EFFECT OF BURST ASSEMBLY OVER TCP PERFORMANCE IN OPTICAL BURST SWITCHING NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Güray Gürel

July, 2006

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ezhan Karaşan (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Nail Akar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. İbrahim Körpeoğlu

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

EFFECT OF BURST ASSEMBLY OVER TCP PERFORMANCE IN OPTICAL BURST SWITCHING NETWORKS

Güray Gürel
M.S. in Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Ezhan Karaşan
July, 2006

Optical Burst Switching (OBS) is proposed as a short-term feasible solution that is capable of efficiently utilizing the optical bandwidth of the future Internet backbone. Performance evaluation of TCP traffic in OBS networks has been under intensive study, as TCP constitutes the majority of Internet traffic. Since burst assembly mechanism is one of the fundamental factors that determine the performance of an OBS network, we focus our attention on burst assembly and specifically, we investigate the influence of the number of burstifiers on TCP performance for an OBS network. We start with a simple OBS network scenario where very large flows are considered and losses resulting from the congestion in the core OBS network are modeled using a burst independent Bernoulli loss model. Then, a background burst traffic is generated in order to create contention at a core node realizing burst-length dependent losses. Finally, simulations are repeated for Internet flows where flow sizes are modeled using a Bounded Pareto distribution. Simulation results show that for an OBS network employing timer-based assembly algorithm, TCP goodput increases as the number of burst assemblers is increased for each loss model. The improvement from one burstifier to moderate number of burst assemblers is significant, but the goodput difference between moderate number of buffers and per-flow aggregation is relatively small, implying that a cost-effective OBS edge switch implementation should use moderate number of assembly buffers per destination. The numerical studies are carried out using nOBS, which is an ns2 based OBS simulation tool, built within this thesis for studying the effects of burst assembly, scheduling and contention resolution algorithms in OBS networks.

Keywords: Optical Burst Switching, Burst Assembly, TCP Performance.

ÖZET

OPTİK ÇOĞUŞUM ANAHTARLAMALI AĞLARDA ÇOĞUŞUM OLUŞUMUNUN TCP PERFORMANSINA ETKİSİ

Güray Gürel
Elektrik Elektronik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Ezhan Karaşan
Temmuz, 2006

Optik Çoğuşum Anahtarlama (OBS), geleceğin İnternet omurgasının yüksek bant genişliğini yüksek verimlilikle kullanabilecek ve kısa vadede uygulanabilir bir çözüm olarak önerilmiştir. İnternet trafiğinin çoğunluğunu oluşturan TCP trafiğinin performans değerlendirmesi, OBS ağlarıyla ilgili yapılan birçok çalışmaya konu olmuştur. Çoğuşum oluşturma mekanizmasının, bir OBS ağının performansına etki eden temel faktörlerin başında yer almasından hareketle tezin geri kalanında çoğuşum oluşturmaya odaklanarak özellikle çoğuşum oluşturucuların sayısının TCP performansı üzerindeki etkisini araştırdık. Optik giriş ve çıkış yönlendiricileri arasında seyahat eden TCP akımlarının alabildiği bant genişligi, değişik TCP versiyonları ve değişik sayıda çoğuşum oluşturucu için gözlemlenmiştir. İlk olarak optik çekirdek ağda çakışmalar sonucu kaybolan çoğuşumların bir Bernoulli kayıp modeliyle temsil edildikleri basit bir OBS ağı izlenmiştir. Bir sonraki aşama olarak arkaplan trafiği içeren daha gerçekçi bir OBS ağında çoğuşumların uzunluğunun performansa olan etkisi incelenmiştir. Son olarak sınırlı Pareto olarak temsil edilen İnternet trafiği altında önceki bulguların geçerliligi denenmiştir. Simulasyon sonuçları, zaman-temelli çoğuşum oluşturan bir OBS ağında, bütün kayıp modelleri için optik çıkış yönlendiricisi başına düşen çoğuşum oluşturucu sayısı arttıkça TCP performansının arttığını göstermektedir. Bir çoğuşum oluşturucudan orta sayıda çoğuşum oluşturucuya çıkıştaki performans artışı anlamlıdır (çoğuşum kayıp oranına, çoğuşum işlem süresi ve kullanılan TCP sürümüne göre %15-%50 civarında). Fakat orta sayıda çoğuşum oluşturucu ile TCP akımı sayısı kadar çoğuşum oluşturucu kullanılan durumdaki performans farkı nispeten azdır. Bu da ederce etkin bir OBS kenar yönlendiricisi uygulamasının orta sayıda çoğuşum oluşturucu içermesi gerektiğine işaret etmektedir. Sayısal analizler, bu tez bünyesinde ns2 üzerine inşa edilmis bir OBS benzetimcisi olan nOBS ile gercekleştirilmiştir.

Anahtar sözcükler: Optik Çoğuşum Anahtarlama, Çoğuşum Oluşturma, TCP Performansı.

Acknowledgement

I gratefully thank my supervisor Assoc. Prof. Dr. Ezhan Karaşan for his supervision and guidance throughout the development of this thesis.

I would like to thank to Dr. Nail Akar and Dr. İbrahim Körpeoğlu for reading the manuscript and commenting on the thesis.

I would like to thank to Onur Alparslan for his efforts in developing the nOBS simulator.

I would like thank to Bilkent Computer Center for allowing me to use their labs.

I hereby express my gratitude to my family for their continuous support.

Contents

List of Figures

List of Tables

List of Acronyms

DWDM	Dense Wavelength Division Multiplexing	1
IP	Internet Protocol	1
MPLS	Multi-protocol Label Switching	1
SONET	Synchronous Optical Networks	1
ATM	Asynchronous Transfer Mode	1
OCS	Optical Circuit Switching	1
OPS	Optical Packet Switching	1
FDLs	Fiber Delay Lines	1
OBS	Optical burst switching	2
JET	just-enough-time	2
TCP	Transmission Control Protocol	3
RTT	round trip time	4
ACK	acknowledgment	4
RTO	Retransmission Timeout	7
MSS	Maximum Segment Size	7
TDA	triple duplicate acknowledgment	8
IWU	interworking unit	9
MPS	maximum payload size	9
FAP	fixed-assembly-period	12
MBMAP	min-burstlength-max-assembly-period	12
AAP	adaptive-assembly-period	12
QoS	Quality of Service	12
DFL	Delayed First Loss	15
RP	Retransmission Penalty	15
LP	Loss Penalty	15
JET	Just-Enough-Time	18
SPN	Share-per-Node	18
LAUC-VF	Latest Available Unused Channel with Void Filling	18
Min-SV	Minimum Starting Void	18

Chapter 1

Introduction

Increasing demand for services with very large bandwidth requirements, e.g. grid networks, facilitates the deployment of optical networking technologies [1]. Using Dense Wavelength Division Multiplexing (DWDM) technology, optical networks are able to meet the huge bandwidth requirements of future Internet Protocol (IP) backbones [2]. Although the total demand is high, individual connections need to use a very small portion of the bandwidth offered by the optical network. Consequently, the evolution of optical technology gained momentum in finding ways of efficient multiplexing access network traffic into optical fiber with as much bandwidth utilization as possible.

One aspect of this evolution is the switching technology employed through the optical network. The Multi-protocol Label Switching (MPLS) protocols enables integration of links of Synchronous Optical Networks (SONET) with Asynchronous Transfer Mode (ATM) cell switches, which provide virtual circuits between IP routers [3]. Recently, IP routers and SONET equipment have evolved to operate together without an ATM switch [3]. In Optical Circuit Switching (OCS), delays during connection establishment and release increase the latency especially for services with small holding times. In addition, as the smallest unit of bandwidth, a wavelength is reserved for the entire duration of the transmission regardless of the rate of the sender. These shortcomings imply that circuit switching is not the optimal switching technology for an optical network carrying IP traffic.

Offering adaptation to changing traffic demands and avoiding the need for reservations, Optical Packet Switching (OPS) becomes a candidate for providing all-optical packet switching for the future Internet backbone. However, optical buffering and signal processing technologies have not matured enough for possible deployment of OPS in core networks in the near future. When an optical packet is processed, it needs to be converted back into electrical domain. These conversions and processing in electrical domain constitute a bottleneck for the optical connection. Ideally, if the whole operation could be done optically, then the bandwidth and speed offered by the optical domain could be fully utilized. OPS research aiming near-term feasibility focuses on electronic control and processing of packet header [4]. In this case, electrical conversion is applied only to the header, which contains routing information, and it is thereafter processed so that the optical switch could be set up for the optical payload following the header. There is a guard band between the header and the payload to account for this processing time. An OPS network can be slotted (synchronous), where packets of constant size are aligned, or unslotted (asynchronous), where packets may be of variable size and have a larger contention probability [5]. Several IP packets may be aggregated to construct an optical packet at edge nodes. The lack of optical buffering may be overcome by the use of electronic buffering for large packets and Fiber Delay Lines (FDLs) for small packets [4]. An FDL is a very long optical fiber to provide fixed amount of delay. Nevertheless, OPS will have to wait for the availability of optical buffering and optical processing to work ideally.

Optical burst switching (OBS) is proposed as a short-term feasible technology that can combine the strengths and avoid the shortcomings of OCS and OPS [6]. Figure 1.1 depicts a typical OBS network. When packets from IP routers reach the edge router, they are aggregated into a larger entity called *burst*. Bursts wait in electronic buffers at the edge router until they are ready to be sent into the optical domain. Some of the wavelengths are reserved for control packets, which include routing, arrival and length information for the bursts. A control packet corresponding to a burst is sent an *offset* time before the burst to account for the processing at the core OBS nodes. When a core node receives a control packet, it

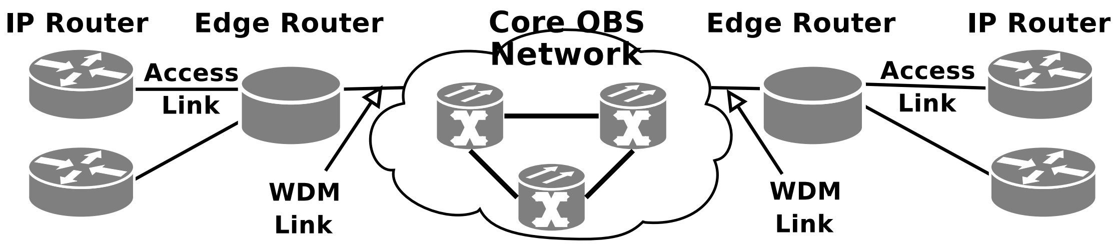


Figure 1.1: An OBS network

converts a copy of the packet to electrical domain and looks whether the switch is idle during the desired reservation interval for the corresponding burst. If it is, then the reservation is made so as to deny possible reservations that may request an overlapping time interval. This is an indication for a future contention at the switch and knowing this beforehand gives the core node enough time to choose between contention resolution mechanisms, e.g. wavelength conversion, deflection routing, FDL usage, dropping the overlapping section from one of the bursts, preemption or just dropping the contending burst. Sending the control packet and then sending the burst without waiting for the response is known as *one-way reservation* and is implemented in many reservation protocols such as just-enough-time (JET) [7]. An OBS network employing JET makes reservations just for the duration of the burst and underutilization due to guard bands as in OPS is avoided. Reservations are only made when the ingress edge router has data to send as opposed to the reservation in circuit switching where the channel is reserved for the whole duration of the transmission. Using reservations enables the control circuitry at the core nodes to prepare before the burst reaches the node. Aggregating IP packets into bursts leads to efficient bandwidth utilization. These superiorities and short-term feasibility make OBS a better alternative compared to circuit switching and OPS.

Performance evaluation of Transmission Control Protocol (TCP) flows in OBS networks has been under intensive study, since TCP constitutes the majority of Internet traffic. As the fundamental factor that determines how TCP traffic is shaped into optical bursts, the burst assembly mechanism may provide valuable improvements in terms of its effects on TCP throughput and therefore constitutes the focus of this thesis.

The need for assembly arises from two properties of OBS networks. First, there is a minimum time required for an optical switch to be configured before an optical packet can pass through it. Secondly, control information is carried through additional headers which become an overhead to the system. When we aggregate packets into bursts, the amount of switching time and overhead per unit amount of application level data decrease resulting in higher bandwidth utilization.

When a packet needs to travel through an OBS network, it is first received by an ingress router (Figure 1.1). The ingress node contains electrical buffers where the packets are aggregated into bursts before they are sent into the optical link. Once a packet is formed, it is not possible to extract packets from the burst before it reaches the egress router. Therefore, the ingress router should have at least one aggregation buffer for each egress router and packets are classified into aggregation buffers according to their destination egress nodes.

Based on the assembly algorithm, the ingress router keeps track of the delay experienced by the first packet in an aggregation buffer and/or the size of the buffer. In the timer-based assembly, a burst is formed when the delay of the first packet reaches a given timeout. Size-based assembly forms bursts when the size of the buffer reaches a threshold. For the hybrid algorithm, either condition results in a burst.

The TCP side of the problem involves the TCP congestion control scheme, which has been explained clearly in [8]. Briefly, a TCP receiver acknowledges the reception of a segment by notifying the sender about the sequence number of the next in-order byte expected. The sender adjusts its rate using two values, namely `CongWin` and `RcvWindow` . The difference in the sequence numbers of the byte to be sent and the byte that has most recently been acknowledged by the receiver cannot exceed the minimum of these two windows. Another important parameter is the round trip time (RTT), which is defined as the time from the transmission of a segment until the reception of its acknowledgment (ACK). Typically, RTT is larger than the transmission time of `CongWin` bytes of data and if we assume that `RcvWindow` is relatively large, then TCP rate adjustment simplifies to the case where the sender sends `CongWin` amount of data in each RTT.

The way packets are assembled affects TCP sender's perception of end-to-end delay and optimal transmission rate. As a result of burstification, segments from many TCP flows are put into a burst that may be successfully delivered or may be dropped due to a contention in the core network. When a burst is dropped, a TCP sender that has segments in the burst experiences a timeout or receives acknowledgments requesting an already transmitted segment. The sender interprets this situation as congestion in the network and reduces its transmission rate. The level of congestion perceived by the sender depends on the number of sender's segments contained in the burst. A burst drop affecting multiple flows implies a synchronous throughput reduction in a large number of flows. To sum up, the implementation of the assembly mechanism, e.g. choice of parameters, number of segments from individual flows, amount of additional delay, etc., is important for proper utilization of optical bandwidth. Many studies examine the burst assembly mechanism and offer ways for better performance, but they overlook the significance of the number of flows sharing an aggregator and there is still room for considerable improvement.

In this thesis, we use an ns2 based [9] simulation tool (nOBS) [10] to evaluate the performance of several TCP versions with respect to burst assembly parameters. nOBS implements various burst assembly, scheduling and routing algorithms and is developed to examine burstification, scheduling, contention resolution algorithms and their effects on TCP performance. nOBS allows selection of the number of aggregators per egress nodes, or equivalently number of flows sharing an aggregator. We simulated TCP performance for a wide range of assembly parameters, number of aggregators and network models using nOBS.

First, a single fiber optical network with Bernoulli loss model is simulated. The behavior of TCP goodput is observed over various parameter ranges and what seems to be contradictory results of previous studies turn out to be the parts of a bigger picture. Also the effects of the mechanisms used to explain the TCP performance, such as *delay penalty* or *delayed first loss gain*, are validated. Contrary to common usage, where single aggregation buffer per egress router is used, we employ multiple buffers per egress router and show that the level of synchronization between TCP flows destined to an egress node decreases as we increase the number of aggregation buffers per egress router. Our results indicate that using moderate number of buffers, it is possible to reach 15-50% performance improvement. This implies a cost effective solution that comprises the ingress router complexity versus improved bandwidth utilization.

Secondly, a simple optical network topology with Poisson background burst traffic is simulated to see the distribution of burst loss probability versus burst length. As in the previous case, TCP flows are generated by infinite sized FTP traffic. It is seen that despite previous assumptions about burst loss probability being independent of burst size, burst loss probability actually increases with the length of the burst. The effect of number of assembly buffers per egress node is also confirmed by the results of these set of simulations.

Finally, the latter network is simulated again, but instead of TCP flows carrying infinite FTP data, we used TCP flows with Poisson arrivals and bounded Pareto flow lengths to understand the behavior of Internet traffic. The results were similar to those of the previous simulations.

The organization of the thesis is as follows: in Chapter 2, related work is presented. The nOBS simulator is explained in Chapter 3. The network model and simulation results for burst size independent and burst size dependent loss models are presented in Chapters 4 and 5, respectively. The conclusions of the thesis is presented in Chapter 6.

Chapter 2

Burst Assembly of TCP Traffic in OBS Networks

The need for assembly first emerged in OPS networks. Size-based assembly has been employed by OPS networks and is also adopted later in the proposal of OBS networks. In addition, OBS networks enabled the use of timer-based and hybrid size/timer-based assembly. In this chapter, we first present some TCP basics related to TCP performance. Then, the concepts of size-based, hybrid size/timer-based and timer-based assembly is described. Finally, the chapter concludes with the examination of the attempts made to name the factors that affect TCP performance in the burst assembly mechanism.

2.1 TCP Basics

The TCP congestion control scheme is clearly explained in [8]. It is usually the case that the sender sends `CongWin` amount of data in each RTT. In other words, the size of the congestion window together with the end-to-end delay determine the instantaneous transmission rate of the sender. The end-to-end delay is affected by the additional assembly time, while the size of the congestion window depends on the reception of acknowledgments.

A *timeout* occurs when the sender does not receive any acknowledgments within Retransmission Timeout (RTO). The RTO value is computed by the sender based on estimated RTT and estimated deviation on RTT. On the start of a TCP connection and after a timeout, the sender is in *slow start* phase and the value of `CongWin` is set to one Maximum Segment Size (MSS). In this phase, the sender increases `CongWin` by 1 for every acknowledged segment until `CongWin` reaches `Threshold` . In other words, size of the congestion window, i.e. `CongWin` , is doubled for every successfully acknowledged window. When `CongWin` reaches `Threshold` , the sender switches to *congestion avoidance* phase, where the size of the congestion window is increased by 1/`CongWin` for every acknowledged segment, or in other words `CongWin` is incremented by 1 for every successfully acknowledged window.

An acknowledgment for an already acknowledged segment, i.e. an ACK indicating that receiver is still expecting the same in-order segment, is called a *duplicate acknowledgment*. A duplicate acknowledgment tells the sender that either there is reordering through the network, or there is loss of some segments from the window. Upon the reception of the third duplicate acknowledgment, the sender decides that the network is congested, but not as heavily as in the timeout case. How triple duplicate acknowledgment (TDA) is treated depends on the TCP version.

When a TDA occurs, `CongWin` is halved, threshold is set to `CongWin` and the phase is switched to congestion avoidance for TCP Reno whereas TCP Tahoe treats a TDA equally with a timeout event [8]. TCP Sack (TCP with selective acknowledgments) uses the same scheme to change `CongWin` as TCP Reno, but in addition, option field is used to indicate the portion of the sender's window that has been correctly received by the receiver [11]. TCP Newreno differs from TCP Reno by its reaction to multiple segment losses from a window. When multiple packets from a window are lost, TCP Reno will halve its congestion window size for every TDA and eventually reach timeout, whereas TCP Newreno transmits one lost packet for every ACK indicating the next lost packet and hence avoids timeout [11].

2.2 Size-based Assembly

Detailed analysis of Internet traffic showed that IP traffic is bursty and its packet length has a distribution with peaks at 40, 576 and 1500 bytes [12, 13]. The self-similar traffic pattern and the diverse packet size distribution significantly reduce the effectiveness of common contention resolution schemes of OPS for high loads [12, 14]. The traffic shaping function of packet aggregation at the edge routers and its improvements on OPS performance have been noted by [12, 13]. The basics of OPS packet assembly is similar to those of the OBS burst assembly and the results obtained for OPS packet assembly can be extended to the OBS aggregation case and vice versa in a qualitative manner.

The interworking unit (IWU) is responsible from packet assembly in each edge OPS router. As the initial principles of OPS relied on synchronous mode of operation and fixed packet size [12], the two design parameters of IWU turn out to be the maximum payload size (MPS) and an *assembly timeout*. If IP packets are larger than MPS, they are fragmented. If they are shorter, they are aggregated into an optical packet of size MPS. If the MPS requirement is not fulfilled for a *timeout* duration, the payload is padded up to MPS and sent into the optical network to avoid excessive queuing delays. This scheme, which is also used in OBS studies, will hereafter be referred to as size-based assembly.

The effects of size-based assembly algorithm over TCP performance in OPS networks have been observed through simulations in [3]. For different values of MPS, the timeout value is also changed accordingly. It is shown that for average transmitter loads greater than 20%, aggregation improves TCP performance, but using larger values of MPS yields poorer performance as a result of the additional queuing delay.

Size-based assembly has also been studied by [15]. The process of padding the optical packet up to MPS when timeout expires brings forth the necessity to introduce packetization efficiency, which is defined as the ratio of data bits to the payload size. The study examines the trade-off between packetization efficiency and packetization delay. According to simulations driven by self-similar traffic, it is seen that small values of timeout causes the packetization efficiency to decrease with increasing MPS. The incoming traffic rate is not enough to fill the MPS-sized optical packet for small timeouts, therefore increasing MPS just increases the number of padded bits and decreases efficiency. For a larger timeout, packetization efficiency first increases with increasing MPS, but starts to decrease after some MPS value. For the largest timeout, packetization efficiency increases in a saturating manner with increasing MPS and gets very close to 1. Although not mentioned in the text, these results indicate that there are regions in the chosen parameter ranges, some of where the timeout is the effective threshold, while for others, the effective threshold is MPS. As another observation, packetization delay is shown to decrease with increasing MPS. Packetization delay increases with increasing timeout. TCP throughput is shown to increase with increasing MPS and increasing timeout. It is also worth to note that packets belonging to the same congestion window are not put together in the same optical packet [15], but no such limitation is present for OBS burst assembly.

Packet aggregation in an OPS network is shown to improve TCP throughput in [16] and it is noted that the improvement increases with optical packet size. *Full aggregation*, which is the aggregation of packets destined to the same egress node in the same optical packet, *per-class* and *per-flow* aggregation schemes are compared from throughput and fairness aspects. Without ingress buffering, the flow-based aggregation is found to give the worst performance as random arrivals of large packets from many aggregation queues to the optical switch implies higher contention probability. Flow-based aggregation may further cause synchronization of flows, in which case packets from some flows are always favored over others. In other words, per-flow aggregation degrades TCP fairness.

The impact of the size-based assembly on TCP throughput in OBS networks constitutes the focus of [17]. Channel utilization improves when larger bursts are used, but increasing burst sizes reduces the efficiency of FDLs, increases end-to-end delays and increases the synchronization between TCP sources whose packets share dropped bursts. This would mean simultaneous decrease of congestion windows of many TCP sources. Using analytical models, the optimal burst size is found to depend on the size of the guard bands between data bursts, FDL lengths, number of TCP sessions, optical channel bandwidth, RTT and average packet size.

2.3 Hybrid size/timer-based algorithm

Apart from size-based algorithm, hybrid size/timer-based algorithm is also used in the analysis of OBS networks. The size of the packet and the delay experienced by the first IP packet in the aggregation buffer is tracked and checked against time and size thresholds. Either the size of the buffer reaching the size threshold or the delay of the first packet reaching timeout causes the generation of a burst. TCP performance in OBS networks with hybrid size/timer-based algorithm is evaluated in [18]. It is noted that TCP reacts to packet drops, end-to-end delay changes and throughput changes. When a burst is dropped, all the TCP sessions having packets in that burst react to the loss event and cause a network wide drop in throughput. Burstification (burst assembly) is triggered when the burst reaches size threshold for high input traffic rates, while assembly timeout becomes the effective threshold for low input traffic rates. The granularity of FDLs also affect the TCP performance in an OBS network. Increasing the burst size increases TCP throughput. Another observation is that TCP sessions that are slower in rate reach their maximum throughput at relatively smaller burst sizes. End-to-end delay increases with increasing burst size threshold as well as increasing assembly timeout. TCP throughput is seen to deteriorate with increasing assembly timeout for low drop probabilities, but no significant change is observed for higher loss probabilities. It is noted that fewer bursts are produced when the burst size is increased resulting in less number of drops. From this expression, it is understood that uniform burst loss model is assumed in this study. The need for a new metric to achieve high goodput while experiencing acceptable delay is pointed out and throughput/delay is given as an example for such a metric.

2.4 Timer-based assembly

Timer-based assembly mechanism for OBS networks is proposed by [19]. Whenever the delay experienced by the first packet in an assembly queue reaches the assembly timeout, the burst is queued for transmission. If the burst is smaller than the minimum burst length, it is padded up to the minimum burst length. This algorithm limits the burst assembly delay. It also shapes self-similar internet traffic so that improved queuing performance is obtained.

The timer-based and hybrid size/timer-based assembly algorithms have been rediscovered by [20] as fixed-assembly-period (FAP) and min-burstlength-max-assembly-period (MBMAP) algorithms, respectively. In addition, the adaptive-assembly-period (AAP) algorithm is proposed. Similar to the calculation of RTT [8], average burst length is obtained and divided by the bandwidth to get the time required to transmit the average-length burst. Multiplication of this value with the assembly factor α, which is greater than 1, yields the new assembly timeout. The OPS and OBS performance in terms of goodput have been compared and it is shown that timer-based OBS assembly performs better than size-based OPS assembly. In comparison of the goodput of the three assembly algorithms, it is claimed that the hybrid size/timer-based algorithm achieves as good performance as the timer-based algorithm for most of the cases. It is said that the adaptive algorithm performs better than timer-based algorithm while the figures show minuscule improvement.

Another adaptive algorithm has been presented by [21]. Intuitively, bursts with larger offsets have greater probability of success in making reservations. This principle is used in many studies about Quality of Service (QoS) to ensure a minimum bandwidth for a class of packets by assigning their bursts with offsets larger than what is used for the rest of the bursts. In the study, the variation of burst size is pointed out as a factor that forces larger offsets to ensure QoS. Using larger offsets increase the end-to-end delay experienced by the packets. Therefore, reducing the variation in the burst size comes up as a desirable property of a burst assembly algorithm. Timer-based assembly, however, creates bursts with a high variation of size. Another disadvantage of timer-based assembly is the continuous blocking problem. When two ingress nodes with same timeout value contend at a core router, the control packets produced by these nodes will have the same time difference. The contention will always be resolved in favor of the burst whose control packet arrives early. As the time difference is constant, this means that the bursts produced by an ingress router are always favored against the bursts produced by the other ingress node in case of a contention. Another desirable property that a burst assembly algorithm should have is to avoid the continuous blocking, which can be achieved through the use of an adaptive timer. In the proposed adaptive algorithm, the packets are aggregated into a FIFO assembly buffer and the size of the queue is compared against predetermined Q_{low} and Q_{high}. Queue sizes smaller than Q_{low} results in decrements in so-called cross-over count, and queue sizes larger than Q_{high} causes the cross-over count to be incremented. Successive increments/decrements causes the algorithm to increase/reduce Q_{low}, Q_{high} and $burstsize$ parameters. This algorithm is claimed to adapt to changing traffic demands and reduce the variation in burst sizes, however, the specifics as to how the algorithm is implemented remain shallow, e.g. when the algorithm is executed (on packet receptions or on periodic timeouts) or how the Q_{high} and Q_{low} should be chosen with respect to average burst size are not mentioned.

2.5 Impact of Burst Assembly on TCP Traffic

In this section, various factors that affect the performance of TCP traffic in OBS networks are discussed.

2.5.1 Delay Penalty and Correlation Gain

The first study that attempts a thorough analysis of the impact of the burstification process by naming the factors that affect TCP performance is [22]. One of the effects of burstification is the increase in RTT and RTO values as a result of the addition of assembly delay and consequent deterioration in TCP performance as also noted by previous studies. The degradation of TCP performance as a result of assembly delay is called *delay penalty.* Another important effect of burstification is the combined successful delivery or combined loss of the packets contained in a burst. In other words, even for statistically independent burst loss events, the packet loss events are highly time correlated. The impact of this correlation on TCP performance is called the *correlation benefit.* As the level of correlation depends on how many packets a burst contains from a particular TCP flow, it is necessary to differentiate TCP sources as *slow*, which have 1 packet from their congestion windows in a given burst, *fast*, which have their entire congestion windows in a given burst, and *medium* sources, which have a portion of their congestion windows in the given burst. The relationship between the assembly timeout T_b, maximum congestion window size W_m, segment size, L(bits), and access bandwidth, B_a(bps), is given as the following:

$$\text{Fast sources} \qquad \frac{W_m.L}{B_a} \leq T_b \tag{2.1}$$

$$\text{Slow sources} \qquad \frac{L}{B_a} \geq T_b \tag{2.2}$$

$$\text{Medium sources} \qquad \frac{L}{B_a} < T_b < \frac{W_m.L}{B_a} \tag{2.3}$$

For a fast TCP source, when the burst containing the congestion window is lost, as no acknowledgments will be received from the TCP destination, RTO will cause the congestion window to drop to 1 and TCP sender will switch to the slow start phase. For the bursts that are not dropped, the acknowledgments for the whole window will cause the congestion window size to be quickly restored to a value close to its maximum (W_m). When a burst is lost containing the packet from a slow source, the loss of this single packet is recovered using the *fast recovery* and *fast retransmit* by TCP Reno, which is the version analyzed in [22]. TCP Reno throughput of slow and fast sources in OBS networks are expressed in terms of RTT, including the assembly time, and burst loss probability, p. Simulation results are shown to coincide with the analytical models. The correlation benefit, C_b, is expressed as:

$$C_b = F.D_p \text{ , where } F = \frac{B}{NB} \quad \text{and} \quad D_p = \frac{RTT}{RTT_0}. \tag{2.4}$$

Here, the *burstification factor*, F, is defined as the ratio of the TCP send rate with and without aggregation and RTT_0 denotes the round trip time in the absence of the assembly time. In other words, the correlation benefit is defined as the TCP rate improvement caused by aggregation without the effect of additional assembly delay. It is noted that correlation benefit is maximized with $p = 1/W_m$ for fast sources while it is constant at 1 with respect to p for slow sources. Its value lies in between these two for medium sources. Increasing B_a increases burstification factor for medium sources, but it does not affect slow or fast sources. In addition, increasing assembly timeout, T_b, increases the segments per burst for medium sources and consequently increases burstification factor.

2.5.2 Delayed First Loss and Retransmission Penalty

The analysis of factors that determine how the burst assembly mechanism affects TCP throughput is studied further in [11], where *correlation benefit* is divided into two sub-factors as the Delayed First Loss (DFL) gain and Retransmission Penalty (RP). Retransmission penalty occurs as a result of the increase in transmission time for retransmitting the lost segments. Delayed first loss is the delay in time before a TCP sender receives indication for a lost segment. This delay causes the congestion window reach to higher values and in result, the sender achieves a higher throughput. A third factor called Loss Penalty (LP) is introduced, which is defined as the throughput reduction as a result of a lost burst. In terms of TCP throughput, B, the number of segments in a burst that are from a particular flow, S, burst loss rate, p, round trip time without assembly, RTT_0, assembly timeout, T_b, maximum window size, W_m and the number of ACKed rounds before the sending window size is increased, b :

$$\text{LP Ratio} = \frac{B(\text{with no loss})}{B(\text{with a burst loss rate p})} \approx W_m\sqrt{\frac{2bp}{3S}} \quad \text{for small p.} \tag{2.5}$$

$$\text{DP Ratio} = \frac{B(\text{with } RTT_0)}{B(\text{with RTT})} \approx \frac{RTT_0 + 2T_b}{RTT_0} \tag{2.6}$$

$$\text{DFL Gain Ratio} = \frac{B(\text{the first loss is delayed})}{B(\text{the first loss is not delayed})} \approx \sqrt{S} \quad \text{for small p.} \tag{2.7}$$

$$\text{RP Ratio} = \frac{B(1 \text{ retransmission})}{B(\text{S retransmissions})} \approx 1+\sqrt{\frac{3Sp}{2b}} \quad \text{small p, large S, Newreno} \quad (2.8)$$

$$\text{RP Ratio} = \frac{B(1 \text{ retransmission})}{B(\text{S retransmissions})} \approx \sqrt{3}(1+\sqrt{\frac{Sp}{2b}}) \quad \text{small p, large S, Reno} \quad (2.9)$$

The additional T_b in (2.6) compared to the D_p value in (2.4) comes from the fact that unlike [22], the ACK segments are burstified in [11]. Given the above ratios, the optimal assembly time is defined as:

$$T_b^{opt} = \arg\max_{T_b}\{\frac{\text{DFL Gain}}{\text{RP} \times \text{DP}}\} \quad (2.10)$$

According to the simulation results presented in the study, TCP throughput first increases then decreases as the assembly time threshold is increased for medium and fast sources. For slow sources, however, the throughput always decreases. When the TCP version performance is compared, it is seen that for relatively low burst loss probabilities, Sack performance is the best, followed by Newreno, and Reno performs the worst. When the loss probability is increased, all versions perform very close to each other.

2.5.3 Burst Size and Interarrival Statistics

In [23], the sizes of the bursts produced by a timer-based algorithm is shown to approximate a gaussian distribution. Added to that, the burst interarrival time distribution for a size-based algorithm is more closely modelled with a gaussian distribution compared to poisson burst arrivals. Unlike [22], this study argues that burst assembly does not change the long range dependency of the Internet traffic. It is shown that timer-based assembly performs better than size-based assembly and it is noted that the performance of the hybrid size/timer-based algorithm should be in between the performances of these two algorithms. The concepts of delay penalty, loss penalty, retransmission penalty and delayed first loss gain are revisited. It is claimed that the performance of Newreno TCP should be the poorest compared to Sack and Reno, because Newreno transmits 1 lost segment in each round, while Sack quickly retransmits lost segments using selective acknowledgments and Reno quickly restores congestion window size with slow start after reaching timeout as a result of continuously halving congestion window.

2.5.4 Effect of TCP Version

The comparison of the performance of TCP implementations in OBS networks is studied in [24]. When a burst containing the whole congestion window of a TCP flow, i.e. a fast flow, is lost, TCP Reno, Newreno and Sack all react with a timeout as RTO expires. If the burst contains just 1 segment from a flow, i.e. a slow flow, all three versions halve their congestion windows and retransmit the lost segment while switching to congestion avoidance phase. However, if the dropped burst contains part of the congestion window of a flow, i.e., a medium flow, then each TCP version behaves differently. As the Reno sender keeps receiving TDAs for the segments in the lost burst, the congestion window will be halved for each TDA. If congestion window size drops to 3 or below, than the sender will not receive triple-duplicate-ACKs and with the expiration of RTO, Reno sender resets to a congestion window size of 1 in slow start phase. On the other hand, a Newreno source transmits 1 lost segment in each round until the whole segments in the lost burst are retransmitted. Sack uses selective acknowledgments and quickly retransmits the segments in a few rounds. The performance of Sack is noted to be better than the performances of Reno and Newreno, but the paper proposes a new TCP version, Burst TCP, to avoid false timeouts and shows performance improvements in OBS networks with respect to other TCP versions.

In this chapter, we summarized previous work related to burst assembly and its effects on TCP performance. Before moving on to our results about burst assembly, we first introduce the nOBS simulator used in this thesis. In the next chapter, components of the simulator are presented, the implementation of OBS router functionalities are described and the ingress node model is given.

Chapter 3

nOBS: an OBS Simulator for TCP Traffic

Figure 3.1 depicts an OBS network from a TCP sender's point of view. TCP segments are routed by IP routers through electrical access links to an ingress router, where they are aggregated into a burst. The burst waits in electrical buffers until it is scheduled on an available wavelength. Then it traverses through a group of optical core routers to reach the egress router. At this point, the topology of the optical core network is ignored and modelled as a cloud. The egress router takes out each individual IP packet and routes them to the TCP receiver through electrical access links. The reverse path, which carries the acknowledgments from the receiver is not shown for the sake of simplicity.

A simulator that is built for analyzing the effects of various OBS mechanisms on TCP performance must ensure reliable TCP simulations. Therefore, a reliable and publicly available TCP simulator, ns2 [9] (version 2.27), is chosen as the basis for nOBS. ns2 provides implementations of different TCP versions, electrical and satellite links, unicast and multicast nodes, applications and traffic generators and many other useful components that can be used to simulate a large range of scenarios. Nevertheless, it does not support optical elements required for OBS simulations.

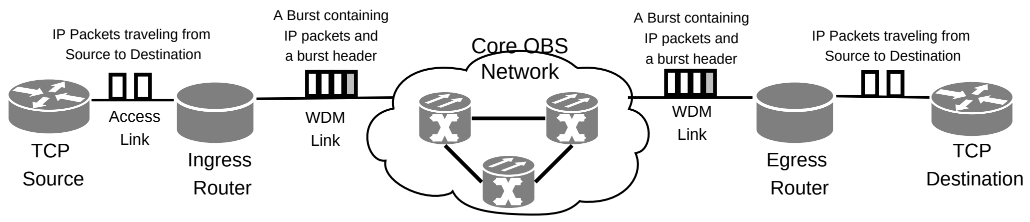

IP Packets traveling from Source to Destination
A Burst containing IP packets and a burst header
Core OBS Network
A Burst containing IP packets and a burst header
IP Packets traveling from Source to Destination
Access Link
WDM Link
WDM Link
TCP Source
Ingress Router
Egress Router
TCP Destination

Figure 3.1: A simple OBS network

nOBS extends ns2 components and defines new classes to introduce the optical domain. Ingress, core and egress node functionalities are combined into the nOBS optical node on top of the ns2 node object. The edge nodes of an OBS network, i.e., ingress and egress nodes, fulfill the burstification and deburstification functions. The optical node architecture in nOBS allows users to specify the parameters of the burst aggregation algorithm as well as how packets belonging to different TCP flows that are forwarded to the same egress node, are mapped into burstifiers. The edge nodes are also responsible for generating and transmitting the burst control packet, which corresponds to the burst header. The control packet has all the necessary information so that each intermediate optical switch in the core OBS network can schedule the data burst and also configure its switching matrix in order to switch the burst optically. nOBS uses the Just-Enough-Time (JET) reservation protocol [7], where the edge node transmits the optical burst after an offset time following the transmission of the control packet. In JET, the control packet tries to reserve resources for the burst just sufficient enough for transmission of the burst on each link it traverses. The core nodes in nOBS perform the scheduling function using wavelength converters and FDLs, if necessary. In nOBS, the wavelength converters and FDLs are combined into pools that are shared among all ports. This sharing architecture is called Share-per-Node (SPN), which achieves the best loss performance among other sharing architectures [25]. The user can specify the number of FDLs and wavelength converters in the pools at each node. The scheduling algorithms that are currently implemented in nOBS are Latest Available Unused Channel with Void Filling (LAUC-VF) [26] and Minimum Starting Void (Min-SV) [27].

The architecture of an OBS node in nOBS is shown in Figure 3.2. The BurstAgent class is responsible from aggregation of incoming IP packets into

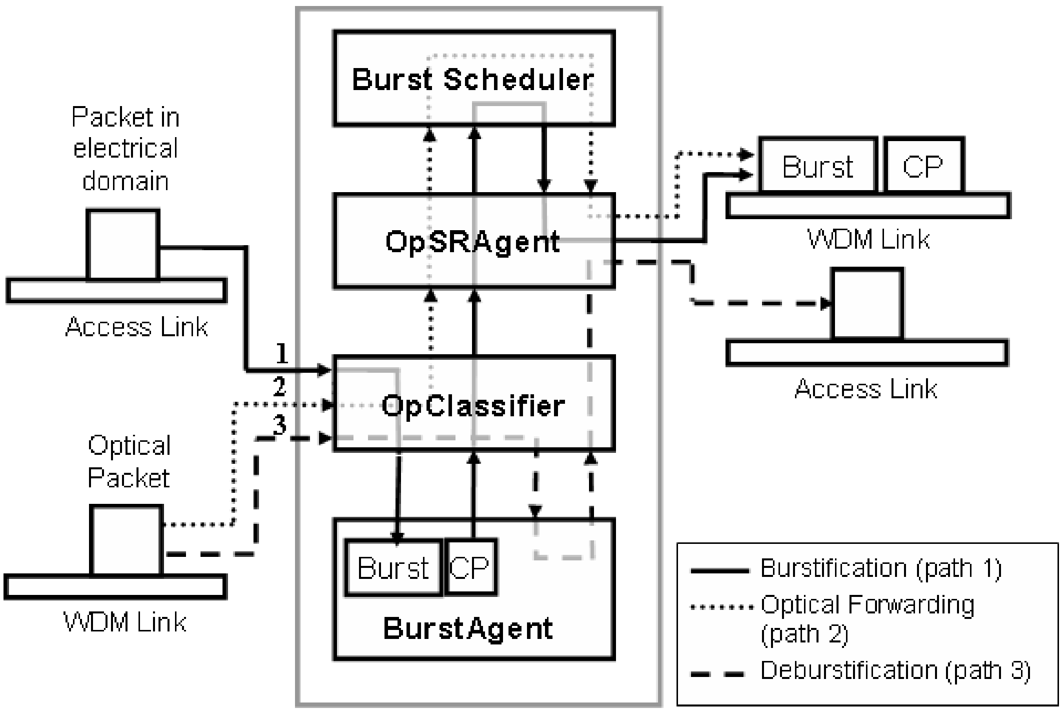


Figure 3.2: Optical node architecture in nOBS

assembly buffers and producing bursts. An optical source routing agent, OpSRAgent, is developed to provide separate layer of routing through the optical network. OpSRAgent is also responsible from writing source routing information to packet headers, checking the optical schedulers to see whether aggregated bursts or incoming control packets can have successful reservations. Optical classifier, OpClassifier, is responsible from delivery and forwarding of packets to the corresponding optical components. In Figure 3.2, ingress, core and egress node functionalities are indicated by paths 1, 2 and 3 respectively.

The process of burstification (path 1) starts with a packet in electrical domain arriving at the optical node through an access link. This packet is first processed by Optical Classifier (OpClassifier). Upon seeing that the next hop for this packet is in the optical domain, OpClassifier forwards the packet to the Burst Agent (BurstAgent). BurstAgent puts the packet in an assembly buffer that corresponds to a burst and control packet pair. When a burst is ready for transmission, its associated control packet is sent to OpClassifier and then forwarded to Optical Source Routing Agent (OpSRAgent). OpSRAgent puts the optical domain routing information into the control packet and the corresponding burst. It then checks for a suitable interval through the Burst Scheduler block. This block includes OpSchedule, OpConverterSchedule and OpticalFDLSchedule, which keep records of the reservations on outgoing channels, wavelength converters and FDLs, respectively. If a suitable interval is found, OpSRAgent sends the control packet and schedules the burst to be transmitted after an offset time. Otherwise, the burst is dropped.

OpSRAgent is basically an ns2 source routing agent improved to handle optical packets. When the simulation scenario is described in the TCL code, all nodes (electrical or optical) are commanded to install an OpSRAgent instance and routes for each node to all possible destinations are determined using the minimum hop routing. In all nodes, newly created packets are sent to OpSRAgent, which writes the path that will be used by the packet in the packet header. In other words, if an application running on ingress router produces data to be sent into the OBS network, the burstification path starts with OpSRAgent, where the route information for the packet is written, followed by the OpClassifier which will forward the packet to the BurstAgent.

In the case of optical forwarding (path 2), an optical packet is received by the OpClassifier through an incoming WDM link. Since the next hop is in the optical domain, OpClassifier forwards the packet to the OpSRAgent, which queries the Burst Scheduler block for a valid reservation. If the optical packet is a control packet and a reservation for the associated burst is possible, then the control packet is forwarded to the corresponding WDM link. If the optical packet is a burst and a reservation has been already made, the burst is forwarded to the WDM link. Otherwise, the optical packet is dropped.

When the next hop for an optical packet is not in the optical domain, OpClassifier sends this optical packet to the BurstAgent for deburstification (path 3). If the optical packet is a control packet, it is dropped. If it is a burst, then the packets inside the burst are sent to the OpClassifier, which forwards them to OpSRAgent. OpSRAgent sends these packets through outgoing electrical links towards their destination nodes.

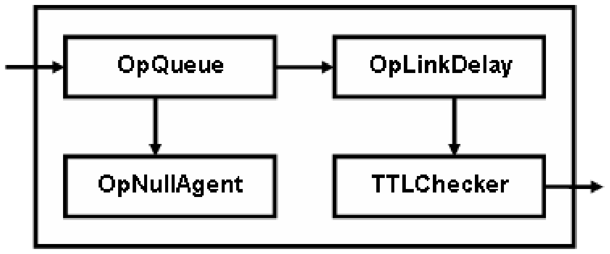


Figure 3.3: WDM link architecture in nOBS

The architecture of an optical link in nOBS is shown in Figure 3.3. This structure is based on the existing ns2 link configuration. Instead of the store-and-forwarding scheme of packet switched networks implemented in ns2, cut-through forwarding is applied. When the loss model associated with the link determines that an optical packet must be dropped, the packet is sent to OpNullAgent component, which frees individual packets inside the burst.

The main components of nOBS, the classifier, the burst agent, the source routing agent and the optical schedulers, are described below in more detail.

3.1 OpClassifier

A new classifier called OpClassifier is implemented in nOBS for classifying and forwarding packets inside optical nodes. The id numbers of optical nodes in the same domain as this node are given to OpClassifier in a TCL script by using the command *optic_nodes* and stored in a table called *opticnodes*. Therefore, OpClassifier knows the nodes that are in the same OBS domain. When a packet arrives to OpClassifier, OpClassifier checks the type and destination of the incoming packet and handles the packet as follows:

- If the incoming packet is not an optical burst and the packet's destination address is not this node, OpClassifier checks the source routing table of the packet. Looking up in the routing table of the packet, OpClassifier checks whether the packet's next node is in *opticnodes*. If it is, the packet needs to enter the OBS domain, furthermore the node that owns this OpClassifier should act as an ingress node and apply burstification. Therefore, OpClassifier forwards this packet to the burstifier agent called BurstAgent. Otherwise, OpClassifier realizes that this packet is coming from the BurstAgent after the deburstification process. In this case, the packet is leaving the OBS domain, so OpClassifier forwards this packet to the source routing agent that will forward the packet to the next hop over an electronic link.

- If the packet is an optical burst and the packet's destination address is this node, it means that a burst has reached its destination. OpClassifier forwards the packet to the BurstAgent for the deburstification process.
- If the packet is an optical burst and the packet's destination address is not this node, it means that this is a burst in transit. Therefore, OpClassifier forwards this packet to the source routing agent that will forward it to the next hop which is specified in the source routing table of the packet.
- If the packet is not an optical burst and the packet's destination address is this node, it means that the packet is coming from the BurstAgent after deburstification process and the receiver of this packet is in this node. OpClassifier forwards this packet to the port classifier, which will forward the packet to its destination agent.

3.2 BurstAgent

BurstAgent is responsible for the burstification of electronic packets and deburstification of optical bursts. A single BurstAgent is attached to OpClassifier in each optical node. When a new packet arrives from OpClassifier, BurstAgent checks whether this packet is an electronic packet or an optical burst. If the packet received from OpClassifier is an optical burst, BurstAgent disassembles the IP packets inside the payload of the burst and sends these IP packets back to the OpClassifier to be delivered to their destination agents.

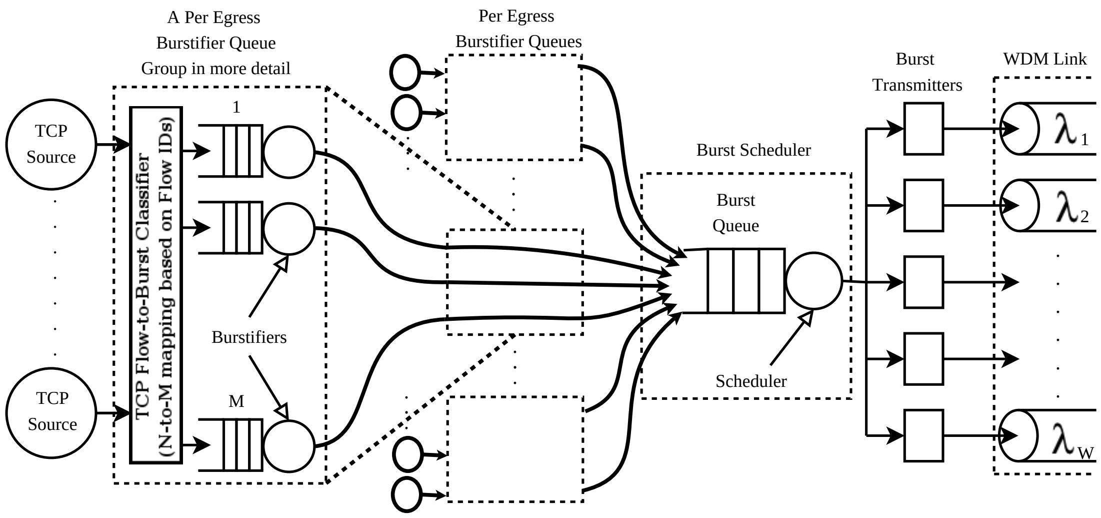

A Per Egress Burstifier Queue Group in more detail
Per Egress Burstifier Queues
Burst Transmitters
WDM Link
TCP Source
TCP Source
TCP Flow-to-Burst Classifier (N-to-M mapping based on Flow IDs)
1
M
Burstifiers
Burst Scheduler
Burst Queue
Scheduler
λ_1
λ_2
λ_W

Figure 3.4: Ingress node model

If the packet is an electronic packet, BurstAgent compares the source routing table of the packet with the list of nodes contained in the table *opticnodes* and finds the corresponding egress node from where this packet will leave the OBS domain. Next, BurstAgent inserts the incoming packet to one of the assembly queues responsible for burstifying packets destined for this destination egress node. The assembly algorithm implemented in the BurstAgent is a hybrid size/timer-based algorithm that keeps track of the size of the burst and the delay experienced by the first packet in the burst. BurstAgent creates a burst when the delay of the first packet reaches a given timeout, or the number of IP packets in the burst reaches a threshold. In our ingress node model, the number of assembly buffers per egress router, M, can be between 1 and the number of flows, N, as shown in Figure 3.4. An incoming packet is forwarded to a per egress burstifier queue group based on the routing information, and it is classified further into an assembly buffer based on the flow ID depending on N and M. If an incoming optical packet is the first packet in the assembly queue, BurstAgent starts the burstification delay timer. When the burst is ready for transmission, BurstAgent creates a control packet carrying all the necessary information for this burst. Before sending the burst, BurstAgent copies the packets in the assembly queue to the burst's payload. Then, BurstAgent sends the control packet to OpClassifier. Sending only the control packet to OpClassifier is enough, because other agents in the node can reach the data packet by using a pointer contained in the control packet pointing to the optical burst to be transmitted.

nOBS also allows the user to select whether ACK packets will be burstified or not. Setting *ackdontburst* variable to 1 allows preventing burstification of ACK packets. In this case, ACK packets are sent to the OBS network as soon they are received and they are carried in the OBS network like ghost packets without any dropping or queuing.

Subclasses of BurstAgent is derived for additional functionality. TrafficGeneratorBurstAgent generates optical bursts whose sizes are exponential with mean $1/\mu$ and whose arrivals are Poisson with rate λ. This burst agent is used to generate background traffic in OBS networks. VariableBurstAgent uses an assembly timeout $T + \varepsilon$ where $\varepsilon \sim N(0, \sigma)$. VariableBurstAgent is used to avoid the continuous blocking problem [21] that occurs among ingress routers using same assembly timeout and contending at a core router.

3.3 OpSRAgent

A new source routing agent called OpSRAgent is implemented in nOBS which is responsible for adding the source routing information to packets, forwarding the packets to links according to the routing information, and controlling when and how to send optical packets using FDLs and wavelength converters. While creating a simulation scenario with nOBS, all the nodes are configured with source routing information within the TCL script. Electrical nodes are configured only with ingress and egress routers of all OBS networks, while optical nodes are informed of routes within the OBS subnetwork they belong. Using a separate source routing module for optical nodes provides the abstraction, i.e., the cloud structure composed of OBS subnetworks, of the core network within the general topology as shown in Figure 3.1.

When OpSRAgent receives a packet, OpSRAgent first checks whether source routing information is available in the packet header and whether this packet is an optical burst or a control packet. If there is no source routing information in the packet header, OpSRAgent considers two scenarios:

1. If this packet is an electronic packet, OpSRAgent writes the routing information to the header of the packet. Then, OpSRAgent checks whether the next hop is an optical node in the same OBS domain. If this is the case, OpSRAgent sends the packet to OpClassifier, which forwards the packet to the BurstAgent for burstification. Otherwise, i.e., if the optical node is the egress node for this packet, OpSRAgent forwards the packet to the next node on an electronic link.

2. If this packet is an optical burst, it means that OpSRAgent has received a newly created burst and control packet pair, so OpSRAgent writes the routing information to the header of both the control packet and the burst.

After ensuring that the source routing information is available in the packet, OpSRAgent checks whether the current node is the destination of this packet. If this is the case, OpSRAgent sends the packet to the OpClassifier. Otherwise, if it is an electronic packet, OpSRAgent sends the packet to the next hop via an electronic link. If this is an optical packet, OpSRAgent tries to send it to an optical link after checking the schedulers. First, OpSRAgent checks the scheduling on this wavelength and link by sending the packet to OpSchedule. OpSchedule returns a result depending on the type of the packet and availability of the channel.

If the packet is a control packet, OpSRAgent takes the following actions based on the result received from the OpSchedule:

1. If there is no contention, OpSRAgent sends the control packet to the optical link for transmission immediately. If this is the first hop of the control packet, OpSRAgent sends the burst corresponding to this control packet to the optical link after delaying the burst for $H\Delta$, where H is the number of hops to be traversed by the burst and Δ is the processing delay per hop.

2. If there is a contention, OpSRAgent checks whether there are unused FDLs or wavelength converters available at the node. If there is, OpSRAgent retries the reservation request, by applying different combinations of available FDLs and converters and chooses the best schedule, if any, according to the scheduling algorithm. OpSchedule learns the availability of FDLs and converters from OpConverterSchedule and OpFDLSchedule, respectively, which are described below. If available FDLs or converters cannot resolve the contention, OpSRAgent drops the control packet.

If the packet is a burst, OpSRAgent takes the following actions based on the result received from the OpSchedule:

1. If there is a reservation for the burst without any contention, OpSRAgent sends the burst to the optical link. If there is a required FDL delay specified in the reservation, OpSRAgent delays the burst before sending to the optical link.
2. If there is no existing reservation for the burst, i.e., the control packet could not succeed in making a reservation for the burst, OpSRAgent drops the burst.

3.4 Optical Schedulers

Each optical node keeps a record of the reservations on outgoing channels, shared FDLs and wavelength converters that are present at the node. OpSchedule holds reservations on outgoing channels while OpConverterSchedule and OpFDLSchedule maintain schedules for wavelength converters and FDLs, respectively. The wavelength converters and FDLs at each node are combined into pools that are shared among all ports at the optical switch, i.e., share-per-node model. The size of the wavelength converter and the FDL pools at each node can be set independently by the user. The user also specifies the maximum FDL delay, which must be limited due to space constraints and for preventing spurious TCP timeouts that degrade the performance significantly [24].

At the ingress node, bursts may be kept in the electrical buffers until they are scheduled and then sent into the optical network. If OpSRAgent cannot find a suitable interval for the burst, it checks possible combinations of wavelength converters and FDLs depending on the node type. If a burst cannot be scheduled, it is dropped. OpSchedule class is responsible for keeping, checking and making reservations on all wavelengths of all links. OpSchedule is connected to the Op-SRAgent. When OpSchedule receives an optical packet from the OpSRAgent, it first checks the type of the packet. If the packet is a control packet, OpSchedule tries to do a reservation for the burst specified in the control packet and returns whether reservation is successful or not. If the packet is a burst, OpSchedule searches for a reservation in its reservation table, which is made earlier by the control packet, and returns whether there is a valid reservation or not. OpSchedule uses Latest Available Unscheduled Channel with Void Filling (LAUC-VF) or Minimum Starting Void (Min-SV) scheduling algorithms in combination with Just Enough Time (JET) signaling. OpSchedule uses a linked-list for storing the reservation list. OpSchedule is responsible for calculating and updating the delay between the control and burst packets.

OpConverterSchedule and OpFDLSchedule are very similar to OpSchedule. These two schedulers are connected to the OpSRAgent, and they are responsible for keeping, checking and making reservations of converters and FDLs at the corresponding nodal pools. They inform the OpSRAgent when OpSRAgent asks for availability in the specified timeline. It is possible to choose whether multiple bursts on a wavelength can use the same FDL subsequently, but the second burst may enter the FDL before the first burst leaves the FDL, by using the single-burst parameter from the TCL script. Both schedulers use linked lists for storing the reservations. An important difference between these two schedulers and Op-Schedule is that when OpSRAgent sends a control packet to the OpSchedule, if reservation is possible, OpSchedule does the reservation directly. However, Op-ConverterSchedule and OpFDLSchedule require a parameter called action. When a control packet is sent to these schedulers, if action variable is set zero, these schedulers only return whether reservation of converter or FDL is possible. They do not do the reservation, unless action variable is set one. This is because the scheduling algorithm may use a combination of FDL and wavelength conversion for resolving the contention, and the OpSRAgent must make sure that both the queried FDL and converter are available. If both schedulers return an affirmative reservation signal, then OpSRAgent informs the schedulers to perform the actual reservations.

In this chapter, the architecture of nOBS was described. In Chapter 4, we present the simulation results obtained by using nOBS for the burst-size independent loss model. We first present the simulation results for the hybrid size/timer-based assembly algorithm to evaluate the claims of previous work. Then, we focus on the comparison of performances of different number of aggregation buffers using the timer-based algorithm. Finally, we investigate the TCP performance improvement brought by increasing the number of burstifiers.

Chapter 4

Burst-size Independent Loss Model

In this chapter, we first validate the previous results about burst assembly. Both size-based and timer-based algorithms can be represented by the hybrid size/timer-based algorithm. As indicated by (2.1), there is a relation between the assembled burst size and the assembly timeout defined by the access bandwidth. In other words, increasing/decreasing the burst size, or equivalently the number of packets inside the burst, implies an increase/decrease in the assembly time required to gather that many packets. Similarly, increasing the assembly timeout causes an increase in the burst size as long as the access bandwidth is constant. As discussed in Chapter 2, some studies indicate that increasing burst size increases TCP performance, while others claim increasing assembly timeout increases the delay on TCP sender and undermines performance. Some others state that as assembly timeout is increased, the performance first increases, then decrease. Therefore, our initial aim is to examine the impact of the burst assembly mechanism on TCP performance for various burst timeout and size threshold ranges.

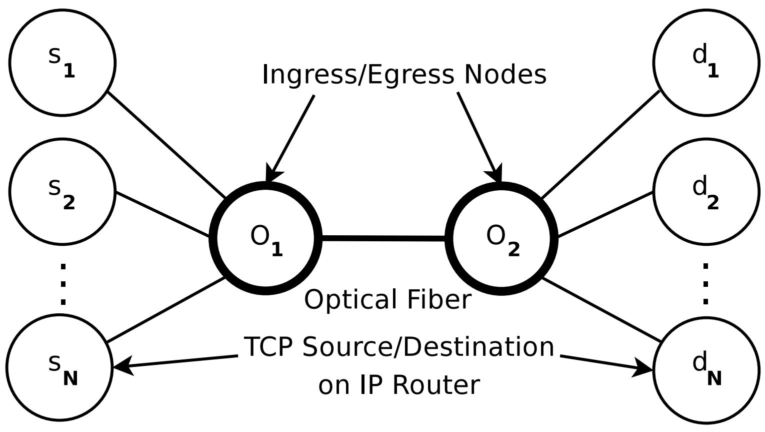


Figure 4.1: Single optical link topology

Secondly, the significance of the reduction in average sending rate as a result of synchronization of TCP flows is analyzed. Most of the studies use per-destination buffering, where all the flows destined to an egress node share the same aggregation buffer. When a burst produced by such an aggregation buffer is dropped, all the flows that have packets in that burst decrease their sending rates simultaneously. In order to examine the effect of using multiple aggregation buffers per egress router, the ingress node model shown in Figure 3.4 is used. In this model, M denotes the number of assembly buffers per egress node. TCP flows are mapped into these assembly buffers based on a simple mapping, i.e., ($\texttt{flow\_id} \mod M$).

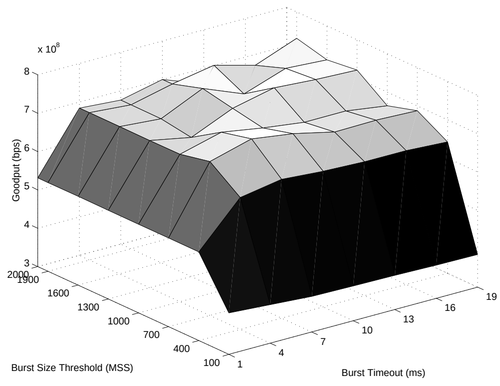
The topology used for studying the effects of burst assembly on TCP performance with burst-size independent loss model is shown in Figure 4.1. For simplicity, the core optical network is modelled as a single fiber with Bernoulli distributed drop probability p to account for losses due to contentions in the core network. This topology is similar to those used in [22, 16]. Moreover, uniform burst loss is adopted in all the studies related to burst assembly. The optical link in $O_2 \rightarrow O_1$ direction and access links are lossless. Sources $s_1 - s_N$ employ infinite FTP flows to the respective destinations $d_1 - d_N$. ACK segments do not experience drops or assembly delays on the return path. The optical duplex link has 1Gbps bandwidth and 10ms propagation delay. Access links are duplex with 155Mbps bandwidth and 1ms delay. As also mentioned in [3], a maximum window size of 64 Kbytes is not sufficient for high-bandwidth delay networks, and since we would also like to eliminate factors other than burst assembly on TCP performance, TCP receiver window size is set to 10000 segments. This means that hardly any limits are implied on the congestion window of the sender by the receiver. The total TCP goodput of $s_1 - s_N$ ($N = 10$) as assembly timeout and size threshold changes for $p = 0.001$, 0.01 and for TCP versions Tahoe, Reno, Newreno and Sack are plotted in Figures 4.2-4.5, respectively.

For Tahoe, Reno and Sack, the simulations with infinite burst-size threshold, i.e., timer-based assembly, are plotted on the largest size-threshold in the figures. The actual maximum burst size reached by the timer-based assembly algorithm is actually larger than the largest size threshold shown in the figures. For TCP Newreno plots, the largest size-threshold also shows the performance of timer-based assembly, however, the size-threshold is chosen to be slightly greater than the largest burst size achieved by the timer-based assembly.

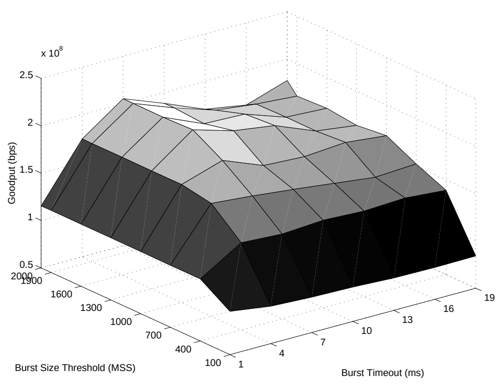
It is seen that the simulation results for all TCP versions and all values of the number of burstifiers per egress node, M, the plots are similar under the same loss probability p. In all the plots, increasing M improves TCP performance in terms of goodput.

For a fixed timeout, it is observed that the goodput increases as the size threshold is increased until the maximum achievable burst size corresponding to the timeout is reached. Increasing the burst size threshold further has no effect on goodput since the assembly algorithm acts as a timer-based burstifier for larger size thresholds.

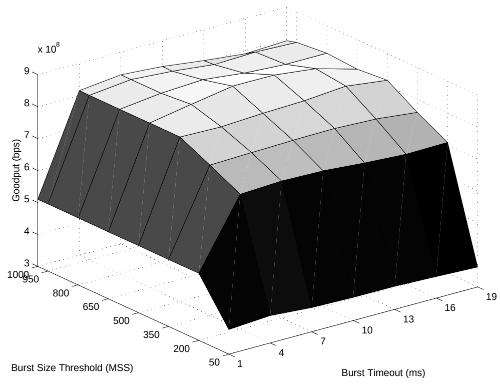
For a fixed burst size threshold, the goodput increases as the burstification timeout is increased, but starts to decrease when the minimum assembly time corresponding to current size threshold is exceeded. This can be explained by looking at those cases where the congestion window is smaller than the burst size threshold. TCP source transmits its window and starts to wait for acknowledgments. Since resulting burst is smaller in size than the threshold, the burst assembler waits for the timeout to expire. Consequently, when the burst timeout is increased further, the goodput decreases.



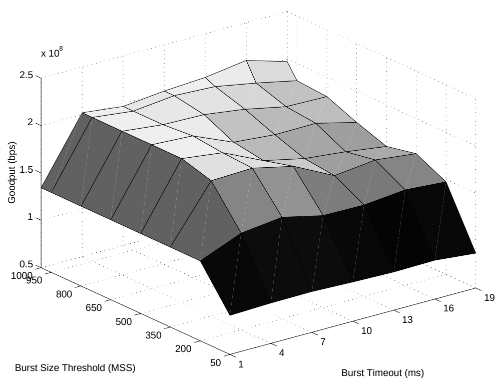
(a) M=1, p=0.001



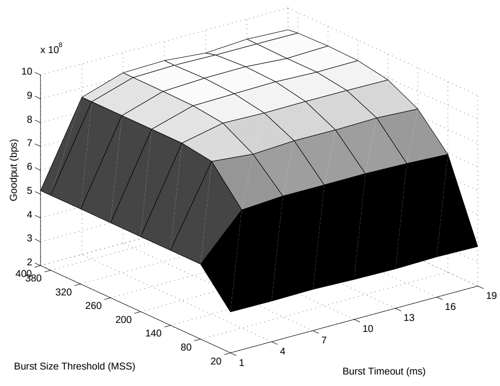
(b) M=1, p=0.01



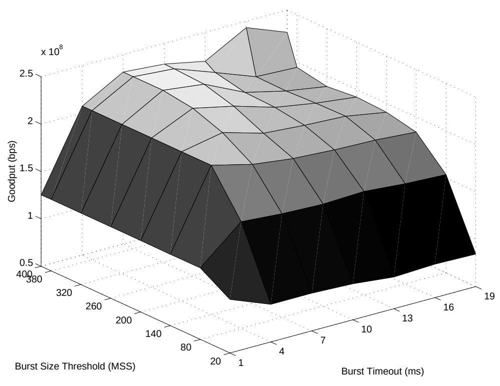
(c) M=2, p=0.001



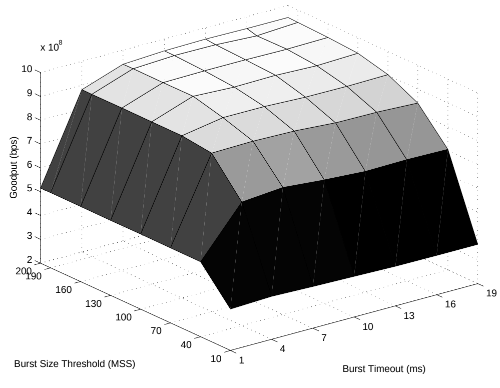
(d) M=2, p=0.01



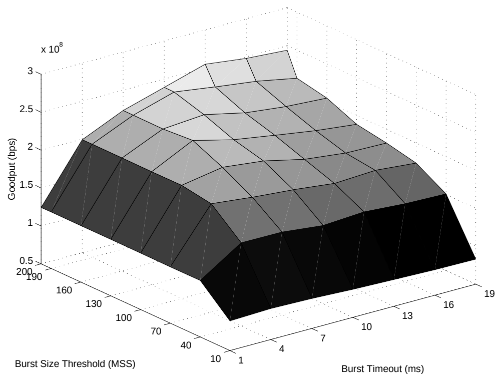
(e) M=5, p=0.001



(f) M=5, p=0.01

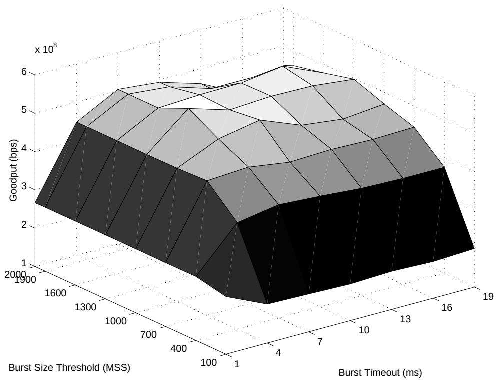


(g) M=10, p=0.001

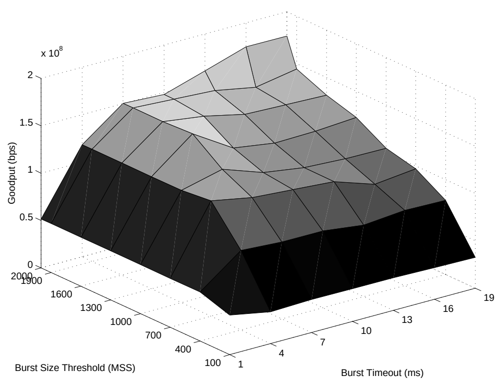


(h) M=10, p=0.01

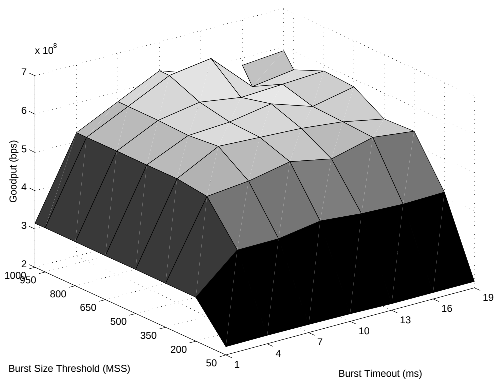
Figure 4.2: Goodput vs size-threshold and assembly timeout for TCP Tahoe



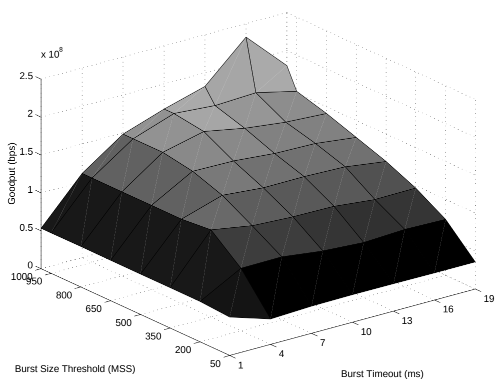
(a) M=1, p=0.001



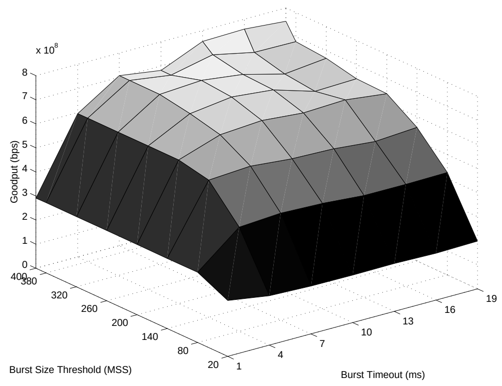
(b) M=1, p=0.01



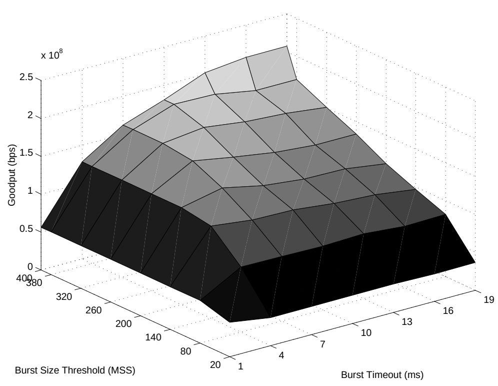
(c) M=2, p=0.001



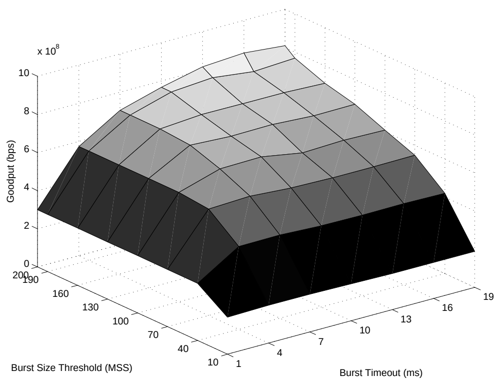
(d) M=2, p=0.01



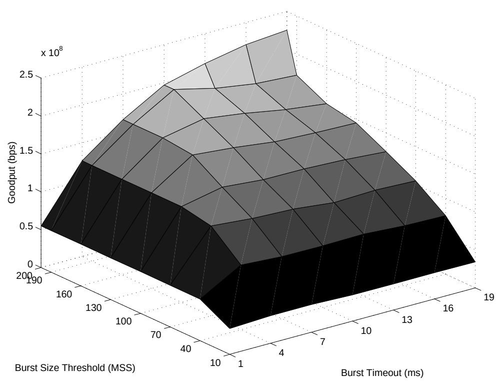
(e) M=5, p=0.001



(f) M=5, p=0.01

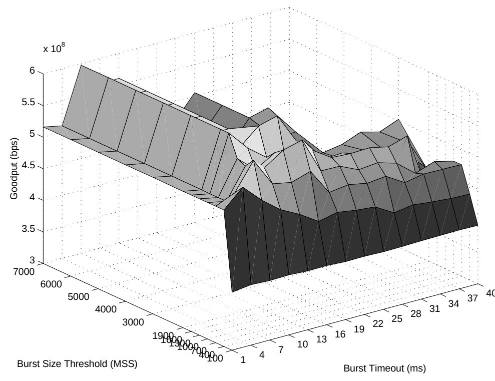


(g) M=10, p=0.001

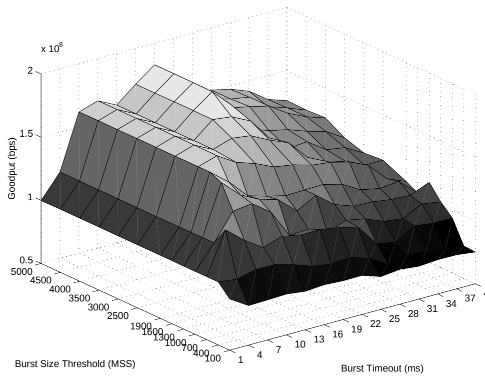


(h) M=10, p=0.01

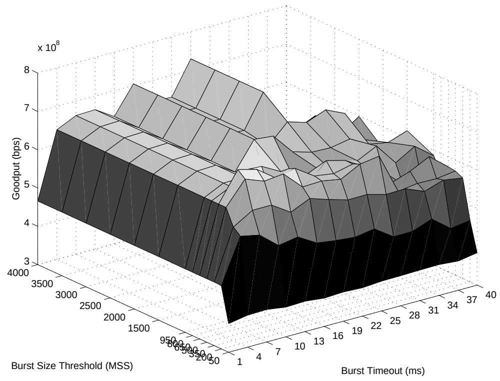
Figure 4.3: Goodput vs size-threshold and assembly timeout for TCP Reno



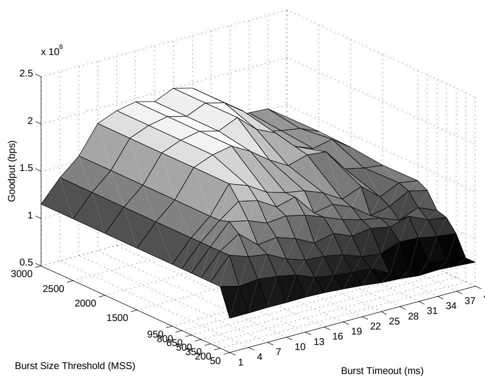
(a) M=1, p=0.001



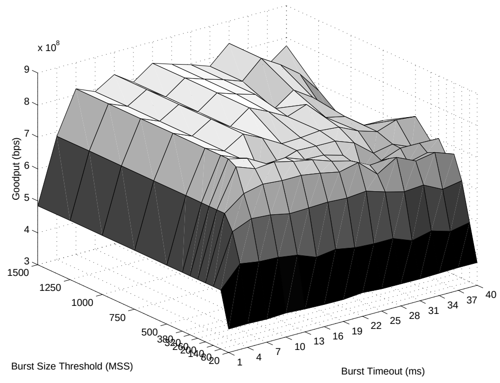
(b) M=1, p=0.01



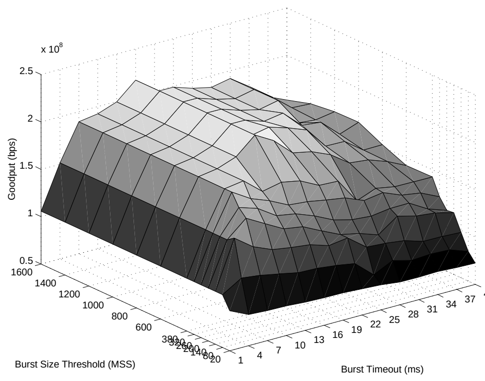
(c) M=2, p=0.001



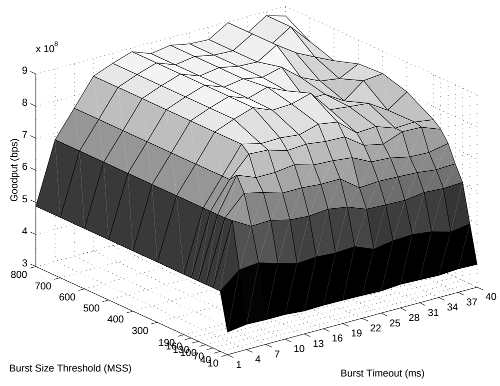
(d) M=2, p=0.01



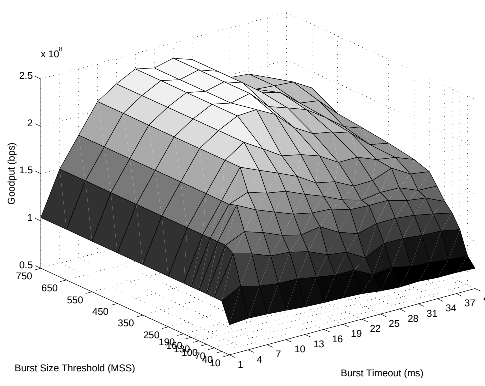
(e) M=5, p=0.001



(f) M=5, p=0.01

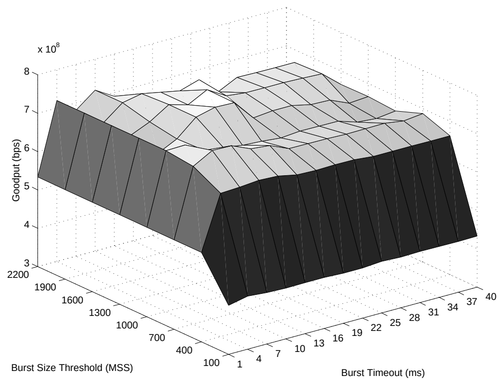


(g) M=10, p=0.001

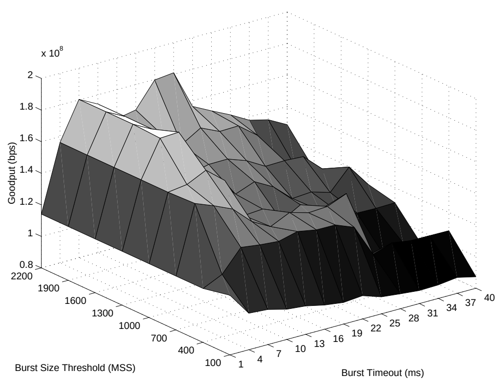


(h) M=10, p=0.01

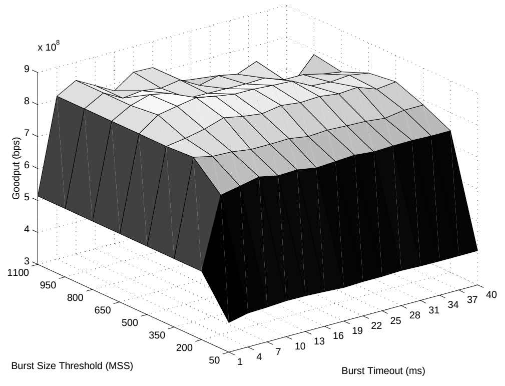
Figure 4.4: Goodput vs size-threshold and assembly timeout for TCP Newreno



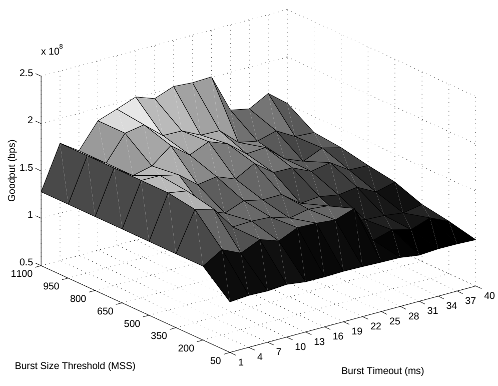
(a) M=1, p=0.001



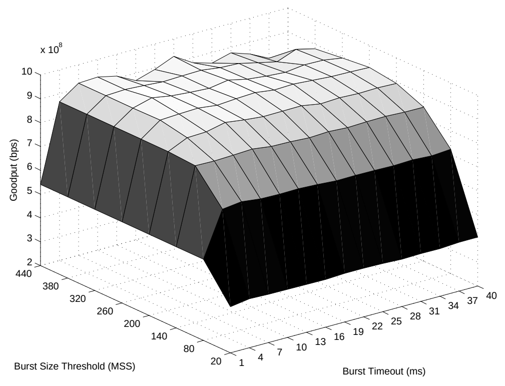
(b) M=1, p=0.01



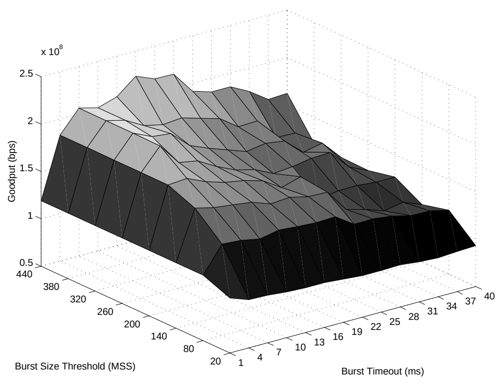
(c) M=2, p=0.001



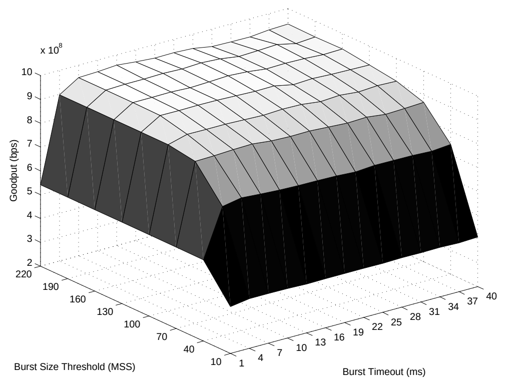
(d) M=2, p=0.01



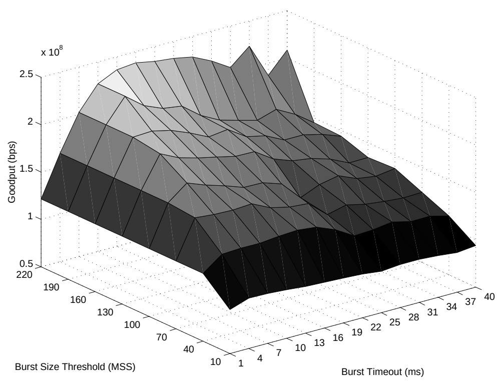
(e) M=5, p=0.001



(f) M=5, p=0.01



(g) M=10, p=0.001

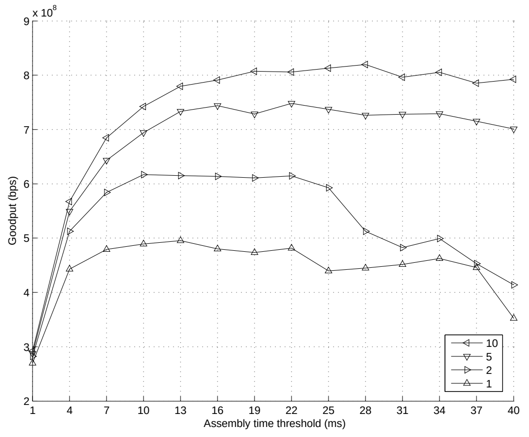


(h) M=10, p=0.01

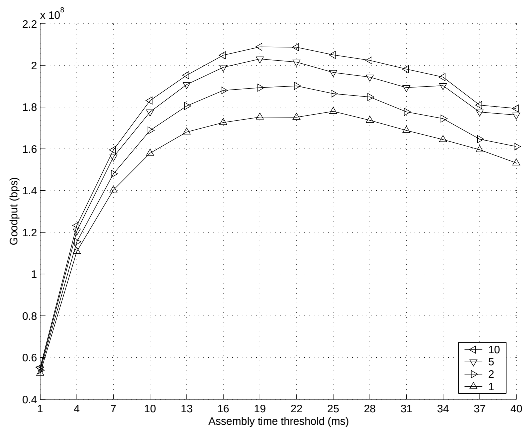
Figure 4.5: Goodput vs size-threshold and assembly timeout for TCP Sack

Timer/size-threshold based algorithm reduces to size threshold for a timeout of infinity, therefore, the performance at the largest timeout reflects the performance of the size-based algorithm. On the other hand, the performance of the timer-based algorithm is plotted on the largest size threshold. In other words, these figures helps to compare the performances of the three burstification algorithms. All the figures indicate that timer-based assembly performs the best.

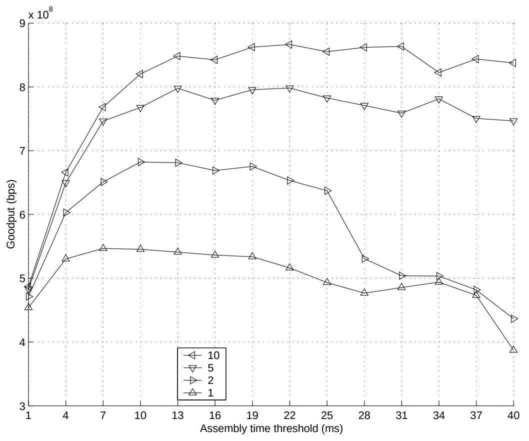
Since the highest goodput is obtained by the timer-based algorithm, we focus on timer-based assembly algorithm in the rest of this thesis to evaluate the impact of the number of the burstifiers on TCP performance. The timer-based assembly algorithm is simulated with a wider range of assembly timeouts and longer simulations to achieve better results for TCP versions Reno, Newreno and Sack, since these three versions are the mostly used ones in practice. Figure 4.6 shows the outputs of these simulations. The goodput values for $M = 1, \quad 2, \quad 5$ and 10 are plotted together for comparison. We observe that increasing the number of burst assemblers significantly improves the goodput for all three TCP versions since synchronization between large number of TCP flows is avoided as the number of burstifiers is increased. At this point, it is necessary to show how TCP senders adjust their transmission rates for different M values. As mentioned in Section 2.1, the size of the congestion window determines the instantaneous transmission rate of the sender. The congestion window sizes of TCP sources $S_1 - S_{10}$ and their sum is plotted for M =1, 2, 5 and 10 in Figures 4.7 and 4.8 for a sample simulation point. The simulation point employs TCP Reno senders with burst loss probability $p = 0.01$ and assembly timeout $T = 22ms$ on the topology depicted in Figure 4.1. When $M = 1$, all the flows share the same aggregation buffer. When a burst is lost, this burst contains segments from every flow, so all flows decrease their congestion window sizes simultaneously. As seen in Figure 4.7(a), flows become synchronized and the sum of their congestion windows show that the channel utilization drops severely after burst losses. For $M = 2$, odd numbered flows, i.e. $S_1, \quad S_3, ..., \quad S_9$ are aggregated in one buffer while the rest of the flows are aggregated in the other buffer. Figure 4.7(b) shows that flows sharing an aggregation buffer are still synchronized among themselves, but



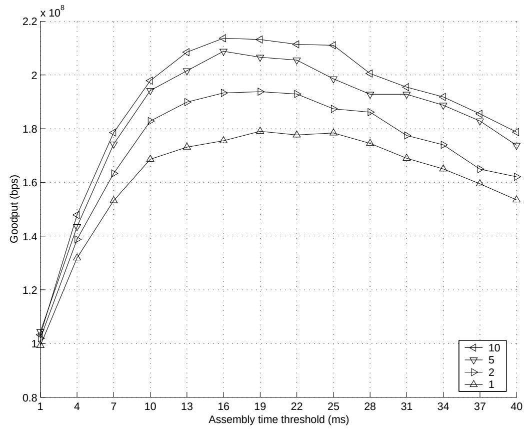
(a) p=0.001, Reno TCP



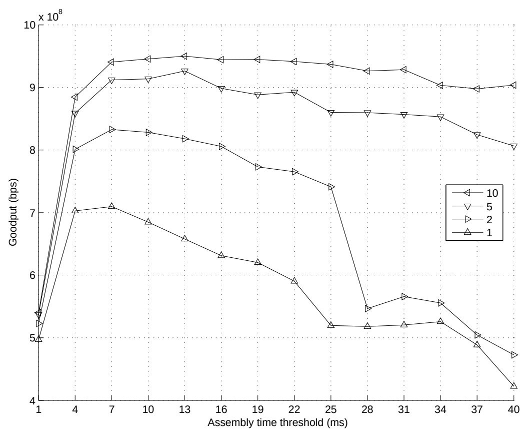
(b) p=0.01, Reno TCP



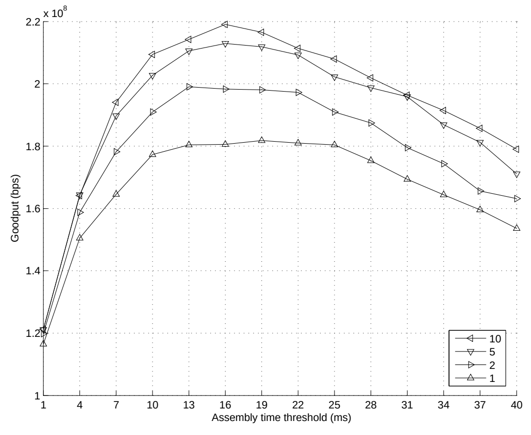
(c) p=0.001, Newreno TCP



(d) p=0.01, Newreno TCP

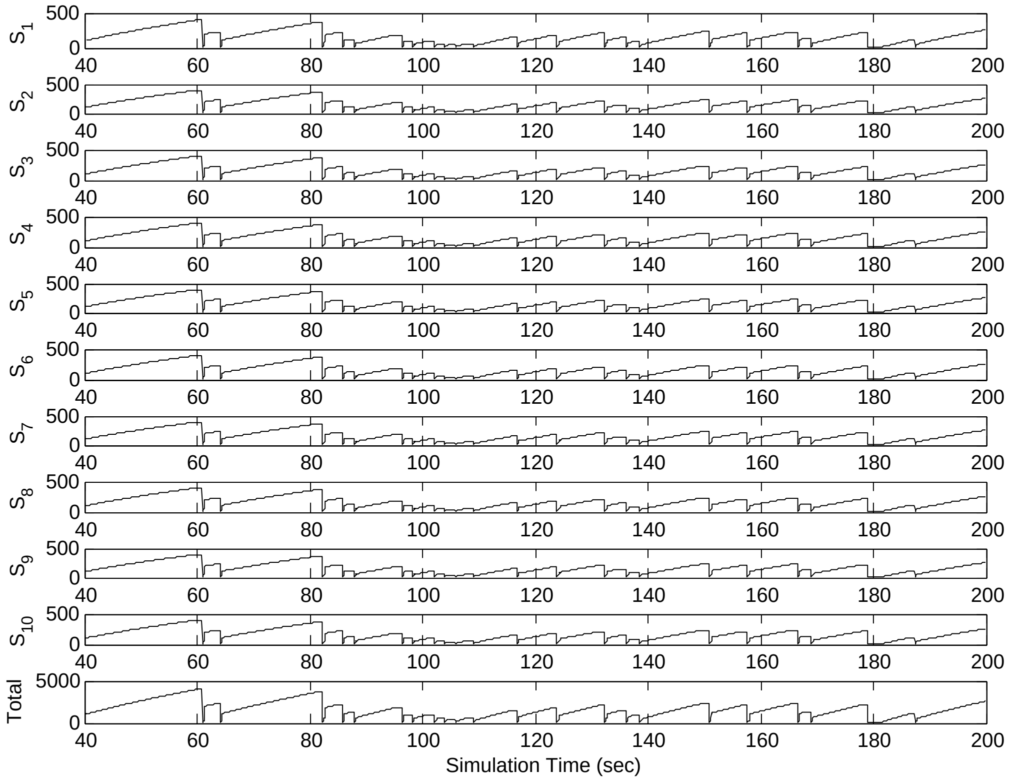


(e) p=0.001, Sack TCP

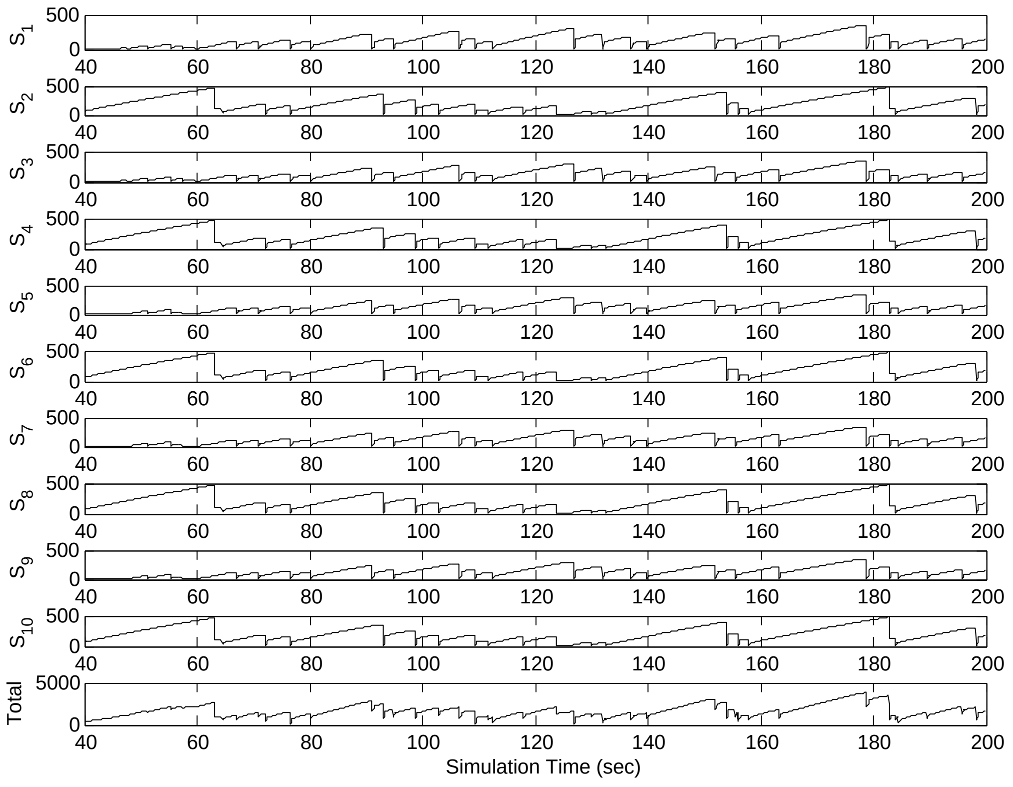


(f) p=0.01, Sack TCP

Figure 4.6: Total goodput with timer-based assembly for $N = 10$, $M = 1, 2, 5, 10$

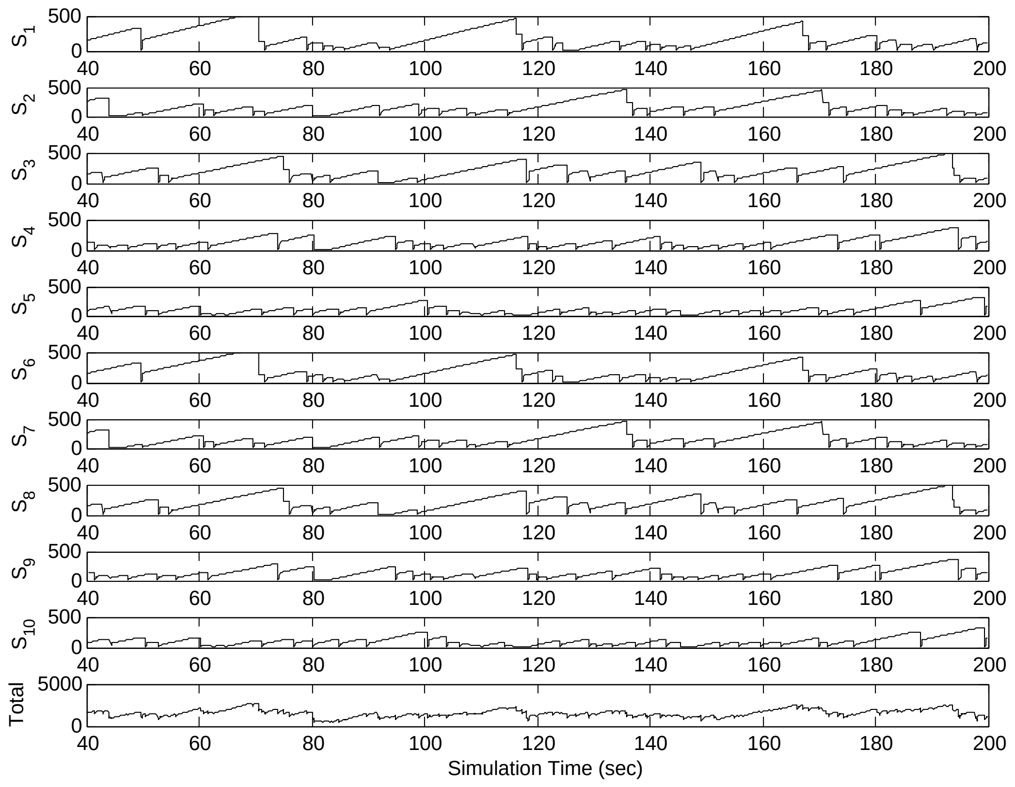


(a) M=1

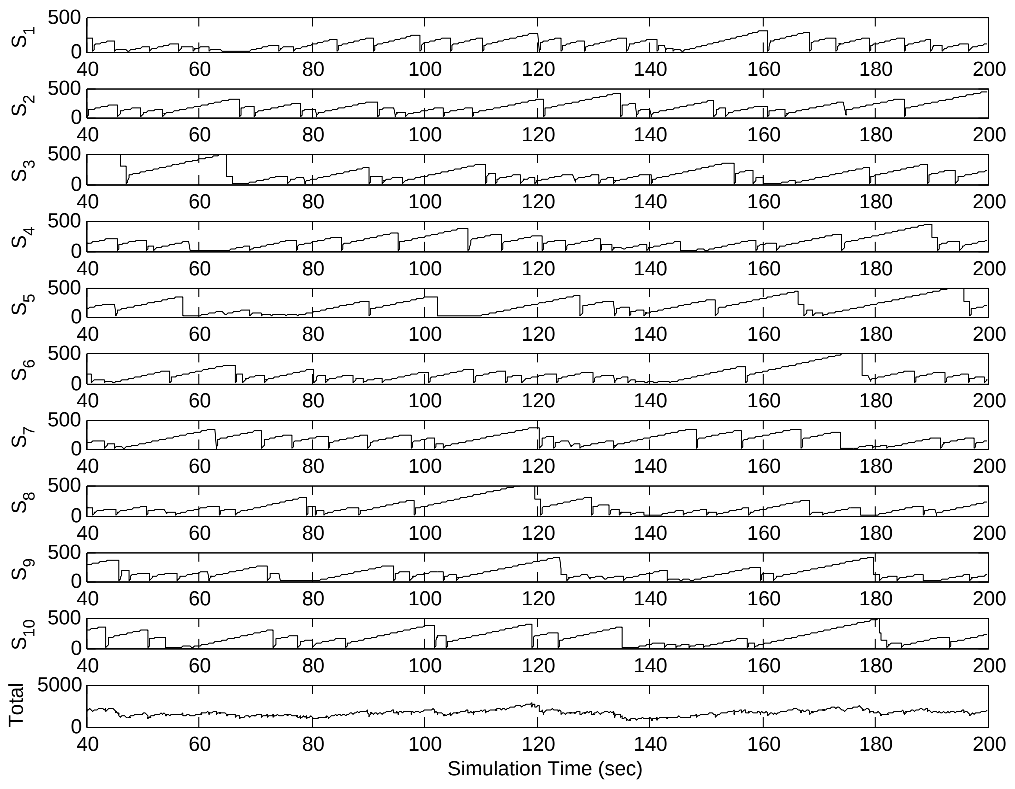


(b) M=2

Figure 4.7: Congestion window sizes for TCP Reno, $p = 0.01$, $T = 22ms$, $M = 1, 2$



(a) M=5



(b) M=10

Figure 4.8: Congestion window sizes for TCP Reno, $p = 0.01$, $T = 22ms$, $M = 5, 10$

the sum of the congestion windows implies that the overall level of synchronization is reduced. According to Figure 4.8(a), even less flows are synchronized. In Figure 4.8(b), no flows are synchronized and the sum of the congestion windows is almost constant. To sum up, when the degree of synchronization is reduced by increasing the number of burstifiers, the congestion windows of flows belonging to different burst assemblers tend to balance each other and the link is better utilized.

Figure 4.6 also shows that as the assembly time is increased, goodput first increases, and then starts to decrease for all three TCP versions. In the region where goodput increases with timeout, the delay penalty is small and DFL gain is dominant, therefore increasing the burst size increases the goodput. On the other hand, the improvement provided by DFL gain saturates after some timeout value and the delay penalty begins to dominate, which causes the goodput to deteriorate.

The correlation gain depends on the number of segments that are burstified into a burst. In the direction where the assembly timeout is decreased, the number of segments from any given flow decrease. For the simulation scenario corresponding to Figure 4.6, the assembly timeout that defines the border of the slow flow regime is given by (2.2) as 53.68μsec. In other words, timeouts corresponding to slow flows are out of the assembly timeout region that we have used in our simulations. As the congestion window sizes change, the flows become fast flows when their congestion window sizes are below $B_a.T_b$ and they become medium flows as the `CongWin` exceeds this value. Therefore, Figure 4.6 shows that the performance improvement brought by additional assemblers per egress node are significant for medium and fast flows, but as we go towards the slow flow regime, the improvement disappears and there is hardly any improvement for slow flows.

Another important observation is that the rate of decrease in goodput as the timeout is increased depends on loss probability p. When p is large, the congestion window cannot increase to large values due to more frequent burst losses. In this case, the increase in the timeout does not increase the burst size significantly

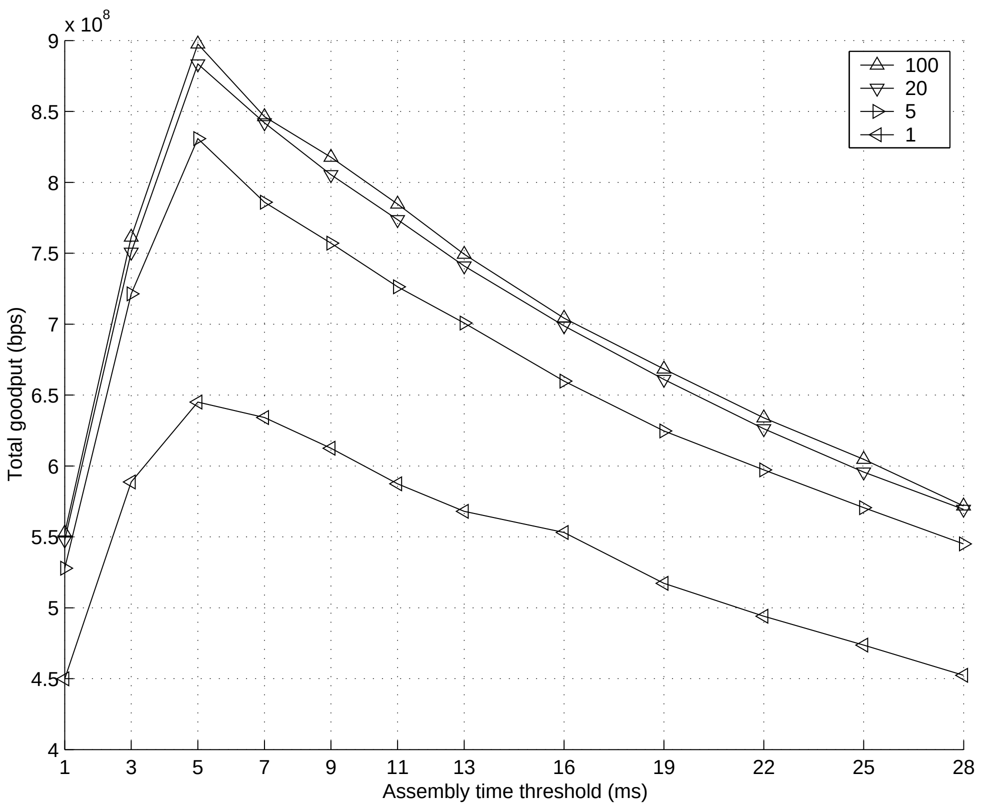


Figure 4.9: Total goodput with timer-based assembly for $N = 100$, $p = 0.01$, $M = 1, 5, 20, 100$ and Newreno TCP

and the increase in DFL gain with increasing timeout is not significant. As a result, the goodput decreases more rapidly with increasing timeout due to the delay penalty. On the other hand, larger bursts are generated as the timeout is increased when p is small, and the DFL gain increases with the timeout. This partially compensates the effect of the delay penalty, and the goodput does not degrade much with the increasing throughput for all three TCP versions. In addition, it is observed that a relatively low number of buffers may perform close to the per-flow aggregation case. Since the cost of additional burstifiers can be compromised by the improvement in goodput, employing moderate number of buffers with respect to the number of flows constitutes a cost-effective solution.

Although all three TCP versions exhibit similar characteristics as the timeout and the number of burstifiers are changed, TCP Sack achieves the highest goodput among the three TCP versions. Sack outperforms the other two versions since it quickly retransmits the lost segments with selective acknowledgements. Reno and Newreno have very close performances, however Newreno slightly outperforms Reno.

In order to evaluate the effects mentioned up until this point in a more realistic environment where the receive window is 64 KBytes and number of TCP flows are sligtly larger, the same network is simulated with $N = 100$ Newreno flows. The bandwidth of the optical link is set to 2.5 Gbps and the burst loss probability p is set to 0.01. The MSS of TCP sources are set to 512 Bytes and the receive windows are set to 128 MSS. Figure 4.9 shows the results of the simulations. The effect of the number of burst assemblers is similar to the previous results obtained for $N = 10$. In addition, it is observed that a relatively low number of buffers may perform close to the per-flow aggregation case. Since the cost of additional burstifiers can be compromised by the improvement in goodput, employing moderate number of buffers with respect to the number of flows constitutes a cost-effective solution.

Another factor that differentiates these figures from prior ones is the window size of the receivers. The sender's congestion window usually stays under the receiver's window, but sometimes reaches values that are slightly larger. That is why the optimal assembly timeout turned out to be faintly larger than 3.4ms, the minimum assembly timeout required to create a burst of size 64KBytes. After the optimal timeout, as the congestion window cannot grow further, DFL gain stays constant at its maximum for large timeouts. Consequently, the effect of DP on goodput can be seen more clearly for large values of the timeout, and the goodput decreases more rapidly with increasing timeout compared to the case with $N = 10$ flows.

In Table 4.1, the goodput enhancement of using multiple burstifiers with respect to the single burstifier case, i.e., per destination burstification, is shown for different TCP versions, number of TCP flows and loss probability. For $N = 10$ and $p = 0.001$, the goodput with per-flow burstification increases 33-65% compared to the case with per-destination burstification for different TCP versions. The goodput enhancement is largest with Reno and smallest with Sack. We also

Table 4.1: Percentage goodput increase versus number of burstifiers for different TCP versions and loss probability

	$N = 10$				$N = 100$	
p	M	Reno	Newreno	Sack	M	Newreno
0.001	2	24.55	24.77	17.31	5	28.82
	5	51.00	45.99	30.50	20	36.99
	10	65.40	58.48	33.84	100	39.17
0.01	2	6.85	8.22	9.48	5	13.53
	5	14.10	16.63	17.16	20	17.67
	10	15.20	19.36	20.52	100	18.78

observe that the goodput achieved with $M = 5$ is very close to the per-flow burstification case. For $N = 10$ and $p = 0.01$, the goodput enhancement with per-flow burstification with respect to per-destination burstification is about 15-20%. Similarly, the goodput achieved with $M = 5$ is very close to the per-flow burstification case. The burstification architecture at the edge router should be designed taking into account both the goodput enhancement and additional management complexity of using multiple burstifiers, and $M = 5$ seems to provide a nice compromise for this case. The goodput enhancement is shown also for $N = 100$ and $p = 0.001, 0.01$ in Table 4.1 for TCP Newreno. The goodput enhancements are 18-39% for $M = 100$ with respect to $M = 1$, and most of the gain achieved by $M = 100$ is provided with $M = 20$, i.e., by using one fifth of the burstifiers.

Having modelled the optical core network as an optical fiber with Bernoulli distributed loss probability, we showed how the results of previous works fit into the larger picture and how assembly mechanism affects the TCP performance under different loss probabilities and TCP versions. We discussed the improvement that could be achieved using multiple aggregation buffers per egress node. In the next chapter, we extend our studies to a more realistic network scenario where the burst drop probability depends on the length of the burst.

Chapter 5

Burst-size Dependent Loss Model

In all the studies related to burst assembly, burst drop rate in the core network is assumed to be independent of the length of the burst. Let us think of a core network without FDLs or wavelength converters. When a core router receives a control packet, the reservation request will be granted if the requested interval does not overlap with any of the previous reservations. This is possible if two conditions are satisfied. First, the reservation should start in a void (a non-reserved time interval between two reservations). Secondly, the reservation should end before the start of any other prereserved interval. The latter condition implies that the duration of the reservation, or equivalently the length of the burst, is important in the failure or fulfillment of the reservation request. The reservation starts an *offset* time ($H.\Delta$) after the reception of the control packet where H is the number of remaining hops and Δ is the per-hop processing time of the control packet. In the typical case where an OBS network contains multiple hops, bursts arriving at a core node would have to fit into voids created by the bursts that are destined to further hops. Consequently, drop probability of a burst should depend on its length. In order to test this conjecture, the multihop core network seen in Figure 5.1 is used. The background burst generator creates bursts destined to $D_1 - D_{20}$. The size of these bursts are exponential with mean $1/\mu$ and burst arrivals are Poisson with rate λ. The bursts carrying segments of $S_1 - S_{20}$ try to fit into voids created by the bursts from the burst generator. We examined the

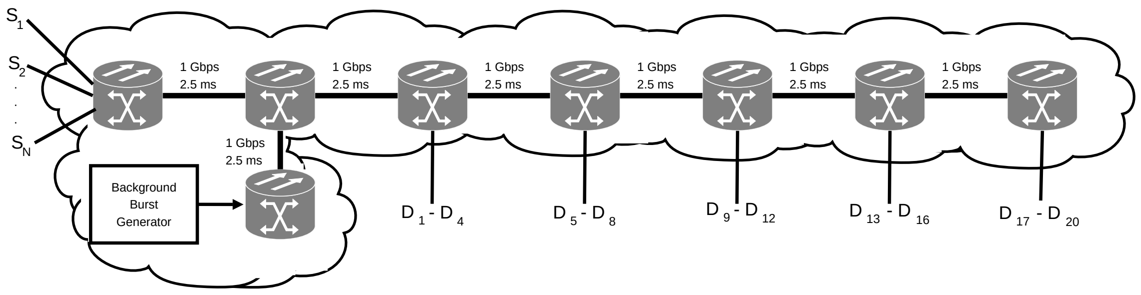


Figure 5.1: Topology used in simulations

scenario in two subsets as infinite size flows and Internet-like traffic sources.

5.1 Infinite size flows

Figure 5.1 shows the network topology used for studying the effects of burst length dependent losses. Sources $S_1 - S_N$ employ an infinite FTP flow to the respective destination $D_1 - D_N$ (N=20). Optical links have 1 Gbps bandwidth and 2.5 msec propagation delay. The background burst generator produces bursts whose sizes are exponentially distributed with $1/\mu$ and burst arrivals are Poisson with rate λ. All bursts are destined uniformly to the five egress nodes connected to $D_1 - D_{20}$. Access links have 50 Mbps bandwidth and 1 msec propagation delay.

Figure 5.2 shows the loss probability for each egress node as a function of the burst length with the parameters $1/\mu$ = 200 nsec, $1/\lambda = 2msec$, $M = 1$, the nodal processing delay $\Delta = 50\mu sec$ and the assembly timeout $T = 10msec$. The statistics of the bursts carrying segments of $S_1 - S_{20}$ are grouped into 10 bins according to the number of packets in the burst, which ranges from 1 to a maximum value of 60 packets. It can be seen that the loss probability is relatively high for the flows with smaller residual offset times, as expected. Moreover, the loss probability increases as the burst size increases. The impact of void filling mechanism in the core router scheduler becomes important for those bursts that are closer to their destinations because they need to fit in the voids created beforehand by the bursts that have larger residual offset times. Consequently, the dependence of the loss probability on the burst size is strongest for the bursts

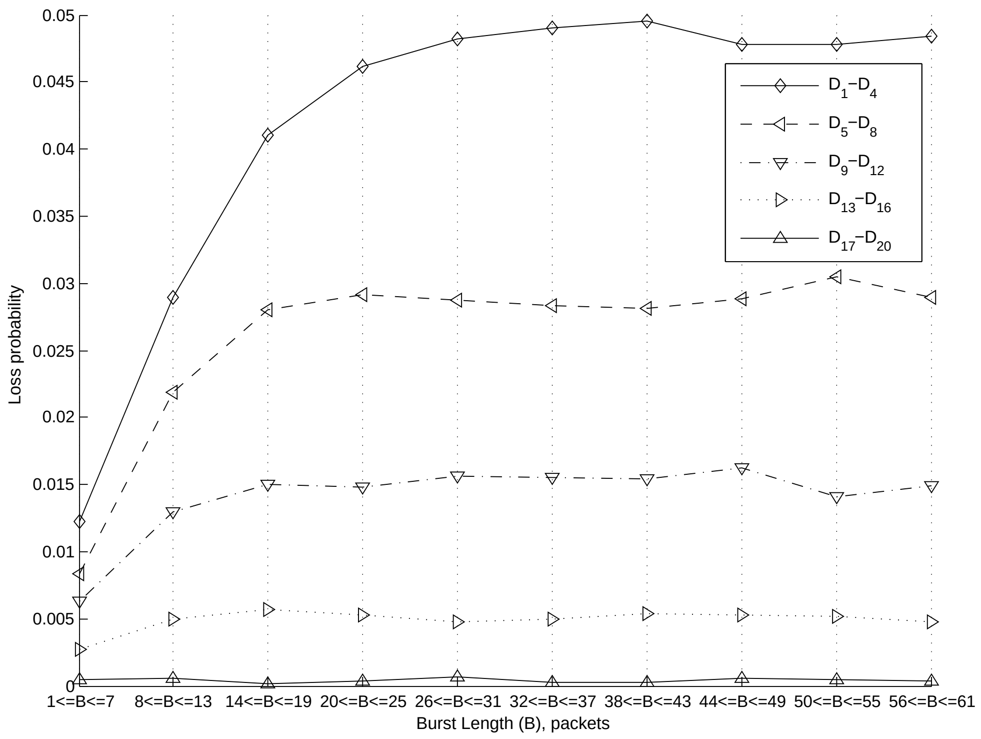


Figure 5.2: Loss probability vs. burst length for different egress nodes

destined to $D_1 - D_4$. Such a correlation is not observed for the bursts destined to $D_{17} - D_{20}$ since bursts destined to $D_{17} - D_{20}$ are not required to fit into voids.

In addition to the mechanisms mentioned in [23] such as DP, the loss penalty and correlation gain, this observation brings forward another critical factor in analysis of TCP performance in OBS networks. The significance of the burst length dependent losses depends on the residual offset time, per-hop processing delay (Δ) and the burst transmission time.

Figures 5.3 and 5.4 plot the goodput and the average burst size as a function of the burst assembly timeout for the nearest and farthest egress nodes, respectively, and for different values of the number of burstification buffers, M, using the parameters $1/\mu = 200nsec$, $1/\lambda = 2msec$, $\Delta = 50\mu sec$, when TCP Reno is used. We observe that for both destinations the average goodputs increase with the number of burstifiers. It is also observed that the average burst size increases linearly with the assembly timeout for flows destined to $D_{17} - D_{20}$. On the other hand, the average burst size first increases and then saturates for the flows destined to $D_1 - D_4$. This is due to the fact that the TCP flows destined to $D_1 - D_4$ experience much more frequent burst losses and consequently they do not achieve very large congestion windows. The saturation of the average burst sizes coupled with the additional assembly delay cause the drop in the average goodput for flows destined for $D_1 - D_4$ as the assembly timeout increases. On the other hand, the TCP flows destined for $D_{17} - D_{20}$ can achieve very large congestion windows and the resulting burst sizes increase with the assembly timeout. The correlation benefit achieved by having longer bursts is partially compensated by the delay penalty, and the average TCP goodput does significantly change as the burst assembly timeout is increased.

We observe from Figures 5.3 and 5.4 that the flows destined for $D_{17} - D_{20}$ achieve much higher goodput compared with the flows destined for $D_1 - D_4$. Although the flows destined for $D_{17} - D_{20}$ experience larger delays, their much smaller loss probability results in higher goodput.

The comparison of Figures 5.3 and 5.4 also reveal that the maximum goodput for the flows destined for $D_1 - D_4$ are achieved at smaller values of the burst assembly timeout compared with the flows destined for $D_{17} - D_{20}$. In fact, the maximum goodput is achieved before the burst size saturates for the flows destined for $D_1 - D_4$. This is due to the fact that the loss probability increases significantly as the burst size increases for the flows destined for $D_1 - D_4$ as it was shown in Figure 5.2. Although the correlation gain is increasing with the burst size, the burst length dependent nature of the burst losses causes the average goodput to start decreasing before the average burst size reaches its maximum. A similar behavior is not observed in Figure 5.4 since the burst losses is independent of the burst size for the flows destined for $D_{17} - D_{20}$.

The performance improvement in the maximum average goodputs achieved by using $M = 2$ and $M = 4$ with respect to the case of $M = 1$ for TCP Reno and TCP Sack are shown in Tables 5.1 and 5.2, respectively. The results show that the improvement in the average goodput is maximum for the flows destined for closer egress nodes, and the average goodput improvement generally increases with the increasing nodal processing delay Δ. The improvements are in the range

Table 5.1: Percentage goodput increase as a function of the number of burstifiers for TCP Reno

		Destination					
Δ (μsec)	M	1-4	5-8	9-12	13-16	17-20	Avg.
50	4	16.91	8.15	6.86	2.91	2.43	6.22
50	2	6.47	4.22	4.19	3.28	1.36	3.87
100	4	34.82	26.83	8.61	6.21	1.89	6.91
100	2	13.91	7.78	2.85	4.91	0.82	2.69
200	4	26.78	35.79	31.73	6.70	15.52	23.15
200	2	13.86	14.86	12.36	4.31	6.01	6.70
500	4	26.49	27.83	31.22	34.97	15.95	36.92
500	2	13.36	10.94	14.53	16.76	3.27	10.24

Table 5.2: Percentage goodput increase as a function of the number of burstifiers for TCP Sack

		Destination					
Δ (μsec)	M	1-4	5-8	9-12	13-16	17-20	Avg.
50	4	39.41	8.47	8.79	5.43	0.38	4.91
50	2	19.72	4.76	3.73	3.15	0.04	3.03
100	4	48.81	54.93	13.05	10.35	0.62	6.33
100	2	26.21	25.25	6.09	8.68	0.46	2.72
200	4	44.79	57.58	45.30	6.91	0.46	24.45
200	2	25.43	25.01	26.07	4.74	0.00	4.35
500	4	47.83	38.83	48.91	54.20	1.29	37.88
500	2	24.76	17.81	25.86	25.44	0.73	8.07

of 17-35% for the closest nodes and the average goodput improvement over all destinations is 6-37% for TCP Reno and $M = 4$. For the case of $M = 2$, the average goodput increases are in the range of 3-10% compared to $M = 1$. The performance improvements for TCP Sack are slightly larger compared to TCP Reno.

5.2 Internet-like traffic sources

In this section, the infinite FTP flows of Section 5.1 are replaced by flows that mimics the Internet traffic. The heavy tail and large variance in flow sizes of typical Internet flows are modelled with Bounded Pareto distribution [28] while flows arrive according to a Poisson process with rate $\lambda' = 0.1$ arrivals/sec. A Bounded Pareto distribution is denoted by tail heaviness α, minimum flow size k and maximum flow size p. The probability density function $f(x)$, cumulative density function $F(x)$ and the n-th moment m_n are given as follows [28]:

$$f(x) = \frac{\alpha k^\alpha}{1-(k/p)^\alpha} x^{-\alpha-1}, \quad k \leq x \leq p, \quad 0 \leq \alpha \leq 2 \tag{5.1}$$

$$F(x) = \frac{1}{1-(k/p)^\alpha}[1-(k/x)^\alpha], \quad k \leq x \leq p, \quad 0 \leq \alpha \leq 2 \tag{5.2}$$

$$m_n = \frac{\alpha}{(n-\alpha)(p^\alpha - k^\alpha)}(p^n k^\alpha - k^n p^\alpha) \tag{5.3}$$

In our simulations, background burst generator is operated with $1/\mu = 200\mu sec$, $1/\lambda = 2msec$. The nodal processing delay is taken as $\Delta = 50\mu sec$. Each IP router $S_1 - S_{20}$ is assigned with a flow generator, which produces TCP Reno flows with Bounded Pareto size distribution and Poisson arrival pattern. The flows assigned to $S_1 - S_{20}$ send their segments to the respective destination $D_1 - D_{20}$. For each flow generator, Bounded Pareto parameters are $\alpha = 1.2$, $k = 10MB$, $p = 1GB$ and flow arrivals are Poisson. TCP flow IDs are uniformly distributed in $\{0, \quad 1, \quad 2, \quad 3\}$, and M is chosen amongst 1,2 and 4.

The average goodput of the TCP flows is shown in Figure 5.5 for each egress node. Once again, it is confirmed that increasing the number of assembly buffers improves TCP performance. The goodputs of further egress nodes are relatively high compared to the goodputs of nearer egress nodes. The reason for this behavior is that the drop probability is lower for bursts with higher residual offsets. For the egress of $D_1 - D_4$, the drop probability is so high that the DFL gain cannot compensate the delay penalty as assembly timeout is increased, therefore goodput constantly decreases. When we look at further egress nodes, it can be seen that the effect of DFL gain becomes dominant and for the egress of $D_{17} - D_{20}$, the decrease in goodput for increasing assembly timeout is minimal.

The simulations for $M = 1$, $M = 2$ and $M = 4$ has been fed with the same Bounded Pareto flows, enabling us to compute the ratios of the goodputs achieved

Table 5.3: Percentage goodput increase as a function of the number of burstifiers

	Destination				
M	$D_1 - D_4$	$D_5 - D_8$	$D_9 - D_{12}$	$D_{13} - D_{16}$	$D_{17} - D_{20}$
4	30.52	19.33	12.03	16.40	17.21
2	15.46	8.82	6.45	9.34	7.43

by any flow under $M = 1$, $M = 2$ and $M = 4$. Figure 5.6 shows goodput ratio of $M = 2$ over $M = 1$ for each individual flow for each egress node. Similarly, Figure 5.7 shows goodput ratio of $M = 4$ over $M = 1$ for each individual flow for each egress node. The plots show that the variation of the goodput improvement is very high for short flows while variation drops for longer flows. As a result of Poisson arrivals and Bounded Pareto flow sizes, the number of flows sending packets to the ingress node changes. As the number of flows changes, the sizes of the bursts generated by the ingress node changes. The segments burstified into large bursts experience higher drop probability, while the segments that are burstified into shorter bursts experience lower drop probability. The serving time of short flows are small compared to the rate of change of number of flows arriving to the burstifier. In other words, the improvement ratios of the short flows depend heavily on the instantaneous number of flows and therefore exhibit larger fluctuations. The large flows, on the other hand, experience to a larger extent the average performance of the system.

Figures 5.6 and 5.7 also show the average of the improvement ratios of individual flows that are grouped into 10 bins (indicated by the thick line). The average improvement ratio stays mostly constant as the flow size increases, so the TCP performance improvement brought by additional assembly buffers does not depend on flow sizes. The numerical values for the average percentage goodput increase for all flows as a function of the number of burstifiers is given in Table 5.3. The results show that the improvement is the most significant for the nearest egress node, while the improvement decreases for the further egress nodes.

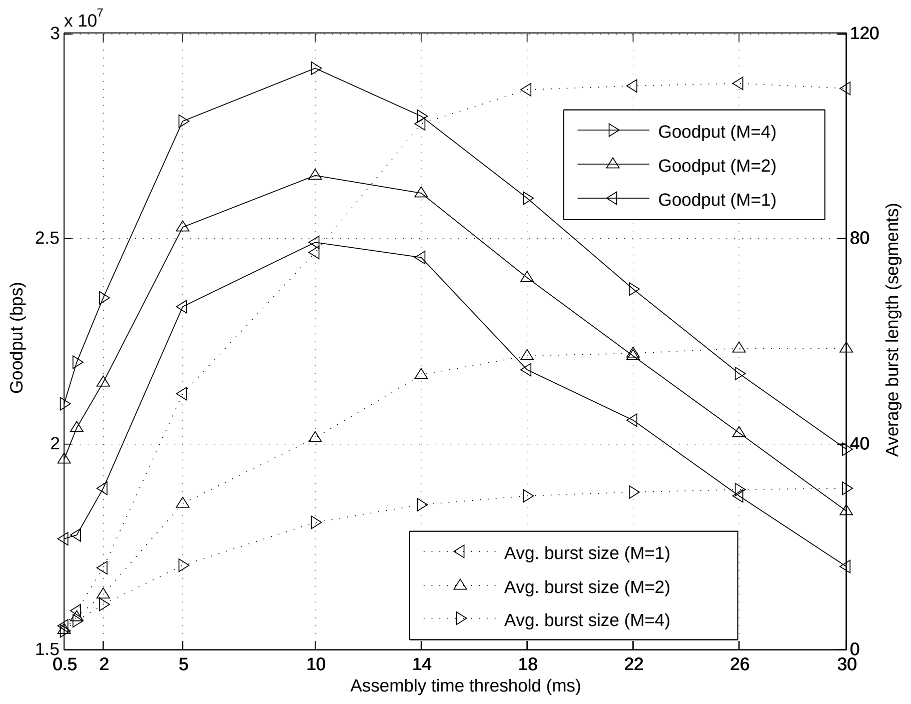


Figure 5.3: Goodput and average burst size vs assembly time threshold for egress node of $D_1 - D_4$

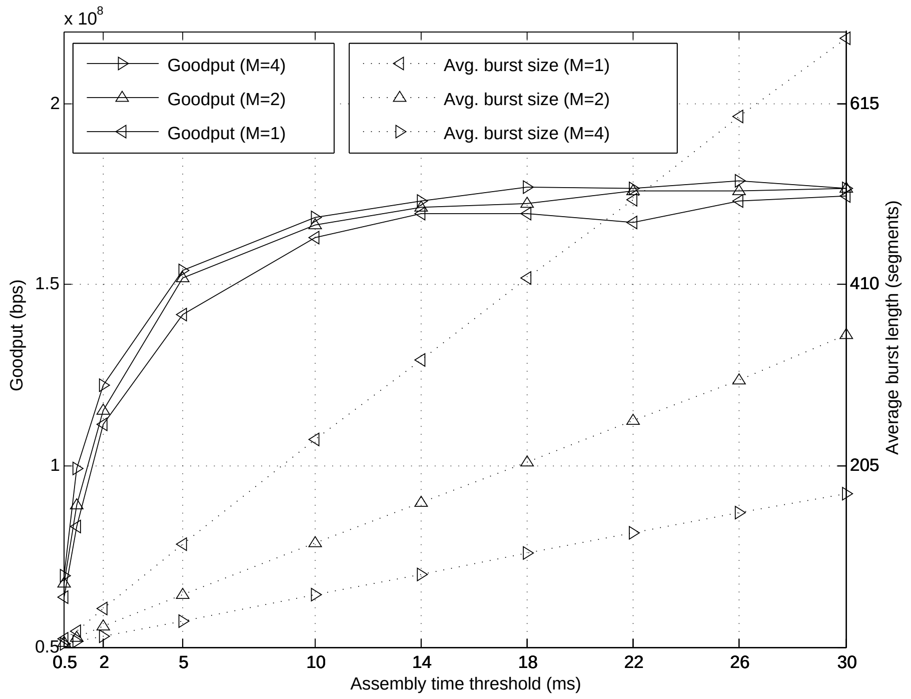
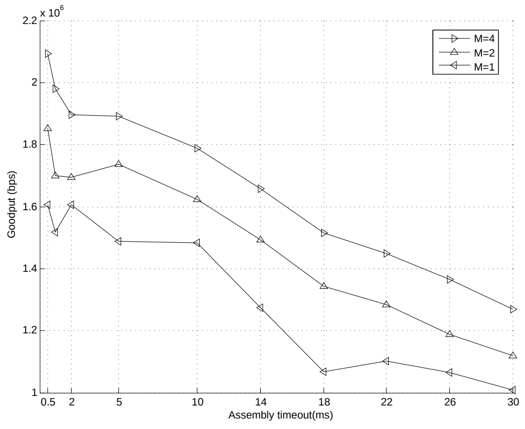
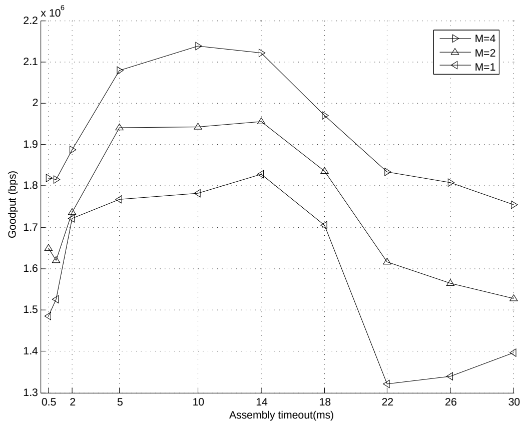


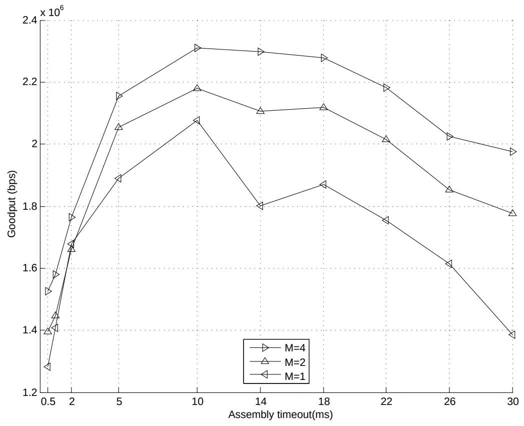
Figure 5.4: Goodput and average burst size vs assembly time threshold for egress node of $D_{17} - D_{20}$



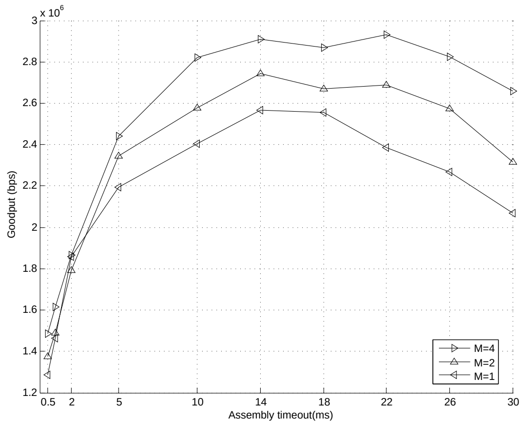
(a) $D_1 - D_4$



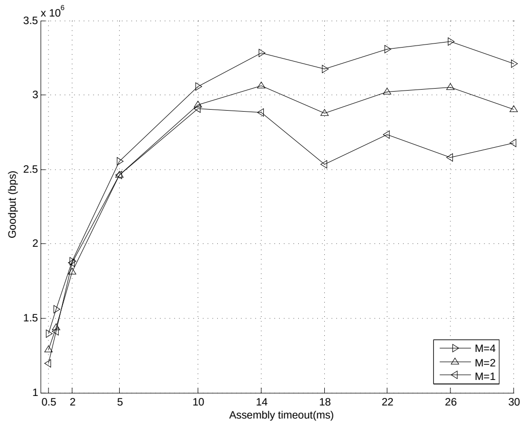
(b) $D_5 - D_8$



(c) $D_9 - D_{12}$

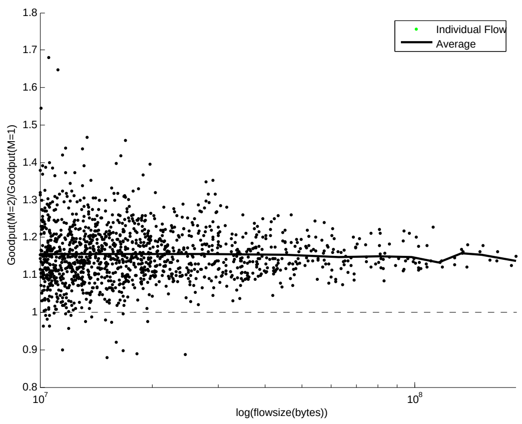


(d) $D_{13} - D_{16}$

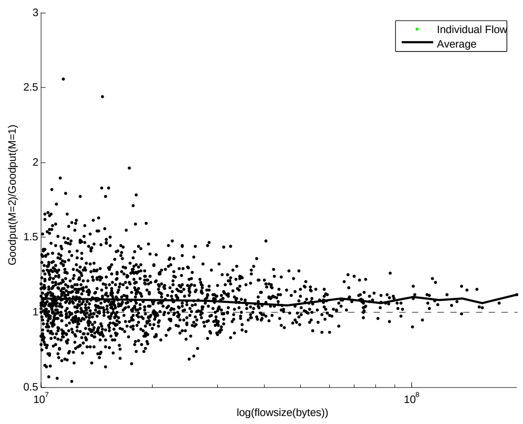


(e) $D_{17} - D_{20}$

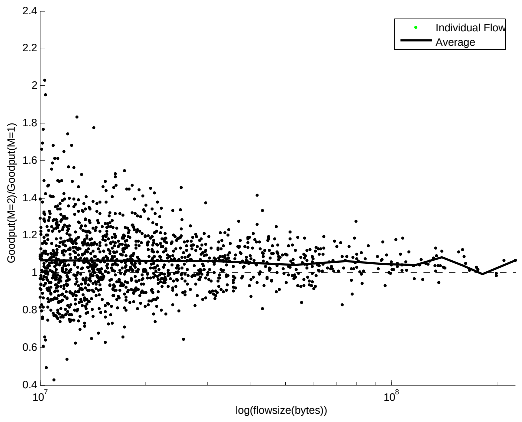
Figure 5.5: Average goodput with timer-based assembly for $N = 10$, $M = 1, 2, 4$



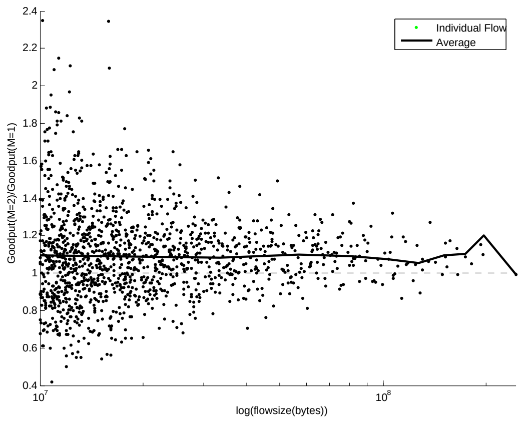
(a) $D_1 - D_4$



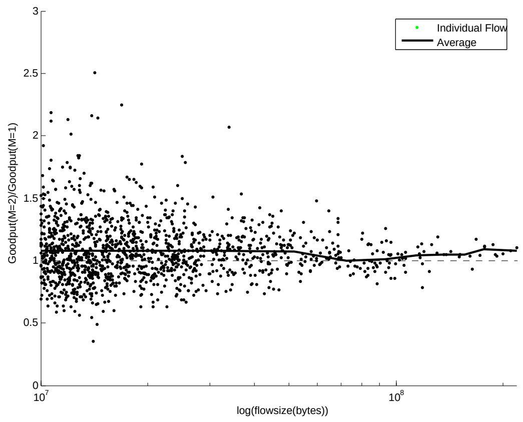
(b) $D_5 - D_8$



(c) $D_9 - D_{12}$

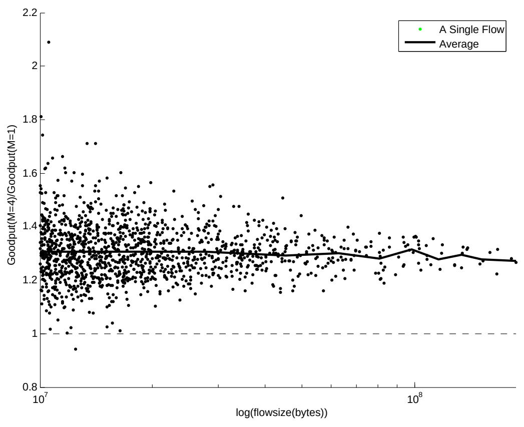


(d) $D_{13} - D_{16}$

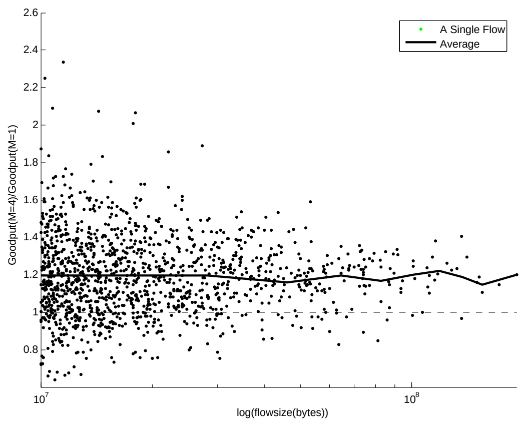


(e) $D_{17} - D_{20}$

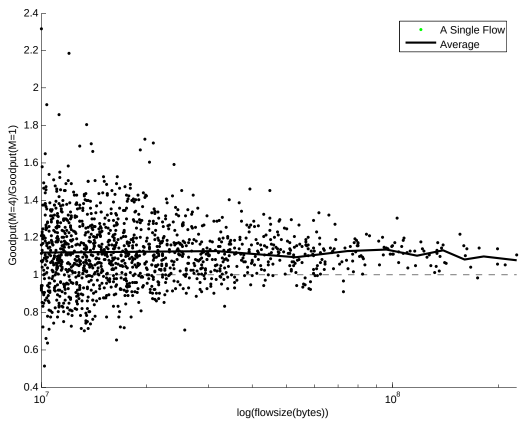
Figure 5.6: Improvement of goodputs of individual flows for $M = 2$ over $M = 1$



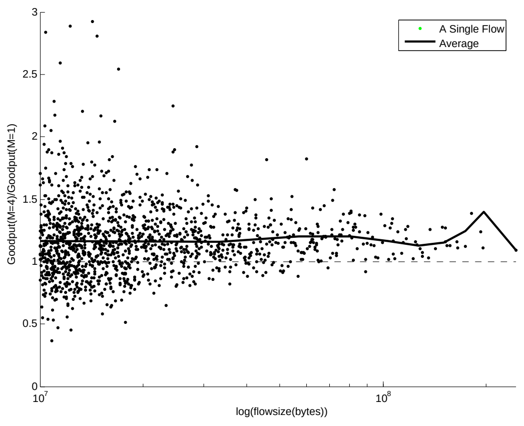
(a) $D_1 - D_4$



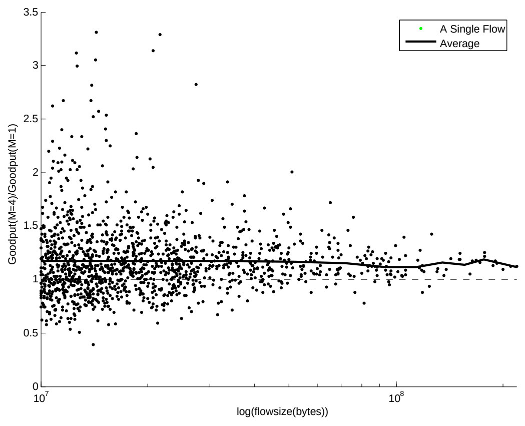
(b) $D_5 - D_8$



(c) $D_9 - D_{12}$



(d) $D_{13} - D_{16}$



(e) $D_{17} - D_{20}$

Figure 5.7: Improvement of goodputs of individual flows for $M = 4$ over $M = 1$

Chapter 6

Conclusions

In this thesis, the performance of TCP over OBS networks is studied in terms of the number of burstifiers used at the edge routers. Providing optical burst switching extensions to ns2, the nOBS simulator enabled us to make reliable TCP/IP performance evaluations in OBS networks. We used nOBS to examine TCP goodput changes for the hybrid size/timer-based algorithm over a wide range of assembly timeouts and size thresholds. We have shown that increasing the size threshold improves goodput until the maximum burst size indicated by current assembly timeout is reached. Increasing the burst size threshold further than the maximum burst size does not affect goodput as the hybrid algorithm acts as timer-based for size thresholds larger than the maximum burst size. Increasing the assembly timeout improves goodput until the minimum assembly time for the current size threshold is reached. Increasing the assembly timeout further introduces additional delays and undermines TCP performance. Timer-based assembly is shown to perform the best while size-based algorithm performs the worst, and the hybrid algorithm performed in between these two algorithms.

We have shown that under the uniform burst loss assumption, the effect of delay penalty is more severe for high burst loss probabilities, as the congestion window sizes of TCP senders cannot reach to large values due to frequent losses and the corresponding DFL gain remains incapable of overcoming delay penalty.

Increasing the number of burst assemblers per destination reduces the negative effects of synchronization between TCP flows occurring as a result of lost bursts containing packets belonging to multiple TCP flows. We show that TCP goodput is increased significantly when edge routers with multiple burstifiers per destination are used, and the goodput increases as the number of burstifiers increase. This conclusion holds for different TCP versions and different burst loss models. We argue that the edge router architecture can be designed with less number of burst assemblers than the per-flow burstification in order to reduce the complexity of managing large number of buffers while achieving nearly maximum goodput.

Bibliography

[1] M. Listanti, V. Eramo, R. Sabella, "Architectural and Technological Issues for Future Optical Internet Networks," IEEE Communications Magazine, 38, no. 9, pp. 82-92, Sep. 2000.

[2] J. Turner, "Terabit burst switching," J. High Speed Networks, Vol. 8, pp. 3-16, 1999.

[3] S. Yao, F. Xue, B. Mukherjee, S.J.B. Yoo, S. Dixit, "Electrical ingress buffering and traffic aggregation for optical packet switching and their effect on TCP-level performance in optical mesh networks," IEEE Communications Magazine 40 (9) (2002) 66-72.

[4] M.J O'Mahony, D. Simeonidou, D. K. Hunter, A. Tzanakaki, "The application of optical packet switching in future communication networks," IEEE Communications Magazine 39 (3) (2001) 128-135.

[5] S. Yao, B. Mukherjee, S. Dixit, "Advances in photonic packet switching: an overview," IEEE Communications Magazine 38 (2) (2000) 84-94.

[6] C. Qiao and M. Yoo, "Optical burst switching (OBS) - A new paradigm for an optical internet," J. High Speed Networks, Vol. 8, pp. 69-84, 1999.

[7] M. Yoo and C. Qiao, "Just-enough-time (JET): a high speed protocol for bursty traffic in optical networks," IEEE/LEOS Technologies for a Global Information Infrastructure, August 1997.

[8] J. F. Kurose, K. W. Ross, “Computer Networking: A Top-Down Approach Featuring the Internet,” Addison-Wesley, 2003, ch. 3.7 TCP Congestion Control

[9] “Network Simulator 2,” developed by L. Berkeley Network Laboratory and University of California Berkeley, http://www.isi.edu/nsnam/ns

[10] G.Gurel, O. Alparslan, E. Karasan, “nOBS: an ns2 based simulation tool for TCP performance evaluation in OBS networks,” Proc. Simulation Tools for Research and Education in Optical Networks, Oct. 2005, France.

[11] X. Yu, C. Qiao, Y. Liu, and D. Towsley, “Performance evaluation of TCP implementations in OBS networks,” Tech. Rep. 2003-13, CSE Dept., SUNY, Buffalo, NY, 2003.

[12] F. Xue, S.J. B. Yoo, “High-capacity multiservice optical label switching for the next generation Internet,” IEEE Communications Magazine, vol. 42, no. 5, pp.S16-S22, May 2004.

[13] F. Xue, Z. Pan et al., “End-to-End Contention Resolution Schemes for an Optical Packet Switching Network with Enhanced Edge Routers,” Journal of Lightwave Technology, vol.21, no. 11, November 2003

[14] J. J. He, D. Simeonidouo, S. Chaudry, “Contention resolution in optical packet switching networks: under long-range dependent traffic,” in Proceedings of OFC’2000, Paper ThU4, pp.295297, 2000.

[15] C. Raffaelli, P. Zaffoni, “Packet Assembly at Optical Packet Network Access and its Effects on TCP Performance,” Proceedings of HPSR 2003, IEEE Workshop on High Performance Switching and Routing, Torino, Italy, 24-27 June, 2003.

[16] Jingyi He, S.-H. Gary Chan, “TCP and UDP performance for Internet over optical packet-switched networks,” Computer Networks 45(4): 505-521 (2004).

[17] D. Hong, France, F. Poppe, J. Reynier, F. Baccelli, G. Petit, "The impact of burstification on TCP throughput in optical burst switching networks," ITC-18, Sep. 2003 in Berlin.

[18] S. Gowda, R.K. Shenai, K.M. Sivalingam, H.C. Cankaya, "Performance evaluation of TCP over optical burst-switched (OBS) WDM networks," Proc. IEEE ICC'03, May 2003, pp. 1433-1437.

[19] A. Ge, F. Callegati, L. S. Tamil, "On Optical Burst Switching and Self-Similar Traffic,"IEEE Communication Letters, vol. 4, no. 3, March 2000.

[20] X. Cao, J. Li, Y. Chen, C. Qiao, "Assembling TCP/IP packets in optical burst switched networks," Proc. IEEE GLOBECOM'02, November 2002, pp. 2808-2812.

[21] S. Oh, H. H. Hong, M. Kang, "A Data Burst Assembly Algorithm in Optical Burst Switching Networks," Etri Journal, vol. 24, no. 4, August 2002.

[22] A. Detti, M. Listanti, "Impact of segments aggregation on TCP Reno flows in optical burst switching networks," Proc. IEEE INFOCOM'02, 2002, pp. 1803-1812.

[23] X. Yu, J. Li, X. Cao, Y. Chen and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," IEEE/OSA Journal of Lightwave Technology, Vol. 22(12), pp. 2722- 2738, Dec. 2004.

[24] X. Yu, C. Qiao, Y. Liu, "TCP Implementations and False Time Out Detection in OBS Networks," Proc. of IEEE INFOCOM'04, 2, pp. 774-784, 7-11 March 2004.

[25] V. Eramo, M. Listanti, P. Pacifici, "A Comparison Study on the Wavelength Converters Number Needed in Synchronous and Asynchronous All-Optical Switching Architectures," IEEE Journal of Lightwave Technology, 21, no. 2, pp. 340-355, Feb. 2003.

[26] Y. Xiong, M. Vandenhoute, H. Cankaya, "Control Architecture In Optical Burst-Switched WDM Networks," IEEE Journal on Selected Areas in Communications, 18, no 10, pp. 1838-1851, Oct. 2000.

[27] J. Xu, C. Qiao, J. Li, G. Xu, “Efficient Channel Scheduling Algorithms In Optical Burst Switched Networks,” Proceedings of IEEE Infocom’03, 3, pp. 2268 - 2278, 30 March-3 April 2003.

[28] I. A. Rai, G. Urvoy-Keller, E. W. Biersack, “Analysis of LAS scheduling for job size distributions with high variance,” in Proceedings of ACM Sigmetrics, CA, USA, 2003, pp. 218-228.