Design and Performance Evaluation of Small-buffered Optical Packet Switched Networks

Submitted to
Graduate School of Information Science and Technology
Osaka University

July 2008

Onur ALPARSLAN

List of Publications

Journal papers

1. O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for small buffered optical packet-switched networks," *Journal of Optical Networking*, vol. 6, no. 9, pp. 1116–1128, September 2007.

2. O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for optical packet switched networks with very small optical RAM," *Photonic Network Communications*, 2008, submitted.

3. O. Alparslan, S. Arakawa, and M. Murata, "Node pacing for optical RAM buffered packet switching networks," *Journal of Optical Networking*, 2008, submitted.

Refereed Conference papers

1. O. Alparslan, S. Arakawa, and M. Murata, "Optical rate-based paced XCP for small buffered optical packet switching networks," in *Proceedings of PFLDnet*, February 2006, pp. 117–124.

2. O. Alparslan, S. Arakawa, and M. Murata, "Performance of paced and non-paced transmission control algorithms in small buffered networks," in *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, June 2006, pp. 115–122. (acceptance ratio 46.3%)

3. O. Alparslan, S. Arakawa, and M. Murata, "Switch architectures for small-buffered optical packet switched networks," in *Proceedings of 12th IEEE Symposium on Computers and Communications (ISCC'07)*, July 2007. (acceptance ratio 40.3%)

4. O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for optical packet switched networks with very small optical RAM," in *Proceedings of IEEE Broadnets*, September 2007.

5. O. Alparslan, S. Arakawa, and M. Murata, "Node pacing for optical packet switching," in *Proceedings of Photonics in Switching*, August 2008, to appear.

Non-Refereed Technical papers

1. O. Alparslan, S. Arakawa, and M. Murata, "Rate-based paced XCP for small buffered optical packet switched networks," IEICE (PN2006-7), Tech. Rep., pp. 35–40, May 2006.

2. O. Alparslan, S. Arakawa, and M. Murata, "A comparative study of switch architectures for small-buffered optical packet switched networks," IEICE (OPE2006-159), Tech. Rep., pp. 167–172, January 2007.

3. O. Alparslan, S. Arakawa, and M. Murata, "Node pacing for optical packet switching," IEICE (PN2007-76), Tech. Rep., pp. 17–20, March 2008.

Preface

Recent advances in the optical networks like dense wavelength division multiplexing (DWDM), which combines and transmits multiple signals simultaneously at different wavelengths on the same fiber, allow achieving ultra high data transmission rates in optical networks. While optical transmission rates were getting faster and faster, electronic components in routers like electronic RAM could not pace due to high costs and electronic limitations and became a bottleneck. Therefore, it became necessary to do some basic operations like buffering and switching in the optical domain without converting packets to electronic domain. However, the lack of high capacity optical RAM makes it difficult to buffer optical packets in optical packet switched (OPS) networks. According to a rule-of-thumb, buffer size of each output link of a router must be $B = RTT \times BW$, where RTT is the average round trip time of flows and BW is the bandwidth of output link, in order to achieve high utilization with TCP flows. However it requires a huge memory size due to high speed of optical links, so it is currently unfeasible.

Currently, the only available solution that can be used for buffering in the optical domain is using FDLs. Contended packets are switched to FDLs in order to be delayed. However, FDLs have important limitations. First of all, FDLs require very long fiber lines that cause signal attenuation inside the routers. There can be a limited number of FDLs in a router due to space considerations, so they can provide a small amount of buffering. Second, FDLs provide only a fixed amount of delay. FDL buffering is possible with today's technology, so many researchers consider FDL buffers to resolve contentions in optical networks. On the other hand, optical RAM is under research and it may be available in the near future.

Optical RAM solves the problems of FDLs like lack of real $O(1)$ reading operation, signal attenuation and bulkiness. However, Optical RAM is not expected to have a large capacity, soon. Therefore, decreasing the huge buffer requirements of OPS networks is necessary in order to make use of optical buffering.

In this thesis, we propose methods for minimizing the optical buffer size requirements of optical networks. We evaluate buffer requirements in case of both FDL and optical RAM buffering. We show that real $O(1)$ reading capability of optical RAM allows further decreasing the buffer size requirements when compared with FDL buffering.

A well-known technique for decreasing buffer requirements is applying pacing, so first we evaluate some paced transmission protocols and compare their performance in small-buffered networks. Simulation results show that all non-paced TCP and XCP variants perform poorly. We show that even pacing alone is not enough for XCP with default parameters to make suitable for small buffered networks, so we introduce a new parameter set for XCP. Simulation results show that buffer requirements on routers are greatly reduced by paced XCP with suitable parameter settings, while keeping the high fairness, fast convergence, and high utilization.

Based on these results, we design and propose a new XCP framework called Rate-based Paced XCP specially designed for small buffered OPS networks. We evaluate the FDL requirements on a meshed network with multiple-hop paths and show how FDL requirements change with slot size, utilization, FDL granularity, scheduling and packet size distribution.

As a next step, we propose new switch architectures specially designed for Rate-based Paced XCP for further decreasing the buffer requirements and switch costs. We investigate and compare input and output buffered optical switch architectures. We show how the FDL requirements of different switch architectures change with FDL granularity and packet size distribution by using a star topology.

We next focus on optical RAM. We replace FDL buffering with optical RAM and evaluate the buffer size requirements in our architecture. We show that proposed architecture has low buffer requirements with optical RAM due to advantage of $O(1)$ reading operation of optical RAM when compared with FDLs.

Using XCP framework may increase the implementation cost of the proposed small buffer architecture. As a last step, by using the ideas and results from the rate-based paced XCP, we propose a new core pacing architecture that is very light-weight, easier to implement and does not require XCP framework.

Simulations show that both of our proposed pacing algorithms and network architectures considerably increase the achievable utilization of very small optical FDL or RAM buffered optical core links or namely throughput of TCP flows using these links.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor Masayuki Murata for his guidance, patience, valuable discussions, and meaningful advices throughout my studies. This thesis would not be possible without his advice and suggestions.

I am deeply grateful to Assistant Professor Shin'ichi Arakawa for his continuous support, appropriate guidance, patience, and invaluable advice. His critical comments and guidance were always informative and helpful.

I also thank to Professor Hirotaka Nakano, Professor Teruo Higashino, Professor Makoto Imase and Professor Koso Murakami for reviewing this thesis and their helpful comments.

I want to thank secretaries, friends and colleagues in the Graduate School of Information Science and Technology of Osaka University for their help.

Finally, I am deeply grateful to my parents for their understanding and support.

Contents

List of Figures

List of Table

Chapter 1

Introduction

1.1 Background

Optical packet-switched (OPS) networks have some major differences and limitations when compared with electronic packet-switched (EPS) networks. One of the difficulties of OPS networks is buffering optical packets in the network. In EPS networks, contention is resolved by storing the contended packets in a random access memory (RAM) and sending out the packets with $O(1)$ reading operation when the output port is free. However, the operation is not possible in the optical domain, because there is no equivalent high capacity optical RAM available for storing packets. Converting packets from optical domain to electronic domain in order to use electronic RAM is not a feasible solution because of the processing limitations of EPS. Current electronic devices are not fast enough to process the data at the ultra high-speed of optical networks. Therefore, processing and switching in the optical domain is necessary.

Many researchers consider long fiber lines called Fiber Delay Line (FDL) buffers to resolve contentions in optical networks, because optical RAM is infeasible or has immature technology. Optical RAM is under research (for example, Takahashi et. al. [1] and NICT project (phase II) [2]). Basic operation of optical RAM is experimentally confirmed for 40-Gbit/s 16-bit optical packets in Ref. [1]. However, optical RAM is not expected to have

a large capacity, soon. Currently, the only available solution that can be used for buffering in the optical domain is using FDLs. Contended packets are switched to FDLs in order to be delayed. However, FDLs have important limitations. First of all, FDLs require very long fiber lines that cause signal attenuation, inside the routers. There can be a limited number of FDLs in a router due to space considerations, so they can provide a small amount of buffering. Second, FDLs provide only a fixed amount of delay.

On the other hand, optical RAM is under research and it may be available in the near future. Optical RAM solves the problems of FDLs like lack of real $O(1)$ reading operation, signal attenuation and bulkiness. Furthermore, optical RAM is expected to have a low power consumption rate that is an important problem for electronic RAM. However, Optical RAM is not expected to have a large capacity, soon. Therefore, decreasing the huge buffer requirements of OPS networks is a must in order to make use of optical buffering.

Having a very small buffering capacity brings some important performance problems to optical packet switched (OPS) networks. According to a rule-of-thumb [3], an output link of a router needs a buffer sized at $B = RTT \times BW$, where RTT is the average round trip time of flows and BW is the bandwidth of output link, in order to achieve high utilization with TCP flows. Recently, Appenzeller et al. [4] showed that when there are many TCP flows sharing the same link, a buffer sized at $B = \frac{RTT \times BW}{\sqrt{n}}$, where n is the number of TCP flows passing through the link, is enough for achieving high utilization. However, a significant decrease in buffer requirements is possible only when there are many flows on the link. This buffer requirement is still high for high speed OPS routers with very small amount of buffering capacity. Further decreasing the buffer requirements is necessary. However, bursty nature of TCP flows causes a high packet drop rate in small buffered networks and limits further decreasing the buffer size.

Recently, [5] proposed that $O(\log W)$ buffers are sufficient where W is the maximum congestion window size of flows when packets are sufficiently paced by modifying TCP senders to used Paced TCP [6] or by using slow access links. Pacing is defined as transmitting ACK (data) packets according to a special criteria, instead of transmitting immediately upon arrival of a data (ACK) packet [6]. However, $O(\log W)$ buffer size depends on the maximum

congestion window size of TCP flows that may change in time. Moreover, using slow access links is not a preferred solution when there are applications that require high-bandwidth on the network. Using Paced TCP for these applications by replacing TCP senders with paced versions can be hard. Furthermore, this proposal was based on the assumption that most of the IP traffic is from TCP flows. A recent paper [7] shows that even small quantities of bursty real-time traffic can interact with well-behaved TCP traffic and increase the buffer requirements.

It may be better to design a general architecture for OPS networks that:

- Can achieve high utilization in a small buffered OPS network independent of the number of TCP or UDP flows;
- Does not require limiting the speed of access links;
- Does not require replacing sender or receiver agents of computers using the network.

Applying pacing to the input traffic at the edge nodes of an OPS network can be a good choice for achieving these goals. Even if TCP pacing is applied at the clients, the aggregated traffic arriving to the OPS network may end up behaving bursty. Therefore, pacing at the edge of OPS network is more effective on minimizing burstiness of traffic entering the OPS domain. Ref. [8] proposes applying traffic shaping at edge nodes of OPS network for minimizing traffic burstiness. It proposes a delay-based pacing algorithm that adaptively chooses packet spacing according to input traffic class for achieving bounded delay requirements. Ref. [9] proposes traffic shaping at edge nodes by using a modified form of renegotiated service with rate prediction to reduce contention in the core. Edge nodes use packet scheduling to rearrange possible contended slots before entering the core, thus reducing core optical buffering. Proposed architecture needs information of relevant network scheduling.

Another possible solution is applying pacing at the core nodes. Ref. [10] proposes a RC traffic shaper for ATM networks that smooths the traffic by adjusting the output rate based on the buffer occupancy that depends on the input traffic rate. Output rate is linearly

proportional to the shaping buffer occupancy. The problem of the proposed algorithm is that it requires a large buffer for preventing cell loss. Large buffer brings higher delay. Furthermore, peak cell input rate must be known. In order to solve these problems, Ref. [11] proposes Interval Filter Shaping Algorithm (IFSA) that smooths cell inter-arrival time with a low latency. IFSA makes use of a low pass filter and special scheduler for smoothing the traffic. The problem of core pacing is that buffer capacity may not be enough for both applying pacing a providing buffering for contending packets at the same time as the buffer size is very small. Small buffer size of core nodes usually causes core pacing to have lower performance than edge pacing that can make use of big electronic buffers for incoming electronic packets.

In this thesis, we propose methods for minimizing the buffer size of optical networks. We propose small buffered architectures for both edge node pacing and core node pacing using optical RAM and FDL-based buffering.

1.2 Outline of Thesis

Rate-based Pacing for Small-buffered Optical Packet Switched Networks [12, 13, 14, 15]

We propose an all-optical network architecture that can achieve high utilization and low packet drop ratio in small buffered OPS networks. XCP [16] is a new congestion control algorithm using a control theory framework. XCP was specifically designed for high-bandwidth and large-delay networks. XCP was first proposed in Ref. [16] as a window-based reliable congestion and transmission control algorithm. XCP framework is selected because XCP framework allows individual control of the utilization level of each wavelength. We show that selecting a target wavelength utilization less than actual wavelength capacity in XCP control algorithm can allow operating at a utilization level that can give a low packet drop rate for a selected FDL granularity. In our architecture, if there is traffic between an edge source-destination node pair, a rate-based XCP macro flow is created, and incoming

TCP and UDP packets of this edge pair are assigned the XCP macro flow as shown in Fig. 2.1 similar to TeXCP [17]. The edge nodes of OPS network apply leaky-bucket pacing to the macro flows by using the rate information provided by XCP for minimizing the burstiness. As a result, there is no need to modify the TCP and UDP agents of computers or limit the speed of access links for decreasing burstiness. We evaluate the FDL requirements and packet drop rates on a realistic mesh NSFNET topology with multiple-hop paths. We show how FDL requirements change with utilization, FDL granularity, scheduling, and packet size distribution.

Switch Architectures for Small-buffered Optical Packet Switched Networks [18]

We investigate and compare input and output buffered optical switch architectures for minimizing the size of optical switching fabric of core nodes while achieving high throughput with small FDL-based buffers. Switching fabric size is an important cost factor in routers, so it is necessary to design a suitable switch architecture for the proposed small-buffered architecture. Many switching fabric architectures like MEMS, optomechanical, electrooptic, thermooptic, liquid-crystal based switches are proposed for optical switching [19]. However, the number of switching elements in the fabric increases together with the overall cost as the number of ports of the switch increases. Moreover increasing the switch size introduces high crosstalk and insertion losses in many proposed switching fabric architectures. We compare the buffering architectures input buffering with VOQ, and output buffering with void filling, by using a star topology. We evaluate the packet drop rates depending on FDL granularity and packet size distribution. We show that input buffering requires comparable number of delay lines as output buffering architectures at 30% utilization, which is typical for backbone links of network operators, with pacing.

Pacing for Optical Packet Switched Networks with Very Small Optical RAM [20, 21, 22]

First, we evaluate several transmission control algorithms in small buffered networks. The algorithms include TCP Reno, TCP NewReno, Highspeed TCP with SACK, and XCP. Simulation results show that all non-paced TCP and XCP variants perform poorly. Furthermore, the results show that even rule-of-thumb buffers are not enough for XCP in some cases because of high burstiness of XCP. We evaluate the effectiveness of pacing method at sender side for making transmission control algorithms suitable for very small buffered networks. We show that pacing alone is not enough for XCP to make suitable for small buffered networks, so we introduce a suitable parameter set. Simulation results show that buffer requirements on routers are greatly reduced by paced XCP with suitable parameter settings, while keeping the high fairness, fast convergence, and high utilization.

As a next step, we evaluate the optical RAM size requirements of our rate-based XCP pacing architecture. We show that proposed architecture has very low buffer requirements with optical RAM due to advantage of $O(1)$ reading operation of optical RAM when compared with FDLs.

Node Pacing for Optical Packet Switching [23, 24, 25]

In the previous chapters, we proposed and evaluated an architecture for decreasing buffering requirements by using XCP framework. However, implementation cost XCP framework can be high. Therefore, we propose a light-weight alternative architecture that can shape traffic at edge or core nodes by using the buffer occupancy information. Ref. [10] proposes a RC traffic shaper for ATM networks that smooths the traffic by adjusting the output rate based on the buffer occupancy that depends on the input traffic rate. The problem of RC traffic shaper is that it requires a large buffer for preventing cell loss. Large buffer brings higher delay. Furthermore, peak cell input rate must be known. Our architecture does not have these problems because our architecture applies pacing by using a piecewise linear output transfer rate control function and making use of average input traffic rate information calculated inside the node. This architecture does not require XCP framework, so it is easier and cheaper to implement.

We evaluate the proposed architecture on an Abilene-inspired mesh topology. We compare different combinations of edge and/or core node pacing architectures with non-paced TCP and paced TCP flows. We show that our node pacing algorithm can considerably increase the achievable utilization of very small optical RAM buffered optical core links or namely throughput of TCP flows using these links.

Chapter 2

Rate-based Pacing for Small-buffered Optical Packet Switched Networks

2.1 Introduction

In this chapter, we introduce an all-optical OPS network architecture that can achieve high utilization and low packet drop ratio by using FDL-based small buffering. We consider an OPS domain where packets enter and exit the OPS domain through edge nodes. We propose using a Explicit Congestion Control Protocol(XCP) based [16] intra-domain congestion control protocol for achieving high utilization and low packet drop ratio with small FDL buffers. XCP [16] is a new congestion control algorithm using a control theory framework. XCP was specifically designed for high-bandwidth and large-delay networks. XCP was first proposed in Ref. [16] as a window-based reliable congestion and transmission control algorithm. XCP framework is selected because XCP framework allows individual control of the utilization level of each wavelength. We show that selecting a target wavelength utilization less than actual wavelength capacity in XCP control algorithm can allow

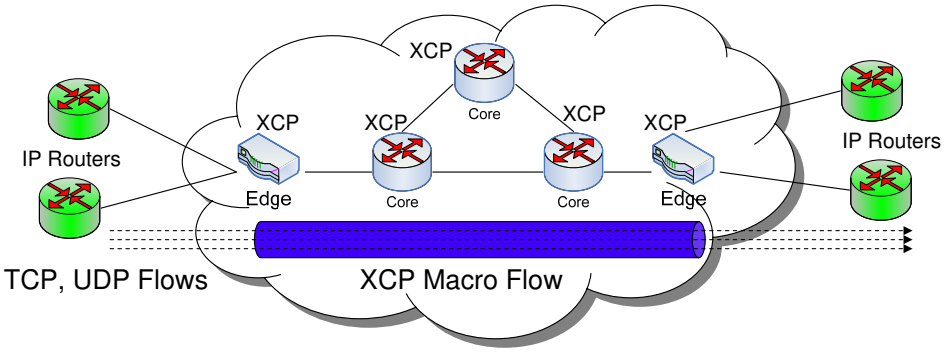


Figure 2.1: XCP macro flows.

operating at a utilization level that can give a low packet drop rate for a selected FDL granularity. XCP is specially designed for high-bandwidth and large-delay networks that makes XCP well-suited to optical networks. In our architecture, if there is traffic between an edge source-destination node pair, a rate-based XCP macro flow is created, and incoming TCP and UDP packets of this edge pair are assigned the XCP macro flow as shown in Fig. 2.1 similar to TeXCP [17]. The edge nodes of OPS network apply leaky-bucket pacing to the macro flows by using the rate information provided by XCP for minimizing the burstiness. As a result, there is no need to modify the TCP and UDP agents of computers or limit the speed of access links for decreasing burstiness.

We evaluate the FDL requirements and packet drop rates on a realistic mesh NSFNET topology with multiple-hop paths. We show how FDL requirements change with utilization, FDL granularity, scheduling, and packet size distribution.

2.2 FDL Architecture, Slots Size and Voids

FDL architecture used in this chapter is a single stage equidistant FDL set with B delay lines. Switch and FDL architecture is shown in Fig. 2.2 [26].

In the FDL architecture, length of delay lines will be given in terms of slot number. FDL length distribution increases linearly (x, $2x$, $3x$, $4x\ldots$) where x is FDL granularity. The

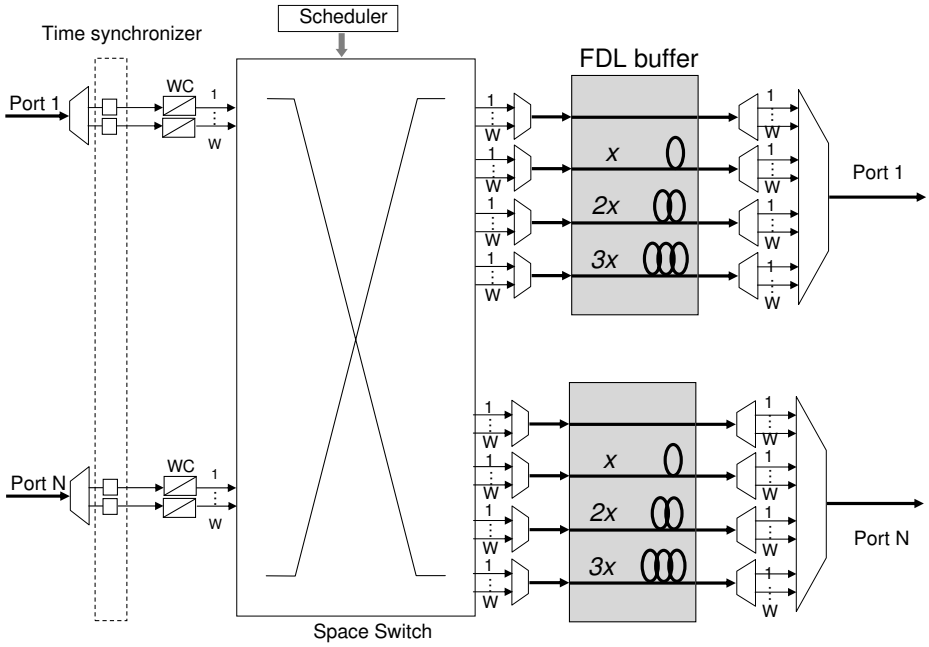


Figure 2.2: Switch and FDL architecture.

number of required FDLs (denoted by B) will be evaluated for different FDL granularities. Minimizing the number of delay lines in a switch is necessary because the size and the cost of the switching fabric increases as the number of delay lines increase. Size of the switching fabric is the main cost factor of routers. Increasing the granularity can decrease the number of required FDL lines and therefore decrease the cost of the switch. However, selecting a too high granularity may increase the packet loss rate as we will show in the simulations.

We are using Slotted Variable-Length Packet Approach (SVLP)[27] for adapting asynchronous, variable-length packets coming from the electrical domain to the synchronous and slotted OPS network. Variable sized IP packets divided into variable number of slots enter OPS network as a train of slots without any burst assembling. OPS routers switch slot train of a packet as a whole and sequentially. Using a slotted architecture decreases the packet drop rate, but requires slot synchronization before switching.

In this approach, because using FDLs and a slot-based architecture may cause voids,

which decrease the effective throughput of output links. Size of voids depends on the slot size, so slot size is an important design parameter that affects the effective utilization and buffer requirements. Voids in the architecture can be classified into following two groups.

2.2.1 Voids in Slots

Voids in slots occur when the packet size is not equal to a multiple of slot size. In this case, padding is applied to the last slot. Padding decreases the utilization efficiency depending on the packet size distribution and the slot size. For example, if the slot size is 500Bytes and if a 501Bytes packet arrives, the packet will be carried in two slots with a total length of 1000Bytes. There will be a 409Bytes void due to padding in the second slot. Using a small slot size can increase the efficiency in general due to less padding. However, efficiency increase may not be high at too small slot sizes as guard bands and slot headers start becoming the main source of decrease in efficiency. Furthermore, buffer requirements may increase as switching operation becomes similar to an unslotted network.

If we select a large slot size, effective throughput is low when the average size of arriving packets is small. For example, if the slot size is 1500Bytes and a 40Bytes packet is carried in the slot, 97.3% of the slot is wasted due to padding (void).

Selecting a right MTU for the OPS link layer is important. Selecting a MTU less than slot payload size does not make sense, because a single packet can never fully utilize the slot capacity. We do not consider burst assembling, so assembling multiple packets in order to fill in voids a slot is not a choice. MTU can be selected as any value equal to or bigger than slot capacity as SVLP-based adaptation divides variable sized IP packets into variable number of slots transparently. On the other hand, the extreme case of using only a single slot for carrying a whole packet without dividing into slots by selecting the slot payload size and MTU of OPS link layer equal to each other can greatly simplify the FDL design process, as FDL granularity can be directly selected as 1 slot size independent of packet size distribution. Furthermore, scheduling algorithms become simpler because there is no need to take into account the length of a train of slots as all packets are inside a single slot

[27] and there is no need for void filling scheduling when FDL granularity is 1 slot size.

2.2.2 Void Slots in FDLs Between Packets

When slot size is small, buffer requirements can be decreased by increasing the FDL granularity. However, if FDL granularity is larger than a single slot, unused void slots may occur in FDLs. FDL sets with granularity larger than single slot can provide only a limited set of required delays. For example, FDL set with granularity of 2 slots can delay the packets by 2, 4, 6, 8... slots, but cannot delay the packets by 1, 3, 5, 7... slots. When the required delay is not supported by the FDL set, FDL set may delay a packet more than the required delay. In this case, extra delaying the packets causes unused void slots in FDLs and output. Using a void-filling scheduling algorithm can fill in some of the void slots. However, void-filling algorithms increase the scheduler complexity. Furthermore, void filling algorithms may cause packet reordering, so they must be carefully applied. We use a void filling algorithm that prevents packet reordering among input-output ports of the switch. The algorithm keeps a list of the departure time of the last buffered packets destined from each input port to each output port. Moreover, it keeps a list of the voids in the buffer reservation table of each output port. When there is a packet to be buffered, in order to preventing reordering we find the first void in the buffer reservation table that starts after the departure of the last buffered packet using the same input output pair. Then starting from this void we search for the first void that the packet fits into by using the FDL set. If there is no suitable void, packet is scheduled after the last packet in the buffer. We show and compare the simulation results of both with and without void-filling scheduling.

Voids are shown as an example in Fig. 2.3. FDL granularity is selected as 4 slots. $m+1^{st}$ packet contends with m^{th} packet, so $m+1^{st}$ packet must be delayed by 5 slots for solving the contention. However, FDL set with granularity of 4 slots can delay the packets by only 4, 8, 12... slots, so the packet is delayed by 8 slots instead of 5 slots. Therefore, 3 void slots occur between m^{th} and $m+1^{st}$ packets. Furthermore, size of these two packets is not equal to a multiple of slot size, so voids occur inside their last slots.

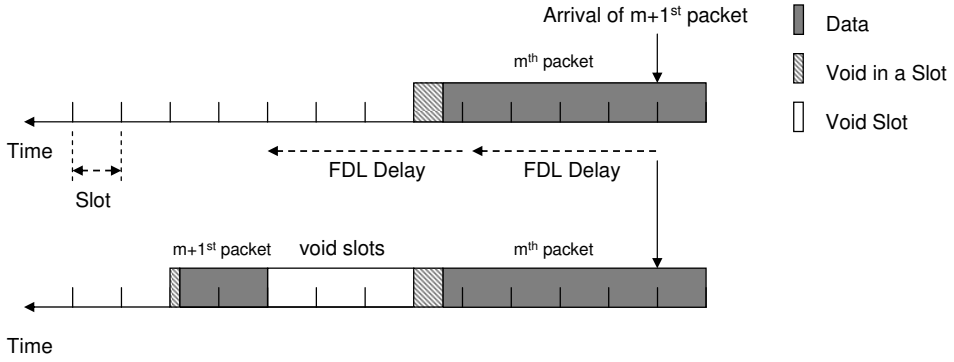


Figure 2.3: Voids.

2.3 XCP Control

2.3.1 XCP Basics

XCP is a new congestion control algorithm specifically designed for high-bandwidth and large-delay networks. XCP makes use of explicit feedbacks received from the network. It decouples the utilization control from fairness control. Core routers are not required to maintain per-flow state information.

Working principle of XCP is as follows: XCP core routers maintain a per-link control-decision timer. When a timeout occurs, Core routers update their link control parameters calculated by Efficiency Controller and Fairness Controller according to link utilization, spare bandwidth and buffer occupancy. XCP sender agent sends its traffic rate to XCP receiver in the header of data packets or a probe packet. On the way to the XCP receiver, XCP core routers read this information and calculate a feedback that shows the desired traffic rate change for this XCP flow and updates the feedback header of the packet. XCP receiver simply sends back the final feedback to the XCP sender. XCP sender updates its sending rate according to this feedback.

Efficiency Controller (EC)

Efficiency Controller is responsible for maximizing link utilization by controlling aggregate traffic. Every router calculates a desired increase or decrease in aggregate traffic for each output port by using the equation $\Phi = \alpha \cdot S - \beta \cdot Q/d$. In this equation, Φ is the total amount of desired change in input traffic. α and β are spare bandwidth control parameter and queue control parameter respectively and d is the control decision interval. S is the spare bandwidth that is the difference between the link capacity and input traffic in the last control interval. Q is the persistent queue size.

Fairness Controller (FC)

After calculating the aggregate feedback Φ, FC is responsible for fairly distributing this feedback to flows. FC uses an AIMD-based control for distributing the feedback. It means that when Φ is positive, fairness controller increases the transmission rate of all flows by the same amount. When Φ is negative, fairness controller decreases the transmission rate of each flow proportional to flow's current transmission rate. However, when Φ is small, convergence to fairness may take a long time. Furthermore, if Φ is zero, XCP stops converging. In order to prevent this problem, bandwidth shuffling, which redistributes a small amount of traffic among flows, is used. This shuffled traffic is calculated by $h = max(0, \gamma \cdot u - |\Phi|)$, where γ is the shuffling parameter and u is the aggregate input traffic rate in the last control interval.

When a router receives a packet containing feedback, it calculates and compares its own feedback with the feedback available in the packet header. If its own feedback is smaller than the one in the header, it updates the feedback in the header with its own feedback. Otherwise, it does not change the feedback available in the header.

When a XCP source agent receives an ACK containing XCP feedback, it updates its congestion window size according to the formula $cwnd = max(cwnd + H\_feedback, s)$, where s is the packet size and $H\_feedback$ is the feedback in the ACK packet.

2.3.2 XCP Variants

XCP was first proposed in Ref. [16] as a window-based reliable congestion and transmission control protocol. The same paper also proposes a XCP-based Core Stateless Fair Queuing as a gradual deployment method. XCP-based Core Stateless Fair Queuing algorithm creates a XCP macro flow, assigns the TCP and UDP flows to the XCP macro flow. The edge nodes forward the TCP and UDP packets inside the XCP macro flow according to the XCP macro flow rate. [16] states that the algorithm can be further simplified by using special probe packets for receiving the feedbacks for macro flow rate calculation instead of attaching congestion header to forwarded packets.

TeXCP [17] is a traffic engineering protocol using a rate-based XCP congestion control allowing traffic engineering by applying load balancing with multi-path routing. TeXCP creates a macro flow and assigns TCP and UDP flows to this macro flow and forwards the packets according to XCP flow rate similar to simplified XCP-based Core Stateless Fair Queuing.

2.3.3 Rate-based Paced XCP

We propose Rate-based Paced XCP as an intra-domain traffic shaping and congestion control protocol in an OPS network domain. In this architecture, XCP sender agent on an edge node multiplexes incoming flows and creates a macro flow like a LSP as shown in Fig. 2.1, and applies leaky bucket pacing according to rate control to the macro flow and sends to a receiver XCP agent on destination edge node. The receiver XCP agent de-multiplexes the macro flow and forwards the packets of individual flows to their destinations.

In the original XCP [16], feedbacks are carried in the header of data packets (per-packet feedback). In this case, core routers must read and update the feedback in the packet header by calculating a new feedback. However, calculating a new feedback for and updating the header of each optical packet at ultra high speed is hard. In our proposal, simplified XCP-based Core Stateless Fair Queuing [16] and TeXCP [17], each macro flow sends its feedback in a separate probe packet once in every control period, instead of writing feedback to packet

headers, so there is no need for calculating a per-packet feedback. Probe packets are carried on a separate single control wavelength, which means that we are separating the control channel and data channels. Using a separate single wavelength with low transmission rate for probe packets allows applying electronic conversion for updating feedback in packet headers and buffering the probe packets in electronic RAM in case of a contention.

Core routers use a separate XCP control agent for each wavelength on an output link. When a probe packet of macro flow i arrives to a core router, FC of the XCP agent responsible for the wavelength that macro flow i was assigned to calculates a positive feedback p_i and a negative feedback n_i for flow i. Positive feedback is calculated by $p_i = \frac{h+max(0,\Phi)}{N}$ and negative feedback is calculated by $n_i = \frac{u_i \cdot (h+max(0,-\Phi))}{u}$, where N is the number of macro flows on this wavelength, u_i is the traffic rate of flow i estimated and sent by the XCP sender in the probe packet and h is the shuffled bandwidth. $feedback = p_i - n_i$ gives the required change in the flow rate as a feedback. When a core router receives a probe packet, router calculates and compares its own feedback with the feedback available in the probe packet. If core router's own feedback is smaller than the one in the probe packet, core router replaces the feedback in the probe packet with its own feedback. Otherwise, core router does not change the feedback. Core routers can estimate the number N by counting the number of probe packets received in the last control interval or use the number of LSPs if GMPLS is available [17]. In [16], the control interval is calculated as the average RTT of flows using the link. In TeXCP and our architecture, control interval is the maximum RTT in the network. TeXCP uses a simplified version of XCP's fairness controller algorithm without the bandwidth shuffling algorithm of XCP, but our algorithm uses the bandwidth shuffling algorithm of XCP. In TeXCP, core routers send both p_i and n_i feedback by probe packets to sender agents, but in our algorithm core routers send only $feedback = p_i - n_i$ like in [16].

As explained in Sec. 2.1.1, Φ is calculated for a wavelength by using the equation $\Phi = \alpha \cdot S - \beta \cdot Q/d$ where S is the spare bandwidth that is the difference between the wavelength capacity and input traffic on this wavelength in the last control interval. Therefore, wavelength capacity must be explicitly given to XCP algorithm for calculating S. Giving

a false capacity value less than actual wavelength capacity causes under-utilization. XCP algorithm converges to the given virtual capacity. We use this property of XCP to limit utilization of OPS network at a level that provides low packet drop ratio with the available FDL set.

2.4 Evaluation

2.4.1 Simulation Settings

Proposed protocol and slotted WDM OPS architecture is implemented over *ns* version 2.28 [28]. It is assumed that there is a backlogged traffic at edge buffers, so each edge node sends traffic to all other edge nodes at the maximum possible rate controlled by XCP. We chose XCP's α, β and γ parameters for edge routers as 0.2, 0.056 and 0.05, respectively. α parameter controls the utilization convergence speed. Higher α value allows faster change in utilization, but also causes higher utilization overshoots. We chose the α parameter as 0.2, which gives a slower but more stable link utilization and decreases utilization overshoots when compared with the value 0.4 selected in [16]. When α parameter is decreased, it is also necessary to decrease γ parameter responsible for bandwidth shuffling. Otherwise, too much under-utilization may occur in some links in case these links carry flows that are bandwidth throttled in other bottleneck links as explained in [16]. Therefore, γ=0.05 is used instead of γ=0.1 in [16]. FDL architecture makes it hard to evaluate and provide a buffer occupancy value to XCP algorithm. Furthermore, our aim is to have a small buffered network and effect of queue parameter in XCP calculations is low as persistent queue size is small (usually zero unless overload) due to a small buffered network, so β parameter is set to zero in the core routers. These may not be the optimum values, but optimization of XCP parameters is left as a future work.

Simulator uses cut-through packet switching for data wavelengths. There is a single slow control wavelength dedicated for probe packets. Control wavelength uses store-and-forward switching. XCP agents start sending data randomly in the first 10s and continue until the simulation ends. Total simulation duration is 40s.

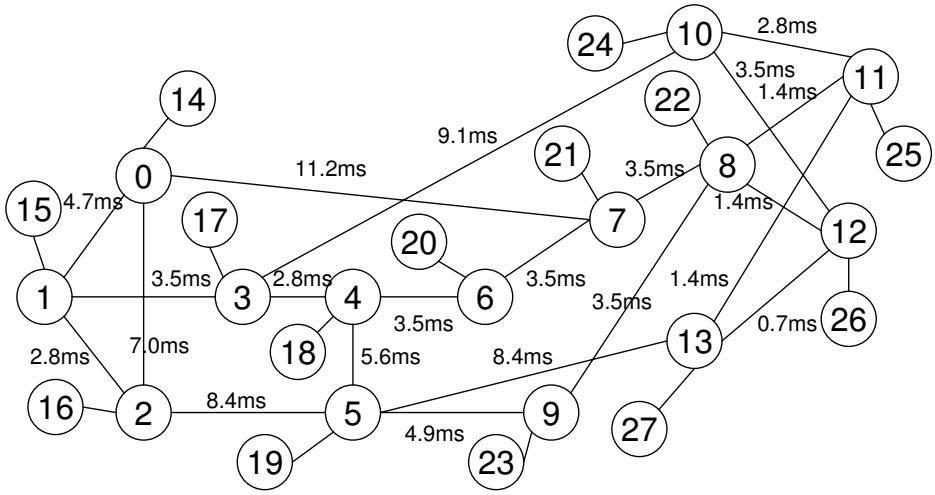


Figure 2.4: NSFNET topology.

Edge nodes use electronic buffering, but core routers use only FDLs for buffering optical data packets. Contention of probe packets on control wavelength is resolved by electronic RAM. O/E/O conversion is not a problem for control wavelength due to its low speed.

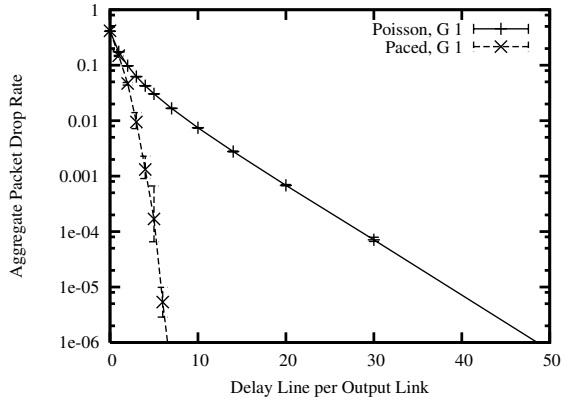
Fig. 2.4 shows the simulated NSFNET topology. The nodes numbered from 0 to 13 are the core nodes and the rest are the edge nodes connected to the core nodes. All links (including edge and core links) have a single data wavelength with the same capacity. All links have the same XCP target utilization. There are a total of 28 nodes (14 core nodes+14 edge nodes) and 35 links (21 core links+14 edge links). The propagation delay of links between core and edge nodes is selected as 0.1ms. All links (including edge and core links) apply optical packet switching. Each edge node sends traffic to all other edge nodes at the maximum possible rate, so there are multiple bottleneck links. XCP control period of core routers and probe packet sending interval of edge routers is selected as 50ms by considering extra processing and queuing delays in the core routers. Target utilization is set to 90% for output links of edge nodes in order to prevent buffer buildups in the buffer between the link and paced XCP sources. The capacity of the data wavelength is set to 1Gbps. The capacity of the XCP control wavelength is 100Mbps.

In all simulations, we evaluate the aggregate packet drop rate inside the OPS core network. We compare the packet drop rate of the proposed pacing architecture with the packet drop rate of poisson traffic arrival. For the traffic matrix of poisson traffic, we use the traffic matrix that proposed XCP based architecture converges to when there is a low packet drop rate with enough buffering and FDL granularity of 1. TCP traffic is well known to be burstier than poisson distribution and aggregation of many TCP flows does not converge to poisson stream [29]. Burstiness increases the buffer requirements. [5] decreases the buffer requirements by making the traffic poisson-like by modifying TCP senders to used Paced TCP or by using slow access links. In this chapter, we show that our proposed architecture can achieve packet drop rates lower than poisson traffic.

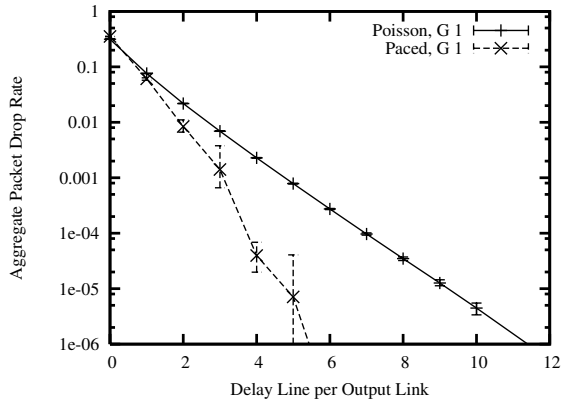
2.4.2 Evaluations

Slot Size is Equal to MTU

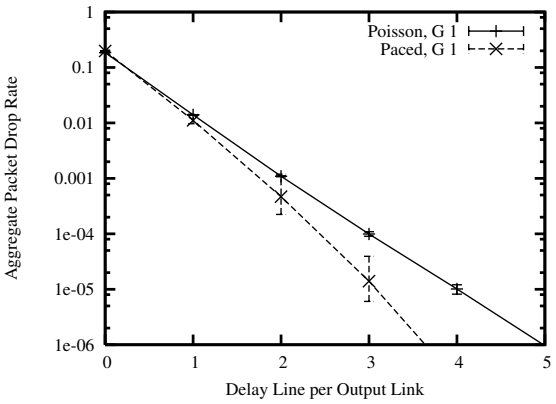
We first compare the performance of proposed architecture and poisson traffic when slot size is equal to MTU, which is selected as 1500Bytes in the simulations. It means each packet is carried inside only a single slot. Fig. 2.5 shows the aggregate packet drop rate of proposed architecture and poisson traffic (called Paced and Poisson in the figures, respectively) at 90%, 60% and 30% target link utilization, which is the utilization due to optical packets including data payload and wasted void padding inside them. x-axis shows the limit of the number of delay lines per output in linear scale and y-axis is the aggregate packet drop rate in the core in log scale. Average, minimum and maximum drop rate results of 10 simulations, which have different flow starting times, are plotted. We see that deviation increases as the drop rate decreases, but the overall tendency is the same. Therefore, in the following figures a single simulation is done for each parameter set. FDL granularity is 1 slots in all plots. Higher granularities do not bring significant improvement, so they are not plotted. As seen in Fig. 2.5, improvement of proposed architecture increases as the link utilization increases. We see that buffer requirement of proposed paced architecture at low packet drop rate like 10^{-6} is around 8 times lower than poisson traffic arrival. Even at 30%



(a)



(b)



(c)

Figure 2.5: Aggregate packet drop rate with limited number of FDLs per link when slot size is 1500Bytes for (a) 90%, (b) 60%, and (c) 30% utilization.

utilization, we see that proposed architecture can get much lower packet drop rates with the same buffering even though poisson traffic has low buffer requirements at this utilization. For example, when there are 3 delay lines per output link, packet drop rate of the proposed architecture is around 10 times lower. The improvement becomes bigger as the number of delay lines increases. This is an expected result as Ref. [30] theoretically shows that multiplexed periodic streams like the paced traffic in our architecture give lower packet drop rate than poisson traffic as they are less burstier [29] than poisson traffic. Furthermore, Ref. [30] shows that drop rate comes closer to poisson as the number of periodic streams increase. Therefore, we can expect to have a lower drop rate by applying pacing to macro flows at the edge nodes as in our architecture than individually pacing TCP flows. When there is no buffering, both paced and poisson traffic has the same packet drop rate as seen in Fig. 2.5. This is an expected result, because packet drop rate depends only on the packet contention probability that is independent of burstiness.

When slot size is equal to MTU, effective throughput may be low when the average size of arriving packets is small. On the other hand, it can greatly simplify the FDL design process, as FDL granularity can be directly selected as 1 slot size independent of packet size distribution as higher granularities do not bring significant improvement and scheduling algorithm becomes simpler.

Slot Size is 500Bytes

When slot size is lower than MTU, packet size distribution must be taken into account as big packets may use multiple slots. In this section we compare the performance of proposed architecture and poisson traffic when the slot size is tentatively selected as 500Bytes that is one third of the MTU.

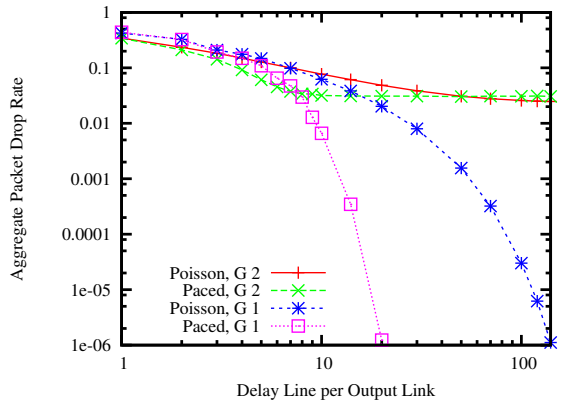
It is hard to do a direct comparison of different slot sizes, because guard bands and slot headers start becoming the main source of decrease in efficiency as we decrease the slot size or increase the link speed. Size of guard bands depends on the type of switching hardware and link speed.

[31] shows that size of packets in Internet2 traffic is mainly composed of very small and

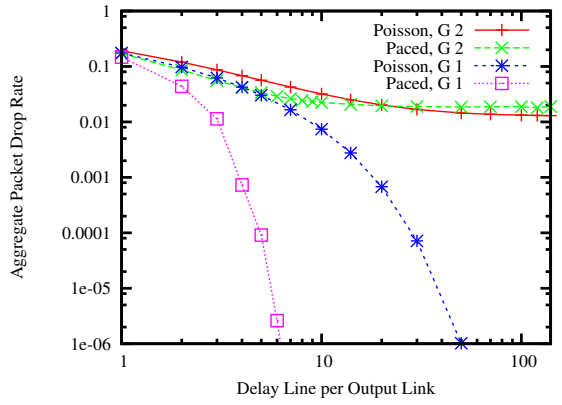
big packets and there is around 3:2 ratio between these two, so this packet size distribution is used in the simulations as a realistic packet size distribution. Simulated packet size distributions are

- All packets are less than or equal to 1 slot (500 Bytes) size
- All packets are 3 slots (1500 Bytes) size
- 60% of packets are less than or equal to 1 slot (500 Bytes) size, 40% of packets are 3 slots (1500 Bytes) size (realistic traffic)

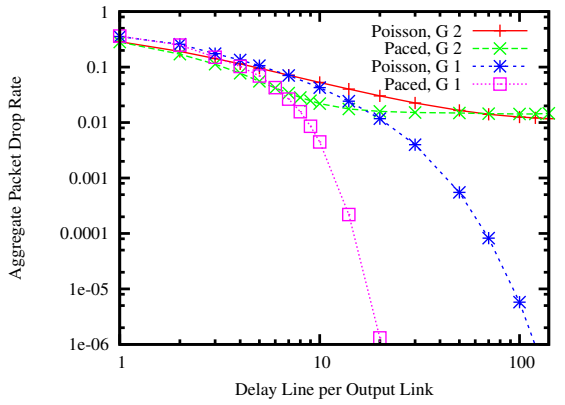
Fig. 2.6 shows the aggregate packet drop rate for different packet size distributions, FDL granularities for proposed architecture and poisson traffic when target utilization is 90%. In all subplots, x-axis shows the limit of the number of delay lines per output and y-axis is the aggregate packet drop rate in the core, both in log scale. G lines in the figure show the applied FDL set granularity. Fig. 2.6(a,b,c) shows the drop rates when packet size distribution is all 1500Bytes, all less than or equal to 500Bytes and realistic distribution, respectively. Note that effective utilization and throughput is different in simulation of different packet size distributions at the same target link utilization due to padding inside the optical packets. When we check these subplots, we see that packet size distribution has a big impact on the FDL requirements. In all subplots, FDL granularity of 1 slot gives a fast decrease in drop rate, but FDL granularity of 2 slot tends to stay constant after a decrease at the beginning because of the the void slots in FDLs due to high granularity. Voids increase the effective load to higher than link capacity, so low packet drop rate can not be achieved with at 90% utilization with small buffers at high FDL granularity. In all subplots, we see that buffer requirement of proposed paced architecture at a low packet drop rate like 10^{-6} is around 8 times lower than poisson traffic arrival like in Fig. 2.5(a). Actually Fig. 2.6(b) is almost the same as Fig. 2.5(a), because Fig. 2.6(b) is merely a case where slot size and packet size distribution are down-scaled to one third of Fig. 2.5(a). Realistic packet size distribution in Fig. 2.6(a) shows that mainly big packets determine the in the buffer requirements, so its buffer requirements are almost the same as the case



(a)



(b)



(c)

Figure 2.6: Aggregate packet drop rate with limited number of FDLs per link for 90% utilization when packet size distribution is (a) all packets are 1500Bytes, (b) all packets are less than or equal to 500Bytes and (c) realistic distribution.

where all packets are 1500Bytes size in Fig. 2.6(c) and around 3 times higher than the case all packets are less than or equal to 1 slot (500 Bytes) size in Fig. 2.6(b).

Fig. 2.7 shows the aggregate packet drop rate when target utilization is 30%. Fig. 2.7(a,c,e) and Fig. 2.7(b,d,f) show the drop rates with the proposed pacing architecture and poisson traffic, respectively. When we check these subplots, we see that packet size distribution has a big impact on the optimum FDL granularity. Fig. 2.7(a,e) shows that proposed architecture has optimum FDL granularity of 7 slots for realistic packet size distribution and when all packets are 1500Bytes. On the other hand, when packet size distribution is all packets are less than or equal to 500Bytes, the optimum granularities are 1, 2 and 3 slots as seen in Fig. 2.7(c). Granularity of 7 slots have much higher packet drop rate these granularities when there are multiple delay lines per output link. In general, small FDL granularities tend to give a sharp decrease in drop rate as number of delay lines increase, but high FDL granularities tend to stay almost constant after with a decrease at the beginning in the proposed architecture. This constant drop rate in Fig. 2.7(a,c) is mainly because of load overshoot due to the void slots in FDLs because of high granularity. When we compare these high granularities for paced and poisson traffic, we see that paced architecture gives higher drop rates mainly because of the synchronized packet contentions due to pacing. Otherwise, paced architecture have lower packet drop rates at the same FDL granularities unless granularity is high. Realistic packet size distribution of paced architecture at granularity of 12 slots in Fig. 2.7(e) shows that contention synchronization is lower when there is a mixed packet size distribution using different number of slots.

Void slots in FDLs are served by the routers as if they are not empty, so void slots increase the effective load. When there are void slots in FDLs, using the size of slots occupied by the packets as a metric does not give a reliable measure of congestion. It can be possible to guarantee preventing overload by carefully selecting the target utilization. In the worst case, all packets entering an FDL occupy minimum number of slots and synchronized packet arrivals cause the maximum number of possible void slots, which is denoted by V, inside the FDL. Therefore, in a single stage equidistant FDL set, lowest efficiency in the worst case is approximately,

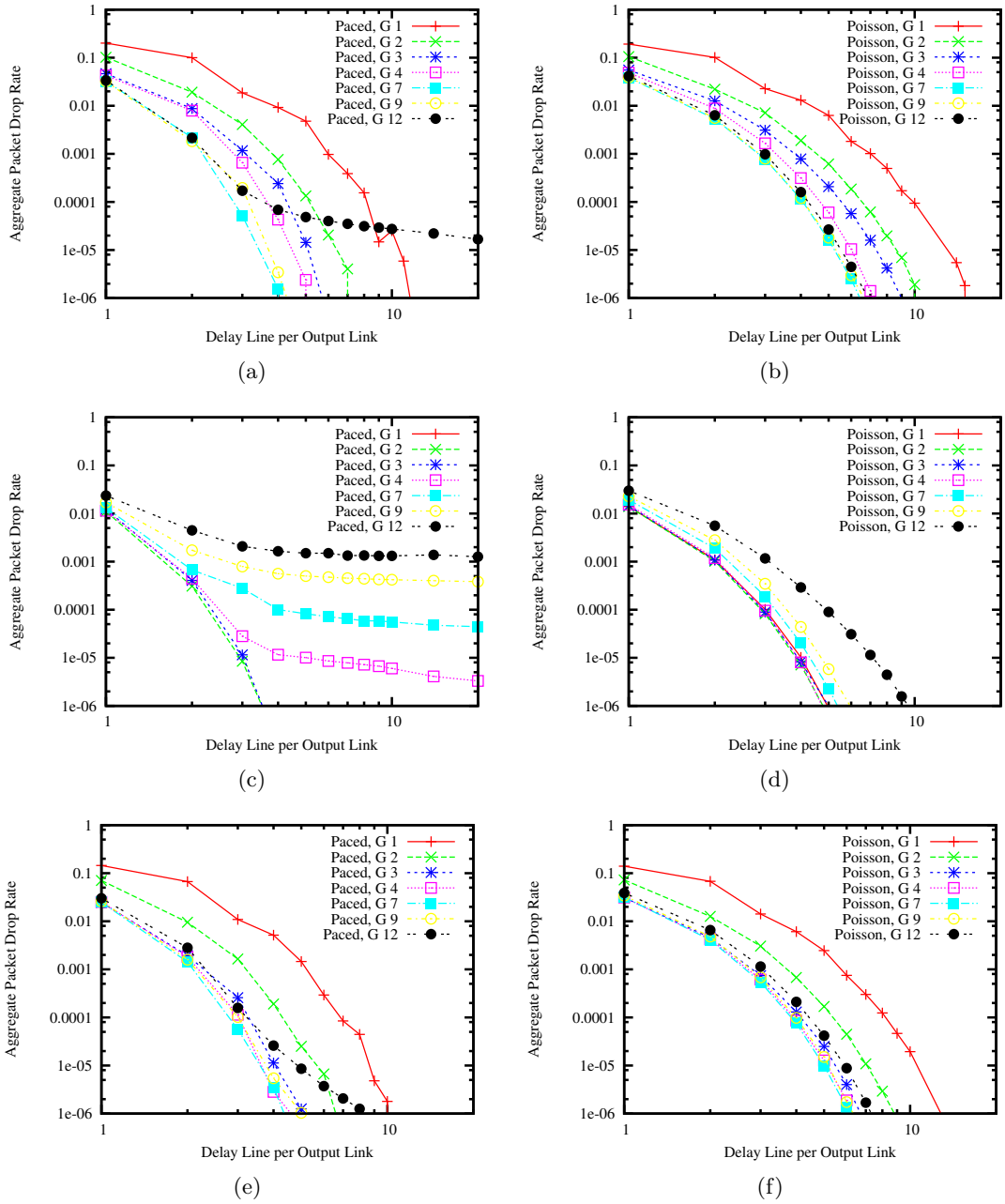


Figure 2.7: Aggregate packet drop rate with limited number of FDLs per link for 30% utilization when packet size distribution is all packets are 1500Bytes with (a) paced and (b) poisson traffic, all packets are less than or equal to 500Bytes with (c) paced and (d) poisson traffic, realistic distribution with (e) paced and (f) poisson traffic.

$$\frac{M}{M+V}, \tag{2.1}$$

where M is the number of slots occupied by the smallest possible packet size and V is the maximum number of void slots may occur upon arrival of a packet. V equals to $x-1$ slots, where x denotes the FDL granularity, as there must be an occupied slot using the link and causing contention. Plugging $V = x - 1$ gives

$$\frac{M}{M+x-1}, \tag{2.2}$$

Setting the target utilization in optical XCP routers to a value smaller than the result of this equation can protect output wavelengths from load overshoots and protects from drops due to void slots. It is better to apply a safety margin for possible rate oscillations and use a target utilization a little lower than the value calculated by using the equation above. When we use the formula to the case of all packets are 1 slot size and 3 slot size for 30% utilization, we get the limit FDL granularities 3 slots and 7 slots, respectively. When we check the simulation results in Fig. 2.7(a,b), we see that all simulated FDL granularities bigger than these granularities have higher drop rate than these granularities.

Scheduling

Using a void-filling scheduling algorithm can greatly decrease the packet drops due to void slots. In other to show the effect of void filling, NSFNET is simulated with realistic packet size distribution at 60% target utilization and 8 delay lines per output link. Slot size is 500Bytes. Fig. 2.8 shows the packet drop rates of proposed architecture and poisson traffic with and without void-filling scheduling algorithm with packet reordering prevention is applied. X-axis shows the FDL granularity in linear scale and terms of slots and y-axis is the aggregate packet drop rate in the core in log scale. We see that when granularity is 1, void filling algorithm give the same performance as non-void filling algorithm, as expected because there are no void slots. As we increase the granularity, we see that void filling algorithms starts giving much lower drop rates. Moreover, proposed architecture gives much

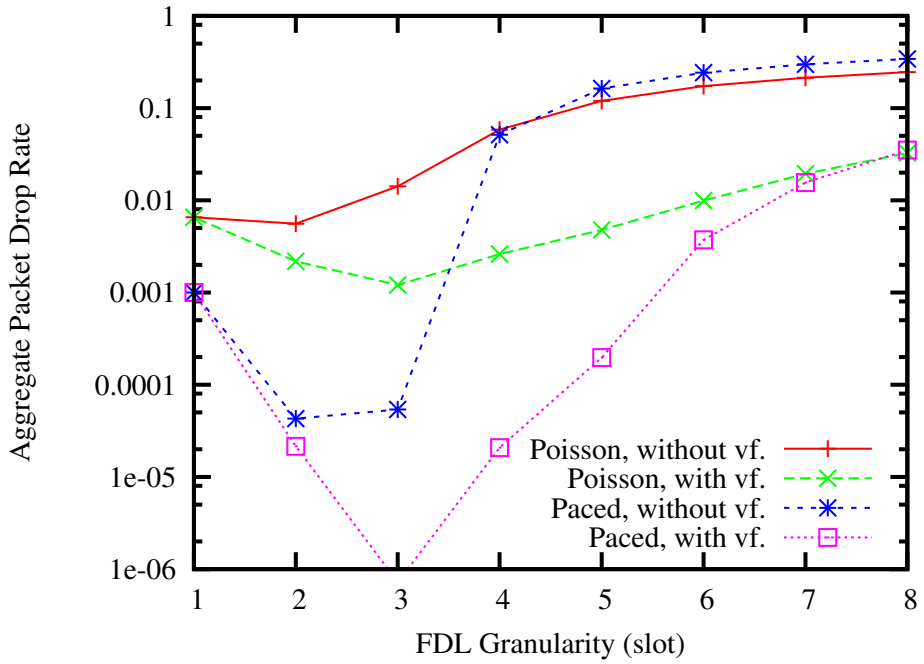


Figure 2.8: Effect of FDL granularity and void filling scheduling on aggregate packet drop rate.

lower drop rates than poisson traffic with the same scheduling. After an optimum granularity, drop rates start increasing due to void slots. As we further increase the granularity, proposed architecture and poisson traffic starts giving similar packet drop rate, because load overshoot due to void slots becomes the main reason of packet drops. Optimum granularity is 2 slots for non-void filling algorithms, while it is 3 slots for void-filling scheduling. In general, we see that applying void-filling scheduling and using the proposed architecture together can greatly decrease the drop rate.

2.5 Conclusions

In this chapter, we proposed an architecture designed for OPS WDM networks with pacing at edge nodes for minimizing the buffer requirements. We evaluated the packet drop rates with extensive simulations on a meshed network with multiple-hop paths and showed how

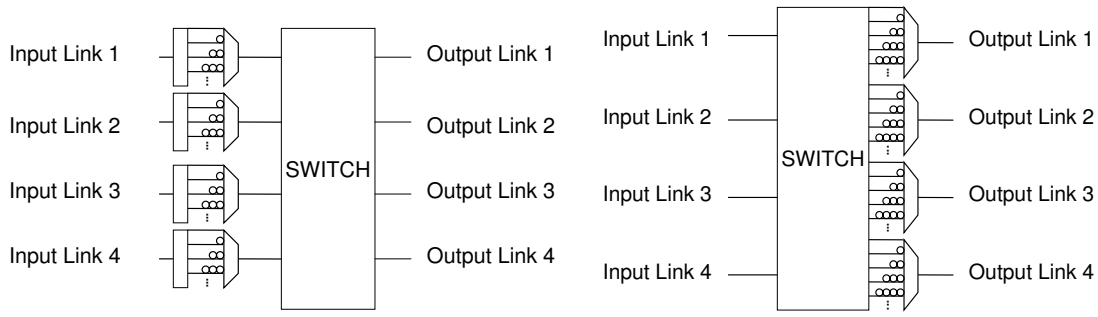
FDL requirements change with slot size, FDL granularity, scheduling and packet size distribution. Simulation results with meshed networks showed that our architecture can provide much low packet loss ratio lower than poisson traffic in core OPS networks with small FDL buffers. We showed that large packets and small packets have different FDL requirements. Small packets require low granularity for low packet drop rate, but large packets require high granularity for decreasing the number of required FDL lines. Therefore, selection of slot size and MTU of the architecture has a strong impact on the buffer requirements. Selecting a big slot size equal to MTU may decrease the efficiency due to padding for small packets, but it has low FDL requirements and it can greatly simplify the design process and allows much simpler scheduling. In case of using a slot size smaller than MTU, we showed that void-filling scheduling can greatly decrease the buffer requirements.

Chapter 3

Switch Architectures for Small-buffered Optical Packet Switched Networks

3.1 Introduction

In Chapter 2, we introduced an all-optical OPS network architecture that can achieve high utilization and low packet drop ratio by using FDL-based small buffering. Switching fabric size is an important cost factor in routers, so it is necessary to design a suitable switch architecture for the proposed small-buffered architecture. Many switching fabric architectures like MEMS, optomechanical, electrooptic, thermooptic, liquid-crystal based switches are proposed for optical switching [19]. However, the number of switching elements in the fabric increases together with the overall cost as the number of ports of the switch increases. Also, increasing the switch size introduces high crosstalk and insertion losses in many proposed switching fabric architectures. These losses require optical amplification that further increases the overall cost as explained in [19]. In Chapter 2, a simple output FDL buffered switch was used as switching architecture. In this chapter, we investigate and compare input and output buffered optical switch architectures for minimizing the size of



(a) Input Buffering (b) Output Buffering

Figure 3.1: Switch and FDL architectures

optical switching fabric of core nodes while achieving higher throughput with small buffers. For this purpose we apply the proposed FDL-based small-buffered network architecture. We show how the FDL requirements of different switch architectures change with FDL granularity and packet size distribution by using a star topology.

3.2 Switch, Scheduler and FDL Architectures

In Chapter 2, we evaluated the FDL requirements of an output buffered switching architecture where FDL lines are connected to the output ports of the switch as shown in Fig. 3.1(b) for a single wavelength. If there are many fiber delay lines per output link, such a switch requires many output ports and therefore a big switching fabric However, switching fabric size is usually one of the biggest factors determining overall router cost, so in this chapter we try to decrease the size of the switching fabric. In Chapter 2, output buffering without void filling was used as the buffering architecture and scheduling algorithm. In this chapter we evaluate the switch size and buffer requirements of a different architecture called input buffering with virtual output queuing (VOQ) scheduling shown in Fig. 3.1(a) for a single wavelength. We also evaluate the buffer requirements of void filling scheduling version of output buffering for comparision. Speedup is 1 in all switches.

Buffers are implemented as a single stage equidistant fiber delay lines like in Chapter 2. FDL length distribution increases linearly (x, $2x$, $3x$, $4x\ldots$) where x is FDL granularity.

The number of required FDLs (denoted by B) is evaluated for different FDL granularities. When output buffering is used, required switch size for a single wavelength is $N \times BN$, where N is the number of links assuming the number of output and input links is the same, as seen in Fig. 3.1(a). On the other hand, input buffering decreases the main switch size to $N \times N$ independent of the number of delay lines. Each input link requires a $1 \times N$ small switch in front of its FDL set. Therefore, input buffering can be implemented by dividing the switching fabric into smaller switches instead of a single and large main switch. This may bring a drastic decrease in switching fabric cost especially if B is high. However, a well-known problem of input buffering is head-of-line blocking, which limits the achievable utilization. We apply virtual output queuing (VOQ) scheduling for minimizing this problem.

A FDL set provides only a limited set of required delays, unless granularity is a single slot. When the required delay is not supported by the FDL set, packets may end up to be delayed more than the required delay. Extra delaying the packets causes unused void slots, which decrease the achievable throughput of output links. Void filling scheduling algorithms decrease the number of such unused slots and decrease the FDL requirements. However, void-filling algorithms increase the scheduler complexity, so a simple output buffering architecture without in output buffering was used in Chapter 2. In this chapter, we evaluate a void filling version of output buffering architecture for a more fair comparison with input buffering architecture where void filling is necessary for VOQ. Void filling algorithms may cause packet reordering, so they must be carefully applied. We prevent packet reordering among the packets that will be switched through the same input-output link pairs in both input buffering and void filling version of output buffering.

3.3 Evaluation

3.3.1 Simulation Settings

Proposed network architecture and buffering models are implemented over *ns* version 2.28 [28]. XCP agents start sending data randomly in the first 10s and continue until the

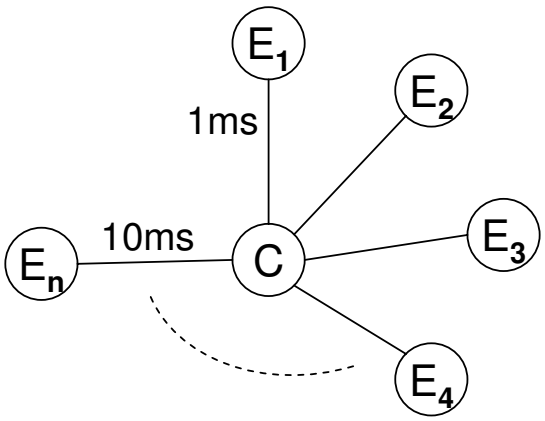


Figure 3.2: Star topology

simulation ends. It is assumed that there is a backlogged traffic at edge buffers, so each edge node sends traffic to all other edge nodes at the maximum possible rate controlled by XCP. We chose XCP's α, β and γ parameters for edge routers as 0.2, 0.056 and 0.05, respectively, as explained and used in Chapter 2. However, input buffering architecture implemented by FDLs makes it hard to provide buffer occupancy data to XCP algorithm. Furthermore, our aim is to have a small buffered network and effect of queue parameter is low as persistent queue size is usually small with such a small buffered network, so β parameter is set to zero in the core routers. Ref. [32] shows that this parameter set is stable. Total simulation duration is 40s.

Slot size is selected as 52Bytes, because Ref. [31] shows that most common small packets on Internet2 are in the range of 40Bytes to 52Bytes. The selection of optimum slot size is left as a future work. Probe packet size is selected as equal to the slot size. FDLs are used for resolving contention of data packets. Contention of probe packets on control wavelength is resolved by electronic RAM. Ref. [31] shows that size of packets in Internet2 traffic is mainly composed of very small and big packets and there is around 3:2 ratio between these two, so this packet size distribution is used in the simulations as a realistic packet size distribution. Simulated packet size distributions are

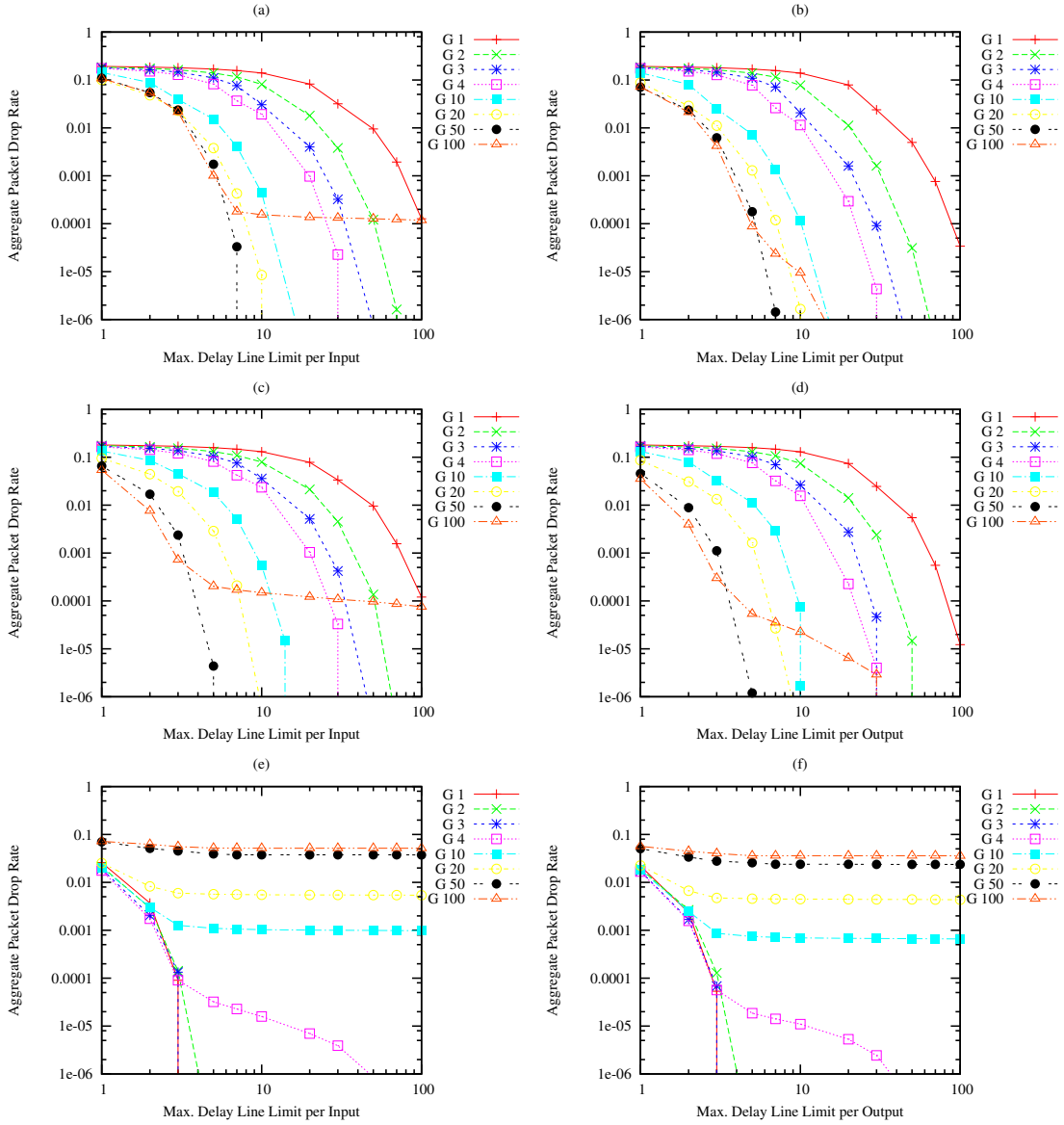


Figure 3.3: Aggregate packet drop rate with limited number of FDLs per link when packet size distribution is realistic packet size distribution with input buffering (a) and output buffering (b), all 1508Bytes with input buffering (c) and output buffering (d), all 52Bytes with input buffering (e) and output buffering (f)

- All packets are 1 slot (52 Bytes) size
- All packets are 29 slots (1508 Bytes) size
- 60% of packets are 1 slot (52 Bytes) size , 40% of packets are 29 slots (1508 Bytes) size (realistic traffic)

The star topology shown in Fig. 3.2 is used for computer simulations. There is a single core node for switching the packets. Star topology is simulated when there are 12 edge nodes in the network. Each source node sends data to all other edge nodes, so each link carries 11 macro flows (LSPs) in each direction. Simulated FDL granularities range between 1 to 100 slots. Target utilization parameter of XCP is set to 30% for output links of core node as Ref. [4] states that network operators usually run backbone links at loads of 10%-30%. Target utilization is set to 90% for output links of edge nodes as they can use electronic RAM for buffering.

There is a single data wavelength on links. Propagation delay of links range between 1ms and 10ms in the network. XCP control period of core routers and probe packet sending interval of edge routers is 40ms. The capacity of the data wavelength is set to 1Gbps when packets are 29 slots size and realistic packet size distribution. When all packets are 1 slots size, wavelength capacity is set to 100Mbps due to simulation time constraints. The capacity of the XCP control wavelength is 100Mbps.

Figure 3.3 shows the aggregate packet drop rate in the simulations. In all subplots, y-axis shows the limit of the number of delay lines per link and x-axis is the aggregate packet drop rate in the core, both in log scale. G lines in the figure show the applied FDL set granularity. When we compare the simulation results of input buffering and output buffering in Fig. 3.3, we see that the delay line requirements for the same packet drop rate are close, especially for the high granularities when packets are big and all granularities when all packets are 52Bytes. FDL requirements of input buffering is a bit higher.

In the graph, we see that granularities between 1-50 slots in big size packet simulations and granularities between 1-3 slots in simulation of 52Bytes packets show a sharp decrease in

drop rate as the number of delay lines increase. However, if we increase the granularity, the drop rate decreases first and then becomes almost constant or decreases with a lower rate, because void slots in FDLs and output due to high FDL granularity causes synchronized packet drops and limits the achievable utilization as explained in Chapter 2.

Packet size is distribution in the network is an important factor on the selection FDL granularity and achievable utilization. If we want a network to have very low packet drop rate, it is necessary to select the FDL granularity according to the worst case scenario that is the case of all packets in the network have the minimum possible size, which is taken as 1 slot size in the simulations this chapter. After selecting an FDL granularity that can achieve the target utilization and required packet drop ratio with small packets, the number of delay lines of the switch can be evaluated and selected according to the FDL requirements in simulation of a traffic composed of big packets and required packet drop ratio. For example, Fig. 3.3(e,f) show that FDL granularities of 1, 2 and 3 slots can achieve very low drop rate when all packets are 1 slot size. When we check the FDL requirements of these granularities with simulation of big packets in Fig. 3.3(a,b,c,d), we see that it is possible to get low packet drop rate with around 30-40 delay lines per link with granularity of 3 slots. On the other hand, around 1% drop rate may be enough for internet traffic. In such a case, using as low as 4-5 delay lines per link with granularity of 20 slots looks enough for all simulated packet size distributions.

3.4 Conclusions

In this chapter, we investigated some optical switch architectures for minimizing the size of optical switching fabric with the proposed network architecture based on pacing the traffic. We compared the buffering architectures input buffering with VOQ, and output buffering with void filling. We evaluated the packet drop rates depending on FDL granularity and packet size distribution.

We showed that input buffering requires comparable number of delay lines as output buffering architectures at 30% utilization, which is typical for backbone links of network

operators, with pacing. Input buffering can be implemented by dividing the switching fabric into smaller switches instead of a single and large main switch, so input switching may decrease costs when the cost of a large and single switch is higher. The drawback of input buffering is that its scheduling algorithm is more complex than scheduler of output buffering, but processing power requirements of input buffering may be decreased with some optimizations.

Chapter 4

Pacing for Optical Packet Switched Networks with Very Small Optical RAM

4.1 Introduction

FDLs have important limitations. First of all, FDLs require very long fiber lines that cause signal attenuation, inside the routers. There can be a limited number of FDLs in a router due to space considerations, so they can provide a small amount of buffering. Second, FDLs provide only a fixed amount of delay. FDL buffering is possible with today's technology, so many researchers consider FDL buffers to resolve contentions in optical networks. On the other hand, optical RAM is under research and it may be available in the near future. Optical RAM solves the problems of FDLs like lack of $O(1)$ reading, signal attenuation and bulkiness. However, Optical RAM is not expected to have a large capacity, soon.

TCP is well-known to behave bursty [33]. Bursty nature of TCP limits decreasing the buffer requirements, because it brings a high packet drop rate in small buffered networks. One possible solution for solving this problem is TCP Pacing. Pacing is defined as transmitting ACK (data) packets according to a special criteria, instead of transmitting

immediately upon arrival of a data (ACK) packet [6]. Pacing is initially proposed as a solution for ACK-compression [6]. Kulik et al, proposed using paced TCP for solving the problem of queuing bottlenecks by preventing bursty behavior of TCP [34]. In Ref. [35], results of extensive simulations show that there are many cases where paced TCP gives worse performance than TCP without pacing, so pacing must be applied carefully. Ref. [5] argues that when pacing is used, $O(\log W)$ buffers are sufficient where W is the maximum congestion window size of each flow. However, architecture in Ref. [5] imposes a limitation on the congestion window size of all flows, because the buffer size in their proposal depends on the maximum window size. When there are not enough number of flows, these flows can utilize only a small amount of bandwidth because of the window size restriction. It gives high utilization when the network is slightly over-provisioned and there are enough number of flows.

First, we evaluate Paced (P.) and Non-Paced (N.P.) variants of TCP Reno, TCP NewReno, HSTCP with SACK [36] and XCP [16] transmission protocols. TCP Reno and NewReno are selected for showing the effect of packet recovery mechanisms when packet losses occur. HSTCP is selected because it can achieve high utilization in high-bandwidth product links much faster than Reno and NewReno without requiring router-assistance, so it is a possible alternative to XCP. In Ref. [16], it is shown that XCP significantly improves the overall performance when compared with TCP in terms of drop rate, utilization, queue build-up, delay and fairness. Therefore, we selected and focused on XCP. However XCP requires router-assistance. The income of router-assistance will be shown by comparing the performance of XCP with other TCP variants. We show that even rule-of-thumb buffers are not enough to N.P. XCP in some cases because of high burstiness and show the effect of pacing. Then, we introduce a methodology based on pacing and careful selection of parameters for decreasing the buffer requirements of P. XCP for very small buffered networks. We show the required buffer size for P. XCP on different traffic and network settings. Then, we compare the performance of other P. TCP variants with P. XCP.

In the second part, we replace evaluate the optical RAM size requirements of our rate-based paced XCP architecture that we proposed in Chapter 2. We show that proposed

architecture has very low buffer requirements with optical RAM due to advantage of $O(1)$ reading operation of optical RAM when compared with FDLs.

4.2 Decreasing Buffer Usage of Transmission Protocols

4.2.1 XCP Parameter Settings

Routers make use of buffers when there is over-utilization, so under-utilization leads to smaller buffer requirements. Long-term traffic over-utilization occurs when total traffic sent by sources has a rate higher than link capacity. It can be prevented by making sure that average input traffic sent by sources is less than bottleneck link capacity. TCP sources continue increasing window size until congestion window size limit of operating system is reached or packet loss occurs. Packet loss is assumed as signal of congestion. Therefore, avoiding over-utilization of links with TCP flows is hard. However, it is possible to guarantee under-utilized operation of the links by carefully setting a parameter in XCP. As explained before, Φ parameter of the efficiency controller of XCP is calculated according to the equation $\Phi = \alpha \cdot S - \beta \cdot Q/d$ where S is the spare bandwidth that is the difference between the link capacity and input traffic in the last control interval. Link capacity is given as a parameter to XCP. Therefore, a router needs to know the capacity (link speed) of its output link when calculating feedbacks to the packets passing through this link. Giving a capacity value less than real link-speed causes under-utilization and a value bigger than real link-speed causes over-utilization. When a link is properly under-utilized, the spare capacity can prevent buffer build-ups as we will show in this chapter.

4.2.2 Pacing

Even when long-term average input traffic rate is less than link capacity, it is possible to see over-utilization on short-term scales when traffic sources are bursty. When two or more packets contend, it causes a temporary over-utilization and buffer is used for solving contention. Ref [16] shows that XCP significantly improves the overall performance when

compared with TCP in terms of drop rate, utilization, queue build-up, delay and fairness. However, XCP is not suitable for small buffered networks, because it can behave even much more burstier than TCP. XCP is affected by all possible sources of burstiness (except slow start burstiness [33]) of TCP. However, there is one more important reason for burstiness in XCP. TCP increases its congestion window at most by one MSS (Maximum Segment Size) when it receives an ACK. Unlike TCP, when XCP receives an ACK, XCP updates its congestion window according to the formula $cwnd = max(cwnd + H\_feedback, s)$. When the congestion window is updated, the increase in window size may be bigger than a single packet, which cause injecting a big burst of packets back-to-back into the network.

Pacing is a possible solution for minimizing burstiness of both XCP and TCP. Pacing can be applied on data packets or ACK packets or both of them. There are different methods for determining the pacing interval. The most common and easiest one is applying an interval between sending times of packets, calculated by $\frac{RTT}{CWND}$, where $CWND$ is the current congestion window and RTT is the estimated round-trip time. There are also some other proposals like using the Bandwidth-Share-Estimate (BSE) of TCP-Westwood [37], or 4-hop propagation delay in wireless networks [38], or randomizing the interval [39].

Pacing is implemented by using a variant of token-based leaky bucket algorithm. Only data packets are paced since data packet pacing has a much bigger impact on the performance than ACK pacing. A packet inside the congestion window is sent to the output link, if there is a token inside the token buffer. After the packet is sent, a token is removed from the token buffer. If there is no token inside the token buffer, packet must wait until a token arrives. Arrival time of next token to the token buffer is found by dividing the size of the last sent packet with the current rate calculated by $\frac{RTT}{CWND}$. Changing the token buffer size affects the burstiness of the flow. Using a token buffer size of only one token gives the least bursty output traffic. Token buffer fill rate must be updated each time $CWND$ or RTT changes. Furthermore, pacing algorithms require fine course timers [34].

Even when the traffic is perfectly smooth and link is under-utilized, it is possible that two or more packets arrive from different input links at the same time. Again, buffer is used for storing contending packets. There are methods like slot-based reservation for

preventing packet contention, but they require complex control mechanisms, so they are not investigated.

4.2.3 Evaluation

We evaluated paced variants of TCP Reno, TCP NewReno, HSTCP with SACK and XCP over ns version 2.28 [28]. Their paced (P.) and non-paced (N.P.) variants are simulated and compared. Original XCP code in ns-2 does not take the arrival of ACK packets into account when calculating the input traffic rate information used in XCP's efficiency controller. It is modified so that ACKs are counted in the estimation of input traffic rate for preventing over-utilization and buffer build-ups due to ACK packets. In all simulations, time-stamps option is used and agents have fine course timers. There is no limit on congestion window size of TCP and XCP flows. Slow start threshold of TCP is 64 packets. Data packet size is 1000Bytes including the headers in both TCP and XCP simulations. A dumbbell topology and a parking-lot topology are used for computer simulations. Simple drop-tail queuing is used. Queue limits are set in terms of Bytes instead of packets for a more realistic simulation.

XCP is simulated with two different parameter sets. One of them is the original parameter set (O.P.) used in [16]. They are α=0.4, γ=0.1 and β=0.226. Capacity of the output links are given as their real link-speed to the XCP algorithm in routers. The second set is the conservative parameter set (C.P.) used for minimizing the buffer usage. α parameter is decreased to 0.2, which gives a slower but more stable link utilization and decreases utilization overshoots. When α parameter is decreased, it is also necessary to decrease γ parameter responsible for bandwidth shuffling. Otherwise, too much under-utilization may occur in some links in case these links carry flows that are bandwidth throttled in other bottleneck links as explained in Ref. [16]. Therefore, γ=0.05 is used instead of γ=0.1 in Ref. [16]. β must be selected according to the formula $\beta = \alpha^2\sqrt{2}$ as proved in Ref. [16], so β=0.056 is straightforward. Capacity of the output links are given as 90% of their real link-speed (target utilization) to the XCP routers. The spare 10% capacity gives a safety

margin against possible oscillations in utilization.

Extensive simulations on mixed flow environments with competing P. and N.P. TCP variants and performance of P. TCP flows in large buffered networks are already available in Ref. [35] and Ref. [39], so their results are not presented here.

4.2.4 Dumbbell Topology Simulations with Long Flows

A dumbbell topology is used for simulations with static flows. The capacity of the bottleneck link is 622Mbps (OC-12). Both edges of bottleneck link have extension links. Extension links are 2.4Gbps (OC-48). RTT distribution between source-destination node pairs ranges from 64ms to 100ms for preventing ACK-clocking. Average RTT is 82ms. Bottleneck delay is 30ms.

A limited number of FTP flows start at random times between [0-10]s and continue until the simulation stops. It is assumed that flows always have data to send. Total simulation duration is 100s. Only the data between [40-100]s is used in average utilization calculations. Two-way traffic is created by applying reverse traffic with the same properties as the forward traffic for showing the possible effects of ACK-compression problem [6].

Required Buffer Size

XCP tries to minimize the buffer occupancy and not to drop any packets. First, we find the maximum queue occupancy of bottleneck link by P. and N.P. XCP with O.P. and C.P. when there are different number of long flows ranging from 2 to 800. Size of all buffers are limited to rule-of-thumb bandwidth-delay product.

In Fig. 4.1(a), x-axis shows the number of long flows and y-axis shows the maximum queue occupancy of bottleneck link, both in log scale. In Fig. 4.1(b), x-axis shows the number of long flows and y-axis shows the average utilization of bottleneck link. X-axis is in log scale. Y-axis is between [0-1] where one means 100% utilization. When the buffer occupancy of P. XCP with O.P. and C.P. are compared, Fig. 4.1(a) shows that C.P. can decrease the buffer requirement greatly, depending on the number of flows. As an example,

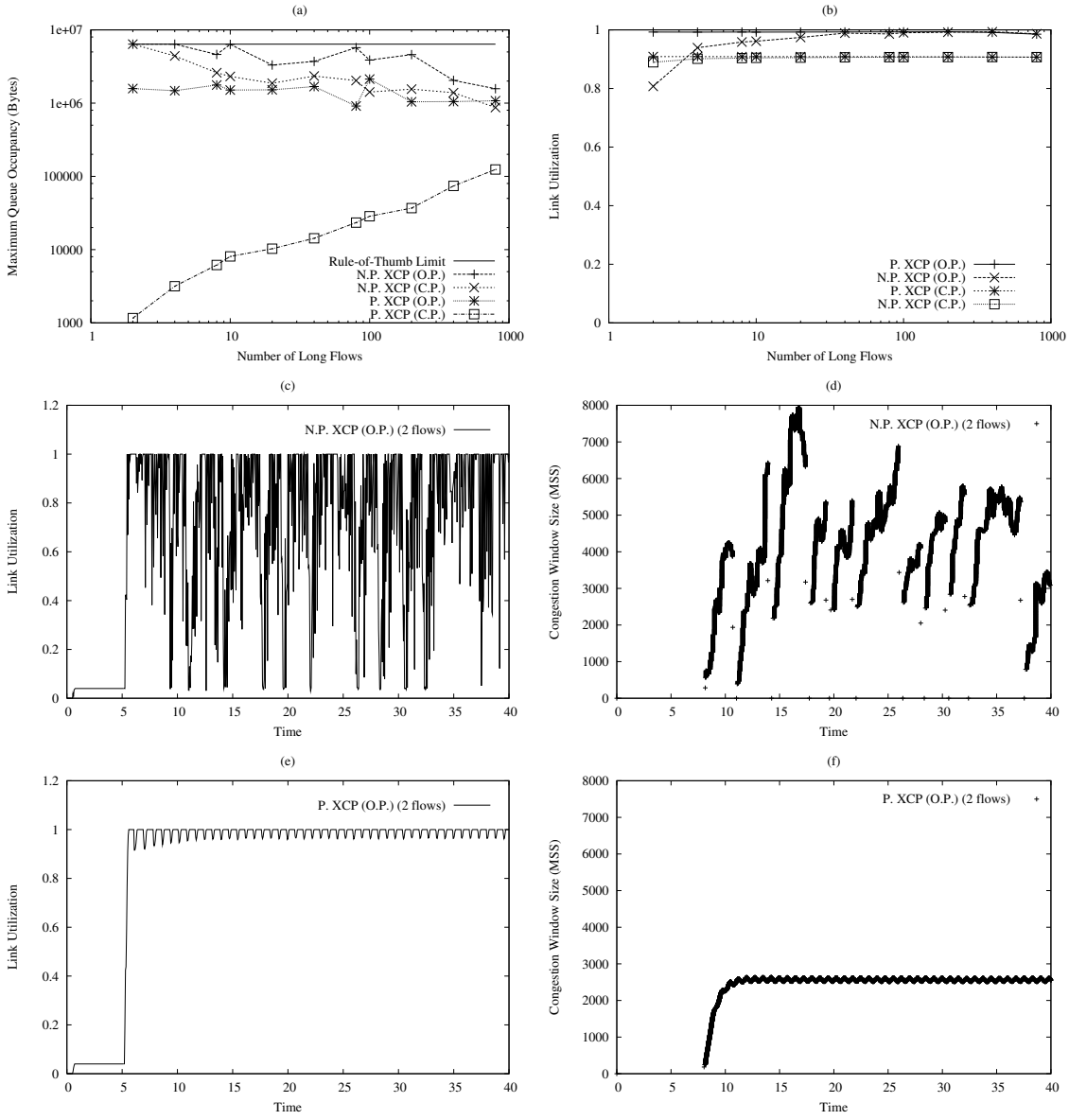


Figure 4.1: (a) Maximum buffer occupancy and (b) bottleneck utilization with rule-of-thumb buffer size. (c) Transient bottleneck utilization and (d) congestion window size of Non-Paced XCP with original parameter set. (e) Transient bottleneck utilization and (f) congestion window size of Paced XCP with original parameter set.

when there are 100 flows, the required buffer size is around 6.4MBytes according to the rule-of-thumb, and around 638KBytes according to Appenzeller's $\frac{RTT \times BW}{\sqrt{N}}$ formula. As seen in

Fig. 4.1(a), the maximum buffer occupancy of P. XCP with C.P. is only 23,360Bytes. The maximum buffer occupancy decreases as the number of flows decrease.

In the Fig. 4.1(a), we see that rule-of-thumb buffer (6.4MBytes) size was not enough for N.P. XCP when the number of flows is low. N.P. XCP becomes oscillatory and starts losing many packets. Therefore N.P. XCP gives low average utilization when the number of flows is low as seen in Fig. 4.1(b). The reason of this behavior is the burstiness of XCP and ACK compression problem. XCP is a bursty protocol, because it may increase window size in big steps, so it may send too many data packets back to back. The ACK packets are compressed by bottleneck link along the data packets of reverse traffic. As the arrival of ACK packets to the sender is delayed, ACK-starvation occurs and sender can not send new data packets in this period, which may cause under-utilization of the bottleneck link if the number of sources is low. Then, the receiver gets a burst of ACK packets compressed at the end of a data burst and starts sending a burst data packets, which can over-utilize the bottleneck link. Therefore, there is a big oscillation in the spare utilization variable of Φ calculation. At the same time, queue size starts oscillating between a very low and a high value due to high burstiness of input traffic. XCP's efficiency controller uses only the persistent (minimum) queue size without taking the maximum buffer occupancy into account for calculating the feedback, so the queue size information in the efficiency controller algorithm does not help stabilizing the system. Therefore, the calculated router feedback information becomes oscillatory, which further increases the burstiness. Routers require a buffer that is much larger than rule-of-thumb so that the buffer can handle the big oscillations in the queue size.

In the simulations, the buffer limit is set to rule-of-thumb, which is not enough for N.P. XCP when the number of flows is low. Therefore packet drops occur. Current XCP implementation in ns-2 drops its window size to half and applies TCP Reno's recovery algorithm. This behaviour can interfere with XCP's own window control algorithm and further increase the oscillations. Figures 4.1(c),(d),(e),(f) show the transient bottleneck link utilization and congestion window size of one of the P. and N.P. XCP flows with O.P. when there are two flows (four flows including the reverse flows). As seen in Fig. 4.1(c),(d),

utilization and congestion window size of N.P. XCP shows unstable behavior and causes many packet drops and timeouts. Even when an unlimited buffer is provided, N.P. XCP shows an oscillatory behavior. Using C.P. decreases the level of oscillations, but can not solve it fully. On the other hand, as seen in Fig. 4.1(f), window size of P. XCP reaches its fair-share in a short time and stays almost constant throughout the simulation.

As the number of flows increases, the oscillations and therefore the required buffer size decreases, because the input traffic rate becomes smoother due to higher level of traffic multiplexing.

Small-buffered Network Simulations

In Fig. 4.1(a), it is seen that maximum buffer occupancy values of P. XCP with C.P. is very small. We set the buffer limit of bottleneck link to those maximum buffer occupancy values of P. XCP with C.P. in simulations with corresponding number of flows and evaluate the performance of other TCP variants and XCP with C.P..

In Fig. 4.2(a), all P. TCP variants achieve over 80% utilization when the buffer size is small. P. XCP with C.P. gives almost constant utilization around the value of capacity parameter 90%. Among the P. TCP variants, P. NewReno gives the second best utilization that is marginally higher than P. HSTCP. P. Reno has the lowest utilization. The reason for not achieving 100% utilization is synchronization of congestion window of flows. When the total input rate exceeds the link capacity, queue starts dropping packets and many flows lose packets due to uniform arrival of packets due to pacing, so many flows decrease their window size, which brings synchronization of flows. Furthermore, many flows lose multiple packets inside a congestion window. Reno is known to have low performance when there are multiple losses inside a window, so its paced variant gives the lowest performance.

When there is synchronization, we can expect to get higher utilization with P. HSTCP than P. NewReno especially when average window size is large. This is because, unlike NewReno, which decreases its window to half, HSTCP decreases its window size to a value higher than half size when window size is large. However, HSTCP increases its window size at steps larger than one MSS per RTT when window size is large. As a result, P. HSTCP

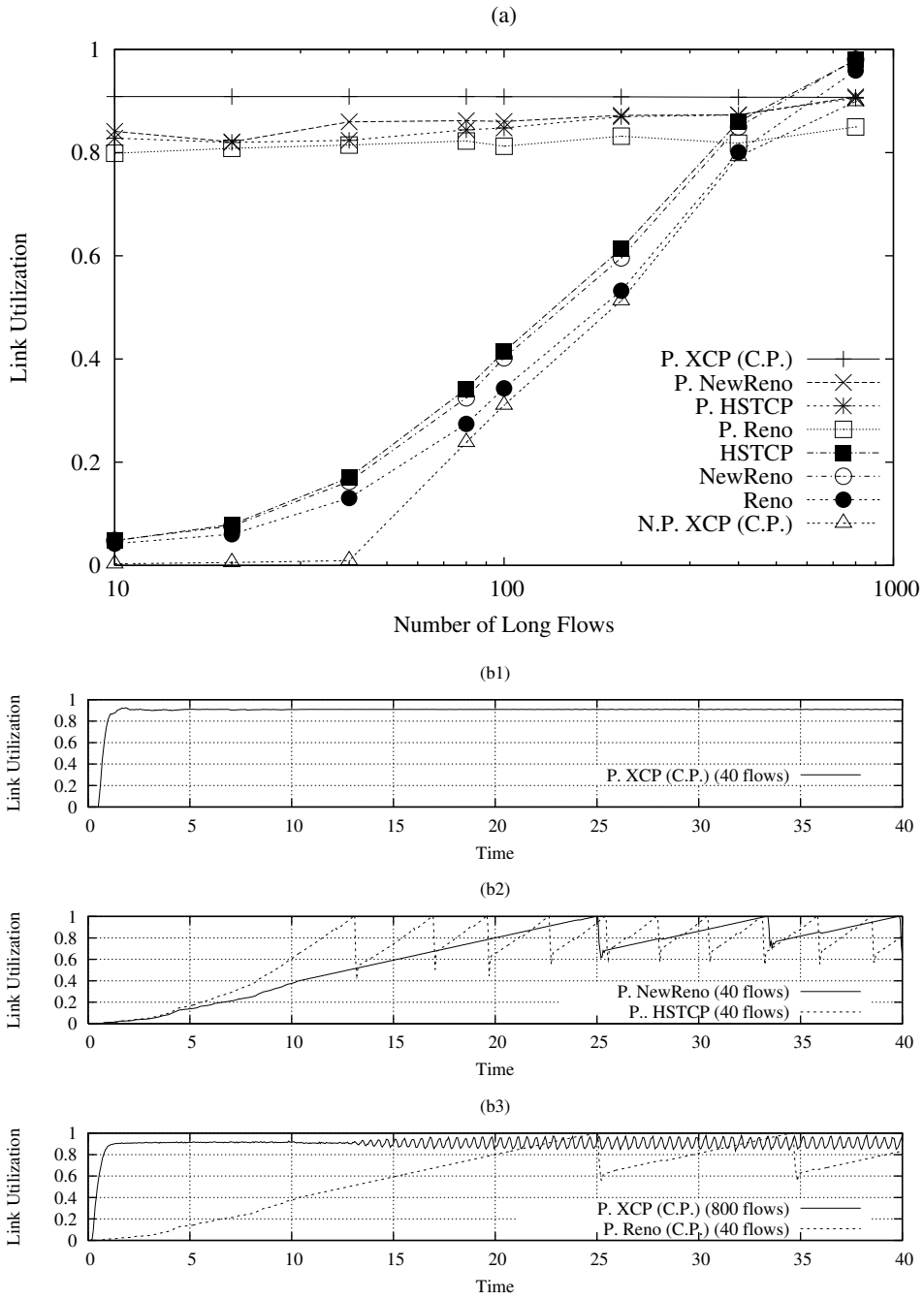


Figure 4.2: (a) Average bottleneck utilization with small buffers. (b) Transient bottleneck utilization of Paced XCP with conservative parameter set and Paced TCP variants.

may cause a heavier congestion and drop more packets than P. NewReno. Therefore, the number of flows losing packets and getting synchronized is more than P. NewReno. Higher level of synchronization among P. HSTCP flows causes lower utilization than P. NewReno.

Fig. 4.2(a) shows that N.P. TCP variants give low utilization unless there is a large number of flows, due to high packet loss rate caused by their high burstiness and small-buffer size of the bottleneck link. However, when there are many flows, the aggregated throughput of flows becomes high enough to achieve high utilization. Furthermore, simulated buffer size increases with the number of flows according to the maximum buffer occupancy of Paced XCP with C.P. as shown in Fig. 4.1(a). When there are 800 flows, the simulated bottleneck buffer size is 124,040Bytes. Average utilization of N.P. TCP variants is over 90% percent. It is important to note that according to Appenzeller's formula, the required buffer size is only around 225KBytes that is close to the simulated buffer size, so our results on N.P. TCP comply with Appenzeller's results.

When we compare the average utilization of N.P. TCP variants, we see that HSTCP gives the best performance. NewReno gives a marginally lower performance and Reno gives the worst performance. Average window size of flows is small, so HSTCP's congestion control algorithm operates with almost the same window increase and decrease factors as TCP. However, HSTCP gives a much better performance than Reno and a marginally better performance than NewReno due to performance improvement of SACK in environments with high loss rate.

When we check the transient utilization of first 40 seconds of P. TCP variants with 40 flows and P. XCP with 40 and 800 flows in Fig. 4.2(b1) and Fig. 4.2(b3), we see that P. XCP with C.P. achieves its target utilization in a short time. P. HSTCP in Fig. 4.2(b2) achieves the second fastest conversation time in terms of fully utilizing the link, because of its fast window increase. P. NewReno in Fig. 4.2(b2) and P. Reno in Fig. 4.2(b3) give the worst convergence time. P. TCP variants show a saw-tooth like transient utilization due to high level of synchronization.

When transient utilization graphs of P. XCP with 40 and 800 flows are compared in Fig. 4.2(b1) and (b3), we see that utilization of 800 flows simulation is almost constant in

the first 15 seconds and then utilization starts to oscillate. In the first 15 seconds, window size of each flow converges to its fair share. Then, small changes in the utilization cause all flows to increase or decrease their window size around their fair share simultaneously. XCP and TCP's congestion window size is a real number. However, effective window size is an integer number of MSS. Therefore, the flow speed changes in discrete steps. If the average window size is small like in 800 flows simulation, synchronized change of congestion window size of many flows causes a big oscillation in input traffic rate. This oscillatory behavior seen when the average window size is small, is also shown in [16] and stated that buffer is used for absorbing this oscillation. However, absorbing oscillations in the buffer is not possible in small buffered networks, so target utilization is chosen less than 100% in order to be sure that the highest rate of the oscillatory input traffic is less than link capacity as in Fig. 4.2(b3).

4.2.5 Dumbbell Topology Simulations with Web-like Traffic

A web-like traffic is simulated by applying dynamic short flows and long FTP flows on a dumbbell topology for showing the performance of pacing algorithms under a more realistic traffic. Simulated link speeds are as given in the previous simulations. There are N nodes for short flows and 40 nodes for long flows giving a total of $N+40$ nodes on each side of the bottleneck link, where N is between [0-400]. RTT distribution of both long and short flows ranges from 64ms to 100ms. Average RTT is 82ms. Bottleneck delay is 30ms. All sources start at random times between [0-10]s. Long flows continue until the simulation ends. File size of short flows is Pareto distributed with shape of 1.35 and average of 30 packets as in [16]. After sending a file, a node stays OFF for an exponential distributed times with average of 0.2s. A node reserved for short flows carries only one flow at a time. There is two-way traffic by applying reverse traffic with same properties as the forward traffic. The average file size of dynamic flows is small, so dynamic flows usually finish before reaching a high window size and the total traffic injected by the dynamic flows is low. Simulation duration is 50s and only data from [20-50]s is used in average utilization calculations.

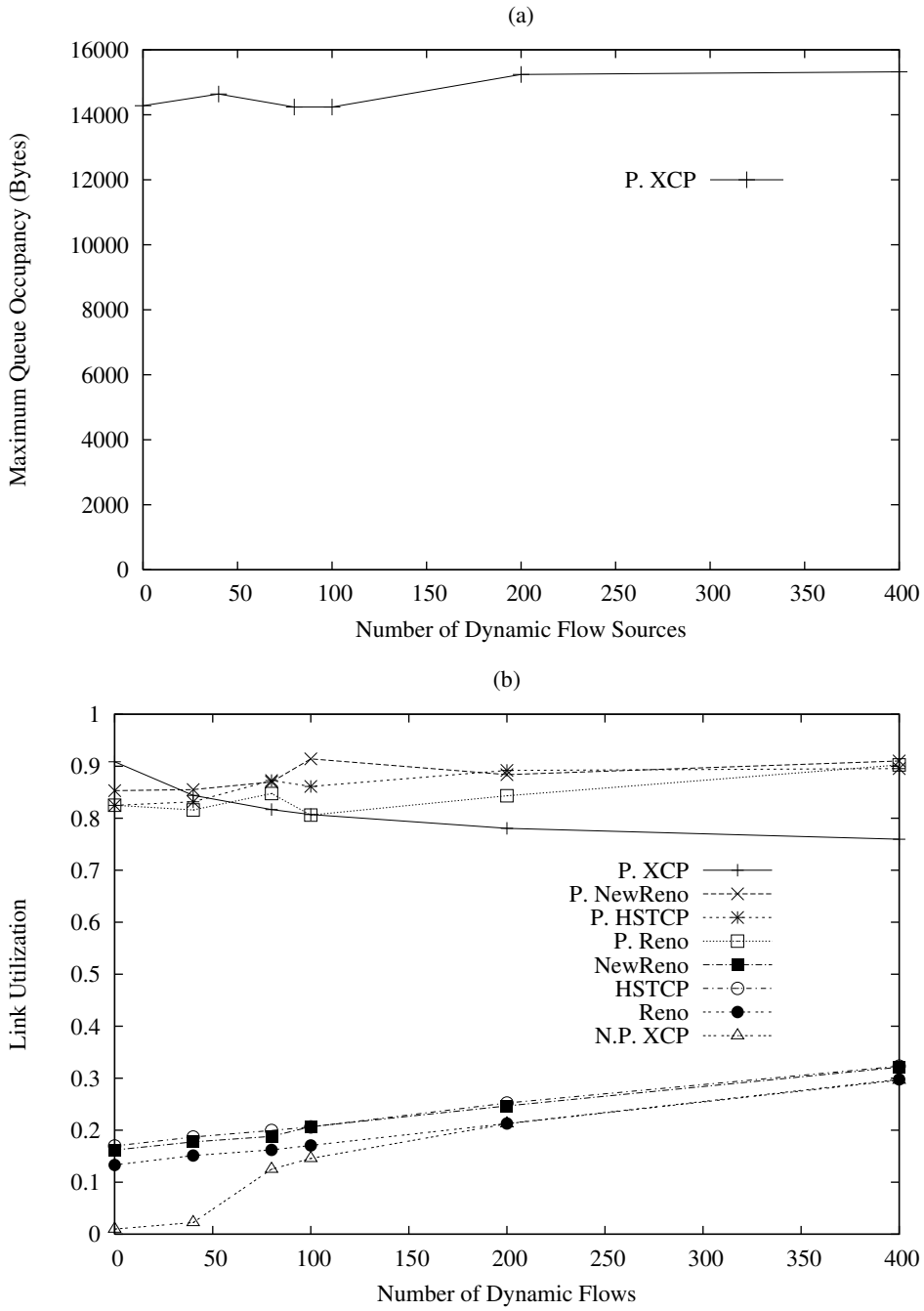


Figure 4.3: (a) Maximum queue occupancy of Paced XCP with conservative parameter set. (b) Average bottleneck utilization with small buffers.

First, we find the maximum queue occupancy of P. XCP with C.P., by setting the size of buffers to rule-of-thumb and changing the number of short flows. As seen in the Fig. 4.3(a), even when the number of dynamic flows is high, maximum buffer occupancy do not show a big difference because of the low rate of traffic injection (less than 30% of bottleneck link utilization) of short flows and decreased utilization of bottleneck link as seen in Fig. 4.3(b). Lower bottleneck utilization decreases the maximum buffer occupancy due to decreased probability of contention.

We simulate N.P. XCP with C.P. and other TCP variants by applying these buffer occupancy values with the corresponding number of short flows. As seen in Fig. 4.3(b), P. XCP gives lower utilization as the number of dynamic flows increase. It is mainly because of the fairness control algorithm of XCP. XCP tries to fairly distribute the bandwidth among all flows. However, the dynamic flows are short, so they usually finish before reaching their fair-share. Their share becomes unused and the achievable utilization decreases as the number of dynamic flows increases.

Dynamic arrival and departure of short flows decreases the synchronization level of flows. Therefore, P. HSTCP gives higher utilization than P. NewReno at some points. Again, P. Reno generally gives worse performance than P. NewReno and P. HSTCP. Average utilization of P. TCP variants increases as the number of dynamic flows increase due to decreases in the level of synchronization.

When we check the average utilization of N.P. TCP variants, we see that NewReno and HSTCP have almost the same performance. Again, Reno gives the worst performance. Utilization of N.P. TCP variants increase as the number of dynamic flows increase due to aggregated bandwidth of long and short flows.

4.2.6 Parking Lot Topology Simulations

A parking lot topology with 3 middle links is used for fairness and utilization simulations. Link capacities are as given in the previous simulations. Each middle link has 10ms propagation delay. There are 10 long flows passing through all three middle links. Also there are

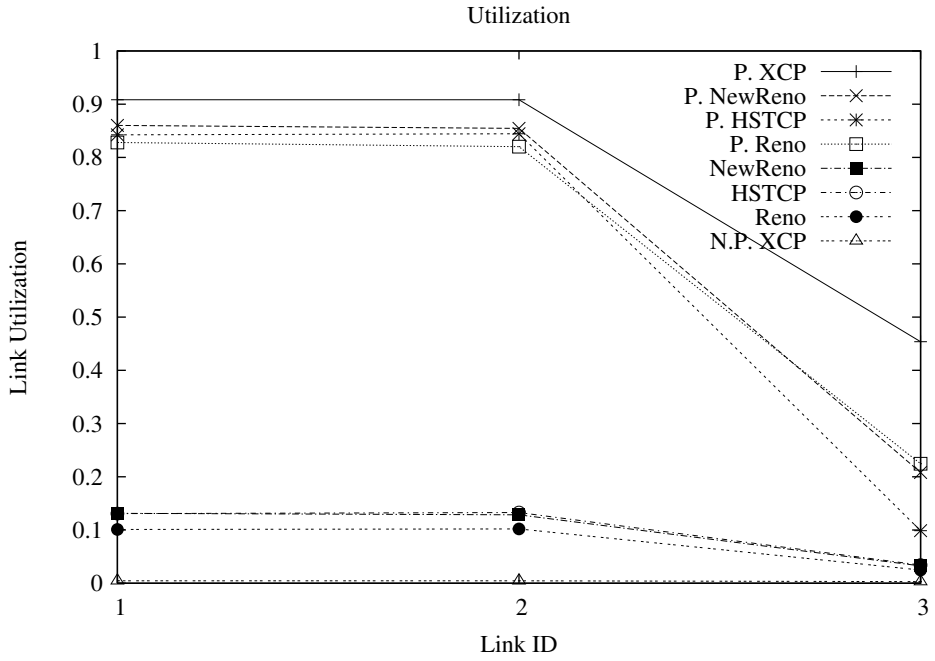


Figure 4.4: Average link utilizations.

a set of 10 short flows passing through first middle link and another set of 10 short flows passing through second middle link. Third middle link has only long flows. Therefore, there are two bottlenecks at the same time. RTT distribution long flows (short flows) range from 64ms (24ms) to 100ms (60ms) with average RTT of 82ms (42ms), respectively. Two-way traffic is created by applying reverse traffic with same properties as the forward traffic. Simulation duration is 100s and only data from [40-100]s is used in average utilization calculations.

First, we find the maximum queue occupancy of P. XCP with C.P., by setting the size of buffers to rule-of-thumb. The maximum queue occupancy is found as 10,200Bytes. Then, we simulate N.P. XCP with C.P. and other P. and N.P. TCP variants by applying this buffer size to all middle links. Fig. 4.4 shows the average utilization of three middle links. X-axis shows the middle link number. When we compare the utilization of P. XCP and P. TCP variants on the first two links, we see that P. XCP gives the highest utilization. P.

P. XCP	P. Reno	P. NewReno	P. HSTCP
1	0.78	0.76	0.44
N.P. XCP	N.P. Reno	N.P. NewReno	N.P. HSTCP
0.76	0.72	0.74	0.73

Table 4.1: Fairness index

NewReno is the second, P. HSTCP is the third and P. Reno is the last one. This result is the same as the dumbbell topology simulations. However, results are different on the third link, which shows the average utilization of long flows. P. XCP has the highest utilization, which points to a good fairness among long and short flows. Long flows are not penalized by high RTT or multiple bottlenecks. P. Reno has the second and P. NewReno has the third highest utilization. P. HSTCP has a low utilization pointing to a high discrimination against long flows with multiple bottlenecks. When we compare the average utilization of N.P. TCP variants, we see that NewReno and HSTCP have a similar utilization and Reno has the worst utilization at all links. N.P. XCP with C.P. has less than 0.5% utilization, which is the lowest among all tested TCP and XCP variants, so it is hard to see utilization of N.P. XCP in the figure.

Fairness of goodput among long and short flows is evaluated by using Jain's fairness index $f = \frac{(\sum_{i=1}^{i=n} x_i)^2}{n \sum_{i=1}^{i=n} x_i^2}$, where n is the number of flows and x_i is the goodput of flow i [40]. Table 4.1 shows that P. XCP has perfect fairness. P. Reno and P. NewReno have similar fairness, which is lower than P. XCP. However, P. HSTCP has a low fairness. N.P. TCP variants have similar fairness. Fairness value of N.P. XCP is meaningless, because it achieved less than 0.5% utilization due to totally unstable behavior.

4.3 Rate-based XCP Pacing with Small Optical RAM

In Chapter 2 and 3, we evaluated the buffer size requirements for the proposed architecture in the case of FDL buffering. In this section, we replace FDL buffering with optical RAM and evaluate the optical RAM size requirements of our proposed architecture. We show that proposed architecture has very low buffer requirements with optical RAM due to advantage

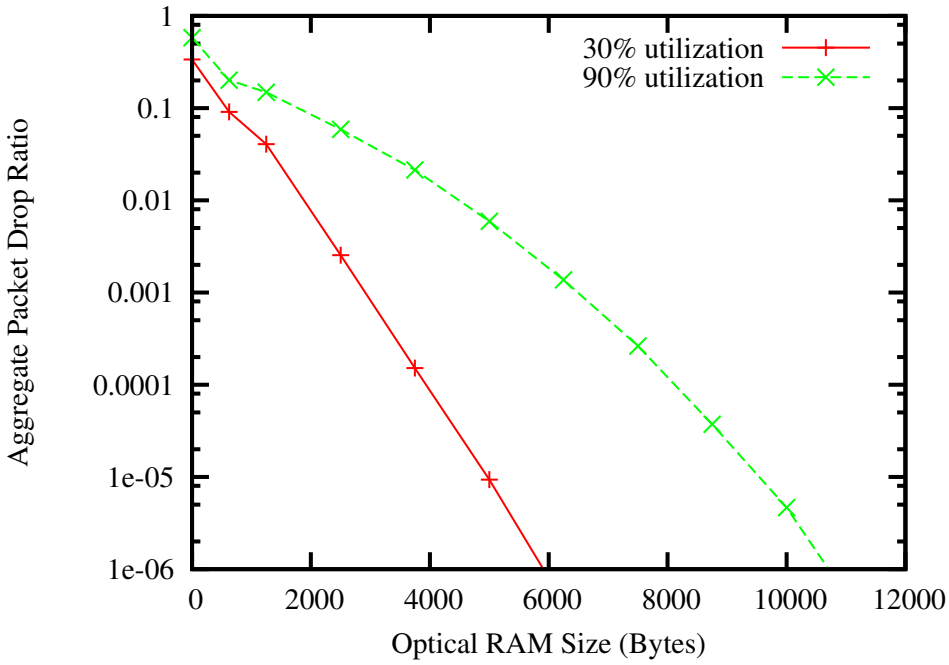


Figure 4.5: Aggregate packet drop ratio as a function of optical RAM size.

of $O(1)$ reading operation of optical RAM when compared with FDLs.

4.3.1 Evaluation and Results

NSFNET network is simulated by using a simulator developed on ns-2 [28]. Simulation parameters are mainly the same as in Chapter 2 and 3. All links (including edge and core links) apply optical packet switching. Edge nodes apply electronic buffering, but core routers use only optical RAM for buffering. All switches employ output buffering and cut-through bit-synchronized switching and buffering. It is assumed that there is a backlogged traffic at edge buffers, so each edge node sends traffic to all other edge nodes at the maximum possible rate controlled by XCP. The capacity of the data wavelength is set to 1Gbps. XCP's α, β and γ parameters are selected as 0.2, 0.056 and 0.05, respectively. XCP control period of core routers and probe packet sending interval of edge routers is 50ms. 60% of the packets are 40Bytes and 40% of the packets are 1500Bytes. MSS is

1500Bytes. Fig. 4.5 shows the ratio of the packets dropped in NSFNET topology vs. the optical RAM size limit of output links of core routers by using our architecture and setting the target wavelength utilization of XCP to 30% and 90%. It can be seen that low packet drop ratio can be achieved with very small optical RAM buffering. If we want very low packet drop ratio like 10^{-6}, only around 4MSS (6Kbytes) and 7MSS (10.5Kbytes) optical RAM per link may be enough for 30% and 90% utilization, respectively. On the other hand, around 0.1% packet drop ratio may be enough for internet traffic. In this case, only around 2MSS (3Kbytes) and 4MSS (6Kbytes) optical RAM per link may be enough for 30% and 90% utilization in NSFNET, respectively.

4.4 Conclusions

TCP and XCP are well-known to behave bursty, so they have a low utilization in very small buffered high bandwidth-delay networks. In the first part, we showed that even rule-of-thumb sized buffers are not enough for XCP in some cases due to high burstiness of XCP. We showed that XCP can be adapted to small buffered networks by applying pacing and a careful selection of parameters. We compared P. XCP and different TCP variants and showed that P. XCP is a strong candidate for achieving high performance in small buffered networks.

A big disadvantage of XCP based algorithms is that they require deployment of XCP capable senders, receivers and routers. On the other hand, it is possible to use P. TCP algorithms by updating only TCP senders.

Even though paced sources minimize the burstiness of traffic they send into the network, queues can change the intervals between the packets and make the traffic burstier in a fashion similar to ACK-compression and increase the buffer requirements of bottleneck link. Simulations on more realistic traffic models and bigger topologies are required for better understanding in the buffer requirements of paced algorithms.

In the second part, we evaluated the packet drop ratio of a mesh OPS network using small optical RAM buffering with our rate-based XCP pacing architecture. Simulation

results show that only a few packet buffers per output may be enough for low packet drop ratio and high utilization.

Chapter 5

Node Pacing for Optical Packet Switching

5.1 Introduction

In Chapter 2, we proposed an architecture for decreasing buffering requirements by using XCP framework. However, implementation cost XCP framework can be high. Therefore, in this chapter we propose a light-weight alternative architecture that does not require XCP framework.

Ref. [8] proposes applying traffic shaping at edge nodes of OPS network for minimizing traffic burstiness. It proposes a delay-based pacing algorithm that adaptively chooses packet spacing according to input traffic class for achieving bounded delay requirements. Ref. [10] proposes a RC traffic shaper for ATM networks that smooths the traffic by adjusting the output rate based on the buffer occupancy that depends on the input traffic rate. Output rate is linearly proportional to the shaping buffer occupancy. The problem of the proposed algorithm is that it requires a large buffer for preventing cell loss. Large buffer brings higher delay. Furthermore, peak cell input rate must be known. In order to solve these problems, Ref. [11] proposes Interval Filter Shaping Algorithm (IFSA) that smooths cell inter-arrival time with a low latency. IFSA makes use of a low pass filter and special scheduler for

smoothing the traffic.

We propose an algorithm that can shape traffic at edge or core nodes by using the buffer occupancy information like the RC traffic shaper. However, our design solves the problems of RC traffic shaper, by using a piecewise linear output transfer rate control function and making use of average input traffic rate information calculated inside the node. We show that our pacing algorithm can considerably increase the achievable utilization of very small optical RAM buffered optical core links or namely throughput of TCP flows using these links.

5.2 Architecture

5.2.1 Traffic Shaper

The RC traffic shaper [10] calculates the cell output rate according to the linear equation

$$\lambda_o(t) = \frac{B_c(t)}{B_c}\lambda_{imax} \tag{5.1}$$

where $\lambda_o(t)$ is the output traffic rate, $B_c(t)$ is the buffer occupancy, B_c is the buffer capacity and λ_{imax} is the maximum input traffic rate. The algorithm sets the output traffic rate equal to input traffic rate when buffer is full, in order to prevent packet drops. However output traffic rate is limited by the link speed and if input traffic rate is higher than the link speed, buffer may overflow and cause packet drops.

We propose using a piecewise linear transfer function instead of linear. Fig. 5.1(a) shows the transfer function where B_t is buffer threshold, B_c is buffer capacity, S_s is initial link speed and S_l is link capacity. Output rate in RC traffic shaper reaches to the input rate only when the buffer is full. However, in our algorithm output rate reaches to the input rate after a buffer threshold is reached. This provides a safety margin for decreasing buffer overflows in case input traffic rate is higher link speed as there is still free space in the buffer. However, this may not be enough for decreasing average buffer occupancy. When this fixed

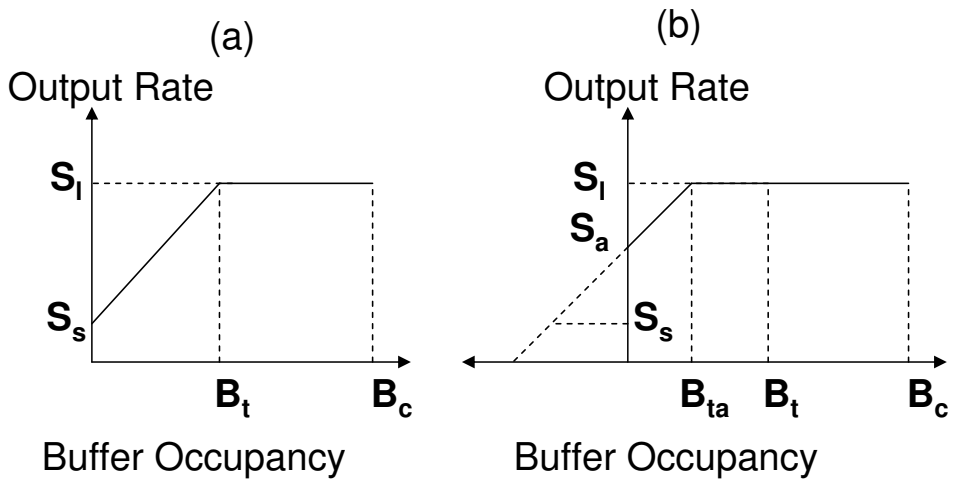


Figure 5.1: (a) Transfer function and (b) adaptive transfer function

transfer function is used, an average output traffic rate higher than S_s still requires a non-zero average buffer occupancy that increases packet drop probability, especially in a very small buffer network. For example, an average output traffic rate equal to link capacity requires an average buffer occupancy of B_t. In order to solve this problem, we make use of both buffer occupancy and average input traffic rate for calculating output traffic (pacing) rate. We adaptively shift the x axis (buffer occupancy) of the transfer function according to average arrival rate, so that pacing uses less buffer space. Fig. 5.1(b) shows the adaptive transfer function where S_a is the average input traffic arrival rate and B_{ta} is the new buffer threshold after shifting the transfer function according to S_a. If S_a is smaller than S_s, S_a is taken as S_s. B_{ta} can be calculated by simply $B_{ta} = B_t \frac{S_l - S_a}{S_l}$. This adaptive transfer function allows output traffic rate being equal to average input traffic rate even when the buffer occupancy is zero, so average buffer occupancy can be decreased.

5.2.2 Switch and Scheduler Architectures

Core nodes use an input buffering architecture with virtual output queuing (VOQ) scheduling. The reason is that input buffered switch architecture has a speedup of 1, so it has

smaller switching fabric size and cost when compared with output buffered and combined input-output buffered switch architectures. However, a well-known problem of input buffering is head-of-line blocking, which limits the achievable utilization. We apply virtual output queuing (VOQ) scheduling for minimizing this problem.

Proposed algorithm requires the output link buffer occupancy information that is normally zero for an input buffered switch as there is no buffer at the output links. Therefore, effective buffer occupancy value of an output link is calculated by sum of virtual output queues destined to this output link.

5.3 Evaluation

5.3.1 Simulation Settings

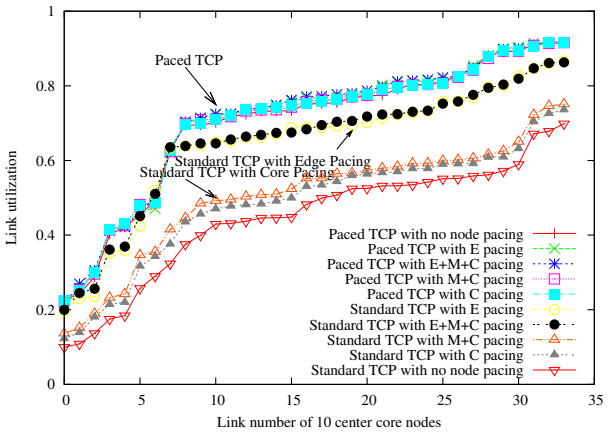
Proposed network architecture and algorithms are implemented over *ns* version 2.32 [28].

Abilene-inspired topology from Ref. [41] is used in simulations. The topology has a total number of 869 nodes that are divided into three groups as:

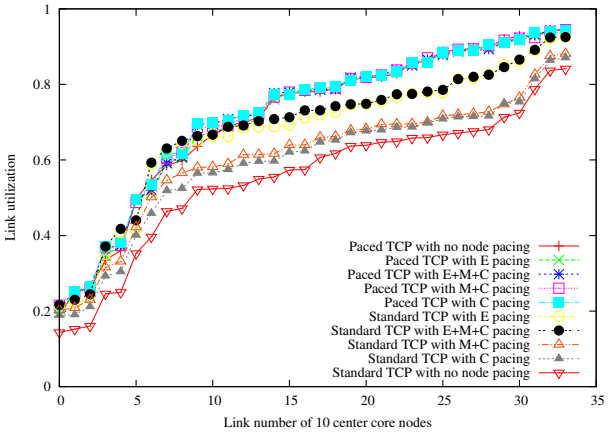
- Center core nodes (C): 75 core nodes that are not connected to edge nodes;
- Middle core nodes (M): 106 core nodes that are connected to edge nodes;
- Edge nodes (E): 698 nodes that IP traffic enters or exits the network topology.

This topology is selected because it is possible have large number of flows multiplexing at the core backbone links and high utilization due to ring architecture. Non-paced TCP Reno and Paced TCP Reno are used as traffic sources. A total of 4581 TCP flows start randomly and send traffic between randomly selected edge node pairs. Total simulation duration is 20s. TCP data packet size is 1500Bytes.

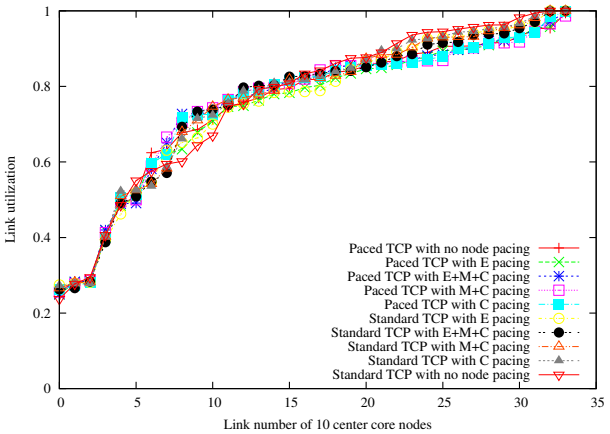
There is a single data wavelength on links. Propagation delay of edge and core links are 0.1ms and 1ms, respectively. All links have 1Gbps capacity. Core node links are simulated with an optical RAM input buffer size (B_c) of 50Kbits, 100Kbits or 2Mbits. Electronic



(a) 50Kbit buffer



(b) 100Kbit buffer



(c) 2Mbit buffer

Figure 5.2: Simulation results

RAM input buffer size of edge nodes is 100Mbits (100ms). B_t is always selected as half of the B_c. S_s is selected as 0.1Gbps and 0.01Gbps for core and edge nodes, respectively.

Different combinations of edge node pacing (E), middle core node pacing (M), center core node pacing (C) and no node pacing are simulated with Non-paced TCP and Paced TCP flows. Figures 5.2(a), 5.2(b) and 5.2(c) show the sorted utilization of backbone links between 10 center core nodes at the very center of the network at the end of the simulation. In the figures, input buffer size of links of core nodes are 50Kbits, 100Kbits and 2Mbits, respectively. In all plots, utilizations are sorted independently from lowest to highest, so there is not a one to one correspondence between link numbers of the plots.

When we check the simulation results in Fig. 5.2(a), we see that simulation results of Paced TCP traffic are almost the same for different node pacing methods. The reason is that Paced TCP traffic is already smooth enough, so extra node pacing does not bring improvement when Paced TCP is used. The next group of lines show the utilization when Non-paced TCP is used with Edge node pacing. Again we see that edge node pacing makes the traffic smooth enough, so additional center and middle core pacing do not make a big difference. Unlike TCP pacing, the edge node pacing paces the traffic without any knowledge of RTT of flows, so it can not space the packets optimally. Therefore, its utilization is a bit lower than TCP pacing. The next group of lines is center core pacing, and center core + middle core pacing methods where the latter has a bit higher utilization. They have a lower utilization than edge pacing, because there are fewer number of core nodes than edge nodes and core nodes have a much smaller buffer (2000 times smaller) than edge nodes that is not enough to fully pace the traffic. The last plot is the simulation result of no pacing that has the lowest utilization as expected. The figure shows that even when node pacing is applied to only core nodes with very small buffers, it is possible to achieve a considerable throughput increase when compared with no pacing case. If node pacing is applied to edge nodes, achievable utilization can be almost doubled at low utilized links, without requiring Paced TCP.

When we increase the core buffer size to 100Kbits in Fig. 5.2(b), we see that the utilization gap becomes lower. However, there is still a considerable gain by core node

pacing over no pacing case. When we increase the core buffer size to 2Mbits in Fig. 5.2(c), we see that simulations give indistinguishable results that shows that 2Mbits buffer size is enough to solve burstiness problem, so pacing does not make a difference.

5.4 Conclusions

In this chapter, we proposed a pacing algorithm at the edge or core backbone nodes in order to increase the utilization efficiency of very small optical RAM buffered OPS networks. It is a light-weight control, so it is easier and cheaper to implement than XCP based control. Our simulation results show that our edge or core based node pacing algorithm can considerably increase the achievable utilization of very small buffered optical core links or namely throughput of TCP flows using these links.

Chapter 6

Conclusions

In this thesis, we proposed methods for minimizing the buffer size of optical networks. In Chapter 2, we proposed a new XCP protocol called Rate-based Paced XCP specially designed for small buffered optical networks and showed that it has a very good performance. We evaluated the packet drop rates with extensive simulations on a meshed network with multiple-hop paths and showed how FDL requirements change with slot size, FDL granularity, scheduling and packet size distribution. Simulation results with meshed networks showed that our architecture can provide much low packet loss ratio lower than poisson traffic in core OPS networks with small FDL buffers. We showed that large packets and small packets have different FDL requirements. Small packets require low granularity for low packet drop rate, but large packets require high granularity for decreasing the number of required FDL lines.

As a next step, in Chapter 3 we proposed new switch architectures specially designed for Rate-based Paced XCP for further decreasing the buffer requirements. We compared the buffering architectures input buffering with VOQ, and output buffering with void filling. We evaluated the packet drop rates depending on FDL granularity and packet size distribution. We showed that input buffering requires comparable number of delay lines as output buffering architectures at 30% utilization, which is typical for backbone links of network operators, with pacing. Input buffering can be implemented by dividing the

switching fabric into smaller switches instead of a single and large main switch, so input switching may decrease costs when the cost of a large and single switch is higher.

In Chapter 4, first we evaluated the performance of paced transmission protocols and compared the performance in small-buffered networks. We showed that even rule-of-thumb sized buffers are not enough for XCP in some cases due to high burstiness of XCP. We showed that XCP can be adapted to small buffered networks by applying pacing and a careful selection of parameters. We compared P. XCP and different TCP variants and showed that P. XCP is a strong candidate for achieving high performance in small buffered networks. Then, we evaluated the optical RAM buffer size requirements for our proposed architecture from Chapter 3. We showed that our architecture has very low buffer requirements with optical RAM due to advantage of $O(1)$ reading operation of optical RAM.

In Chapter 5, we proposed a core pacing architecture that is very light-weight, easy to implement and can operate without using XCP framework. We compared the buffering architectures input buffering with VOQ, and output buffering with void filling. We evaluated the packet drop rates depending on FDL granularity and packet size distribution. We showed that input buffering requires comparable number of delay lines as output buffering architectures at 30% utilization, which is typical for backbone links of network operators, with pacing.

As a future work, we will consider using hybrid optical switch architectures like using OPS and optical circuit switching at the same time. Circuit switching paradigm requires no buffering inside the network as there is no multiplexing. Furthermore, a relatively slow switching fabric can be used in the routers, so circuit switched architectures have much lower cost than packet switched routers. However, circuit switched architectures have lower utilization efficiency due to lack of multiplexing, especially on meshed networks. On the other hand, packet switched architectures can achieve higher utilization due to high level of multiplexing. However, buffer size requirement of packet switching is high for small buffered optical networks. A hybrid packet/circuit switching architecture may relax the traffic and decrease the buffer requirements and the router cost.

Bibliography

[1] R. Takahashi, T. Nakahara, K. Takahata, H. Takenouchi, T. Yasui, N. Kondo, and H. Suzuki, "Photonic random access memory for 40-Gb/s 16-b burst optical packets," *IEEE Photonics Technology Letters*, vol. 16, pp. 1185–1187, 2004.

[2] T. Aoyama, "New generation network(NWGN) beyond NGN in Japan," Web page: http://akari-project.nict.go.jp/document/INFOCOM2007.pdf, 2007.

[3] C. Villamizar and C. Song, "High performance TCP in ANSNET," *Computer Communication Review*, vol. 24, no. 5, pp. 45–60, 1994.

[4] G. Appenzeller, J. Sommers, and N. McKeown, "Sizing router buffers," in *Proceedings of ACM SIGCOMM*, 2004, pp. 281–292.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with very small buffers," *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 83–90, 2005.

[6] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," in *Proceeedings of ACM SIGCOMM*, 1991, pp. 133–147.

[7] G. Theagarajan, S. Ravichandran, and V. Sivaraman, "An experimental study of router buffer sizing for mixed TCP and real-time traffic," in *Proceedings of IEE ICON*, 2006.

[8] V. Sivaraman, H. Elgindy, D. Moreland, and D. Ostry, "Packet pacing in short buffer optical packet switched networks," in *Proceedings of IEEE INFOCOM*, 2006.

[9] Z. Lu, D. K. Hunter, and I. D. Henning, "Contention reduction in core optical packet switches through electronic traffic smoothing and scheduling at the network edge," *IEEE/OSA Journal of Lightwave Technology*, vol. 24, pp. 4828–4837, 2006.

[10] T.-Y. Tung, Y.-J. Chen, and J.-F. Chang, "Design and analysis of RC traffic shaper," *IEICE Transactions on Communications*, vol. E81-B, pp. 1–12, 1998.

[11] H. Zhu, Z. Ma, Z. Cao, and Y. Wang, "Low latency traffic interval shaping algorithm for traffic access control," *Chinese Journal of Electronics*, vol. 11, pp. 247–251, 2002.

[12] O. Alparslan, S. Arakawa, and M. Murata, "Optical rate-based paced XCP for small buffered optical packet switching networks," in *Proceedings of PFLDnet*, February 2006, pp. 117–124.

[13] ——, "Rate-based pacing for small buffered optical packet-switched networks," *Journal of Optical Networking*, vol. 6, no. 9, pp. 1116–1128, September 2007.

[14] ——, "A comparative study of switch architectures for small-buffered optical packet switched networks," IEICE (OPE2006-159), Tech. Rep., January 2007.

[15] ——, "Rate-based paced XCP for small buffered optical packet switched networks," IEICE (PN2006-7), Tech. Rep., May 2006.

[16] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product," in *Proceedings of ACM SIGCOMM*, 2002, pp. 42–49.

[17] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proceedings of ACM SIGCOMM*, 2005, pp. 253–264.

[18] O. Alparslan, S. Arakawa, and M. Murata, "Switch architectures for small-buffered optical packet switched networks," in *Proceedings of 12th IEEE Symposium on Computers and Communications (ISCC'07)*, July 2007.

[19] G. Papadimitriou, C. Papazoglou, and A. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," *Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384–405, 2003.

[20] O. Alparslan, S. Arakawa, and M. Murata, "Performance of paced and non-paced transmission control algorithms in small buffered networks," in *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, June 2006, pp. 115–122.

[21] ——, "Rate-based pacing for optical packet switched networks with very small optical RAM," in *Proceedings of IEEE Broadnets*, September 2007.

[22] ——, "Rate-based pacing for optical packet switched networks with very small optical RAM," *Photonic Network Communications*, 2008, submitted.

[23] ——, "Node pacing for optical packet switching," in *Proceedings of Photonics in Switching*, August 2008, to Appear.

[24] ——, "Node pacing for optical packet switching," IEICE (PN2007-76), Tech. Rep., March 2008.

[25] ——, "Node pacing for optical RAM buffered packet switching networks," *Journal of Optical Networking*, 2008, submitted.

[26] T. Yamaguchi, K. Baba, M. Murata, and K. Kitayama, "Scheduling algorithm with consideration to void space reduction in photonic packet switch," *IEICE Transactions on Communications*, vol. E86-B, pp. 2310–2318, 2003.

[27] D. Careglio, J. Sole-Pareta, and S. Spadaro, "Optical slot size dimensioning in IP/MPLS over OPS networks," in *Proceedings of 7th IEEE International Conference on Telecommunications (ConTEL2003)*, vol. 2, 2003, pp. 759–764.

[28] S. McCanne and S. Floyd, "ns Network Simulator," Web page: http://www.isi.edu/nsnam/ns/, July 2002.

[29] H. Jiang and C. Dovrolis, "Why is the Internet traffic bursty in short time scales?" in *Proceedings of the SIGMETRICS*, 2005, pp. 241–252.

[30] J. Roberts and J. Virtamo, "The superposition of periodic cell arrival streams in an ATM multiplexer," *IEEE Transactions on Communication*, vol. 39, pp. 298–303, 1991.

[31] S. Shalunov and B. Teitelbaum, "TCP use and performance on Internet2," Web page: http://ben.teitelbaum.us/internet2/papers/i2tcp-imeas2001.pdf, 2001.

[32] H. Balakrishnan, N. Dukkipati, N. McKeown, and C. J. Tomlin, "Stability analysis of explicit congestion control protocols," Stanford University Department of Aeronautics and Astronautics Report: SUDAAR 776, 2005.

[33] H. Jiang and C. Dovrolis, "Source-level IP packet bursts: Causes and effects," in *Proceedings of ACM SIGCOMM/Usenix Internet Measurement Conference*, 2003, pp. 301–306.

[34] J. Kulik, R. Coulter, D. Rockwell, and C. Partridge, "A simulation study of paced TCP," BBN, Tech. Rep., 1999.

[35] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of INFOCOM*, 2000, pp. 1157–1165.

[36] S. Floyd, "Highspeed TCP for large congestion windows," RFC 3649, 2003.

[37] A. Razdan, A. Nandan, R. Wang, M. Y. Sanadidi, and M. Gerla, "Enhancing TCP performance in networks with small buffers," in *Proceedings of 11th International Conference on Computer Communications and Networks*, 2002.

[38] S. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proceedings of 6th ACM International Symposium on Mobile Ad hoc Networking and Computing*, 2005, pp. 288–299.

[39] K. Chandrayana, S. Ramakrishnan, B. Sikdar, S. Kalyanaraman, O. Tickoo, and A. Balan, "On randomizing the sending times in TCP and other window based algorithms," RPI ECSE Networks Laboratory, Tech. Rep., 2001.

[40] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley and Sons, Inc., 1991.

[41] L. Li, W. W. D. Alderson, and J. Doyle, "A first-principles approach to understanding the Internet's router-level topology," in *Proceeedings of ACM SIGCOMM*, 2004, pp. 3–14.