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

Keywords: packet switching; circuit switching, response time, QoS, switch capacity, network architecture.

Abstract -- **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, and falling, particularly at the core of the Internet, which means that statistical multiplexing is less important than it once was. And 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 reponse time experienced by users. We use simple M/G/1 and M/G/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 there are many flows per 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.**

I. Introduction

Innovations in optical technology have created renewed interest in *circuit switching (CS)* as a technique for building fast, simple, optical switches. Technologies such as MEMs [25][26][27], integrated waveguides [28], optical gratings [29], tunable lasers [30], and holography [31] have made possible very high capacity switch fabrics. Today, optical switches are circuit switches and so readily find a home in SONET crossconnects, add-drop multiplexors, and patch-panels. Circuit switches are characterized by simple data paths requiring no per-packet processing, and no packet buffers.

Optical switching technology does not (yet) lend itself to *packet switching (PS)*. Internet routers are packet switches which, by their very nature, require large amounts of memory to buffer packets during periods of congestion, as well as complex per-packet processing. Unfortunately, there are currently no commercially viable techniques for building large optical packet buffers or packet processors.

The potential for huge capacity circuit switches has lead to a variety of proposals that try to integrate circuit switching into the otherwise packet switched Internet. These include MPLambdaS [13], OIF [14] and ODSI [15], which address the issues of signaling, protection and restoration, but leave the control algorithms up to the network equipment vendors and network operators. Other proposals, such as Optical Burst Switching [17] and Zing [18], address the issue of control, but require a change in the way the Internet operates.

In a recent paper [1], we introduced an alternative approach, called TCP Switching, that allows the integration of circuit switched clouds with the rest of the Internet in an evolutionary fashion. Based on the idea of IP Switching [16], TCP Switching creates a new circuit for each application flow, exposing the control of circuits to IP. This architecture exploits the fact that most data communications are connection-oriented.

TCP Switching works as follows: When the first packet of a flow (usually a TCP SYN message, but not necessarily) arrives to the edge of a circuit-switched cloud (see Figure 1), the boundary router establishes a dedicated circuit for the flow. All subsequent packets in the flow are injected into the same 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

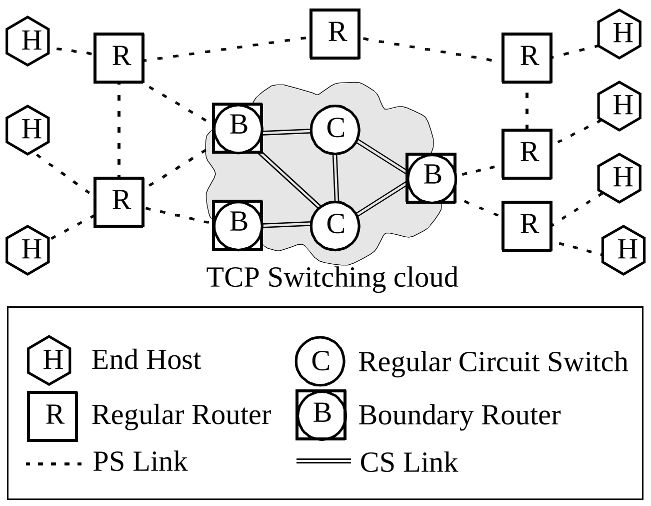

Figure 1: TCP Switching network architecture.

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 determines its next hop, then creates a circuit for it on the correct outgoing interface. Although TCP Switching is optimized for the most common case — TCP connections — it also works with less common UDP and ICMP flows.

There are, of course, many important questions to answer about the feasibility and wisdom of TCP Switching. For example, how much state does a boundary router need to maintain for all these circuits, and how complicated is it to manage the state? In our previous paper [1] we argued that the amount of state, and the complexity of maintaining it, is quite manageable in dedicated hardware.[1]

But 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 circuit switching. 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 users? To the extent that we can answer this question, our results are not limited to TCP Switching, and apply to any proposal that incorporates circuit switching into the Internet.

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 (over 63% of all packets/bytes are http/ftp [5][6]), whether they are web pages, programs, images or songs. 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 might outperform the other. We conclude that, while CS does not make much sense for the local area or access network, there would be little difference in the performance of CS and PS at the core of the Internet. We also use simulation to support our findings.

We will 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?

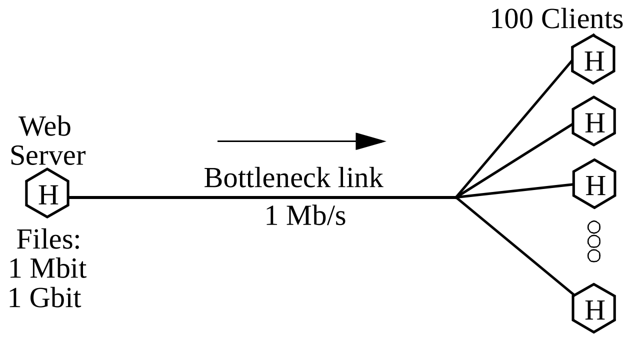

Figure 2: Network scenario for both motivating examples.

II. Motivating Examples

Our first example demonstrates a scenario under which CS leads to improved user response time. Consider the simple network in Figure 2 with 100 clients all simultaneously trying to download a 1Mbit[2] file from a remote server that is connected to a 1Mb/s link. With PS, all 100 clients will finish at the same time, after 100 sec. On the

[1]. Because of the many questions that arise from TCP Switching, and the controversial nature of the topic, we encourage the reader to read [1] before proceeding. It may not totally allay the reader's concerns, but it provides a more thorough background to the problem and some discussion of the salient advantages and disadvantages of circuit switching.

[2]. For the purposes of this paper, we will define "1Mbit" to be equal to 1,000,000 bits, not 2^{20} bits, in order to simplify our examples.

other hand, with CS and circuits of 1 Mbit/s, the average response time is 50.5 sec, half as much as for PS. Furthermore the worst client using CS performs as well as all clients when using PS, and all but one (99% in this case) of the clients are better 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 result used in operating systems that scheduling according to shortest remaining job first leads to the fastest average job completion time.

Our second example demonstrates a scenario under which PS leads to improved response time. Consider the same scenario, but assume that one client starts downloading a much longer file of 1,000Mbits slightly before the others. With CS this client hogs the link for 1000 sec, preventing any of the other flows from making progress. And so the average response time for CS is 1049.5 sec versus just 108.5 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 hogs the link and prevents all others from being serviced.

Which scenario is more representative of the Internet today? We will argue that the second scenario (for which PS performs better) is most likely in local area and access networks, whereas the wide area network requires a slightly different model.

III. Response Time

A. Model for LAN and access networks

We start by trying to model the response time for parts of the network where a single flow can utilize the whole link. For example, this is true in a LAN, or an access network. It does not hold for the core of the network, where links operate at 2.5Gb/s and higher.

In what follows, we use a simple continuous time queueing model for PS and CS to try and capture the salient differences between them. 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. A server, as shown in Figure 3, 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 jobs in progress. 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.

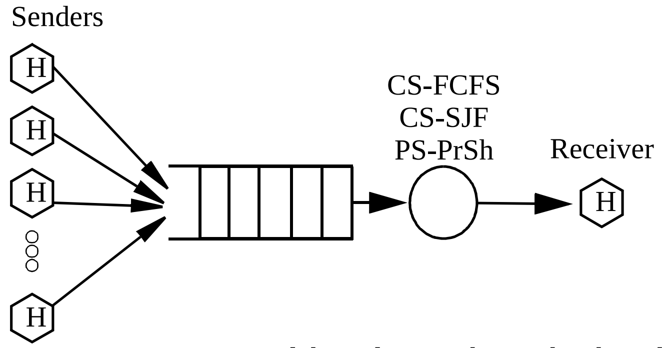

Figure 3: Queueing model used to analyze a bottleneck link using CS and PS.

The CS model is non-preemptive, modeling the behavior that when a circuit is created it occupies all the link bandwidth, and it 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 [2] that CS-SJF has the smallest response time among all non-preemptive policies in an M/G/1 system, and so CS-SJF represents the best case performance. However, CS-SJF requires knowledge of the duration of 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. Therefore, we consider CS-FCFS as a simpler and more practical service discipline.

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, slow-start mechanism, 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.

It can be easily shown [2][3] that the average response time, $E[T]$, as a function of the flow size, X is:

For M/G/1/PS-PrSh:

$$E[T] = E[X] + E[W] = E[X] + \frac{\rho}{1-\rho} \cdot E[X] , \qquad (1)$$

for M/G/1/CS-FCFS:

$$E[T] = E[X] + E[W] = E[X] + \frac{\rho}{1-\rho} \cdot \frac{E[X^2]}{2 \cdot E[X]} \text{, and} \qquad (2)$$

for M/G/1/CS-SJF:

$$E[T] = E[X] + E[W] =$$

$$E[X] + \frac{\rho \cdot E[X^2]}{2 \cdot E[X]} \cdot \int_0^{\infty} \frac{b(x) \cdot dx}{\left(1 - \frac{\rho}{E[X]} \cdot \int_0^{x} y \cdot b(y) \cdot dy\right)^2} , \qquad (3)$$

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

Figure 4 compares the response time for CS (for both CS-SJF and CS-FCFS) against the response time for PS-PrSh. In the graph, response times are shown relative to the response time for PS; i.e. the horizontal line at height 1 represents PS-PrSh. The flow sizes, X, are drawn from the bimodal distribution, $b(x) = \alpha \cdot A + (1 - \alpha) \cdot B$. A is held constant and B is varied to keep the average job size $E[X]$ constant.

The figure is best understood by revisiting the two motivating examples in Section II. When α is small, the flows are almost all of the same size and, as we saw in the first example, the response times for both CS-FCFS and CS-SJF are about 50% those for PS-PrSh for high loads

On the other hand when α approaches 1, the variance of the flow size tends to ∞, and we have a situation similar to our second example, where occasional long flows block short flows, leading to very high response time.

We can determine exactly when CS-FCFS outperforms PS-PrSh. The ratio of their expected waiting time is $E[X^2]/(2 \cdot E[X]^2)$, 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. Going back to Figure 4, we can see that as expected CS-SJF behaves better than CS-FCFS. For low loads and small or moderate flow size variance the waiting time is close to zero, the response time is dominated by the flow size, and it is almost independent of the switching discipline.

It has been reported [24] that the distribution of flow durations is heavy-tailed and that it has a very high variance, with an equivalent α greater than 0.9999. 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 with parameters similar to the ones found in [24].

We can conclude that in a LAN environment, where a single end host can fill up all the link bandwidth, PS leads to much lower expected response time that CS.

B. 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 2.5Gb/s (OC48c) today [7], whereas most flows are constrained to 56Kb/s or below [8]. Even if the network is empty, a single flow cannot completely fill the link. To reflect this we need to adjust our model by capping the maximum rate that a flow can receive. In other words, we can use an M/G/N model instead. Unfortunately, there are no simple closed-form solutions, so we resort to simulation 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 hogging to take place,

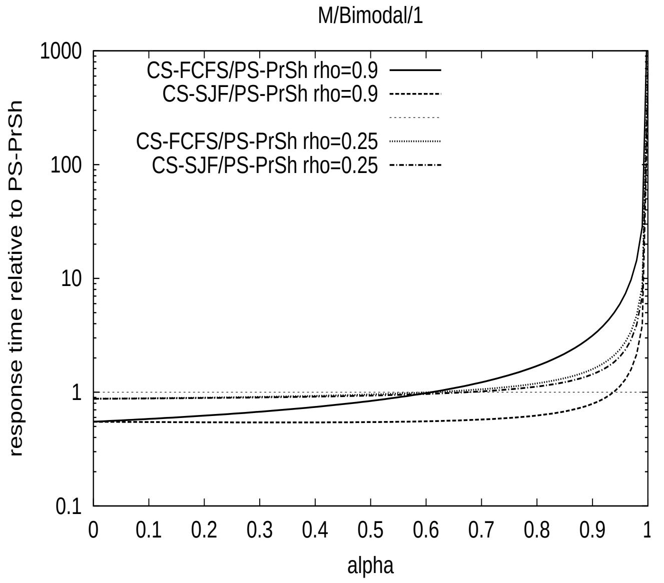

Figure 4: Relative response time of CS-FCFS and CS-SJF with respect to PS-ShPr for an M/G/1 system. The load is $\rho = 0.25$ *and* $\rho = 0.9$.

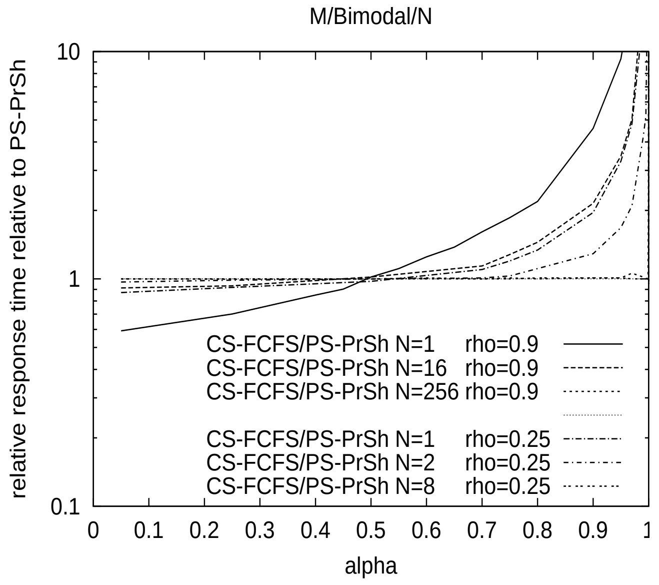

Figure 5: Relative response time for CS-FCFS with respect to PS-PrSh for an increasing core-to-access link-capacity ratio, N. The value of N at which the curve flattens increases with the load, ρ.

blocking new short jobs that arrive while the long jobs are in progress. On the other hand, CS-FCFS spreads the long jobs over time (unless they all arrived at the same time), and is therefore less likely to cause hogging. For this reason and because of the difficulties implementing CS-SJF in a real network, we will no longer consider it in our M/G/N model.

Figure 5 compares the response time for CS-FCFS against PS-PrSh for different flow-rates. The ratio, *N*, between the link rate and the maximum flow rate is varied from 1 to 256. We observe that as the number of flows carried by the 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* flows is extremely small. The waiting time becomes negligible, since all jobs reside in the servers, and so CS and PS behave similarly.

In the core, *N* will usually be large. For example a 2.5Gb/s (OC48) link can carry $N = 44,000$ circuits operating at 56kb/s. On the other hand in metropolitan networks, *N* will be much smaller, which means that 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 requires some knowledge of the duration of a flow when it starts.

In summary the response time for CS and PS is similar for current network workloads at the core of the Internet. On the other hand, CS does not seem well suited to local area and access networks, unless some smart scheduling is used to restrict circuits for use by long flows. As a reminder these are only theoretical results and they do not include important factors like the packetization of information, the delay in feedback loops, or flow control algorithms of the transport protocol. The next section explores what happens when these factors are considered.

C. Simulation of real network

To complete our study, we have used ns-2 [4] to simulate a network, where we have replaced the PS core with a CS core using TCP Switching. This is illustrated in Figure 6. As described in Section I, TCP Switching maps a user flows to its own circuit. 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 56Kb/s links. Servers are connected using 1.5Mb/s links and the core operates at 10Mb/s. As one can see, the flow rates at the core are heavily capped by the access links of the end hosts, with a ratio $N = 180$. Average link load in the core links is less than 20%, which is consistent with the observations in [9][10].

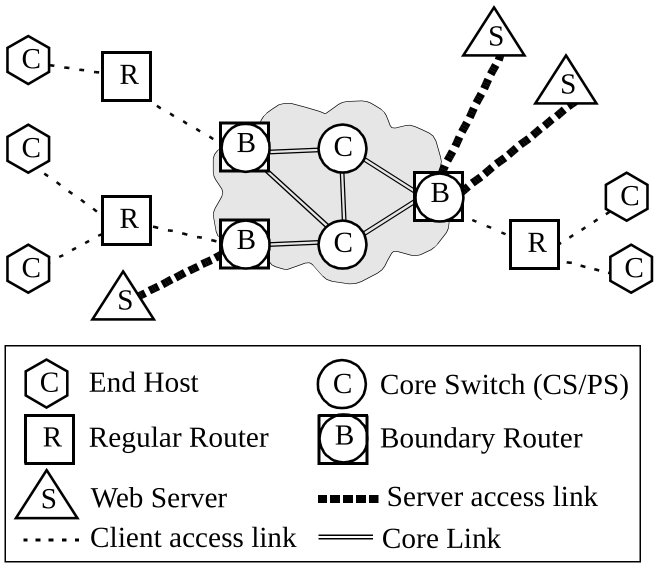

Figure 6: Topology used in the simulation

We assume that the circuits are established using fast, lightweight signaling, that doesn't require a full round-trip time (RTT). This is described and discussed in [1].

Figure 7 shows the average goodput and the average response time for 100 and 400 end hosts as a function of the file size. We can see the effects of TCP congestion control algorithms; first, 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 other observation is that PS and CS behave very similarly. If there are many flows, and the core has a long RTT, the performance of CS is slightly better than PS, otherwise CS behaves slightly worse than PS.

The reason for CS having worse goodput is that the transmission time of packets along thin circuits increases the RTT and this reduces the TCP throughput [32]. For example to transmit a 1500 byte packet over a 56 Kb/s circuit takes 214 ms (vs. the 8 ms of a 1.5 Mb/s link), which is quite comparable to the RTT on most paths in the Internet. In the future, as access links increase in capacity, this increase in the RTT will become less relevant. On the other hand if the number of flows is big, packets start making slower progress than the equivalent circuits because they have to divide the bandwidth among more.

IV. 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

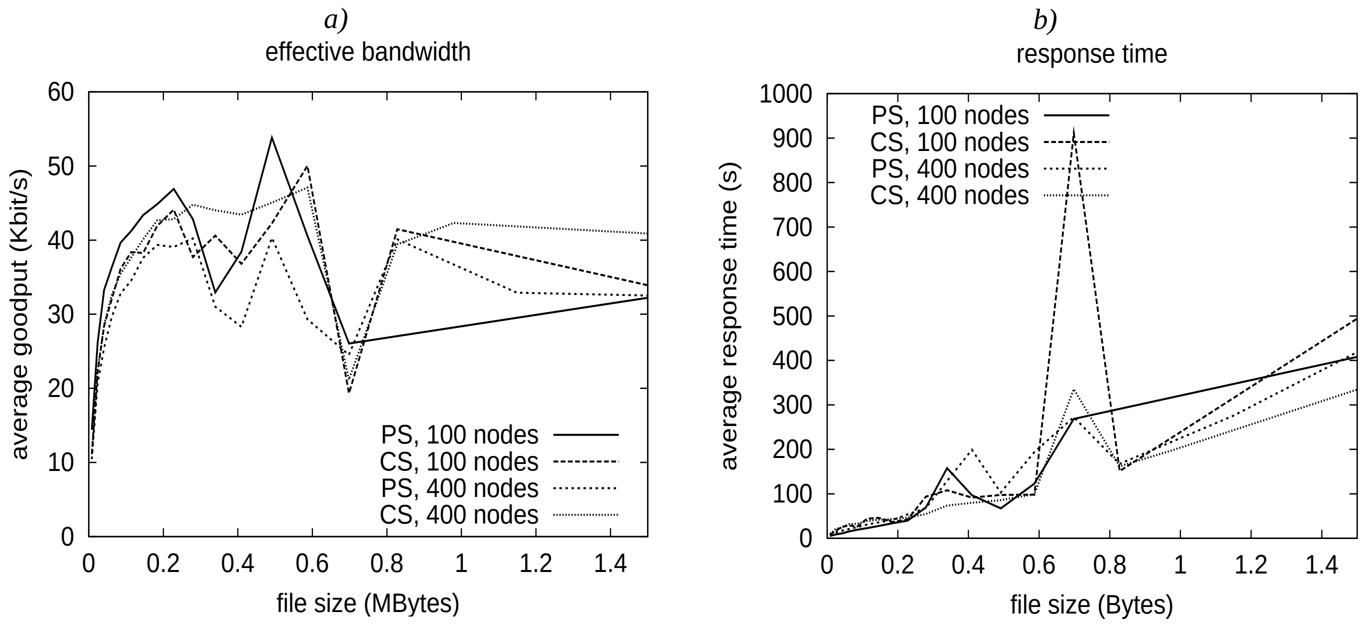

Figure 7: a) Average goodput and b) average response time with respect to the size of the transfered file for 100 and 400 nodes.

delay jitter. In PS there are two main ways to provide QoS guarantees; one way is to do complex per-packet processing and to keep and maintain state either per-flow or at some higher aggregation. This is what WFQ [19], GPS [20], DiffServ [21], DRR [22] and other packet scheduling disciplines 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. While this may be practical, it is no more efficient in link utilization than circuit switching, and still requires (unnecessary) per-packet processing.

Of course, the QoS of CS comes at the cost of wasted bandwidth, but as reported in [9][10][11][12] link bandwidth is no longer a scarce resource at the core of the Internet. QoS in a CS network is to some people simple, while to others inflexible. The (only) QoS decisions that a non-preemptive CS network can make, is the order in which the circuits are established, and the size of the circuit allocated to a flow. 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. We have already considered an example of a call admission controller in previous sections: CS-FCFS. This is the CS equivalents 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 beyond best-effort, and relying on operators to overprovision the network. It might be argued that in practice, the network would be so overprovisioned (as it is already is 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 that RSVP [36] and DiffServ [37] are built on top of PS. In fact, RSVP (or a simplified version) could be used directly.

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, the user (or server) can inform the network of a flow's duration, and specify: its rate, and its 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.

V. Switching Capacity

So far, we have studied 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 core switches in the core. They would like to have the largest capacity switch in the smallest volume, to save space in congested central offices. Operators prefer large capacity switches because, for a given overall network capacity, fewer linecards are required to interconnect switches within the same central office, saving space, power and money. As central offices have become more congested in recent years, the physical size of a switch has become important, as well as its power supply and cooling requirements.

We argue that in terms of aggregate capacity, an electronic CS can always outperform an electronic PS, 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.

On the other hand 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 in to 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/managing/deleting the state associated with the circuit. The state management involves an admission controller which 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. Creating and maintaining the state for each circuit means keeping track of which slots are currently assigned, and then timing out slots which 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: 160Gb/s for a router [33], 640Gb/s for a mostly electronic circuit switch [34], and 1.6Tb/s for an all-optical circuit switch [35].[3] All figures are for bi-directional switching capacity.

VI. Conclusions:

In this paper we have argued that, for a given network capacity, users would experience little difference in performance if CS were used instead of PS at the core of the Internet. Furthermore, (and this is the main opportunity) circuit switches are simpler and therefore can be made faster, smaller and consume less power. As a result, the capacity of the network can be increased by using more CS in the core. TCP Switching provides one possible way to do this, without major disruption or flag days.

It might be argued that with more capacity, the network will naturally become overprovisioned, and so differentiated services and service guarantees become moot. While the network is overprovisioned already, experience suggests that 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 CS more simply than with PS. On one hand, overprovisioning could be used to make blocking very rare; or admission controllers could make simple decisions based on the rate and duration of flows.

VII. Acknowledgements

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

[3]. While we have tried to be careful in our comparison, comparing product specifications from different vendors is not necessarily comparing "apples with apples", and should be taken only as a rough indication of their relative capacities.

VIII. References

[1] P. Molinero-Fernández, and N. McKeown. "TCP Switching: Exposing circuits to IP". Hot Interconnects IX, Stanford, CA, Aug 2001.

[2] L. Kleinrock. "Queueing Systems, Volume I: Theory". Wiley-Interscience, New York, 1975.

[3] R. W. Conway, W. L. Maxwell, L. W. Miller. "Theory of Scheduling", Addison Wesley, 1967.

[4] The Network Simulator, ns-2, http://www.isi.edu/nsnam/ns/

[5] NLANR network traffic packet header traces, http://moat.nlanr.ne t/Traces/

[6] Caida, Cooperative Association for Internet Data Analysis. "Characterizing Traffic Workload". http://www.caida.org/outreach/resources/learn/trafficworkload/

[7] Caida, Cooperative Association for Internet Data Analysis. "Mapnet: Macroscopic Internet Visualization and Measurement". http://www.caida.org/tools/visualization/mapnet/

[8] Jupiter Media Metrix. "HardScan Report". New York, NY, Mar. 2000.

[9] A. M. Odlyzko. "Data networks are mostly empty and for good reason". IT Professional 1 (no. 2), pp. 67-69, Mar/Apr. 1999.

[10] K. G. Coffman, and A. M. Odlyzko. "Internet growth: Is there a 'Moore's Law' for data traffic?". http://www.research.att.com/~amo/doc/networks.html

[11] C. Gutscher. "Optical Companies: Seeing clearly? Some fund managers view optical companies as too expensive". Wall Street Journal, New York, NY, Nov. 13, 2000.

[12] M. Heinzl. "Operators of Fiber-Optic Networks Face Capacity Glut, Falling Prices Wall Street Journal". Wall Street Journal, New York, NY, Oct. 19, 2000.

[13] D. Awduche, and Y. Rekhter. "Multiprotocol Lambda Switching: Combining MPLS Traffic Engineering Control with Optical Crossconnects". IEEE Communications Magazine, Mar 2001.

[14] G. Bernstein, B. Rajagopalan, and D Spears. "OIF UNI 1.0 -Controlling Optical Networks". White paper, Optical Internetworking Forum, Mar 2001.

[15] A. Copley. "Optical Domain Service Interconnect (ODSI): Defining Mechanisms for Enabling On-Demand High-Speed Capacity from the Optical Domain". IEEE Communications Magazine, Oct 2000.

[16] P. Newman, G. Minshall, and T. Lyon. "IP Switching: ATM Under IP". IEEE/ACM Transactions on Networking, vol. 6, no. 2, pp. 117-129, 1998.

[17] M. Yoo, C. Qiao, and S. Dixit. "Optical Burst Switching for Service Differentiation in the Next-Generation Optical Internet". IEEE Communications Magazine, Feb 2001.

[18] M. Veeraraghavan, M. Karol, R. Karri, R. Grobler, and T. Moors. "Architectures and Protocols that Enable New Applications on Optical Networks". IEEE Communications Magazine, Mar 2001.

[19] A. Demers, S. Keshav, and S. Shenker. "Analysis and Simulation of a Fair-queueing Algorithm", Procedings of ACM SIGCOMM 89, pp 1-12, Austin, TX, Sep 1989.

[20] A. K. Parekh and R. G. Gallager. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case". IEEE/ACM Transactions on Networking, Vol. 1 No. 3, pp. 344-357, Jun 1993.

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services". RFC 2475.

[22] M. Shreedar, G. Varghese. "Efficient Fair Queuing using Deficit Round Robin". Procedings of ACM SIGCOMM 95, Cambridge, MA, Aug/Sep 1995.

[23] M. Harchol-Balter and A. B. Downey. "Exploiting process lifetime distributions for dynamic load balancing". In Proceedings of the 1996 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 13-24, May 1996.

[24] M. E. Crovella and A. Bestavros. "Self-similarity in world wide web traffic evidence and possible causes". In Proceedings of the ACM SIGMETRICS 96, pp. 160-169, Philadelphia, PA, May 1996.

[25] P. M. Hagelin, U. Krishnamoorthy, C. M. Arft, J. P. Heritage, and O. Solgaard. "Micromachined Scalable Fiber-Optic Switch", Technical Digest of the 1999 Optical Society of America, Photonics in Switching Conference, pp. 4-6, Santa Barbara, CA, July 1999.

[26] D. T. Neilson, V. A. Aksyuk, S. Arney, N. R. Basavanhally, K. S. Bhalla, D. J. Bishop, B. A.

Boie, C. A. Bolle, J. V. Gates, A. M. Gottlieb, J. P. Hickey, N. A. Jackman, P. R. Kolodner, S. K. Korotky, B. Mikkelsen, F. Pardo, G. Raybon, R. Ruel, R. E. Scott, T. W. Van Blarcum, L. Zhang, and C. R. Giles. “Fully Provisioned 112x112 Micro-mechanical Optical Crossconnect with 35.8Tb/s Demonstrated Capacity”. Optical Fiber Conference 2000, PD12-1, Baltimore, MD, Mar. 2000.

[27] P. Heywood. Xros. ”Xros Launches First 1000-Port All Optical Cross Connect”. Light Reading, Mar. 2000.

[28] P. V. Lambeck, G. J. Veldhuis, T. Nauta, C. Gui, and J. W. Berenschot. “Electrostatically driven high-extinction mechano-optical on/off switch”. Proceedings of MOEMS Conference, Monterey, CA, July 98.

[29] D. M. Burns, V. M Bright., S. C. Gustafson, and E. A. Watson. “Optical Beam Steering Using Surface Micromachined Gratings and Optical Phase Arrays”. Proceedings of the SPIE, vol. 3131, pp. 99-110, 1997.

[30] 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 Journal of Lightwave Technology, vol. 14, pp.1042-1047, 1996.

[31] B. Pesach, G. Bartal, E. Refaeli, A. J. Agranat, J. Krupnik, and D. Sadot. “Free-space optical cross-connect switch by use of electroholography”. Applied Optics 39(5), pp. 746-758, Feb 2000.

[32] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. “Modeling TCP throughput: a simple model and its empirical validation”. ACM SIGCOMM 98, Vancouver, BC, Sep. 1998.

[33] Cisco Systems. “Cisco 12416 Internet Router: Data Sheet”. http://www.cisco.com/warp/public/cc/pd/rt/12000/12416/prodlit/itro_ds.htm

[34] Ciena. “CIENA MultiWave CoreDirector“. http://www.ciena.com/downloads/products/coredirector.pdf

[35] Lucent Technologies. “WaveStar OLS 1.6T“. http://www.lucent.com/livelink/152114_Brochure.pdf

[36] L. Zhang, R. Braden, S. Berson, S. Herzog, andS. Jamin. “Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification”. RFC2205, IETF, Sep. 1997.

[37] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. “An Architecture for Differentiated Services”. RFC2475, IETF, Dec. 1998.