TCP Switching: Exposing Circuits to IP

Pablo Molinero-Fernández, Nick McKeown
Stanford University
{molinero,nickm}@stanford.edu

Abstract

There has been much discussion about the best way to combine the benefits of new optical circuit switching technology with the established packet switched Internet. In this paper, we explore how electronic and/or optical circuit switching might be introduced in an evolutionary manner.

Circuit switches have much simpler data paths, and being potentially must faster than packet switches. As a result, very high capacity all-optical circuit switches are feasible today (e.g. WDM, MEMs, and wavelength conversion systems), whereas all-optical packet switches are a long way from being commercially practical, because we still do not know how to buffer photons. Of course, the main disadvantage of circuit switching is that link capacity must be peak-allocated, eliminating the benefits of statistical multiplexing, leading to the inefficient use of links. It is a premise of this paper that link capacity is abundant and will become more so with time. The bottleneck in the Internet today is most often in the routers, not the links.

There are other perceived disadvantages of circuit switching, which we consider. These include: (1) the amount of state maintained by switches, and its influence on network-wide fault tolerance, (2) the overhead of signaling, and (3) the difference in paradigm from the already successful packet switching used in the Internet. We propose an approach - which we call TCP Switching - that can co-exist with the existing Internet in an evolutionary way. TCP Switching exposes circuits to IP, establishing a new circuit for each application-level flow in the network. TCP Switching takes its name from, and strongly resembles, IP Switching in which a new ATM virtual circuit is established for each application-level flow.

We realize that the idea of TCP Switching is controversial and, no doubt, incomplete. It is the intention of this paper to contribute to the discussion on how circuit switching and optical switching technology can be exploited in the Internet.

1. Introduction: The Internet as a packet switched network

We tend to take for granted that the Internet is a packet switched network comprised of just routers, links and end hosts. In reality there is a lot of circuit switching in the Internet, both at the core (SONET/SDH) and in the last mile (modems, DSL). However, these circuits are treated by IP as static point-to-point links between adjacent nodes; the physical circuits and IP are completely decoupled. In this paper we will explore how to integrate *packet switching* (PS) and *circuit switching* (CS) more tightly in the same network.

One question worth asking is why the Internet is packet switched in the first place. Textbooks will tell us that the original reasons for PS (as opposed to CS) are twofold: The first one being *bandwidth efficiency*; PS requires less link capacity to carry the same amount of traffic. In the 70s and 80s links were leased lines and they were an expensive and scarce resource. The second reason is *robustness*. Because the only state needed in the routers are the routing tables, re-routing around a link/node failure is as easy as updating these tables.

However, neither of these reasons appears to hold today. In terms of bandwidth efficiency, it is reported that most links today are only lightly utilized [5][6]. Furthermore, it is anticipated that utilization will decrease over time due, in part, to the huge investment in optical fibers that has created a glut of capacity in the core [8][9]. As for robustness, PS routing protocols do not necessarily lead to simple and rapid reconfiguration. Routing protocols today are extremely complicated, and can take seconds or even minutes to re-route around failures [17]. On the other hand, most CS equipment (e.g. SONET) is required to recover in less than 50 ms [18]. So having per-circuit state in the network does not prevent a network from being robust.

2. Is circuit switching appropriate in the Internet?

Although the original reasons for using packet switching may no longer hold, this is not an argument for circuit switching. We need to consider carefully both the advantages and disadvantages of using circuit switching at the core of the Internet, where the high link data rates will expose the limitations of any switching technique.

Circuit switching seems to have the following advantages:

- CS requires no buffers in the switches. A packet switch must, by definition, maintain packet buffers for periods of congestion. The need for buffering prevents us from building feasible all-optical routers. Even in an electronic core router these buffers must be both large and fast. A router usually maintains about (*R* x *RTT*) bits of packet buffers, where R is the line rate, and *RTT* the round-trip time between any two end-hosts (about 0.25s today). For example, a 10Gb/s line card maintains about 2.5Gbits (300Mbytes) of storage. In terms of speed, the buffers must run at least as fast as the line. Today, it is hard to design a buffer operating at 10Gb/s, and harder still to design one at 40Gb/s.
- CS simplifies (essentially eliminates) per-packet processing. A packet switched router must process each packet as it arrives, performing an address lookup and modifying the packet header. The processing might be performed by a network processor, or a dedicated ASIC. While the processor can be expected to double in speed every 18 months, the link capacity in DWDM has been doubling every 7 months. It is likely that at some point network processors, and later ASICs, will run out of steam, at which point CS may be the only alternative.

This leads us to state that for a given technology circuit switches are faster (i.e. have more capacity) than packet switches.

Other advantages of CS are:

- CS lends itself to simple, intuitive (and degenerate) QoS, because the bandwidth is peak allocated, and the delay jitter is zero. In PS there are two ways of providing QoS guarantees; one way is to do complex per-packet processing and to maintain state either per-flow or at some higher aggregation, for example using WFQ [19], GPS [20], DiffServ [21], DRR [22] and other packet scheduling disciplines. These schemes are complex, significantly increasing the complexity of the router (both hardware and software) and are hard for customers to understand, configure and use. 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 per-packet processing.
- CS replaces per-packet scheduling or drop decisions with simple per-flow admission decisions. The admission control decisions are simple: is there or is there not sufficient bandwidth on the outgoing link for the requested flow? And because flows consist of multiple packets, the decisions are made less often than for PS.
- CS can, in many situations, actually reduce the *response time*[†] seen by users. When either the variation of the flow durations is small or when the peak flow rate is much smaller than the link bandwidth (e.g. in the core of the network), then the response time for CS is smaller than for PS.

On the other hand, circuit switching has some disadvantages:

- CS requires circuits to be established. This circuit could carry multiplexed traffic between two routers (e.g. SONET) or a single application flow (which is what we will consider here). Either way, the circuit needs to be established, state needs to be kept, and then the circuit needs to be freed upon completion. If the number of circuits is large, a circuit switch may need to maintain a lot of state and the amount of signaling can be significant. We will argue later that, in practice, the number of flows (and the rate at which they are added and removed) will be quite manageable in simple hardware, even for a high capacity switch.
- CS requires that circuits are multiples of a common minimum circuit size. For example, it is common for SONET cross-connects today to provision circuits in multiples of STS-1 (51Mb/s). If we desire a circuit that is not an exact multiple, then link capacity will be wasted. This problem can be minimized by using a small circuit granularity (for example, 64kb/s), but this requires the switch to maintain more state.
- CS leads to some probability of blocking while waiting for a circuit to be available. If a circuit is not available when needed, we must wait, or be blocked, until a circuit is free. This is very different from the link-sharing paradigm present in the Internet today in which packets will still make (albeit slow) progress over a congested link.
- The Internet is not prepared to handle the dynamic raw circuits of CS. We will need an evolutionary solution to be able to use CS.

3. Typical flows in the Internet today

In what follows, we will explore how circuits could be established in the core of the Internet for each flow. To understand whether this is feasible, or sensible, we will need a good understanding of the Internet today. Because over 90% of the traffic is TCP (both in terms of packets and bytes), we need to know what a TCP flow is, and how it behaves. We have studied traceroute measurements, as well as packet traces from OC-3 and OC-12 links in vBNS

[†] Response time is defined here to be the time from when a request is sent until the response completes the download.

Table 1.- Characteristics of TCP flows in the current Internet

	80%-percentile	average	median
TCP flow duration	< 4-10 sec	< 3-7 sec	< 0.5-1.2 sec
Packets per flow	< 12 pkts	< 10-200 pkts†	< 5-9 pkts
Bytes per flow	< 2.5-4 KBytes	< 9-90 KBytes	< 0.6-1.3 KBytes
Bandwidth (=bytes/duration)	< 50-100 Kbps	< 20-140 Kbps†	<8-15 Kbps
% Flow with retransmissions	< 7.8%	< 5.6%	< 4.7%
% Flows experiencing reroute	< 0.19%	< 0.39%†	< 0.02%
Asymmetrical connections	Around 40% of the flows transmit an ACK after the FIN, i.e. they are acknowledging a data packet that was sent in the other direction.		

.† This is because the tail of the distribution is very long and it has a lot of mass

[1] (available at NLANR [2][3]). Table 1 describes the typical flow in the Internet.

In summary, TCP connections typically last less than 10s, carry less than 4 Kbytes of data and consist of fewer than 12 packets in each direction. Few TCP flows see any losses (<6%) and even fewer see a route change (<0.4%[1]).

4. TCP Switching: a cloud of CS inside a PS network

Up until now, we have considered both PS and CS. It is not our goal to propose, or even consider, the complete replacement of PS by CS. On the contrary, our goal is to find a way to use the benefits of CS in the core of the Internet (where it is, in fact, already present), and allow the circuit switches to be controlled by IP. In the rest of this paper we propose and explore a network architecture called *TCP Switching*.

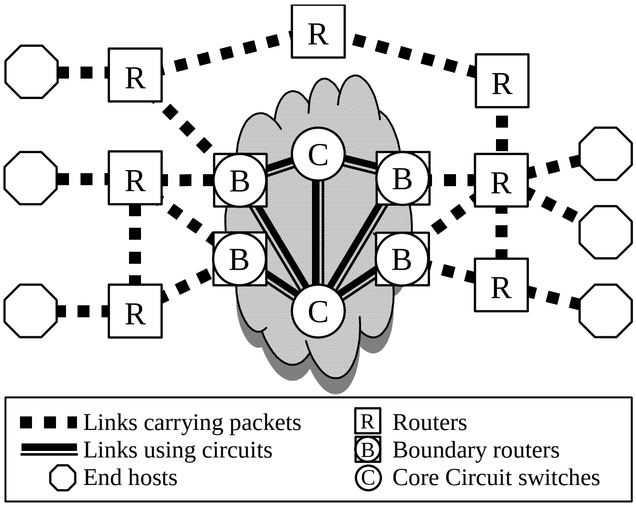

Figure 1.- TCP Switching as a cloud of circuit switching inside a packet switched network

TCP Switching consists of establishing fast, lightweight circuits triggered by application-level flows in the Internet. We call it TCP Switching because most flows today (over 90%) are TCP, and so TCP Switching is optimized for the common case of TCP connections; but the technique is not limited to TCP, and any uni- or bi-directional flow can be accommodated, albeit less efficiently. TCP Switching can be deployed in an evolutionary way, by creating self-contained TCP Switching clouds inside a PS network, as Figure 1 shows.

The PS portion of the network is unchanged. The core of the CS portion of the network is built from pure circuit switches (such as SONET cross connects) with simplified signaling to setup and teardown circuits. Boundary routers which act as gateways between the PS and CS world, mapping data between packet switched flows and circuits. They are most likely conventional routers with CS line cards.

TCP Switching requires no additional signaling, as the first observed packet in a flow triggers the creation of a new circuit. For TCP connections, the first packet is most often a SYN or a SYN+ACK packet.[2] Circuits are removed by a TCP FIN packet (the most common case), or by an inactivity timeout. Because in most cases TCP Switching re-uses the 3-way TCP handshake to establish circuits, there is little or no performance hit due to signaling. In the TCP example, the source sends a SYN message that traverses a cloud of TCP Switches sitting somewhere in the network. The SYN is intercepted by the ingress switch, which will first try to see if it has enough capacity on its outgoing link to carry a new circuit. If it does, it will simply start using the new capacity (for example, by filling an empty time slot, or using a new wavelength), and then forwards the message

[1] This figure is similar to the ones found by Paxson [16] and Labovitz et al. [17].

[2] Unless there was a route change, or packet mis-sequencing, in the PS portion of the network.

Table 2.- Some design choices in TCP Switching

Choice	One option	Alternative option	Notes
Circuit establishment	triggered by only TCP SYNs	triggered by first packet seen in a flow (can be any)	If there is a path reroute outside the TCP Switched cloud the SYN will not be observed. This is rare in practice
Circuit release	triggered by FIN (hard state)	triggered by inactivity timeout (soft state)	Neither is perfect. Connections with asymmetrical closings or with long inactivity periods can be severed
Handling of UDP packets	UDP traffic multiplexed through permanent circuits between edge routers	UDP packets contain sufficient state to create a connection, too. If TCP SYNs are not used, UDP and TCP flows are treated the same way.	UDP represents a small (but important) amount of traffic.
Signaling	in-band	out-of-band	In-band signaling requires no additional exchanges, but it is more complex
Routing of circuits	hop-by-hop routing	centralized or source routing	A centralized algorithm can provide global optimization and path diversity.
Circuit granularities	flat, i.e. all switches have the same granularity	hierarchical, i.e. different switches have a different granularities (in the core flows are bundled together)	A coarser granularity means that the switch can go faster, as it has to do less processing

downstream to set up the circuit. The intermediate switches do the same; check for capacity on the outgoing link, then forward the setup message if the request is acceptable. If any switch along the path has insufficient capacity on its outgoing link for a new flow, it can simply drop the packet, or buffer it for a short time, dropping it if no capacity becomes available.

Once the circuit is set up the SYN will reach the end host. The destination host will receive the SYN message as if it were a normal TCP connection. It will respond with a SYN+ACK, which will create a separate, independent circuit in the opposite direction. Once the handshake is complete, the TCP flow has its own private point-to-point circuit between the ingress and egress of the CS cloud.

We believe the complexity of the TCP Switches can be made small. Most of the additional processing is performed by the ingress boundary router, and takes place once per flow, rather than once per packet. It must recognize the first packet in a flow, decide which outgoing link should be used (i.e. routing), and then determine if there is sufficient capacity on the outgoing link to carry the new circuit.

Recognizing the first packet in a new flow requires a four-field, exact-match classifier. The routing decision can be made using the routing protocol already present on the router. The decision as to whether there is enough capacity to carry the new flow might be as simple as a counter representing the number of unused time slots or wavelengths.

When packets arrived for existing circuits, the ingress boundary router must be able to determine to which flow and circuit the packet belongs (using the classifier). The size of the classifier is determined by the number of circuits on the outgoing link. For example, an OC192c link carrying 64kb/s circuits requires 156,000 entries in its table.

At the egress boundary router, arriving data needs to be reassembled into packets (assuming they were broken into time slots. If they were carried on their own wavelength, then no reassembly should be needed).

Because most flows are short lived, the circuit establishment and tear down must be really fast. Given the simplicity of the signaling, we believe this can be done in hardware.

5. One example of TCP Switching

There are a number of ways to design a TCP Switch. Table 2 shows some of the choices.

We have been experimenting with TCP Switching via prototyping (in Linux) and simulation (using ns-2 [7]). We needed to make our own design choices in our experiments, and so list them here.

The core circuit switches are assumed to carry 64Kb/s circuits to match the access links of most network users. High capacity flows use multiple circuits. In our design, we opt for in-band signaling, which requires a modification to the circuit switch. (Alternatively, out-of-band signaling could be used, and would require no hardware changes to the circuit switch). We consider circuits to be idle or active. Idle circuits are ones that have not been allocated to a flow, and so carry no data. When the first packet starts flowing on an idle circuit (not necessarily a TCP SYN packet), the circuit switch tries to establish a new circuit. This allows the switch to carry UDP flows without modification, and makes the system robust against re-routing of TCP flows (about 0.4% of flows are re-routed today). The routing information is present in the first packet (i.e. the IP packet header) and is used to find the next hop for the circuit using regular IP routing tables. If a circuit cannot be established, the data is dropped and the boundary router is required to detect a timeout.

New flows are detected using a simple exact match classifier. The headers of arriving packets (IP addresses and TCP port numbers) are compared against a table of active flows to see if the flow belongs to an existing circuit, or whether a new circuit should be created. Given the duration of measured flows, we can expect the rate of the exact-match lookup to be approximately 31 Mpps and 50,000 new connections per second for an OC192c link. This is quite manageable in dedicated hardware.

Around 40% of the TCP flows have ACK packets arriving after the FIN. This happens because the communication in the other direction is still active, so it is unwise to close a connection right after seeing a FIN. Thus, we choose to use soft state, and wait until the connection times out due to inactivity. Interestingly, our simulations show that the performance is not very sensitive for a timeout value of around 60 seconds. This is similar to the results obtained for IP Switching [13].

The admission controller (in the boundary router) queues the calls that cannot be accepted for a period up to 1.5 minutes, which is half of the TCP connection establishment timeout value prescribed by RFC 1122 [4]. We have experimented with ways to determine the data rate assigned to a new flow, but this is beyond the scope of this paper. In brief, it could be obtained from a SLA/policy database or it could be carried as an option in the TCP header.

6. Experimentation

We are implementing TCP Switches in three different ways. First, we have simulated a TCP Switched network using ns-2 [7], and have used it to study the overall performance of a network that includes TCP Switching, the users' response times, and how TCP congestion control mechanisms perform.

Most interestingly, we have found that there are quite a few conditions under which CS gives users a faster download response time than with PS, for the same network topology; for example, when there is low variation in file sizes on a web server, and many contending clients, or also when the peak flow rate is much smaller than the link data rate. We have been able to confirm this theoretically, and will describe it in an upcoming paper.

We have also implemented an ingress boundary router as a kernel module in Linux 2.4 on a Pentium III operating at 1GHz. We measured the increased forwarding delay for each packet: regular IP forwarding takes 17 µs, and the addition of a (fairly inefficient) classifier and output packet scheduler for PS style QoS, increases the delay to 77 µs. On the other hand TCP Switching forwarding takes between 17 and 25 µs, and the circuit setup time in software is about 57 µs. These figures indicate that the performance of a TCP Switching boundary router is comparable to that of a regular router. Of course, we expect these numbers to be much reduced if implemented in dedicated hardware, and so we are just starting on an FPGA-based implementation.

7. Previous work

Recently several researchers have described the integration of IP and circuit switching in the core. There have been three main approaches to define signaling mechanisms that would add dynamism to the establishment and release of SONET/SDH circuits. They are MPLamdaS [10], OIF [11] and ODSI [12]. These three working groups provide the control mechanisms, and it is left up to the vendors to define how to monitor the traffic, what triggers the circuit establishment and how to allocate the bandwidth.

There are two architectures that try to address this decision making. The first one is optical burst switching [14], which queues packets up to a threshold, and then establishes a circuit with an explicit connection release time, also known as a burst. In the second one, Veeraraghavan et al. [15] define an end-to-end, circuit-switched network that is parallel to the packet switched Internet. Only big file transfers are transmitted through the CS network.

Our approach differs in the sense that TCP Switching re-uses the connection establishment mechanism to create a circuit for each flow (TCP, UDP or other type). We exploit the connection-oriented nature of the current Internet. We create a circuit only when (and as soon as) a flow starts, and we maintain this circuit until the moment the flow ends. In this respect, TCP Switching is most similar to IP Switching [13] in which user flows triggered the establishment of ATM virtual circuits.

8. Conclusions

New optical technology is being used to build very high capacity circuit switches. While these switches are already being used to replace SONET switches, it is not yet clear how we should control these circuit switches.

We have explored whether it might make sense to establish a new circuits for each application flow. At first glance, such fine granularity would appear to be expensive and impractical. It uses links inefficiently, seems to require complex processing and signaling, the maintenance of lots of hard state, and to suffer from poor robustness.

We agree that links are used inefficiently; but with the increased over-provisioning of link capacity, and with the difficulty of building high performance packet switches, we believe this might be the right (and possibly only long-term) tradeoff to make.

We believe that the signaling can be kept very simple if the first packet in a flow triggers the establishment of a new circuit, and that the processing is quite doable in dedicated hardware. The per-flow state consists of a mapping from flows to circuits; and the number is small enough to maintain in on-chip memory, even for OC192c links. Hard-state can be avoided by using inactivity timeouts for each circuit.

TCP Switching relies on the observation that almost all traffic today is carried in end-to-end connections; and there is no reason to think this will change. Given that the control mechanisms already exist to establish and remove TCP connections, why not use these same mechanisms to establish circuits?

TCP Switching provides one evolutionary path to exposing circuits to IP. Our experiments suggest so far that circuit switching can lead to lower response times for network users, and that TCP Switching is relatively easy to implement in a boundary router or in a core switch.

But given the controversy surrounding this topic, and the incompleteness of our work, we expect this is just one contribution to the ongoing debate.

9. Acknowledgments

We would like to thank Byung-Gon Chu, Ranganath Rao, Bo Yang and Feng Wang (students of CS344 at Stanford) for their contributions to the implementation of the TCP Switch in Linux and ns-2.

10. References

[1] Very High Performance Backbone Network Service, http://www.vbns.org

[2] NLANR traceroute measurements, http://amp.nlanr.net/

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

[4] R. Braden. "Requirements for Internet Hosts -- Communication Layers". RFC 1122.

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

[6] 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

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

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

[9] Mark Heinzl. "Operators of Fiber-Optic Networks Face Capacity Glut, Falling Prices Wall Street Journal". Wall Street Journal, New York, N.Y., Oct. 19, 2000

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] V. Paxson. "End-to-end routing behavior in the Internet". Proceedings of ACM SIGCOMM 96, pp. 25-38, Stanford, CA, Aug 1996.

[17] C. Labovitz, A. Ahuja, and F. Jahanian. "Experimental Study of Internet Stability and Wide-Area Network Failures". Proceedings of ACM SIGCOMM 00, Stockholm, Sweden, Aug/Sep 2000.

[18] T1 Committee. "Synchronous Optical Network (SONET) - Automatic Protection Switching", T1.105.01-2000, ANSI Standard, Mar 2000.

[19] A. Demers, S. Keshav, and S. Shenker. "Analysis and Simulation of a Fair-queueing Algorithm", Proceedings 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". Proceedings of ACM SIGCOMM 95, Cambridge, MA, Aug/Sep 1995.