Dual-Layer Congestion Control for Transmission Control Protocol Carried by Optical Packet Switching With User Data Protocol Background Traffic

Zheng Lu and David K. Hunter

***Abstract*—A congestion control scheme called dual-layer congestion control (DLCC) is proposed for use when transporting Internet traffic over optical-packet-switched networks. It further reduces the core optical buffering requirement over existing proposals; indeed each optical core switch is assumed in the modeling work to have a shared optical buffering capacity of only 20 optical packets for all ports. Furthermore, it does not depend for its operation on having a certain number of Transmission Control Protocol (TCP) flows carried over each link. The scheme is designed to operate in conjunction with an edge-smoothing algorithm that segments IP datagrams into fixed-length optical slots to be carried by the core. It expedites the response of TCP to congestion in the optical core network, both by reducing the rate of packet transmission over the optical packet core and by throttling TCP sources via the transmission of additional triple duplicate ACK segments. Packet loss performance and edge-buffering capacity requirements are evaluated through mathematical analysis, showing that the packet loss rate can be decreased through the use of DLCC by a factor of up to six times and also showing that electronic edge-buffering requirements are reduced through the use of DLCC. Furthermore, simulation modeling shows that DLCC yields a TCP goodput improvement of between 2 and 10 times, depending on the volume of background User Data Protocol (UDP) traffic and the round-trip time. This demonstrates that DLCC is viable and enhances network performance.**

***Index Terms*—Congestion control; Optical buffering; Optical packet switching (OPS); Transmission Control Protocol (TCP).**

Manuscript received October 27, 2008; revised January 6, 2009; accepted January 9, 2009; published June 12, 2009 (Doc. ID 103337).

Z. Lu is with Acorah Software Products Ltd., Wokingham, Berkshire RG40 1XS, UK.

D. K. Hunter is with the School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK (e-mail: dkhunter@essex.ac.uk).

Digital Object Identifier 10.1364/JOCN.1.0000A1

I. Introduction

It has often been predicted that optical packet switching (OPS) will provide flexibility, efficient resource utilization, and high capacity in future communications networks [1]. These performance characteristics of OPS will be of crucial importance to the future Internet for a number of important reasons. First, the number of users will continue to increase as the digital divide narrows in many countries, and as the number of connections per user increases, optical networking will be a natural solution to accommodate these requirements. Furthermore, the envisaged penetration of fiber to the home (FTTH) and the increased use and storage of digital media will drive the requirement for a high-capacity, flexible network core based on high-capacity OPS networking. Finally, these trends will also be driven by future transmission capacity requirements implied by the ever-increasing processing power and memory capacity available to users.

However, there would be problems in a realistic optical-packet-switched network carrying traffic generated by Transmission Control Protocol/Internet Protocol (TCP/IP) because of the limited buffer size attainable in practice with optical-delay-line technologies. Indeed, optical delay lines, which are commonly proposed as buffers for contention resolution, are bulky and require temperature stabilization [2], whereas solutions based on slow light will be unsuitable for implementation of large buffer capacities [3]. This paper proposes a congestion control mechanism that does not require modifications to TCP and is suitable for optical-packet-switched networks with small buffers for contention resolution. All implementations of TCP regard loss of a segment (i.e., a TCP packet) as an indication of network congestion; hence because the very small buffers feasible with optical packet switching are liable to overflow, resulting in lost segments, this normally presents an impediment to TCP's correct operation. Unlike existing work [4], the

1943-0620/09/0200A1-16/$15.00

mechanism proposed in this paper to overcome this problem does not depend for efficient operation on having a specific number of TCP flows present on each link. Furthermore, the algorithm is designed to yield favorable interaction between elastic TCP flows and nonreactive User Data Protocol (UDP) flows, which would both be present in a realistic mixture of Internet traffic types.

It is shown in this paper via mathematical analysis and computer simulation that efficient operation of an optical-packet-switching network carrying both TCP and UDP traffic is possible with core optical packet switches each having an optical buffering capacity of only 20 fixed-length slots shared between all ports. This is achieved in conjunction with an existing edge-smoothing scheme [5] that encapsulates IP datagrams into fixed-length slots that are transported over the core optical-packet-switched network. Analytical results are provided for packet loss, and simulation is used to evaluate goodput, a crucial performance parameter when evaluating TCP.

TCP traffic is considered with UDP background traffic in this study, representing the major traffic types in the current Internet. The existing TCP end-to-end congestion control mechanism was designed for networks with comparatively low transmission capacity; however, it is not appropriate in its original form for use with the very high data transmission rates and the very restricted optical buffer capacities that are likely to arise with future implementations of optical packet switching. While existing work has considered the problem of transporting TCP over networks with small router buffers [4], the proposals in this paper have the two advantages of first permitting further reduction in the permitted optical buffer capacity while second their size is not dependent on the number of TCP flows present.

In this paper, a dual-layer congestion control (DLCC) mechanism is proposed, employing electronic buffers in the routers at the network edge, but very small optical buffers in the core optical packet switches. If appropriate, it reacts to congestion in the core by quickly reducing the transmission rate from the network edge into the optical packet core while, over a longer time scale, quenching electronic traffic at the source before it is fed into the edge routers. Without DLCC, the high transmission speed of OPS and the infeasibility of even modestly sized optical buffers would result in many optical packets being lost in the event of congestion due to the relatively slow reaction of TCP's congestion control mechanism, which takes of the order of a round-trip time (RTT) to react. Indeed, in slotted OPS, the slot time is only a few microseconds, whereas the TCP RTT for a wide area network (WAN) is orders of magnitude greater, ranging from several milliseconds to several hundreds of milliseconds. When operating a network at high loads, congestion can become a serious problem, and DLCC overcomes this by expediting the network's response.

It has already been shown that smoothing of traffic at the edge of the core can reduce the size of optical contention resolution buffers that are required in core switches [5,6]. Although DLCC can interoperate with any shaping or smoothing implementation at the edge routers, in this paper a simple existing traffic-smoothing scheme is assumed [5], which is summarized for completeness in Section II. In the remainder of this paper, Section II introduces a simple traffic-smoothing scheme to smooth traffic at each edge router, while Section III describes the DLCC scheme in detail, and Section IV provides a discussion of its scalability and implementation aspects. To demonstrate the performance improvements possible with DLCC, an analytical model is derived in Section V, with the results being discussed in Section VI, while Section VII illustrates a simulation scenario and discusses the results. Section VIII concludes the paper.

II. Traffic Smoothing

For completeness, this section summarizes the traffic-smoothing scheme that was used in conjunction with DLCC to obtain the performance evaluation results reported in this paper. A full description can be found in a previous publication [5]. Time is divided into intervals, called negotiation intervals. At the network edge, each incoming edge-to-edge traffic flow is smoothed by a buffer into a stream of packets that may change rate at the beginning of each negotiation interval based on stepwise rate estimation. A flow is defined as the data stream from one ingress edge router to another egress edge router; each edge router has a buffer to carry the flow destined for each other edge router. Throughout each negotiation interval, the packets in each flow are transmitted by the ingress edge router at a constant rate, much as constant bit rate (CBR) traffic in asynchronous transfer mode (ATM). At the beginning of each negotiation interval, the packet transmission rate over the core may be changed through renegotiation. During renegotiation, an edge router checks that sufficient capacity is available in the core network to accommodate the new required transmission capacity; if not, the request is rejected. The data rate of an edge-to-edge flow during a negotiation interval is fixed.

To facilitate estimation of the service rate to be requested for the next negotiation interval, time is split up into small units. After each such time unit, the moving average arrival rate into each buffer is recalculated using the moving average calculation of Eq. (1). In the simulations reported later, the size of a unit

lay between 5 and 100 ms, depending on traffic type. U_i is the measured arrival rate for traffic entering the buffer during unit i, and R_i is the smoothed arrival rate, defined via Eq. (1) below. The service rate to be requested at the next renegotiation is obtained by taking the mean of R_i over all units in the last negotiation interval. The parameter ν decides to what extent previous experience and to what extent the current measured rate should influence the new service rate and is determined by considering the arrival rate, the current service rate, and the available network capacity [5]. This heuristic algorithm is based on a moving average calculation of the arrival rate into the buffer and may be described as follows:

$$R_{i+1} = \nu R_i + (1 - \nu)(U_i + B/\tau) . \quad (1)$$

A detailed description of the above equation and how it describes the operation of the edge-smoothing algorithm can be found in an earlier paper [5].

III. Dual-Layer Congestion Control

This section introduces the DLCC concept in detail, with a discussion first of its context in relation to existing TCP congestion control algorithms, followed by a description of how congestion is detected in the core and a discussion of the way that DLCC responds to this at the network edge.

A. Motivation for DLCC

There is considerable existing literature on TCP congestion control. It has been pointed out that when congestion occurs with drop-tail routers, global synchronization may be introduced into the network [7]—in other words, when a buffer in a router overflows, packets that belong to several TCP connections are often dropped simultaneously, and these decrease their congestion windows at the same time because of TCP's fast recovery feature. (Below, discussion of synchronized TCP flows indicates that this type of global synchronization is assumed to take place.) Active queue management (AQM) schemes such as random early detection (RED) [8] and weighted RED (WRED) implement congestion avoidance at routers through random dropping of incoming packets when the measured moving average mean buffer occupancy reaches a certain level and thus avoid global synchronization. Rather than using dropped packets to indicate congestion, an explicit congestion notification bit can notify possible congestion to a TCP source [9].

These existing AQM schemes allow TCP to react to congestion earlier than otherwise, before the congestion becomes serious and buffer overflow occurs. However, they were intended for a conventional TCP/IP environment and cannot be applied directly to optical packet networks because their implementation requires measurement of the mean router buffer size to determine the extent of congestion, which is not feasible or appropriate with the very small packet buffers in the optical core switches that are considered in this paper. TCP's slow reaction, of the order of a round-trip time, when coupled with the high transmission rates inherent in optical packet switching would seriously degrade network performance. Furthermore, these techniques require modifications to TCP. To address these issues in OPS networks, link-utilization-based detection and transmission rate adjustment for TCP ACK segments has been proposed [10]; however, the proposal in this paper provides a more aggressive response to core congestion, thus exhibiting superior performance improvements, as demonstrated later.

Like the existing AQM schemes discussed above, DLCC, which is introduced in this paper, provides a more timely response to congestion. However, unlike RED, it is compatible with the very small optical packet buffers that are feasible with OPS. DLCC works in both the electronic and optical domains to react quickly to serious core network congestion, and it quenches TCP sources when necessary. Furthermore, it does not require specific changes to TCP implementations at traffic sources—widely used implementations with fast retransmit and fast recovery are sufficient.

When considering the optical buffer capacity required when all end-to-end traffic sources employ TCP, it has been argued [4] that the widely used TCP rule of thumb is not suitable for backbone buffer dimensioning and that much TCP multiplexed backbone traffic tends to be Poisson-like rather than self-similar or bursty because synchronization of TCP congestion windows, as described above, is rare in backbone networks. Hence it is argued [4] that there is no need to dimension the buffers of backbone core routers by using this rule of thumb. In the proposal by Appenzeller and Keslassy [4], the size of the buffer required on each router port depends on the number of TCP connections present; this limitation is overcome in this paper while permitting efficient operation with further reduction in optical buffer capacity. Indeed, the proposals in this paper resolve congestion in optical core switches, when mixing elastic TCP traffic with nonreactive UDP traffic, in order to reduce the mismatch between the relatively slow reaction of TCP and the fast optical packet loss that would otherwise occur because there are many optical packets in transit.

B. Congestion Detection

In DLCC, each core switch constantly monitors the optical packet arrival rate and records it in memory. If packet loss occurs, the switch checks the current rate in order to guess whether this loss is caused by con-

gestion or by routine contention between slots; the latter could occur under any conditions. Therefore, packet loss caused by such contention under normal load will not initiate DLCC's congestion control mechanism. The optical packet arrival rate at an output port is measured as the sum of all traffic from input ports that is directed to that output port. If this rate is greater than the switch's port transmission rate, or this rate reaches a specified threshold rate that is less than this, then any loss incurred is identified as being due to congestion (referred to as heavy congestion later in this paper).

To differentiate between such contention and congestion, a simple algorithm is employed in core optical packet switches, which maintains a congestion descriptor; this is the average arrival rate on a switch input in packets per second for the previous $2T$ s. T is the estimated longest RTT in the Internet, typically between 0.4 and 1.0 s, and it is chosen to facilitate differentiation between contention and congestion via the following mechanism. First, if packet loss has taken place, and the congestion descriptor indicates that the packet loss is due to a short burst of traffic, no interaction with TCP is necessary because TCP can respond appropriately to the situation within the next RTT; such a situation is regarded as contention. However, if packet loss has taken place, but the congestion descriptor exceeds the threshold for more than two RTTs, loss can be regarded as being due to serious congestion, and DLCC must be invoked through notification of congestion.

Thus packet loss from core OPS buffers, in conjunction with the status of the congestion descriptor, determine the state of congestion, implementing a form of explicit congestion notification [9,11] by sending notifications of congestion back to the edge routers. A decision about congestion is made based not only on buffer overflow, but also on the congestion descriptor, because the optical buffers are so small, and this mechanism is necessary to prevent confusion with contention. Other forms of AQM, such as RED have also been proposed [8]; however, as discussed above, optical packet switching lacks the relatively large buffers that are required to implement these successfully.

If the OPS packets are from different ingress-to-egress flows, then the congestion loss notification is sent to their corresponding ingress edge routers. These notifications can be sent either by dedicated packets or by piggybacking the notifications onto reverse path packets to save transmission capacity. Piggybacking can be on a hop-by-hop basis, where the notification is destroyed and regenerated at each switch. This is useful if there are no packets traversing the entire reverse path but there are packets on each reverse link. If there are no suitable packets at all on such a reverse link, then packets are generated specially to carry congestion loss notifications—this will not compete for transmission capacity with any data packets because none exist. However, it is not necessary to analyze and simulate the effect of this here.

DLCC informs the sending TCP process quickly of congestion in the optical core so that it can react more quickly than unaided TCP. However, contention (as defined above) generally does not trigger the DLCC control mechanism because it is not relevant and would constitute an additional and unnecessary processing burden. Identifying losses due to contention and congestion, as differentiated above, and treating them differently, permits each to be handled appropriately.

C. Dual-Layer Reaction at the Edge to Congestion

Congestion loss notifications are sent from congested core switches to the corresponding edge routers that initiate the congesting traffic flows. This subsection describes how edge routers react to such notifications in dual-layer congestion control.

Ingress edge routers reduce their sending rate upon the arrival of congestion loss notification messages according to the procedure described below (Fig. 1). The traffic from TCP/IP sources arrives at an ingress edge router and proceeds in the form of fixed-length slotted optical packets through the core network. The diagram shows heavy congestion losses at a certain core switch, and with DLCC, this switch sends back notifi-

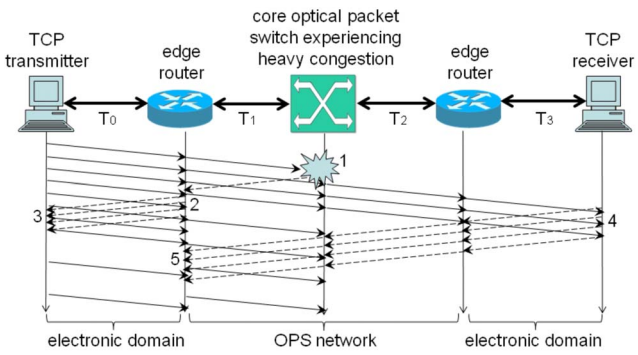


Fig. 1. (Color online) Timeline illustration of DLCC, showing at the top the scenario used for modeling, which consists of a TCP transmitter, edge routers, a core switch experiencing heavy congestion and exhibiting packet loss, and the TCP receiver. T_0, T_1, T_2, and T_3 are the propagation delays of each network segment. The following events take place with DLCC in order to expedite TCP's response to congestion: 1. A core optical packet switch experiences packet loss and determines that this is due to heavy congestion. 2. The edge router receives a congestion notification from the core switch and reduces its transmission rate of smoothed optical packets into the core. 3. Four artificial ACK packets arrive at the TCP transmitter from the edge router, and TCP then halves its congestion window through the fast recovery feature. 4. The TCP receiver sends duplicate ACKs because a segment has been lost. 5. These duplicate ACKs are discarded by the edge router because it sent the duplicate ACKs to the TCP transmitter earlier.

cation to the appropriate edge router, based on its knowledge of the routing of flows between edge routers. After the ingress edge router receives the notification, it operates in two ways simultaneously through the following two operations:

1. It immediately sends four artificial ACKs for the same sequence number (representing the start of the lost segment) to the corresponding TCP source, which TCP will interpret as requesting retransmission of the lost segment (three duplicate ACKs). This is referred as the OPS ACKs scheme below. The information is available to do this, without undue overhead, because the scheme retains for at least one round-trip time in each edge router the socket information from each TCP data segment. The transmission of the four artificial ACKs reduces congestion by halving the congestion window of TCP traffic sources through TCP's fast recovery feature, hence having the advantage over proposals involving ACK pacing [10] of reducing the transmission rate of the sources dramatically. Although it is true that sending artificial ACK segments in this way violates the layered protocol structure of TCP/IP, firewalls and network address translation (NAT) are widely deployed in practice and do not respect the layered protocol structure either.
2. Through the procedure described in detail below, the ingress edge router also reduces the sending rate of slotted optical packets into the core by an amount dependent on the number of received congestion loss notifications. It freezes rate negotiation until the ingress buffer occupancy starts to decrease. This reduces congestion by buffering at the network edge any extra traffic that arises before the sending TCP process has time to respond to the four artificial ACK segments sent in operation 1 above.

Edge buffer occupancy is increased by operation 2 above, but it can reduce congestion faster than operation 1; however, to optimize performance, both are used in combination. Although operation 1 is slower, it alleviates congestion at the source. Even without this OPS ACKs mechanism, the TCP receiver will ultimately send the duplicate ACK packets by itself, but the response to congestion would then occur more slowly.

Slot-by-slot measurement at the network edge of the rate of congestion notification from the network core and rapid reaction to this information are necessary when implementing operation 2 above. This is necessary to reduce the optical packet sending rate at the network edge immediately and by an appropriate amount, so that congestion in the core can be resolved as soon as possible without unnecessarily sacrificing transmission capacity utilization. Each notification message carries information about the optical packet loss rate R^{loss} in a congested core switch; however, the calculation of this only takes into account loss of optical packets during congestion periods; packet losses that occur during contention periods are ignored for the purpose of calculating R^{loss}.

R_{schedule} is defined as the original scheduled sending rate between a given pair of edge routers, calculated immediately after the previous rate renegotiation by the algorithm outlined in Section II. After an edge router receives the first notification with loss rate R_o^{loss}, all subsequent scheduled optical packets in each flow sent over the corresponding link by that edge router are separated from the next packet by a number of slots equal to the reciprocal of its present sending rate minus the overflow rate; i.e., the new rate is $R_{\text{schedule}}-R_o^{\text{loss}}$. (The sending rate is the reciprocal of the time, measured in slots, between adjacent packets in the same flow between two given edge routers.) The second notification carries loss rate information equal to R_1^{loss}, and traffic slots after this notification are sent at the rate of $R_1^{\text{send}}=R_{\text{schedule}}-R_1^{\text{loss}}$. When the ith notification is received, where $i>1$, then $R_i^{\text{loss}}=(R_1^{\text{loss}}+R_2^{\text{loss}}+\cdots+R_{i-1}^{\text{loss}})/(i-1)$ and $R_i^{\text{send}}=R_{\text{schedule}}-R_i^{\text{loss}}$.

Then the sending rate is changed to R_i^{send}, and this sending rate is retained, even if no further notification is received. If there are no further loss notifications, transmission at this rate continues until, when a decision is made about whether to renegotiate the transmission rate, the ingress buffer occupancy measured by the edge-smoothing algorithm of Section II has decreased. When this happens, a new value of R_{schedule} is computed, and this is adopted as the transmission rate. Therefore, the overall effect is to decrease the rate of optical packet transmission, depending upon the received notifications.

During this process, nothing has been done thus far to reduce the external traffic arrival rate entering the ingress edge router, but the transmission rate from the edge buffer into the network core has been reduced. This results in the edge buffer occupancy increasing over at least one TCP end-to-end round-trip time. To overcome this buffer buildup, the ingress edge router immediately sends three duplicate ACKs to each corresponding TCP source, using the OPS ACKs scheme described above.

IV. Scalability and Implementation Issues

The discussion in this section explains why dual-layer congestion control is a scalable solution that is relatively easy to implement with no disruption to user installations. Indeed, DLCC does not require any modifications to the TCP implementation at each

source, but rather any widely used TCP implementation with fast retransmit and fast recovery is appropriate. Signaling of congestion notifications only requires extra packets if no other packets are already being sent along the entire reverse path.

The receiving TCP process also generates triple duplicate ACKs in response to loss; however, these are discarded by the edge router that earlier sent the four artificial ACK segments (Fig. 1). Once the ingress router receives notification from the core of packet loss due to congestion, it sends back three duplicate ACKs to the corresponding source, as discussed above. After approximately one RTT, it receives identical duplicate ACKs from the receiver. It checks to see whether these have already been generated, and if so, it does not forward them to the source. It is possible that ACKs from the TCP receiver might be missed by the edge router due to asymmetric routing or merely packet loss due to buffer overflow. If this happens, the ACKs will be routed through another edge router to the TCP sender. This affects neither the effectiveness of DLCC nor the execution of TCP congestion control. The TCP sender will ignore these ACKs because it previously received three duplicate ACKs from the edge ingress router. If these ACKs are merely dropped in the reverse path, DLCC and TCP will not be affected either, because three ACKs have already been sent to the TCP transmitter by the edge router. Thus Fig. 1 illustrates why the amount of signaling traffic with and without the OPS ACKs scheme is the same.

Each electronic interface in every ingress edge router records the socket information from data segments in TCP flows it handles, so that ACKs sent from the TCP process in each receiver can be identified and discarded if appropriate. However, this information about each TCP segment need only be retained for approximately one RTT; hence the storage requirements are well within the memory capabilities of modern router hardware.

Datagram headers arriving from users are examined at the edge, and based upon this, optical packet headers have a 1 bit field that indicates whether the data was generated by TCP or some other layer 4 protocol. For UDP traffic, the congestion control scheme could be combined with resource-aware routing algorithms to resolve congestion. In this case, when congestion occurs, an edge router would reduce loss by buffering the extra traffic. This would provide time for the edge router to compute a new noncongested route, then start new negotiation in order to transmit a flow over it. However, this is not considered in the modeling work below. Essentially, DLCC is a type of TCP-friendly rate control protocol that mainly interacts with TCP's fast retransmit and fast recovery mechanisms.

V. Mathematical Performance Analysis

In this section, mathematical models are proposed to analyze performance aspects of DLCC and hence justify its validity. These models consider traffic from both UDP traffic sources with abrupt fluctuations in capacity demand and TCP connections having congestion windows that can be either synchronized or nonsynchronized. Before introducing the model itself, there is discussion of Internet traffic, the behavior of TCP, and the traffic configurations used for modeling.

A. Internet Traffic

The DLCC algorithm is designed to yield favorable interaction between elastic TCP flows and nonreactive UDP flows. This subsection is a brief discussion of the mixture of traffic types that are found on the Internet, justifying the need to consider traffic generated by both TCP and UDP.

More than 10 years ago, in 1998, approximately 95% of the bytes, 90% of the packets, and 80% of the flows on the Internet were generated by TCP [12]. However, with ever-increasing network capacity, real-time applications are now generating a larger fraction of Internet traffic. Real-time applications, including voice over IP (VoIP), video, videoconferences, peer-to-peer applications, and games are being used more frequently, and many of them rely heavily on UDP or run exclusively over it. Nevertheless, TCP traffic is still an important constituent of Internet traffic.

Because it offers guaranteed delivery of data through retransmission and because the transmission rate can change (for example, it can reduce suddenly due to the fast recovery mechanism initiated by segment loss), TCP is not suitable for streaming continuous audio and video data. Audio and video streams carried over TCP experience unacceptable delays and interruptions, leading to very low quality of service (QoS).

In a particular IP backbone, it was reported in 2003 that on some links over 60% of the traffic was generated by new applications such as distributed file sharing and streaming media, whereas only 30% was web traffic [13]. It was found, also in 2003, that for typical IP backbones, TCP traffic constituted between 60% and 90% of the total load, with UDP traffic constituting between 10% and 40%, and all other protocols combined producing less than 5% [14]. However, because TCP always seeks out and uses the available transmission capacity, it competes for transmission capacity with other traffic and is influenced by both other TCP traffic and by UDP traffic. Transmission capacity overprovisioning meant that in one set of measurements, again from 2003, most of the links were utilized by under 50%, and less than 10% of the links

in the backbone experienced utilization higher than 50% in any given 5 min interval [15].

Typical non-real-time applications, such as file transfer, web access, remote procedure calls, and distributed file services neither depend on nor generate well-defined traffic characteristics, and TCP is appropriate for them. However, with the use of more and more assorted video and audio online services, real-time applications will become more popular. These applications cannot be carried over TCP.

The performance of optical-packet-switched networks under TCP and UDP traffic has been analyzed by comparing various packet aggregation schemes at the edge of the core network [16]. The performance analysis in this paper still conforms to the composition of Internet traffic discussed above [12,17], with UDP being assumed to carry voice and video traffic. When congestion occurs, UDP sources are unaware of this and continue to generate traffic as before, because UDP does not have congestion control, resulting in poorer control over congestion than with TCP traffic alone.

B. Characteristics of Elastic TCP

When a network link is congested, UDP streams do not interact with TCP senders and hence still maintain their sending rate. Although fair sharing takes place between two TCP flows, it does not take place between a TCP flow and a UDP stream.

In the models in this paper, the same packet size is assumed throughout—for both external traffic and traffic in the optical packet core—in order to eliminate the influence of different packet sizes on traffic shaping. Hence if an optical packet is dropped in the network core, only one TCP segment is lost. However, the effect on network performance of dropping a long optical burst that carries many TCP segments has been studied already [18,19].

TCP connections can be long lived, with fixed rwnd (the window advertised by the receiver—default in many systems), or short lived (with a small amount of data for a particular connection). The sending rate is swnd=min(cwnd,rwnd), where cwnd is the window size determined by the sender's congestion control algorithm. The default value of rwnd is normally not infinite, so the sending rate will often be fixed at rwnd, with the Internet traffic being very smooth. The TCP fill-up effect, whereby TCP seeks out and uses spare capacity in the network, is not evident once cwnd increases beyond the value of rwnd, because then swnd=rwnd, which does not change dynamically as cwnd does. In this case, TCP will generally not consume all the available transmission capacity, being fixed at a certain sending rate that is restrained by rwnd. There will be little segment loss but the utilization could be very low, because TCP's ability to seek out and use available transmission capacity has not been fully utilized. It only probes in this way to determine what network capacity is available when cwnd $<$ rwnd and stops doing this when the congestion window (cwnd) increases beyond the value of rwnd [20,21]. Hence, this discussion shows that in reality, there are several factors influencing TCP's ability to exploit fully the available transmission capacity.

1. Only for a long-lived connection, where sufficient data has to be transported from the transmitter to the receiver, can TCP's ability to seek out and use network transmission capacity be utilized fully.
2. Only if the receiver's advertised window rwnd is always greater than the sender's cwnd, can the sender keep increasing its sending window size. A normal PC's default rwnd is fixed at a value between 16 and 64 Kbytes, depending on implementation. This default value must be configured manually if high utilization is to be realized in high-speed networks.

However, the worst case must be considered, where rwnd is infinite or at any rate always larger than cwnd, and TCP flows are long lived. Hence, in the analysis and simulation below, swnd is always assumed to be equal to cwnd, and hence available network transmission capacity is always assumed to be consumed by the sending TCP processes if the application is generating data sufficiently quickly. Although this ability of TCP to respond to and use available capacity could increase overall transmission capacity utilization in the network, it may also cause congestion and unfair transmission capacity sharing. To see this, consider that although TCP connections themselves will ultimately share available transmission capacity fairly, if some user applications have more TCP connections than others, then they will obtain the use of more capacity. Hence there is always fair sharing between individual TCP connections, but there may be unfair sharing between applications. This should be avoided by the access network operators. The access router should either employ some type of QoS control implementation or adjust rwnd appropriately to promote fairness and hence also achieve high transmission capacity utilization.

C. Traffic Configurations Used for Modeling

UDP traffic sources do not directly notify their change of rate to TCP sources, which, as described above, keep absorbing the remaining transmission capacity in the network. When UDP's transmission capacity usage decreases, TCP will absorb the new free capacity, but when UDP's capacity usage increases, TCP will compete with UDP and segment loss will occur over at least one TCP RTT. UDP's transmission

rate is dictated by the application, and unlike TCP, it does not react to the presence of other traffic in the network.

TCP Reno is assumed in the models in this paper, with fast retransmission and fast recovery implemented. It remains in the congestion avoidance phase after cwnd has been halved due to fast recovery after segment loss. It is assumed in the modeling work below that the receiver's advertised window is always larger than the sender's congestion window. (When rwnd < cwnd always, the sending window is always fixed at rwnd.)

The flight size is the amount of data that has been sent by TCP but not yet acknowledged [20]. For a given value of the TCP congestion window (cwnd), the mean transmission rate during any RTT is swnd/RTT. If rwnd > cwnd always, this is equal to cwnd/RTT. Although, during a particular RTT, TCP flight data entering an edge router and smoothed data within the core can have the same mean rate, the flight data might not be evenly distributed throughout the RTT. Smoothing that takes place at each edge router makes the bursty flight of data regularly distributed throughout the RTT.

D. Single TCP Flow With Changing UDP

The single TCP flow modeled here is long lived, with rwnd > cwnd always. Therefore, the sending window swnd = min(cwnd, rwnd) is always equal to cwnd. The normal combined TCP and IP header size with no options in either case is 40 bytes and is denoted by H. Below is the notation used in the mathematical analysis:

- a_i is the UDP data rate during a given period, measured as a fraction of overall transmission capacity. Such a period begins every time the rate of this background UDP traffic changes.
- a_{i+1} is the UDP rate during the next period, measured as a fraction of overall transmission capacity.
- T_i^{UDP} is the length in seconds of a period during which UDP traffic has a constant rate—it may change rate upon the start of the next period. It is also assumed that the UDP traffic rates during different T_i^{UDP} are statistically independent. Such a period can range in duration from less than the time taken for one TCP sawtooth to the time taken for several.
- $P(x=a_i)$ is the probability of UDP traffic having rate a_i during period T_i^{UDP}.
- L_i is the number of optical packets lost, given heavy congestion, during period i with a UDP data rate of a_i.
- $\bar{L}$ is the mean number of optical packets lost during heavy congestion over the entire duration being analyzed.
- P_L is the congestion loss rate, namely, the lost traffic during a congestion period divided by the full bottleneck link capacity, where both must be measured using the same units (bits, bytes, or packets).
- B_i is an event, defining heavy congestion caused by UDP during period i.
- $2T$ is the TCP end-to-end RTT.
- W_k is the initial value in bytes of cwnd at the start of the current interval, where the index k is used to identify different possible initial values.
- $W_{a_i}^r$ is the value of cwnd in bytes at the beginning of period i, with a UDP data rate of a_i, where the index r is used to identify different possible initial values.
- C is the bottleneck link capacity.
- R is the maximum proportion of the full link transmission capacity that UDP can use; hence the volume of UDP traffic is bounded by this.
- S is the TCP segment size in bytes—it is assumed that these are all the same size in the model.
- $W_{a_i}^p$ is the peak cwnd size for a given a_i in bytes, where the index p is used to identify different possible peak values. This peak size of the congestion window (cwnd) occurs immediately prior to TCP fast recovery.
- M is the peak cwnd size for a given a_i, expressed as a number of TCP segments.
- $\overline{B_1^c}$ is the extra edge router buffer capacity required to accommodate the extra traffic that arises without rapid notification from the edge router to the TCP transmitter if the OPS ACKs scheme is not implemented (see also the definition in Section VII). Recall that this extra traffic builds up in the buffer when the edge router reduces its transmission rate over the core, but before the TCP transmitter reduces its rate because of the duplicate ACKs sent by the TCP receiver itself.
- $\overline{B_2^c}$ is the extra edge router buffer capacity required with the OPS ACKs scheme to accommodate extra traffic described above. Here, rapid notification from the edge router to the TCP transmitter takes place through transmission of artificial duplicate ACKs.

High-capacity long-lived TCP flows with infinite rwnd are assumed, so that the TCP transmission rate halves during congestion avoidance rather than TCP going into slow start. Now two cases are considered: without DLCC and with DLCC.

1) Without DLCC: If packet loss occurs, it takes place within one RTT of the start of one of the periods referred to above, because by this time, TCP will have adjusted its transmission rate to accommodate the new rate of UDP transmission in the new period. Hence to obtain the loss rate, it is necessary to divide

by the round-trip time of $2T$. Also, division by C takes place so that P_L is expressed as a ratio:

$$P_L = \frac{\Sigma_{a_i} P(x = a_i \cap B_i) L_i / 2T}{C} = \frac{\Sigma_{a_i} P(B_i | x = a_i) P(x = a_i) L_i}{2TC}. \tag{2}$$

P_L is the loss rate due to heavy congestion and can now be used to calculate how much traffic is lost; the probability of heavy congestion in period i is the product of the probability that the UDP transmission rate has increased and the probability that the new UDP transmission rate is greater than X, the amount of transmission capacity left available by TCP were no packet loss to occur:

$$\begin{aligned} P(B_i | x = a_i) &= P(a_{i+1} > a_i) \\ &\quad \times P\left(a_{i+1} > C - \frac{W^r_{a_i}}{2T} \times \frac{S+H}{S} \right) \\ &= P(a_{i+1} > a_i) \times P(a_{i+1} > X). \end{aligned} \tag{3}$$

Here, $C - W^r_{a_i}(S+H)/(2TS) = X$. From $0 < a_{i+1} < CR$ and $0 < a_i < CR$, it follows that $-a_i < a_{i+1} - a_i < CR - a_i$ and $a_{i+1} - a_i > 0$ when $a_i < a_{i+1} < CR$, so assuming a uniform statistical distribution for both a_i and a_{i+1}, it follows that, for some specific value of a_i, the probability that the UDP traffic level has increased is given by

$$\begin{aligned} P(a_{i+1} - a_i > 0) &= P(a_{i+1} > a_i) \\ &= \frac{CR - a_i}{CR - a_i - (-a_i)} = \frac{CR - a_i}{CR}. \end{aligned} \tag{4}$$

For congestion to be possible, it must be true that the amount of capacity left free by TCP (assuming no packet loss were to occur) must be less than the maximum capacity that UDP can use, i.e., $0 < X < CR$, which, when combined with $a_i < a_{i+1} < CR$ from above, yields

$$\begin{aligned} &P(a_{i+1} > a_i) P(a_{i+1} > X) \\ &\quad = P(a_{i+1} > \max(a_i, X)) \\ &\quad = \frac{\int_{X=0}^{a_i} \int_{a_{i+1}=a_i}^{CR} da_{i+1} dX + \int_{X=a_i}^{CR} \int_{a_{i+1}=X}^{CR} da_{i+1} dX}{\int_{X=0}^{CR} \int_{a_{i+1}=a_i}^{CR} da_{i+1} dX}. \end{aligned} \tag{5}$$

The peak congestion window size for period i is $W^p_{a_i} = [2T(C - a_i)S]/(S+H)$ and, expressed as a number of TCP segments, the peak value of cwnd is $M = [2T(C - a_i)]/(S+H)$. Thus the value of the congestion window at the beginning of period i can be expressed as

$$\begin{aligned} W^r_{a_i} &= \frac{(T_i^{\mathrm{UDP}} - W_k 2T/S) \bmod (2TM)/2}{(2TM)/2} W^p_{a_i} \\ &= \left\{ (T_i^{\mathrm{UDP}} - W_k 2T/S) \bmod \left[\frac{2T^2 (C - a_i)}{S+H} \right] \right\} \frac{S}{T}. \end{aligned} \tag{6}$$

The use of the mod operator in Eq. (6) arises because of the characteristic sawtooth shape of a graph of the TCP congestion window against time when TCP is operating in equilibrium. W_k, the value of cwnd at the beginning of the overall interval being considered (which may consist of many periods), is assumed to be evenly distributed within the interval $[0, W^p_{a_i}]$ and $k = W_k/S$, so

$$L_i = \frac{1}{M} \sum_{k=1}^{M} \sum_{a_{i+1}=a_i}^{CR} P(x = a_{i+1}) \times (a_{i+1} - X) \times 2T,$$

where $a_{i+1} > C - W^r_{a_i}(S+H)/(2TS) = X$.

If $P(x = a_i)$ is known, then it may be substituted into Eq. (2). After the beginning of this period, cwnd rises during TCP's congestion avoidance phase. The mean data loss conditional upon heavy congestion occurring is $\bar{L} = \Sigma_{a_i} P(x = a_i) L_i$, which is also the amount of buffer storage required at each core switch port in order to accommodate temporary traffic surges. The peak loss rate is always restricted by the value of R because the congestion control algorithm in TCP always seeks out and consumes available transmission capacity.

2) With DLCC: Because of DLCC's rapid reaction to congestion, the equation for the loss in period i that was provided above is modified by replacing the round-trip time by twice the propagation delay between the edge router and the core optical switch experiencing heavy congestion:

$$L_i = \frac{1}{M} \sum_{k=1}^{M} \sum_{a_{i+1}=a_i}^{CR} P(x = a_{i+1}) \times (a_{i+1} - X) \times 2T_1.$$

In the network scenario shown at the top of Fig. 1, the propagation delay experienced when traversing the network is separated into periods of length T_0, T_1, T_2, and T_3, with $T = T_0 + T_1 + T_2 + T_3$. Without sending artificial ACK segments to the transmitting TCP process (i.e., without using the OPS ACKs scheme),

$$\begin{aligned} \bar{B}_1^c = \sum_{a_i} P(x = a_i) \frac{1}{M} \sum_{k=1}^{M} \sum_{a_{i+1}=a_i}^{CR} P(x = a_{i+1}) \\ \times (a_{i+1} - X) \times (2T - 2T_1). \end{aligned}$$

With full dual-layer congestion control, incorporating the OPS ACKs scheme,

$$\bar{B}_2^c = \sum_{a_i} P(x=a_i)\frac{1}{M}\sum_{k=1}^{M}\sum_{a_{i+1}=a_i}^{CR} P(x=a_{i+1}) \times (a_{i+1}-X)\times 2T_0.$$

E. Several Synchronized TCP Flows With Changing UDP Transmission Rate

TCP global synchronization results in fast recovery occurring periodically at the same time in all TCP processes and arises due to two mechanisms:

1. The sliding window flow control algorithm in TCP, which tends to produce periodic bursts of packets.
2. The drop-tail queue at the bottleneck, which drops all packets that arrive when the buffer is full [7].

Although TCP global synchronization is rare in the current Internet, it is nevertheless considered here. As discussed above, optical core switches can only have very small buffers, so TCP global synchronization is unlikely. A similar conclusion was previously reached where relatively large optical buffers larger than 50 packets were assumed [22].

When analyzing many (N) synchronized TCP flows sharing a link, high throughput is achieved with buffers of at least $C\times$RTT packets in size. Here C is the bottleneck link capacity, and RTT is TCP's end-to-end round-trip time. In a WAN such as the current Internet, RTT may be several hundreds of milliseconds or more. It is not technologically feasible for OPS core switches to have such large buffers in the foreseeable future—in the OPS network core switches modeled by the simulations later in this paper, it was assumed that each had sufficient fiber delay line (FDL) buffering to hold 20 slots. Due to the sawtooth fluctuations in the congestion window (cwnd), the throughput involved when having only a single TCP flow on a link, or having multiple synchronized TCP flows instead, is always lower than the bottleneck link capacity. Again, two cases are considered below: without DLCC and with DLCC.

1) Without DLCC: Throughout, the suffix N denotes the number of TCP flows. For each TCP flow, using similar reasoning to the derivation above,

$$W_{a_i}^{p_N} = \frac{2T(C-a_i)S}{N(S+H)}, \tag{7}$$

$$M_N = \frac{2T(C-a_i)}{N(S+H)}, \tag{8}$$

$$X_N = C - NW_{a_i}^{r_N}(S+H)/(2TS)\ , \tag{9}$$

$$W_k = kS. \tag{10}$$

The mean packet loss for N synchronized TCP flows is therefore obtained by combining Eqs. (7)–(10):

$$L_1^N = \frac{1}{M_N}\sum_{k=1}^{M_N}\sum_{a_{i+1}=a_i}^{CR} P(x=a_{i+1})\times(a_{i+1}-X_N)\times 2T.$$

Comparing this with the earlier result for one larger TCP flow shows that, as expected, this new result is equal to L_i, because as far as the analysis is concerned, N TCP flows make use of capacity in exactly the same way as one larger flow.

2) With DLCC: Based on the similar reasoning to that used for a single TCP flow, the following can readily be derived:

$$L_1^N = \frac{1}{M_N}\sum_{k=1}^{M_N}\sum_{a_{i+1}=a_i}^{CR} P(x=a_{i+1})\times(a_{i+1}-X_N)\times 2T_1.$$

Without sending artificial ACK segments to the transmitting TCP process (i.e., without using the OPS ACKs scheme),

$$\bar{B}_1^c = \sum_{a_i} P(x=a_i)\frac{1}{M_N}\sum_{k=1}^{M_N}\sum_{a_{i+1}=a_i}^{CR} P(x=a_{i+1}) \times(a_{i+1}-X_N)(2T-2T_1).$$

With full dual-layer congestion control, incorporating the OPS ACKs scheme,

$$\bar{B}_2^c = \sum_{a_i} P(x=a_i)\frac{1}{M_N}\sum_{k=1}^{M_N}\sum_{a_{i+1}=a_i}^{CR} P(x=a_{i+1})(a_{i+1}-X_N)2T_0.$$

F. Large Number of Nonsynchronized TCP Flows With Changing UDP Traffic Level

Even with many TCP flows being carried over the same link, global synchronization is extremely unlikely in real wide area networks because the requirements for it are as follows:

1. All TCP flows should have their segments lost in the bottleneck router simultaneously.
2. All TCP flows should start simultaneously or at least carry out TCP fast recovery at nearly the same time.
3. All receiver TCP processes should have rwnd $>$cwnd.
4. All TCP flows should have the same RTTs and be long lived.

It is extremely unlikely for all of these conditions to be satisfied at the same time in a real wide area network. With more than 200 flows, global synchronization has never been observed in practice, and measurements in the current Internet give no indication of it [4]. Also, the buffer size required for many TCP flows to achieve

nearly full utilization is just $C \times \mathrm{RTT}/\sqrt{N}$ [4,23,24], challenging the existing rule of thumb, namely, $C \times \mathrm{RTT}$.

Assume that the link capacity is nearly fully utilized by many TCP flows. With a large number of flows, i.e., $N \to \infty$, the transmission capacity occupied by TCP is always $1 - a_i$. Intuitively, this is because one or more TCP flows' windows halving in size will not cause the transmission utilization to change significantly, due to the infinitesimally small contribution by one TCP flow to the overall traffic carried. Due to the fair capacity sharing mechanism inherent in TCP, for period i, the mean capacity used by each flow is $(C - a_i)/N$. Thus because all TCP flows are unsynchronized, one flow having a packet dropped does not influence throughput appreciably, whereas if many TCP flows were synchronized, they all would halve their congestion window simultaneously due to fast recovery, resulting in reduced throughput.

The mean increase in capacity, r_i^e, for the UDP rate when heavy congestion occurs is given by

$$r_i^e = \sum_{a_{i+1}=a_i}^{CR} P(x = a_{i+1}) \times (a_{i+1} - a_i).$$

1) Without DLCC

As discussed above, $(C - a_i)/N$ is the mean capacity of each TCP flow, arising due to fair sharing of transmission capacity between flows. Assume that the UDP transmission rate increases, i.e., that $a_{i+1} > a_i$. Also assume that $K = (a_{i+1} - a_i)/[(C - a_i)/2N]$, so that K is the number of TCP flows that must halve their transmission rate to accommodate the extra UDP traffic, which is equal to the ratio of the increase in the UDP rate to half the mean rate of one TCP flow. Then the duration of the heavy congestion period is given by

$$2T + \frac{(K-1)(M_N 2T/2)}{N}.$$

Hence the mean number of optical packets lost during a heavy congestion period is

$$L_i = r_i^e 2T + \frac{r_i^e}{2}\frac{(K-1)(M_N 2T/2)}{N}.$$

Here, $P(B \mid (x = a_i)) = P(a_{i+1} > a_i)$, so

$$P(a_{i+1} > a_i) = \frac{CR - a_i}{CR}.$$

Hence the number of packets lost due to heavy congestion over the entire duration being analyzed is

$$\bar{L} = \sum_{a_i} P(x = a_i) L_i.$$

2) With DLCC

Using similar reasoning to that above, the mean number of optical packets lost during a heavy congestion period is

$$L_i = r_i^e 2T_1.$$

Hence the number of packets lost due to heavy congestion over the entire duration being analyzed is now

$$\bar{L} = \sum_{a_i} P(x = a_i) r_i^e 2T_1.$$

Without sending artificial ACK segments to the transmitting TCP process (i.e., without using the OPS ACKs scheme),

$$\bar{B}_1^c = \sum_{a_i} P(x = a_i) r_i^e (2T - 2T_1)\ .$$

With full dual-layer congestion control, incorporating the OPS ACKs scheme,

$$\bar{B}_2^c = \sum_{a_i} P(x = a_i) r_i^e 2T_0.$$

VI. Numerical Results From Mathematical Analysis

This section presents and discusses results from the mathematical model of Section V in order to justify the validity of the DLCC concept. The scenario used for analysis is shown at the top of Fig. 1. R, the maximum proportion of the link capacity that UDP traffic can use is 0.1, 0.2, 0.3, or 0.4, reflecting the increasing use of new streaming applications. The remaining capacity is used by TCP traffic, with the full link capacity C being 10 Gbits/s. The number of nonsynchronized TCP flows is 100, 200, 400, 1000, 4000, or 10,000. The TCP end-to-end round-trip time is varied between 60, 100, 180, 280, 420, and 560 ms, which covers both short RTTs (60–280 ms) and long RTTs (280–560 ms). Nonreactive background UDP traffic is considered in these results. In the computation below, each possible density of UDP traffic is assumed to be equiprobable, in order to provide an upper bound on the worst case, where the nonreactive UDP traffic has frequent and abrupt fluctuations. In Fig. 1, it is assumed that $T_1/T_0 = T_2/T_3 = 0.5$. Mean packet loss during a heavy congestion period and buffering requirements are calculated in terms of the number of optical packets, each of which is assumed to be fixed at 2000 bytes in length.

Figures 2 and 3 show the mean heavy congestion loss rate for both a single TCP flow and for multiple synchronized TCP flows and also for 10,000 nonsynchronized TCP flows under both noncontrolled (without DLCC) and controlled (with DLCC) scenarios, with changing parameters R and RTT. For both a

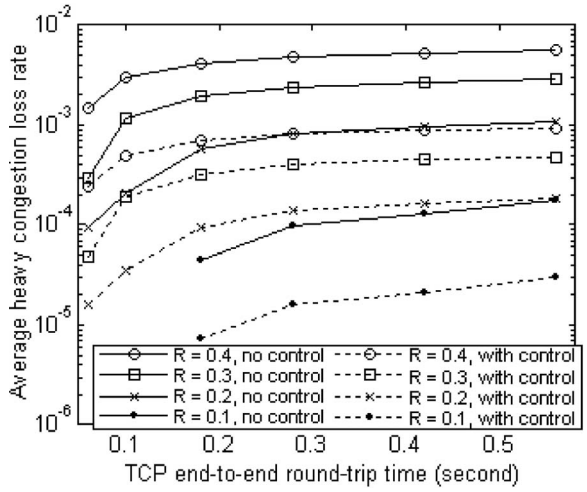


Fig. 2. Mean loss rate during heavy congestion for a single TCP flow and for multiple synchronized TCP flows.

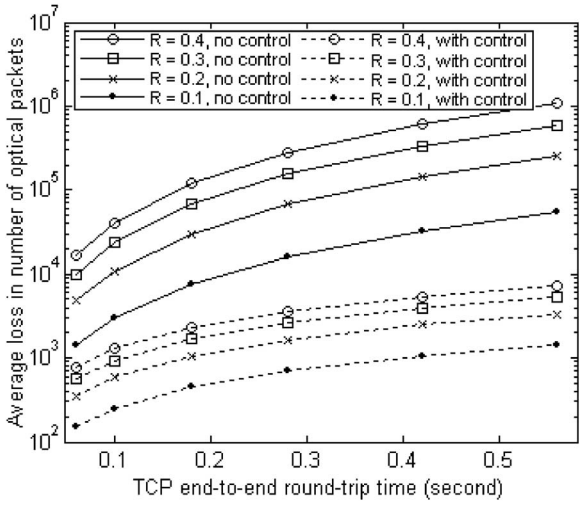


Fig. 4. Mean total number of lost optical packets over one period for 10,000 nonsynchronized TCP flows.

single TCP flow and for multiple synchronized TCP flows, the packet loss rate increases with RTT, whereas for 10,000 nonsynchronized TCP flows the loss rate is virtually unaffected by it. However, both cases show obvious degradation of the loss rate as the maximum UDP stream ratio R increases. Controlled schemes show better performance than noncontrolled schemes, especially for 10,000 nonsynchronized TCP flows.

Figure 4 shows the mean total loss per heavy congestion period measured in number of optical packets, showing the lost traffic for 10,000 nonsynchronized TCP flows and demonstrating that the controlled scheme has performance superior to the noncontrolled scheme. This improvement is shown in Figs. 5 and 6 and is achieved through the action of edge electronic buffering. It is also seen from these that dual-layer control reduces the edge electronic buffering requirements for both a single TCP flow and multiple synchronized TCP flows and also for the case of 10,000 nonsynchronized TCP flows. The controlled scheme reduces the packet loss due to congestion in two ways:

1. It moves much of the buffer capacity requirements from the optical core of the network to the edge. The buffer requirement calculated above for the noncontrolled scheme assumes only core optical buffering; however, the controlled scheme proposed here also has edge electronic buffers in order to reduce core buffering.
2. It uses a form of explicit congestion notification to actively reduce the core packet loss due to congestion loss and also to reduce the core buffering requirements.

Figure 7 shows the mean lost traffic during one heavy congestion period, with changes in both the RTT and

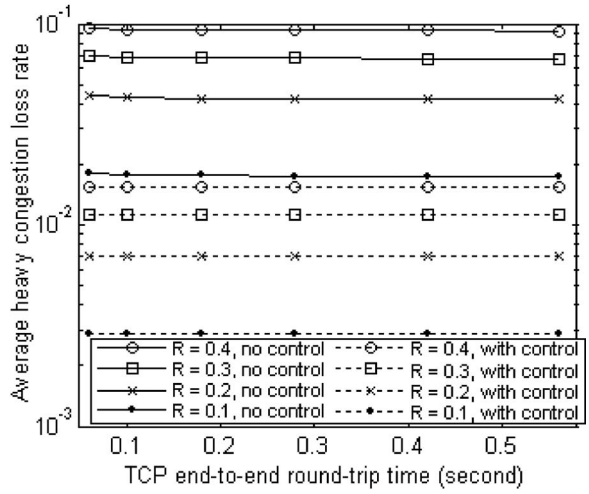


Fig. 3. Mean loss rate during heavy congestion for 10,000 nonsynchronized TCP flows.

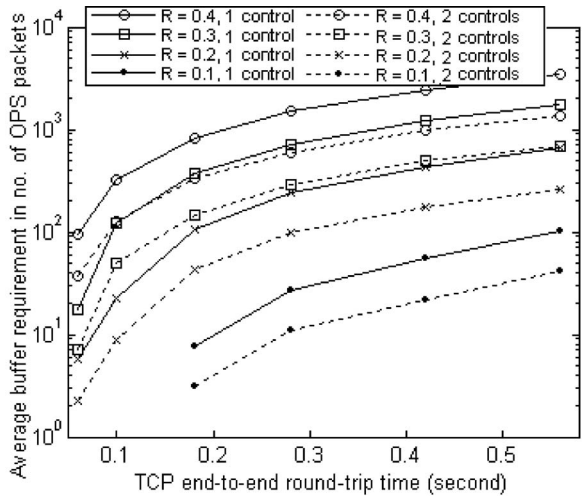


Fig. 5. Mean electronic edge buffer requirement in number of optical packets for a single TCP flow and for multiple synchronized TCP flows. With "1 control," the OPS ACKs scheme is not implemented, whereas "2 controls" refers to the use of full dual-layer control.

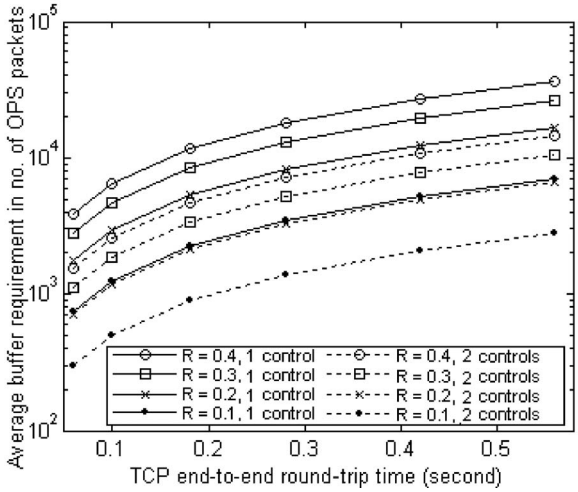


Fig. 6. Mean electronic edge buffer requirement in number of optical packets for 10,000 nonsynchronized TCP flows. With "1 control," the OPS ACKs scheme is not implemented, whereas "2 controls" refers to the use of full dual-layer control.

the number of nonsynchronized TCP flows. Performance is degraded as RTT increases in all situations, but shows convergence with increasing RTT. Packet loss decreases as the number of nonsynchronized TCP flows increases, and this decrease slows down with very many TCP flows.

VII. Simulation

In addition to the mathematical modeling reported in Sections V and VI, complementary aspects of DLCC performance, especially TCP goodput, were also evaluated via simulation. The simulation, implemented using the commercially available OPNET simulator [25], carried out an event-based simulation of the system as time elapsed; hence unlike the analytical model, it was necessarily limited in terms of traffic capacity, number of events simulated, number of TCP connections, and simulation time. However, the analytical modeling studied heavy congestion only, whereas the simulation considered all aspects of network behavior.

The simulation topology is shown at the top of Fig. 1. Edge routers could be configured either to support or not support DLCC. Forty FTP applications were configured to transfer files simultaneously; each file was sufficiently large so that, at the end of the measured simulation period, no file transfer had terminated. The TCP implementation employed was Reno with a very large value of rwnd, so that the window used was always equal to the congestion window set by the transmitter, cwnd. In each core optical packet switch, a simple feed-forward switch structure was assumed with 20 slot buffer positions shared by all the ports in the whole switch. For each UDP traffic source, bursts of traffic occurred at intervals of 10 s, with each burst consuming 10%, 20%, 30%, or 40% of

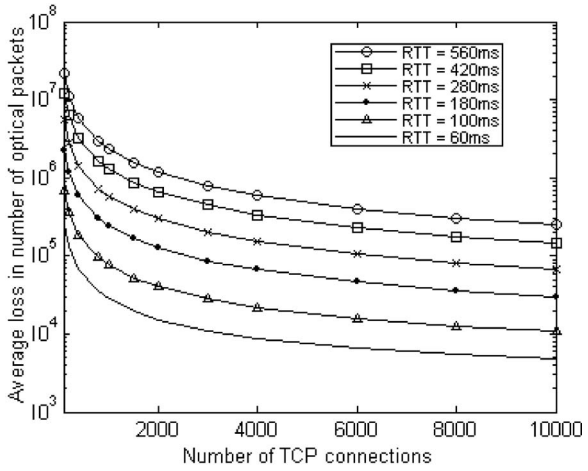


Fig. 7. Mean loss of the scenarios with control (with DLCC) during one congestion period, expressed as number of optical packets against number of nonsynchronized TCP flows.

the link transmission capacity, and in each burst, no further packets were sent after 200 ms. In the simulation topology shown at the top of Fig. 1, the propagation times are related by $T_1/T_0=T_2/T_3=0.5$. Each IP datagram was 576 bytes long, and for simplicity, each optical packet only contained exactly one IP datagram, although this does not affect the conclusions. For convenience during simulation, the overall capacity of each core link was set to 55 Mbits/s on each core switch, although this affects neither the conclusions nor the principle being demonstrated.

Six simulation runs, each having a simulation time of 44 min 37 s, were carried out for each point in Fig. 8 through to Fig. 12, in order to compute 95% confidence intervals. In these graphs, the confidence intervals are shown as dotted curves on either side of the

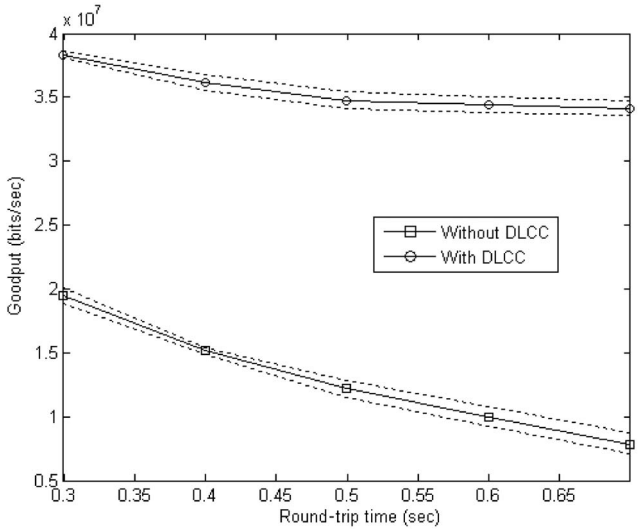


Fig. 8. Goodput comparison with varying end-to-end round-trip time, assuming that a maximum of 30% of the transmission capacity can be used by UDP traffic.

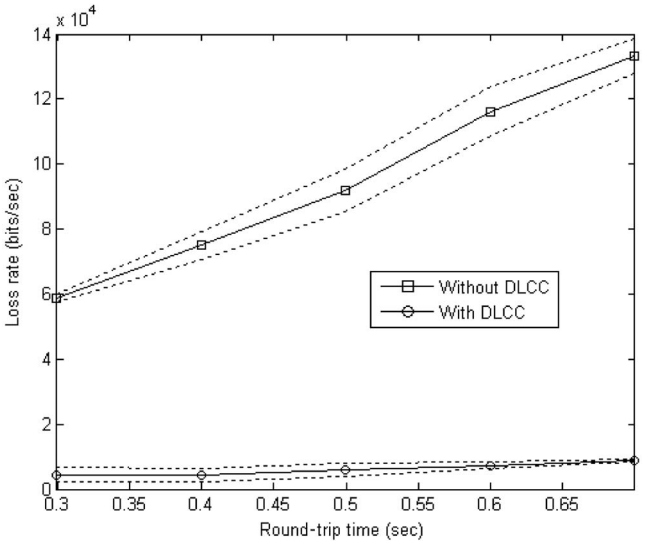


Fig. 9. Loss rate comparison with varying end-to-end round-trip time, assuming that a maximum of 30% of the transmission capacity can be used by UDP traffic.

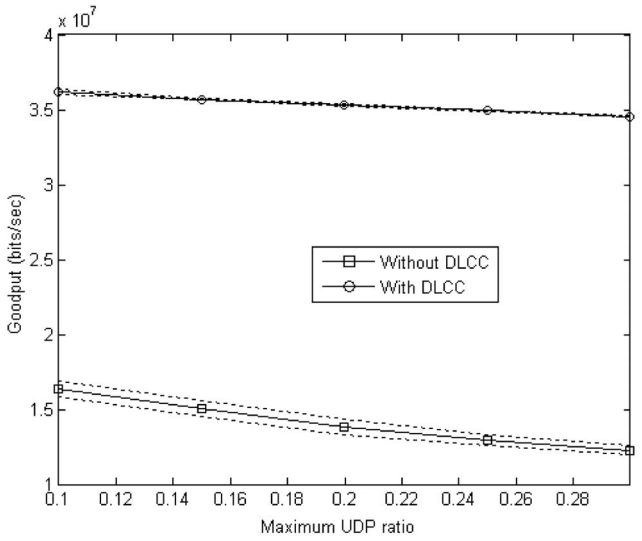


Fig. 11. Goodput comparison with varying maximum UDP capacity ratio, assuming a round-trip time of 500 ms.

mean, which is shown as a solid curve in each case. These simulations collected the following statistics:

- **Goodput** is calculated as the total traffic carried over the network (measured in bits) minus the total dropped traffic (also measured in bits), divided by the total simulation time.
- **Loss rate** is calculated as the total dropped traffic (measured in bits) divided by the total simulation time.
- **Extra edge buffer capacity** is measured in bits and is calculated as the moving average of the edge buffer capacity that would be required over the whole duration of the simulation in order to retain extra traffic that would occur without the OPS ACKs scheme after notification from the core is received.

Figures 8–10 assume that a maximum of 30% of the transmission capacity can be used by UDP traffic. To mimic a large WAN, Figs. 11 and 12 both assume a 500 ms end-to-end round-trip time.

Figures 8 and 11 show that with either changing the RTT or changing the ratio of the UDP capacity, the scenario without DLCC always has poorer goodput performance; it is also seen that with either increasing the RTT or increasing the proportion of transmission capacity used by UDP traffic, the goodput for both scenarios shows a downward trend. With DLCC, goodput decreases with increased RTT because the distance (and hence the propagation delay) between a congested core switch and a corresponding edge router

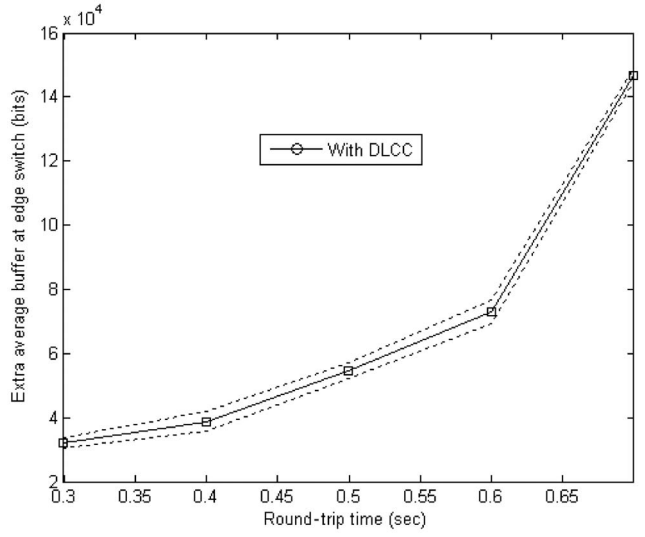


Fig. 10. Extra edge buffer capacity with varying end-to-end round-trip time, assuming that a maximum of 30% of the transmission capacity can be used by UDP traffic.

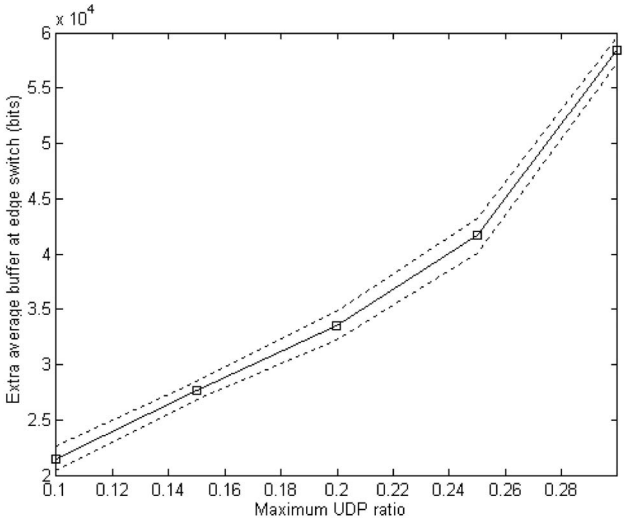


Fig. 12. Extra edge buffer capacity with varying maximum UDP capacity ratio, assuming a round-trip time of 500 ms.

is increased, which delays the arrival of each congestion notification at the edge router that responds to it. Figure 9 shows that the loss rate for DLCC only changes slightly with varying RTT, whereas without DLCC, it changes dramatically. This implies that DLCC is better at preventing packet loss, due to its quicker reaction to congestion.

Figures 10 and 12 show that the extra edge buffer capacitying requirement increases with either an increase of the RTT or an increase in the ratio of the UDP capacity. This explains the superior performance in terms of goodput and loss rate for DLCC. It also implies that DLCC trades off extra core switch FDL buffering (which is difficult and expensive, or even impossible, to implement) with edge router electronic buffering that is easy and inexpensive to implement. Comparison of Figs. 8 and 9 with performance evaluation results for a different approach to the same problem, involving ACK pacing [10], show that DLCC exhibits lower packet loss and higher gains in goodput over a classical network with no additional congestion control mechanisms. This is because DLCC initiates a more aggressive response to core network congestion than ACK pacing, halving the transmitter's congestion window size via TCP's fast recovery scheme.

VIII. Conclusions

In this paper, a dual-layer congestion control scheme was proposed, which copes with the lack of congestion control in optical-packet-switched networks and the mismatch of slow TCP end-to-end congestion control with the high transmission capacities inherent in future deployment of OPS. It interacts with both electronic and optical domain traffic in order to both react quickly to heavy congestion in the optical-packet-switched core and also quench electronic domain traffic at the origin over a longer time scale. There is no specific requirement for protocol support or any modifications to existing implementations of TCP at each source. Signaling is not end-to-end flow based, but edge-to-edge flow based, and requires little signaling traffic, therefore exhibiting good scalability.

An analytical model was constructed to model congestion when elastic long-lived TCP flows confront nonreactive UDP streams. It shows that dual-layer congestion control improves network performance, decreasing packet loss by up to six times, due to the deployment of a combination of edge electronic buffering and early congestion reporting to edge routers. This scheme shifts much of the buffering requirement from the optical core switches to the edge routers, where electronic buffering is freely available; therefore, the core switch optical buffering requirements can be greatly reduced. Indeed, each optical core switch is assumed in the modeling work to have a shared optical buffering capacity of only 20 optical packets for all ports. DLCC also uses a form of explicit congestion notification to actively reduce both segment loss due to congestion in the core of the network and buffering requirements at the traffic source. Furthermore, simulation modeling shows that DLCC yields a TCP goodput improvement of between 2 and 10 times, depending on the amount of background UDP traffic and the round-trip time. Both analytical and simulation models are derived from different perspectives to validate the effectiveness and trade-offs implicit in this scheme.

Acknowledgments

This work was supported by the SOAPS (Smoothed Optical ATD Packet Switching) project, a collaborative LINK project funded by the UK Department of Trade and Industry. The authors thank the anonymous reviewers of this paper as well as Joseph Sventek of Glasgow University and Martin Reed of the University of Essex for their helpful comments on previous drafts.

References

[1] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *J. Lightwave Technol.*, vol. 16, pp. 2081–2094, 1998.

[2] R. S. Tucker, "The role of optics and electronics in high-capacity routers," *J. Lightwave Technol.*, vol. 24, pp. 4655–4673, 2006.

[3] R. S. Tucker, P.-C. Ku, and C. J. Chang-Hasnain, "Slow-light optical buffers: capabilities and fundamental limitations," *J. Lightwave Technol.*, vol. 23, pp. 4046–4066, 2005.

[4] G. Appenzeller, I. Keslassy, and K. McKeown, "Sizing router buffers," *ACM SIGCOMM Comput. Rev.*, vol. 34, no. 4, pp. 281–292, Oct. 2004.

[5] Z. Lu, D. K. Hunter, and I. D. Henning, "Contention reduction in core optical packet switches through electronic traffic smoothing and scheduling at the network edge," *J. Lightwave Technol.*, vol. 24, pp. 4828–4837, 2006.

[6] O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for small buffered optical packet-switched networks," *J. Opt. Netw.*, vol. 6, pp. 1116–1128, 2007.

[7] E. Hashem, "Analysis of random drop for gateway congestion control," Laboratory for Computer Science, MIT, Cambridge, MA, Rep. LCS TR-465, 1989.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, pp. 397–413, 1993.

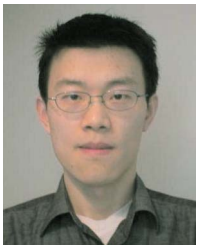
[9] S. Floyd, "TCP and explicit congestion notification," *Comput. Commun. Rev.*, vol. 24, pp. 10–23, 1994.

[10] F. Xue and S. Yoo, "TCP-aware active congestion control in optical packet-switched networks," in *Optical Fiber Communication Conf.*, 2003, paper MF108.

[11] S. Floyd, "A report on recent developments in TCP congestion control," *IEEE Commun. Mag.*, vol. 39, no. 4, pp. 84–90, Feb. 2001.

[12] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: recent traffic measurements from an Internet backbone," in *Proc. INET*, Geneva, Switzerland, 1998, paper 473.

[13] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, pp. 6–16, 2003.
[14] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of Internet traffic in 1998–2003," in *Proc. Winter Int. Symp. Information and Communication Technologies*, Cancun, Mexico, 2004, pp. 1–6.
[15] S. Iyer, S. Bhattacharyya, N. Taft, N. McKeown, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *22nd Annu. Joint Conf. IEEE Computer and Communications Societies*, 2003, pp. 406–416.
[16] J. He and S.-H. G. Chan, "TCP and UDP performance for Internet over optical packet-switched networks," in *IEEE Int. Conf. Communications*, 2003, pp. 1350–1354.
[17] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network*, vol. 11, pp. 10–23, 1997.
[18] X. Yu, Y. Chen, and C. Qiao, "Performance evaluation of optical burst switching with assembled burst traffic input," in *IEEE Global Telecommunications Conf.*, 2002, pp. 2318–2322.
[19] X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *J. Lightwave Technol.*, vol. 22, pp. 2722–2738, 2004.
[20] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, IETF, 1999.
[21] S. Floyd, "Congestion control principles," RFC 2914, IETF, 2000.
[22] G. Raina, D. Towsley, and D. Wischik, "Part II: Control theory for buffer sizing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 79–82, July 2005.
[23] D. Wischik and N. McKeown, "Part I: Buffer sizes for core routers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 75–78, July 2005.
[24] D. Wischik, "Buffer requirements for high-speed routers," in *31st European Conf. Optical Communication*, 2005, pp. 23–26.
[25] OPNET network simulator, http://www.opnet.com.



Zheng Lu received the B.Eng. degree in electronic engineering from the PLA University of Science and Technology, Nanjing, China, and the M.Sc. degree (with distinction) in telecommunications and information systems from the University of Essex, Colchester, UK, in 2002 and 2003, respectively. In 2008, he graduated from the University of Essex with a Ph.D., which was awarded for research on protocols for optical-packet-switched networks. From January 2007 until August 2008, he was a Research Officer at the University of Essex. Since September 2008, he has been working for Acorah Software Products Ltd., Wokingham, Berkshire, UK. His research interests concentrate on algorithms, protocol design, and modeling for data communications and wireless sensor networks.

David K. Hunter (M'09) became a Student Member (S) of IEEE in 1988, a Member (M) in 1990, and a Senior Member (SM) in 2000. In 1987, he obtained a first class honors B.Eng. in electronics and microprocessor engineering from the University of Strathclyde, Glasgow, UK, and received a Ph.D. from the same university in 1991 for research into optical TDM switch architectures. After obtaining his Ph.D., he was a Research Fellow then a Senior Research Fellow at the University of Strathclyde, researching optical networking and optical packet switching, and holding an EPSRC Advanced Fellowship from 1995 to 2000. After spending a year as a Senior Researcher in Marconi Labs, Cambridge, UK, he moved to the University of Essex, Colchester, UK, in August 2002, where he is a Reader in the School of Computer Science and Electronic Engineering. He has authored or co-authored over 125 publications. Dr. Hunter has acted as an external PhD examiner for the Universities of Cambridge, London, Strathclyde, and Essex. From 1999 until 2003 he was an Associate Editor for the *IEEE Transactions on Communications*, and he was an Associate Editor for the IEEE/OSA *Journal of Lightwave Technology* from 2001 until 2006. He participated in editing a special issue of that journal, on optical networks, that was published in December 2000. He is a Chartered Engineer, a Member of the IET, and a Professional Member of the ACM.