:	June 30, 2013
Submission date:	June 20, 2013
Department:	Department of Telematics
NTNU Supervisor:	Steinar Bjørnstad
KTH Supervisor:	Markus Hidell

Abstract

IP traffic increase has resulted in a demand for greater capacity of the underlying Ethernet network. As a consequence, not only Internet Service Providers (ISPs) but also telecom operators have migrated their mobile back-haul networks from legacy SONET/SDH circuit-switched equipment to packet-based networks.

This inevitable shift brings higher throughput efficiency and lower costs; however, the guaranteed QoS and minimal delay and packet delay variation (PDV) that can only be offered by circuit-switched technologies such as SONET/SDH are still essential and are becoming more vital for transport and metro networks, as well as for mobile back-haul networks, as the range and demands of applications increase.

Fusion network offers "both an Ethernet wavelength transport and the ability to exploit vacant wavelength capacity using statistical multiplexing *without interfering* with the performance of the wavelength transport" [RVH] by dividing the traffic into two *service classes* while still using the capacity of the same wavelength in a *wavelength routed optical network* (WRON) [SBS06]:

1. A *Guaranteed Service Transport* (GST) service class supporting QoS demands such as no packet loss and fixed low delay for the circuit-switched traffic.

2. A *statistical multiplexing* (SM) service class offering high bandwidth efficiency for the best-effort packet-switched traffic.

Experimentation was carried out using two TransPacket's H1 nodes and the Spirent TestCenter as a packet generator/analyzer with the objective of demonstrating that the fusion technology, using TransPacket's H1 muxponders allow transporting GST traffic with circuit QoS; that is with no packet loss, no PDV and minimum delay independent of the insertion of statistically multiplexed traffic.

Results indicated that the GST traffic performance is completely independent of the added SM traffic and its load. GST was always given absolute priority and remained with a constant average end-to-end delay of 21.47 μs, no packet loss and a minimum PDV of 50 ns while SM traffic load increased, increasing the overall 10GE lightpath utilization up to 99.5%.

Acknowledgements

This report serves as a Master Thesis for the Master's Programme in Security and Mobile Computing at the Norwegian University of Science and Technology, NTNU and the Royal Institute of Technology, KTH. The assignment was given by TransPacket's CEO and NTNU professor Steinar Bjørnstad and the project was carried out remotely in room F251, inside the Electrical Engineering building within the Department of Telematics in the Gløshaugen campus of NTNU.

First of all, I would like to thank my supervisors Steinar Bjørnstad and Markus Hidell for their patience, their guidance, their feedback and their help throughout all this process. Furthermore, I would like to thank PhD student Raimena Veisllari for her constant help and availability. She was there to answer any question and doubt I ever had during experimentation, and she made sure to push me when I needed to be pushed. I would like to think that the quality of this thesis reflects not only on my hard work, but on the constant help from all of them.

I am also thankful to the NordSecMob Consortium and the Erasmus Mundus Commission, for without their financial support, all of this would not have happened. Thanks to May-Britt Eklund Larsson and Mona Nordaune for all of their administrative assistance and help to make sure this programme runs as smoothly as it does.

And of course, thank you to my family and friends, who have always believed in me and encouraged me every step of the way.

Trondheim, June 20, 2013

Edgar Gerardo Sánchez Gómez

Abbreviations

BP Buffer Priority

CEO Chief Executive in Office

CLI Command Line Interface

CWDM Coarse Wavelength Division Multiplexing

DEMUX Demultiplexor

DVB Digital Video Broadcast

DWDM Dense Wavelength Division Multiplexing

D-WRON Dynamic - Wavelength Routed Optical Network

EMS Element Management System

FDL Fiber Delay Line

FDM Frequency-Division Multiplexing

GST Guaranteed Service Transport

GUI Graphical User Interface

HCT High-Class Transport

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IP Internet Protocol

IPTV Internet Protocol Television

ISP Internet Service Provider

IT Information Technology

KTH Royal Institute of Technology

LTE Long Term Evolution

MSc Master of Science

MTU Maximum Transmission Unit

MUX Multiplexor

NCT Normal Class Transport

NETCONF Network Configuration Protocol

NMS Network Management System

NTNU Norwegian University of Science and Technology

OBS Optical Burst Switching

OEO Optical-Electrical-Optical

OpMiGua Optical Migration Capable Networks with Service Guarantees

OPS Optical Packet Switching

ORION Overspill Routing in Optical Networks

OS Operating System

OTN Optical Transport Network

OXC Optical Cross Connect

PBS Polarization Beam Splitter

PDV Packet Delay Variation

PGA Packet Generator/Analyzer

PLR Packet Loss Ratio

PM Polarization Maintaining Coupler

QoS Quality of Service

RDC Remote Network Connection

RMON Remote Network Monitoring

RPC Remote Procedure Call

SFP Small Form-factor Pluggable

SLA Service Level Agreement

SM Statistical Multiplexing

SNMP Simple Network Management Protocol

SONET/SDH Synchronous Optical Network / Synchronous Digital Hierarchy

SSH Secure Shell

S-WRON Static - Wavelength Routed Optical Network

TDM Time-Division Multiplexing

VLAN Virtual Local Area Network

VLP Variable Length Packet

WDM Wavelength Division Multiplexing

WRON Wavelength Routed Optical Network

XFP 10 Gigabit Small Form Factor Pluggable

XML Extensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network users are not concerned with the intricacies of how the Internet works. They only want to experience fast delivery of content without any problems along the way. Whenever a user complaints about poor application performance, you rarely hear things such as 'the packets are taking a sub-optimal path to the destination' or 'the network utilization is very high'. What they will always say is 'the network is too slow' or 'it takes too much time to load this website or application', etcetera. This is due to the fact that users' complaints are based on their quality of experience when using an application and this quality of experience is almost all the time related to time [Dav08].

"Ideally, we would like Internet services to be able to move as much data as we want between any two end systems, instantaneously, without any data loss" [KR10]. However this is impossible to accomplish in reality. Instead, computer networks introduce *delays* or *latency*, as well as *jitter* or *packet delay variation* (PDV) between source and destination, which impacts negatively in the application performance. The latency and jitter concepts are explained in detail in chapter 2.

1.1 Motivation

While traditional applications such as web browsing, e-mail and file sharing can tolerate more than 100 milliseconds of latency [MRV10], more and more services such as IPTV, high quality video conferencing, remote surgeries, online banking and cloud computing [HWD12] demand low latency and jitter in combination with high bandwidth and low packet loss in order for them to work successfully. These applications are known as *time-sensitive applications*.

Legacy *circuit switching* technologies such as Synchronous Optical Network / Synchronous Digital Hierarchy (SONET/SDH) can offer the necessary Quality of Service (QoS) to enable zero packet loss and low predictable latency, however, since resources are permanently allocated to the established connections between source and destination, there is a low utilization of the links.

On the other hand, using *packet switching*, the link capacity is dynamically shared for traffic with changing intensity and connections with different bandwidth needs. This results in a higher throughput efficiency.

In today's networking demands "there is a need for flexibility to carry different types of services over the same converged network; hence, to enable high network throughput efficiency while still supporting demanding services like" [RVB] time-sensitive applications.

TransPacket is a startup-company that has implemented the novel *fusion* technology, also called the Optical Migration Capable Networks with Service Guarantees (*OpMiGua*) integrated hybrid networks. The fusion concept has the main objective of combining the best properties from both circuit and packet switched networks into a hybrid solution.

It is highly motivating to work on this thesis since it deals with a unique and patented optical networking technology developed by a Norwegian company and whose Chief Executive in Office (CEO) is my main supervisor, professor Steinar Bjørnstad.

1.2 Objective

It is the objective of this thesis work to perform a network experiment involving TransPacket H1 nodes measuring performance parameters like latency, latency variation (or PDV) and packet loss. The whole purpose is to be able to prove that Internet traffic can be separated into high and low priority, and that the former will experience no packet delay variation and no packet loss, while at the same time having almost no waste of resources using TransPacket's Fusion Network approach.

1.3 Scope

This master thesis work is meant to be based on experimental grounds, getting in touch with actual configurations of TransPacket's H1 nodes, setting up the physical connections handling optical fibers, and making tests using a packet generator and analyzer. Due to time and resource constraints, a single network scenario is explored, where both H1 nodes are situated in the same place.

1.4 Methodology

The method I used throughout this thesis can roughly be divided into two parts: the research of scientific papers relevant to this topic; and the hands-on experimentation. Each was then subdivided into different tasks:

Research:

- Background research
- Theoretical research of fusion network
- Previous work and results

Hand-on experimentation:

- Getting acquainted with the laboratory and hardware used
- Set up a network scenario

- Start experimentation
- Gather results and analyze them

Over 40 hours of research and around 60 hours of experimentation was done. Throughout all this time, I wrote this paper in parallel. There were feedbacks and comments from both my supervisors and PhD student Raimena.

1.5 Document Structure

The remainder of this paper is organized as follows:

Chapter 2: Background presents the background theory of circuit switching and packet switching, emphasizing their differences in how they handle data and introduce latency, PDV and ultimately packet loss. A detailed insight on latency is given as well. At the end, an overview of a hybrid solution is presented.

Chapter 3: Fusion Networking describes the principles, properties and main characteristics of an integrated hybrid network known as fusion. It also presents the design characterisitcs of a hybrid node.

Chapter 4: TransPacket H1 Fusion Networking Muxponder presents the capabilities, key features, aggregation properties and management system of the H1 node, and compares it with other known hardware.

Chapter 5: Previous Work shows past experimentations that have already been performed with fusion networking and its results; specifically the ones documented in [RVH] and [RVB] by PhD student Raimena Veisllari, and professors Steinar Bjørnstad and D. R. Hjelme in collaboration with Kurosh Bozorgebrahimi.

Chapter 6: Laboratory Environment lists the hardware and software necessary to perform the experiment as well as the network overview of how such hardware is remotely connected in order to make configurations.

Chapter 7: Network Scenario presents the physical topology chosen to perform the experiments and its logical representation as well as the objective to achieve using this network scenario.

Chapter 8: Experiment Procedure describes the test procedure for carrying out the experiment. From making the physical connections all the way to configuring the H1 nodes and setting up the Spirent TestCenter to generate and analyze traffic from its correspondent ports.

Chapter 9: Results presents the results obtained by the experiments done. A comparison of SM and GST will be made, using plots that graphically represent the results obtained.

Chapter 10: Discussion deals with a more comprehensive discussion based on the results presented in chapter 9. It also presents the minimum requirements in terms of delay in order for certain time-sensitive applications and general data to work properly, and compare them with the GST and SM performance achieved during experimentation.

Chapter 11: Conclusion wraps up and concludes the paper.

Chapter 12: Further Work gives suggestions on how to proceed with this topic or experiments that need to be further investigated since they were not covered in this paper mainly due to time constraints.

Chapter 2

Background

In this chapter, the background theory of circuit switching and packet switching will be presented, emphasizing their differences in how they handle data and introduce latency, PDV and ultimately packet loss. An overview of a hybrid solution will be presented as well.

2.1 Circuit Switching and Packet Switching

There are two fundamental approaches when transferring data through a network of links and switches: circuit switching and packet switching. In circuit-switched networks, the resources such as buffers and link transmission rate needed along a path to provide communication between a source and a destination are *reserved* for the duration of such communication session between them [KR10]. Contrary to circuit switching, in packet-switched networks, these resources are *not* reserved; the messages sent during a session use the resources *on demand*, and as a consequence, may have to wait in a queue before being able to access the communication link [KR10].

The telephone network is the best example of a circuit-switched network. Consider what happens when one person wants to communicate with another person over a telephone network. First, before any information can be sent, the 'caller' needs to establish a connection between him and the recipient by dialing the recipient's phone number. This connection is called a *circuit* [KR10]. Once this circuit has been established, communication can start with a reserved constant transmission rate [KR10] for the duration of the connection. Since bandwidth has been reserved for this circuit, the sender can transfer with a *guaranteed* constant rate.

The Internet Protocol (IP) based Internet is the perfect example of a packet-switched network. The same as with circuit switching, whenever a host wants to send data to another host, this data is transmitted over a series of communication links that connect the sender and the receiver together. However, with packet switching, the packet is sent into the network without reserving any bandwidth whatsoever [KR10]. A link is said to be *congested* when other packets also need to be transmitted over the same link at the same time. If this occurs, then our packet will have to wait in a *buffer* at the sending side of the transmission link, causing it to suffer a delay. And so, "the Internet makes its *best effort* to deliver packets in a timely manner, but it does not make any guarantees" [KR10] like circuit switching does.

2.1.1 Circuit Switching

A circuit-switched network is illustrated in Figure 2.1. In this particular example, there are four circuit switches interconnected by four links. Each link has n circuits, which means each link can support n simultaneous connections. As mentioned previously, whenever two hosts want to communicate with each other, the circuit-switched network has to first establish a dedicated *end-to-end connection* between them. And so, from this example, if Host A wants to communicate with Host B, the network must first reserve one circuit on each of the two links connecting them together. This is where it gets interesting; "because each link has n circuits, for each link used by the end-to-end connection, the connection gets a fraction $1/n$ of the link's bandwidth for the duration of the connection" [KR10].

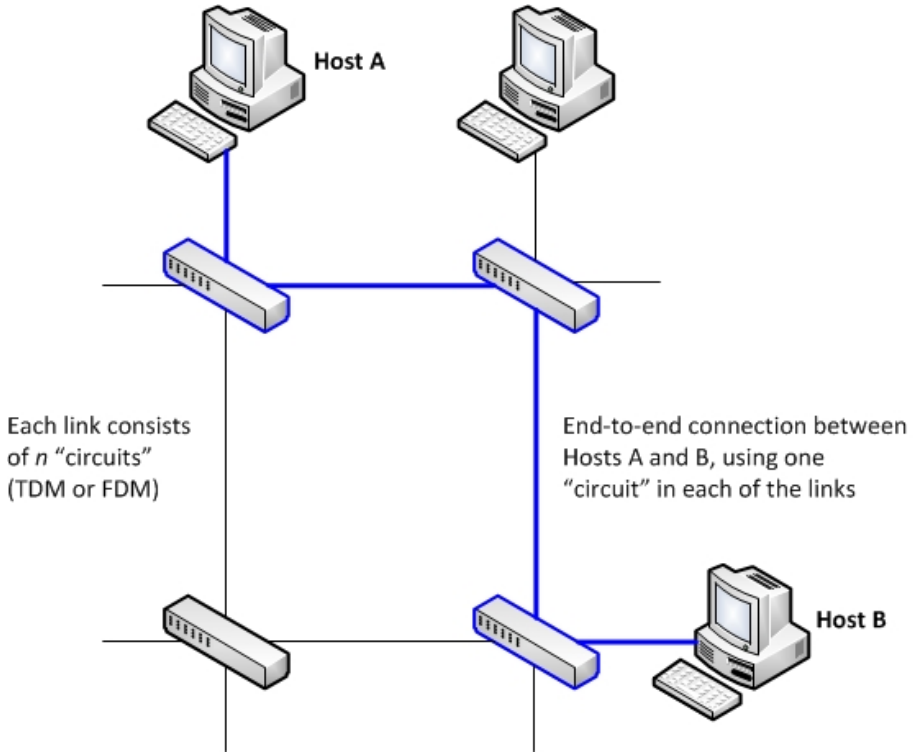

Figure 2.1: A simple circuit-switched network consisting of four switches and four links.

But how can a single link carry n circuits or channels? The answer is *multiplexing*. Multiplexing is the process of subdividing a link into multiple channels for resource sharing. As seen in Figure 2.2, a multiplexor or MUX has n inputs, and 1 output with n channels; whereas a demultiplexor or DEMUX has 1 input with n channels and separates them into n outputs.

Figure 2.2: A MUX-DEMUX example.

This way, multiple sender/receiver pairs can share the same link. There are two ways to implement this: using either *frequency-division multiplexing* (FDM) or *time-division multiplexing* (TDM).

Using FDM, "the frequency spectrum of a link is divided up among the connections established across the link" [KR10]. This means that each connection within a single link has a *frequency band* dedicated to it for the duration of the connection. This frequency band is said to have a specific width, for example, in telephone networks this width is of 4 kHz (4,000 hertz) [KR10], and therefore this is most commonly known as *bandwidth*.

With TDM, the transmission is divided into time slots. Once the connection between sender and receiver has been established, the network dedicates one time slot in every frame for the sole use of that connection [KR10]. Figure 2.3 illustrates both FDM and TDM for a link that supports up to four circuits. For FDM, the frequency domain is segmented into four bands, each with a bandwidth of 4 KHz. For TDM on the other hand, the time domain is segmented into frames with four time slots in each frame.

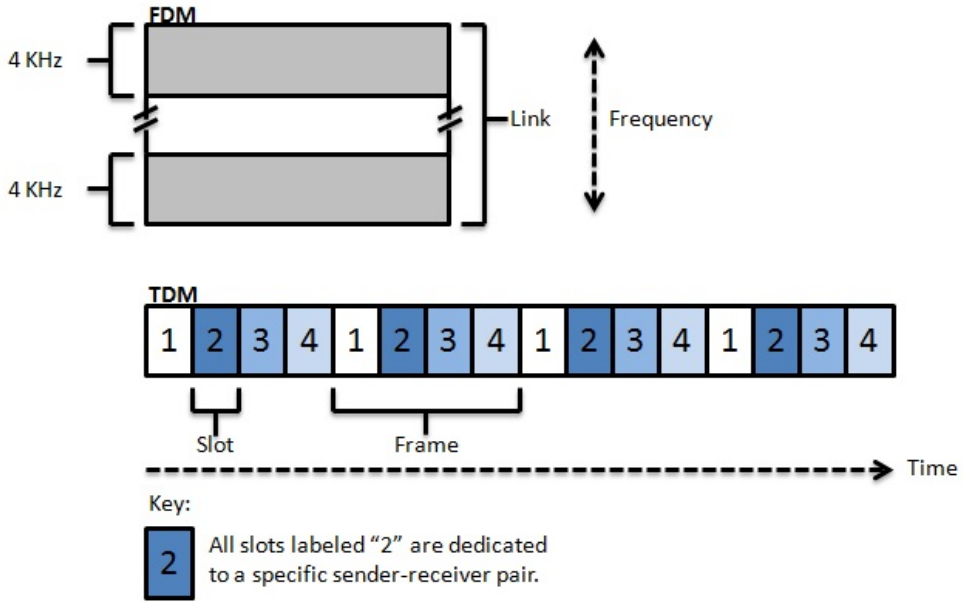

Figure 2.3: With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time.

2.1.1.1 Circuit Switching Drawback

Circuit switching is a wasteful technology because of the fact that the dedicated circuits are idle during *silent periods* [KR10]. This means that the network resources, whether they are frequency bands or time slots in the links along the connection's path, cannot be used by any other ongoing connections, even though they are not being used. As a result, network resources are being wasted and so the network capacity is not used efficiently.

2.1.2 Packet Switching

Modern computer networks break long messages into smaller chunks of data known as *packets* [KR10]. These packets travel between source and destination through the communication links, however, unlike circuit switching, they can take different routes since a connection or circuit does not need to be created before transmission starts. And so, packets are transmitted with the full transmission rate of the link.

Most routers and link-layer switches use a transmission approach called *store-and-forward*. This means that the switch must receive the entire packet before it can begin to transmit the first bit of that packet onto the outbound link [KR10]. This process introduces a delay at the input link of each switch or router along the route from source to destination.

We still need to go deeper into how forwarding works, in order to understand the source of delay in packet-switched networks. Figure 2.4 shows the architecture of a router with all of its components.

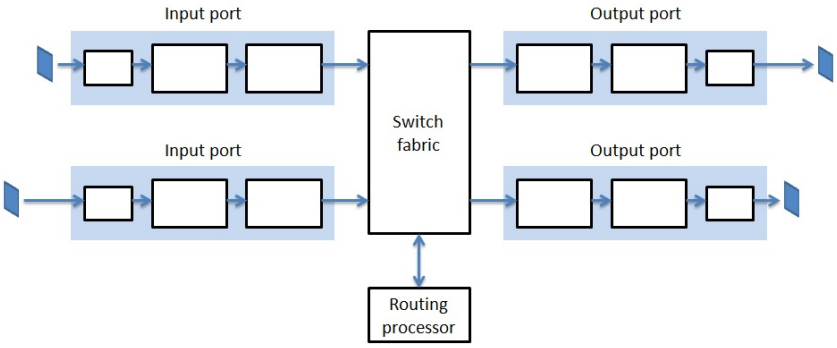

Figure 2.4: Router architecture.

The *switching fabric* simply connects the router's input ports to its output ports. The *routing processor* executes the routing protocols (out of the scope for this paper) which feed information to the forwarding table [KR10]. The *input* and *output ports* need to be described in detail.

Figure 2.5 shows a detailed view of the input port with all of its functions. The *line termination* module and *link processing* module are the interfaces for the physical and data link layers [KR10].

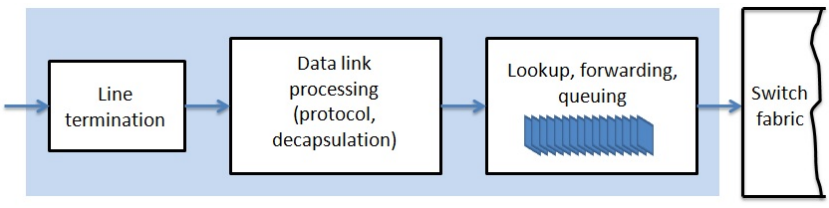

Figure 2.5: Input port functions of a router.

The *lookup/forwarding* module is in charge of deciding which output port to send it to by looking at the forwarding table. Once the router knows which output port to forward the packet to, it is sent to the switching fabric. This is where it gets interesting. A packet

may be temporarily blocked from entering the switching fabric [KR10] if there are many other packets using it coming from the other input ports. A blocked packet is then moved to a queue at the input port and scheduled to cross the switching fabric after some time, causing delay.

Furthermore, a switch or router has an *output buffer* for every link attached to it, which stores the packets that are to be sent out on that specific link. The idea is that if a packet arrives at a switch and is meant to be transmitted out of one of the outbound links, and this link is busy with the transmission of another packet, then the arriving packet must wait in the output buffer [KR10], adding another type of delay called *queuing delay* (see section 2.2). This delay is very complicated because it is variable and depends entirely on how congested the network is and the size of the buffers. If a buffer is completely filled with other packets waiting to be transmitted, the arrival of a new packet will cause packet losses to occur.

Once the output port receives a packet, it goes through a queue if the switch fabric delivers the packets at a higher rate than the output link rate (see Figure 2.6); process which also adds latency. The link processing and line termination modules are also needed as in the input port.

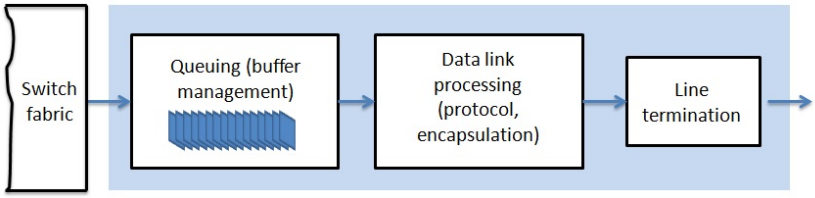

Figure 2.6: Output port functions of a router.

Suppose that the input line and output line speeds are identical and that there are n input ports and n output ports. Also suppose that the switching fabric speed is n times faster than the line speeds. In this case, there will not be queues at any input line, since the worst case scenario is that all n input lines are receiving packets, but the switch will be able to transfer n packets from input to output ports in the time it takes each of the n input ports to simultaneously receive a single packet [KR10].

Unfortunately this is not as simple in the output ports because in the worst case scenario, all the packets from the n input ports are destined to the same output port. And since only a single packet can be transmitted at a time, the n packets will have to wait in a queue before being transmitted out into the network [KR10]. If the number of packets in a single output port queue is too big, the router will start to drop packets.

Figure 2.7 illustrates an example of a packet-switched network. As it can be seen, hosts A and B are sending packets to host E. First they send their respective packets along 10 Mbps Ethernet links to the first packet switch. This switch will then forward these packets to the 1.5 Mbps link. The problem arises when the arrival rate of the packets exceeds the rate at which these packets can be forwarded out, and so congestion occurs. There is a detailed explanation of all the kinds of delays introduced in packet switching on section 2.2.

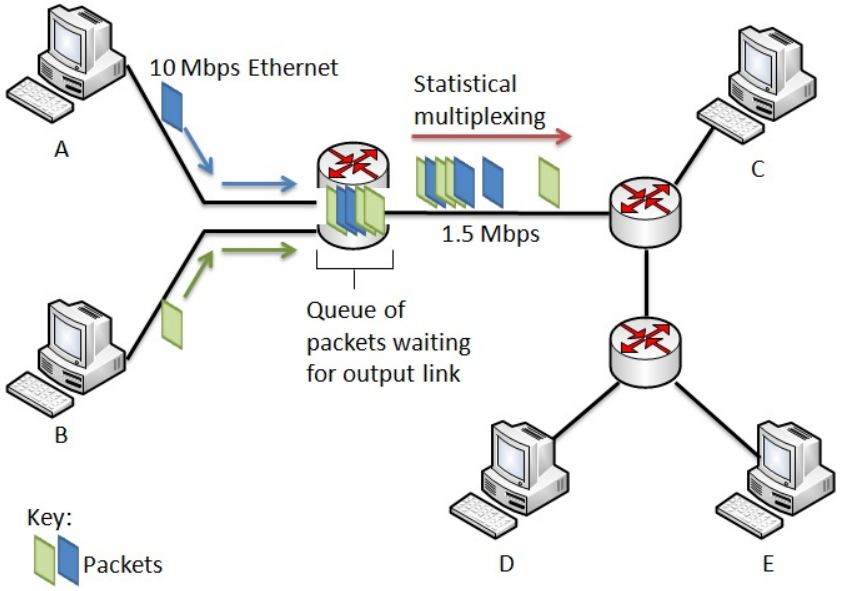

Figure 2.7: A simple packet-switched network with two sources sending packets through the same router creating a queue.

2.1.2.1 Packet Switching Drawback

Because of the output port queuing, a *packet scheduler* at the output port needs to choose which packet is to be sent next, and which other ones should remain in the queue. This process is the key to provide *quality-of-service* (QoS) guarantees such as minimizing delays. Unfortunately, the Internet today as it uses mainly IP, provides a best-effort service, which means IP does not care for minimizing end-to-end delay nor minimizing PDV.

"The Internet has mostly taken an egalitarian approach to packet scheduling in router queues. All packets receive equal service; no packets, including delay-sensitive packets, receive special priority in the router queues" [KR10].

2.2 Contributors of Latency in Packet-Switched Networks

Whenever a packet travels from one node to the next, the packet suffers from several types of delay at each node along the way. These delays include the *processing delay*, *queuing delay*, *transmission delay* and *propagation delay* [KR10] as seen in Figure 2.8. The sum of all of these variables results in the *total nodal delay*. Let us closely examine each one of these delays in the context of Figure 2.8 to have a better understanding of them. As part of its end-to-end route between source and destination, a packet is sent from the sender through router A to router B. Our goal is to characterize the nodal delay at router A.

When a packet arrives at router A, it must examine the packet's header information to determine the proper outbound link that will lead to the destination and then it directs the packet to this link. In this example, there is only one outbound link which leads directly to router B. Remember also that a packet can only be transmitted on a link whenever there is no other packet being transmitted on this link and if there are no other packets in front of the queue in the router's buffer [KR10].

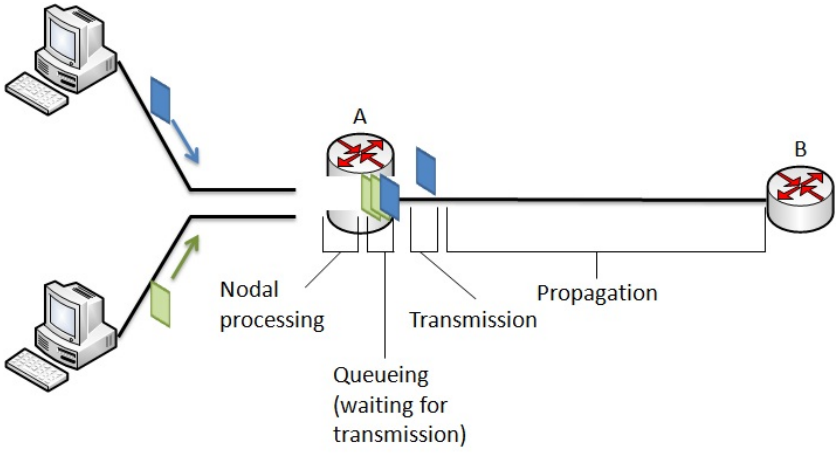

Figure 2.8: Total nodal delay at router A.

Processing Delay It is the amount of time the node takes to look at the packet's header and perform a destination address lookup to determine where to forward the packet [KR10]. It can also include the time it takes to check for bit-level errors in the packet. This kind of delay in modern high-speed routers is almost insignificant, in the order of microseconds or less [KR10]. After the node knows which outbound link to use, the router sends the packet to the queue that precedes the link to router B.

Queuing Delay It is the time the packet has to wait in the queue (at the buffer) before being transmitted onto the link. The queuing delay is naturally dependent on the number of packets that are queued before the specific packet. If the queue is empty and there is no packet being transmitted then the queuing delay is zero. On the other hand, if the traffic is heavy and the queue is long, the queuing delay will be large [KR10]. For time-sensitive applications the queuing delay is large. Queuing delays are on the order of microseconds to milliseconds [KR10].

Transmission Delay Assuming that packets are transmitted in a first-come-first-served basis, a packet can only be transmitted after all the packets in front of the queue have been transmitted. In order to calculate the transmission delay we need to define two variables; the *length of the packet* denoted by L bits [KR10], and the *transmission rate* of the link from router A to B denoted by R bits/sec [KR10]. Transmission delay is then L/R. This kind of delay is usually on the order of microseconds to milliseconds as well [KR10].

Propagation Delay Once a bit has been pushed into the link during the transmission, it then needs to propagate all the way to router B. Propagation delay can be defined as the time it takes for the bit to propagate or travel from the beginning of the link to router B [KR10]. Of course, the propagation speed will entirely depend on the physical medium being used. Three major mediums are fiber optics, copper cables or via satellite. Propagation delay is then d/s where d is the distance between router A and B, and s is the propagation speed of the link. This kind of delay depends on the distance between source and destination, but in general, in wide-area networks it is on the order or milliseconds [KR10].

It is very common for people to not being able to differentiate between transmission delay

and propagation delay. Transmission delay is the time it takes for the router to push the packet out and it is a function of the packet's length and the link transmission rate, but has nothing to do with the distance between the two routers. Propagation delay is the time it takes for the packet to go from one router to the next and it is a function of the distance between the two routers, but has nothing to do with the packet's length or the link transmission rate.

In order to understand this better, we can create an analogy. Let us assume there is a highway that has a tollbooth every 100 km as seen in Figure 2.9. The highway represents the link, whereas the tollbooths represent the routers. Let us also assume cars travel (propagate) at a constant speed of 100 km/h. 10 cars are travelling in a caravan, that is, they follow each other in a fixed order. Each car represents a bit and the whole caravan represents one packet. Another assumption is that each tollbooth services (transmits) a car at a rate of one car every 12 seconds. In this example, let us pretend there are no other cars in the highway, which means there is no queuing delay.

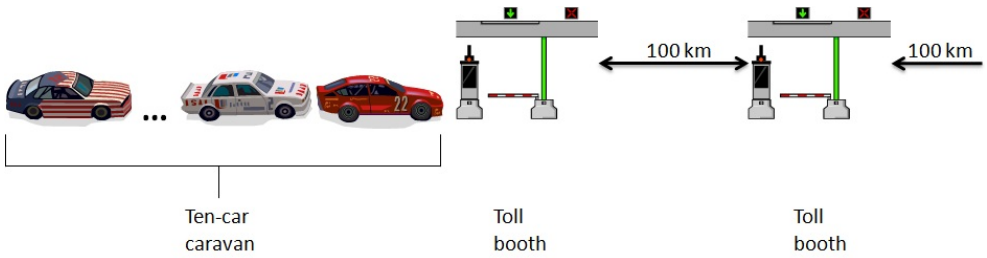

Figure 2.9: Caravan analogy.

The transmission delay (time for the tollbooth to push the caravan out onto the highway) is

$$\frac{L}{R} = \frac{10bits(cars)}{5bpm(carsperminute)} = 2minutes$$

The propagation delay (time for the cars to get from one tollbooth to the next) is

$$\frac{d}{s} = \frac{100km}{100km/h} = 1hour$$

So the time it takes for the caravan (packet) when is stored at the front of the tollbooth until it is stored in front of the next one is the sum of the transmission delay and the propagation delay, which in this case it is 62 minutes.

To summarize and as mentioned at the beginning of this section, the total nodal delay is the sum of the processing delay, queuing delay, transmission delay and propagation delay, which can be expressed as follows

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

For the purpose of this paper, we will only focus on queuing delay (d_{queue}) and propagation delay (d_{prop}) for the following reasons. The contribution of each of these delays can vary significantly from being negligible to being a dominant player in the total delay. However, d_{proc} is insignificant nowadays in modern routers when configured correctly [Dav08], so it is often negligible. So is the case for d_{trans}, which becomes insignificant with data rates of 10 Mbps and higher [KR10], which in today's networking environment, almost everyone has.

The d_{prop} can of course be negligible as well when the source and destination are in the same network or close together, however in a real life environment, it is very likely the transmission has to travel half around the world; and it is also an opportunity to compare different physical layer technologies and how they influence on this delay. By far the most complicated and interesting delay is d_{queue}, and in an unrealistic perfect environment, it could be negligible. The d_{queue} can be different for each packet and so in order to measure this delay, experts use statistics [KR10].

Last but not least, it is important to mention that it is highly unlikely that the end-to-end latency remains constant. There is a phenomenon called *PDV* explained in RFC3393 [IET02], which is most commonly known as *delay jitter*. This term "refers to the variance in the arrival rate of packets from the same data flow" [Dav08], which means the time from when a packet is generated at the source until it is received at the destination can change from packet to packet of the same message as seen in Figure 2.10.

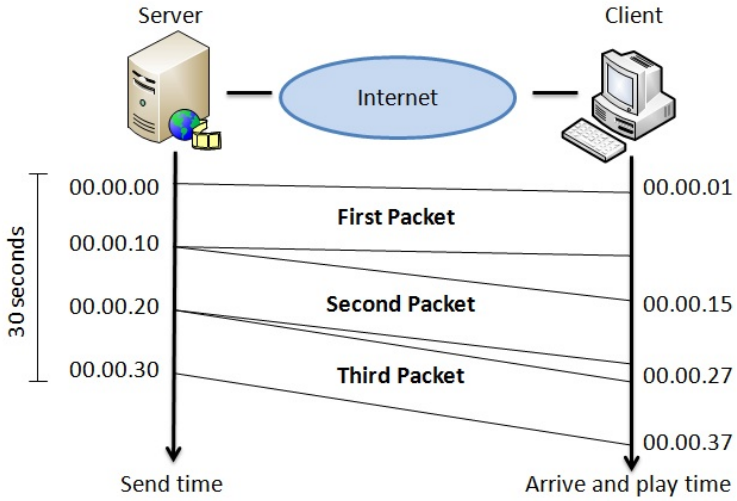

Figure 2.10: Delay jitter between source and destination.

Let us consider two consecutive packets. The sender sends the second packet 20 milliseconds after the first packet, however the second packet arrives more than 20 milliseconds after the first one at the receiving end. This could happen for instance if the first packet arrived at an almost empty queue at the destination and just before the second packet arrived at the queue, a large number of packets from other sources arrive at the same queue, which means the second packet has a longer queuing delay. It is important to note that it could also be the case that the second packet arrives less than 20 milliseconds apart, and this would also count as delay jitter.

The three main reasons delay jitter exists are:

- Variance in transmission delay due to variance in packet sizes [Dav08] (different L)
- Variance in queuing delay due to packet spacing from multiple sources at a common outbound link [Dav08] (as seen in the example above)
- Packets taking different routes to reach the destination due to load balancing or maybe some routing issues

2.3 Packet Switching versus Circuit Switching

Now that both packet switching and circuit switching have been thoroughly explained, can we answer the question, which one is better? This is a tricky one and I believe there is no absolute correct answer. Critics of packet switching argue that this technology is not suitable for real-time services or time-sensitive applications because of its variable and unpredictable end-to-end delays [KR10], which make it impossible to offer any QoS guarantees as the ones offered by circuit switching. On the other hand, packet switching proponents argue that it offers better sharing of bandwidth than circuit switching and that is it simpler, more efficient and less costly to implement [KR10].

But why exactly is packet switching more efficient? The best way to make this clear is by using statistics based on an example. We have the following assumptions:

1. There are 35 users are sharing a 1 Mbps link.
2. Each user generates data at a constant rate of 100 kbps whenever it is active, and generates no data at all in periods of inactivity.
3. A user is active only 10 percent of the time.

With circuit switching, this is easy to calculate. Since each user requires one tenth of the bandwidth to be reserved at all times even if the user is not active, this circuit-switched link can support only 10 simultaneous users (the other 25 will not be able to transmit data), that is

$$\frac{1Mbps}{100kbps} = 10users$$

With packet switching however, this calculation is not so straightforward. We know that the probability that a specific user is active is 0.1 (10%). Using the *binomial distribution*, the probability that there are 11 or more simultaneous active users (out of 35) at a given time is

$$1 - \sum_{n=0}^{10} \binom{35}{n} p^n (1-p)^{35-n}$$

Where $p = 0.1$ and n ranges from 0 to 10. I decided to calculate it this way since this implies making the calculation 11 times, instead of 25 times (from 11 to 35). This is why in the end; we need to subtract the result from 1. And so the result is 0.0004.

We can then calculate that the probability of having 10 or fewer simultaneous active users at a given time is $1 - 0.0004 = 0.9996$. This means that with 99.96% probability, the aggregate arrival rate of data is less than or equal to 1 Mbps, and so packets will flow through the link without any delay as is the case with circuit switching. Of course, when there are more than 10 simultaneous active users, then packets will start to queue at the output buffer until the aggregate input rate falls back below 1 Mbps. And since the probability of this happening is 0.04%, we can say that in this particular example, packet switching provides essentially the same performance as circuit switching, but "does so while allowing for more than three times the number of users" [KR10].

Another example that clarifies how packet switching is more efficient is as follows. Let us assume there are 10 users, one of which suddenly creates one thousand packets, each of 1,000 bits. The other 9 users are idle. If the network is TDM circuit switching with 10 slots per frame (1 frame, 1 second) and each slot consists of 1,000 bits, then the active user can only use its allocated time slot. So the user sends 1,000 bits and waits for the other 9 time slots to pass even though they are not being used to send the following 1,000 bits. It will take 10 seconds to transmit the entire data. If packet switching was being used, then the active user can continuously send its packets at the full link rate of 1 Mbps, since no one else is demanding any bandwidth, and so it will take the user 1 second to transmit its data.

Both examples above show how packet switching performance can be better than that of circuit switching. It is basically because of the on-demand approach of sharing resources with packet switching, which has come to be known as *statistical multiplexing* (SM) [KR10]. However, if we were to saturate the network by having a lot of users and they are active most of the time, we would see that packet switching would introduce increasing delay to the point of overloading the network.

2.4 Hybrid Optical Network Architectures

What about a *hybrid optical network architecture* that combines the advantages of circuit switching and packet switching, while at the same time avoids their disadvantages? According to [GPJKD06], a hybrid network architecture is one that "does not apply one network technology to transport all traffic, but instead combines several switching technologies into one architecture". It is important to note that I have used the word optical, and this is because the focus of this paper is a hybrid technology using optical media.

In [GPJKD06], three different classes of hybrid optical networks have been identified. In order of the degree of interaction and integration (from least to most) of the networking technologies, these are:

1. Client-server
2. Parallel
3. Integrated

2.4.1 Client-Server Hybrid Optical Network

As its name implies, this class of hybrid optical network uses a hierarchy of optical layer networks, where the lower layer "functions as a server setting up a virtual topology for the upper client layer" [GPJKD06]. The client layer is an *optical burst switching* (OBS) or *optical packet switching* (OPS) network, whereas the server layer is a wavelength switching network.

And so, the OBS or OPS nodes are in charge of aggregating traffic. These nodes are interconnected by direct lightpaths at the server layer, which would work as a circuit-switched network. In other words, "optical bursts or packets are switched only in the client layer nodes and transparently flow in lightpaths through the circuit-switched server layer nodes" [GPJKD06]. Figure 2.11 shows such a hybrid network.

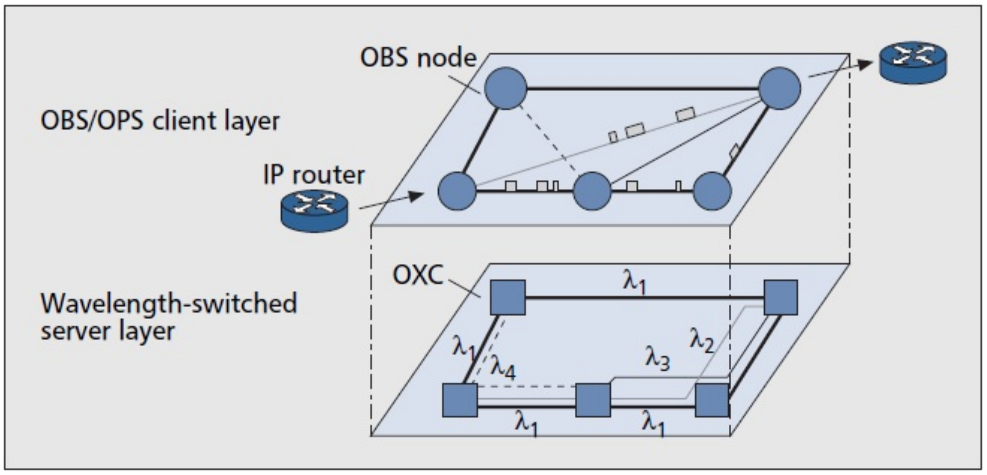

Figure 2.11: Client-server hybrid optical network [GPJKD06].

However, this particular hybrid architecture does not solve the problem. According to [GPJKD06], only low network utilization is achieved because to increase connectivity, this architecture uses a virtual topology which yields less traffic per link and thus reduced multiplexing gain.

2.4.2 Parallel Hybrid Optical Network

In this class of hybrid architecture, "two or more optical layer networks, offering different transport services, are installed in parallel" [GPJKD06]. And then, an intelligent edge node decides whether to use them individually or combine them to optimally serve customer service requirements. This decision is made based on explicit user request, traffic characteristics or QoS requirements. Figure 2.12 shows the representation of a parallel hybrid optical network.

On the downside, "the design of such parallel hybrid architectures has to trade-off efficiency and realization complexity" [GPJKD06].

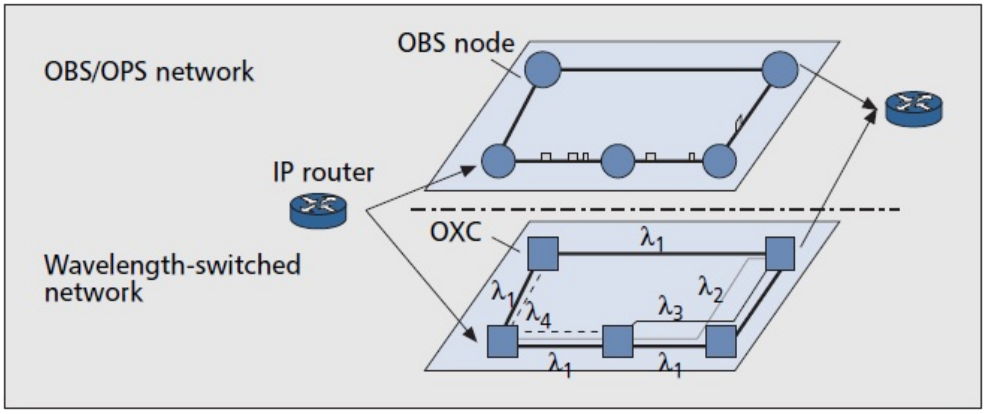

Figure 2.12: Parallel hybrid optical network [GPJKD06].

2.4.3 Integrated Hybrid Optical Network

Last but not least, comes a class that completely integrates both technologies. They share the network resources such as bandwidth simultaneously. So for example, each node in an integrated hybrid network can choose to send traffic wavelength-switched through a predetermined connection or circuit; or ignore that circuit path and process the traffic in a packet-switched manner.

Figure 2.13 shows this kind of hybrid network. As you can see, each node has both a packet-switched and a wavelength-switched device. How do they decide which one to use? One option is to transmit all packets over the end-to-end lightpath since it removes the need for intermediate processing [GPJKD06]; and when congestion occurs, the node can switch to packet switching. A second option is to use the wavelength-switched device for high-priority traffic and the packet-switched one for the rest to achieve a certain QoS.

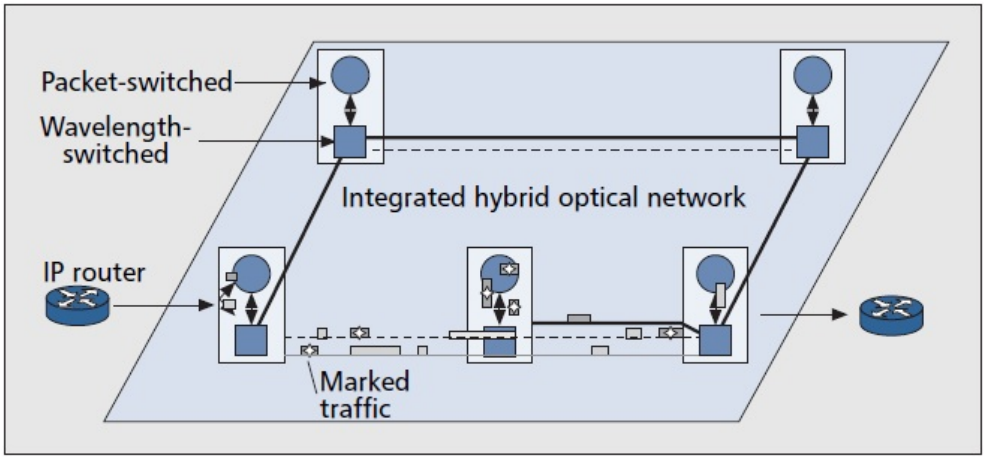

Figure 2.13: Integrated hybrid optical network [GPJKD06].

This method is the most resource efficient, however it is also the most complex. According to [GPJKD06], there are only two proposals of integrated hybrid optical networks: the OpMiGua and *Overspill Routing in Optical Networks* (ORION). This thesis work is entirely focused on the OpMiGua approach, also known as *Fusion Networking*.

Chapter 3

Fusion Networking

IP traffic increase has resulted in a demand for greater capacity of the underlying Ethernet network. As a consequence, not only Internet Service Providers (ISPs) but also telecom operators have migrated their mobile back-haul networks from legacy SONET/SDH circuit-switched equipment to packet-based networks.

This inevitable shift brings higher throughput efficiency and lower costs; however, the guaranteed QoS and minimal delay and PDV that can only be offered by circuit-switched technologies such as SONET/SDH are still essential and are becoming more vital for transport and metro networks, as well as for mobile back-haul networks, as the range and demands of applications increase.

This chapter describes a proposed solution for combining the best characteristics of packet switching and circuit switching into a single architecture called fusion networking.

3.1 Fusion Networking Principle

Fusion Network developers have cleverly explained the principle behind this technology using an interesting analogy in [Tra12b]. The analogy compares circuit, packet and fusion networking and it is summarized below:

First of all, imagine there is a direct express train from A to B with no intermediate stops whatsoever. This kind of direct train represents the properties of circuit switching: there is minimum latency, minimum PDV and no packet (passenger) loss. This is because the passengers that get on the train at station A are guaranteed a ride all the way to their destination B without any stops. However, as it is with a network, if the express train is not even remotely full since there are only a few passengers wanting to go from A to B, the train capacity will not be used efficiently.

How can this be avoided? There must be plenty of passengers in intermediate stations wanting to go to the same destination B or any other destination on the way to B, but with circuit switching, the train ignores these passengers. With packet switching on the other hand, the train stops at every intermediate station. On the downside, at each station the passengers already onboard must leave the train and queue together with the passengers that were already at the station. After they have all been checked in they are allowed to go into the train until the seats are all occupied. As expected, train capacity is more

efficiently used since passenger flows from all stations are aggregated and seats vacated by any passenger leaving the train are likely to be occupied by new passengers joining at intermediate stations. As a consequence of improving the train capacity efficiency, the travelling time from A to B has increased greatly by introducing different delays (PDV) at each station; and some passengers may not even be able to get in the train because there are no more free seats (packet loss).

Then fusion networking appeared. Let us continue with the train analogy, but it is important to remark that this is unfeasible for actual passengers. We again have the same direct express train for *first class* passengers from A to B, and this train does not stop at any of the intermediate stations. At each intermediate station there are *second class* passengers queuing and can enter or leave the train while in speed. If the train is full at any given station (meaning capacity is used efficiently), the passengers waiting at such station will not be able to get in the train, and still the first class passengers do not lose their seats (no packet loss) and experience a minimum fixed delay (no PDV). The best of both worlds have been combined in what is known as fusion networking.

3.2 Fusion Networking Properties

Fusion networking as originally defined by TransPacket in [Tra12a] "combines the best from packet switching and circuit switching in a unique way ... into a fully Ethernet compliant network without the use of legacy circuit-switching techniques. The combination of the properties enabled across the network is unique in the market: high throughput, zero packet loss, ultra-low latency and ultra-low latency variation".

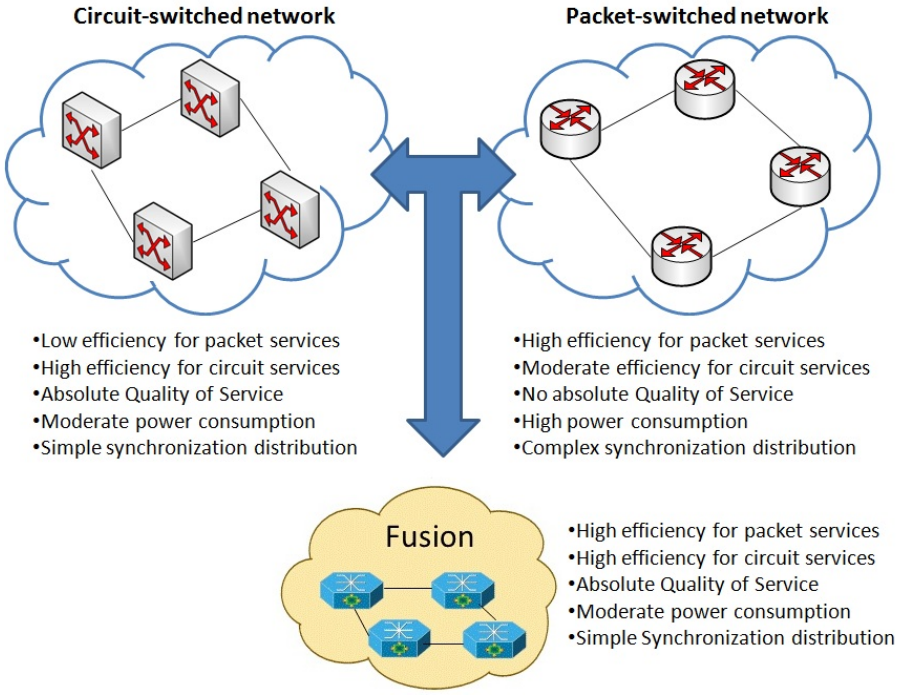

Figure 3.1: Combining the best properties from packet and circuit switching into the fusion network.

This way the increasing traffic demands on fixed networks (e.g., video services) and on mobile networks (e.g., Long Term Evolution (LTE) and its strict demands of synchronization and timing [Tra12b]) can be properly served. Figure 3.1 shows the best properties of packet and circuit switching, and which of those properties have been combined to create the fusion network.

But how is this accomplished? The fusion network technology divides the traffic into two *service classes* while still using the capacity of the same wavelength in a *wavelength routed optical network* (WRON) [SBS06]:

1. A *Guaranteed Service Transport* (GST) service class supporting QoS demands such as no packet loss and fixed low delay for the circuit-switched traffic.

2. A *statistical multiplexing* (SM) service class offering high bandwidth efficiency for the best-effort packet-switched traffic.

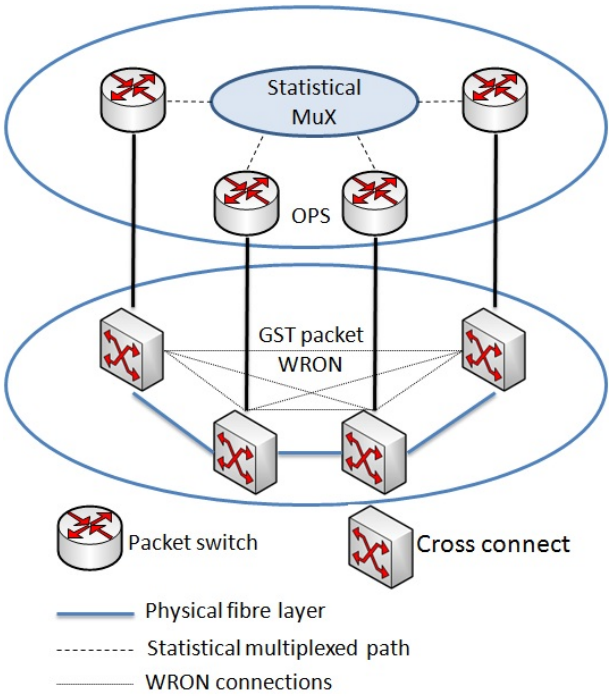

Figure 3.2: A fusion network model illustrating the efficient sharing of the physical fiber layer. The WRON can either be static (S-WRON) or dynamic (D-WRON). The OXCs and packet switches are physically co-located as separate units with a common control unit, or they can be integrated sharing physical resources in the node.

The basic idea of fusion networking is to divide the resources in time without using timeslots [RVB] with packet granularity [SBS06]. Basically, GST packets follow pre-assigned wavelength paths in a WRON through optical cross connects (OXCs) from sender to destination; while the SM packets are switched in packet switches according to their header information in an OPS. A simple representation of a fusion network can be seen in Figure 3.2. The lightpaths are created by interconnecting fibers and wavelengths through OXCs.

High throughput efficiency is ensured by *interleaving* the SM packets with the GST packets [SBS06]. This is achieved by following two simple rules: 1) GST packets in a traffic flow do not contend with any other GST packets of other flows since there is at least one assigned wavelength for each source-destination combination [Vei10]. 2) GST packets following the WRON path are given absolute priority when contending with SM packets [SBS06].

The result is a network that offers "both an Ethernet wavelength transport and the ability to exploit vacant wavelength capacity using statistical multiplexing *without interfering* with the performance of the wavelength transport" [RVH].

3.3 Transparent Ethernet

Packet Loss Ratio (PLR) of SM packets may be improved despite the absolute guarantee of the GST traffic [SBS06] by the bypassing of the packet switches from the GST packets, since it lowers the processing overhead of the intermediate nodes.

Bypassing of routers and switches also helps saving costs in optical networks since, according to [Tra11b], "typically as much as 70% of the traffic is transit traffic". In optical networks, bypassing is traditionally achieved by using optical add/drop network elements that allow the bypass of one or more real wavelengths.

TransPacket transmission equipment uses however, another approach. They use *virtual wavelengths* for bypassing intermediate nodes. There are two differences between real wavelengths and virtual wavelengths. "Unused bandwidth may at all times be employed by intermediate nodes along the virtual wavelength path. Secondly, the granularity of a virtual wavelength is dynamic, allowing statistical multiplexing and full capacity utilization" [Tra11b]. Figure 3.3 shows the bypassing of an intermediate router with virtual wavelengths.

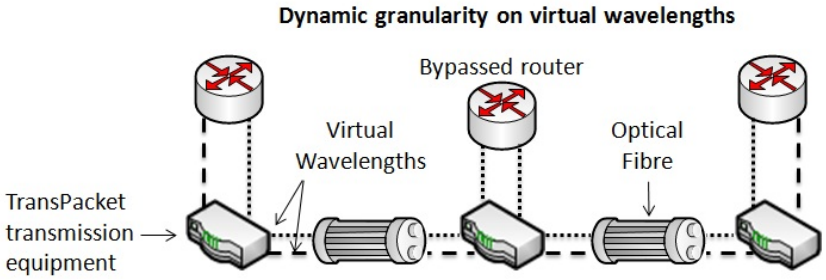

Figure 3.3: Bypassing using virtual wavelengths.

Operators are struggling with an increasing demand for bandwidth to be met within the same budget of cost [Tra12a]. And so, bypassing expensive routers using TransPacket's fusion technology is the most cost-effective method for upgrading network capacity without compromising QoS.

3.4 Hybrid Asynchronous Node Design

Let us now dig deeper into the actual hybrid node design. A simple functional diagram of a hybrid asynchronous node can be seen in Figure 3.4. There are three main design characteristics that need to be taken into account:

1. GST and SM packet separation and combination
2. GST priority
3. SM QoS differentiation

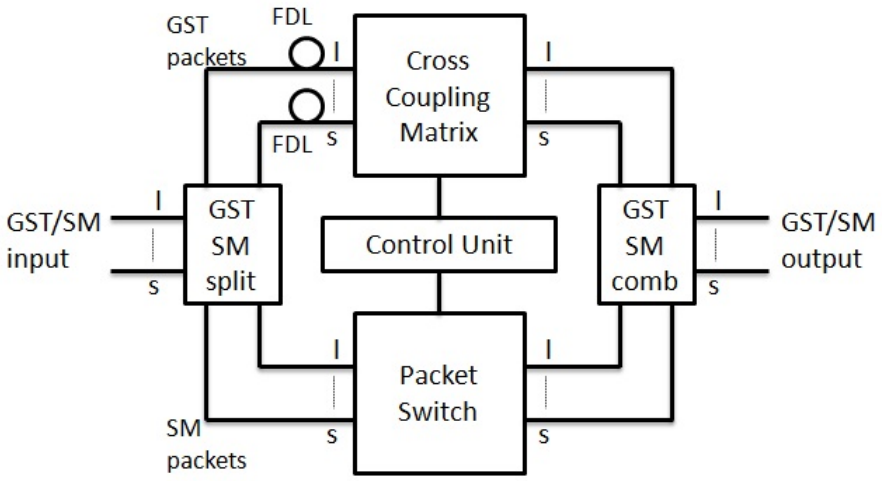

Figure 3.4: Functional diagram of an asynchronous hybrid node. The number of inputs is given as $s = n * N$, where n is the number of wavelengths per link and N is the number of fibers. GST packets are delayed in FDLs to avoid contention between GST and SM packets.

3.4.1 GST and SM Packet Separation and Combination

As shown in Figure 3.4, at the input of the node, packets are divided into the two main classes mentioned in section 3.2: the GST packets forwarded through a WRON cross-coupling matrix; and the SM packets switched with a packet switch module [SBS06]. And then at the output of the node both GST and SM packets are combined. But how exactly does the node separate and brings these packets together?

A couple of approaches have been proved to work when it comes to separating and combining GST and SM packets. On [SBS06], they use *polarization* separation and combination mechanisms. At the input interface, a *polarization beam splitter* (PBS) is assigned for each wavelength [Vei10], separating GST and SM packets. At the output interface, a *polarization maintaining coupler* (PM) combines them into a single wavelength again, ready to be sent out in a fiber. Another approach is to use *VLAN-tags* to identify and separate GST and SM packets.

3.4.2 GST Priority

There are two kinds of packet switching; it can be either asynchronous or synchronous. The former deals with variable length packets (VLP) and a random arrival time with no alignment [SBS06]; the latter deals with fixed length packets, each of which has a time slot assigned to them. It is important to know the difference, since this difference defines how they guarantee GST packet priority.

For asynchronous packet switching, GST priority is ensured by reserving the destined node output when a GST packet arrives at the node input [SBS06]. The traditional way to do this is to purposely delay the GST packets using *fiber delay lines* (FDLs) before they enter the cross-coupling matrix as seen in Figure 3.4. The delay time must be equal to the longest SM packet, that way, at the time the GST packet reaches the node output; it will always be empty, avoiding the contention between GST and SM packets.

For synchronous packet switching, ensuring GST priority is simpler. Since it is a time-slotted design, all node outputs are always free at the end of each time slot [SBS06]. Priority of all arriving packets in a time slot needs to be checked, and possibly reject SM packets contending with GST packets.

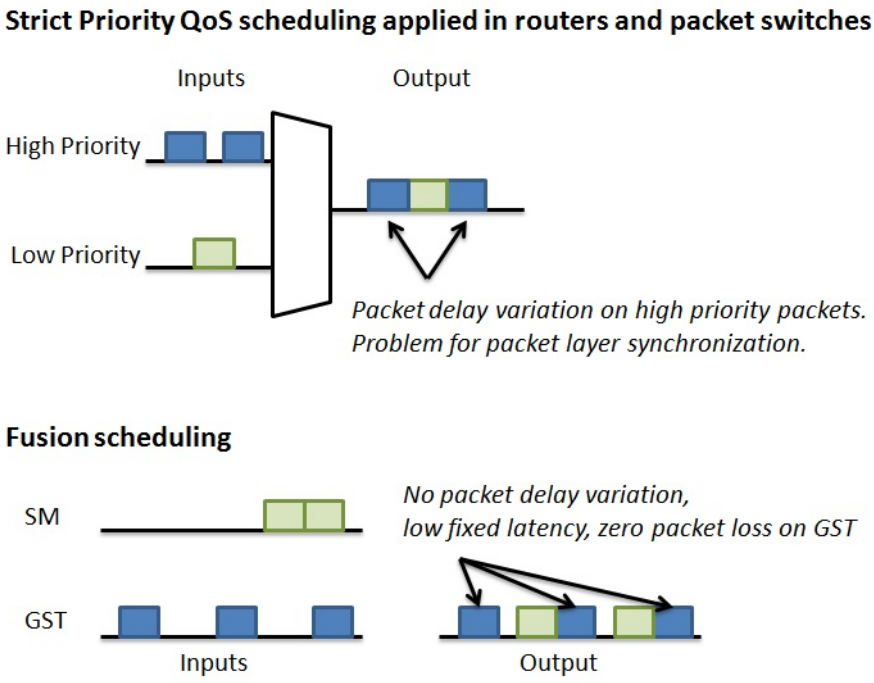

Figure 3.5: Upper figure shows the strict priority QoS scheduling of packet switches and routers. If two packets of low and high priority arrive at the same time, the low priority packet will have to wait until the high priority queue is empty in order to be scheduled. However, if a high priority packet suddenly arrives while a low priority packet is being scheduled, it will have to wait until the low priority packet has been scheduled, causing PDV on high priority packets. Fusion scheduling, shown in the lower figure, avoids PDV on high-priority GST packets since low-priority SM packets are inserted only if there is a free gap between GST packets.

Whether it is an asynchronous node or a synchronous one, these methods will produce the same result: a network where low-priority SM packets are inserted *only if* there is a free gap between high-priority GST packets, hence these GST packets will have a low fixed latency (no PDV) and no packet loss. Figure 3.5 shows the difference between normal scheduling and this fusion scheduling.

3.4.3 SM QoS Differentiation

As explained in section 3.4.2, GST packets have a higher priority than SM packets. However, in [SBS06], SM packets are further divided into two QoS classes: the *high-class transport* (HCT) bearer service and the *normal class transport* (NCT) bearer service. This differentiation between HCT and NCT classes is performed in an electronic buffer within the asynchronous node.

Why is such differentiation needed? An important goal in any node design is to reduce costs [SBS06]. In order to do so when using an electronic buffer for SM packets, the number of interfaces needs to be kept at a minimum, which also means reducing the number of optical to electronic to optical (OEO) interfaces. Dividing SM packets into HCT and NCT increases buffer resource utilization, thus reducing the number of required buffer interfaces.

HCT class packets are given priority over NCT class packets, and are scheduled from the buffer as soon as a wavelength to the destination becomes vacant. This scheme, called *buffer priority* (BP) [SBS06], minimizes delay in comparison to NCT class.

What about service differentiation regarding packet loss? The HCT class packets have access to all the buffer inputs, whereas the NCT class packets have limited access. This means that a packet that belongs to the HCT class has a higher probability to be buffered than one that belongs to the NCT class. And so, according to [SBS06], HCT class has a low PLR of 10^{-6} and NCT a moderate PLR of 10^{-3}.

Chapter 4

TransPacket H1 Fusion Networking Muxponder

Knowing how fusion networking works and before going into the experimentation part of this thesis, it is important to know TransPacket's unique *fusion networking add-drop muxponder*, simply called *H1*. This chapter focuses on describing what is so special about the H1 node, its capabilities, key features, aggregation properties, management features and how it compares to other known hardware.

4.1 The H1 Node

The H1 fusion networking add-drop muxponder from TransPacket is an Ethernet based product with ten 1GE (Gigabit Ethernet) client interfaces and two 10GE (10 Gigabit Ethernet) line interfaces. Figure 4.1 shows an image of this H1 node.

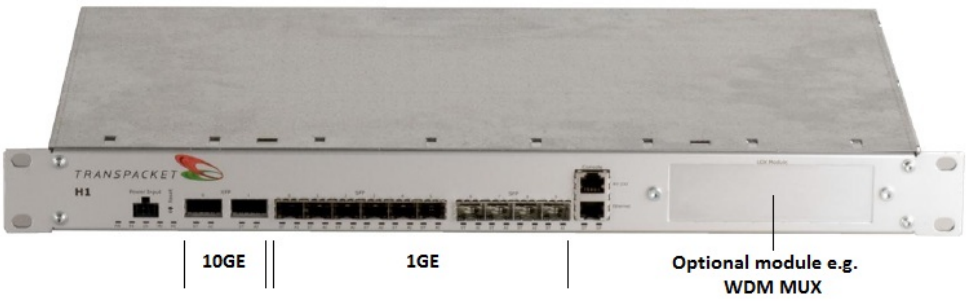

Figure 4.1: TransPacket's Fusion H1 add-drop muxponder. [Tra11a]

"H1 addresses the increasing demand for improved operator revenues through high capacity and high quality optical network transport services, dynamic networking and management simplicity" [Tra12a]. It allows Ethernet connections with wavelength-grade QoS (ultra-low latency, ultra-low PDV and zero packet loss), which allows operators to offer services for transport of e.g. synchronization information (IEEE 1588) [Tra12a], high-quality video and high-frequency trading applications. On top of that, high-capacity utilization and cost-efficiency is achieved using statistical multiplexing when aggregating traffic on top of a circuit path without impacting the performance of the wavelength transport.

4.2 H1 Capabilities

The H1 muxponder was specifically developed to carry out all of the functionalities of fusion networking. However, TransPacket's product overview in [Tra11a] summarizes the main capabilities of H1, listed below:

Transparent timing transport. The 1GE interface channels are transparent to all Ethernet frames and can also be transparent on timing, allowing tunneling of timing-critical IEEE 1588 time-stamped packets.

Router bypass. TransPacket H1 enables router bypass with QoS known from wavelengths, but with the granularity and bandwidth utilization known from packet switching by means of configuring transparent Ethernet lines (see section 3.3) across the network.

Digital Video Broadcast (DVB-T/H). DVB-T/H is fully supported by TransPacket H1, addressing its strict demands on frequency and time accuracy.

Integrated fusion and optical networking. The H1 muxponder functions can be efficiently combined with optical *wavelength division multiplexing* (WDM) networking within the same compact unit (see Figure 4.1) able to support a wide range of wavelengths over a *coarse WDM* (CWDM) and *dense WDM* (DWDM) network infrastructure.

OTN transport support. *Optical Transport Network* (OTN) is provided by an XFP plug-in module, such that the H1 can interface seamlessly into a wide area OTN network typically used for longer distance transmission.

4.3 H1 Key Features

TransPacket has also listed all of the key features the H1 node possesses in [Tra11a]. This is relevant information to know exactly what we are dealing with when working with H1 nodes.

- Ultra-low latency variation (PDV) for e.g. mobile backhaul services
- Ultra-low latency for mission-critical applications: video services, high-frequency trading, etc
- Zero packet loss
- Deep buffers (up to 200 ms) for statistical multiplexed aggregation, handling bursty traffic patterns
- Increased revenue on your fiber infrastructure by means of the Fusion network principle
- Future proof QoS handling existing and future services
- Compact and cost-effective aggregation of 1GE client signals into 10GE line signal
- Gigabit transparent Ethernet line channels with ultra-low latency and latency variation

- Configurable Add-Drop function
- Allows switch/router bypass, unique packet granularity on add-drop or bypass connections
- Scalable Integrated Fusion and WDM solution enabling pay-as-you-grow upgrades
- Cost-effective and energy-aware network design lowering Total Cost of Ownership
- 1U solution for small footprint and cost optimization
- Remote in-band, or out-of-band management
- Pluggable SFP for client interfaces and XFP for uplink allowing maximum flexibility and ease of operation
- CWDM and DWDM networking
- Support for 10GE Tunable XFPs and for 10GE OTN XFPs

If you are interested in knowing more details of the H1 node, you can go to Appendix A and see its *technical specifications*.

4.4 H1 Aggregation Properties

One of the most important things we need to know when working with the H1 fusion networking muxponder is how it can aggregate traffic. In [Tra12d], it is explained that "the H1 can aggregate ten GE streams on a 10GE without packet loss and is also capable of aggregating GE streams from the interfaces on top of traffic bypassing between the 10GE interfaces". In simpler words, this means that H1 can forward packets back and forth between the GE interfaces and the 10GE interfaces; and it is also possible to forward packets between the two 10GE interfaces, however, that cannot be done between the GE interfaces.

What can be sent between the different interfaces depends on what kind of traffic is intended to be aggregated. As explained in section 3.2, traffic in fusion networks is divided into high-priority GST traffic and low-priority SM traffic. H1 interfaces can be configured to operate either in *GST mode* or *SM mode*. As it is to be expected, interfaces configured in GST mode will always receive absolute priority and traffic from interfaces configured in SM mode may be aggregated in addition to the GST traffic without affecting GST's latency, PDV and packet loss [Tra12d].

4.4.1 Aggregation of SM Traffic

There is no limitation on the number of GE interfaces that can be configured in SM mode (from 1 to 10); however, 10GE interfaces cannot be configured as SM interfaces. With that in mind, aggregation of SM traffic can only happen from GE interfaces onto the 10GE interfaces. According to [Tra12d], it takes less than 3 μs to forward traffic from a GE to a 10GE interface. However, as SM traffic uses a packet-switched approach, when traffic is aggregated from several GE interfaces in SM mode, latency will vary according to the traffic pattern and if the sum of the aggregated traffic load is over 10 Gb/s, packets will start to be dropped.

4.4.2 Aggregation of GST Traffic

As we already know, GST traffic has zero packet loss and a fixed latency which depends on the size of the *maximum transmission unit* (MTU) configured for the system: the larger the MTU, the higher the latency [Tra12d]. On the other hand, PDV is independent of the MTU and is always constant at less than 100 ns.

As seen in Figure 4.2, there are three different scenarios on which GST traffic can be aggregated. 1) Between 10GE interfaces, traffic is always forwarded in GST mode. If this occurs, additional GST traffic may not be aggregated from any of the GE interfaces. 2) From a single GE interface onto a 10GE interface. This setup does not allow however to aggregate GST traffic from the other 10GE interface. 3) From two GE interfaces simultaneously, provided that each of the GE interfaces is aggregated onto each of the two different 10GE interfaces respectively.

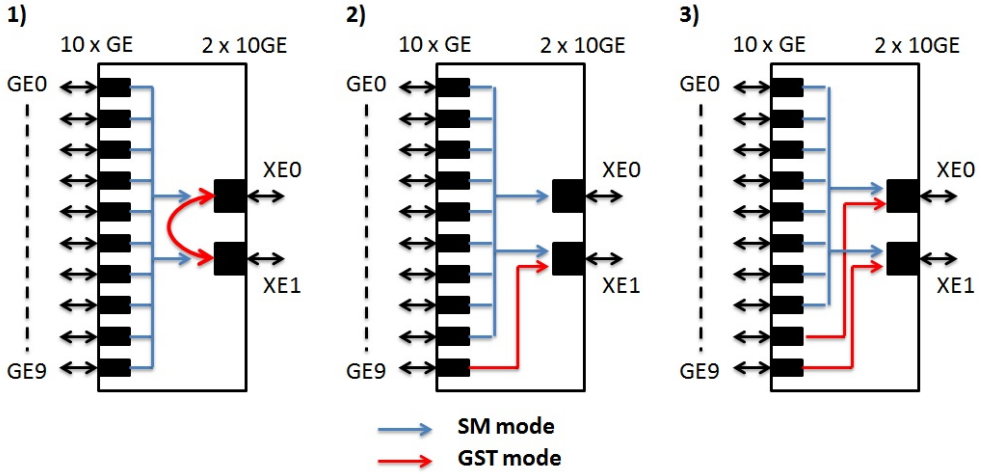

Figure 4.2: Possible GST traffic aggregation on the H1 node in combination with SM aggregation.

4.5 H1 Management

Nowadays, new services are developed in shorter and shorter timeframes, and so it has become quite a challenge to support them. "Service levels must be guaranteed through Service Level Agreements (SLAs) across multiple technologies and multiple networks" [Tra12a]. The way to address these challenges is to use standardized management interfaces which would integrate multiple technologies and vendors within one management system.

As can be seen in Figure 4.3, the Fusion H1 can easily be accessed and managed using a simple and intuitive *Command Line Interface* (CLI) based on industry standards [Tra12a]. Configuration can also be made through a web-based *Graphical User Interface* (GUI). You can also manage and monitor in-band or out-of-band, based on the Internet Engineering Task Force (IETF) standardized *Netconf* (XML), *Simple Network Management Protocol* (SNMP) and even the *Remote Network Monitoring* (RMON).

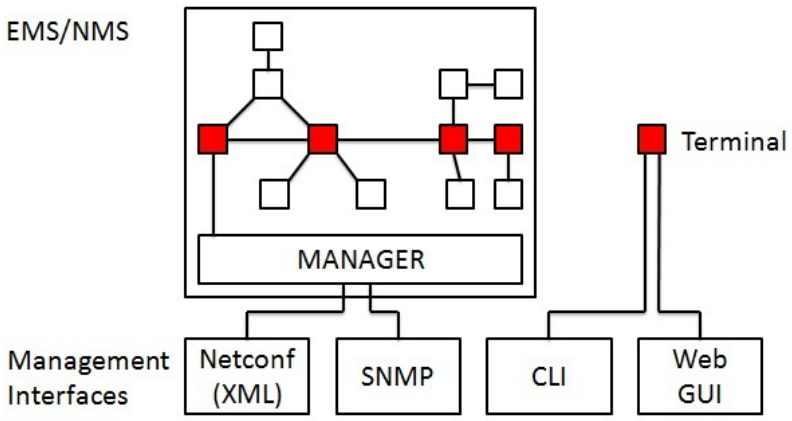

Figure 4.3: H1 management system. EMS=Element Management System; NMS=Network Management System. Figure adapted from [Tra11a].

4.6 H1 Comparison to other Hardware

TransPacket has made a comparison of its H1 node to other known hardware in [Tra12b]. It compares them based on key features such as transparency, latency, PDV, packet loss and more. Table 4.1 presents such information.

Table 4.1: Comparison of a hybrid muxponder such as H1 to hardware from layers 1, 2 and 3 (L1, L2, L3). *On hybrid lines. **When congested on SM aggregation interfaces.

	Hybrid Muxponder	**L1 Circuit Muxponder**	**Ethernet Muxponder**	**L2 Switch**	**L3 Router**
Transparency	Yes	Yes	No	No	No
Latency	Low	Low	Moderate	Moderate	High
PDV	Zero or negligible	Low	High	High	High
Packet Loss	No* and Yes**	No	Yes, when congested	Yes, when congested	Yes, when congested
Aggregation	Yes	No	Yes	Yes	Yes
Dynamic bandwidth utilization on circuit paths	Yes	No	N.A.	N.A.	N.A.
Absolute QoS	Yes	Yes	No	No	No
Power	Low	Moderate	Moderate	Moderate	High
1U size	Yes	Yes	Yes	Yes	Yes

As can be seen, H1 provides a unique combination of properties that no other technology can provide. It is a more flexible and cost effective solution than the conventional approaches relying on existing L1, L2 or L3 equipment.

Chapter 5

Previous Work

Before describing the lab setup used for the experiments that will support the data found for this thesis, it is important to document the previous work that has already been done using the fusion network approach. I will focus specifically on [RVH] written by PhD student Raimena Veisllari, and professors Steinar Bjørnstad and D. R. Hjelme which for the purpose of this thesis will be called "Paper 1"; and on [RVB] written again by Raimena and Steinar in collaboration with Kurosh Bozorgebrahimi, called "Paper 2".

5.1 Paper 1

Figure 5.1 shows the schematic diagram of the fusion node employed in this experiment. As can be seen, the node has two 10GE interfaces for GST transport and only four GE interfaces all of which are configured in SM mode. A VLAN-ID has been used to identify each packet's type of service. SM packets received at a 10GE interface are dropped to one of the GE interfaces, whereas GST packets received at a 10GE interface are passed through to the other 10GE interface with absolute priority with a fixed delay δ equal to the SM packets' maximum length.

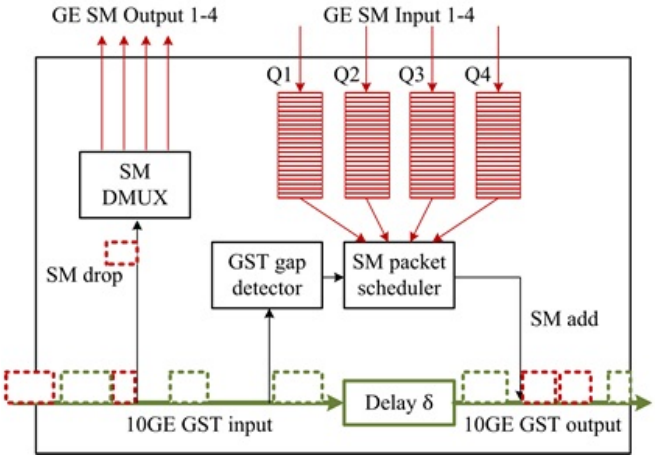

Figure 5.1: Schematic diagram of the unidirectional transport through the fusion add/drop muxponder used in Paper 1. [RVH]

The purpose of δ, as discussed in section 3.4.2, is to avoid contention between GST and SM packets; and also it helps detect the *gap-length* between GST packets which is used

by the *SM packet scheduler* to search the head of the queues (in this case Q1 to Q4) for a packet that is smaller than the detected gap. If such a packet exists, then it is scheduled and the process continues until the gap is full, no packet is smaller than the gap or the SM queues are empty.

The experiment was done using two fusion nodes named A and B and a packet generator and detector as seen in Figure 5.2. Transmission was performed in a unidirectional way, starting from the packet generator to node A, from node A to node B and from node B back to the packet detector.

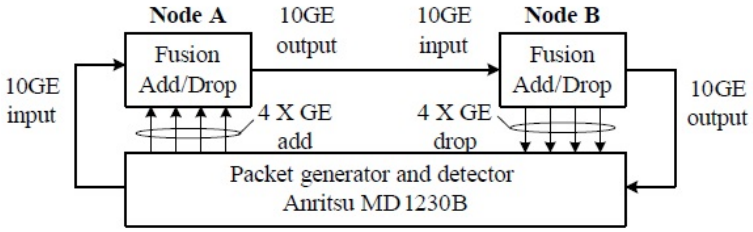

Figure 5.2: Experimental test-bed used in Paper 1. [RVH]

GST traffic was sent in bursts of 35 packets with a fixed gap between them of 4500 bytes, using the 10GE wavelength, bypassing both nodes and then received by the packet detector which measured packet loss, delay and PDV. The fixed delay δ was set to 12.8 μs and the GST traffic rate was 5.7 Gb/s. On the other hand, SM packet streams were generated and sent to node A using the GE interfaces; then they are scheduled by node A on the 10GE wavelength and dropped by node B to individual GE interfaces were packet delay and loss were also measured. The buffering capacity for each GE interface was 268 Mb which corresponds to a maximum delay of 268 ms.

Two different experiments were performed in order to measure the effect of the SM scheduling algorithm on GST's packet loss, delay and PDV as well as on SM's used capacity. On the first one, the SM load was put on a single GE interface; whereas on the second one, the SM load was distributed on four GE interfaces.

5.1.1 Results

They performed simulations and actual experiments, both of which yielded similar results. For the GST transport, no matter how the SM traffic was inserted, there was no packet loss; a total delay of 28.4 μs (14.2 μs per node including δ), which means the node processing delay is 1.4 μs at each node; and a PDV of less than 100 ns. Hence GST traffic performance was proven to be independent of the aggregated SM traffic.

Results for SM traffic however, did change from one experiment to the other. The difference lies on the *saturation threshold load value*, point after which the PLR and delay increase rapidly. When using a single GE interface for aggregating SM traffic it was found that the saturation point occurred at a load of 0.91 (see Figures 5.3 and 5.4), whereas in the second experiment where the same total SM load is given by four SM streams, the saturation point occurred at a load of 1.35 (see Figures 5.3 and 5.4). This means that by using a scheduling algorithm with multiple SM queues the throughput efficiency increased by 48%.

Why does this happen? First we need to define the *residual left-over capacity* (C^r_{SM}) by the GST stream (C_{GST}) on the 10GE channel as ($C^r_{SM} = 10Gb/s - C_{GST}$). However, this left-over capacity is not used completely, since it is separated into vacant gaps of different lengths, and so they are used only if there are SM packets at the head of the queue in the buffer that can fit in a given gap. Hence, the *effective left-over capacity* for SM traffic is ($C^e_{SM} = p * C^r_{SM}$) where p is the probability that the length of the SM packet at the head of the queue is less than or equal to the length of the vacant gap ($L_{SM} \leq L_g$).

This C^e_{SM} represents the saturation threshold value. And so, by using more SM streams (more queues) the probability p that $L_{SM} \leq L_g$ increases, increasing C^e_{SM} as well. And it also increased given the fact that the total SM buffer size is four times the one stream case.

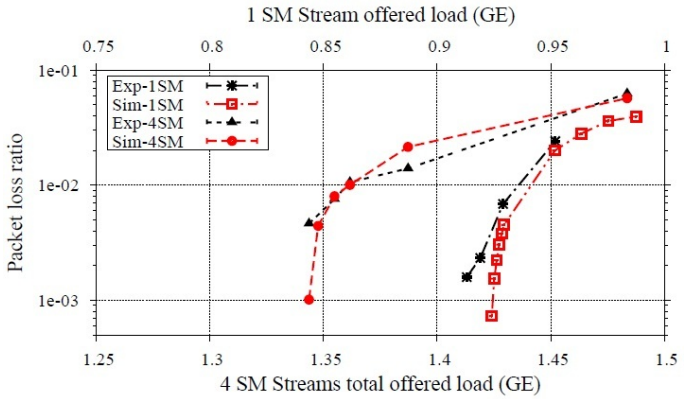

Figure 5.3: PLR as a function of the total added SM load. For the 1 SM stream case (top), you can see the PLR rises rapidly at a load of 0.91 to $1e^{-2}$; whereas for the 4 SM streams case (bottom) this occurs at a load of 1.35; the GST average rate is 5.7 Gb/s [RVH]

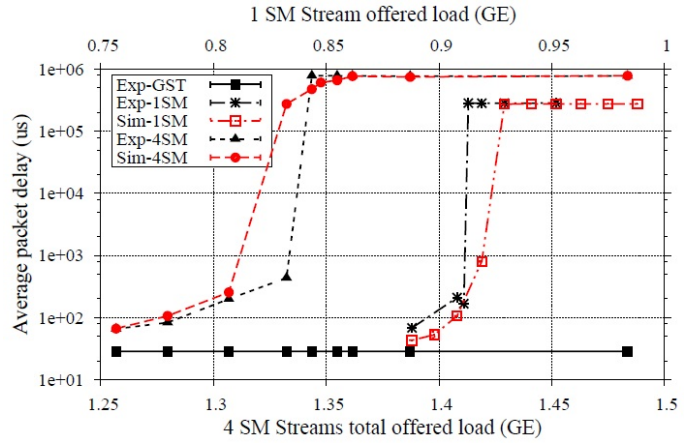

Figure 5.4: Delay as a function of the total added SM load. For the 1 SM stream case (top), you can see the delay rises rapidly at a load of 0.91 to 280 ms; whereas for the 4 SM streams case (bottom) this occurs at a load of 1.35 and to a delay of 780 ms; GST delay is shown and it is constant (no PDV) and its average rate is 5.7 Gb/s. [RVH]

5.2 Paper 2

Figure 5.5 shows the field-trial setup used in Paper 2. As it can be seen, there are three H1 nodes as well as three Spirent SPT-2000 and two Iperf traffic generators. Two different experiments were performed.

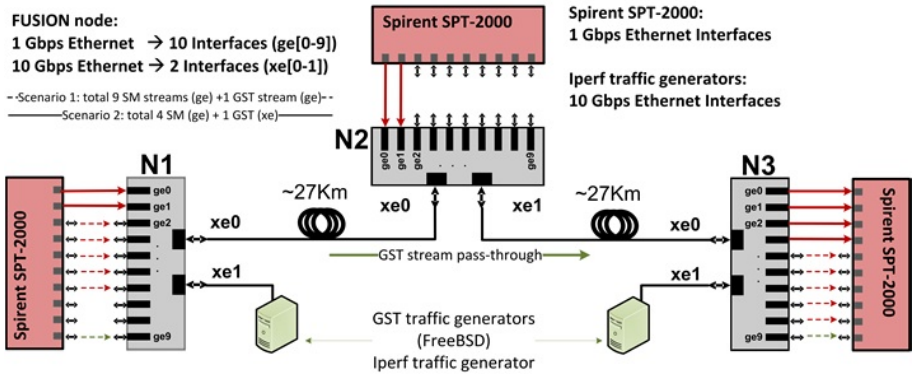

Figure 5.5: Experimental test-bed used in Paper 2. [RVB]

For the first experiment, GST traffic performance was measured by generating a GST stream from a GE port from the packet generator (Spirent) and added at node 1 (N1). The other nine ports are configured with SM traffic. The GST stream is then forwarded to node 2 (N2), it bypasses it and is then received at node 3 (N3). The GST load on the GE interface L_{GE}^{GST} was set to 0.99. The GST performance was measured for different network loads by changing the load L_{GE}^{SM} of the other nine added SM streams up until network congestion occurred. The measurements were conducted up to a total load L_{10GE}^{T} on the 10GE lightpath.

The second experiment's objective was to prove that by adding/dropping SM traffic on the same lightpath used for router bypass of circuit traffic, the network throughput increases. In this scenario, the GST traffic was generated by the Iperf network generator connected to xe1 interface at N1. This stream passes through N1 and N2 and then reaches N3's xe1 interface as its destination. The average load L_{10GE}^{GST} of the GST stream in the 10GE interface was set to 0.535. Only four SM GE streams were added, two by N1 and two by N2, all destined to N3. All of these streams had an equal average load L_{GE}^{SM} which changed throughout the experiment.

5.2.1 Results

As seen in Figure 5.6, the results for experiment one confirm that the GST traffic on the 10GE lightpath experiences no packet loss and no PDV when inserting SM packets up to congestion; and that SM insertion increases the 10GE lightpath utilization up to 97% without SM packet losses. This concludes that the hybrid network can offer an absolute priority to GST packets while at the same time increasing the lightpath utilization without any packet loss.

Now that we know for sure that GST packets have an absolute priority over SM packets, the second experiment focuses on the SM streams themselves. As mentioned before, the

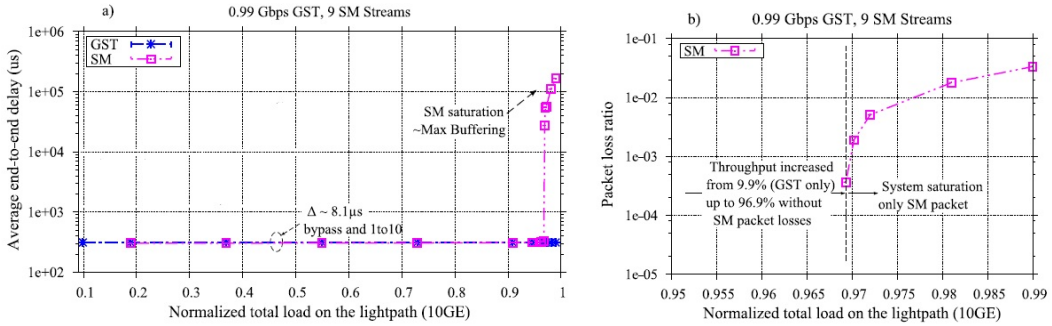

Figure 5.6: a) Measured delay on the 10GE lightpath for SM and GST packets. At 0.97 load, the SM packets start to saturate the network and so the delay increases rapidly, but GST experiences no delay; b) Total PLR for the SM traffic added on the lightpath. At 0.97 load, SM packets start to be dropped but the GST stream does not experience any losses. [RVB]

Iperf generates a GST share of 5.35 Gb/s. There are two cases that need to be compared for the SM streams added on N1: 1) bypassing N2 like GST traffic and thus having priority over the N2 streams; 2) dropped and re-added at N2 towards N3, which means both streams compete to be scheduled. The results for both SM cases in terms of delay and PLR can be seen in Figure 5.7.

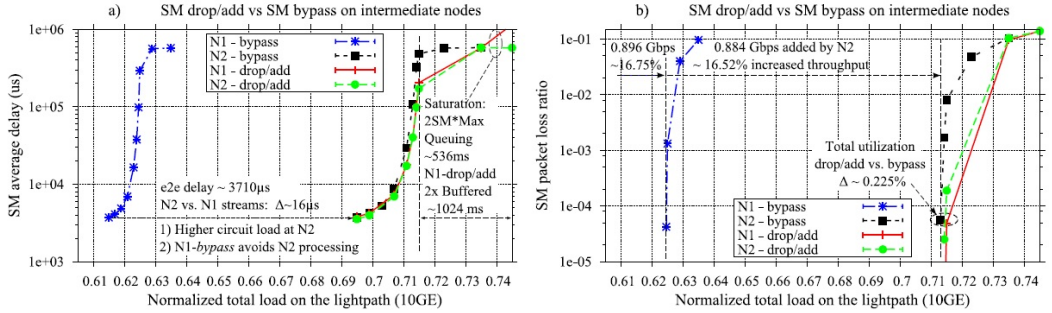

Figure 5.7: a) Measured delay on the 10GE lightpath for SM drop/add packets and bypass; b) Total PLR for the SM traffic added on the lightpath for drop/add and bypass. [RVB]

The graphs show that the SM insertion on N1 increases utilization to 6.246 Gb/s and then N2 increases it further to 7.13 Gb/s. And so the authors of this paper deducted that "the lightpath utilization increases with the number of streams added by the downstream nodes . . . the reason is that the probability of finding packets fitting in gaps increases with the number of nodes, e.g. when scheduling packets from both N1 and N2, compared to just N1" [RVB]. It can also be seen that the drop/add approach is more efficient for N1 since it increases the load to 0.71 instead of 0.62, however bypassing has the advantage of lower processing overhead.

If interested in more experimental demonstrations of a hybrid network, you can also see [SBE06] and [MNE06]. The results of these papers are consistent with the ones presented here.

Chapter 6

Laboratory Environment

The experiments carried out for this thesis were performed as collaboration by two Master students, Kefu He, student of MSc. of Telematics; and myself, Edgar Sánchez, student of MSc. in Security and Mobile Computing.

In order to perform such experiments, certain hardware (see section 6.1) and software (see section 6.2) was needed. The laboratory equipment was installed in room F251, inside the Electrical Engineering building within the Department of Telematics in the Gløshaugen campus of NTNU. A picture of such lab can be seen in Figure 6.1. The rest of the equipment resides within the offices of Uninett Sigma AS, in Abels gate 5, 7030, Trondheim, Norway.

Figure 6.1: The LAB environment at the Department of Telematics in NTNU.

6.1 Hardware

We were given two desktop computers in order to perform the experiments. These computers would function as the management interface to the H1 nodes, as well as the *packet generator/analyzer* (PGA). Their hardware configuration can be seen in Figure 6.2.

There were of course two H1 nodes which would be configured remotely, since they were not physically available in the lab. The technical specifications of which can be seen in

Appendix A.

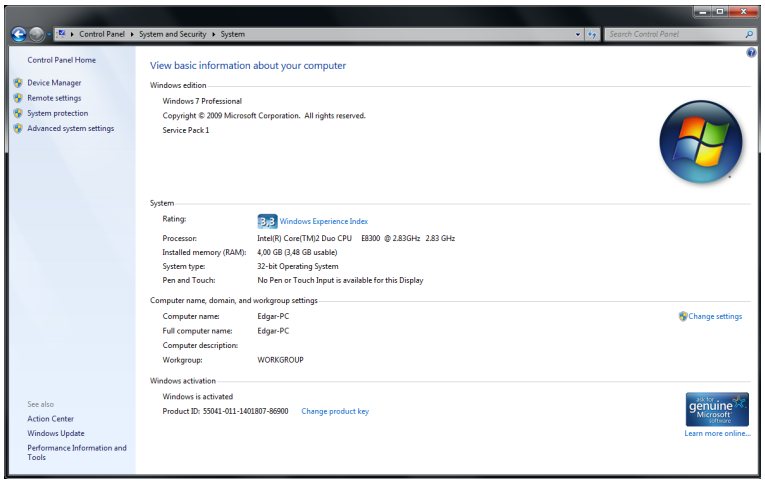

Figure 6.2: Hardware and Software specifications of desktop computers used.

And last but not least, the PGA. As the H1 nodes, the PGA is also located in the facilities of Uninett. They are using a Spirent TestCenter with an SPT-2000A-HS chassis (suitable for lab use only). An image of it can be seen in Figure 6.3. Some technical specifications found in [Com07] include:

- Supports up to sixteen 10/100/1000 Mbps Ethernet ports, sixteen GE ports, twenty-four dual media GE ports, four 10GE, four UniPHY and four WAN ports.
- Horizontal module slot orientation.
- Front panel LEDs: power, fan, link, status, and over temperature.
- Back panel LEDs: activity/collision, link/error, 10/100Mbps, and 1000Mbps.
- Ethernet connection: 10Base-T connector 10/100/1000 Mbps (half/full duplex in 10/100 mode).

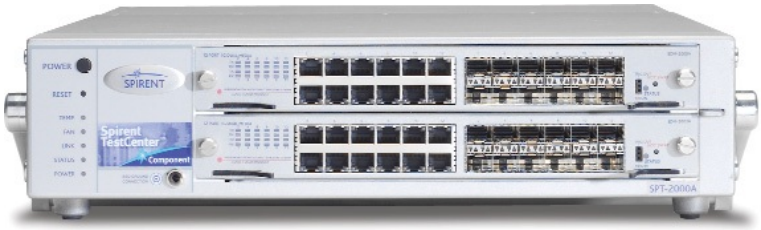

Figure 6.3: Spirent SPT-2000A-HS chassis. [Com07]

Two interface modules have been installed in the chassis, one EDM-2003B 12 port 10/100/1000 dual media and one HyperMetrics mX 10G SFP+ 2 ports. Information about these modules can be found in [Com07, Com13a, Com13b].

6.2 Software

As can also be seen in Figure 6.2, both computers had a 32-bit Windows 7 Professional as the *operating system* (OS) installed in them. The open source telnet and *secure shell* (SSH) client for Windows platform called PuTTY was also installed to provide a secure remote connection to the H1 nodes.

Once inside the H1 node, its configuration is done through a CLI based on YANG and netconf as the configuration protocol. "YANG is a data modeling language used to model configuration and state data manipulated by the *Network Configuration Protocol* (NETCONF) protocol, NETCONF remote procedure calls, and NETCONF notifications" [IET10]. Whereas NETCONF "provides operators and application developers with a standard framework and a set of standard *Remote Procedure Call* (RPC) methods to manipulate the configuration of a network device" [Bie13]. YANG is IETF RFC6020 and netconf IETF RFC6241. They are both outside the scope of this thesis, but you can get more information in [IET10] and [Bie13] respectively.

Finally, using the Remote Desktop Connection (RDC) of Windows, we could remotely connect to the Spirent node, in order to use the Spirent TestCenter to generate traffic and analyze it. Table 6.1 shows a summary list of the software needed for each device.

Table 6.1: Software needs for each device.

Device	**Amount**	**Software**
Desktop Computer	2	Windows 7 Professional
		PuTTY
		Remote Desktop Connection
TransPacket H1 Muxponder	2	YANGCLI
		NETCONF
Spirent SPT-2000A-HS	1	Spirent TestCenter

6.3 System and Network Overview

First of all, a fixed IP address was configured on both computers A and B. A's address is 129.241.205.81, whereas B's address is 129.241.209.220. After assigning such IP addresses, access to the H1 nodes was given. In order to remotely log into H1, one needs to know the IP address of its management Ethernet port (me0), which in this case were 158.38.152.165 (connected to A) and 158.38.152.166 (connected to B). This connection is made using an SSH client program such as PuTTY at the standard TCP/IP port 22 as can be seen in Figure 6.4. A username and password are needed in order to access the CLI.

Similarly, an account was created at Uninett for us to be able to access the Spirent TestCenter. Through means of the Remote Desktop Connection, we can connect to the Spirent node by simply using its name "spirent.uninett.no" as seen in Figure 6.4 and then providing a valid username and password. Once inside, the interface is the same as the one of a regular Windows computer. One only needs to open the Spirent TestCenter, reserve the ports needed for the experiment and start configuring them.

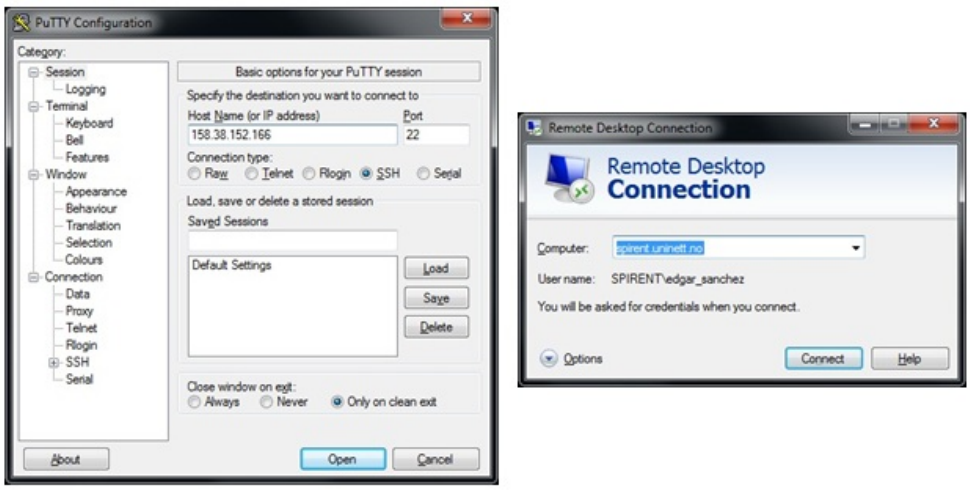

Figure 6.4: Left: SSH secure connection to H1 node. Right: Connection to the Spirent box using the Remote Desktop Connection.

A graphic description of this network set up can be seen in Figure 6.5. Let us remember that only computers A and B were physically available to us, and that the H1 nodes and the Spirent box were in the premises of Uninett. Also, both connections are secure since they both require a username and password, even if the connection to Spirent is not using SSH.

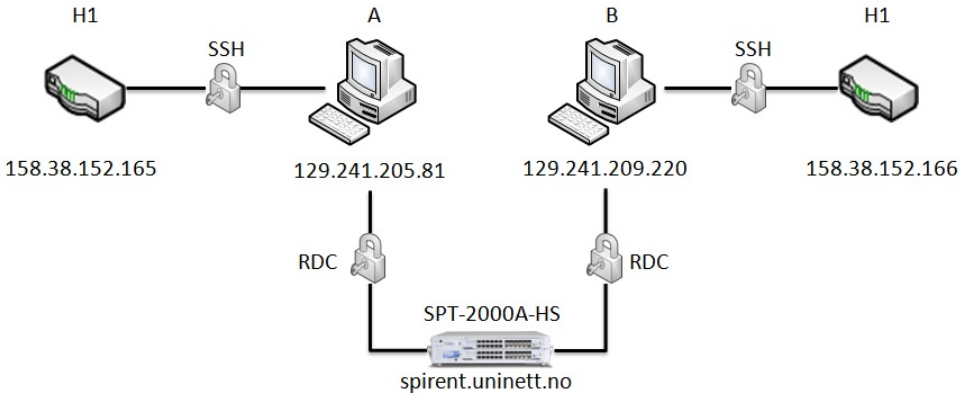

Figure 6.5: Graphic representation of how the H1 nodes and Spirent box are remotely accessed by the users for management.

Chapter 7

Network Scenario

Now that the conditions of the laboratory environment as well as the remote connections for the configurations of the H1 nodes and the Spirent box have been presented, the network scenario upon which the experiment was carried is shown in this chapter. This topology was proposed and agreed upon by the main supervisor of this thesis Steinar Bjørnstad; PhD student Raimena Veisllari; Kefu He and myself.

7.1 Physical Topology

Figure 7.1 shows the exact representation of how the two H1 nodes are connected to each other and to the Spirent box physically. It is important to note that optical connections are not like electrical ones, in the sense that each cable has four connectors; two pairs of transmitting (Tx) and receiving (Rx) ends for bidirectional communication.

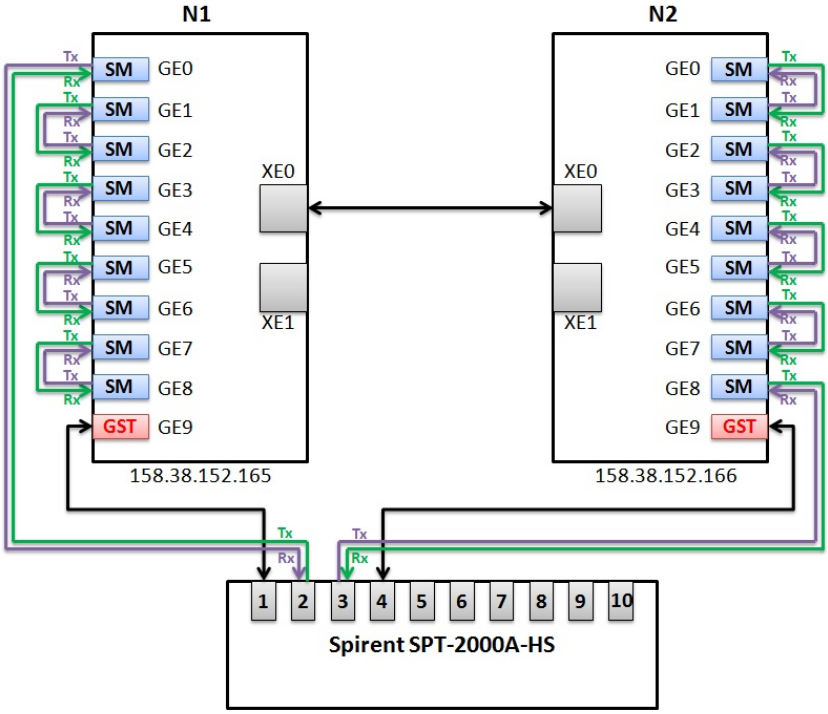

Figure 7.1: Physical topology used for the experiment.

This can be seen in the figure as well and so a pair of green-purple arrows represents a single optical cable. On the other hand a single color represents one flow of data, i.e., if you follow the green arrows, you will see data starts at port 2 from the Spirent box, goes to GE0 of N1, passes through the XE0 10GE connection and is forwarded to GE0 of N2 (using VLAN-tags), then to GE1 of N2, passes again through the 10GE connection on the other direction and so on until it reaches port 3 on the Spirent box.

So there are two streams of data, from port 2 to port 3 (green) and from port 3 to port 2 (purple), both of which will be analyzed in the Spirent TestCenter (see chapter 8). However, both these streams are configured as SM traffic. Port GE9 on both N1 and N2 are configured as GST and connected to port 1 and 4 of the Spirent box respectively. These connections have been represented as a single bidirectional arrow for simplicity, but the flows are the same; from 1 to 4 and from 4 to 1.

7.2 Logical Topology

As mentioned above, there are two SM streams of data flowing in opposite directions. If you follow each one from source to destination looking at Figure 7.1, you will see that because of all the loops, both streams go through the 10GE link 9 times, which logically speaking, it is the same as going through 9 nodes.

On the other hand, by definition, the GST streams only go through the 10GE link once to get from its source to its destination, same as bypassing those 9 nodes. And so the same physical topology can be more accurately represented in Figure 7.2.

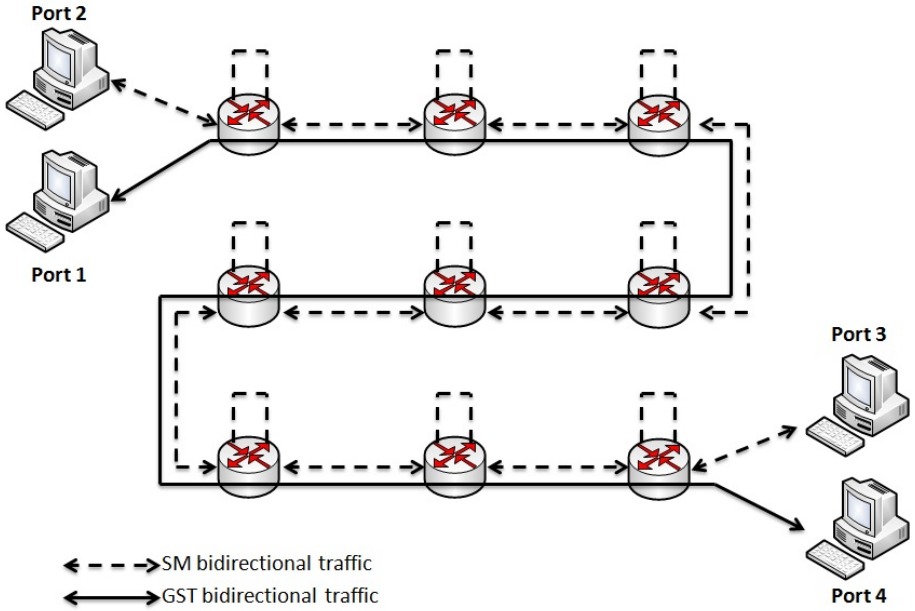

Figure 7.2: Logical representation of the physical topology. Each computer represents a Spirent port whereas each node represents a time when an SM stream needs to go through the 10GE link (aggregation). The arrows are bidirectional and they both represent the sigle 10GE fiber cable connected at XE0 of N1 and N2.

As can be seen in Figure 7.2, the SM packets are switched at each node according to their header information, whereas the GST packets bypass all of the intermediate nodes and go directly from the source to the destination, ensuring absolute priority.

7.3 Field-trial Setup Objective

The network topology was setup as it was in order to further demonstrate that the fusion technology, using TransPacket's H1 muxponders allow transporting GST traffic with circuit QoS; that is with no packet loss, no PDV and minimum delay independent of the insertion of statistically multiplexed traffic.

I will also demonstrate that link resources are almost fully utilized by sending SM packets in the rest of the ports. A single SM stream was looped from one node to the other, back and forth and its load was linearly increased until network congestion was reached. The main objective is to see the exact SM saturation point, which will be the point at which the network is best utilized while at the same time providing full priority and no packet loss to GST streams.

Chapter 8

Experiment Procedure

In this chapter the test procedure for carrying out the experiment is described. From making the physical connections all the way to configuring the H1 nodes and setting up the Spirent TestCenter to generate and analyze traffic from its correspondent ports. A step-by-step of everything that was done to reach the results (see chapter 9) is explained in this chapter. I also tried to keep up with a time plan agreed upon by my supervisors and myself for both the experiments and deliverables of the actual written thesis. Appendix B shows this time plan.

8.1 Physical Connectivity

After having planned the network scenario described in chapter 7, the first thing to do was to make the physical connections. In order to do that, we contacted Kurosh Bozorgebrahimi who is in charge of the TransPacket H1 nodes at Uninett, where the nodes actually are. He gave us access to the server room and with his help and expertise in handling optical connections, we managed to set it up exactly as the physical topology suggested. It is important to note that optical connections are very sensitive to dirt, dust and even fat from fingers, so extra attention and care needs to be paid when handling such cables. Figure 8.1 shows two pictures taken at Uninett of the connections made.

Figure 8.1: Left: Optical connections made in H1 nodes; mainly loops and the 10GE connection between each other. Right: Spirent node and its optical connections to the H1 nodes.

8.2 H1 Configuration

Accessing the H1 nodes for configuration has been explained in chapter 6, section 6.3. This section deals with the configurations made to both H1 nodes, the reasons why they were configured this way and how they were configured.

Once inside the H1 node, the factory default username and password need to be provided, which according to [Tra12c] are

user: *root*
password: *hadm1_123*

After successful login, a Unix shell is accessed and the device's CLI configuration tool can be started by typing the *cli* command. The real configuration starts from this point forward. This is not meant to be a configuration guide to H1 but some commands will be displayed here to show what was done during this experiment. If you are interested in looking at the command list for H1 see [Tra12c].

8.2.1 Enabling Interfaces

By default all of the traffic interfaces (ge0-9, xe0-1) on H1 are administratively down and so the first configuration needed was to enable these interfaces (except xe1) by creating a container for each one. The command used to perform this operation in e.g. ge0 is

create /interfaces/interface[name='ge0']

Ports ge0 to ge9 are configured as access interfaces and the command above is enough to configure them as such; however, port xe0 needs to be configured as a trunk interface with the following command

create /interfaces/interface[name='xe0']/ethernet-switching/trunk-interface

The interfaces will now be enabled but the link status will only be up provided an operational interface is connected to the configured port, whether that is another port within the same node, or a port in another H1 node or Spirent box.

8.2.2 Creating VLANs

Once all interfaces configured are enabled and up, VLANs need to be created since the actual switching is made through the use of vlan-tags. Ten VLANs were created, one for each ge interface with the command

create /vlans/vlan - - name=vlanx id=10x

Where x goes from 0 to 9.

8.2.3 Adding Interfaces to VLANs

After all VLANs exist, interfaces need to be added as a member of a VLAN. It is important to note that access interfaces can only be member of a single VLAN, whereas trunk interfaces can be part of one or more.

Careful thought needs to be given when assigning interfaces to VLANs. In this case, the flow of information goes from one ge0 to the other, from ge1 to ge1, ge2 to ge2 and so on, passing all the time through xe0 on both nodes. For this reason, the same configuration applies to N1 and N2, having ge0 be a part of vlan0, ge1 of vlan1, ge2 of vlan2 up until ge9; and xe0 will become a member of all the VLANs.

For the access interfaces the command used to add them as members of a certain VLAN is

create /interfaces/interface[name='gex']/ethernet-switching/access-interface/vlan - - vlan-name=vlanx

Where x again goes from 0 to 9. And for xe0, the only configured trunk interface, the command changes a little to

create /interfaces/interface[name='xe0']/ethernet-switching/trunk-interface/vlans/vlan - - vlan-name=vlan0

This command needs to be applied ten times, changing the *vlan-name* field in order to add xe0 to each existing VLAN.

8.2.4 Switching SM mode to GST mode

Last but not least, all access interfaces are by default set to SM and as seen in our topology in Figure 7.1, port ge9 on N1 and N2 should be configured as GST. The command used to set this interface as GST is as follows

create /interfaces/interface[name='ge9']/ethernet-switching/access-interface/vlan - - vlan-name=vlan9 priority=gst

After all these steps have been performed on both N1 and N2, these nodes are fully operational and ready to receive traffic. The running configuration of N2 (same as N1 except for the IP address of the management interface me0) can be seen in Appendix C as well as a summary of the VLANs which shows more clearly access and trunk interfaces in SM or GST mode.

8.3 Spirent TestCenter Configuration

The last step of the experiment procedure is to configure the PGA, which is the Spirent TestCenter. Chapter 6, section 6.3 explains how to remotely connect to it through means of Windows RDC; this section deals with a detailed description of its actual configuration.

The configuration of the Spirent TestCenter can be divided into three main parts:

- Port Reservation and Configuration
- Traffic Generator
- Traffic Analyzer

8.3.1 Port Reservation and Configuration

The first step after opening the Spirent TestCenter is to reserve the ports that will be needed, which in this case are ports 1 to 4. In order to do this, you need to click on *Connect to Chassis and Reserve Ports* button. The chassis available will be displayed and if you click on it all of its ports will be listed as seen in Figure 8.2. We selected ports 1 to 4 and click ok.

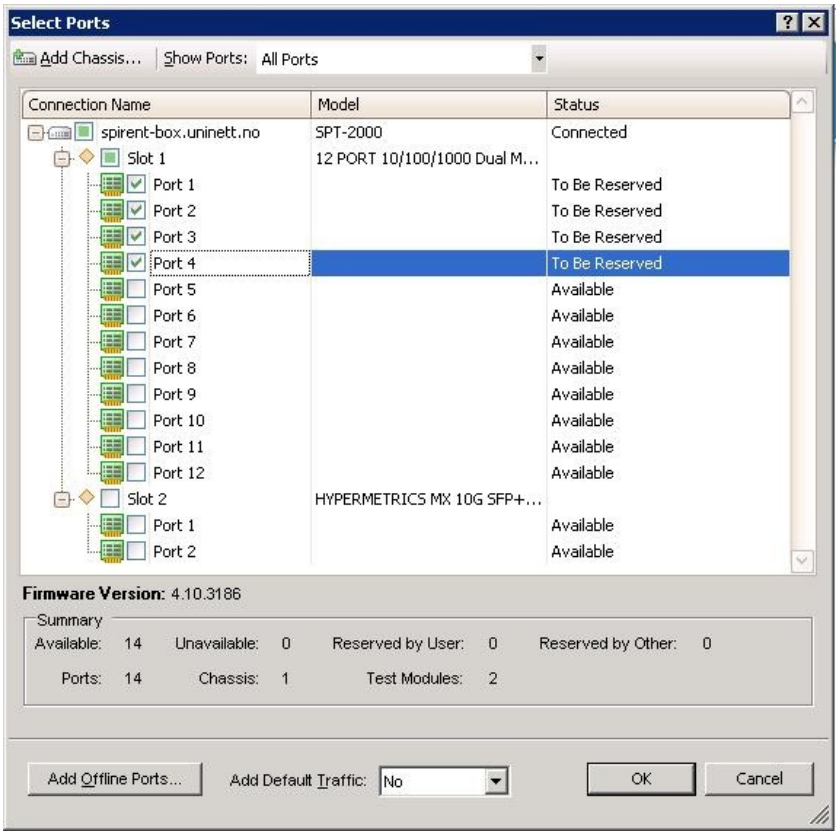

Figure 8.2: Reserving ports 1 to 4 in the Spirent TestCenter.

The GUI of the Spirent TestCenter is then shown and can be seen in Figure 8.3. On the left side there is a map of the most important parts, which need to be configured. You can see and configure *All Ports*, *All Devices*, *All Traffic Generators*, *All Stream Blocks* and *All Traffic Analyzers*; or you can go one by one, by clicking on each port.

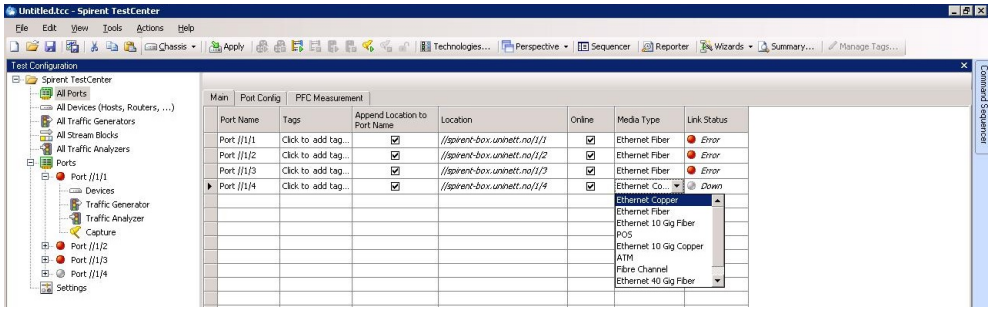

Figure 8.3: GUI of the Spirent TestCenter.

The link status of all the ports is at the beginning set to *Error* and this is because by default, the media type is set to *Ethernet Copper*, and the physical cables used are actually Ethernet fibers. So first we click on *All Ports* and change the media type to *Ethernet Fiber*. The link status will then change to *Down*. Next we turn off the *Auto Negotiation* set up and click on *Apply*. The link state of all ports should now be *Up* with a green-colored circle, line speed of 1 Gbps, full duplex and an MTU of 1500 bytes as seen in Figure 8.4.

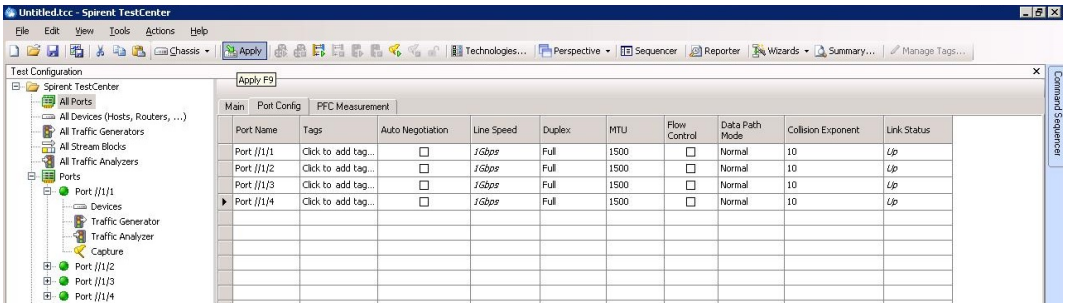

Figure 8.4: Port configuration in Spirent TestCenter.

Now that the four ports are up and running, we need to create devices onto them for actually sending and receiving data. Go to *All Devices* and click on *Add*. A wizard will guide you through this process. First we select all four ports and choose *Ethernet* as port type. No encapsulation is needed so we choose *None* when asked about it. Next, change the *Devices per port* from 1 to 2 (one for sending, one for receiving) and the *Device block mode* to "One device per block" and click on *Finish*. There should be 8 devices in total as seen in Figure 8.5.

Emulated Device Interface | 6rd/6to4 | DS-Lite

Port Name	Device Name	Tags	Port Name	Device Count	Incoming Links	Outgoing Links	Encapsulation	Router ID
Port //1/1	Device 1	Click to ad...	*Port //1/1*	1			EthernetII	192.0.0.1
Port //1/1	Device 2	Click to ad...	*Port //1/1*	1			EthernetII	192.0.1.1
Port //1/2	Device 3	Click to ad...	*Port //1/2*	1			EthernetII	192.0.2.1
Port //1/2	Device 4	Click to ad...	*Port //1/2*	1			EthernetII	192.0.3.1
Port //1/3	Device 5	Click to ad...	*Port //1/3*	1			EthernetII	192.0.4.1
Port //1/3	Device 6	Click to ad...	*Port //1/3*	1			EthernetII	192.0.5.1
Port //1/4	Device 7	Click to ad...	*Port //1/4*	1			EthernetII	192.0.6.1
Port //1/4	Device 8	Click to ad...	*Port //1/4*	1			EthernetII	192.0.7.1

Figure 8.5: Created devices per port: one for sending and one for receiving.

8.3.2 Traffic Generator

By this point, all ports are fully operational and are able to transmit and receive data. The next step is to configure the stream blocks that are to be generated.

To open the *Traffic Wizard* go to *All Stream Blocks* and once again choose all four ports. Then the source-destination pairs need to be paired according to the topology explained in chapter 7, section 7.1.

Ports 1 to 4, 4 to 1, 2 to 3 and 3 to 2 were set up using all 8 devices as can be seen in Figure 8.6. It is important to note that the *Orientation* was set to *Bidirectional* and that *Stream only generation* is selected.

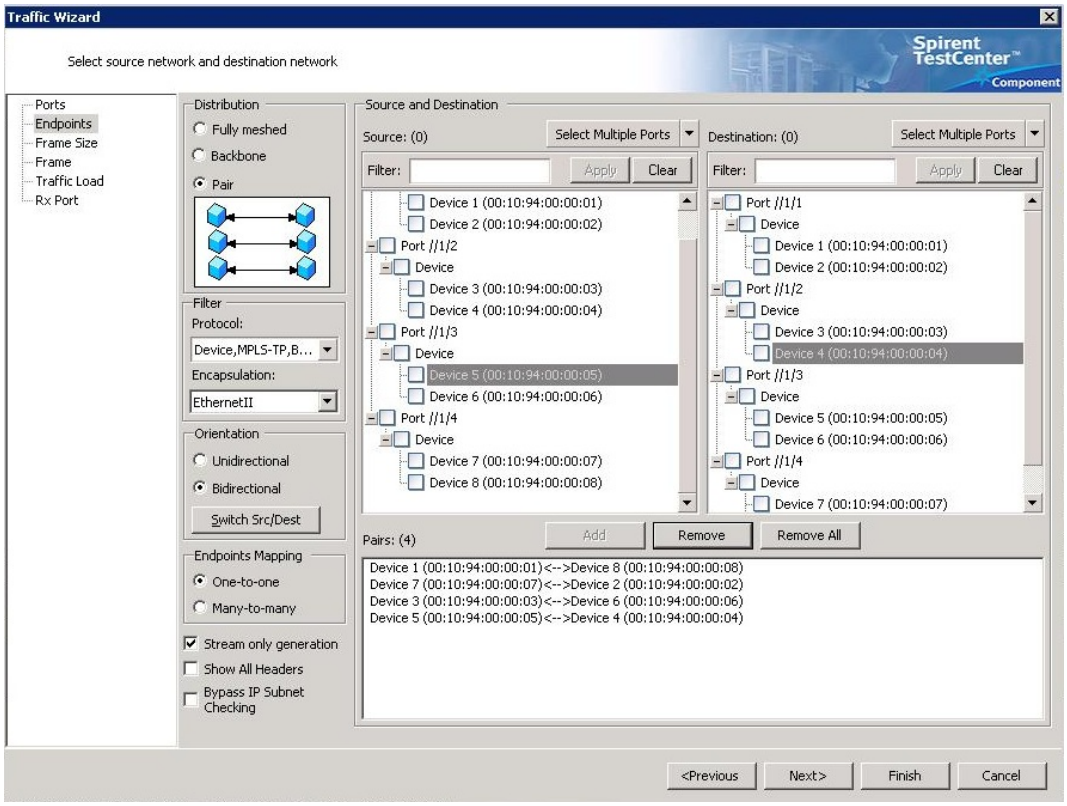

Figure 8.6: Traffic Wizard: selecting source-destination pairs.

Next we configure the *Frame Size* to *Random* from 64 up to 1518 bytes which is Ethernet's MTU. Finally, we need to set up the *Traffic Load*. The configuration of such depends on whether the port will be SM or GST. For ports 1 and 4, we used a fixed load of 95%, whereas for ports 2 and 3, we ran several experiments running from 10% all the way to 99.5% to find the point of saturation (but more on that later).

The traffic wizard is then done and you can see such configuration on *All Traffic Generators*. From there one can change quickly some configuration parameters as well, like the load and burst size. The way we performed this experiment was with a certain time interval and so instead of *Continuous*, we set the *Duration Mode* to *Seconds* and the *Duration* to *5,000*. Also, for each experiment, we changed the *Random Length Seed* parameter to any random number, as long as they were all different at each port; this is to have a more realistic scenario of what actually happens with real traffic.

8.3.3 Traffic Analyzer

All ports, devices and stream blocks have now been configured properly and the Spirent TestCenter is now ready to run. In order to run the experiment first click on the *Start capture on all ports* button. You will be able to see a "(C)" next to each port, meaning that it is capturing. Then click on *Start traffic on all ports* button to actually start the transmission and analysis of data.

The results will be shown at the bottom of the GUI as a table with different informa-

tion such as the total sent frames and received on any given port; the average, maximum and minimum latency; the amount of dropped frames; and the PDV of a stream. Figure 8.7 shows an example of how this is supposed to look like.

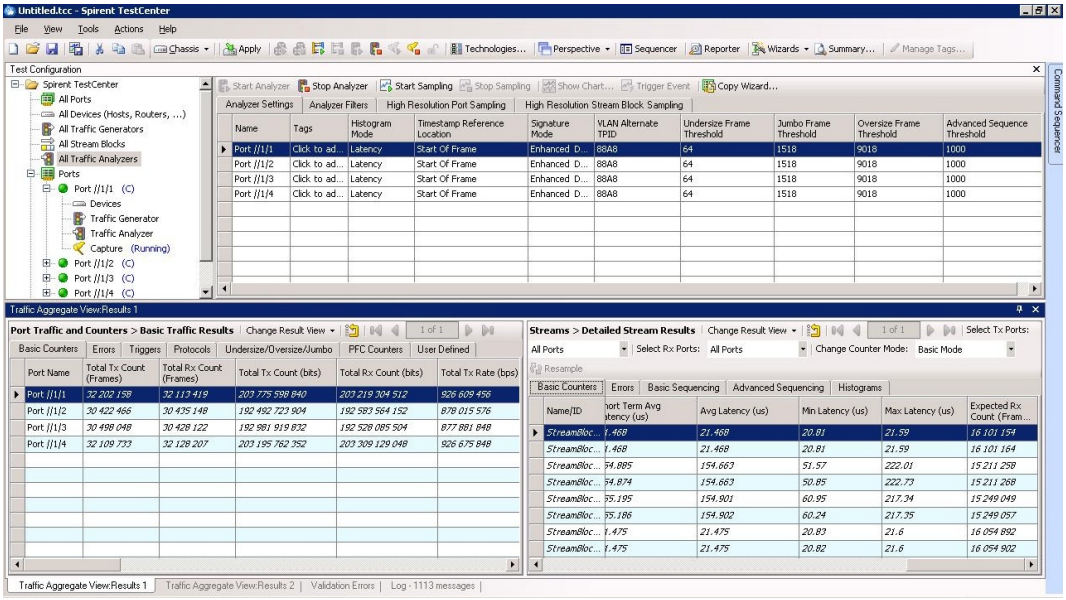

Figure 8.7: Traffic analyzer displaying the results at the bottom of the Spirent TestCenter's GUI.

A total of 18 experiments were performed, changing the load of the SM ports from 0.1 to 0.995. The results of such experiments are shown in chapter 9.

Chapter 9

Results

This chapter presents the results obtained by the experiments done (please note that these results are discussed further in chapter 10). Here, only the values will be explained and whether or not they match the expectations. Main focus will be put on three main characteristics of the SM and GST packets:

- Average end-to-end delay
- Packet Loss Ratio
- Packet Delay Variation

A comparison of SM and GST will be made, using plots that graphically represent the results obtained. But first some of the raw data received from the Spirent TestCenter and calculations made are presented.

9.1 Data

As mentioned in chapter 8, section 8.3.3, a total of 18 experiments were performed. The way this was done was to set a fixed GST load of 95% or $L_{GE}^{GST} = 0.95$ and then change the SM load or L_{GE}^{SM} from 10, 20, 30 and so on up to 90%, 95, 96, 97, 97.2, 97.5, 97.8, 98, 99 and 99.5% as can be seen in Table 9.1. The reason of having a finer granularity between 97 and 98% is explained in section 9.3.

Appendix D has a very detailed set of tables that show absolutely all of the relevant raw data collected by me, separated by each different SM load. Table 9.1 shows the actual total load being used in the 10GE lightpath (L_{10GE}^{T}) or xe0 port of the H1 nodes. Because the 1GE SM stream goes through the 10GE link 9 times as explained in chapter 7, section 7.2; and the L_{GE}^{GST} is always equal to 0.95, then the formula to calculate the total load on the 10GE lightpath is

$$L_{10GE}^{T} = (1Gbps * L_{GE}^{SM} * 9) + (1Gbps * 0.95)$$

The L_{10GE}^{T} serves as the base point of comparison between the SM and the GST streams, meaning such values will be shown in the x-axis of the plots for the end-to-end delay, PLR and PDV.

Table 9.1: 1GE SM and GST load; and total load in the 10GE lightpath for the 18 experiments.

L_{GE}^{SM}	L_{GE}^{GST}	L_{10GE}^{T}
0.1	0.95	1.85
0.2	0.95	2.75
0.3	0.95	3.65
0.4	0.95	4.55
0.5	0.95	5.45
0.6	0.95	6.35
0.7	0.95	7.25
0.8	0.95	8.15
0.9	0.95	9.05
0.95	0.95	9.5
0.96	0.95	9.59
0.97	0.95	9.68
0.972	0.95	9.698
0.975	0.95	9.725
0.978	0.95	9.752
0.98	0.95	9.77
0.99	0.95	9.86
0.995	0.95	9.905

Next there are the results of each of one of these characteristics. It is important to note that the results shown come from the average of both streams, i.e. from port 2 to 3 and 3 to 2, which individually are very similar but not exactly the same.

9.2 Average End-to-End Delay

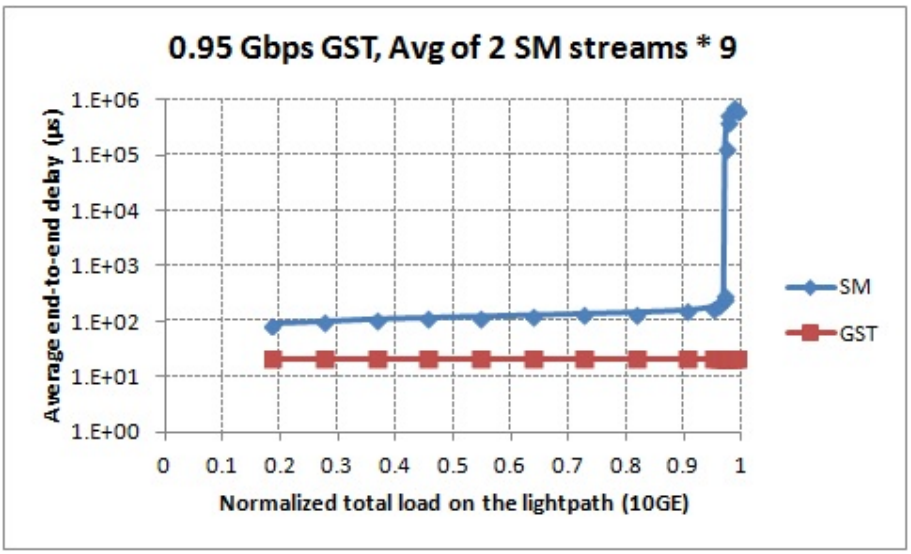

Figure 9.1: The average packet latency for both SM and GST traffic as a function of the normalized offered load on the 10GE lightpath.

Figure 9.1 illustrates the average end-to-end delay of packets for both SM and GST traffic as a function of the normalized L^T_{10GE}. The results are exactly as expected:

SM The SM packet delay increases slowly as the 10GE lightpath load increases up until $L^T_{10GE} = 0.9698$ which is when $L^{SM}_{GE} = 0.972$. At the next tiny load increase of only 0.003 (0.3%), i.e. $L^{SM}_{GE} = 0.975$ and $L^T_{10GE} = 0.9725$; the end-to-end delay increases exponentially from exactly 301.69 μs to 134,891.18 μs. From there on the SM delay keeps increasing dramatically to a maximum value of 712,526.37 μs or 0.7 seconds. It can be said then, that there is a fine granularity on the load when a spike in delay occurs.

GST The GST average end-to-end delay is 21.47 μs through all the measurements, which means it remained constant regardless of the network conditions and total load utilization; a characteristic that according to the definition, needs to be fulfilled.

9.3 Packet Loss Ratio

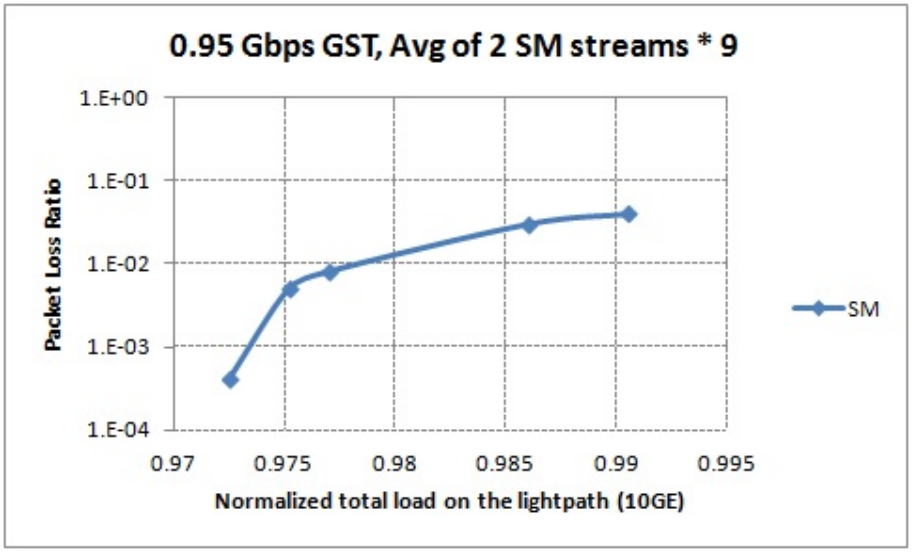

Figure 9.2: The total packet loss ratio for the SM traffic added on the lightpath. The GST stream does not experience any losses.

Figure 9.2 illustrates the total PLR experienced by the traffic, which as can be seen, is only experienced by SM traffic and not GST traffic as it was expected:

SM From $L^{SM}_{GE} = 0.1$ to $L^{SM}_{GE} = 0.97$ there was absolutely no packet losses. Every single packet sent was received on the other side. However, when $L^{SM}_{GE} = 0.98$ was tested, there was a PLR of 0.813%. One of the objectives, as mentioned in chapter 7, section 7.3, is to find the exact SM saturation point, at which packets start to get dropped. This is the reason why between $L^{SM}_{GE} = 0.97$ and $L^{SM}_{GE} = 0.98$, there is a finer granularity of experiments, to find such point. The results indicate that packets started to get lost at $L^{SM}_{GE} = 0.975$ or $L^T_{10GE} = 0.9725$. We can then conclude that SM insertion increases the 10GE lightpath utilization up to 96.98% without any packet losses and that with SM PLR of $1E^{-02}$ (exactly 3.977%) for the total load $L^T_{10GE} = 0.9905$, the network performs as a

saturated statistical multiplexing packet network with high utilization while providing a service with circuit QoS properties.

GST As seen and mentioned in Figure 9.2, GST presents no packet losses whatsoever regardless of whether the network reaches the saturation point (gets congested) or not. So, in combination with the results in section 9.2, it is safe to say that GST traffic is transported through the network with absolute priority and neither delay nor PLR are affected by the SM insertion.

9.4 Packet Delay Variation

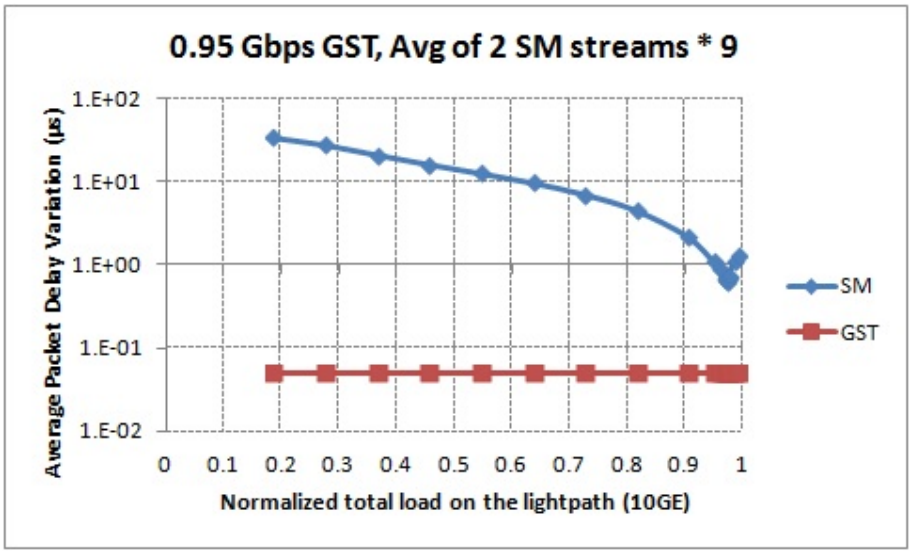

Figure 9.3: The average packet delay variation for both SM and GST traffic as a function of the normalized offered load on the 10GE lightpath.

Figure 9.3 shows the average PDV or jitter for both SM and GST traffic as a function of the normalized L^T_{10GE}. The results were as expected, especially for the GST traffic; however, for the SM the shape of the curve was surprising, although correct:

SM As can be seen in Figure 9.3, the greater the load the lower the PDV up until the saturation point, where the PDV starts to increase again. This is because "in general, higher levels of jitter are more likely to occur on either slow or heavily congested links" [LLC]. When the load is low, then buffers will never be full, and so since we are using the store and forward switching (see chapter 2, section 2.1.2), it depends on the size of the packet being sent (64 to 1518 bytes). The delay of sending a 64-byte packet and a 1518-byte packet is different and so PDV is large. When the load is optimal, meaning large but still no congestion occurs, i.e. $L^T_{10GE} = 0.9698$, the buffers are being utilized more efficiently, that is, almost every packet, no matter what their size is, will have to wait approximately the same amount of time in the buffer before being transmitted out, and so the PDV will decrease. This trend of course changes when congestion occurs, and packets start to get dropped due to buffers being full.

GST In correlation to the average end-to-end delay seen in section 9.3, it can already be assumed that the PDV of GST traffic will be constant, and it is. The PDV value was as little as 0.05 μs or 50 ns through all the measurements in average. The PDV peak value (seen in Appendix D) ranged from 0.53 to 0.56 μs. So yet again, the GST traffic maintains the absolute priority it is supposed to have, with no packet loss, little delay and a tiny constant PDV of 50 ns.

Chapter 10

Discussion

This chapter deals with a more comprehensive discussion based on the results presented in chapter 9. The main result of the experiment is that using TransPacket's H1 nodes, all of the fusion networking characteristics and advantages described in chapter 3 are 100% fulfilled.

That is, the GST traffic performance is completely independent of the added SM traffic and its load. GST was always given absolute priority and remained with a constant average end-to-end delay of 21.47 μs, no packet loss and a minimum PDV of 50 ns while SM traffic load increased, increasing the overall 10GE lightpath utilization up to 99.5%.

The insertion of SM traffic is necessary to achieve the optimal utilization of the link resources (high throughput efficiency) and have as little waste as possible. This was definitely achieved during experimentation and found that the point of saturation was at $L^T_{10GE} = 0.9725$, where statistically multiplexed packets started to overflow the node's buffers and so packets started to get lost, at which point delay increased exponentially and PDV went up as well.

And so it was found that with this network setup, the highest throughput efficiency achieved without any losses is at $L^T_{10GE} = 0.9698$ or 96.98% of the 10GE lightpath. This way, demanding services such as real-time traffic, synchronization and control information can be transported as GST traffic with absolute priority; and all other data transmitted as SM traffic with an average end-to-end delay of 301.69 μs, no losses and a PDV of 670 ns. But, is that enough?

Section 10.1 presents the minimum requirements in terms of delay in order for certain time-sensitive applications and general data to work properly, and compare them with the GST and SM performance achieved in this experiment.

10.1 Delay Requirements for Time-Sensitive Applications and General Data

Time-sensitive applications are services which performance is greatly affected when there is end-to-end latency, i.e. real-time traffic such as online gaming, telemedicine and e-health, online banking and videoconferencing; and also synchronization and control information.

These demanding services are increasingly being used and so it is very important that they receive the priority and QoS only GST traffic can provide.

General data is concerned with traditional applications such as web browsing, e-mail and file sharing and so they fit into the SM type of traffic for transmission.

Online Gaming "Latency determines not only how players experience online gameplay but also how to design the games to mitigate its effects and meet player expectations" [CC06]. Unfortunately, measuring the maximum amount of delay that would be acceptable for online games is not so straightforward, since there are different kinds of games and actions within that game. For example, shooting a high precision weapon such as a sniper rifle at a moving target is more delay sensitive than moving around a map. Mark and Kajal Claypool have come up with a categorization of actions in games, and how they are differently affected by latency in [CC06], but that is outside the scope of this paper. According to [BC04, CD06, Cla05, NC04], the maximum threshold for the most highly sensitive online games is 100 milliseconds.

Telemedicine and E-health "The core capability of telemedicine is to deliver rapid and high quality healthcare across large distances" [MJLH09]. *Teleoperation* or *telesurgery* is one of the most important services within telemedicine, since any minor error caused by network latency or any other factor could kill patients. Several experiments have been conducted by universities in the US, England, France and Japan using the US Army's RAVEN to record its latency times under different circumstances. The results are found in [MJLH09]: the time delays range from 16 (0.5 km) to 172 milliseconds (7,700 km).

Online Banking "Banking is an information-intensive business and thus Information Technology (IT) has become increasingly important in this industry" [SF06]. The the effect of latency is directly proportional to the amount of money the transaction is dealing with. According to [Tod11], "even a one millisecond advantage could be worth up to 100 million USD a year to the bottom line of a large hedge fund".

Videoconferencing This is becoming more and more popular not only for businesses, but also for regular users, since it allows them to interact with other people around the world across long distances. "In order for the participants in a videoconference call to interact naturally, the end-to-end delay should be below human perception" [BO00]. According to several experiments performed in [BO00], to achieve this natural interaction the end-to-end one way delay must be of approximately no more than 100 milliseconds.

Control Information Meaning the exchange of status, signaling and routing information in a network is highly sensitive to packet loss and according to [NSR11] has an upper bound delay of 100 milliseconds.

General Data Data from applications like web-browsing, e-mail and file sharing have very variable bandwidth demands and are sensitive to packet loss; however, they have low real-time demands which make the use of retransmissions possible. That is why, according to [NSR11], the upper bound delay of this kind of data can go up to 1 full second.

Figure 10.1 shows the end-to-end delay gathered from the results of both GST and SM

traffic (but only GST traffic is shown), with the upper bound delays identified above for each one of the different kinds of demanding services or time-sensitive applications.

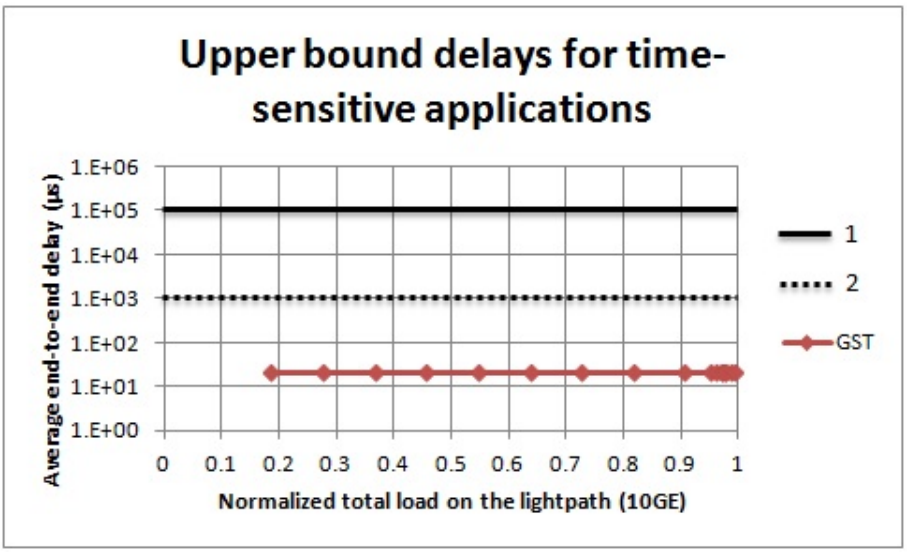

Figure 10.1: The GST average end-to-end delay is 21.47 μs through all the measurements no matter the SM insertion. Upper bound delay 1 is at 100 ms for online gaming, video-conferencing and control information. Upper bound delay 2 is at 1 ms for online banking. The range between both upper bounds have been proven to be acceptable for telemedicine and e-health.

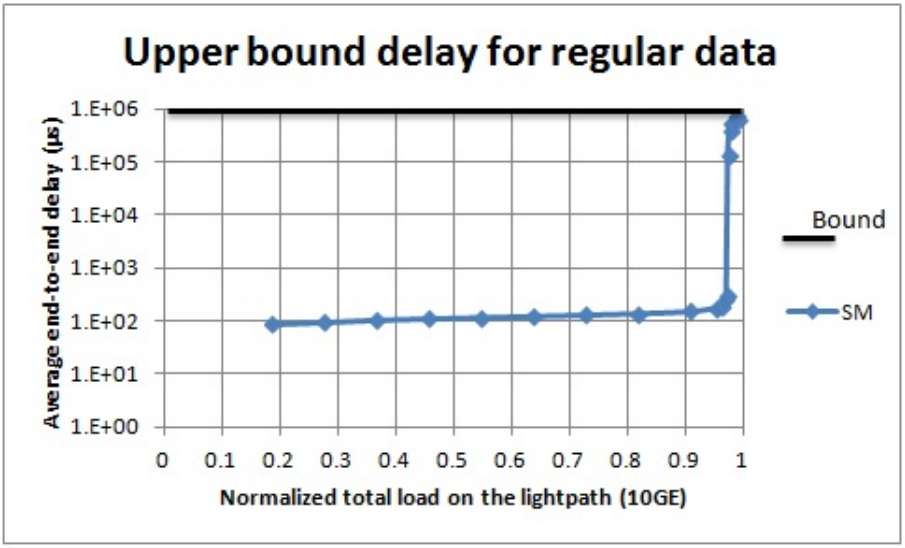

Figure 10.2: The SM packet delay increases slowly as the 10GE lightpath load increases up until $L^{T}_{10GE} = 0.9698$. At which point it increases exponentially from exactly 301.69 μs to 134,891.18 μs. From there on the SM delay keeps increasing dramatically to a maximum value of 712,526.37 μs or 0.7 seconds. The upper bound delay for general data is 1 second.

Figure 10.2 shows the end-to-end delay gathered from the results of both GST and SM traffic (but this time only SM traffic is shown), with the upper bound delay identified

above for general data.

And so, as can be seen from Figures 10.1 and 10.2, the experiment results have proven that the H1 nodes are capable of transporting the most demanding services with absolute transfer guarantees and circuit-switched-like QoS; while at the same time achieving a high throughput efficiency of 96.98% and transport the rest of the data with an average delay of 300 μs and no packet loss. It is worth mentioning that all of the upper bound delays are the extreme cases in which the specific kind of traffic will still work decently, but a good network performance should always be way below such values, as the ones achieved by the H1 nodes.

However, a very important detail in such results is the fact that both H1 nodes communicating and used during experimentation are physically located in the same room. And so these results represent only the processing, queuing and transmission delays mentioned in chapter 2, section 2.2; but the propagation delay, which depends entirely on the physical distance between source and destination and the propagation speed of the physical link used, has not been taken into account. Section 10.2 shows an approximation of how much latency is added by propagation using fibers.

10.2 Adding Propagation Delay to Results

As mentioned above, propagation delay depends entirely on the distance between source and destination and the propagation speed of the physical medium being used, which in this case it is fibers, as we are working with an optical network.

According to [MRV10, Inc11, Nor12] latency sources in fiber optic networks include (1) the *fiber* itself, (2) the *optical components* and (3) the *opto-electrical components* needed to carry out the function of optical networks from source to destination as seen in Figure 10.3. Each of these latency sources are described below.

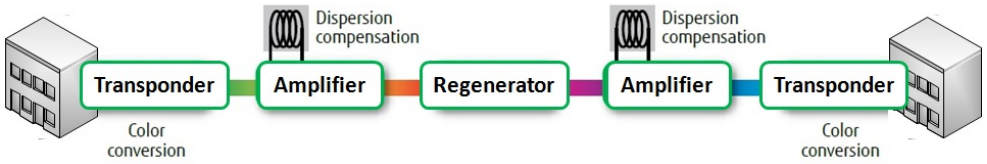

Figure 10.3: Optical nerworking functions that can increase latency.

1. Light in a vacuum travels at 3×10^8 m/sec, which in terms of latency translates to 3.33 μsec/km. However the optical signal propagation speed through a fiber cable is lower than the speed of light because of the cable's so called *refractive index* property. According to [Nor12], this speed reduction increases latency to 4.76 μsec/km. On this source of latency, it is important to note also that the fiber is not always installed on the shortest physical way between two points which would also add to latency. Rerouting fibers, however, is a very expensive process and only extremely time-sensitive applications, such as online banking, would be willing to pay.

2. The optical component that introduces the most latency is the *Dispersion Compensating Fiber* (DCF). The DCF is used to remove the effects of the so called *chromatic dispersion* of the fiber which degrades the fiber's performance and limits its maximum reachable distance. Only used in long-distance networks, a DCF is a long spool of special fiber that is usually 20-25% of the total fiber length, therefore adding 20-25% of latency. The longer the distance, the more DCFs need to be in place, adding up to a few milliseconds of latency.

   A second optical component is the *optical amplifier*. Amplification of the signal is needed in long distance networks, since the light source fades as it travels through the fiber. They also remove the need for *optical-electrical-optical* (OEO) conversion which of course removes latency. The most common optical amplifier is the *Erbium-Doped Fiber Amplifier* (EDFA). An EDFA adds tens of meters of extra fiber. Each EDFA introduces only hundreds of nanoseconds of delay, but when a lot of EDFAs are needed, as is the case for long distance networks, they can introduce considerable latency.

3. The opto-electrical components include *transponders* and *muxponders*. The former is in charge of *color conversion* (traffic signals are converted from gray to a specific light color or wavelength). The latter aggregates multiple colors into a single high-speed channel. Both components introduce latencies of about 5-10 μs each. If *Forward Error Correction* (FEC) is performed, the extra processing creates even more latency.

Figure 10.4 shows an example of two data centers 400 km apart with 5 spans of 80 km, meaning there are 6 amplifiers. From the information above, we can calculate the total latency introduced by the different latency sources. These calculations can also be seen in Figure 10.4.

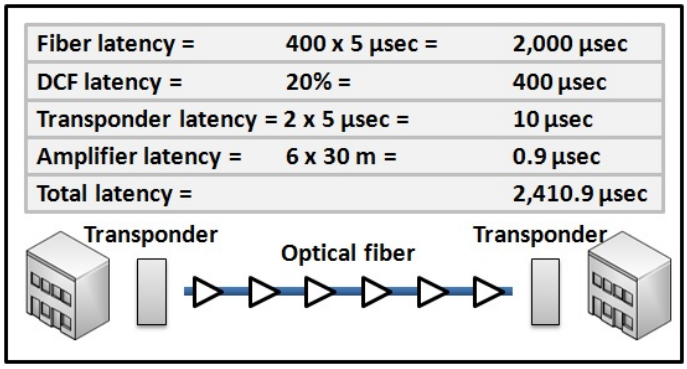

Figure 10.4: Latency introduced on a 400 km long fiber network.

The total latency of such optical network is 2,410.9 μs. This might be perceived as very low latency considering it is not even close to a millisecond, but in delay-sensitive applications even 1 μs can have a negative effect. That is why new technologies are arising in the optical world, to strip off as much latency as physically possible.

When it comes to the optical fiber itself, there is nothing anyone can do, unless someone has found a way to change the laws of physics of light speed.

A new technology known as *Fiber Bragg Grating* (FBG) [Inc11, Nor12] has enabled the development of the *Dispersion Compensation Module* (DCM). DCM is used to compensate for the fiber's chromatic dispersion as DCF does, however, DCM does not introduce any delay when doing so. This will shave off up to 25% of the total latency.

For signal amplification, an alternative to EDFA is also available. The *Raman amplifiers* [Inc11, Nor12] do not require extra meters of fiber to amplify the signal, and so they deliver low-latency amplification. This will shave off the small latency introduced by all the EDFAs in the network.

Last but definitely not least, there are the so called *ultra low-latency transponders* [Nor12]. These are transponders that do not perform complex operations like FEC or management channels. These kinds of transponders can have up to 1,000X better latency performance than regular transponders.

Figure 10.5 shows how latency has been reduced by applying these changes to the example shown in Figure 10.4.

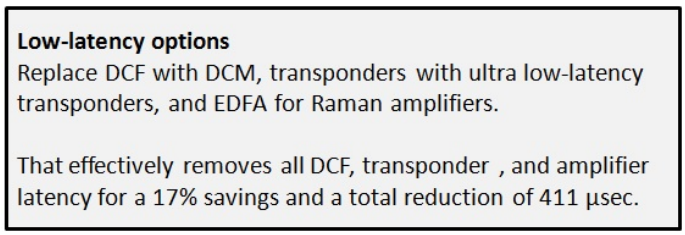

Figure 10.5: Lowering latency in optical networks.

So in the end, it is fair to say that the optical network latency can be limited to the physical limitations of the fiber, to around 5 μs per kilometer.

So by mere approximations, this would mean that it would take around 20,000 km (or half the circumference of the Earth) between source and destination, for the end-to-end delay of GST traffic to be over the 100 ms threshold of most of the time-sensitive applications described in section 10.1. And for SM traffic, it would take around 200,000 km (or five times the circumference of the Earth) distance to reach over one second delay.

So it is safe to say, that even after adding up the propagation delay, using the fusion technology provided by TransPacket's H1 nodes, demanding services are transported as GST traffic with absolute priority, no packet loss, no PDV and minimum delay (way under upper delay bounds found in Figure 10.1); as well as with regular data, sent as SM traffic, with a delay way under one second (300 μs + distance*5 μs), no packet loss and a PDV of nanoseconds; and a high throughput efficiency of 96.98% of the 10GE lightpath.

Chapter 11

Conclusion

This thesis work started with the motivation of addressing the following networking problem: how can we offer the necessary QoS to enable zero packet loss and low predictable latency to guarantee the transport of demanding services (i.e. real-time traffic, synchronization and control information) with absolute priority while at the same time sharing link capacity dynamically for traffic with changing intensity and connections with different bandwidth needs resulting in a higher throughput efficiency?

The first step was to start with a theoretical study of hybrid optical network architectures that would be able to address this problem and combine the best characteristics of circuit-switching and packet-switching. It was found that research has been made on this topic, although it was not extensive. Three main architectures have been discussed in chapter 2, section 2.4. But the main focus of this paper was to work with TransPacket's Fusion Networking solution, which is an integrated hybrid optical network; that is they share the network resources such as bandwidth simultaneously, i.e. each node in an integrated hybrid network can choose to send traffic wavelength-switched through a predetermined connection or circuit; or ignore that circuit path and process the traffic in a packet-switched manner.

In the fusion network approach, traffic is divided into two service classes while still using the capacity of the same wavelength in a WRON: the high priority traffic labeled GST for the circuit-switched class; and the best-effort packet-switched traffic labeled SM offering high bandwidth efficiency. During experimentation, we worked with TransPacket's H1 nodes, and VLAN configurations were set in order to differentiate between SM and GST traffic.

Once a full background study of how fusion technology works and the characteristics and advantages it theoretically brings was made, we had to set-up a network scenario in which we could at first-hand, work closely with the H1 nodes, perform several experiments and get results that would either prove fusion network performs as it does in theory, or not.

In order to accurately measure the GST traffic performance, we generated two 1GE GST streams from the packet generator in opposite directions and averaged the results together. Also, SM traffic was added from both N1 and N2, each as a single stream that would go through the 10GE link 9 times before arriving at its destination. The average GST load was set to 0.95, and its performance was measured for different network loads on the 10GE

link by increasing SM load until network congestion occurred.

The experimental results point out that the integrated packet/circuit hybrid optical network known as fusion, carried out by TransPacket's H1 nodes has the capability of transporting even the most demanding services with no packet loss, ultra-low fixed delay and almost no PDV (properties that were only offered by legacy circuit technologies like SONET/SDH); and these values were not impacted or in any way affected by the insertion of SM traffic.

The high throughput efficiency and utilization of the 10GE Ethernet wavelength was demonstrated by increasing the SM load until SM packets started to get dropped. The saturation point of this specific network setup was found to be at $L_{10GE}^{T} = 0.9725$ and so according to the results, this network was able to transmit the rest of the data as SM traffic without packet losses, an average delay of 300 μs and a PDV of 670 ns at 96.98% wavelength utilization.

We also discussed that such results did not include any propagation delay, given the fact that both H1 nodes were physically connected in the same place. But chapter 10, section 10.2 concluded that this delay is calculated to be 5 μs per kilometer using fibers. However, for further work (see chapter 12) an actual hands-on experiment where two or more H1 nodes in different cities or countries are communicating with each other is still necessary to have actual results on how much this propagation delay contributes or affects the network performance.

Apart from that, I can conclude that fusion technology works and can be used instead of legacy circuit technologies like SONET/SDH to transport demanding services with absolute priority and a QoS that can only be provided by circuit-switching; but offering a very high throughput as the one offered by packet-switching. Fusion networking: the best of both worlds.

Chapter 12

Further Work

This master thesis gave a full theoretical background of fusion networking as well as a complete hands-on experiment that provided results to support the theory. However, there is always more to be done. This chapter gives suggestions on how to proceed with this topic or experiments that need to be further investigated since they were not covered in this paper mainly due to time constraints.

As mentioned in chapters 10 and 11, the experiment performed for this thesis did not include any propagation delay since both H1 nodes were situated in the same location. And so, the next logical step is to perform a new experiment where N1 and N2 (and maybe even N3) are physically located hundreds of kilometers apart. I would personally suggest a connection between Trondheim (Uninett premises) and Oslo (TransPacket offices).

Last but not least, aggregation of 9 GE wavelengths using an H1 with a built-in C2 CWDM MUX/DEMUX module is theoretically possible. This means that the link will then be able to support up to 90 Gb/s transport capacity. It would be interesting to perform an experiment where these C2 modules are used in combination with the H1 nodes as further work.

Bibliography

[BC04] Coughlan R. Lusher C. Plunkett J. Agu E. Beigbeder, T. and M. Claypool. *The effects of loss and latency on user performance in Unreal Tournament 2003. In Proceedings of ACM Network and System Support for Games Workshop (NetGames), Portland, 2004*, 2004.

[Bie13] Andy Bierman. *Welcome to Netconf Central: XML-based Network Configuration Tools Online YANG Module Database. Netconf Central*, March, 2013. http://netconfcentral.org/.

[BO00] Mario Baldi and Yoram Ofek. *End-to-end delay analysis of videoconferencing over packet-switched networks. Networking, IEEE/ACM Transactions on 8, no. 4*, pages 479–492, 2000.

[CC06] Mark Claypool and Kajal Claypool. *Latency and player actions in online games. Communications of the ACM 49, no. 11*, pages 40–45, 2006.

[CD06] Claypool K. Claypool, M. and F. Damaa. *The effects of frame rate and resolution on users playing first-person shooter games. In Proceedings of ACM/SPIE Multimedia Computing and Networking Conference, San Jose, CA*, 2006.

[Cla05] M. Claypool. *The effect of latency on user performance in real-time strategy games. Elsevier Computer Networks 49*, pages 52–70, September, 2005.

[Com07] Spirent Communications. *Spirent TestCenter: SPT-2000A, SPT-5000A and SPT-9000A chassis. Spirent Communications White Paper*, 2007.

[Com13a] Spirent Communications. *HyperMetrix neXt mX 10G Module*. 2013. http://www.spirent.com/Products/Spirent-TestCenter/HyperMetrics_mX_10G.

[Com13b] Spirent Communications. *Spirent TestCenter HyperMetrics*. 2013. http://www.spirent.com/Products/Spirent-TestCenter/HyperMetrics_Main.

[Dav08] Kevin Davis. *Latency and Jitter. Service Assurance Daily*, June 23, 2008. http://www.serviceassurancedaily.com/2008/06/latency-and-jitter/.

[GPJKD06] Christoph M. Gauger, Mario Pickavet Paul J. Kühn, Erik Van Breusegem, and Piet Demeester. *Hybrid Optical Network Architectures: Bringing Packets and Circuits Together. IEEE Communications Magazine*, August 2006.

[HWD12] Alberto Colmenero Henrik Wessing, Tony Breach and Lars Dittmann. *Technology evaluation for time sensitive data transport. Nordunet Conference 2012, Oslo*, 2012.

[IET02] IETF. *RFC3393. IETF RFC*, November, 2002. http://tools.ietf.org/html/rfc3393.

[IET10] IETF. *RFC6020. IETF RFC*, October, 2010. http://tools.ietf.org/html/rfc6020.

[Inc11] Fujitsu Network Communications Inc. *Low-latency networks for trading infrastructure. Fujitsu White Paper*, 2011.

[KR10] Jim Kurose and Keith Ross. *Computer Networking: A Top-Down Approach.* Pearson, 5th edition, 2010.

[LLC] VoIP Troubleshooter LLC. *Indepth: Jitter.* http://www.voiptroubleshooter.com/indepth/jittersources.html.

[MJLH09] Hawkeye King Diana C.W. Friedman Thomas S. Lendvay Andrew S. Wright Mika N. Sinanan Mitchell J.H. Lum, Jacob Rosen and Blake Hannaford. *Teleoperation in Surgical Robotics - Network Latency Effects on Surgical Performance. 31st Annual International Conference of the IEEE EMBS, Minnesota, 2009*, September 2, 2009.

[MNE06] O. Austad V. L. Tuft D. R. Hjelme A. S. Sudbø M. Nord, S. Bjørnstad and L. E. Eriksen. *OpMiGua Hybrid Circuit- and Packet-Switched Test-Bed Demonstration and Performance. IEEE Photonics Technology Letters, Vol. 18, No. 24*, December 15, 2006.

[MRV10] MRV. *The Low Latency Network - Design Considerations. MRV White Paper*, May 4, 2010.

[NC04] J. Nichols and M. Claypool. *The effects of latency on online Madden NFL Football. In Proceedings of the 14th ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), County Cork, Ireland*, 2004.

[Nor12] Sten Nordell. *Network Latency - how low can you go? Lightwave*, November/December, 2012. http://online.qmags.com/LW1112?sessionID=9E2691A3BEE5C822E8B6821E2&cid=644386&eid=16872#pg19&mode1.

[NSR11] Michele Savi Norvald Stol and Carla Raffaelli. *3-Level Integrated Hybrid Optical Network (3LIHON) to Meet Future QoS Requirements. IEEE Globecom*, 2011.

[RVB] Steinar Bjørnstad Raimena Veisllari and Kurosh Bozorgebrahimi. *Integrated Packet/Circuit Hybrid Network Field-Trial.*

[RVH] Steinar Bjørnstad Raimena Veisllari and D.R. Hjelme. *Experimental demonstration of high throughput, ultra-low delay variation packet/circuit fusion network.*

[SBE06] V. L. Tuft O. Austad D. R. Hjelme S. Bjørnstad, M. Nord and L. E. Eriksen. *Experimental demonstration of OpMiGua hybrid circuit/packet nodes. Proc. SEE ECOC, Cannes, France*, 2006.

[SBS06] Dag Roar Hjelme Steinar Bjørnstad and Norvald Stol. *A Packet-Switched Hybrid Optical Network with Service Guarantees. IEEE Journal on Selected Areas in Communications, Vol. 24, No. 8*, August 2006.

[SF06] Ya-Yueh Shih and Kwoting Fang. *Effects of network quality attributes on customer adoption intentions of internet banking. Total Quality Management & Business Excellence 17, no. 1*, pages 61–77, 2006.

[Tod11] Sligo Today. *Transatlantic Super Cable To Boost Trading. Sligo Today*, September, 2011.
http://www.sligotoday.ie/printable.php?id=16381&PHPSESSID=2bb3019812b1c749f6514fc81974ec09.

[Tra11a] TransPacket. *Transpacket H1; a fusion networking add-drop muxponder.* 2011.
http://www.transpacket.com/wp-content/uploads/2011/12/TransPacket-H1-Product-overview-v1.pdf.

[Tra11b] TransPacket. *Transparent Ethernet.* 2011.
http://www.transpacket.com/hybrid-networks/virtual-wavelengths/.

[Tra12a] TransPacket. *Fusion. TransPacket Fusion Networks*, 2012.

[Tra12b] TransPacket. *Fusion networking explained. TransPacket White Paper*, June 11, 2012.

[Tra12c] TransPacket. *H1 Quick Start Guide.* November 13, 2012.

[Tra12d] TransPacket. *Technical Guide H1.* November 2, 2012.

[Vei10] Raimena Veisllari. *Employing Ethernet Multiple Spanning Tree Protocol in an OpMiGua Network. Master Thesis*, June 2010.

Appendix A

TransPacket H1 Technical Specifications

Line interface	10Gb/s rate, pluggable. Uncolored, CWDM 8 channels, DWDM up to 40 channels
	Tunable XFP optical interface
	Optional XFP for OTN support
Client interface	Up to 9 x 1GbE; SFP: Uncolored, CWDM or DWDM
Aggregation scheme	Up to 9GbE transparent Ethernet lines
	Up to 10GbE interfaces for statistical multiplexed aggregation
Latency	1.2 μs delay (RFC 1242 store and forward metric) for GE to 10GE aggregation
Latency variation	50 ns packet delay variation (PDV)
Optional plug-in	Optical WDM, OADM or other
Application areas	Metro networks Mobile backhaul networks High frequency trading High quality video transport Business access segment
Supported functionality	Transparent transport of all types of Ethernet-framed packets VLAN tags Add/Drop/Bypass features - Transparent Ethernet lines configured as bypass or add/drop - Packet channels add/drop or bypass with QoS guarantees Synchronous Ethernet ITU-T 1588 transparent; Extremely low latency and latency variation, enables transparent tunneling of packets
Quality of Service	Wavelength grade service on Transparent Ethernet ports - No packet loss - Future proof QoS Channel option: zero packet loss, minimum latency and latency variation
Management & OAM	In-band or out-of-band management CLI for installation and commissioning SNMP Netconf

	Web-browser RMON
Power consumption	Typical 48 W (Including optics)
	Dual 24 V or -48V feed
Physical size	43x430x300mm

The specifications and information presented here are subject to change without further notice, as TransPacket sees fit.

Appendix B

Master Thesis Outline and Time Plan

B.1 Outline

TransPacket is a startup-company that has implemented the novel fusion technology, also called OpMiGua integrated hybrid networks (www.transpacket.com). In the project the student will perform a network experiment involving TransPacket H1 nodes measuring performance parameters like latency, latency variation (packet delay variation) and packet loss. The experiment shall be performed in the premises of NTNU but remote operation of experimental equipment is available.

A rough outline of the master thesis is as follows:

- What is the problem? What is the motivation, objective and scope behind this project?
- Comparison of circuit-switching and packet-switching. Why do we need fusion technology?
- Background knowledge of the fusion technology and hybrid networks.
- Overview of TransPacket's H1 fusion networking muxponder.
- Previous work done and its results.
- Introduction to the lab environment, the software and tools used during experimentation.
- Network scenario used for the experiments.
- Experimental procedure.
- What were the results, what was achieved?
- Discussion of the results.
- Conclusions.
- Further work to be done.

B.2 Time Plan

Time Plan	
Jan 28	Master thesis start up
Weeks 5 & 6	Reading background theory
Week 7	Write final project description
Weeks 8 & 9	Start writing process
Week 10	Start experimentation in the lab
Week 11	Submit an interim report of the thesis to my supervisors for feedback
Weeks 12 to 15	Lab work and result gathering
Week 16	Submit a second report for feedback
Weeks 17 to 20	More lab work and experimentation
Week 21	Submit a third report for feedback
Weeks 22 to 24	Wrap up experimentation and document final results
Weeks 25 & 26	Polish written thesis and final delivery

Appendix C

H1 Configuration

C.1 Running Configuration

yangcli root@localhost> xget-config select=/ source=running

RPC Data Reply 1 for session 36:

```
rpc-reply {
    data {
        arp {
        }
        hadm {
            forward-unmatched-packets true
        }
        interfaces {
            interface xe0 {
                name xe0
                ethernet-switching {
                    trunk-interface {
                        vlans {
                            vlan vlan0 {
                                vlan-name vlan0
                            }
                            vlan vlan1 {
                                vlan-name vlan1
                            }
                            vlan vlan2 {
                                vlan-name vlan2
                            }
                            vlan vlan3 {
                                vlan-name vlan3
                            }
                            vlan vlan4 {
                                vlan-name vlan4
                            }
                            vlan vlan5 {
```

```
                    vlan-name vlan5
                }
                vlan vlan6 {
                    vlan-name vlan6
                }
                vlan vlan7 {
                    vlan-name vlan7
                }
                vlan vlan8 {
                    vlan-name vlan8
                }
                vlan vlan9 {
                    vlan-name vlan9
                }
            }
        }
    }
}
interface ge0 {
    name ge0
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan0
            }
        }
    }
}
interface ge1 {
    name ge1
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan1
            }
        }
    }
}
interface ge2 {
    name ge2
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan2
            }
        }
    }
}
```

```
interface ge3 {
    name ge3
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan3
            }
        }
    }
}
interface ge4 {
    name ge4
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan4
            }
        }
    }
}
interface ge5 {
    name ge5
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan5
            }
        }
    }
}
interface ge6 {
    name ge6
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan6
            }
        }
    }
}
interface ge7 {
    name ge7
    ethernet-switching {
        access-interface {
            vlan {
                vlan-name vlan7
            }
        }
```

```
        }
    }
    interface ge8 {
        name ge8
        ethernet-switching {
            access-interface {
                vlan {
                    vlan-name vlan8
                }
            }
        }
    }
    interface ge9 {
        name ge9
        ethernet-switching {
            access-interface {
                vlan {
                    vlan-name vlan9
                    priority gst
                }
            }
        }
    }
    interface me0 {
        name me0
        inet {
            address 158.38.152.166/24
        }
    }
}
nacm {
}
routes {
    route 0.0.0.0/0 {
        destination-prefix 0.0.0.0/0
        next-hop 158.38.152.1
    }
}
vlans {
    vlan vlan0 {
        id 100
        name vlan0
    }
    vlan vlan1 {
        id 101
        name vlan1
    }
    vlan vlan2 {
```

```
            id 102
            name vlan2
        }
        vlan vlan3 {
            id 103
            name vlan3
        }
        vlan vlan4 {
            id 104
            name vlan4
        }
        vlan vlan5 {
            id 105
            name vlan5
        }
        vlan vlan6 {
            id 106
            name vlan6
        }
        vlan vlan7 {
            id 107
            name vlan7
        }
        vlan vlan8 {
            id 108
            name vlan8
        }
        vlan vlan9 {
            id 109
            name vlan9
        }
    }
  }
}
```

C.2 VLAN Summary

yangcli root@localhost> show-vlans

RPC Data Reply 2 for session 37:

```
rpc-reply {
  info '>>
VLAN: vlan0
802.1Q Tag: 100
        xe0,    trunk
        ge0,    access, sm
```

```
VLAN: vlan1
802.1Q Tag: 101
        xe0,    trunk
        ge1,    access, sm

VLAN: vlan2
802.1Q Tag: 102
        xe0,    trunk
        ge2,    access, sm

VLAN: vlan3
802.1Q Tag: 103
        xe0,    trunk
        ge3,    access, sm

VLAN: vlan4
802.1Q Tag: 104
        xe0,    trunk
        ge4,    access, sm

VLAN: vlan5
802.1Q Tag: 105
        xe0,    trunk
        ge5,    access, sm

VLAN: vlan6
802.1Q Tag: 106
        xe0,    trunk
        ge6,    access, sm

VLAN: vlan7
802.1Q Tag: 107
        xe0,    trunk
        ge7,    access, sm

VLAN: vlan8
802.1Q Tag: 108
        xe0,    trunk
        ge8,    access, sm

VLAN: vlan9
802.1Q Tag: 109
        xe0,    trunk
        ge9,    access, gst

<<'
}
```

Appendix D

Raw Data

D.1 SM 10%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,511	732,130,897	4,632,855,825,552	4,632,855,630,264	4,749,996,707,312	4,749,996,573,784
Port 2	77,065,384	77,065,757	487,667,646,456	487,645,822,328	499,998,107,896	499,976,343,448
Port 3	77,065,757	77,065,384	487,645,822,328	487,667,646,456	499,976,343,448	499,998,107,896
Port 4	732,130,897	732,130,511	4,632,855,630,264	4,632,855,825,552	4,749,996,573,784	4,749,996,707,312

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	88.44	26.94	174.49	0	0.000	34.31	112.29
Port 3 -> 2	88.48	27.01	171.95	0	0.000	34.22	108.9
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.2 SM 20%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,133,334	732,131,463	4,632,854,408,656	4,632,855,345,896	4,749,995,742,096	4,749,996,379,976
Port 2	154,130,941	154,130,408	975,332,880,808	975,346,816,200	999,993,831,368	1,000,007,681,480
Port 3	154,130,408	154,130,941	975,346,816,200	975,332,880,808	1,000,007,681,480	999,993,831,368
Port 4	732,131,463	732,133,334	4,632,855,345,896	4,632,854,408,656	4,749,996,379,976	4,749,995,742,096

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	96.32	26.93	184.53	0	0.000	28.07	104.59
Port 3 -> 2	96.31	26.94	177.12	0	0.000	27.98	103.31
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.55

D.3 SM 30%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,126,957	732,129,916	4,632,857,601,976	4,632,856,122,840	4,749,997,915,096	4,749,996,909,400
Port 2	231,197,287	231,197,346	1,462,989,791,840	1,462,988,887,040	1,499,981,357,760	1,499,980,462,400
Port 3	231,197,346	231,197,287	1,462,988,887,040	1,462,989,791,840	1,499,980,462,400	1,499,981,357,760
Port 4	732,129,916	732,126,957	4,632,856,122,840	4,632,857,601,976	4,749,996,909,400	4,749,997,915,096

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	106.13	26.88	185.21	0	0.000	21.46	115.89
Port 3 -> 2	106	27.07	182.03	0	0.000	21.16	115.32
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.4 SM 40%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,891	732,133,431	4,632,855,629,608	4,632,854,355,936	4,749,996,572,168	4,749,995,704,896
Port 2	308,262,068	308,263,532	1,950,671,633,848	1,950,657,153,400	1,999,993,564,728	1,999,979,318,520
Port 3	308,263,532	308,262,068	1,950,657,153,400	1,950,671,633,848	1,999,979,318,520	1,999,993,564,728
Port 4	732,133,431	732,130,891	4,632,854,355,936	4,632,855,629,608	4,749,995,704,896	4,749,996,572,168

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	112.87	27.26	185.54	0	0.000	16.52	111.63
Port 3 -> 2	112.96	27.27	185.03	0	0.000	16.32	109.23
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.5 SM 50%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,006	732,129,474	4,632,856,075,784	4,632,856,340,344	4,749,996,876,744	4,749,997,056,184
Port 2	385,329,763	385,329,604	2,438,327,745,176	2,438,328,796,648	2,499,980,507,256	2,499,981,533,288
Port 3	385,329,604	385,329,763	2,438,328,796,648	2,438,327,745,176	2,499,981,533,288	2,499,980,507,256
Port 4	732,129,474	732,130,006	4,632,856,340,344	4,632,856,075,784	4,749,997,056,184	4,749,996,876,744

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	118.92	27.45	194.75	0	0.000	12.87	108.97
Port 3 -> 2	119.03	27.63	193.39	0	0.000	12.77	108.25
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.6 SM 60%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,126,957	732,129,916	4,632,857,601,976	4,632,856,122,840	4,749,997,915,096	4,749,996,909,400
Port 2	462,395,672	462,397,726	2,926,001,283,064	2,925,992,079,232	2,999,984,590,584	2,999,975,715,392
Port 3	462,397,726	462,395,672	2,925,992,079,232	2,926,001,283,064	2,999,975,715,392	2,999,984,590,584
Port 4	732,129,916	732,126,957	4,632,856,122,840	4,632,857,601,976	4,749,996,909,400	4,749,997,915,096

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	125.48	29.01	196.67	0	0.000	9.87	106.24
Port 3 -> 2	125.6	29.14	199.58	0	0.000	9.84	106.04
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.53
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.55

D.7 SM 70%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,891	732,131,812	4,632,855,629,608	4,632,855,172,408	4,749,996,572,168	4,749,996,262,328
Port 2	539,462,074	539,461,730	3,413,673,243,632	3,413,674,255,008	3,499,987,175,472	3,499,988,131,808
Port 3	539,461,730	539,462,074	3,413,674,255,008	3,413,673,243,632	3,499,988,131,808	3,499,987,175,472
Port 4	732,131,812	732,130,891	4,632,855,172,408	4,632,855,629,608	4,749,996,262,328	4,749,996,572,168

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	132.55	31.09	204.42	0	0.000	7.17	101.29
Port 3 -> 2	132.76	33.53	214.58	0	0.000	7.17	93
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.55
Port 4 -> 1	21.47	20.81	21.6	0	0.000	0.05	0.56

D.8 SM 80%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,052	732,129,797	4,632,856,054,128	4,632,856,178,640	4,749,996,862,448	4,749,996,946,160
Port 2	616,530,651	616,529,093	3,901,342,381,056	3,901,345,159,512	3,999,987,285,216	3,999,989,814,392
Port 3	616,529,093	616,530,651	3,901,345,159,512	3,901,342,381,056	3,999,989,814,392	3,999,987,285,216
Port 4	732,129,797	732,130,052	4,632,856,178,640	4,632,856,054,128	4,749,996,946,160	4,749,996,862,448

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	140.89	45.75	211.53	0	0.000	4.6	75.97
Port 3 -> 2	141.12	62.44	216.44	0	0.000	4.64	71.84
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.55

D.9 SM 90%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,126,892	732,129,850	4,632,857,636,080	4,632,856,155,256	4,749,997,938,800	4,749,996,931,256
Port 2	693,594,213	693,597,928	4,389,020,571,496	4,389,017,298,600	4,499,995,645,576	4,499,992,967,080
Port 3	693,597,928	693,594,213	4,389,017,298,600	4,389,020,571,496	4,499,992,967,080	4,499,995,645,576
Port 4	732,129,850	732,126,892	4,632,856,155,256	4,632,857,636,080	4,749,996,931,256	4,749,997,938,800

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	154.65	48.38	229.09	0	0.000	2.22	72.44
Port 3 -> 2	154.88	102.93	239.85	0	0.000	2.26	66.73
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.10 SM 95%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,057	732,132,244	4,632,856,055,104	4,632,854,957,944	4,749,996,864,224	4,749,996,116,984
Port 2	732,130,246	732,131,791	4,632,855,952,840	4,632,855,178,296	4,749,996,792,200	4,749,996,264,856
Port 3	732,131,791	732,130,246	4,632,855,178,296	4,632,855,952,840	4,749,996,264,856	4,749,996,792,200
Port 4	732,132,244	732,130,057	4,632,854,957,944	4,632,856,055,104	4,749,996,116,984	4,749,996,864,224

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	178.66	52.36	269.82	0	0.000	1.13	73.32
Port 3 -> 2	178.93	103.36	261.79	0	0.000	1.13	61.45
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.11 SM 96%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,129,108	732,133,105	4,632,856,531,144	4,632,854,519,624	4,749,997,188,424	4,749,995,816,424
Port 2	739,837,667	739,837,265	4,681,623,201,680	4,681,623,369,584	4,799,997,228,400	4,799,997,331,984
Port 3	739,837,265	739,837,667	4,681,623,369,584	4,681,623,201,680	4,799,997,331,984	4,799,997,228,400
Port 4	732,133,105	732,129,108	4,632,854,519,624	4,632,856,531,144	4,749,995,816,424	4,749,997,188,424

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	194.25	52.39	292.06	0	0.000	0.92	68.46
Port 3 -> 2	194.49	108.75	286.61	0	0.000	0.91	55.07
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.55

D.12 SM 97%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,129,165	732,129,700	4,632,856,502,992	4,632,856,233,168	4,749,997,169,392	4,749,996,985,168
Port 2	747,544,696	747,543,428	4,730,390,689,256	4,730,391,155,592	4,849,997,840,616	4,849,998,104,072
Port 3	747,543,428	747,544,696	4,730,391,155,592	4,730,390,689,256	4,849,998,104,072	4,849,997,840,616
Port 4	732,129,700	732,129,165	4,632,856,233,168	4,632,856,502,992	4,749,996,985,168	4,749,997,169,392

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	255.72	50.75	423.62	0	0.000	0.71	70.27
Port 3 -> 2	255.13	112.43	412.88	0	0.000	0.71	64.34
Port 1 -> 4	21.47	20.79	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.13 SM 97.2%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,131,050	732,133,334	4,632,855,551,528	4,632,854,408,656	4,749,996,519,528	4,749,995,742,096
Port 2	749,084,760	749,087,568	4,740,144,656,512	4,740,143,687,168	4,859,998,218,112	4,859,997,698,048
Port 3	749,087,568	749,084,760	4,740,143,687,168	4,740,144,656,512	4,859,997,698,048	4,859,998,218,112
Port 4	732,133,334	732,131,050	4,632,854,408,656	4,632,855,551,528	4,749,995,742,096	4,749,996,519,528

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	302.05	52.46	549.66	0	0.000	0.67	77.5
Port 3 -> 2	301.32	105.6	557.74	0	0.000	0.67	63.32
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.55
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.54

D.14 SM 97.5%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,477	732,132,836	4,632,855,839,144	4,632,854,657,728	4,749,996,715,464	4,749,995,911,488
Port 2	751,397,630	751,397,006	4,754,774,651,152	4,754,774,849,688	4,874,998,271,952	4,874,998,370,648
Port 3	751,397,006	750,776,430	4,754,774,849,688	4,749,898,044,040	4,874,998,370,648	4,870,022,272,840
Port 4	732,132,836	732,130,477	4,632,854,657,728	4,632,855,839,144	4,749,995,911,488	4,749,996,715,464

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	262,649.27	50.62	275,283.11	621,200	0.082	0.64	122.08
Port 3 -> 2	7,133.08	108.62	59,352.89	0	0.000	0.61	97.36
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.55

D.15 SM 97.8%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,477	732,132,836	4,632,855,839,144	4,632,854,657,728	4,749,996,715,464	4,749,995,911,488
Port 2	753,708,810	749,333,561	4,769,405,185,888	4,736,346,264,008	4,889,998,595,488	4,856,239,633,768
Port 3	753,710,460	750,506,487	4,769,404,681,560	4,745,196,965,672	4,889,998,355,160	4,865,278,003,592
Port 4	732,132,836	732,130,477	4,632,854,657,728	4,632,855,839,144	4,749,995,911,488	4,749,996,715,464

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	517,695.65	51.78	544,692.02	3,202,323	0.424	0.67	113.73
Port 3 -> 2	275,091.54	105.41	346,964.99	4,376,899	0.580	0.71	124.05
Port 1 -> 4	21.47	20.8	21.59	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.8	21.6	0	0.000	0.05	0.56

D.16 SM 98%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,129,165	732,129,700	4,632,856,502,992	4,632,856,233,168	4,749,997,169,392	4,749,996,985,168
Port 2	755,250,052	748,717,998	4,779,158,732,088	4,729,882,738,888	4,899,998,740,408	4,849,677,618,568
Port 3	755,250,948	749,502,194	4,779,158,472,472	4,735,660,597,232	4,899,998,624,152	4,855,580,948,272
Port 4	732,129,700	732,129,165	4,632,856,233,168	4,632,856,502,992	4,749,996,985,168	4,749,997,169,392

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	532,510.53	51.85	547,566.25	5,747,858	0.761	0.72	126.06
Port 3 -> 2	513,152.71	103.11	573,188.61	6,532,950	0.865	0.75	128.8
Port 1 -> 4	21.47	20.8	21.6	0	0.000	0.05	0.54
Port 4 -> 1	21.47	20.81	21.6	0	0.000	0.05	0.54

D.17 SM 99%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,127,348	732,131,888	4,632,857,407,608	4,632,855,137,216	4,749,997,783,288	4,749,996,239,296
Port 2	762,960,341	730,127,698	4,827,925,482,312	4,580,199,212,752	4,949,999,136,872	4,697,019,644,432
Port 3	762,956,673	750,126,943	4,827,926,308,280	4,729,540,680,024	4,949,999,375,960	4,849,560,990,904
Port 4	732,131,888	732,127,348	4,632,855,137,216	4,632,857,407,608	4,749,996,239,296	4,749,997,783,288

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	740,926.28	50.56	829,094.49	12,833,398	1.682	0.78	183.41
Port 3 -> 2	684,126.46	107.77	877,712.63	32,828,975	4.302	1.51	158.74
Port 1 -> 4	21.47	20.81	21.6	0	0.000	0.05	0.55
Port 4 -> 1	21.47	20.8	21.59	0	0.000	0.05	0.54

D.18 SM 99.5%

	Total Tx Count (frames)	Total Rx Count (frames)	Total Tx Count (bits)	Total Rx Count (bits)	Tx L1 (bits)	Rx L1 (bits)
Port 1	732,130,850	732,126,957	4,632,855,653,856	4,632,857,601,976	4,749,996,589,856	4,749,997,915,096
Port 2	766,808,243	724,004,546	4,852,310,426,520	4,529,716,567,080	4,974,999,745,400	4,645,557,294,440
Port 3	766,811,322	748,614,015	4,852,309,836,952	4,713,982,791,080	4,974,999,648,472	4,833,761,033,480
Port 4	732,126,957	732,130,850	4,632,857,601,976	4,632,855,653,856	4,749,997,915,096	4,749,996,589,856

	Avg Latency (μs)	Min Latency (μs)	Max Latency (μs)	Dropped Count (Frames)	Dropped Frame %	Avg Jitter (μs)	Max Jitter (μs)
Port 2 -> 3	514,354.85	52.03	720,923.39	18,194,228	2.372	0.84	147.77
Port 3 -> 2	760,839.1	104.61	898,070.25	42,806,776	5.582	1.75	161.2
Port 1 -> 4	21.47	20.8	21.6	0	0.000	0.05	0.53
Port 4 -> 1	21.47	20.8	21.59	0	0.000	0.05	0.54