Using Stop-and-Wait to Improve TCP Throughput in Fast Optical Switching (FOS) Networks over Short Physical Distances

Pablo Jesus Argibay-Losada*, Kseniia Nozhnina†, Gokhan Sahin‡§, Chunming Qiao*
*LANDER, Dept. of CSE, SUNY at Buffalo, NY (USA) email: pargibay@ieee.org, qiao@computer.org
† State University of Information and Communication Technologies, Kiev (Ukraine), email: knozhnina@ieee.org
‡ ECE Dept., Miami University OH (USA), email: sahing@miamioh.edu
§ Also with Instituto de Telecomunicacoes, University of Aveiro (Portugal), as Nokia-Siemens Networks (NSN) Chair

***Abstract*—Due to lack of optical RAM buffers in fast optical switches (FOS), statistical multiplexing technologies using FOS like Optical Burst Switching (OBS) or Optical Packet Switching (OPS) are expected to have higher data losses than conventional electronic networks. Consequently, applications transferring data by means of TCP can suffer from a lower throughput. In this paper, we show that this low-throughput problem is mainly an artifact caused by the conventional TCP congestion control algorithms, and can be remedied by using a simple yet effective stop-and-wait congestion control algorithm instead, as long as the propagation delay between TCP source and TCP destination is small compared to the transmission time of an optical packet. We show that such a condition holds for a wide range of scenarios, including fat-tree-based data center networks. We also show that the throughput achieved in a FOS network using stop-and-wait can be the same as, or higher than that in an equivalent, conventional electronic network.**

I. Introduction

Today's high speed electronic networks suffer from high energy consumption levels, large footprints, high complexity, and high capital and operational costs. Next generation optical technologies are expected to play a significant role in overcoming these problems, especially with the advances in fast optical switches (FOS). FOS allow us to create networks that avoid large quantities of optoelectronic conversions, reducing the need for power-hungry, costly transceivers and other electronic switching/routing devices. Compared to conventional optical circuit switching (OCS) using slower optical switching devices such as reconfigurable-optical-add-drop-multiplexers (ROADMs) [1], FOS can achieve a better utilization and efficiency. This is due to the fact that FOS can be reconfigured in very short times: recent prototypes have demonstrated capabilities for switching tens of optical channels in a few tens of microseconds [2], [3]. While lack of optical RAM buffers is often regarded as a serious problem in FOS networks, their bufferless nature also makes FOS networks simpler than their electronic counterparts. Data center networks, due to their high bandwidths and fast changing interconnection patterns [4], are one of the environments where FOS have a great potential. Research has been especially intense in combining electronic and optical switching [2], [5], opening the door to progressively growing volumes of traffic managed through FOS.

Optical Burst Switching (OBS) [6] is a statistical multiplexing network technology that takes advantage of the availability of FOS. In this paper, we will focus on OBS as a representative switching technology for FOS although the general discussions also apply to other technologies using FOS in a statistical multiplexing mode like Optical Packet Switching (OPS). In OBS, each fiber contains one or more wavelengths, serving as control channels, that are terminated electronically at each switch, and several others, serving as data channels, that are not. A packet on the control channel carries information about a data packet that is expected to arrive soon on one of the data channels; this data packet is commonly referred to as a burst. Since most traffic in high-speed networks is comprised by packets with large payloads and relatively small control headers, the idea in OBS is that only the information in the small control header is processed electronically by placing it on the control channel, while the payload cuts through the switches undisturbed in optical format. In this way, large amounts of power consumption are avoided, and savings in cooling equipment and footprint can be realized.

OBS can be used as either a MAN/WAN network technology, interconnecting several feeder networks (e.g., for Tier 3 to Tier 1 Internet Service Providers) or as a fast interconnection network for users directly connected to them (e.g., in a LAN to connect servers inside a data center). Architectures proposed for the former usually emphasize the processes for aggregating data from external users into the data bursts —in what is commonly called an assembling process— and their potential advantages and disadvantages in the network design. Architectures for the latter do not usually have that degree of freedom since it is much more likely that only a single flow will exist in a given route at a specific time, and therefore bursts will often contain packets from only a single flow. In this paper we focus on this latter type of networks.

Although OBS networks promise large improvements in terms of energy efficiency, speed and cost, several issues must be resolved before they can reliably substitute conventional electronic networks. The lack of optical RAM buffers in the former leads to higher losses than in the latter at the same traffic loads. This loss-prone environment usually induces

978-14799-3360-0/14/$31.00 ©2014 IEEE

large throughput reductions in applications if left unmanaged. Although the research community has studied these problems in depth, few practical solutions have been proposed to date and the research community is actively engaged in their study. From the point of view of network configuration, some solutions have involved the use of buffer-like mechanisms like fiber-delay-lines (FDLs) inside the FOS [7]–[9], or generic contention resolution techniques like deflection routing [10]–[12]. However, the inherent inflexibility of the involved optical components with respect to their electronic counterparts usually causes those solutions to be either bulky or impractical —e.g., large numbers of finely grained FDLs in the FOS to emulate electronic buffers— or not able to decrease losses to levels that are either negligible, or comparable to those found in electronic networks at similar loads.

Most data in current networks is transported over TCP, which is usually very sensitive to losses. Therefore, there has also been a strong interest in the study of the performance of TCP in OBS networks. Several works have identified the throughput reductions that can be expected there due to increased losses; e.g., see [13] for a study of the TCP Reno version, from which many common TCP stacks are derived, and [14] for an empirical study showing the relative sensitivity to losses of many common TCP implementations. Other works have tried to optimize TCP performance by means of predictive techniques inside the OBS network; this is the case of [15], reporting losses down to five times less than in normal conditions, which in turn lead to estimated average throughput gains up to $\simeq$2.2 times the original levels. Other proposals rely on the modification of the burst assembly period at the network edge in environments where this is a variable that can be set, like [16], which also reports throughput improvement factors in the range from 2 to 3. Another option is to modify the behavior of the usual additive-increase-multiplicative-decrease (AIMD) congestion control (CC) algorithms present inside most common TCP sources; this is the case of [17], which reports throughput gains up to 1.4 in unmodified OBS networks, or up to 1.9 if retransmission of lost bursts is allowed. These and other similar techniques can help to improve TCP performance in specific OBS scenarios. However, they are not usually enough to overcome the large sensitivity that TCP shows to the loss levels that can be expected in OBS networks, and which can decrease throughput by several orders of magnitude. Applications using TCP in OBS networks can therefore expect much lower throughputs than in equivalent electronic networks.

In this paper, we show that this high sensitivity to losses of applications using TCP in OBS is just an artifact of the algorithms used for CC in most TCP implementations —which are usually derivatives of AIMD or designed to be compatible with it, e.g., TCP-friendly. By changing these to stop-and-wait (SAW) algorithms, we show that this sensitivity can be eliminated, allowing applications to have throughputs like those found in conventional electronic networks, and therefore eliminating the influence that the lack of buffers inside the OBS network has in application performance. The only condition for this to happen is that the propagation delay between TCP source and TCP destination be small compared to the transmission time of the optical data packet. We show that this condition holds for a wide range of scenarios, especially when the involved geographical distances are short, like those expected to be found in data centers.

The main contribution of this paper is the proposal of SAW as an effective CC algorithm for TCP connections in OBS networks. We will show how it can increase throughput up to several orders of magnitude with respect to AIMD-based alternatives, thereby rivaling the performance of electronic networks in similar conditions. This, in turn, implies that in order to take maximum advantage of an OBS network and to maximize its throughput, it is not necessary to resort to complex or bulky hardware inside the network, or even to abandon the statistical multiplexing environment typical of OBS in favor of end-to-end resource reservation alternatives like those based on lightpaths. Instead, it suffices to change the CC algorithm inside the end points of the connections. The network core remains simple, allowing the network to take full advantage of the simplicity, energy-efficiency and cost benefits of all-optical networks. Our second contribution is a study of the conditions under which that behavior happens, showing that wide ranges of common scenarios can benefit from it, especially those involving short geographical distances like data centers. Our third contribution is an extensive simulation study of SAW in a useful and illustrative setting: a fat-tree data center network. We also note that the proposed CC algorithm, SAW, should work equally well in other FOS networks that switch packet by packet without the need for optical RAM buffers, like OPS.

After describing in Section II the OBS networks we consider here, Section III presents our SAW algorithm and the theoretical base that explains its good performance in bufferless FOS networks. Section IV contains an exhaustive simulation study of SAW in a fat-tree data center network. Finally, Section V summarizes our main findings and concludes the paper.

II. OBS NETWORK ARCHITECTURE

We consider OBS networks interconnecting groups of servers, which are the endpoints of all data transfers. Each connection performs a bulk data transfer of a given size from a sender application to a receiver application, hosted in different servers. Connections can appear at any time between any pair of servers, and their rates are controlled by TCP on behalf of the applications. There is a given maximum transfer unit size (MTU) for data bursts in the network. Each server has one bidirectional interface to a FOS, with one fiber for each direction. Inside each fiber, there is one wavelength acting as control channel and w additional wavelengths acting as data channels for bursts. There is no wavelength conversion available at any point in the network. In traditional OBS terminology, the servers are OBS edge nodes, and the FOS are OBS core nodes.

FOS offer the advantage of avoiding the need for data payloads to be processed and stored electronically inside the

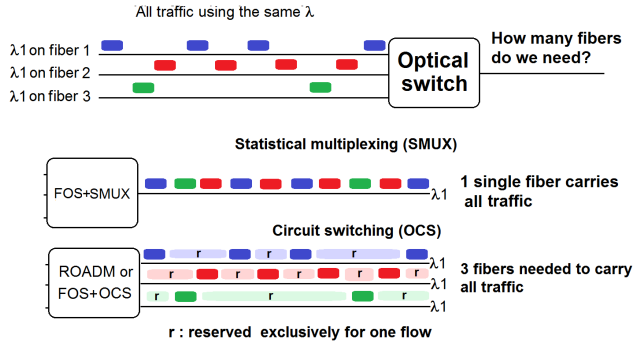


Fig. 1. Advantages of statistical multiplexing over OCS when FOS are available; example with the number of needed output fibers.

switches. Although these bypassing capabilities are already being used inside ROADMs in current MANs and WANs, ROADMs are usually built with micro-electro-mechanical-switches (MEMS), liquid crystal (LC) or liquid crystal on silicon (LCoS) [18], whose switching times are often in the range of milliseconds and are therefore not adequate to switch data payloads shorter than that range (e.g., a 9KB data packet at 1Gb/s only lasts 72μs). Accordingly, ROADMs are normally used to create OCS networks, reserving entire wavelengths for permanent circuits between end nodes (i.e., lightpaths), avoiding in this way inefficient reconfigurations. However, the drawback of OCS comes with the underutilization of lightpaths, given that they cannot be used by traffic in other routes even when they are idle. OCS needs thus extremely careful resource provisioning, and when the network finally reaches the point of not being able to carry traffic despite possibly having a large amount of available-but-unusable capacity, some other solutions must be implemented; e.g., adding one or more routers to instead switch traffic at the IP level (i.e., electronically) in as many hops as necessary in order to reach the destination. In contrast to this, packet-by-packet switching with FOS avoids OCS's wastage of bandwidth using datagram-based transmission. This is made possible by breakthroughs in several technologies, e.g., LiNbO3 electro-optical [19], [20], PLZT [21] or SOA [22]-based switches, with switching times in the order of nanoseconds. The development of FOS thus addresses the fundamental inefficiencies of OCS, allowing statistical multiplexing of datagrams in the optical domain. We depict in Fig. 1 an example of these advantages in an scenario with 3 flows trying to traverse the same output link of an optical switch. Using the FOS and statistical multiplexing techniques, all three input traffic flows can share the same output wavelength in the same output fiber. On the other hand, with OCS, three wavelengths in three output fibers would be needed.

The FOS are interconnected in an arbitrary topology, and shortest path routing based on hop count is used. If there are several shortest paths between two nodes, random load balancing will be performed among all of them, with each path assigned the same probability of getting a data burst. Load balancing algorithms are usually designed to achieve a trade-off between implementation complexity and probability of packet reordering inside a given flow. In our case, we are not concerned with sophisticated load balancing algorithms to avoid reordering because, by their own design, SAW algorithms only have one unacknowledged packet in flight at any given time, and therefore they cannot suffer reordering in the same way as conventional TCP [1].

When a sending application sends data to the TCP layer inside a server, whichever CC algorithm is in use decides whether to send the data to the OBS layer for transmission. The OBS layer stores all packets coming from the TCP layer in a sending buffer, creating one or more data packets as a function of whether the data size is larger than the MTU. For each data packet, a control packet is created and sent immediately through the control channel. The control packet tries to reserve resources as soon as it arrives at each FOS, i.e., the control plane works in a just-in-time (JIT) mode [23]. The data burst only contains a single data packet from its originating TCP flow, and it is delayed for some time —the offset time— in order to account for the transmission times, potential queueing times and processing times of its control packet and for the FOS switching times in each switch in the path. Then, the data burst is sent, cutting through the FOS on its path until reaching the destination server, or until blocked due to contention for resources in some intermediate FOS, or until overtaking a slower control packet. The value of the offset time is dependent on the specific scenario; given that the size of the control packet has a direct influence on both the transmission times and processing times inside the switches, it is convenient that it be as small as possible. The minimum amount of information that the control packet must contain also depends on the specific OBS implementation. For example, one option is to store the destination server address, the wavelength that the burst is traveling on, and the amount of time that the control packet requests from the switch for the data burst. This last quantity can be computed by the origin server as a function of the data packet size according to a common unit of time. For example, since with 16 bits we can address up to $\simeq$65000 servers and store packet lengths up to 64KB, we could have a control packet with 40 bits size —i.e., only 5 bytes— that is able to indicate those characteristics plus any channel in a group of 256. Assuming that the control channel works in an Ethernet fashion with 8 bytes for preamble and start-of-frame delimiters, that leads to a total transmission size of 13 bytes, which, at 1Gb/s, only need $\simeq 0.1\mu$s to be transmitted in each hop of the path. In a 6-hop path that would lead to a contribution of $\simeq 0.6\mu$s for the offset time. We should also add to that value the switching time of the switches (or the maximum of the path if they are different), as well as a budget to take into account potential queueing and processing delays. After the reservation time expires, each switch will consider the requested channel free again. Another possibility here

[1]The sensitivity of conventional TCP to reordering is due to the fact that it can confuse it with losses; this, in turn, stems from the fact that conventional TCP flows usually have more than one unacknowledged packet in flight at any given time.

would be to only have the destination address and wavelength in the control packet, and to create a second control packet at the sending server, to be sent just immediately after the burst has departed, and whose only task is informing the intermediate FOS that they can free the resources that they have assigned to the data burst. Although all these details are important from an implementation point of view, and many variants have been proposed in the literature to deal with them, their impact on performance will decrease as technology improves and is able to deal with them in efficient ways. From now on, and unless stated otherwise, we will assume that the impact of the additional offset time is negligible in terms of network or application performance, effectively treating it as zero.

III. SAW FOR CONGESTION CONTROL

In this section we describe the theoretical reasons that explain the performance improvements that can be expected of SAW algorithms when used in a CC role in OBS networks like the ones we described above. We start by focusing on the relevant characteristics of conventional TCP implementations, and then we describe the SAW algorithms that we propose.

A. TCP congestion control

The throughput of a TCP flow is strongly determined by the CC algorithm it uses. The research community has developed several CC algorithms; additive-increase-multiplicative-decrease (AIMD) ones are common (e.g., Tahoe, Reno, NewReno, SACK). Others are based on AIMD or are compatible with it in some operating range (e.g., CUBIC, Compound TCP, TCP-friendly). Several ways of detecting congestion can also exist: e.g., TCP Vegas using delay, Reno using losses, or Compound TCP using both. An interesting observation for our work is that some TCP variants have been optimized for the high speed, high latency scenarios usually called long-fat networks (LFN), where many packets from the same flow can be in flight at any given time (like FAST, CUBIC or Compound TCP). The OBS networks we described above, although their links are extremely fast, are not LFN since they do not have buffers and therefore very few packets, or even not a complete one, can fit in the path between the sender and receiver servers. All these versions, which we will call *conventional TCP* from now on, usually achieve similar performance when used in OBS networks in terms of their throughputs [14], or, if some of them outperform the others in some specific scenario, they are not usually able to overcome the penalties that the loss-prone behavior of OBS exerts on applications in a consistent fashion in multiple network scenarios. These penalties are usually the result of two main problems that we describe now.

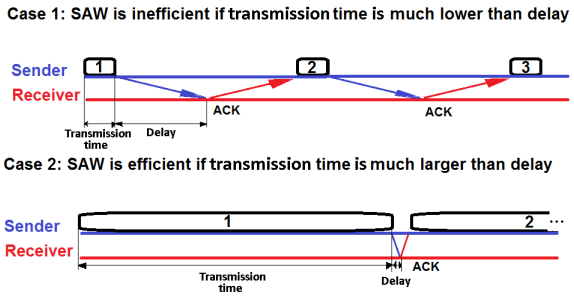
First, conventional TCP stacks allow in general to send back-to-back packet trains inside the network whenever the CC algorithms at their cores tell them to do so. This is not especially problematic in conventional networks, where switches have buffers, since the short-term congestion caused by those packet trains can be managed with the buffers in order to avoid or mitigate losses. It is only if those trains remain large for long periods of time that losses can appear because of buffer overflows or the action of random early dropping algorithms. In OBS networks, however, as there is no possibility of buffering, those trains of back-to-back packets imply a much bigger risk of losses than in conventional networks. In this sense, OBS networks could potentially benefit from smoother traffic patterns than the ones generated by conventional TCP stacks (e.g., like the ideas that led to the proposal of techniques like TCP pacing [24]).

A second additional problem of conventional TCP stacks, more pressing than the previous one —and the main basis for the frequent observations in the literature about OBS sensitivity to losses—, is the huge effect that the loss levels likely to be found in OBS networks have in the throughput of applications. TCP evolved in environments where losses were much lower than in OBS, so it does not perform well in the presence of losses that are considerably higher. In fact, this observation applies to any environment where TCP is used and losses are high, and has also led to large research efforts in several areas unrelated to OBS —e.g., loss-prone wireless networks, with solutions advocating link-layer retransmissions to deal with large losses in order to help TCP.

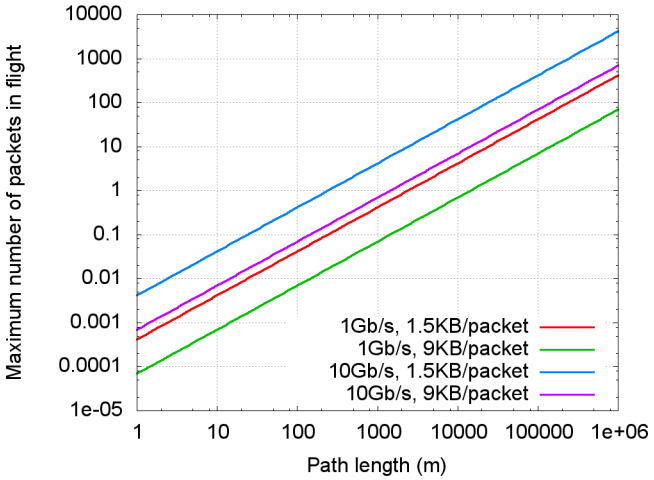
In summary, the algorithms used for CC in conventional TCP stacks have shown to be effective in many types of buffered networks, where losses are relatively low. However, their throughput is extremely sensitive to higher loss levels, like those found in bufferless environments like OBS. In our experience, from the two problems we described above, the second one is by far the dominant reason behind the noticeable throughput decrease that conventional TCP flows suffer in OBS networks. Both of them are direct consequences of the CC algorithms used in the core of TCP. Therefore, the resulting loss-related throughput sensitivity is an artifact that can be eliminated by changing those CC algorithms for another, more suitable, CC algorithm adapted to the bufferless OBS environment. As we will see next, SAW algorithms are especially good candidates for this task.

B. Stop-And-Wait (SAW)

SAW is the generic name of a family of well-known algorithms that have been proposed in the literature for use in flow-control roles in computer networks [25]. Their usage is remarkably simple: the sender sends a data packet and waits for an acknowledgment packet (ACK) to arrive from the receiver. If no ACK has been received after a given time T, the packet is retransmitted. The receiver sends an ACK back to the sender as soon as it receives a data packet. The value of T can be made dependent on the situation; e.g., it can be progressively increased if several timeouts have occurred without receiving an ACK according to a finite state machine, etc. Different policies for computing T lead to different SAW versions. SAW algorithms have simplicity as a distinctive advantage. However, they are not widely used in the Internet in general because they can have poor performance in many common scenarios, with much lower throughputs than those achieved by means of other algorithms —e.g., AIMD. More



(a) Two diagrams to assess whether SAW algorithms present good (case 2) or poor (case 1) performance in terms of efficient usage of bandwidth.



(b) Maximum number of packets in flight from sender to receiver in an OBS path.

Fig. 2. Performance of stop-and-wait (SAW) algorithms.

specifically, whenever the delay between sender and receiver is much larger than the transmission time of a packet, the throughput of SAW will be low [25]; Fig. 2(a) shows an example of this. Common scenarios in current buffered data networks are not usually suitable for SAW, especially in the common case with more than one hop between sender and receiver, due to the store-and-forward operation of conventional networks —e.g, in a path with n hops, even if the propagation time is negligible, the minimum one-way delay will be $n-1$ times the transmission time of the packet, and therefore SAW will not be particularly efficient there. However, if the previous condition does not hold, i.e., the delay between server and receiver is not much larger than the packet transmission time, SAW usually becomes an extremely efficient way of transmitting data. For example, the situations where the head of a data packet has reached the receiver before the tail of the packet has departed the sender are good candidates for using SAW. This was precisely the case in the low-speed LAN network environments where SAW was proposed, but it is also paradoxically the case of high-speed OBS networks such as those we described in Section 2 due to their lack of buffers from sender to receiver. Therefore, SAW is a simple and efficient choice to be used for CC purposes there. In other words, the main factor that makes OBS lossy, i.e, its bufferless nature, is also the factor that allows an extremely low one-way delay between senders and receivers and that therefore makes SAW an extremely effective replacement for the CC algorithms usually found in TCP.

Let us illustrate numerically the previous point. For the sake of the argument, we consider two networks with the same topology and the same high-bandwidth links, where one of them is a conventional, buffered, electronic network, and the other one is an OBS network. The first will probably constitute a LFN, where many packets can be in flight at the same time from sender to receiver, mainly stored in intermediate buffers —much of the past and ongoing research in CC in LFNs is related precisely to the fact that, with so many potential packets in flight, there are many options to optimize application performance. However, as we noted above, the OBS network is quite the opposite of a LFN, with much fewer packets in flight at any given time, and likely not even a single one. To depict this, we show in Fig. 2(b) the number α of packets that fit in the path from sender to receiver in an OBS network as a function of the geographic length of the path d (covering distances in LAN, MAN and WAN scenarios), the bandwidth of the data wavelength b (cases of 1Gb/s and 10Gb/s) and the packet size k (1.5KB and 9KB, roughly the MTU of normal and jumbo Ethernet frames). Taking the speed of light in the optical fiber as $v = 2 \cdot 10^8$m/s, α equals

$$\alpha = \frac{d/v}{8 \cdot k/b} = \frac{d \cdot b}{8 \cdot k \cdot v},$$

We see in Fig. 2(b) that α is usually much lower than 1, especially for the smaller, LAN-like distances. SAW will thus be in general a good choice for CC purposes in OBS networks of limited geographical size like data centers. As the path length increases, so does the number of packets that can fit inside the path, and SAW starts experiencing efficiency problems due to the increased delay between the sender and the receiver. However, as we will see in the next section, this does not imply that SAW will be a bad choice for even WAN-like distances compared to conventional TCP, because the latter also experiences many problems related to the bufferless environment. Actually, as we will show, SAW is so efficient compared with conventional TCP that the path length should grow to unrealistically high values in the order of tens of thousands of kilometers before its performance decreases to the levels that conventional TCP achieves. We also note that the values of α in conventional electronic networks would typically be anywhere in the range from tens to hundreds or even thousands of packets as a result of the availability of large buffers.

In summary, the bufferless nature of OBS implies that the transmission time of a packet will be comparable to, or higher than, the delay between the sender and the receiver. This, in turn, implies that there will be very few packets belonging to a given route inside the OBS network, and very likely not even a single one. As a consequence, and in contrast to what

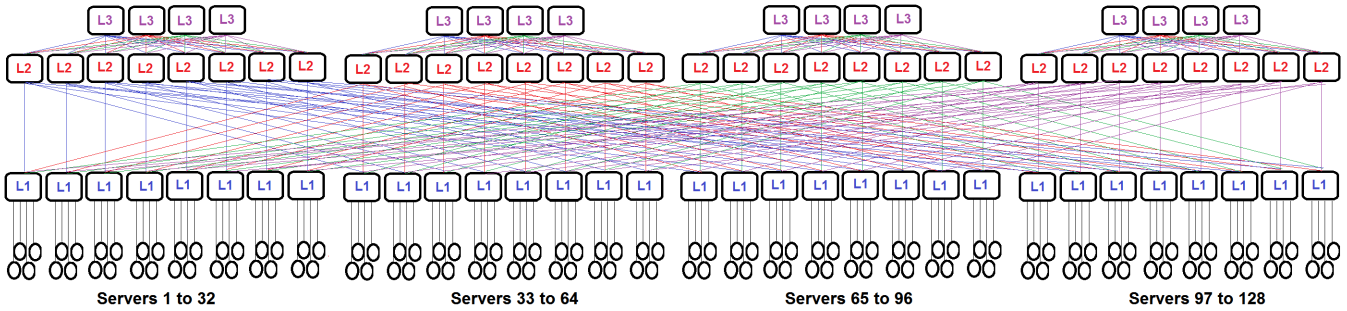


Fig. 3. Fat tree network topology with 3 switch levels (labeled L1, L2 and L3), interconnecting 128 servers.

happens in conventional electronic networks, SAW algorithms are a convenient, efficient choice to use in a CC role with TCP in OBS networks. To showcase this, in the next section we will perform a comprehensive study of SAW in a useful scenario: a fat-tree data center network.

IV. Throughput of SAW in a fat-tree

In this section we study the throughput performance of SAW in OBS networks like the ones we described in Section 2. We consider the fat-tree (FT) network topology depicted in Fig. 3. FT topologies are useful in data center environments due to their large bisection bandwidths, to the multiple paths available between pairs of servers, and to the possibility of setting up the network using identical, low-port-count, low-cost switches. In general, a three-level FT like the one we consider here, where each switch has p bidirectional ports, interconnects a maximum of $p^3/4$ servers, and contains a total of $5p^2/4$ switches. In our example, switches have 8 ports each, leading to a topology composed of 80 switches and interconnecting 128 servers. All links are bidirectional, with 1 fiber in each direction and 1 control channel wavelength and 1 data wavelength in each fiber, both operating at 1Gb/s. Each link has a length of 100m.

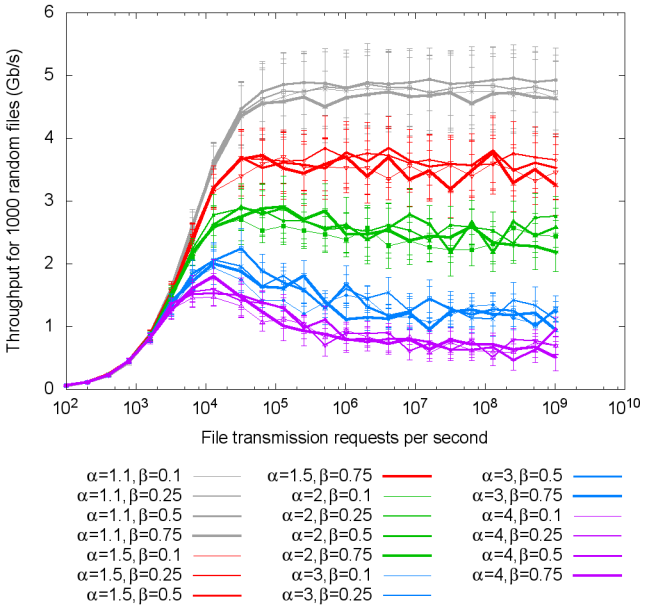
Bulk data transfers happen between random pairs of servers, and their size in bytes is randomly chosen according to a Lognormal($\mu = 8.46$, $\sigma = 2.38$) distribution [26]–[28], whose mean is $\simeq$ 100kB, its median is close to 5kB and most of its bytes belong to flows larger than 100MB. This distribution reflects the empirical observations in common networks that most bytes tend to belong to a few large flows, while most flows are relatively small in size. The MTU is 9kB, the maximum conventional size of Jumbo Ethernet frames; packet size can be lower than that if the application does not have enough data to send at the end of its transmission.

The initial value for the retransmission timeout (RTO) T in SAW is 1ms, and its maximum value is 60s. Each time that a timeout occurs without having received an ACK for the packet in flight, T is increased by a factor α, i.e., $T \leftarrow \alpha \cdot T$; when an ACK is received, T is set to an intermediate value between the current estimation of the round-trip-time (RTT) and the current T, weighed by a factor $\beta \in (0,1)$: $T \leftarrow \beta \cdot \text{RTT} + (1-\beta) \cdot T$. This algorithm adapts to congestion periods by making SAW less aggressive in a multiplicative way, and reacting to less congested periods by making the algorithm more aggresive by means of setting T close to the RTT but always a little higher. The throughput of SAW will depend on the values of α, β and the loss probability inside the network. The loss probability in any path, in turn, depends on α, β, having in this way a coupled system analogous to those with the TCP CC protocols commonly found in conventional electronic networks.

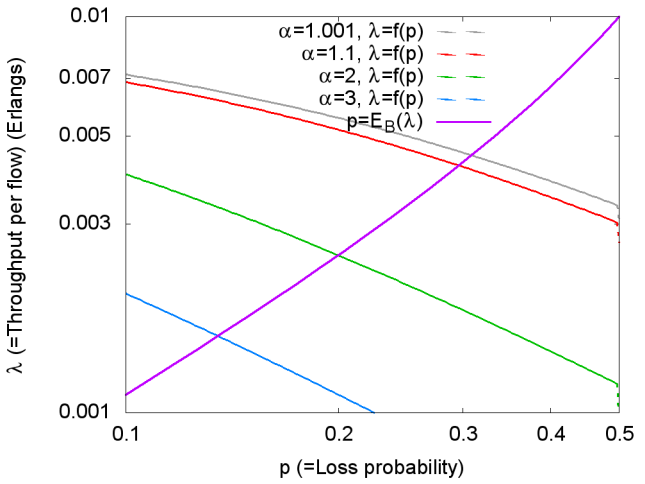
The bulk-data transfer patterns are determined by a process that sets them up at given points in time. For example, in a MapReduce [29] environment, after the mappers have processed the information in their nodes and start shuffling information, multitude of transfers start to take place among the servers, and the reducers will start after those transfers have finished. We define the network throughput as the sum of the sizes of all the files exchanged by all the flows, divided by the time elapsed between the arrival of the first connection request until the end of the last connection. Fig. 4(a) shows the network throughput as a function of the connection arrival rate to the network and of the parameters α, β. We show the 95% confidence intervals for the throughput, computed over random sets of 1000 files.

The rightmost part of the figure corresponds to arrival rates high enough to have them in practice arrive at the same time at the network —e.g., like what we would have in a MapReduce environment where mappers start and finish at roughly the same time, leading to the transmissions across the network during the shuffling phase to also start at roughly the same time. In this case, the throughput depends heavily on how efficient the CC algorithm is when sharing resources at the beginning of the transmissions —when all flows coexist— and on how efficient it is at the end, when there only remain a few large flows because the rest have already finished. The leftmost part of the figure corresponds to more spread-out data transfers that, in extreme cases, can lead to idle periods between them; this case does not load the network so much as the previous one, leading to lower throughputs, as we can observe.

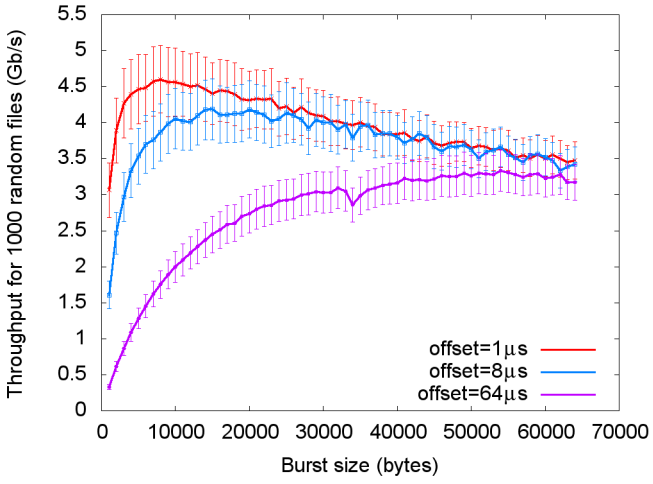
We can clearly see the big impact that α, β have on the network throughput. For the depicted α, β pairs, we can see that SAW performs best for the most aggressive value of α (i.e., the lowest, 1.1), and β does not appear to have a large influence. Therefore, SAW only needs to be slightly sensitive



(a) Throughput as a function of the α, β determining the RTO.



(b) Analytical model of a bottleneck with SAW flows.



(c) Throughput as a function of the offset time and burst size.

Fig. 4. Throughput of SAW in the fat-tree topology.

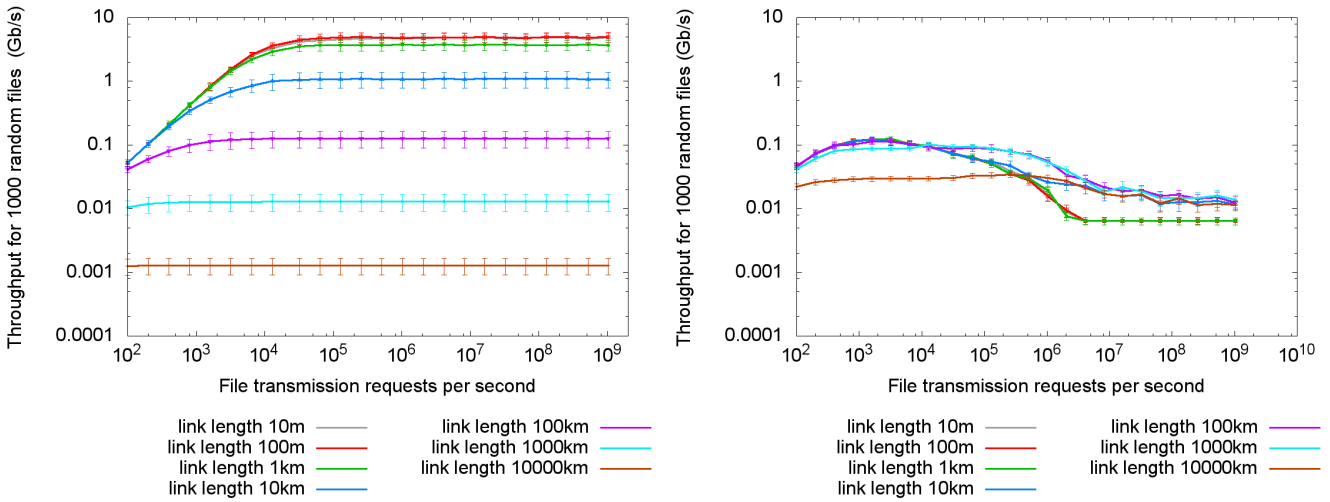
to congestion (i.e., decreasing sending rate) to perform well, allowing it to be simple and robust.

One interesting question here is whether this throughput improvement with decreasing α is a behavior that is maintained in scenarios other than the fat tree. Although the performance modeling and study of SAW in generic topologies under different traffic matrices is a complex task, we can gain insight into this question by means of simplified mathematical analysis in a bottleneck shared by multiple flows. Any flow, if it is large enough to experience losses, will consist in the repetition of two types of packet trains: one with successfully sent packets followed by one with lost packets. The lengths of these trains are random: if p is the packet loss probability, the average lengths of the successful and lost trains will be $1/p$ and $1/(1-p)$ packets, respectively. If the transmission time of the burst is much larger than the propagation delay, it is straightforward to estimate the throughput of a given flow as

$$\lambda = \gamma \frac{1/p + 1/(1-p)}{1/p + \sum_{i=0}^{x} \alpha^i} \qquad (1)$$

where $x \triangleq \lceil 1/(1-p) \rceil$. $\gamma \in (0,1)$ is the average number of active flows at a given time, and it empirically takes into account the combined effect of the processes creating the flows, their durations and their routes. If there are n flows crossing the bottleneck at a given time, we can estimate p with the Erlang-B function for one channel, under the assumption of many input links to the bottleneck: $p = E_B(1, n \cdot \lambda/(1-p))$. We plot Eq. (1) in Fig. 4(b); we show the four curves corresponding to the cases $\alpha \in \{1.001, 1.1, 2, 3\}$ when $\gamma = 0.01$. We also plot the Erlang-B function for $n = 100$ flows; for each α, the intersection of its curve with the Erlang-B one gives an approximation of the throughput of the flow. We can see that, the lower the α, the higher the throughput, as we saw in the fat tree simulations. We can also see that, when α is close to 1 (e.g., the 1.1 value we used in the simulations), decreasing it more (to 1.001) does not result in a significant predicted throughput gain. Therefore, α should be set at some value not much larger than 1, but not too close in order to avoid false timeouts. Interestingly, the ratio between the throughputs for the cases of $\alpha = 1.1$ and $\alpha = 2$ is similar to the one observed in Fig. 4(a) for the fat tree; i.e., around 2. This suggests that $\alpha = 2$ can be a practical limit for the upper bound of α when configuring SAW, in order not to loose too much throughput.

SAW also depends on the MTU and on factors closely related to the specific hardware in use, like the offset time. Fig. 4(c) depicts that dependence in the fat tree for $\alpha = 1.1, \beta = 0.5$ and an arrival rate of $10^6 s^{-1}$. We can see how throughput increases when offset times decrease, since there is less idle time ahead of each burst. And, given an offset time value, if the burst size increases, the throughput will increase as well because the amount of time spent switching bursts grows with respect to the offset time. In other words, the burst size can be used as a degree of freedom to compensate for technological factors like slow optical switches in order to increase throughput. In any case, the influence of these



(a) Throughput of SAW.

(b) Throughput of AIMD.

Fig. 5. Performance of SAW and AIMD in the fat-tree topology (128 servers, 1Gb/s per data wavelength per link).

factors is expected to progressively decrease due to advances in optical switching technology.

Fig. 5 compares the throughput of SAW with the throughput of AIMD —i.e., conventional TCP— as a function of the link length (in the case of conventional TCP we have used TCP SACK as an example; results with other common TCP flavors are similar to the ones shown here). Fig. 5(a) depicts the case of SAW, and Fig. 5(b) the case of AIMD. Considered link lengths range from a low value of 10m per link (i.e., a small data center network), to an unrealistically large value of 10000km —included here just to showcase the effect of geographical length in the performance of SAW, as we observed in the previous sections.

We can see how the throughput of SAW is largely unaffected for link lengths of up to 1km (in our fat tree that means a maximum path length of 6km), therefore easily covering most data centers. As observed before, SAW throughput decreases from that point due to the increased delays between servers and receivers. However, even for a link length of 1km (i.e., maximum length of 6 km), the network throughput would only decrease to $\simeq$1.5 times less than the maximum, and that geographical range is already large for a data center network.

Fig. 5(b) depicts the case of AIMD. The most important observation from it, in comparison to Fig.5(a), is that its throughput is much lower than the throughput of SAW. We can see that, in contrast to SAW, the link length does not have much influence on the performance of AIMD —all lines are essentially in the same region of the figure—, but this insensitivity does not help at all with achieving high throughputs. For example, for the case of a 100m link length and large interarrival rates, SAW achieves a throughput of 5Gb/s, while AIMD only achieves $\simeq$10Mb/s —i.e., more than 2 orders of magnitude, or 500 times, less throughput than SAW. The rest of the cases are similar. Even as link lengths increase from a few meters to one thousand kilometers, SAW easily outperforms AIMD. Again, it is important to note that the longest depicted lengths are unrealistic for fat-trees or other data center networks; we have included them just to depict how link lengths have to go up to an unrealistic value of 10000km before SAW performs clearly worse than AIMD.

Given the observations in the literature about the lossy nature of OBS and its impact on applications, it is also helpful to compare SAW in the OBS fat-tree network with AIMD in an equivalent, conventional electronic network. We depict that comparison in Fig. 6. In order to show comparable cases, topology and data link capacities are the same in both networks; buffer sizes in the electronic case are set to 50 ms of traffic at line speed (we simulated different values and that was the one maximizing TCP performance in the electronic network). We show the cases of SAW in OBS, AIMD in OBS, and AIMD in the electronic network[2]. We can clearly see again the huge improvements that SAW achieves over AIMD in OBS, and also how OBS can easily obtain similar or better performance than an equivalent electronic network with buffers.

In summary, if TCP flows are going to be used in an OBS network like the ones we described in Section 2, we can increase network throughput dramatically in most situations by changing their CC algorithms to SAW. We can even achieve better throughput performance than in equivalent conventional electronic networks that use common TCP stacks, while enjoying the energy efficiency, simplicity and performance advantages of next-generation optical networks.

V. Conclusion

We have analyzed the throughput of a family of stop-and-wait (SAW) algorithms for congestion control of TCP flows in statistical multiplexing networks with fast optical switches

[2]In the electronic case, we have tested the SACK, NewReno and Linux versions available in the ns-2 network simulator [30].

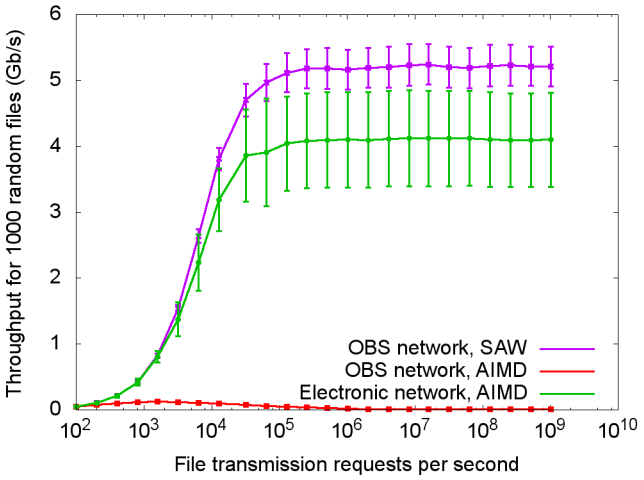


Fig. 6. TCP throughput in OBS (SAW and AIMD cases) and in an equivalent electronic network (AIMD case) in the fat-tree topology.

(FOS) like OBS networks. Due to their simplicity and suitability for bufferless environments, SAW algorithms constitute an effective solution to the low-throughput performance problems that can occur when conventional TCP flows are used in OBS. The only required condition is that the transmission time of a packet is similar to, or larger than, the delay between TCP sender and TCP receiver. We showed that this condition holds in many practical cases. We have studied in detail the advantages of SAW in a fat-tree network, by showing how SAW can easily outperform conventional TCP by several orders of magnitude for common data patterns when distances are short, as in data centers. In the fat-tree studied in this paper, SAW obtains nearly 500 times more throughput than conventional AIMD, also rivaling or exceeding the performance obtained by the latter in an equivalent conventional electronic network. As a result, SAW allows us to address the burst loss concerns that have been reported in the literature regarding the bufferless OBS networks. The proposed SAW and the general results obtained also apply to similar network technologies using FOS without the need for optical RAM buffers and having a short end-to-end propagation delay, like Optical Packet Switching (OPS).

REFERENCES

[1] T. A. Strasser, J.L.Wagener, "Wavelength-Selective Switches for ROADM Applications," IEEE Journal of Selected Topics in Quantum Electronics, vol. 16, no. 5, pp. 1150–1157, sep./oct. 2010.

[2] N. Farrington, A. Forencich, P.C. Sun, S. Fainman, J. Ford, A. Vahdat, G. Porter, G. Papen, "A 10μs Hybrid Optical-Circuit/Electrical-Packet Network for Datacenters," OFC/NFOEC 2013, Anaheim, CA (USA).

[3] N. Farrington, G. Porter, P.C. Sun, A. Forencich, J. Ford, Y. Fainman, G. Papen, A. Vahdat, "A Demonstration of Ultra-Low-Latency Data Center Optical Circuit Switching," ACM SIGCOMM 2012, Helsinki (Finland).

[4] N. Farrington, A. Andreyev, "Facebook's Data Center Network Architecture," IEEE Optical Interconnects Conf. 2013, Santa Fe, NM (USA).

[5] N. Farrington, Y. Fainman, H. Liu, G. Papen, A. Vahdat, "Hardware Requirements for Optical Circuit Switched Data Center Networks," OFC/NFOEC 2011, Los Angeles, CA (USA).

[6] Y. Chen, C. Qiao, X. Yu, "Optical burst switching: a new area in optical networking research," IEEE Communications Magazine, vol. 18, no. 3, pp. 16–23, may 2004.

[7] K. Merchant, J. McGeehan, A. Willner, S. Ovadia, P. Kamath, J.Touch, J. Bannister, "Analysis of an optical burst switching router with tunable multiwavelength recirculating buffers," IEEE Journal of Lightwave Technology, vol. 23, no. 10, pp.3302–3312, oct. 2005.

[8] J. Xu, C. Qiao, J. Li, G. Xu, "Efficient burst scheduling algorithms in optical burst-switched networks using geometric techniques," IEEE Journal on Selected Areas in Communications, vol. 22, no. 9, pp. 1796–1811, nov. 2004.

[9] X. Lu, B.L. Mark, "Performance modeling of optical-burst switching with fiber delay lines," IEEE Trans. on Communications, vol. 52, no. 12, pp.2175–2183, dec. 2004.

[10] E. Wong, J. Baliga, M. Zukerman, A. Zalesky, G. Raskutti, "A New Method for Blocking Probability Evaluation in OBS/OPS Networks With Deflection Routing," IEEE Journal of Lightwave Technology, vol. 27, no. 23, pp. 5335–5347, dec. 2009.

[11] S. Lee, K. Sriram, H. Kim, J. Song, "Contention-Based Limited Deection Routing Protocol in Optical Burst-Switched Networks," IEEE Journal on Selected Areas in Communications, vol. 25, no. 8, pp. 1596–1611, 2005.

[12] J. Baliga, E.W.M. Wong, M. Zukerman, "Analysis of Bufferless OBS/OPS Networks with Multiple Deflections," IEEE Communications Letters, vol. 13, no. 12, pp. 974–976, dec. 2009.

[13] A. Detti, M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks", IEEE INFOCOM 2002, New York, NY (USA).

[14] L. Liu, H. Guo, T. Tsuritani, Y. Yin, J. Wu, X. Hong, J. Lin, M. Suzuki, "Dynamic Provisioning of Self-Organized Consumer Grid Services Over Integrated OBS/WSON Networks," IEEE Journal of Lightwave Technology, vol. 30, no. 5, pp. 734–753, mar. 2012.

[15] K. Ramantas, K. Vlachos, "A TCP-Specific Traffic Proling and Prediction Scheme for Performance Optimization in OBS Networks," IEEE Journal of Optical Communications and Networking, vol. 3, no. 12, pp. 924–936, dec. 2011.

[16] S. Peng, Z. Li, Y. He, A. Xu, "TCP Window-Based Flow Oriented Dynamic Assembly Algorithm for OBS Networks," IEEE Journal of Lightwave Technology, vol. 27, no. 8, pp. 670–678, 2009.

[17] B. Shihada, P.H. Ho, Q. Zhang, "A Novel Congestion Detection Scheme in TCP Over OBS Networks," IEEE Journal of Lightwave Technology, vol. 27, no. 4, pp. 386–395, feb. 2009.

[18] E.B. Basch, R. Egorov, S. Gringeri, S. Elby, "Architectural tradeoffs for reconfigurable dense wavelength-division multiplexing systems," IEEE Journal on Selected Topics in Quantum Electronics, vol.12, no.4, pp.615–626, jul.-aug. 2006.

[19] G. Papadimitriou, C. Papazoglou, A. Pomportsis, "Optical Switching: Switch Fabrics, Techniques, and Architectures," IEEE Journal of Lightwave Technology, vol. 21, no. 2. pp. 384–405, feb. 2003.

[20] J. Gripp, J.E. Simsarian, J.D. LeGrange, P.G. Bernasconi, D.T.Neilson, "Architectures, Components, and Subsystems for Future Optical Packet Switches," IEEE Journal of Selected Topics in Quantum Electronics, vol. 16, no. 5, pp.1394–1404, sep./oct. 2010.

[21] K. Nashimoto, "Nano-Second Speed PLZT Waveguide Switches and Filters," EpiPhotonics White paper, dec. 2011.

[22] H. Brahmi, G. Giannoulis, M. Menif, V. Katopodis, D. Kalavrouziotis, C. Kouloumentas, P. Groumas, G. Kanakis, C. Stamatiadis, H. Avramopoulos, D. Erasme, "On the fly all-optical packet switching based on hybrid WDM/OCDMA labeling scheme," Optics Communications, vol. 312, pp. 175–184, feb. 2014.

[23] J. Wei, R. McFarland, "Just-In-Time Signaling for WDM Optical Burst Switching Networks," IEEE Journal of Lightwave Technology, vol. 18, no. 12, pp. 2019–2037, dec. 2000.

[24] A. Aggarwal, S. Savage, T. Anderson, "Understanding the performance of TCP pacing," IEEE INFOCOM 2000, Tel Aviv (Israel).

[25] D. Bertsekas, R. Gallagher, "Data networks," 2nd edition, Prentice Hall, ISBN: 0132009161, 1992.

[26] N. Agrawal, W. Bolosky, J. Douceur, J. Lorch, "A five-year study of file-system metadata," ACM Trans. on Storage, vol. 3, no. 3, a. 9, oct. 2007.

[27] K.M. Evans, G.H. Kuenning, "A Study of Irregularities in File size Distributions," SPECTS 2002, San Diego, CA (USA).

[28] J. Douceur, W. Bolosky, "A large-scale study of file-system contents," ACM SIGMETRICS 1999, Atlanta, GA (USA).

[29] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters," Comm. of the ACM, vol. 51, no. 1, pp. 107–113, jan. 2008.

[30] USC ISI, "The network simulator – ns-2"