The performance of circuit switching in the Internet

Pablo Molinero-Fernández and Nick McKeown

Computer Systems Laboratory, Stanford University, Stanford, CA 94305-9030
{*molinero,nickm*}*@stanford.edu*

http://klamath.stanford.edu/TCPSwitching

This paper studies the performance of an Internet that uses circuit switching instead of, or in addition to, packet switching. On the face of it, this would seem a pointless exercise; the Internet is packet switched, and was deliberately built that way to enable the efficiencies afforded by statistical multiplexing, and the robustness of fast re-routing around failures. But link utilization is low particularly at the core of the Internet, which makes statistical multiplexing less important than it once was. Moreover, circuit switches today are capable of rapid reconfiguration around failures. There is also renewed interest in circuit switching because of the ease of building very high capacity optical circuit switches. While several proposals have suggested ways in which circuit switching may be introduced into the Internet, this paper is based on TCP Switching, in which a new circuit is created for each application flow. Here, we explore the performance of a network that uses TCP Switching, with particular emphasis on the response time experienced by users. We use simple M/GI/1 and M/GI/N queues to model application flows in both packet-switched and circuit-switched networks, as well as ns-2 simulations. We conclude that because of high bandwidth long-lived flows, it does not make sense to use circuit switching in shared access or local area networks. But our results suggest that in the core of the network, where high capacity is needed most, and where peak flow rate is limited by the access link, there is little or no difference in performance between circuit switching and packet switching. Given that circuit switches can be built to be much faster than packet switches, this suggests that a circuit-switched core warrants further investigation. © 2003 Optical Society of America

OCIS codes: 000.5490, 060.4250

1. Introduction

Electronic Internet routers are currently limited by the capacity they can provide with a limited space and power budget. Internet traffic growth poses a problem to carriers, too. While Internet traffic continues to double every year,[1,2] the capacity of electronic routers seems to follow Moore's law, doubling every 18 months for a given power consumption and volume. If it continues, this mismatch in growth rates between demand and capacity will become a huge problem for carriers. For example, at this rate, in ten years time we will need ten times more routers, implying a ten-fold increase in power consumption and space. Not only does this increase require ten times more central offices to house the routers, but it also increases the complexity of the network. This is a heavy economic and logistical burden for the transport providers, who are already deeply in debt. In these difficult economic times, these providers would prefer routers that are higher capacity, simpler, lower cost and lower power.

Recent advances in optical communications give an opportunity to overcome this gap between demand and capacity in the Internet, since optics do not have the bandwidth limitations of electronics. However, optical switching technology does not (yet) lend itself to *packet switching (PS)*. Internet routers are packet switches that, by their very nature, require large amounts of memory to buffer packets during periods of congestion, as well as per-packet processing. Unfortunately, there are currently no commercially viable techniques for building large optical packet buffers or packet processors; and it looks unlikely there will be any time soon.

On the other hand, *circuit switching (CS)* does not have these buffering and processing requirements in the data path, and thus it is a good candidate for building fast, simple, optical switches. Technologies such as MEMS,[3,4] integrated waveguides,[5] optical gratings,[6,7] tunable lasers,[8] and holography[9] have made possible very high capacity switch fabrics. At present, optical switches are circuit switches and so readily find a home in crossconnects, add-drop multiplexors, and patch-panels. Circuit switches are characterized by simple data paths requiring no per-packet processing, and no packet buffers.

Today, these circuit switches are mainly used in the core of the network, but the circuits that they carry are treated as point-to-point links by the *Internet Protocol, IP*. This paper studies the performance of an Internet where CS plays an active role in the core, adjusting automatically and dynamically the bandwidth between IP routers.

Two main questions need to be answered; the first is: How do we introduce CS at the core (not the edge) of the Internet in an evolutionary way? There is a variety of proposals that integrate both switching techniques: Generalized Multiprotocol Label Switching (GMPLS),[10] Optical Internetworking Forum (OIF),[11] Optical Domain Service Interconnect (ODSI),[12] OBS,[13] Zing,[14] and TCP Switching.[15] The other question is: How would the network perform as far as the end-user is concerned if there were circuits at the core?

In a previous paper, we concentrated on the first question[15] by proposing an approach called TCP Switching and analyzing its feasibility. In this paper, we concentrate on the second question. In particular, we look at the end-user experience with the extreme case in which each application flow at the edge of the network triggers a new circuit at the core (we call this TCP Switching). TCP Switching is based on the idea of IP Switching.[16] It exploits the fact that most data communications are connection-oriented, and, thus, existing connections can easily be mapped to circuits in the backbone. Despite its name, TCP Switching works with any application flow, and so it not only works with Transmission Control Protocol (TCP) connections, but also with less common User Datagram Protocol (UDP) flows, and Internet Control Message Protocol (ICMP) and Domain Name System (DNS) messages. We encourage the reader to read Ref.,[15,17] as they provide a more thorough background to the problem and some discussion of the salient advantages and disadvantages of circuit switching.

However, it is not the purpose of this paper to argue how good or bad TCP Switching is in terms of its implementation or ease of integration into the current Internet. Instead, we will explore how the Internet would perform if it included a significant amount of CS. In particular, our goal is to examine the obvious question (and preconception): Won't circuit switching lead to a much less efficient Internet because of the loss of statistical multiplexing? And, therefore, doesn't packet switching lead to lower costs for the operator and faster response times for the

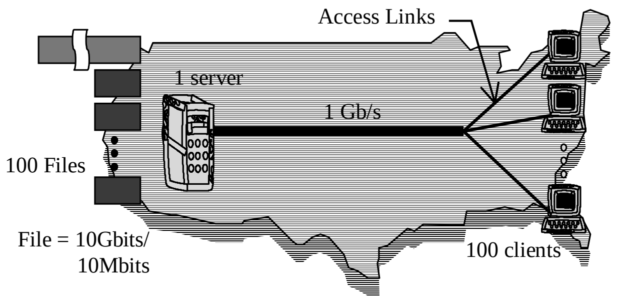

Fig. 1. Network scenario for both motivating examples. The bottleneck link is a transcontinental link of 1 Gbit/s. In the first scenario, all files are of the same length (10 Mbits). For the second scenario, the first file that gets requested is 1000 times longer (10 Gbits) than the rest.

users? While we are not necessarily arguing that TCP Switching is the best way to introduce CS into the core of the Internet, this extreme approach is something analyzable. Even though we do not explicitly consider CS of higher granularity in this paper, our results are not limited to TCP Switching, and they should give us an indication of how any CS technique will perform as far as the user is concerned and whether increased deployment of CS (in optical or electronic forms) is sensible.

The main performance metric that we will use is the *response time* of a flow, defined as the time from when a user requests a file from a remote server until the last byte of that file arrives. We choose this metric because the most common use of the Internet today is to download files, whether they are web pages, programs, images, songs or videos (Web browsing and file sharing represent over 65% of Internet transferred bytes today[18]). We will start by describing two different illustrative examples (albeit contrived), one in which CS outperforms PS, and one in which PS outperforms CS. We then use simple analytical models of CS and PS to determine the conditions under which one technique performs better than the other. We conclude that, while CS does not make much sense for the local area or access network, there would be little change in performance if we introduced CS into the core of the Internet. Finally, we use simulation of complete networks to support our findings.

We will also consider two other performance questions. First, how do CS and PS compare in terms of their ability to provide Quality of Service (QoS)? And second, how do CS and PS compare with respect to switching capacity? Our conclusion is that CS and PS both provide similar qualities of service to the end-user; however, circuit switches can be built with much higher capacity than packet switches. It is our opinion that this will inevitably mean more use of CS at the core of the Internet.

2. Response time

2.A. Motivating examples

We will use a simple example to demonstrate a scenario under which CS leads to improved user response time. Consider the network in Figure 1 with 100 clients in the East Coast of the US all trying simultaneously to download a 10-Mbit file from a remote server that is connected to a 1 Gbit/s link. Throughout this paper, we will define “1 Mbit” to be equal to 10^6 bits, not 2^{10} bits, in order to simplify our examples. With PS, all 100 clients will finish at the same time, after 1 sec. On the other hand, with CS and circuits of 1 Gbit/s, the average response time is 0.505 sec, half as much as for PS. Furthermore the worst client using CS performs as well as all the clients using PS, and all but one of the clients (99% in this case) are better

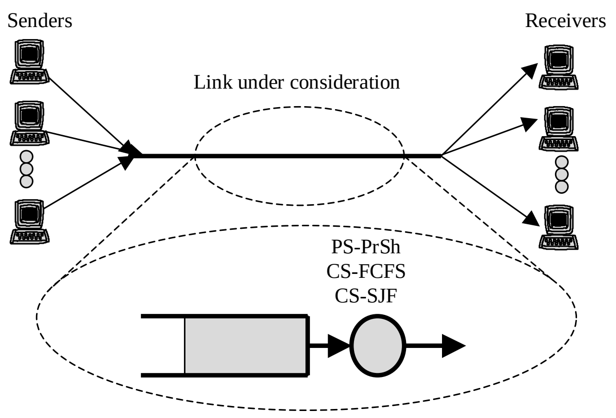

Fig. 2. Queueing model used to analyze a bottleneck link using CS and PS.

off with CS. We observe that in this case it is better to complete each job one at a time, rather than making slow and steady progress with all of them simultaneously. This result is reminiscent of the scheduling in operating systems where the *shortest remaining job first* policy leads to the fastest average job completion time.

Our second example demonstrates a scenario under which PS leads to improved response time. Consider the previous scenario, but assume that one client starts downloading a much longer file of 10 Gbits slightly before the others. With CS, this client hogs the link for 10 sec, preventing any of the other flows from making progress. And so the average response time for CS is 10.495 sec versus just 1.099 sec for PS. In this case all but one of the clients are better off with PS. Because active circuits cannot be preempted, the performance of CS falters as soon as a big flow monopolizes the link and prevents all others from being serviced. With PS, one long flow can slow down the link, but it will not block it.

Which scenario is more representative of the Internet today? We will argue that the second scenario (for which PS performs better) is similar to the edge of the network (i.e., LAN and access networks). However, we will argue that neither scenario represents the *core* of the Internet. This is because core links have much higher capacity than edge links, and so a single flow cannot block the shared link. We will need a different model to capture this effect. But first, let's consider a simple analytical model of how flows share links at the edge of the network.

2.B. Model for LANs and access networks

We start by trying to model the average response time for parts of the network where a single flow can fill the whole link. In what follows, we use a simple continuous time queueing model for PS and CS to try and capture the salient differences between them. We will use an M/GI/1 queue. The model assumes that traffic consists of a sequence of jobs, each representing the downloading of a file. Performance is assumed to be dominated by a single bottleneck link of capacity R, as shown in Figure 2. A service policy decides the order in which data is transferred over the bottleneck link. To model PS, we assume *Processor Sharing (PS-PrSh)*, and so all jobs share the bottleneck link equally, and each makes progress at rate R/k, where k is the number of active flows. To model CS, we assume that the server takes one job at a time and serves each job to completion, at rate R, before moving onto the next.

The CS model is non-preemptive, modeling the behavior of a circuit that occupies all the link bandwidth when it is created, and which cannot be preempted by another circuit until the flow finishes. To determine the order in which flows occupy the link, we consider two different service disciplines: *First Come First Serve (CS-FCFS)* and *Shortest Job First (CS-SJF)*. It is well known[19] that CS-SJF has the smallest average response time among all non-preemptive policies in an M/GI/1 system, and so CS-SJF represents the best-case performance for CS policies. However, CS-SJF requires knowledge of the amount of work required for each job. In this context, it means the router would need to know the duration of a flow before it starts, which is information not available in a TCP connection or other types of flows. Therefore, we consider CS-FCFS as a simpler and more practical service discipline, since it only requires a queue to remember the arrival order of the flows.

In our model, flows are assumed to be reactive and able to immediately adapt to the bandwidth that they are given. The model does not capture real-life effects such as packetization, packet drops, retransmissions, congestion control, etc., all of which will tend to increase the response time. This model can be seen as a benchmark that compares how the two switching techniques compare under idealized conditions. Later, the results will be corroborated in simulations of full networks.

The average response time, $E[T]$, as a function of the flow size, X, is:[19,20]

For M/GI/1/PS-PrSh:

$$E[T] = E[X] + E[W] = E[X] + \frac{\rho}{1-\rho}E[X] \tag{1}$$

for M/GI/1/CS-FCFS:

$$E[T] = E[X] + E[W] = E[X] + \frac{\rho}{1-\rho} \times \frac{E[X^2]}{2E[X]} \tag{2}$$

for M/GI/1/CS-SJF:

$$E[T] = E[X] + \frac{\rho E[X^2]}{2E[X]} \int_0^\infty \frac{f(x)dx}{(1-\frac{\rho}{E[X]}\int_0^{x^+} yf(y)dy)(1-\frac{\rho}{E[X]}\int_0^{x^-} yf(y)dy)} \tag{3}$$

where $0 \leq \rho < 1$ is the system load, W is the waiting time of a job in the queue, and $f(x)$ is the distribution of the flow sizes.

To study the effect of the flow size variance, we use a bimodal distribution for the flow sizes, X, such that $f(x) = \alpha\delta(x-A) + (1-\alpha)\delta(x-B)$, with $\alpha \in (0,1)$. A is held constant to 1500 bytes, and B is varied to keep the average job size, $E[X]$, (and thus the link load) constant.

Figure 3.a shows the average response time for CS (for both CS-SJF and CS-FCFS) with respect to that for PS-PrSh for different values of α and for link loads of 25% and 90%. A value of below 1 indicates a faster response time for CS, a value above 1 shows PS is faster.

The figure is best understood by revisiting the two motivating examples in section 2.A. When α is small, almost all flows are of size $B \approx E[X]$, and the flow size variance is small, $\sigma_X^2 \approx \alpha(E[X]-A)^2 \ll (E[X])^2$. As we saw in the first example, in this case the average waiting times for both CS-FCFS and CS-SJF are about 50% of those for PS-PrSh for high loads.

On the other hand, when α approaches 1, most flows are of size A, and only a few are of size $B = (E[X]-\alpha A)/(1-\alpha) \longrightarrow \infty$. Then $\sigma_X^2 \approx A^2 + (E[X]-A)^2/(1-\alpha) - (E[X])^2 \longrightarrow \infty$, and so the waiting time of CS also grows to ∞. This case is similar to our second example, where occasional very long flows[1] block short flows,

[1]These long flows are several orders of magnitude longer than the average flow length.

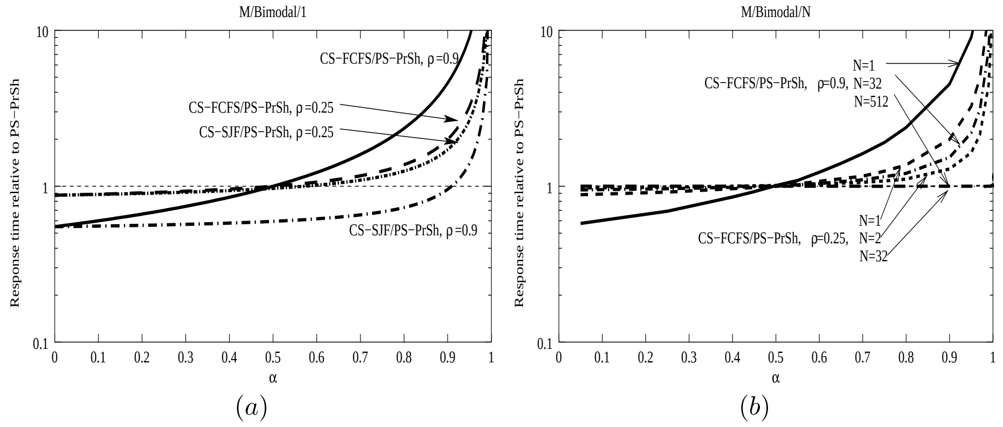

Fig. 3. Relative average response time of CS-FCFS and CS-SJF with respect to PS-PrSh for (a) a single server, and (b) an increasing core-to-access link-capacity ratio, N. Arrivals are Poisson and flow sizes are bimodal with parameter α. Link loads are $\rho = 0.25$ and $\rho = 0.9$. In (b) the value of N at which the curve flattens increases with the load, ρ, but it is smaller than 512, even for high loads.

leading to very high response time.

We can determine exactly when CS-FCFS outperforms PS-PrSh. The ratio between the expected waiting times of CS-FCFS and PS-PrSh is $E[X^2]/(2E[X]^2)$, and so as long as the coefficient of variation $C^2 = E[X^2]/E[X]^2 - 1$ is less than 1, CS-FCFS always behaves better than PS-PrSh. On the other hand, CS-SJF behaves better than CS-FCFS, as expected, and it is able to provide a faster response time than PS-PrSh for a wider range of flow size variances, especially for high loads when the job queue is often non-empty and the reordering of the queue makes a difference. However, CS-SJF cannot avoid the eventual blocking by long flows when the variance is high, and then it performs worse than PS-PrSh.

It has been reported[21] that the distribution of flow durations is heavy-tailed and that it has a very high variance, with an equivalent α greater than 0.999. This suggests that PS-PrSh is significantly better than either of the CS disciplines. We have further verified this conclusion using a Bounded-Pareto distribution for the flow size, such that $f(x) \propto x^{-\gamma-1}$. The continuous line in Figures 4(a) and 4(b) (for $N = 1$) show the results. It should not be surprising how bad CS fares with respect to PS given the high variance in flow sizes of the Pareto distribution.

We can conclude that in a LAN environment, where a single end host can fill up all the link bandwidth, PS leads to more than 500-fold lower expected response time that circuit switching because of link blocking.

2.C. Model for the core of the Internet

At the core of the Internet, flow rates are limited by the access link rate. For example, most core links operate at 2.5 Gbit/s (OC48c) today,[22] whereas most flows are constrained to 56 Kbit/s or below by the access link.[23] Even if we consider DSL, cable modems and Ethernet, when the network is empty a single user flow cannot fill the core link on its own. To reflect this, we need to adjust our model by capping the maximum rate that a flow can receive. We will use N to denote the ratio between the data rates of the core link and the access link. For example, when a flow from a 56 Kbit/s modem crosses a 2.5 Gbit/s core link, $N = 44,000$. Now,

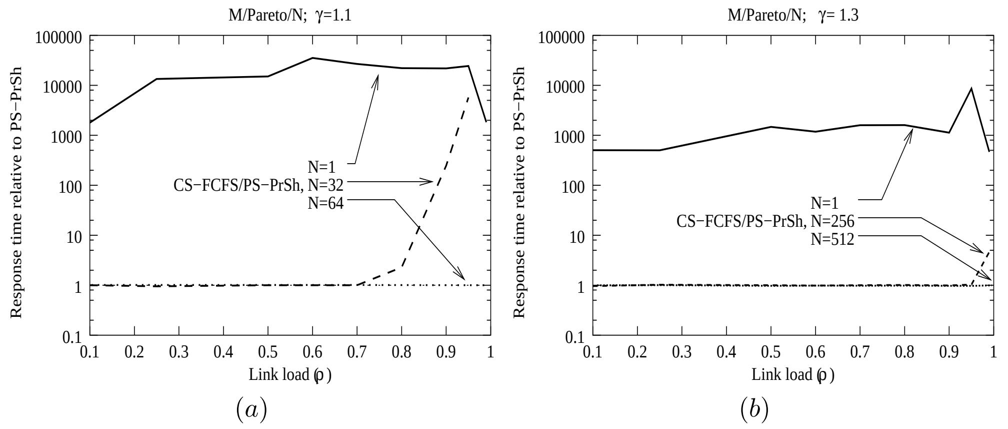

Fig. 4. Relative average response time for CS-FCFS with respect to PS-PrSh for an increasing core-to-access link-capacity ratio, N. Arrivals are Poisson and flow sizes follow a Pareto distribution with $\gamma = 1.1$ (a) and $\gamma = 1.3$ (b). The value of N at which the curves flatten is smaller than 512.

in our fluid model a single flow can use at most $1/N$ of the whole link capacity, so rather than having a full server, we will use N parallel servers, each with $1/N$ of the total capacity. In other words, we will use an M/GI/N model instead. For this model, there is an analytical solution[24] for the PS-PrSh discipline; however, there is no simple closed-form solution for CS-FCFS or CS-SJF, so we resort to simulation for these disciplines instead.

With CS, the more circuits that run in parallel, the less likely it is that enough long flows appear at the same time to hog all the circuits. It is also interesting to note that CS-SJF will not necessarily behave better than CS-FCFS all the time, as CS-SJF tends to delay all long jobs and then serve them in a batch when there are no other jobs left. This makes it more likely for blocking to take place, blocking all short jobs that arrive while the batch of long jobs is being served. On the other hand, CS-FCFS spreads the long jobs over time (unless they all arrived at the same time), and it is therefore less likely to cause blocking. For this reason and because of the difficulties implementing CS-SJF in a real network, we will no longer consider it in our M/GI/N model.

Figure 3.b compares the average response time for CS-FCFS against PS-PrSh for bimodal service times and different link loads. The ratio, N, between the core-link rate and the maximum flow rate is varied from 1 to 512. We observe that as the number of flows carried by the core link increases, the performance of CS-FCFS improves and approaches that of PS-PrSh. This is because for large N the probability that there are more than N simultaneous long flows is extremely small. The waiting time becomes negligible, since all jobs reside in the servers, and so CS and PS behave similarly. Figures 4.a and 4.b show similar results for bounded Pareto flow sizes as we vary the link load. Again, as N becomes greater or equal to 512, there is no difference between CS and PS in terms of average response time for any link load. We have also studied the variance of the response time for both flow size distributions, and we also saw little difference once N is greater or equal to 512.

To understand what happens when N increases, we can study Figure 5. As we can see, the number of flows in the system (shown in the upper three graphs) increases drastically whenever the number of long jobs (shown in the bottom three

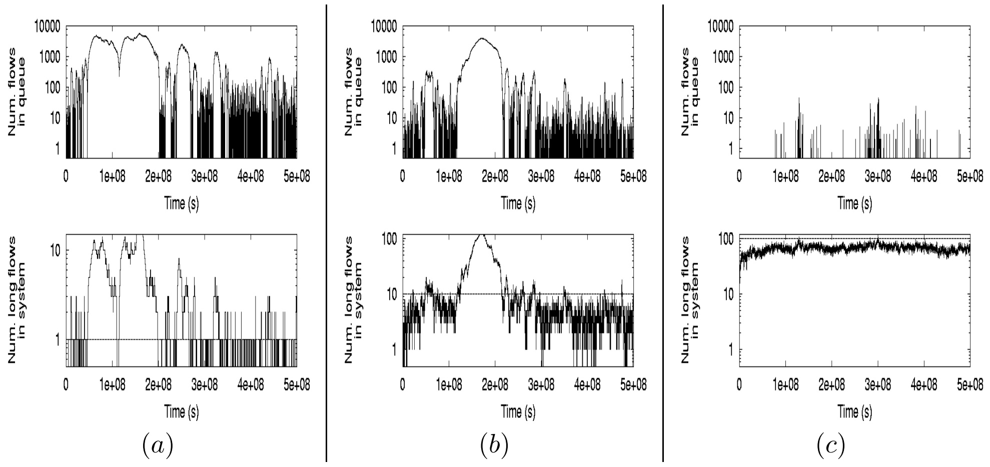

Fig. 5. The top three graphs show the total number of jobs in the queue of a M/Pareto/N/CS-FCFS for (a) $N = 1$, (b) $N = 10$, and (c) $N = 100$. The bottom graphs only show the number of long jobs in the system (both in the queue and in the N servers). Whenever there are more than N long jobs, the queue builds up. A long job is one that is 3 times the average job size.

graphs) is larger than the number of servers, which causes a long-lasting hogging. Until the hogging is cleared, there is an accumulation of (mainly) short jobs, which increases the response time. As the number of servers increases, the occurrence of hogging events is less frequent because the number of long flows in the system is smaller than the number of servers, N, almost all the time.

In the core, N will usually be very large. On the other hand, in metropolitan networks, N might be smaller than the critical N, at which CS and PS have the same response time. Then, in a MAN a small number of simultaneous, long-lived circuits might hog the link. This could be overcome by reserving some of the circuits for short flows, so that they are not held back by the long ones, but it requires some knowledge of the duration of a flow when it starts. One way of forfeiting this knowledge of the flow length could be to accept all flows, and only when they last longer than a certain threshold, they are classified as long flows. However, this approach has the disadvantage that long flows may be blocked in the middle of the connection.

In summary, the response time for CS and PS is similar for current network workloads at the core of the Internet, and, thus, CS remains a valid candidate for the backbone.

We reiterate that these are only theoretical results and they do not include important factors like the packetization of information, the contention along several hops, the delay in feedback loops, or the flow control algorithms of the transport protocol. The next section explores what happens when these factors are considered.

2.D. Simulation of a real network

To complete our study, we have used ns-2[25, 26] to simulate a computer network, where we have replaced the PS core with a CS core using TCP Switching. TCP Switching works as follows: When the first packet of a flow arrives to the edge of a circuit-switched cloud (see Figure 6), the boundary router establishes a dedicated circuit for the flow. All subsequent packets in the flow are injected into the same

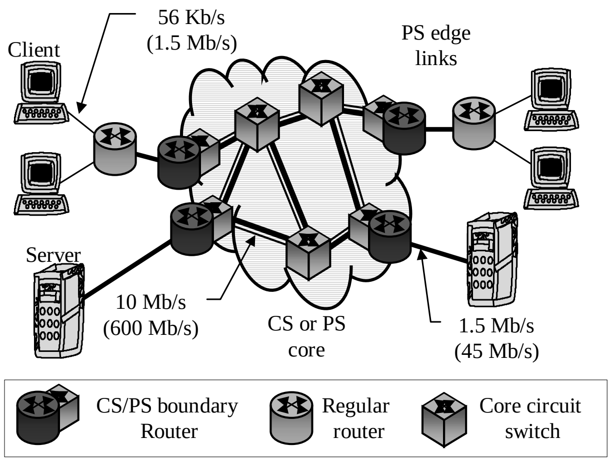

Fig. 6. Topology used in the ns-2 simulation. Two separate access link speeds of 56 Kbit/s (phone modem) and 1.5 Mbit/s (DSL, cable modem) were considered for the clients. The server-link and core-link capacities were scaled accordingly.

circuit to traverse the circuit-switched cloud. At the egress of the cloud, data is removed from the circuit, reassembled into packets and sent on its way over the packet-switched network. The boundary routers are regular Internet routers, with new linecards that can create and maintain circuits for each flow. The core switches within the cloud are regular circuit switches with their signaling software replaced by the Internet routing protocols. When a core switch sees data arriving on a previously idle circuit, it examines the first packet to determine its next hop, then it creates a circuit for it on the correct outgoing interface. In our simulations, we assume that the local area and access network is packet switched because, as we have already seen, there is little use in having circuits in the edge links.

In our setup, web clients are connected to the network using 56 Kbit/s links. Servers are connected using 1.5 Mbit/s links and the core operates at 10 Mbit/s. Later, we upgraded the access link to 1.5 Mbit/s, and we scaled the rest of the network proportionally. As one can see, flow rates are heavily capped by the access links of the end hosts, with a core-to-access ratio $N > 180$. Average link loads in the core links are less than 20%, which is consistent with previous observations in the Internet.[1, 27]

We assume that the circuits are established using fast, lightweight signaling,[15] which does not require confirmation from the egress boundary router, and thus it does not have to wait for a full round-trip time (RTT) to start injecting data into the circuit. We also assume that the crossconnect reconfiguration latency is very small. Of course, this may not be true for some crossconnect technologies, and then this latency should be added to the flow response time. However, unless the reconfiguration time is a significant fraction of the flow duration (typically a few seconds[15]), then its impact on the response time will be negligible.

Figures 7.a and 7.b show the goodput [2] and the response time, respectively, as a function of the file size. One can see the effects of TCP congestion control algorithms; the shortest flows have a very low goodput. This is mainly due to the slow-start mechanism that begins the connection at a low rate.

The key observation is that PS and CS behave very similarly, with CS having

[2]The goodput is the number bytes of user information that are transmitted divided by the flow duration (a.k.a. response time).

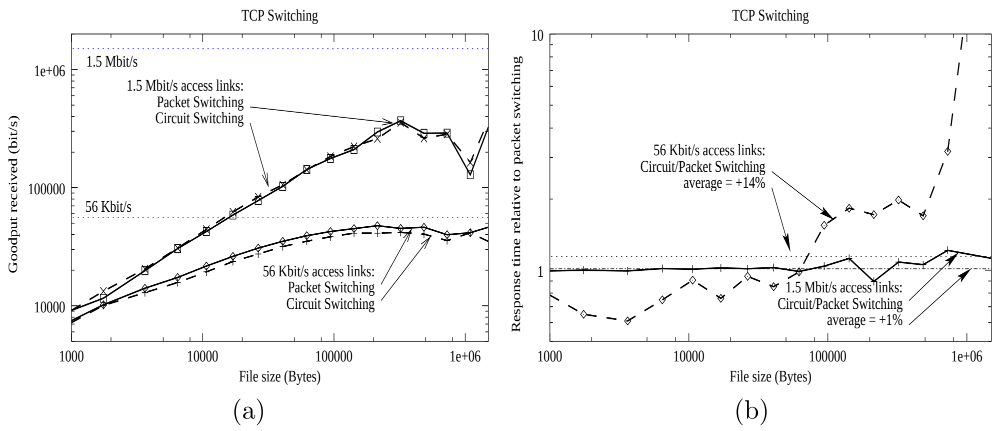

Fig. 7. Goodput (a) and relative response time (b) as a function of the size of the transferred file for access links of 56 Kbit/s and 1.5 Mbit/s. In subfigure (b) the average response time of TCP Switching over all file transfers is 14% greater than that of PS for 56 Kbit/s access links, and only 1% greater for 1.5 Mbit/s access links.

a slightly worse average response time (14% worse for 56 Kbit/s access links), but the difference becomes smaller, the faster the access link becomes (only 1% worse for 1.5 Mbit/s access links).

The reason for CS having worse goodput (and thus response time) is that the transmission time of packets along thin circuits in the core increases the RTT, and this reduces the TCP throughput.[28] For example, to transmit a 1500-byte packet over a 56 Kbit/s circuit takes 214 ms (vs. the 8 ms of a 1.5 Mbit/s link), which is comparable to the RTT on most paths in the Internet. PS does not have this problem because packets are transmitted at the rate of the core link (10 Mbit/s or 600 Mbit/s). In the future, as access links increase in capacity, this increase in the RTT will become less relevant, and CS and PS will deliver the same response time. We can conclude that, whereas the use of CS in LANs and access networks is undesirable for the end user, users will see little or no difference in terms of response time when using either CS or PS in the core.

3. Quality of Service (QoS)

Applications such as video streaming and audio conferencing do not care as much for response time as they do for minimum guaranteed bandwidth, bounded delay jitter and limited loss.

Because circuits are peak-allocated, CS provides simple (and somewhat degenerate) QoS, and thus there is no delay jitter. In PS there are two main ways to provide QoS guarantees: one way is to perform complex per-packet processing and to keep state either per-flow or at some higher aggregation. This is what Weighted Fair Queueing (WFQ),[29] Generalized Processor Sharing (GPS),[30] Differentiated Services (DiffServ),[31] Deficit Round Robin (DRR)[32] and other QoS mechanisms do. These schemes tend to be so complex that they significantly increase the complexity of the router (both hardware and software) and are hard for users to understand and configure. Another way for PS to support QoS is to heavily overprovision the link capacity and essentially emulate circuit switching.[33] While this may be practical, it is no more efficient in link utilization than circuit switching and still requires (unnecessary) per-packet processing and buffering.

Of course, the QoS of CS comes at the cost of wasted bandwidth, but as reported before,[1, 27, 34, 35] link bandwidth is no longer a scarce resource at the core of the Internet. In CS, packet-by-packet scheduling is replaced by a call admission controller that determines the order in which flows may proceed. Flows that are not yet scheduled are blocked while they wait their turn. In CS, we are, in a sense, replacing the queueing delays and packet drops with the blocking of flows by the admission control mechanism. However, as we have seen in the previous section, the waiting time until admission does not affect the final response time seen by the end user.

We have already considered an example of a call admission controller in previous sections: CS-FCFS. This is the CS equivalent of best-effort forwarding in a PS router. The admission decision is as simple as comparing the requested bandwidth (or a default bandwidth per flow) to the capacity available on the outgoing link. If no circuit is available right now, the flow is blocked until bandwidth is available.

As with PS, there will be proponents of introducing no QoS in the circuit-switched core beyond best effort, and relying on operators to over-provision the network. It might be argued that in practice, the network would be so overprovisioned (as it is already in the PS Internet), that blocking probabilities would be as negligible as they are in the telephone network today.

But like PS, there is a hierarchy of different QoS schemes that could be used in a CS network. These could be as simple as default priorities to different users, or different applications. Or they could be more complex schemes built on top of TCP Switching in a similar way to RSVP[36] and DiffServ[31] are built on top of PS. In fact, RSVP (or a simplified version) could be used directly with CS.

Finally, the QoS guarantees provided by a CS network are slightly different than for a PS network. In a CS network, constant bandwidth and zero delay variation come for free once a circuit has been established. Instead of specifying a bound on the losses, delay, jitter or minimum bandwidth, the user (or server) can inform the network of a flow's duration, and specify its desired rate and blocking probability (or a bound on the time that a flow can be blocked). We believe that these measures of service quality are simpler for users to understand and for operators to work with than those envisaged for PS networks.

4. Switching capacity

Up until this point, we have discussed the performance of a CS network primarily from the point of view of end users; however, they are not the only stakeholders in the Internet. One important metric for network operators is the capacity of individual switches in the core. They prefer to have the switching largest capacity within a switch volume and power consumption limit, because, for a given overall network capacity, fewer linecards are required to interconnect switches within the same central office, saving space, power and money, and reducing network complexity. This has become very important, as central offices have become more congested in recent years.

We argue that in terms of aggregate capacity, an electronic circuit switch can always outperform an electronic packet switch because its job is much simpler. The capacity is dictated by the main forwarding path through the system, which, for CS, is simpler and is not limited by memory technology.

In a packet switch, several operations need to be performed on every arriving packet; next-hop lookup, header update (TTL decrement and new checksum), packet buffering, switch fabric scheduling and output scheduling. Sometimes per-packet policing and filtering are also needed.

In contrast, once a circuit has been established CS requires no per-packet processing and no buffering, because arrivals and departures are planned to coincide with a pre-programmed cross-connection (plus or minus a small fixed delay at each switch). If the circuits did not change, circuit switches could be built to be approximately an order of magnitude faster than packet switches.

However, we also need to consider the cost of establishing and releasing circuits. This operation can be decomposed into two parts: the forwarding of a control message along the circuit's path (this is basically the same as forwarding a packet in PS), and creating/deleting the state associated with the circuit. The state management involves an admission controller that decides whether or not there is sufficient bandwidth on the outgoing link to support a new circuit and finding a time slot in which to transfer data from the input port to the output port. Both operations are relatively simple. Maintaining the state for each circuit means keeping track of which slots are currently assigned, and then timing out slots that are no longer used.

Finally, we can exploit the higher capacity of optical technology by building all-optical circuit switches, whereas we cannot currently build optical packet switches because of the buffering and per-packet processing needs.

In summary, CS can be decomposed into a fast path without per-packet processing and a slower path for establishing/releasing circuits, which is similar in complexity to forwarding a single packet in PS. However, the slow path needs to be taken much less often (for example, the average TCP connection lasts more than 10 packets, which means that connections are established at an average rate of at least 1/10 that of packet processing in a router). For these reasons we argue that circuit switches can operate much faster than packet switches.

To support this claim, consider the fastest widely available PS and CS systems today: 320 Gbit/s for a router,[37] 3.84 Tbit/s for a mostly electronic circuit switch,[38] and 2.56 Tbit/s for an all-optical circuit switch.[39] All figures are for bi-directional switching capacity for a single rack solution.

5. Conclusions

In this paper, we have argued that, for a given network capacity, users would experience little difference in performance if circuit switching were used instead of packet switching at the core of the Internet.

The main opportunity for circuit switches comes from their simplicity, and therefore they can be made faster, smaller and to consume less power. As a result, the capacity of the network can be increased by using more circuit switching in the core. TCP Switching provides one possible way to do this, without major disruption or flag days.

It might be argued that with this additional capacity, the network will naturally become overprovisioned, and so differentiated services and service guarantees become moot. However, experience suggests that even with overprovisioning there is always congestion somewhere in the network, and some applications would benefit from the deployment of QoS. It will be up to the network operators to determine whether or not the demand warrants the cost and complexity of deployment. But should they do so, we believe that QoS can be provided with circuit switching more simply than with packet switching.

Acknowledgements

We would like to thank Mor Harchol-Balter for her suggestions for the analysis of the CS-SJF discipline.

References and Links

1. K. Coffman, and A. Odlyzko, "Internet growth: Is there a 'Moore's Law' for data traffic?," in *Handbook of Massive Data Sets*, J. Abello, P. Pardalos, and M. Resende, eds., (Kluwer, 2001).
2. RHK, Telecommunication Industry Analysis, "United States Internet traffic experiences annual growth of 100%, but just 17% Revenue Growth ", Press release 157, (2002).
3. P. Hagelin, U. Krishnamoorthy, J. Heritage, and O. Solgaard, ""Scalable optical cross-connect switch using micromachined mirrors," IEEE Photonics Technology Letters, Vol. 12(7), pp. 882-885, (2000).
4. D. Bishop, C. Giles, and G. Austin, "The Lucent LambdaRouter: MEMS technology of the future here today," IEEE Communications Mag., Vol. 40(3), pp. 75-79, (2002).
5. M. Hoffmann, P. Kopka, and E. Voges, "Low-loss fiber-matched low-temperature PECVD waveguides with small-core dimensions for optical communication systems", IEEE Photonic Technology Letters, Vol. 9(9), pp. 1238-1240, (1997).
6. U. Krishnamoorthy, K. Li, K. Yu, D. Lee, J.P. Heritage, O. Solgaard, "Dual mode micromirrors for optical phased array applications," Sensors and Actuators: A. Physical, Vol. 97-98C, pp 22-26, (2002).
7. D. Burns, V Bright, S. Gustafson, and E. Watson, "Optical beam steering using surface micromachined gratings and optical phase arrays," in *Optical Scanning Systems: Design and Applications*, L. Beiser and S. F. Sagan, eds., Proc. SPIE 3131, pp. 99-110 (1997).
8. H. Yasaka, H. Sanjoh, H. Ishii, Y. Yoshikuni, and K. Oe, "Repeated wavelength conversion of 10 Gb/s signals and converted signal gating using wavelength-tunable semiconductor lasers," IEEE J. Lightwave Technology, vol. 14(6), pp. 1042-1047, (1996).
9. B. Pesach, G. Bartal, E. Refaeli, A. Agranat, J. Krupnik, and D. Sadot, "Free-space optical cross-connect switch by use of electroholography," Applied Optics 39(5), pp. 746-758, (2000).
10. A. Banerjee, J. Drake, J. Lang, B. Turner, D. Awduche, L. Berger, K. Kompella, and Y. Rekhter, "Generalized Multiprotocol Label Switching: An overview of signaling enhancements and recovery techniques," IEEE Communications Mag., vol. 39(1), pp. 144-150, (2001).
11. G. Bernstein, B. Rajagopalan, and D. Spears, "OIF UNI 1.0 -Controlling optical networks," White paper, Optical Internetworking Forum, (2001).
12. A. Copley, "Optical Domain Service Interconnect (ODSI): Defining mechanisms for enabling on-demand high-speed capacity from the optical domain," IEEE Communications Mag., vol. 38(10), pp. 168-174 (2000).
13. M. Yoo, C. Qiao, and S. Dixit, "Optical Burst Switching for service differentiation in the next-generation optical Internet," IEEE Communications Mag., vol. 39(2), pp.98-104, (2001).
14. M. Veeraraghavan, M. Karol, R. Karri, R. Grobler, and T. Moors, "Architectures and protocols that enable new applications on optical networks," IEEE Communications Mag., vol. 39(3), pp. 118-127, (2001).
15. P. Molinero-Fernández, and N. McKeown, "TCP Switching: Exposing circuits to IP," IEEE Micro Mag., vol. 22(1), pp. 82-89, (2002).
16. P. Newman, G. Minshall, and T. Lyon, "IP Switching: ATM under IP," IEEE/ACM Trans. on Networking, vol. 6(2), pp. 117-129 (1998).
17. P. Molinero-Fernández, N. McKeown, and H. Zhang, "Is IP going to take over the world (of communications)?," ACM HotNets-I Workshop, Princeton, NJ, (October 2002). Also to appear in ACM Computer Communications Review, January 2003.

18. CAIDA, Cooperative Association for Internet Data Analysis, "OC48 analysis summary: Distributions of traffic stratified by application," (2002), http://www.caida.org/analysis/workload/oc48/stats_20020109/ apps_index.xml.
19. L. Kleinrock, *Queueing Systems, Volume I: Theory* (Wiley-Interscience, New York, 1975).
20. R. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling* (Addison Wesley, Reading, MA, 1967).
21. M. Harchol-Balter, and A. Downey, "Exploiting process lifetime distributions for dynamic load balancing," ACM Trans. on Computer Systems, Vol. 15(3), pp. 253-285, (1997).
22. CAIDA, "Mapnet: Macroscopic Internet Visualization and Measurement," (2002) http://www.caida.org/tools/visualization/mapnet/.
23. A. Daum, "Broadband: The revolution's on hold in Europe just now," GartnerG2, Inc., (2001).
24. J. Cohen, "The multiple phase service network with generalized processor sharing," Acta Informatica 12, pp. 245-284, (1979).
25. The Network Simulator, ns-2, http://www.isi.edu/nsnam/ns/.
26. Ns-2 models used in the simulations, http://klamath.stanford.edu/TCPSwitching.
27. A. Odlyzko, "Data networks are mostly empty and for good reason," IT Professional 1 (no. 2), pp. 67-69, (1999).
28. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of ACM SIGCOMM*, (ACM, Vancouver, BC, 1998), pp. 303-314.
29. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair-queueing algorithm," in *Proceedings of ACM SIGCOMM 89*, (ACM, Austin, TX, 1989), pp 1-12.
30. A. Parekh, and R. Gallager, "A Generalized Processor Sharing approach to flow control in integrated services networks: The single-node case," IEEE/ACM Trans. on Networking, Vol. 1(3), pp. 344-357, (1993).
31. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC 2475: An architecture for Differentiated Services," (Internet Engineering Task Force, 1998).
32. M. Shreedar, and G. Varghese, "Efficient fair queuing using Deficit Round Robin," in *Proceedings of ACM SIGCOMM*, (ACM Cambridge, MA, 1995), pp. 231-242.
33. C. Fraleigh, "Provisioning IP backbone networks to support delay sensitive traffic," Ph.D. Thesis, Stanford University, (2002).
34. TeleGeography, Inc., "International bandwidth 2001, The TeleGeography guide to bandwidth supply and demand," TeleGeography, Inc., (2001).
35. M. Heinzl, "Operators of fiber-optic networks face capacity glut, falling prices," Wall Street Journal, New York, NY, (Oct. 19, 2000).
36. L. Zhang, R. Braden, S. Berson, S. Herzog, and S. Jamin, "RFC 2205: Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification," (Internet Engineering Task Force, 1997).
37. Juniper Networks, "T640 Internet routing node: datasheet," (2002).
38. Nortel Networks, "OPTera Connect HDX optical switch"," (2002).
39. Lucent Technologies, "LambdaXtreme Transport," (2002).