ELSEVIER

Computer Networks 39 (2002) 207–219

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Avoiding congestion collapse on the Internet using TCP tunnels [☆]

B.P. Lee [1], R.K. Balan [2], L. Jacob, W.K.G. Seah, A.L. Ananda [*]

Department of Computer Science, Centre for Internet Research, School of Computing, National University of Singapore, Lower Kent Ridge Road, 3 Science Drive 2, Singapore 117543, Singapore

Received 12 October 2001; accepted 14 December 2001

Responsible Editor: I.F. Akyildiz

Abstract

This paper discusses the application of TCP tunnels on the Internet and how Internet traffic can benefit from the congestion control mechanism of the tunnels. Primarily, we show the TCP tunnels offer TCP-friendly flows protection from TCP-unfriendly traffic. TCP tunnels also reduce the many flows situation on the Internet to that of a few flows. In addition, TCP tunnels eliminate unnecessary packet loss in the core routers of the congested backbones, which waste precious bandwidth leading to congestion collapse due to unresponsive UDP flows. We finally highlight that the use of TCP tunnels can, in principle, help prevent certain forms of congestion collapse described by Floyd and Fall [IEEE/ACM Trans Networking 7 (4) (1999) 458].

The deployment of TCP tunnels on the Internet and the issues involved are also discussed and we conclude that with the recent RFC2309 recommendation of using random early drop as the default packet-drop policy in Internet routers, coupled with the implementation of a pure tunnel environment on backbone networks makes the deployment of TCP tunnels a feasible endeavour worthy of further investigation. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: TCP tunnels; Aggregation; Quality of service; Congestion collapse; Queue management; Flow back-pressure; Random early drop routers

1. Introduction

The evolution and deployment of new network technologies involving the provision of bandwidth has continually fed the insatiable end user's appetite for even more bandwidth. Whenever a leap in network technology for provisioning bandwidth occurs, the increasing end user demands for more bandwidth threatens to consume the available capacity and stress the limits of technology. In addition, the increasing use of TCP-unfriendly and unresponsive flows, due to the proliferation of

[☆] A version of this paper was presented in the IEEE LCN 2000 conference, Tampa, Florida.

[*] Corresponding author.

E-mail addresses: lee.bp@unityintegration.com (B.P. Lee), rajeshkr@comp.nus.edu.sg (R.K. Balan), jacobl@comp.nus.edu.sg (L. Jacob), ananda@comp.nus.edu.sg (A.L. Ananda).

[1] Present address: Unity Wireless Integration, 1123 Serangoon Road, #02-01 UMW Building, Singapore 328207, Singapore.

[2] Present address: Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15232, USA.

multimedia applications involving voice and video, is straining the capacity of the Internet and compounds the problem of tackling congestion on the Internet.

A TCP tunnel is a TCP circuit, which carries IP packets over the Internet. By encapsulating user flows using this construct, we hope to benefit from the congestion control mechanism of TCP/IP through the segregation of unresponsive flows from TCP-friendly flows. Kung and Wang [10] did related work on the properties of TCP trunks, which are similar to TCP tunnels. In their paper, they demonstrate how TCP trunking protects web users from competing ftp traffic. They also examine how TCP trunking may be used to allow a site to control its offered load into a backbone network so that the site can assure some quality of service (QoS) for its packets over the backbone.

This paper studies the behaviour and properties of TCP tunnels in conjunction with random early drop (RED) routers [7] and examines the benefits and disadvantages they confer on the IP traffic. We also study the impact on the congestion within network backbones, and the protection that tunnels offer with respect to the various competing classes of traffic in terms of bandwidth allocation and reduced retransmissions. We try to identify the network conditions and scenarios where the deployment of TCP tunnels would be beneficial to overall network performance. We also discuss how TCP tunnels can assist in avoiding certain forms of congestion collapse on the Internet as considered by Floyd and Fall [6].

We believe that this study of TCP tunnels would be most relevant now, considering the fact that commerce on the Internet has fueled significant interest and efforts in the deployment of virtual private networks (VPNs) over the Internet, particularly in assisting resource sharing, work collaboration and meeting privacy concerns. Typically, VPNs are layered over the Internet and these VPNs are deployed using tunnels. These tunnels appear as layer-2 constructs to user applications.

We envision one or more TCP tunnels installed between border routers, which connect the LANs to the WANs or the Internet cloud (see Fig. 1). Later, we demonstrate that this architecture has the effect of confining packet drops to the LAN side and helps to keep congestion low on core routers (also known as WAN routers).

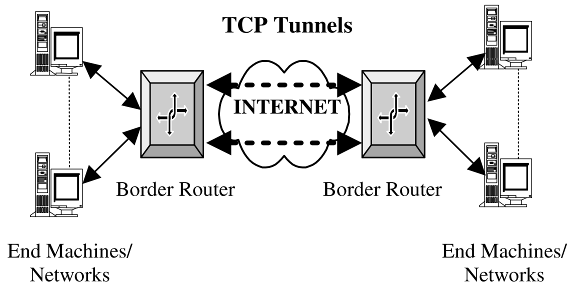

Fig. 1. TCP tunnels as transport mechanisms for IP flows over the Internet.

TCP tunnels can also be deployed by Internet Service Providers (ISP) on point-to-point links to take advantage of attributes of TCP tunnels and offer better service. For instance, if traffic from interactive applications such as telnet or http arrive at a last hop bottleneck link and are liable to be dropped, TCP tunnels can be deployed over this link to offer protection from losses due to congestion.

The experiments that are described in this paper make reference to the testbed depicted in Fig. 2. RedHat 6.0 with Linux Kernel version 2.2.12 are installed on our PCs, all of which are Intel Celeron 300A processor-based with 128 MB of main memory each. The network cards we use are the Intel Ether Express Pro 10/100 (100 Mbps) models.

We hacked a traffic shaping kernel module called "rshaper" and installed it onto the delay/error box. The delay/error box buffers and delays

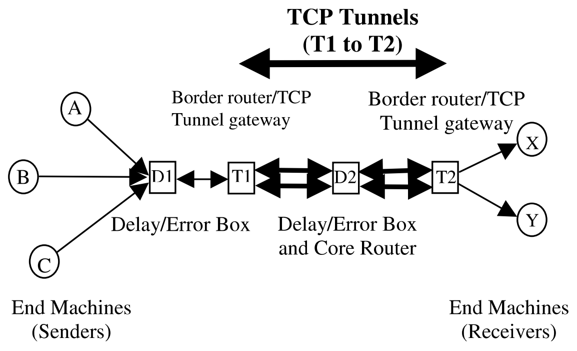

Fig. 2. Testbed.

packets to simulate various latencies, and selectively drops and corrupts packets according to the uniform distribution. We use *ttcp* [11] and *iperf* [8] to generate bulk TCP traffic and *Surge* [2] for generating http traffic. *tcpdump* is used to collect the traffic and the traffic analysis is carried out using *tcptrace*. RED and Tail-Drop are implemented using the Linux Traffic Control Extensions [1].

We developed our TCP tunnel implementation in C. The TCP tunnel transmitter receives IP packets from the in-coming interface using *libpcap* routines and re-transmits these packets over one or more TCP connections, which we term TCP tunnels. We configure a Tail-Drop queue of 1000 packets leading into each TCP tunnel. The TCP tunnel receiver reads the IP frames from the various TCP connections and writes them out to the outgoing interface using raw sockets.

The rest of the paper is as organised as follows: Section 2 will cover the protection that tunnels can offer to TCP-friendly flows in various scenarios. The effects of TCP tunnels and RED on router queue lengths are discussed in Section 3. Section 4 elaborates on the congestion control property of TCP tunnels and explains how this attribute gives rise to back-pressure which pushes congestion to edge routers from the core routers. The types of congestion collapse that may arise on the Internet and the congestion control property of tunnels that can help to prevent some forms of congestion collapse are discussed in Section 5. Section 6 covers the possible application of tunnels in the provisioning of proportional differentiated services over the Internet and Section 7 touches on the deployment issues for TCP tunnels.

2. Protection of TCP-friendly flows

UDP traffic is not TCP-friendly [6]. Such TCP-unfriendly traffic does not respond to packet drops which typically signals congestion and critical situations of starved router and link resources. This aggressive behaviour degrades and even shuts out TCP-friendly flows such as bulk transfers (ftp) and interactive applications (http and telnet) and prevents them from obtaining their fair share of their bandwidth when they compete for bandwidth over a congested link.

In this section, we examine how TCP tunnels can be deployed to isolate different types of traffic from one another and protect interactive user applications from aggressive flows.

2.1. Protection of TCP bulk flows from unresponsive UDP flows

In this section, we show how TCP flows carrying bulk traffic are protected from aggressive UDP flows over a congested link. We assume that a number of tunnels are deployed over the link, each carrying a specific kind of traffic, and that they compete with each other for bandwidth. All traffic flows, UDP or TCP, traversing the congested link has to go through at least one of the tunnels. As a result, UDP flows are prevented from clogging the congested link as the tunnels carrying the UDP flows will suffer packet drops and will be forced to throttle their sending rates. Here, we see that the UDP flows are endowed with a crude form of end-to-end congestion control by virtue of being wrapped in TCP tunnels.

We demonstrate this by sending a 10 Mbps UDP flow (CBR data) from A to X and 100 TCP flows (representing 100 concurrent file transfers of 512 KB each) from C to Y (see Fig. 3). The end-to-end round trip time (RTT) is 10 ms and all the links are 10 Mbps except the link D2–T2 which is 1 Mbps. The UDP flow is very aggressive and

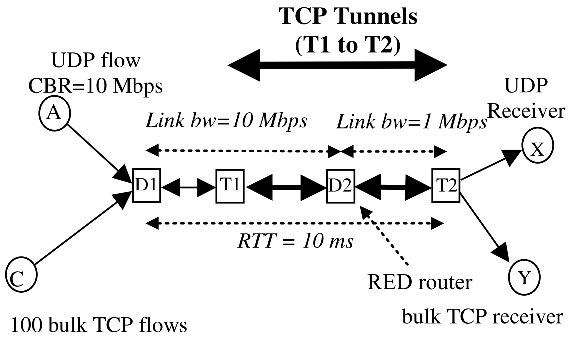

Fig. 3. Tunnel offering protection of TCP bulk flows from unresponsive UDP flows.

Table 1
Per TCP flow statistics obtained from tcptrace analysis program

Per TCP flow statistics	Without TCP tunnel	With TCP tunnel	% Improvement
RTT average (ms)	4133.49	15821.83	(−) 282.77
RTT minimum (ms)	1292.33	1200.14	(+) 7.13
RTT maximum (ms)	4563.39	17103.36	(−) 274.79
Number of RTT samples	83.16	168.50	(+) 102.62
Retransmitted bytes	385783.90	57773.36	(+) 567.75
Retransmitted packets	266.78	39.93	(+) 568.12
Retransmitted data segments over total segments	28.10	6.10	(+) 360.51
Data segments sent	624.45	397.47	(+) 57.11
Acknowledgement segments sent	324.27	247.32	(+) 31.11
Data segments over total segments	65.82	61.66	(+) 6.74
Total segments	948.72	644.79	(+) 47.14
Average throughput (kbits/s)	1.18	2.85	(+) 141.53

unresponsive to the behaviour of other co-existing. The MTU is 1500 bytes for both sources of traffic. The bottleneck link of 1 Mbps is fed by a RED queue (with parameters $\min_{th} = 20$ KB, $\max_{th} = 520$ KB, $\max_p = 0.02$). The run is considered complete when all file transfers are complete. We use the above scenario to run two experiments; one without TCP tunnels and one with TCP tunnels (tunnels deployed between T1 and T2). The results of this experiment are shown in Table 1.

Without the tunnels, we observe that the UDP flow consumes most of the bandwidth, leaving each TCP flow with a mean throughput of 1.18 Kbps (and standard deviation of 0.15 Kbps).

Things are different when we use TCP tunnels to isolate both types of traffic. Here, the mean throughput of the TCP flows increases to 2.85 Kbps (standard deviation of 1.92 Kbps) (see Fig. 4). The UDP flow is carried over one tunnel and the TCP traffic is carried over a separate tunnel. Both tunnels are fed by their own Tail-Drop queues, each of which has a capacity of 1000 packets. Should the TCP tunnels be served by a single shared queue, the greedy non-responsive UDP flow would just hog this queue (on T1) and shut out the TCP flows.

When we run the UDP flow against a few TCP flows (less than 10), we observe that the TCP flows are totally shut out and none could complete their transfers. However, the situation improves when we increase the number of competing TCP flows to a significant figure, say, 100 as given above.

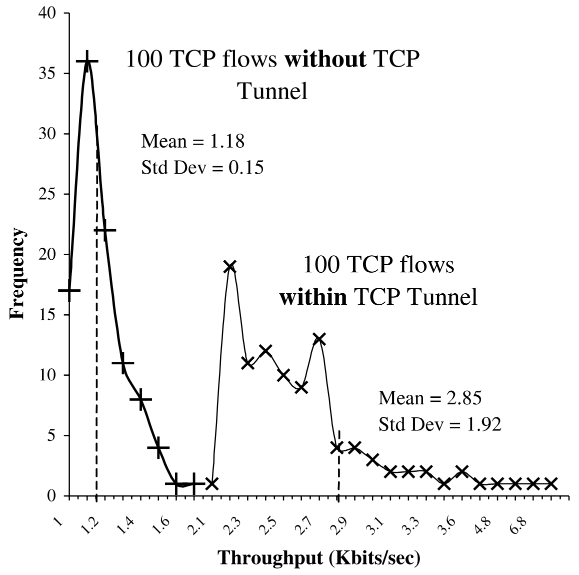

Fig. 4. Histogram of average throughput of 100 (TCP flows) concurrent file transfers of 512 KB each.

In the course of the experiments, we notice that the aggressiveness of the TCP tunnels is determined by the nature of the tunnels' payloads. A tunnel carrying UDP traffic is more aggressive when pitted against another tunnel, which carries a few TCP flows. Presumably, when either tunnel suffers a packet drop, it reduces its congestion window and propagates this effect to its payload. In the case of the tunnel carrying the TCP flows, the user TCP flows continually decrease their transmission rates, unlike the UDP flow whose transmission rate is unaffected.

From Table 1, we see that TCP tunnels reduce the overall amount of traffic sent. The amount of retransmissions per connection is reduced by over 500% as the tunnel provides reliability in the face of congestion. Packet loss from the tunnel due to heavy congestion has minimum impact on TCP flows and precious link bandwidth is not wasted on retransmissions.

However, the tunnels degrade the TCP RTT estimations badly (up to 280%!). Note that when the TCP tunnel is forced to throttle its sending rate due to congestion at the bottleneck link, packets get queued up at the tunnel ingress. The sources, which are oblivious to the congestion, continue to send at an increased rate until the queues at the tunnel ingress overflow. Only packets lost due to this overflow, are retransmitted by the sender.

2.2. Protection of interactive traffic over congested links

Interactive end user applications such as http often suffer long delays and even timeouts over WAN links as they usually traverse over a bottleneck link and compete with other types of traffic. Usually, the bottleneck link is a single hop of low latency. The http connections suffer packet drops at this link, which leads to inevitable timeouts.

An ISP may choose to deploy TCP tunnels over this short latency bottleneck link to isolate the interactive traffic from other aggressive traffic. The TCP tunnels will also handle retransmissions over this congested link and minimise retransmissions by the end user applications and possible timeouts.

To demonstrate this scenario, we set-up the configuration shown in Fig. 5 using the testbed. T1 and T2 are the tunnel end-points and the TCP tunnels are layered over this 1 Mbps bottleneck. Note that this bottleneck link is limited to 1 Mbps in the forward direction (towards X and Y) and the reverse path is of bandwidth 10 Mbps. The experimental run lasts 300 s. The 100 http connections from B to Y are started first and the UDP flow of 5 Mbps, from A to X, is started 10 s later. Note that each of the traffic classes is carried by a separate tunnel.

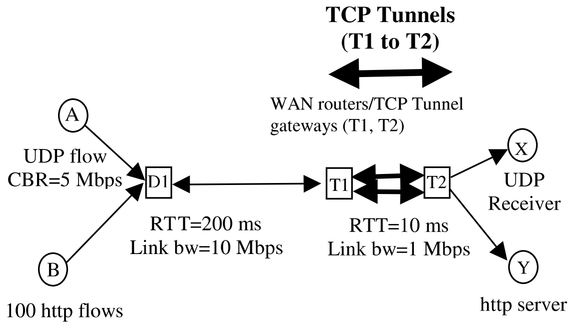

Fig. 5. Tunnel offering reliability for interactive flows over congested bottleneck link (T1 to T2).

From Table 2, we see that the http connections protected by the TCP tunnels managed to achieve higher average packet and aggregated throughput. More importantly, for the same experimental run and duration, a greater number of successful and complete http connections were made with the help of the tunnels. Without the protection of the tunnels, numerous http connections suffered timeouts and were unable to complete their transfers within the same interval (i.e. 300 s).

2.3. Protection from fragmentation

The performance of user connections on the Internet is limited by the Path MTU [14]. This refers to the largest size that an IP datagram can take which permits it to traverse the network without undergoing fragmentation.

In [9], it was argued that fragmentation is inefficient, the loss of any fragment degrades

Table 2
Http flow statistics

Per http flow statistics	Without TCP tunnels	With TCP tunnels
Packets in forward direction (http requests)	13.64	25.48
Packets in reverse direction (http data)	13.11	30.00
Total packets transferred	27.75	55.48
Forward/reverse aggregated throughput (bps)	1260/ 64,247	1582/ 126,325
Successful/total http connections made	45/208 (21.6%)	290/624 (46.5%)

performance due to retransmission and that efficient reassembly of fragments is hard. Kent and Mogul [9] concluded that the disadvantages of fragmentation far outweigh its benefits and fragmentation should be avoided as far as possible.

A TCP tunnel behaves like a level 2 (datalink) circuit and hides the different MTUs of the underlying networks and links it spans over the Internet. The user flows (in the form of IP packets which is the tunnel's payload) are oblivious to these different MTUs and hence are free to use their own MTU values, presumably the largest (limited by the sender's interface's MTU setting) to maximise throughput. If the use of larger MTUs is permitted, each packet carries less header overhead and this allows TCP connections to ramp up their congestion windows quicker.

To get an idea of the relationship between throughput and MTU sizes, we vary the Path MTUs as shown in Table 3 (1st column) and measure the resulting throughput of a TCP. Our first scenario is without TCP tunnels while the second is with TCP tunnels. In both scenarios, a round-trip delay of 211 ms was set between A and X (Fig. 6) which were the end machines used to create the TCP connection used in the two scenarios. The different MTUs were set on the path between T1 and T2 (Fig. 6) which has a round-trip delay of 10 ms. For the scenario with TCP tunnels, the tunnels were deployed between T1 and T2 to encompass the path with the different MTU. Ten runs were conducted per MTU setting. Note that the traffic source's MTU (a UDP flow with CBR of 9 Mbps) setting is fixed at 1500 bytes for all runs. The links' bandwidths were 10 Mbps throughout.

From Table 3, it can be seen that the tunnels improved the throughput of the end-to-end connection for lower MTU values across T1 to T2. This was because the tunnels confined the fragmentation caused by the lower MTU path to the tunnel only and did not allow the fragmentation to be propagated to the rest of the network.

Table 3
Mean connection throughput for various MTUs

Path MTU (T1–T2) (bytes)	Mean throughput of 10 runs (Mbps)	
	No TCP tunnel	With TCP tunnel
1500	7.40	7.40 (+0%)
1006	7.10	7.40 (+4%)
576	6.50	7.08 (+9%)
296	5.00	6.70 (+34%)

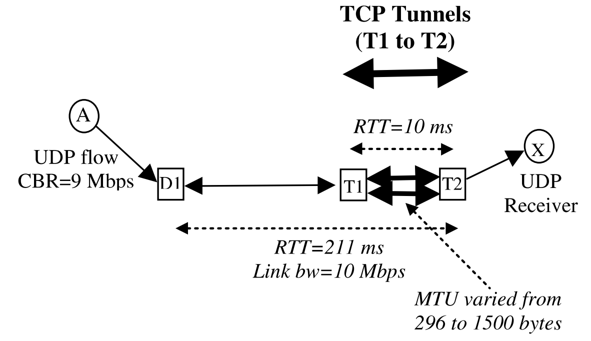

Fig. 6. Tunnel offering protection from varying MTU sizes over link T1 to T2.

Similarly, UDP flows would experience higher throughput by sending packets with large MTUs over links with smaller MTUs via tunnels. Finally, the use of Path MTU Discovery does not obviate the relevance of TCP tunnels. With Path MTU Discovery, the sender still has to send IP frames no larger than the smallest MTU along the path to avoid fragmentation. TCP tunnels allow senders to send packets at the largest possible MTU (limited by the MTU settings on their interfaces) despite the smaller MTUs along the path. The presence of these smaller MTU packets are totally transparent to the sender and the tunnels protect the sender from fragmentation caused by these smaller MTUs.

3. Effects of tunnels on router queue lengths

Tail-Drop has been adopted widely on routers as the default packet-dropping policy. However, this policy causes many TCP flows to throttle their sending rates at the same time, when the queue overflows. These flows would then ramp up their transmission rates until the queue overflows again. This synchronisation in flow behaviour causes burst effects and may reduce average flow throughput and link utilisation. In addition, the average queue length of Tail-Drop routers tends to be high and this introduces longer packet delays.

RED is designed to penalise TCP-friendly flows, which consume more than their fair share of bandwidth by randomly dropping packets from the flows when the average queue size exceeds the minimum threshold. RED eliminates the synchronisation and burst effects seen in Tail-Drop. RED, however, has no effect on UDP or unresponsive flows, which do not respond to packet drops. The flows that suffer from RED-induced packet drops, would be the good guys, the TCP-friendly flows.

To underscore our point on the queue lengths of routers adopting RED and Tail-Drop packet-dropping policies, we run experiments with an increasing number of TCP flows from a 10 Mbps source which traverse a bottleneck link of 1 Mbps. The runs are repeated for RED0.02, RED0.1 (RED with maximum drop probability, max_p, of 0.02 and 0.1, respectively) and Tail-Drop. The minimum and maximum thresholds for the RED queues, min_{th} and max_{th}, are 20 and 520 KB, respectively. The sizes of both RED and Tail-Drop queues are 520 KB.

Figs. 7 and 8 summarise the results. Note that the various queues' hard limits of 520 KB translate to a maximum queue occupancy of 354 packets of MTU 1500 bytes each. The confidence interval for Tail-Drop in Fig. 7 extends past this limit as the queues contain some small amount of spurious traffic due to Network Time Protocol (NTP) synchronisation between the machines on our testbed. Note that the confidence intervals for all queues given in Fig. 6 are 95%.

From Fig. 7, we can see that the mean queue length for Tail-Drop is high and the instantaneous queue lengths are highly variable. The RED policies manage to keep the average queue lengths low compared to Tail-Drop. The average queue length under RED scales with the number of TCP flows. RED0.1 demonstrates that increasing the maximum packet-drop probability has the effect of reducing the average queue length. Fig. 8 shows that the instantaneous queue length of RED0.02 is close to over-flowing the queue for 80 connections.

Although RED is able to keep the average queue length low for a small number of flows, its behaviour seems to approximate that of Tail-Drop when the number of flows that it has to deal with becomes large. When RED drops a packet, it affects 1 out of N connections only. It decreases the load factor by $1/2N$ only. RED simply cannot scale with N.

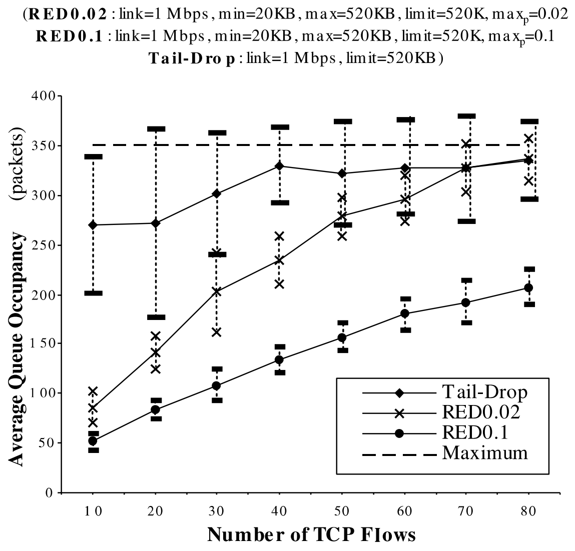

Fig. 7. Average queue occupancy for RED and Tail-Drop (no TCP tunnels).

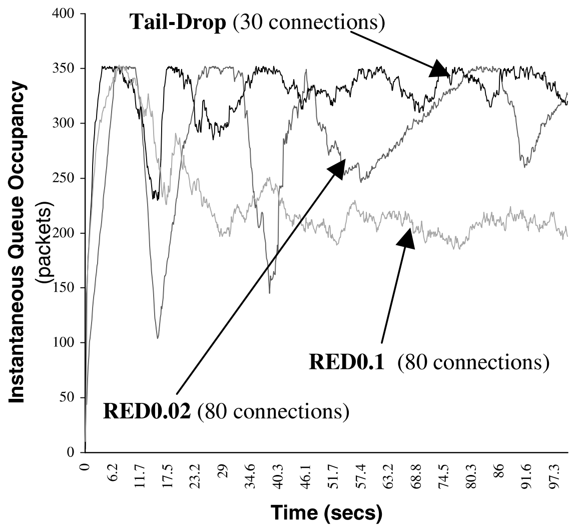

Fig. 8. Instantaneous queue occupancy for RED and Tail-Drop (no TCP tunnels).

The point we are emphasising is that the deployment of TCP tunnels reduces the number of flows within the core routers. The core routers only need to manage the few TCP tunnel flows. Therefore, as RED is being increasingly adopted by routers as the default packet-dropping policy (recommended by RFC2309) [15], we feel that TCP tunnels in conjunction with RED make an important contribution towards controlling congestion on the Internet.

4. Congestion control and back-pressure effects of TCP tunnels

In [5], the authors discuss the problems of unresponsive flows and the danger of congestion collapse of the Internet, while in [12], the authors discuss the use of back-pressure in moving congestion away from the core. The rates of UDP flows are not limited by congestion control and such aggressive flows only starve TCP-friendly flows of their rightful share of bandwidth. In addition, UDP flows typically exacerbate congestion problems and waste precious bandwidth when their packets are dropped during the onset of congestion.

TCP tunnels provide a possible solution out of this dilemma. Tunnels could be used to wrap UDP flows at edge routers. The tunnels carry the UDP flows over the Internet and are then subjected to the ever-changing conditions of the network. When congestion occurs, the packet drops force the tunnels to throttle their sending rates. The advantage here is that the UDP flows are now TCP-friendly (so to speak) and sensitive to packet drops from the perspective of the core routers. This is of particular relevance when RED is deployed on routers as the default packet-dropping policy. In short, packet losses in the congested core routers are now pushed to edge routers.

The main objection to this idea is that the tunnel would then introduce unnecessary delay and jitter variances to delay-sensitive UDP flows, on the order of several RTTs as the tunnel attempts retransmission (the tunnel being a TCP circuit offering reliable delivery of data). This will be discussed further in Section 8.

A huge advantage of carrying user flows over TCP tunnels and deploying tunnels in WAN environments is that congestion will be pushed to the edge routers from the core routers (similar to the back-pressure effect in ATM networks). The tunnels would seek out their fair share of bandwidth as network conditions change and packets are dropped from the tunnels. In response to the packet drops, the tunnels would throttle their sending rates and drop packets of user flows at the ingress tunnel gateways. The packet drops experienced by the tunnels within the backbone of the WAN network (at the core routers) would be low, since these losses are from the TCP tunnels themselves which would throttle their transmission rates in response. Contrast this with the situation where huge numbers of user flows enter the core without the deployment of tunnels. The link efficiency and utilization of the core is expected to be higher when tunnels are deployed. Notice that the deployment of tunnels is akin to the Virtual Paths of ATM networks.

To demonstrate the back-pressure effect, we run 50 bulk TCP flows on each of the machines A, B, and C to destinations X and Y (see Fig. 2). We also monitor the queue lengths at the core router (D2), and at the TCP tunnel router (T1). Each one of the three tunnels is fed by a separate Tail-Drop queue (maximum size of 1000 packets), and RED is adopted as the packet-drop mechanism on the core router, with the following settings: $\text{min}_{\text{th}} = 20$ KB, $\text{max}_{\text{th}} = 520$ KB, limit $= 520$ KB, $\text{max}_{\text{p}} = 0.02$.

From Fig. 9, we can see that the queue length of the core router is high. It reaches the limit of the RED queue. When TCP tunnels are deployed, the core router experiences a much lower queue length (peak queue length = 96). Fig. 10 displays the instantaneous queue lengths of the three Tail-Drop queues of the tunnel router along with that of the RED queue of the core router. Notice that the queue lengths at the tunnel router are much higher than the queue length at the core router, thus indicating that the congestion has been moved from the core to the edge.

Using the above experimental set-up, we run experiments to determine the percentage of dropped packets within the core router as the TCP load is steadily increased. From Table 4, it can be seen

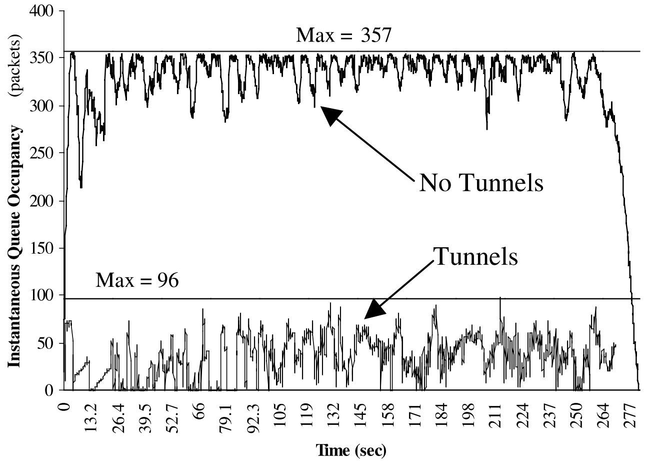

Fig. 9. Instantaneous queue occupancies for core router with and without TCP tunnels.

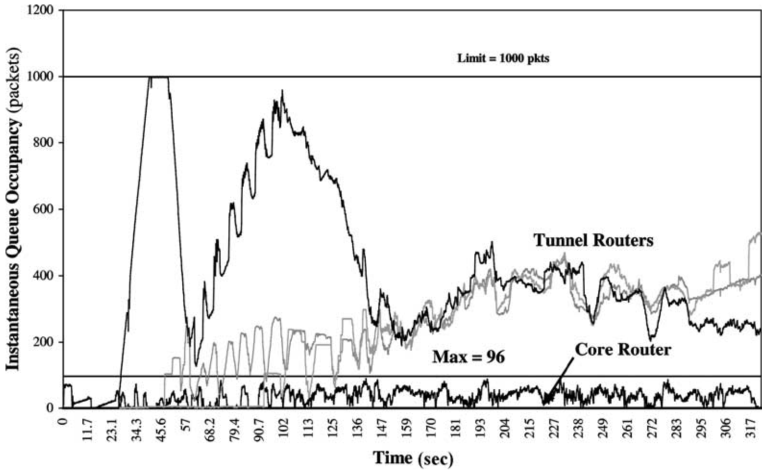

Fig. 10. Instantaneous queue occupancies for core and tunnel routers using TCP tunnels.

Table 4
Percentage of dropped packets in the core router

Number TCP flows in the core	Without tunnels (%)	With tunnels (%)
60	3.12	0.15
120	7.27	0.14
180	11.80	0.15
240	19.41	0.16

that tunnels significantly reduce and even stabilise the percentage of dropped packets in the core router even when the number of TCP flows increases.

Interestingly, the very low and stable values of packet drops experienced by user flows over TCP tunnels demonstrate traffic flow stability and the tunnels are sharing bandwidth fairly. We are still currently investigating this phenomenon.

5. Avoiding congestion collapse

In [6], the authors discuss at length the various forms of congestion collapse.

Classical congestion collapse [13] occurs when the TCP connections unnecessarily retransmit packets, which are still in transit or have been received, during heavy network load. The network reaches saturation and continues to operate under degraded conditions where the throughput is much lower than normal. This form of congestion collapse has been avoided by the adoption of congestion control principles put forward by Van Jacobson [16]. TCP tunnels can help to avoid this form of congestion collapse as the tunnels reduce the number of unnecessary retransmissions as established in Section 2.1.

Congestion collapse from undelivered packets results when precious bandwidth is consumed and wasted by packets, which are dropped before reaching their final destination. Open-loop applications such as voice, video and music applications probably will be the number one contributors to this type of congestion collapse as such applications do not utilise congestion control. The tunnel guarantees delivery of the permitted amount of open-loop user traffic to their destination and no user packets are lost within the tunnel. The use of TCP tunnels pushes the congestion from the core routers to the edge routers which causes packet drops to occur at the entry points, rather than further down the network.

Fragmentation-based congestion collapse arises when packets are dropped because their fragments cannot be reassembled into complete packets as some fragments are lost, possible due to congestion experienced enroute to the receivers. The continued congestion by such packets, which will probably be dropped by the receivers wastes bandwidth and results in reduced throughput. Early Packet Discard and Path MTU Discovery were highlighted in [6] as two mechanisms useful in combating this form of congestion collapse. We claim TCP tunnelling could be one more of such mechanisms as it confines the fragmentation to the tunnel alone and permits the use of the largest possible MTU for a TCP connection.

Congestion collapse from increased control traffic manifests when increasing load is accompanied by increased control traffic such as route updates, multicast join requests etc. TCP tunnels can prevent control traffic and their related user traffic from congesting the busy backbones by re-mapping the tunnels carrying those types of traffic to relatively uncongested paths in the network.

6. Providing quality of service guarantees

It was proposed by [10] that TCP Trunks could be used as a means of assuring QoS by exploiting the elastic property of the TCP circuit (tunnel). TCPs congestion and flow control would dynamically probe for available bandwidth when the load conditions of the network change. Kung and Wang [10] also suggested that the tunnel and its payload (user flows) could be allocated a guaranteed minimum bandwidth and be given more bandwidth when spare link capacity was available. Few other details were offered on how this was possible and it seems that deploying TCP tunnels is tricky. However, the deployment of TCP tunnels and its benefits would be more apparent if we attempt to qualify its applications to certain scenarios.

Using TCP tunnels to guarantee absolute QoS is not feasible without some form of route pinning. The reservation of resources necessary to support user services have to be instantiated at every hop of the network, even for tunnels, either manually or through some form of signalling (e.g. RSVP). This alone is a huge obstacle in deploying QoS-enabled tunnels on the Internet. However, the aggregation of user flows into tunnels offering different classes of service naturally complements any network QoS system (over WAN) as it contributes to scalability and eases the problem of flow identification and classification.

A more feasible approach in adapting TCP tunnels for QoS over the Internet would be to offer proportional differentiated services (proportional QoS) vis-a-vis absolute differentiated services (absolute QoS) [3,4]. Consider capacity differentiation as an example. A CBQ system could offer traffic class A 30% of the available bandwidth and traffic class B the remaining portion (70%) over the Internet. However, there is no minimum bandwidth guarantee that we can extract from the Internet without the cooperation of hops along the route and the bandwidth available changes from time to time. But, tunnels carrying both classes of traffic are always probing and utilising their fair share of the total bandwidth and we think it is not inconceivable that a CBQ system, applying some form of weighted fair queueing, would be able to deliver the flow of both traffic classes over the Internet in the promised proportions.

7. Deployment of TCP tunnels

The greatest mismatch of link bandwidths occurs at the edges of the Internet cloud. This mismatch is still true today as LANs now offer Gigabit speeds while WANs struggle to keep up, managing at most Megabit capacities. We find that TCP tunnels are most suitable for deployment at border routers. As traffic transit at such boundaries, TCP tunnels can be suitably positioned there to manage various classes of traffic and yet drop excess traffic closest to the sources, rather than allowing the unwanted traffic to be discarded in the core backbones and consuming precious bandwidth in the process.

The Internet has managed to meet the needs of its communities and users simply because its concept and deployment have been rigorously kept simple and scalable. Aggregation using tunnels reduces the problem of many flows to that of a few flows. This allows ISPs to identify and provide for classes of services in a scalable manner. The adoption of tunnels by ISPs would be invisible to user applications and these ISPs could provide incentives to encourage use of tunnels by favouring tunnels in their routers through the allocation of traffic priorities and judicious application of preferred scheduling and queueing policies.

Tunnels lend themselves easily to traffic management and provisioning through MPLS and RSVP. Tunnel flows can be uniquely identified by their IP addresses and ports, i.e. ⟨source IP, destination IP, source TCP port, destination TCP port⟩. However, it is critical to ensure that the deployment of tunnels do not alter the behaviour of their payload, especially those of TCP flows.

8. Summary and future research

We have demonstrated that TCP tunnels offer TCP-friendly flows protection against aggressive and unresponsive flows. In addition, we discover that TCP tunnels hide the different MTU values (typically smaller) of the underlying network links from the user flows and this property allows users flows to send traffic at their greatest MTU settings (dictated by their end network interfaces).

We also discover that the aggregation property of user flows in TCP tunnels results in much lower average queue lengths and dramatically reduces packet loss within the core routers. The reduction of packet loss concludes that less bandwidth is being wasted in the core backbone while the lower average queue lengths indicate that the core routers can be provisioned with less memory resources for handling tunnel flows. In other words, tunnels allow the routers to handle even more flows than otherwise possible, using the same amount of memory resources. By carrying greedy flows over TCP tunnels, the core routers are able to control the amount of traffic from unresponsive flows such as UDP (from multimedia applications) in the event of congestion within the network backbones. Therefore, precious limited bandwidth is not hogged by unresponsive flows, which will be wasted when their packets are dropped by the core routers when congestion occurs.

Aggregating flows into a TCP tunnel unfortunately introduces synchronisation and burst effects. If the network is in a steady state, the bandwidth and round trip times offered by the tunnel is also steady. However, changes in network conditions may cause wild swings in tunnel behaviour and transmit such effects to the user flows, which it transports. This will probably impact delay-sensitive traffic badly, especially if the tunnel attempts data recovery over a long latency link. In fact, in Section 2.1, we find that TCP tunnels degrade RTT estimations due to the tunnels' reliability since they are, after all, reliable circuits which recover from packet loss. It would also be interesting to examine the effect of the sizes of the queues, which feed the TCP tunnels, have on the RTTs of user TCP flows.

In order to avoid the debilitating effects of TCP tunnels on the RTTs of TCP flows, a tunnel with TCP-like properties, offering the flow and congestion control characteristics of TCP/IP and yet, without its (TCP's) reliable delivery of data, may be required. Such a protocol would not be unlike RTP complemented by flow and congestion control principles of TCP. Using such a tunnel for transporting IP packets would allow us to reap the benefits of aggregation and yet, eliminate the disadvantage of inaccurate RTT estimations by its payload of TCP user connections. More importantly, this new protocol would be of greater relevance as a transport protocol for end-to-end delay sensitive applications.

Ack compression occurs when the regular spacing of acks in the reverse direction is disrupted by other traffic interspersed among the acks. The most likely place where this happens is when acks are mixed with other traffic in the queue leading to a bottleneck link. Ack compression may reduce overall throughput and cause connections to become bursty. We believe that tunnels can help alleviate the problems of ack compression as they (tunnels) prevent unrelated traffic packets entering from intermediate hops from interjecting between the acks. The intermediate hops only carry tunnel traffic. The tunnels also help preserve the order of the ack packets as they enter and exit the tunnels.

We agree with [3,4] that it looks feasible and promising to assure QoS over the Internet in the form of proportional differentiated services (proportional QoS). We are exploring this concept by deploying TCP tunnels offering classes of user traffic different classes of QoS guarantees. We feel that TCP tunnels with its elastic bandwidth and congestion control properties could aid in delivering predictable proportional shares of QoS to end users on the Internet.

We have highlighted some further avenues of exploration and the need for further research, especially in determining the effects of tunnels on the behaviour of TCP flows and in examining the feasibility of deploying tunnels on actual WANs and on the Internet.

Acknowledgements

The authors would like to acknowledge the National Science and Technology Board (NSTB), Singapore and the Singapore Advanced Research and Education Network (SingAREN) project for supporting the above work under the Broadband21 project grant.

References

[1] W. Almesberger, Linux network traffic control—implementation overview, April 1999, available from ftp://lrcftp.epfl.ch/pub/people/almesber/pub/tcio-current.ps.

[2] P. Barford, M. Crovella, Generating representative Web workloads for network and server performance evaluation, Proc. ACM SIGMETRICS'98, June 1998.

[3] C. Dovrolis, P. Ramanathan, A case for relative differentiated services and the proportional differentiated model, IEEE Network 13 (5) (1999) 26–34.

[4] C. Dovrolis, D. Stiliadis, P. Ramanathan, Proportional differentiated services: delay differentiation and packet scheduling, Computer Communication Review 29 (4) (1999) 109–120.

[5] S. Floyd, K. Fall, Router mechanisms to support end-to-end congestion control, Technical Report, Network Research Group—Lawrence Berkeley National Laboratory, February 1997, available from ftp://ftp.ee.lbl.gov/papers/collapse.ps.

[6] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.

[7] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans Networking 1 (4) (1993) 397–413.

[8] M. Gates, A. Warshavsky, Iperf version 1.1, available from http://www.east.isi.edu/~tgibbons/iperf/index.html.

[9] C.A. Kent, J.C. Mogul, Fragmentation considered harmful, Research Report 87/3, Digital Western Research Laboratory, December 1987.

[10] H.T. Kung, S.Y. Wang, TCP trunking: design, implementation and performance, Proceedings of the 7th International Conference on Network Protocols (ICNP'99), October 1999.

[11] M. Muuss, T. Slattery, ttcp: Test TCP, US Army Ballistics Research Lab (BRL), November 1985, enhanced version by Silicon Graphics Incorporated, October 1991. Available from ftp://ftp.sgi.com/sgi/src/ttcp.

[12] C.M. Pazos, J.C. Sanchez Agrelo, M. Gerla, Using back-pressure to improve TCP performance with many flows, INFOCOM 1999, vol. 2, pp. 431–438.

[13] J. Nagle, Congestion control in IP/TCP internetworks, RFC896.

[14] J. Mogul, S. Deering, Path MTU discovery, RFC1191.

[15] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, Promoting the use of end-to-end congestion control in the Internet, RFC2309.

[16] M. Allman, V. Paxson, W. Stevens, TCP congestion control, RFC2581.

Lee Boon Peng completed his M.Sc. (Computer Science) at the School of Computing, National University of Singapore. He is currently an engineer with Unity Integration Corp., a total solutions provider in Transportation and Mobile Information Management Systems, while maintaining an active research interest in network protocol design and performance.

Rajesh Krishna Balan obtained his Bachelors in Computer Science (with honours) and Masters in Computer Science from the National University of Singapore in 1998 and 2001 respectively. Presently he is pursuing a Ph.D. in Computer Science at Carnegie Mellon University. He is currently working in the area of ubiquitious computing from the Operating System point of view.

Lillykutty Jacob obtained her M. Tech. Degree in Electrical Engineering (Communication Engineering) from the Indian Institute of Technology at Madras in 1985, and Ph.D. degree in Electrical Communication Engineering (computer networks) from the Indian Institute of Science, Bangalore, in 1993. She was a research fellow in the Department of Computer Science, Korea Advanced Institute of Science and Technology, S. Korea, during 1996–1997. Since 1985 she has been with the Regional Engineering College at Calicut, India. Currently, she is with the school of Computing, National University of Singapore, where she is a visiting academic fellow. Her research interests include Quality-of-Service and Resource Management n Internet, Network Protocols, and Performance Modelling and Analysis. She is a member of IEEE.

Winston Seah is the Programme Director of the Internet Technologies programme in CWC, he worked in the Department of Electrical Engineering, National University of Singapore (NUS), and the Ministry of Defence.

Dr Seah received the Dr. Eng. degree from Kyoto University, Kyoto, Japan, in 1997 and, the M. Eng. (Electrical Engineering) and B.Sc (Computer and Information Sciences) degrees from NUS in 1993 and 1987 respectively. For his postgraduate studies, he received the Monbusho Postgraduate Scholarship from the Government of Japan, as well as scholarships from the Foundation for C & C Promotion (NEC funded) and the International Communication Foundation (KDD funded) in Japan.

Dr Seah also holds joint teaching positions in the Department of Electrical and Computer Engineering, and Department of Computer Science in NUS, where he lectures in mobile computing and computer networks courses. He is actively involved in Research and Development in the areas of mobile/wireless Internet technologies, mobile ad hoc networks and Internet quality of services (QoS).

Akkihebbal L. Ananda and an Associate Professor in the Computer Science Department of the School of Computing at the National University of Singapore. He is also the Director of the Centre for Internet Research. He is actively associated with Singapore Advanced Research and Education Project (SingAREN) and has involved in network research and connectivity issues relating to Internet2. He is one of the key players in developing the NUS's campus secure plug-and-play network which has around 12,000 points campus wide. His research areas of interest include high-speed computer networks. He is a member of the IEEE Computer and Communications Societies.

Ananda obtained his B.E. degree in Electronics from the University of Bangalore, India, in 1971; his M. Tech degree in Electrical Engineering from the Indian Institute of Technology, Kanpur in 1974–1980 he worked as a system software engineer in India.