Noname manuscript No.
(will be inserted by the editor)

Improving TCP Performance over Optical Burst-Switched (OBS) Networks Using Forward Segment Redundancy

Neal Charbonneau · Deepak Chandran · Vinod M. Vokkarane

Received: Wednesday, March 2, 2011; Revised:

Abstract Random contentions occur in optical burst-switched (OBS) networks because of one-way signaling and lack of optical buffers. These contentions can occur at low loads and are not necessarily an indication of congestion. The loss caused by them, however, causes TCP at the transport layer to reduce its send rate drastically, which is unnecessary and reduces overall performance. In this paper, we propose *forward segment redundancy* (FSR), a proactive technique to prevent data loss during random contentions in the optical core. With FSR, redundant TCP segments are appended to each burst at the edge and redundant burst segmentation (RBS) is implemented in the core so that when a contention occurs, primarily redundant data is dropped. We develop an analytical throughput model for TCP over OBS with FSR and perform extensive simulations. FSR is found to improve TCP's performance by an order of magnitude at high loads and by over two times at lower loads.

Keywords: Loss Recovery; TCP; OBS

1 Introduction

Optical WDM networks are a promising technology to support the next-generation Internet and its applications. Optical networks are capable of data transmission rates in the order of terabits per second. There are three switching architectures that are usually considered for optical WDM networks. In optical circuit-switched (OCS) networks, a connection request must reserve a path and a wavelength before data transmission begins. This reservation process guarantees bandwidth but increases delay. OCS also provides coarse-grained traffic granularity. A request must reserve an entire wavelength even if it does not require all of the bandwidth. These two factors mean that OCS does not support bursty Internet traffic very well. Optical packet switching (OPS) is another architecture proposed for optical WDM networks. The switching unit is a packet, as in regular electronic networks. The technology required to support OPS is not yet mature or commercially viable. Electronic networks require buffers in routers to temporarily store packets for an arbitrary amount of time when an output port is unavailable. The lack of optical buffering makes contention resolution difficult in OPS. The high data rates of optical networks also imply that switching times in OPS networks must be in the order of a few nanoseconds. Optical burst switching [1] provides a middle-ground between OPS and OCS. The switching unit in OBS networks is a burst, which is composed of a number of individual packets. The larger switching unit means that switches do not require such low switching times. OBS uses one-way resource reservation instead of two-way reservation as in OCS, which can significantly reduce transfer latency. In an OBS network, incoming traffic is assembled into a data burst (containing a number of packets) at a network ingress node. Before sending the burst, a burst header packet (BHP) is sent, which is processed electronically at each core node to set up the optical switching fabric. The BHP specifies the arrival time and duration of the burst so the switch knows when to reconfigure the switching fabric for the next burst, a technique known as just enough time (JET) [1]. After an offset time, the burst is then sent into the network and switched all-optically. Note, the resource reservation is one-way. There is no acknowledgment sent back once the BHP reaches the destination. This is demonstrated in Fig. 1.

This work was supported in part by the National Science Foundation (NSF) under Grant CNS-0626798. Portions of this paper have appeared in IEEE ANTS 2008.

Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA
E-mail: vvokkarane@ieee.org

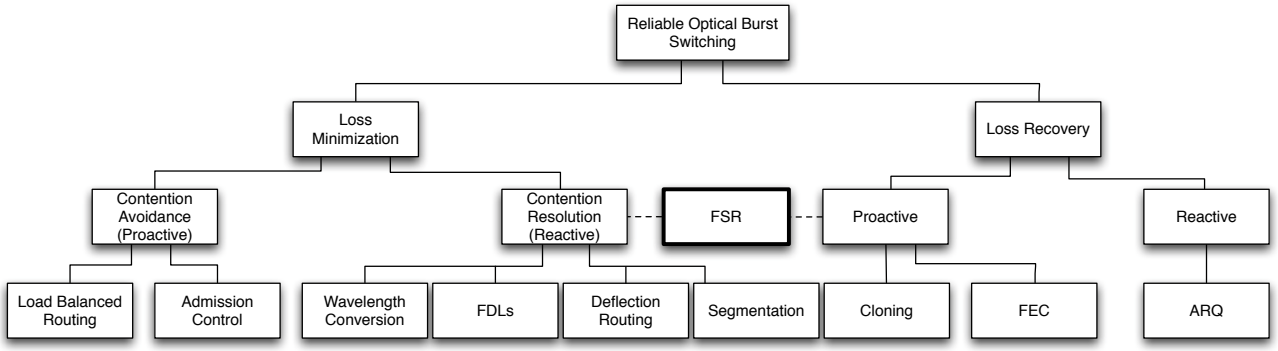


Fig. 2 Reliable OBS framework. Our proposed FSR technique is a combination of both contention resolution and proactive loss recovery.

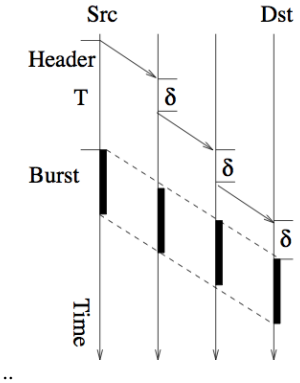


Fig. 1 Use of offset time (T) in OBS networks. The δ represents the BHP processing and switch configuration times.

This allows for bursty traffic and low delays since there is no connection set up between any ingress-egress pair. The disadvantage of OBS is random burst contentions at core nodes. Since there is no guaranteed end-to-end resource reservation and no optical buffering, there may be random contentions in the core when multiple BHPs try to schedule their bursts for the same output port at the same time. In this paper, we propose a technique to help alleviate this problem.

The majority of Internet traffic is carried over TCP, therefore it is important to study the performance of TCP over OBS networks. However, it has been shown that TCP performs poorly over OBS networks [2,3]. TCP was designed for electronic networks with buffers where a loss indicates congestion, which should cause TCP to lower its send rate by cutting its congestion window. The lowering of the send rate in electronic networks is to allow the network to stabilize since it is congested. A loss in an OBS network, however, may not indicate congestion, which implies that TCP may unnecessarily lower its send rate leading to poor performance. When the TCP sender detects a loss through triple duplicate acknowledgements, the sender will reduce its congestion window by half. In the event of a timeout, the TCP sender reduces its congestion window to one MSS. In both cases, the send rate is drastically reduced even though the loss was due to a random contention, not congestion in the network. It is important to provide some reliability in OBS networks to improve TCP's performance.

A number of mechanisms have been proposed to deal with these random contentions, including fiber delay lines (FDLs), wavelength conversion, deflection routing, burst segmentation [4], retransmission [2], and forward error correction (FEC) [5,**?**]. There have also been techniques that focus on burst assembly to reduce contentions in the core [6]. We classify these techniques in Section 2. In this paper, we propose a novel hybrid proactive loss recovery and loss minimization technique called *forward segment redundancy* (FSR). FSR combines modified burst segmentation with forward redundancy. The work in this paper is primarily an extension of our previous work on burst segmentation [4] and a simple redundancy scheme we presented in [7]. Burst segmentation is a contention resolution mechanism that divides each burst into transport units called segments made up of either a single packet or multiple packets. When a contention occurs in the core, one of the bursts is split into a head and a tail while the other burst remains unchanged. Only the head or tail is dropped instead of the entire burst, reducing packet loss. We will discuss segmentation further in Section 4. In this paper, we add redundant data to each burst and also consider a modified burst segmentation algorithm.

The primary contributions of this paper are an improved segmentation with redundancy scheme to reduce data loss caused by random contentions and also a study of its impact on TCP performance. The paper is organized as follows. We provide a classification of schemes for reliable OBS in Section 2. We then discuss details about OBS node architecture in Section 3 followed by our proposed FSR approach in Section 4. We present an analytical model of TCP throughput in Section 5. We then discuss simulation results in Section 6 and conclude in Section 7.

2 Reliability in OBS

Reliability in OBS can be categorized into two broad techniques: *loss minimization* and *loss recovery*. Fig. 2 provides a generic classification of reliability techniques. Loss minimization techniques attempt to reduce the probability of loss in the network *or* minimize the amount of data loss if a contention does occur, known as contention avoidance and contention resolution, respectively. Contention avoidance, a type of loss minimization, is the technique that aims to decrease the probability of contention in the network. One example of a contention avoidance technique is load-balanced routing [8]. Load-balanced routing attempts to send bursts on the least congested path to reduce the probability of a contention. Instead of trying to avoid contentions, a mechanism can be implemented to reduce the loss resulting from a contention after it occurs. This is known as contention resolution (also a type of loss minimization). An example of contention resolution is burst segmentation, which only drops overlapping portions of the burst during a contention to minimize loss.

Loss recovery techniques focus on handling loss instead of trying to minimize the probability of contentions, as in loss minimization. Loss recovery involves either responding to loss after it happens or sending extra data into the network to recover from a potential loss in the forward direction, known as reactive and proactive recovery, respectively. One example of a reactive loss recovery technique is burst retransmission, also known as automatic retransmission request (ARQ). Data is sent into the network under the assumption that a contention will not occur, but if it does happen the loss will be handled by retransmitting the dropped burst using an ARQ packet. A proactive approach would be sending redundant data into the network, in the form of redundant packets or forward error correction codes, so that when a contention occurs, the original data can still be recovered in the forward direction.

Our proposed technique, FSR, is a combination of loss minimization and loss recovery as shown in Fig. 2. FSR adds redundant data to each burst sent into the network, hence it utilizes proactive loss recovery. Forward redundancy by itself would not work in an OBS network where an entire burst is dropped during a contention, since that would drop all of the redundant data as well. To deal with this, we propose redundant burst segmentation (RBS), a loss minimization technique at the core. FSR is comparable to burst closing [9]. In burst cloning, a separate, duplicate burst is sent into the network. Burst cloning is similar to FSR with 100% redundancy but instead of appending the redundant data to the same burst, it is sent out as a separate burst. Not only does this involve extra control overhead in terms of BHPs, but it is not as flexible as FSR. Cloning provides only 100% redundancy. Cloning also does not handle out-of-ordering well. Consider the scenario where an original burst suffers a head drop and arrives at the egress before its clone. The clone will then cause false fast retransmits since the segments arrived out-of-order. With FSR, the redundant data is in the same burst, so after a head drop the redundant and original data can be reordered at the egress without any additional delay. FSR is a simple technique that does not require a significant amount of processing at the edge and does not have unreasonable hardware requirements (which we discuss later) like ARQ that requires large electronic buffers or FEC that requires complicated error correcting codes to be generated at high data rates.

3 Node Architecture and Burst Scheduling

In this section we provide a brief overview of the core node architecture for OBS networks. We assume that full wavelength conversion is available. We also use the terms "wavelength" and "channel" interchangeably. At the ingress nodes to the OBS network, packets that share the same destination are grouped into a single buffer, which will be used to create a data burst. The burst assembly strategy determines when these packets will be put into a burst and sent over the network. This may happen once the burst reaches a certain size and/or a timer expires. Once the burst is ready to be sent, a BHP is created. The BHP contains routing information and information about the corresponding data burst. The BHP is sent before the burst by an offset time as discussed earlier and shown in Fig. 1. In OBS networks, the wavelengths on a particular fiber are typically broken down into two subsets. If there are K wavelengths on a particular fiber, k may be used as control channels to carry only BHPs, while the remaining $K - k$ can be used to carry data bursts. A data burst can only be sent on a data channel after its BHP has been sent on a control channel.

The BHP is converted from an optical signal to electronics at each hop and processed by each core node to configure the switching fabric for the burst that will be arriving after the offset time. The offset time between the BHP and the burst must be large enough so that the BHP has time to configure all switches along the path. In Fig. 1 we can see that at the destination there is no longer a gap between BHP and data burst. When the burst is sent, it goes through the core network all-optically since the BHP has already configured the switches along its path.

Fig. 3 shows a simplified $N \times N$ OBS core node. The fibers are demultiplexed into their wavelengths, which are then mapped into data channels and control channels. The control channels are terminated at the switch control unit (SCU) while the data channels are connected to the switching matrix. The SCU contains a routing table and a forwarding table, similar to an electronic router, and it uses them to decide which outgoing data and control channels

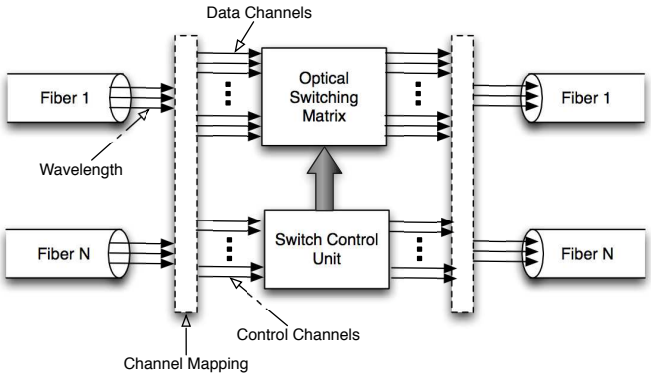


Fig. 3 Architecture of core node in OBS [10]. Each fiber is demultiplexed into its wavelengths and the data/control channels are sent to the appropriate elements. The control channels under go O/E/O conversion, while the data channels remain entirely in the optical domain.

to forward each arriving burst and BHP. Once this is determined, it configures the switch matrix (using information in the BHP), which will allow the burst to cut-through when it arrives. There may be additional components, such as fiber delay lines (FDLs) before the switching matrix and within the switching matrix.

The SCU's main component is the scheduler which performs the scheduling of the burst and BHP. There is a scheduler for each fiber. The scheduler maintains information about the data channels of that fiber. From the BHP, the scheduler determines when the burst will arrive and how long it lasts. Using this, the scheduler searches for a free wavelength on the required output port (determined by routing information) to schedule the burst on. Once this is found, the scheduler sends the information to the switching matrix (information like incoming data wavelength, outgoing data wavelength, time to switch, duration), updates the BHP and transmits the BHP the appropriate control channel (converting it back to optics), so the BHP can configure the remaining downstream switches.

A number of different scheduling algorithms have been proposed that determine which wavelength a burst is to be placed on. In this work we will use latest available unscheduled channel (LAUC) [10]. The scheduler maintains the latest unscheduled time for each data channel. We will denote this time as $LAUC_i$ for channel i. The algorithm attempts to minimize gaps, or voids, in the schedule by scheduling the data on channel j that minimizes $t - LAUC_j$, where t is the burst arrival time. LAUC requires a scan of all data channels. If no channel is available, one of the contention resolution schemes mention can be used, otherwise the data burst for the BHP must be dropped.

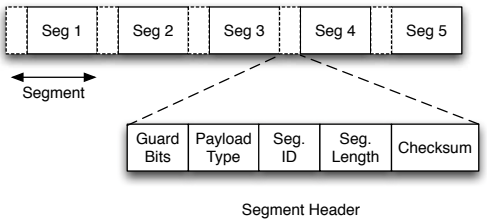


Fig. 4 Each bursts consists of a number of segments having a header and a payload.

4 Forward Segment Redundancy (FSR)

FSR is a proactive loss recovery technique combined with a loss minimization technique. It does not require feedback from the receiver or the network about loss. Implementing FSR requires a redundant segmentation technique in the core and also requires appending redundant data to the burst at the edge before being sent into the core. Both of these aspects will be discussed in detail in the following subsections. First, we discuss the basics of burst segmentation.

Burst segmentation works by dividing the data burst into a number of segments, each of which has a payload and a header as shown in Fig. 4. The segments may be made up of one or more data packets. As we discussed in the previous section, when no outgoing data channel can be found by the scheduler, the burst must be dropped. Burst segmentation drops only overlapping segments of one of the two bursts, instead of dropping one of them entirely, which will help minimize loss. If the contention involves more than two bursts, say n, then overlapping segments from $n - 1$ bursts are dropped. The optical core nodes are unaware of the segments. When a contention occurs, the core nodes do not use any knowledge of the actual segments when choosing how to segment the bursts. While this leads to sub-optimal decisions on how to segment bursts, it is important to keep the complexity in the core low to enable fast switching. We show an example of a contention in Fig. 5 (without redundant data). While discussing segmentation, we will refer to a burst already scheduled on a port as the *original* burst and the later arriving burst to the same port as the *contending* burst. In the figure, both of the bursts are scheduled for the same output port on the same wavelength. Here, we choose to drop segments in the original burst that overlap with the contending burst.

The contending region in the figure is the time duration that the bursts overlap. The BHP of each burst contains the burst length and offset time (when the burst will arrive). The LAUC scheduler at each core node maintains time availability on each wavelength. The contention region is simply the difference between the burst arrival time and $LAUC_i$ for a given channel i (as seen in Fig. 5). Normally, we would drop the entire contending burst. Instead, the original burst

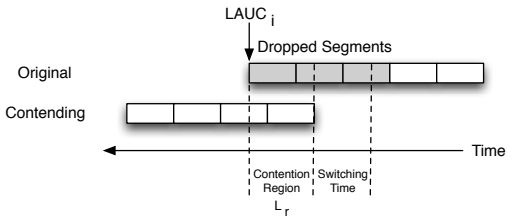


Fig. 5 Example of burst contention (without redundant data) where we drop segments from the original burst. The core is not aware of the segments in each burst, the egress node processes the incoming burst electronically to determine which segments are intact.

will be truncated. To accomplish this, the switching fabric is instructed to switch the original burst as normal, but it then switches to the configuration for the contending burst before the entire original burst has completely passed through, causing some of the original burst to be lost. The switching time required for this is also shown in the figure. This does not require any extra hardware at the core switches. After processing the BHP of the contending burst, the core node can also create a new trailer BHP to update downstream nodes with the new size of the original burst. This is unlikely to be an issue in practice, however, due to relatively few contentions under normal loads and paths with few hops.

Since the core nodes are not aware of segments, this means the network egress nodes, where burst disassembly occurs, must deal with the segments. The egress can use information in the header of each segment to determine which segments are intact. Once the valid segments are determined, using segment headers, the packets are removed from the segments and sent to the access network. The segment size is an important parameter for any segmentation technique. Larger segments mean less header overhead in the burst, but this also leads to more data loss during contention. One simple solution is to use Ethernet frames as segments. We can use information already in the Ethernet header to act as segment headers. We use this approach in the paper.

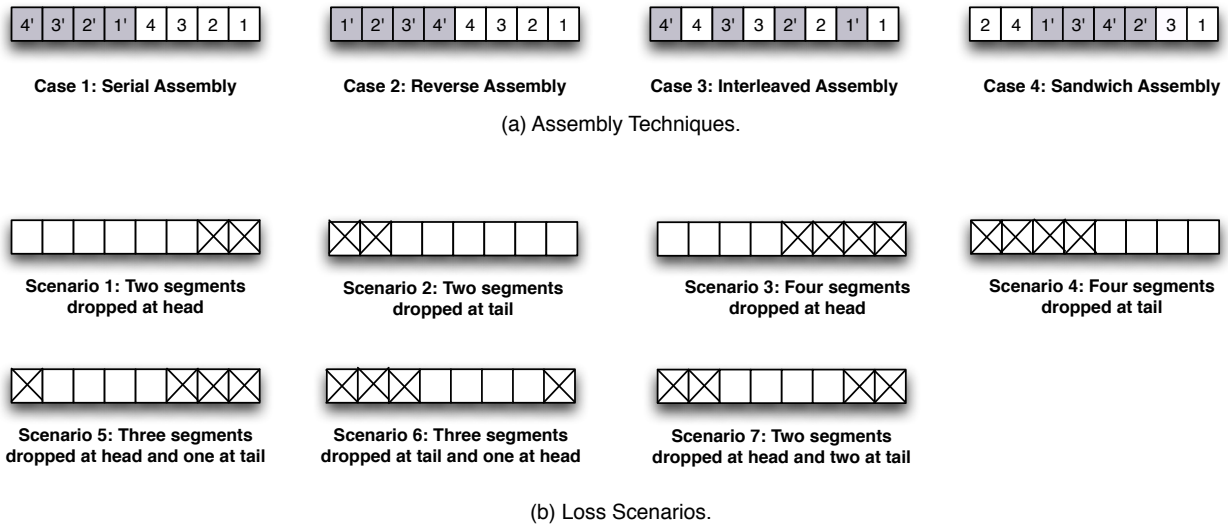
We use burst segmentation as a basis for our segmentation policies in FSR. FSR is a combination of modified burst assembly and modified burst segmentation. We discuss burst assembly first then segmentation.

4.1 Burst Assembly: Amount and Placement of Redundant Segments

In this section we discuss how redundant data is appended to each burst at the ingress nodes. This step does not add significantly complexity to edge nodes. Before each data burst is created, the packets are stored in an electronic buffer as a number of burst segments. We simply duplicate these segments in the electronic domain before or as we convert them to the optical domain. Note, this duplication is at the burst segment level, not the flow level. The complexity of adding redundant segments is not a function of the number of flows arriving at an ingress node. The ingress is responsible for including the length of the redundant data in the BHP along with the length of the burst itself (original and redundant data).

We use percentages of the original burst size to determine how much redundant data to append. For example, with 100% FSR, the burst size is effectively doubled with half of it being original data and the other half being redundant data. We add the redundant data serially to the end of the original data burst. There are many different techniques to add redundant data to the original burst, but adding it serially to the tail provides the best performance and is easiest to implement. In conjunction with RBS technique (discussed in the next section) at the core, it is possible that segments at the head and segments at the tail will be dropped. The segments nearer to the head or nearer to the tail have a higher probability of being dropped in transit than those toward the middle of the burst. If a segment is dropped at the head, we would not want the corresponding redundant segment to be located at the tail or directly after the original segment because both of these locations have a much higher probability of being dropped if there is another contention downstream. Serially appending the redundant segments provides the optimal distance between the original and redundant segments in the case one of them is dropped. The best case for an assembly mechanism would be that it is allowed to drop 50% of the new burst (assuming 100% FSR) since that 50% is redundant data and the remaining is the original data. Fig. 6(a) shows four types of possible assembly approaches. Case 1 is the type of assembly we have chosen. Case 2 is similar to case 1 except that the redundant data is appended in reverse order. Case 3 interleaves the redundant and original data while case 4 sandwiches the redundant data in between the original data. In Fig. 6(b) we show seven different loss scenarios for a burst with eight segments (four redundant, four original) with up to 50% loss. Table I shows the amount of original segment loss for each loss scenario and each assembly technique. It shows that in all scenarios, appending the data serially (case 1) is able to recover all of the original data, while the other assembly mechanisms are not. This is true for other loss scenarios not depicted in Fig. 6(b).

Because of the placement of redundant data and the possibility of both head and/or tail drops in the core, it is sometimes necessary to reorder segments at the egress node in order to prevent triple duplicates being sent to the TCP source. For example, considering our approach to append redundant data serially in Fig. 6, case 1. Assume each of the segments 1-4 corresponds to a TCP segment. If segment 1 is dropped, then the receiver will receive segments in the order: 2,3,4,1', causing triple duplicate ACKs (the cumulative ACK would



(a) Assembly Techniques.

(b) Loss Scenarios.

Fig. 6 Burst assembly strategies with 100% forward redundancy and loss scenarios with up to 50% packet loss.

Table 1 Burst Assembly: Number of original segments lost.

Scenario \Assembly	Case 1	Case 2	Case 3	Case 4
Scenario 1	0	0	1	0
Scenario 2	0	0	1	0
Scenario 3	0	0	2	0
Scenario 4	0	0	2	0
Scenario 5	0	1	1	1
Scenario 6	0	1	1	1
Scenario 7	0	2	2	0

follow directly after the triple duplicate, but when sender will still receive the triple duplicates). This loss detection is known as false fast retransmission. This problem is not difficult to resolve. When the burst arrives at the egress, all of the segments must be scanned to determine which are intact before the individual packets are removed. During this process, the egress can also reorder the segments. Again the reordering of segments is at the burst level, not at the TCP flow level, so the complexity is independent of the number of flows multiplexed into a single burst. The proper order of burst level segments will also ensure that all flows in the burst have properly ordered packets.

So far we stated that we generate redundant segments and append them to the end of the burst. We have assumed that 100% FSR is used so there is no choice in the selection of segments to duplicate, they are all duplicated. With different redundancy percentages, however, we must determine which segments get duplicated and appended to the burst. We leave this as a topic for future work and assume that an optimal algorithm is used. That is, we can recovery *any* of the X% of segments lost where X is the redundancy percentage. We will show later, 100% redundancy has the best performance and in that case there is no need for an algorithm to determine which segments to duplicate.

4.2 Redundant Burst Segmentation (RBS)

In the proposed redundant burst segmentation (RBS) technique, we modify the rules that define which burst to segment during a contention. We define two different burst priorities that affect how the segmentation works. A burst that contains redundant data is considered low priority, P_1, while a burst with no redundant data is considered high priority, P_0.

RBS requires only one additional field to stored in the BHP and one additional variable to be stored by the LAUC scheduler for each data channel. The hardware requirements are the same as segmentation in the core. A traditional BHP may contain routing information, payload, offset time, data burst length, and data channel carrying the burst. RBS adds a field that stores the length of the redundant data. This field is initialized by our burst assembly algorithm at the ingress node.

The core nodes use the following RBS policies:

- *Combined head and tail drop (HTD)*: In this case, a portion of each burst is dropped. We drop a part of the tail of the original burst and a part of the head of the contending burst. How much of each is determined when the contention occurs (discussed in this section).
- *Head drop (HD)*: The head of the contending burst is dropped so that there is no longer contention.
- *Tail drop (TD)*: The tail of the original burst is dropped so that there is no longer contention.
- *Drop contending burst (DC)*: In this case, the entire contending burst is dropped and the original burst is scheduled.

The choice of the policy depends on the priority of each burst, the length of each burst, and the length of the redundant data. When we discuss the length of a burst, we assume

Table 2 RBS policies.

Contention Scenario	Priority Relationship	Length vs. Overlap	Policy
1	$R_o > 0$ *and* $R_c > 0$	$L_c > L_r$	HTD at *midpoint*
2	$R_o > 0$ *and* $R_c > 0$	$L_c \leq L_r$	HTD at T_m
3	$R_o = 0$ *and* $R_c = 0$	any	HTD at *midpoint*
4	$R_o > 0$ *and* $R_c = 0$	any	TD
5	$R_c > 0$ *and* $R_o = 0$	any	HD

the units are in the time domain. We can convert from bytes to time by using the bandwidth of the channels. The HTD policy must also determine how much of each burst must be dropped. There are five cases shown in Table II. Before explaining Table II we will use the following notation. Every parameter in this list can be found in the BHP of the contending burst.

- L_c: length of the contending burst.
- R_c: the length of the redundant data in the contending burst.
- T_c: The time at which the contending burst will arrive.

As we discussed earlier, we assume each node uses LAUC to schedule bursts. For each wavelength, LAUC maintains the latest available time, we will denote this as $LAUC_i$, the latest available time of wavelength i. In addition to $LAUC_i$, we add one more variable to the LAUC scheduler called R_i, which represents the length of the redundant data of the last burst scheduled on wavelength i. The scheduler will store this value (from that burst's BHP) when the burst is successfully scheduled.

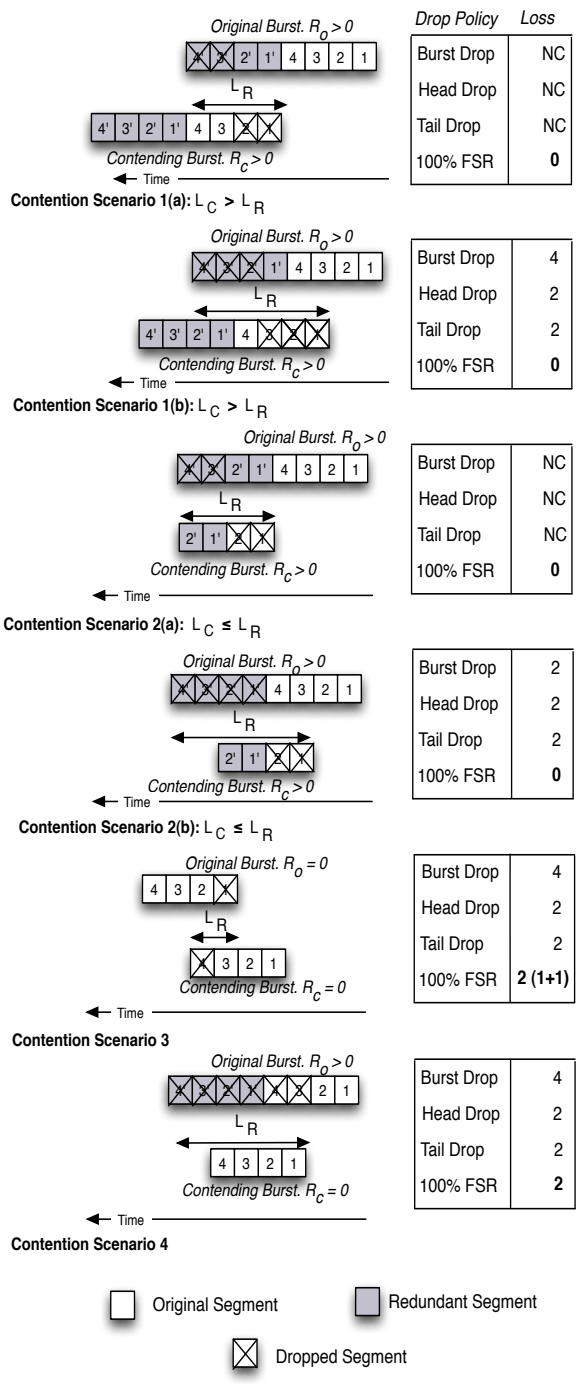
Now we define the variables that must be calculated for RBS. These values are calculated by the scheduler in the switch control unit every time a BHP arrives. The LAUC scheduler will select the wavelength with the minimum contention region to schedule a burst if no free wavelengths are found. Let this selected wavelength be i. Let $L_r = LAUC_i - T_c$ be the length of the contending region shown in Figure. 5. We define the *midpoint* to be the middle of the contention region. This can be calculated as $\textit{midpoint} = LAUC_i - (T_c - LAUC_i)/2 = LAUC_i - L_r/2$. We define the variable L_m to be equal to half the size of the contending burst, $L_m = (T_c + L_c)/2$ and T_m to be the middle of the contending burst, $T_m = L_m + T_c$.

Given the definition of these variables, we will now explain our segmentation rules in Table II with the help of Fig. 7. In the figure, we show the redundant data as shaded segments and the original data as white segments. Next to each contention scenario, we display the number of segments lost if no contention resolution were used (Burst Drop), if traditional segmentation were used (Head/Tail Drop), and if our proposed FSR were used. *NC* means no contention would occur. We assume that FSR uses 100% redundancy. The point of the tables in the figure is to show that FSR never causes more loss than the other techniques. There are cases where FSR results in contention while the others do not, but in these scenarios no original data is lost. These are simplified examples with only one or two segments dropped. In an actual network a combination of these scenarios may occur for any given burst with a greater number of segments lost. We later show that FSR provides significant performance improvements so the extra processing required by FSR is a good tradeoff.

In Fig. 7 we show two cases of Contention Scenario 1 from Table II. In 1(a), the contention is caused by the redundancy itself. In other words, if we did not use FSR, there would be no contention. This has no negative impact, however, we still lose no data. Contention Scenario 1(b) shows a scenario were FSR results in no data loss while the others do. The midpoint of L_r is determined and combined head/tail drop is performed for FSR. We also show two cases for Contention Scenario 2, where one case is a result of the redundancy and the other case results in less data loss. In these two cases, HTD is performed at the middle of the contending burst instead of the middle of L_r. Again, even when FSR itself causes contention, there is no negative impact on performance. Contention Scenario 3 in the figure shows the case where neither burst has any redundant data. We can see with the HTD scheme each burst loses one segment whereas in traditional segmentation one of the bursts would lose two while the other would lose one. This may result in more fairness than traditional segmentation. Lastly, we show Contention Scenario 4 where one burst has redundant data and the other does not. Here, FSR performs the same as traditional head or tail drop. We omit Contention Scenario 5 since it is the same as 4 but with head drop instead of tail drop.

There is one final scenario not depicted in Table II. For contention scenario 2, we found that if the original burst is large and the contending burst is small, splitting the bursts according to the center of the smaller contending burst may result in loss of many original packets from the original burst. We check for this by determining if $L_m > R_o$ and if so we simply drop the contending burst.

In all cases, FSR performed as good as or better than traditional burst segmentation. Even though the act of adding redundant data into the core may cause additional contentions, these scenarios do not lead to any additional data loss that would not occur otherwise. Given this and the fact that FSR is a simple modification to traditional segmentation, it makes sense to use FSR instead of traditional segmentation. We will discuss the impact of redundant data in more detail in the simulation section. We will show that there is a point where too much redundant data begins to degrade perfor-



Drop Policy	Loss
Burst Drop	NC
Head Drop	NC
Tail Drop	NC
100% FSR	**0**

Burst Drop	4
Head Drop	2
Tail Drop	2
100% FSR	**0**

Burst Drop	NC
Head Drop	NC
Tail Drop	NC
100% FSR	**0**

Burst Drop	2
Head Drop	2
Tail Drop	2
100% FSR	**0**

Burst Drop	4
Head Drop	2
Tail Drop	2
100% FSR	**2 (1+1)**

Burst Drop	4
Head Drop	2
Tail Drop	2
100% FSR	**2**

Fig. 7 Illustrations of contention scenarios for FSR with modified burst segmentation. Each contention scenario matches the scenarios in Table II. The tables show the amount of actual data loss (in number of segments) for different contention resolution techniques.

mance, but we can choose redundancy levels before this point that still significantly improve performance.

Again, the hardware requirements for RBS are similar to traditional burst segmentation, which is accepted as a feasible contention resolution technique. Our new approach adds the calculation of four variables, which can be done efficiently in hardware, and the storage of one additional field in the LAUC scheduler. The actual segments are still transparent to the optical core.

5 Analytical TCP Throughput Model

In this section we introduce an extension to the TCP SACK over OBS throughput model in [11]. We assume a burst contains only TCP segments and each burst segment corresponds to one TCP segment. We model the placement of redundant data for bursts containing TCP segments. The model in [11] assumes that when a contention occurs, the contending burst is dropped entirely. In the case of FSR, there are different scenarios for burst contention. There may be no original data loss, only redundant data. Second, some original and redundant data may be lost. Lastly, the entire burst may be dropped. We will incorporate these possibilities into the model in [11] to accurately model TCP SACK with FSR over OBS.

We use the same terminology and variables defined in [11]. A TCP sending *round* starts when TCP sends its window and ends when it receives the first acknowledgement. We assume the time it takes to send a window is less than the RTT, so the duration of the round is *RTT*. We also assume that each received TCP segment generates an acknowledgement ($b = 1$ in [11]). The model is based on defining the operation of TCP during what are called *triple duplicate periods* (TDP) and *timeout periods* (TOP). A TDP period is the interval between two consecutive triple duplicate loss scenarios or periods starting after or ending in a timeout loss scenario. The TOP is the duration of a series of timeout events. These periods are shown in Fig. 8, where W is the congestion window size. By determining the amount of data sent in these periods and how long these periods last, the throughput can be obtained. The analysis in [11] looks at two separate cases, one where the number of TCP segments in a burst is less than TCP's maximum receiver window and one where TCP's maximum window limits the number of TCP segments that go into a burst. We consider the first case here. The second case is a simple extension of the first and is straightforward to derive once the first case is derived.

We use the following notations:

p_c: burst contention probability.
p_l: probability a contention leads to data loss (but not burst dropping).
p_{td}: probability of a burst contention leading to a TD loss indication.
p_d: probability a burst is dropped entirely.
S: number of TCP segments assembled into a burst.
B: TCP throughput.
W_m: maximum TCP window size (in segments).

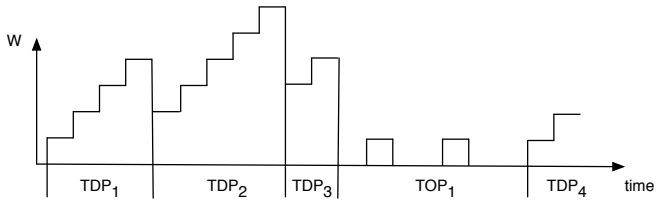


Fig. 8 Illustration of triple duplicate periods and time out periods for TCP.

$|TDP|$: duration of a triple duplicate period.
Y: number of TCP segments sent during a TDP.
$|TOP|$: duration of a TOP period.
H: number of TCP segments sent in $|TOP|$.
W_x: sending window size of last round in a TDP.
Q: probability that a loss indication ending a TDP is a TO.

The TCP throughput is obtained by:

$$B = \frac{E[Y] + Q \times E[H]}{E[TDP] + Q \times E[TOP]}. \quad (1)$$

The input to the model is the burst contention probability (p_c), number of TCP segments in a burst (S), the RTT (including burst assembly times), and TCP's retransmission time out (RTO). Based on these parameters and the behavior of TCP SACK, the expected values in the above equation are obtained.

As previously discussed, the contention probability is not the same as the burst drop probability nor is it equal to the probability of data loss. We first define p_l, which is the probability that some original data is lost during a contention. In this model we assume that 1) each burst when it is sent uses 100% FSR and 2) that during a random burst contention, the point of segmentation is uniformly distributed over the length of the burst. We model the first contention scenario described in Table II because it happens most frequently. Each burst can be thought of as having two regions, one containing redundant data and one containing original data. Data loss is only possible if the contention point is larger than either of the regions. Assuming each region has equal size (100% FSR), this occurs one half of the time ($0.5p_c$). Because the other bust also gets segmented, we only drop segments that cover half of the contending region, which means that the actual probability of a contention causing loss is expected to be $p_l = 0.25p_c$. This probability can be modified for any level of redundancy. To derive p_{td}, we use similar reasoning as [2]. Consider the round where a TD loss even occurs. The window size is $E[W_x]$ and the TCP segments making up the window are distributed across several bursts ($\frac{E[W_x]}{S}$ bursts). Only the first burst to lose any original data will trigger a TD event. TCP SACK will not leave fast retransmission until all TCP segments are acknowledged, so any other bursts that lose original data in this round will not trigger additional TD events. Since burst contentions in OBS are independent events, the average number of bursts experiencing data loss but not triggering TDs is $(\frac{E[W_x]}{S} - 1)p_l$. The ratio of the number of contentions leading to TD events to the number of contentions that do not lead to TD events is equal to the ratio of the probability of a TD event to the probability of a non-TD event, which is derived from [2]:

$$\frac{1}{(\frac{E[W_x]}{S} - 1)p_l} = \frac{p_{td}}{p_l - p_{td}}. \quad (2)$$

This allows us to derive p_{td} as:

$$p_{td} = \frac{p_l}{(\frac{E[W_x]}{S} - 1)p_l + 1}. \quad (3)$$

In [11], p_c is the probability of burst contention *and* the probability of a TD event. We can now use p_{td} in place of p_c in [11].

We will now discuss the required changes to the equations in [11] for the throughput in (1). (2) in [11] derives an equation for $E[Y]$. Since they assume the entire burst is lost during contention, the equation contains the '$-S$' term. With segmentation, only a portion of the burst is lost. We can expect, on average, that $\frac{S}{2}$ segments are lost. (4) in [11] has p replaced with p_{td}. This leads to a new derivation of (5) in [11] as:

$$E[Y] = \frac{3}{2}E[W_x] + \frac{S}{p_{td}} - \frac{S}{2}. \quad (4)$$

(8) in [11] is also updated with the term '$-\frac{S}{2}$' instead of '$-S$'. These changes lead to an updated value of $E[W_x]$ (10) in [11], which becomes

$$E[W_x] = \frac{8}{3} + \frac{8}{6}\sqrt{4 - \frac{3}{2}(S - \frac{4S}{p_c})}. \quad (5)$$

We can now derive the final equation of $E[Y]$ from (4) and (5). We use the same derivation of $E[|TDP|]$ as [11] except we use our derived $E[W_x]$.

We must now define the expected values dealing with TO losses. To do this, we must define the probability of a burst being dropped given segmentation is used. Again, we assume that the contention point is uniformly distributed across the entire burst. The entire bust is dropped if a tail drop occurs and the contention point is at the head of the burst, or if a head drop occurs and the contention point is the tail of the burst. We can then define p_d as the probability

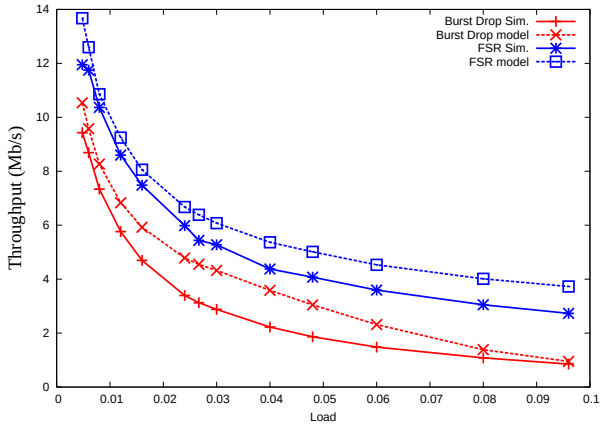


Fig. 9 Verification of the analytical model. The FSR model is the model presented in Section 5. The Burst Drop model is the model presented in [11].

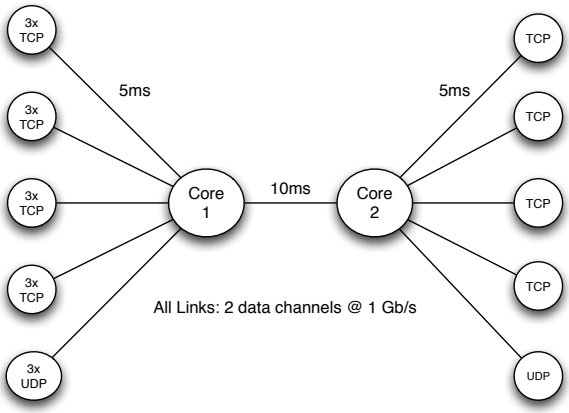


Fig. 10 Simulation Topology.

of the contention point being either the first or last segment, as

$$p_d = \frac{1}{S} p_c. \tag{6}$$

We use the same values of $E[H]$, $E[|TOP|]$, and Q as in [11] except that we replace p with p_d.

The final expression for B can now be defined as:

$$B = \frac{\frac{5}{2}E[W_x] + \frac{2S}{p_c} - \frac{3S}{2} + (\frac{1}{S}p_c^{E[W_x]/S-1})\frac{\frac{1}{S}p_c}{1-\frac{1}{S}p_c}}{RTT(0.5E[W_x]+1) + (\frac{1}{S}p_c^{E[W_x]/S-1})RTO\frac{f(\frac{1}{S}p_c)}{1-\frac{1}{S}p_c}}, \tag{7}$$

where $E[W_x]$ is defined in (5). If the contention probability, p_c, is very small, the throughput can be estimated as:

$$B \approx \frac{W_m}{RTT}. \tag{8}$$

6 Numerical Results

In this section we discuss the accuracy of the analytical model as well as provide extensive simulation results for FSR. We begin with verification of the analytical model.

6.1 Analytical Model Verification

In this section we verify the analytical model for TCP SACK throughput presented in the previous section. To verify the model we create a simple dumbbell topology with a single TCP SACK source sharing an OBS core node with a UDP source (similar to Fig. 10). The UDP source generates bursts according to a Poisson process with an arrival rate λ and an exponentially distributed length with mean $\mu = S$ (i.e., the mean burst size of the generated UDP burst is equal to the number of TCP segments per burst). We increase the average arrival rate to cause higher levels of contention. This contention probability is then used as input to our model. We can verify the results with only a single TCP source because with random burst contention, all TCP sources would be affected equally, hence one TCP source can be used as a representative of all TCP sources with the same implementation [11].

We use an access network with a bandwidth of 30Mb/s. This, along with a 4ms burst assembly timer, ensure that each burst contains (at most) 15 TCP segments ($S = 15$). The round trip time, including access networks, burst assembly time, and the core network is 52ms. The TCP sender's maximum window size, W_m, is set to 100 TCP segments.

The results for varying loads ($\frac{\lambda}{\mu}$) of incoming UDP bursts are shown in Fig. 9. We compare our model presented in the previous section for FSR to the burst drop model presented in [11]. The figure shows that FSR has better performance than traditional burst dropping for all loads and that our model is accurate, though it slightly underestimates loss. The addition of redundancy into the network results in more contentions for FSR than burst dropping. For example, at the highest load shown in the figure, OBS with FSR had a contention probability of 16% while OBS with burst dropping had a contention probability of 8.5%. The higher contention probability does not hurt performance. As shown in the figure FSR consistently outperforms burst dropping. We explore the impact of redundant data further in the next section.

6.2 Simulation Results

This section describes the simulation results. The simulation section is organized as follows. We first discuss the performance of FSR compared to traditional segmentation and no contention resolution (i.e. the entire contending burst is dropped if there is contention). We then look at the impact of burst size on FSR. Lastly, we generalize the topology and look at FSR's performance over multiple hops instead of a single bottleneck link.

Simulations were performed using ns2 and the OWns module. The simulation topology used is shown in Fig. 10. We use a simple topology with a single bottleneck link between two core nodes shared by a number of edge nodes. We will generalize this in Section 6.2.3. Each of the nodes on the left (the ingress nodes) sends to the corresponding node on the right (the egress nodes). There are four ingress nodes with three attached TCP SACK sources each and one edge node with three UDP sources (as shown in the figure). We use the UDP flows to generate congestion in the network. Varying the UDP send rate allows us to simulate networks with different loads (since UDP does not have congestion control). As noted in the topology, each link has two data channels with a rate of 1Gb/s each. The total capacity of the bottleneck link is 2Gb/s. We vary the combined UDP send rate from 150Mb/s up to 1.8Gb/s to simulate various levels of load over the network to observe the impact on the TCP senders.

The TCP flows have a packet size of 1KB and the receiver window is set high enough so that it does not limit send rates. Each TCP flow sends a 100MB file using FTP and each of the UDP flows generate Pareto traffic. The Pareto traffic has a burst time of 500ms and an idle time of 500ms with a shape parameter of 1.5 (resulting in a Hurst parameter of 0.75).

The burst assembler uses mixed-timer-threshold policy with a max burst size of 200KB and a 4ms timer. A good value (200KB) for the burst size is determined through simulation is the following subsections. We chose a trade-off between loss performance and end-to-end delay as will be discussed later.

The bursts containing UDP datagrams are treated differently than the bursts containing TCP. The UDP bursts are considered low priority, as though they contain redundant data, but they do not have redundant data. This means that when UDP bursts contend with TCP bursts, it is likely contention scenario 1 or 2 from Table II will occur. We do not give UDP bursts higher priority even though they contains no redundant data because we assume UDP traffic is loss tolerant. In all the simulations the redundant data is appended serially to the tail of the burst if FSR is enabled. We assume the segment size is one Ethernet frame. We also perform reordering of the redundant and original segments at the egress, as discussed in Section 4.

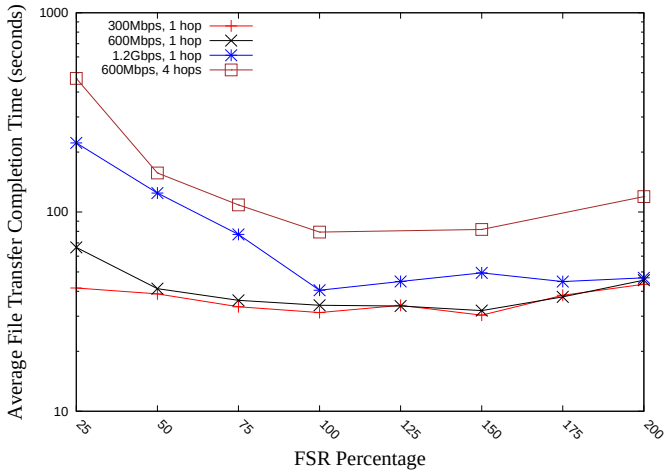


Fig. 12 Comparison of completion time vs. FSR percentages.

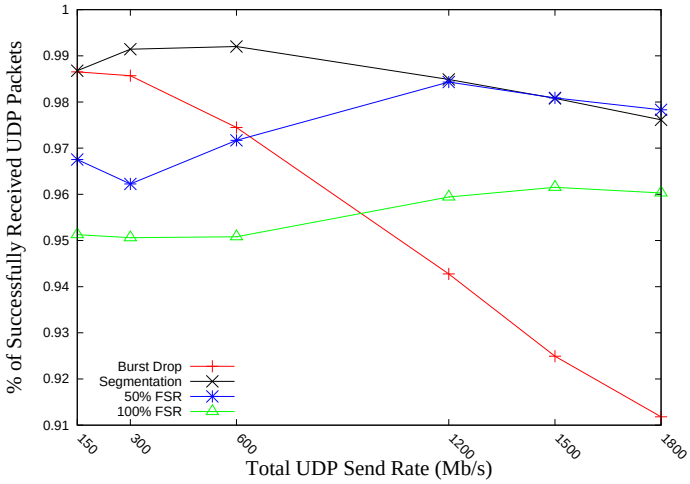
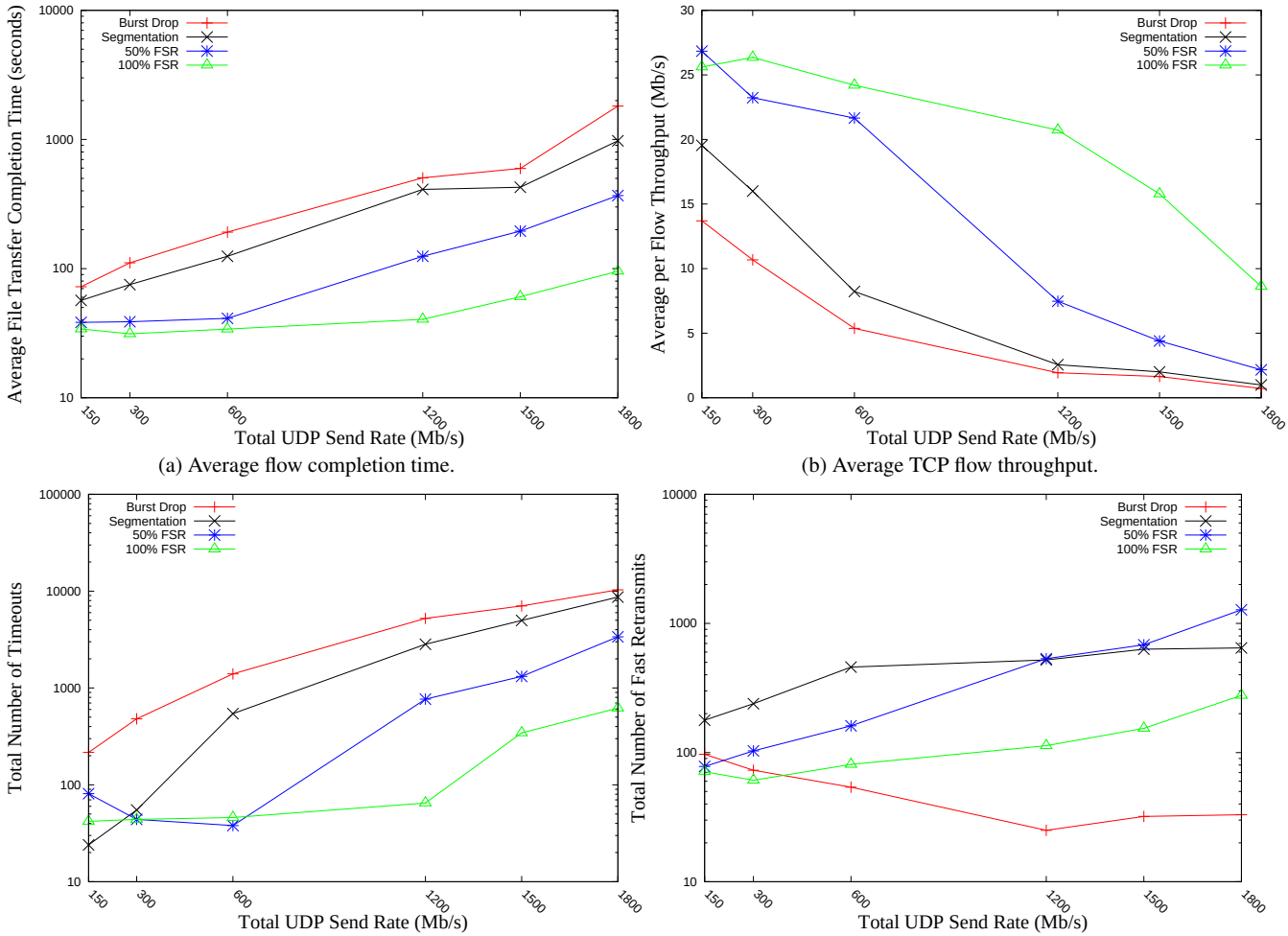


Fig. 13 Impact of redundancy on background UDP traffic.

6.2.1 FSR

In this section we compare the performance of FSR with varying degrees of redundancy, 50% and 100%, to regular segmentation and to no segmentation (burst drop policy). We use background UDP traffic to cause data loss as discussed previously.

Fig. 11(a) shows the average flow completion time to send a 100MB file among each the 12 TCP flows. There is a clear distinction between different levels of redundancy with 100% redundancy performing the best. At high send rates there is an order of magnitude (10-20 times) performance difference between 100% redundancy and both segmentation and burst drop. From Fig. 11(c) we can observe that FSR greatly reduces the number of timeouts experienced by the TCP flows. At 1800Mb/s total UDP send rate there is over an order of magnitude more timeouts with traditional



(a) Average flow completion time.

(b) Average TCP flow throughput.

(c) Total number of timeouts experienced across all TCP flows.

(d) Total number of fast retransmissions experienced across all TCP flows.

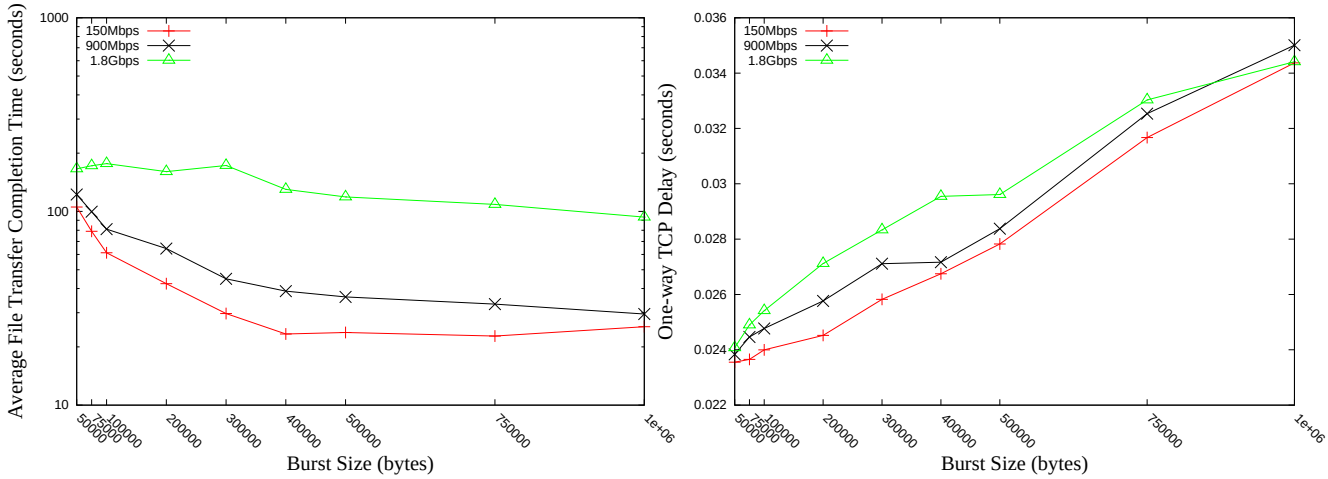
Fig. 11 Performance comparison of FTP file transfers, each of the 12 flows sending 100MB.

segmentation and burst drop compared to 100% FSR. This is because these techniques are not able to prevent enough loss at high UDP rates which results in small window sizes. With the smaller window sizes, the TCP sender's entire congestion window will fit into a single burst and a burst loss will usually result in an timeout.

While FSR reduces the total number of timeouts, it also increases the number of fast retransmissions compared to burst drop as seen in Fig. 11(d). However, the difference between number of fast retransmits for 100% FSR and no segmentation is not as high as the difference in timeouts. The increase in fast retransmissions is caused by segmentation when only some packets in a burst are dropped instead of the entire burst. Traditional segmentation has the same issue with fast retransmissions as FSR as shown in the figure. 100% FSR is able to prevent more data loss and therefore results in fewer fast retransmissions compared to 50% FSR and traditional segmentation.

One major concern about sending redundant data in the network is how this redundant data impacts performance. In other words, is it possible to send so much redundant data into the network that it begins to degrade performance instead of improve it. For up to 100% FSR, based on the results so far, it is clear that the redundant data does not degrade performance. The redundant data does indeed cause more burst contentions, but as we showed in Fig. 7, many result in no data loss.

In order to determine when redundancy does begin to degrade performance, we ran simulations for greater than 100% redundancy. Fig. 12 shows comparison of flow completion time versus different FSR percentages for different UDP send rates. The graph shows that as FSR percentage reaches 100% it seems to hit an optimal value. After 100% the completion time begins to increase again. Note, we ran simulations for FSR percentages higher than 200% and the upward trend continues. This seems to suggest that there is an optimal value for the FSR percentage around 100%. The graph also shows this is the case for a multi-hop network (discussed in subsection 6.2.3). This clearly shows that there is a point where too much redundancy begins degrading per-



(a) Average flow completion time using vs. burst size.

(b) One-way TCP delay vs. burst size.

Fig. 14 FSR with varying maximum burst size.

formance, but our choice of 100% redundancy is before this point.

We also investigate the impact of redundancy on the background UDP traffic. The background UDP traffic does not contain redundant data, but the bursts are segmented if they encounter a contention. These types of bursts may carry lower priority or loss-tolerant traffic. Fig. 13 shows the percentage of successfully delivered UDP packets for different levels of redundancy and different loads. As we expect, burst segmentation should provide the best performance for background traffic because no redundant data is introduced into the network and when contention occurs only some segments of a burst are dropped. We can also see that FSR with 100% redundancy does cause some additional UDP packet loss compared to traditional segmentation because of the extra redundant data. This small amount of background traffic loss, however, corresponds to a large performance increase for the TCP traffic as we have seen in the previous figures (Fig. 11 (b)). The figure also shows that with burst dropping, the loss rate of background UDP traffic increases as load increases. At high loads FSR is actually better even with extra contentions caused by redundant data. This is because with FSR all bursts are segmented, so even though there are more contentions, only some segments are dropped instead of the entire burst. While the background traffic's packet loss rate increases slightly, we believe this is a fair tradeoff to gain significant improvement of TCP's performance.

6.2.2 Burst Size

In this section we examine the impact of burst size on FSR. We run simulations varying the burst size at a UDP send rate of 150Mb/s, 600Mb/s, and 1800Mb/s. We use the same settings as the previous simulations except that we increase the burst assembly time to 10ms to ensure that the maximum burst size is being used.

Fig. 14(a) shows the average completion time. The figure shows that as burst size increases, the completion time decreases. Larger bursts are able to send more data resulting in fewer contentions and lower completion times. As a result of the larger bursts, the one-way delay TCP experiences through the network increases as shown in Fig. 14(b). The increase in delay is a combination of longer time being spent in the burst assembler and also the longer transmission delays for larger bursts.

6.2.3 Multiple-hop Paths

So far we have evaluated FSR's performance over a single-hop network. This means that each burst will only undergo a single contention at most. To make the results more general, we simulate bursts traversing multiple hops. We can think of this as taking a path out of a larger network and analyzing the performance of bursts on this path. To do this, we extended the topology in Fig. 10 to include multiple hops where each hop is shared by different UDP source and destination pairs. The settings and number of flows are the same (there are three UDP flows for each hop) and the total UDP send rate is fixed at 600Mb/s. Fig. 15 plots the average flow completion time. As the path length increases, there is a greater chance of contention because contention can occur at each node in the path. As expected, the completion time increases as the path length increases due to more contentions. FSR still has the best performance compared to the other techniques.

The simulation results presented are also valid for larger networks as well. We ran simulations over NSFnet with variable bit rate UDP traffic between all source-destination pairs and eight source-destination TCP pairs with three TCP flows

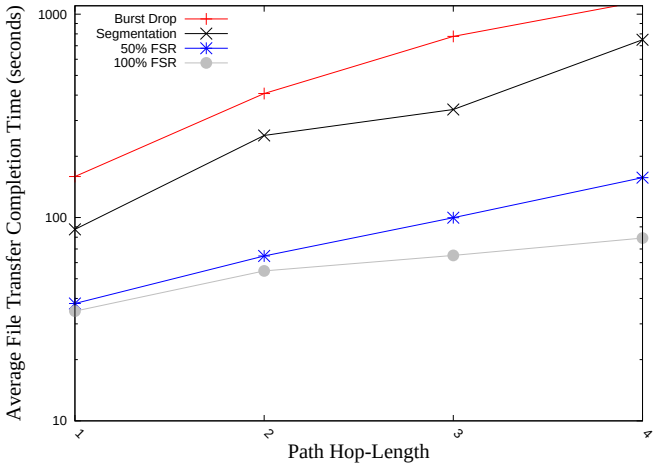


Fig. 15 Comparison of performance of FTP file transfers, each of the 12 flows sending 100MB, over multiple hops.

each. With this topology we observed similar results. We use a simple topology to reduce simulation run-time.

6.3 Discussion

We have shown that our proposed FSR technique with 100% redundancy provides better performance compared to burst dropping and traditional segmentation even with relatively high levels of traffic in the network. There is a point where too much redundancy begins to degrade performance, but we have shown through simulation that this is not the case for 100% FSR. The implementation complexity is essentially equivalent to that of traditional burst segmentation, so FSR is feasible both in the core and at the edge. In addition to the simulations presented here, we also ran simulations with other types of TCP, including high speed versions like HS-TCP and CUBIC. The results are similar in those cases. Lastly, we also compared FSR with the performance of burst retransmission we presented in [2]. We found that FSR provides better performance without the requirement of large edge buffers to store bursts for retransmission.

7 Conclusion

In this paper we have evaluated the performance of the proposed proactive loss recovery mechanism, forward segment redundancy. We proposed an analytical throughput model of TCP over OBS with FSR and validated it using simulations. We have compared the performance of FSR with traditional burst segmentation and burst drop. FSR improves file transfer completion time over burst drop by an order of magnitude at sustained high network loads and by over two times at lower loads. It also provides significant improvement over traditional segmentation.

FSR adds redundant data into the network, but this does not impact the network performance because the redundant data is given lower priority. There is no buffering or complicated FEC generation required at the ingress node, though it does increase the scheduling complexity of core nodes due to segmentation. The complexity is no greater than traditional burst segmentation. We have also found that 100% FSR appears to be an optimal value for both single-hop flows and multi-hop flows.

References

1. C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69–84, Jan. 1999.
2. Q. Zhang, V. M. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over optical burst-switched networks with burst retransmission," *Proceedings, IEEE Globecom 2005, Photonic Technologies for Communications Symposium*, Nov. 2005.
3. X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," *Proceedings, IEEE INFOCOM*, Mar. 2004.
4. V. M. Vokkarane and J. P. Jue, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," *SPIE Optical Networks Magazine*, vol. 4, no. 6, pp. 81–89, Nov.-Dec. 2003.
5. S. Arima, T. Tachibana, and S. Kasahara, "Fec-based burst loss recovery for multiple-bursts transmission in optical burst switching networks," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, Dec. 2005, vol. 4, pp. 5 pp.–2061.
6. B. Kantarci and S. Oktug, "Loss rate-based burst assembly to resolve contention in optical burst switching networks," *Communications, IET*, vol. 2, no. 1, pp. 137–143, January 2008.
7. V.M. Vokkarane and Qiong Zhang, "Forward redundancy: a loss recovery mechanism for optical burst-switched networks," in *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*, 0-0 2006, pp. 5 pp.–5.
8. B. Komatireddy, D. Chandran, and V.M. Vokkarane, "TCP-aware load-balanced routing in optical burst-switched (OBS) networks," *Proceedings, Optical Fiber Communication and the National Fiber Optic Engineers Conference*, pp. 1–3, March 2007.
9. X. Huang, V.M. Vokkarane, and J.P. Jue, "Burst cloning: a proactive scheme to reduce data loss in optical burst-switched networks," *Proceedings, IEEE International Conference on Communications*, vol. 3, pp. 1673–1677 Vol. 3, May 2005.
10. Y. Xiong, M. Vanderhoute, and H.C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838–1854, Oct. 2000.
11. X. Yu, C. Qiao, Y. Liu, and D. Towsley, "Performance evaluation of TCP implementations in OBS networks," in Technical Report 2003-13, *The State University of New York at Buffalo*, 2003.