TCP Switching: Exposing Circuits to IP

A SIMPLE TECHNIQUE COULD ELIMINATE CHOKE POINTS ON THE INTERNET. TCP SWITCHING—WHICH INTEGRATES CIRCUIT SWITCHING BENEFITS IN THE PACKET SWITCHED INTERNET—DEMONSTRATES THE POSSIBILITIES.

Pablo Molinero-Fernández
Stanford University

Nick McKeown
Stanford University

Circuit switches have simpler data paths and are potentially much faster than packet switches. Taking advantage of this difference makes very high capacity all-optical circuit switches feasible, whereas all-optical packet switches are a long way from commercial practicality. Peak-allocation, that is eliminating the benefits of statistical multiplexing, is circuit switches' main disadvantage, and what prevents their widespread adoption. However, the rising amount of already abundant link capacity will eliminate this drawback. Our research focuses on how the existing IP infrastructure can incorporate fast, simple (and perhaps optical) circuit switches. Several approaches to this already exist,[1-5] but we propose a technique called transmission-control protocol (TCP) switching in which each application flow (usually an individual TCP connection) triggers its own end-to-end circuit creation across a circuit switched core. Based on IP switching,[6] TCP switching incorporates modified circuit switches that use existing IP routing protocols to establish circuits. Routing occurs hop by hop, and circuit maintenance uses soft state, that is, it is removed through an inactivity timeout.

Many believe that only routers, links, and end hosts, all using packet switching, comprise the Internet. In reality, the Internet uses circuit switching, both at its core (Sonet, SDH, dense wave digital multiplexing or DWDM), and in its last mile (modems, DSL). Internet Protocol treats these circuits as static, point-to-point links connecting adjacent nodes; the physical circuits and IP belong to different layers, and they are completely decoupled since they operate autonomously and without cooperation. Decoupling of layers has many advantages. It lets the circuit switched physical layer evolve independently of IP—to both Sonet/SDH, and DWDM. IP runs over a huge variety of physical layers regardless of the underlying technology. However, a lot of repetition exists between the packet-switched IP layer and the circuit-switched physical layer. For example, a network must route both IP datagrams and circuit paths, yet, they use different routing protocols, and their implementations are incompatible. This makes simple and obvious operations infeasible. For example, if traffic increases between two neighboring routers, no simple or standard way exists to automatically increase the capacity between them. The problem is with how circuit switches interact with IP routers. TCP switching presents a method of interaction, promising automatic and dynamic circuit allocation.

Packet-switched Internet

With the speed benefits offered by circuit switching, why does the Internet use packet switching? Because of data traffic's bursty

0272-1732/02/$17.00 © 2002 IEEE

nature, statistical multiplexing allows heavier traffic with packet switching than circuit switching over the same links. In the 1970s and 1980s, bandwidth efficiency became paramount because Internet service providers leased their linked lines—these lines were an expensive and scarce resource. Packet switching was also thought to provide more robustness than circuit switching. Because the routing tables present in the router individually direct each packet, getting around a link or node failure only requires an update to these tables.

Bandwidth efficiency and ease of reconfiguration advantages do not hold today. Reports show that networks only lightly utilize most links, frequently as low as 10 to 15 percent of their capacity.[7] Furthermore, we anticipate utilization to continue decreasing, due, in part, to the huge investment in optical fibers and the glut of capacity in the Internet core.[8] Link capacity is no longer a scarce resource. In terms of robustness, packet-switched routing protocols do not necessarily lead to simpler and quicker reconfiguration. Routing protocols in use now are extremely complicated and can take seconds or even minutes to converge, and reroute around failures.[9] On the other hand, most circuit-switched equipment, Sonet for example, must recover in less than 50 ms.[10] Therefore, having per-circuit state in a network does not necessarily prevent robustness.

Circuit-switched alternative

Although the original reasons for using packet switching no longer hold, this does not automatically make circuit switching the best option. Understanding the characteristics of circuit switching is necessary before deploying it in the Internet core. Circuit switching offers many advantages.

No Buffers

Circuit switching requires no buffers, while a packet switch maintains buffers to hold packets during heavy traffic. These buffers must be both large and fast. Routers maintain about ($R \times RTT$) bits of packet buffers, where R is the line rate, and RTT is the typical round-trip time between any two end hosts (about 0.25 seconds). For example, a 10-Gbps line card maintains about 2.5 Gbits (300 Mbytes) of storage. To buffer packets as quickly as they arrive, the buffers must run at least as fast as the line rate. Commercially available memory devices are either optimized for storage but not speed (DRAMs), or speed but not storage (SRAMs), and so are ill suited for large, fast packet buffers. Designers find it challenging to build packet buffers for a 10-Gbps line card, and it is even more difficult to achieve 40 Gbps, particularly when power consumption is an issue. Most router capacity is limited by memory availability. That is, routers require a fast and large memory, but need minimum power consumption. Circuit switching eliminates this problem. Data arrives for processing on one channel, and the switch immediately places the incoming information flow onto the outgoing channel, therefore requiring little memory.

Optical switching

Circuit switches can take advantage of optical switching technology. Advances in optical circuit switching technology in space-[11,12] and wave-division switching,[13] promise large increases in switching capacity, and reductions in power consumption. Several optical circuit switching products exist already. However, packet switches using only optics are currently infeasible because of the lack of a viable optical buffering technology.

Switching capacity

Circuit switching eliminates per-packet processing. On the other hand, a router must process each packet as it arrives, performing an address lookup and modifying the packet header. A network processor or a dedicated ASIC might perform this processing. Processors double in speed about every eighteen months, but the link capacity in DWDM currently doubles every seven months. Circuit switching may offer the only alternative that takes advantage of the Internet's growing capacity.

This reason and the previously discussed absence of buffering and the advantage of optical technology, lead us to conclude that, for a given technology, circuit switches provide more capacity than packet switches. An informal survey of commercial routers and circuit switches shows that the latter (for example, Ciena's MultiWave CoreDirector or

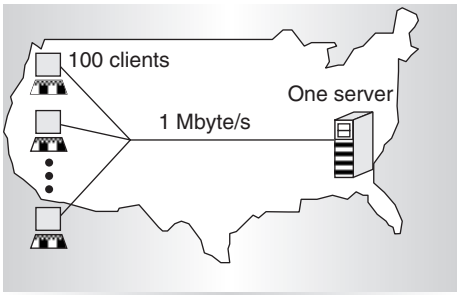

Figure 1. A simple network scenario, which demonstrates the relative response times of circuit switching, and packet switching.

Lucent's WaveStar OLS 1.6T) currently implement about four to 10 times as much switching capacity as routers (for example, Cisco's 12416 Internet router.)

Quality of service

Circuit switching lends itself to simple, intuitive (and degenerate) QoS. Bandwidth is peak allocated, and there is no delay jitter. In packet switching, there are two ways to provide bandwidth and delay guarantees: heavy overprovision of link capacity, essentially, emulating circuit switching, or use of well-known algorithms such as weighted fair queueing,[14] generalized processor sharing,[15] and deficit round robin.[16] Emulating circuit switching offers no advantage in link utilization, yet, it still requires per-packet processing. The downside to algorithms is that they require complex per-packet processing and the maintenance of large amounts of state, significantly increasing router complexity. Further, users often find it difficult to understand, configure, and use these algorithms.

Simple admission control

Circuit switching replaces per-packet scheduling, or drop decisions, with per-flow admission decisions. Deciding whether or not to admit a flow is simple: if there is a spare circuit on the outgoing link, then the switch accepts the flow; otherwise the switch blocks the flow. In addition, because flows consist of multiple packets, the switch makes decisions less often than in packet switching.

User response time

In some situations, circuit switching improves response time (the time from the point a user's host first sends a request until completion of the file download). For example, Figure 1 illustrates a case in which 100 clients simultaneously request the same 1-Mbit file from a remote server via a 1-Mbps bottleneck link (to simplify this example we consider 1 Mbit to be 106 bits, not 210 bits). With packet switching, all downloads finish after 100 ms, whereas with circuit switching the average response time is 50.5 ms, and all but one of the individual downloads finish sooner.

However, circuit switching will not always give a lower response time, particularly when one flow can dominate the link. If we modify our example so that the first client begins the download of a 1-Gbit file slightly before the other clients, it will take up the link for a long time. The average response time in this case for circuit switching is 1049.5 ms, compared to 108.5 ms for packet switching. However, this example does not represent what typically occurs in the Internet core. In practice, one circuit cannot take up the whole link because a link supports a large number of circuits. The speed of an application flow's access link limits its rate. Simulations of various topologies (using the ns-2 network simulator; http://www.isi.edu/nsnam/ns/) suggest that, in practice, the response time is very similar for packet and circuit switching. We have been able to confirm this using simple analytical models.

Circuit-switching disadvantages

Circuit switching does have some disadvantages. However we will argue that they are not significant enough as to prevent the adoption of circuit switching in the Internet.

State maintenance

Circuit switching requires establishing circuits and the associated state between the switches before data can be transferred. A large number of circuits might require a circuit switch to maintain a lot of state. If this is hard state, then maintenance is complex. However, in practice, by observing real packet traces we have found the number of flows, and the rate at which they are added and removed, is quite manageable in simple hardware using soft state. This holds true even for a high-capacity switch. The "Typical Internet flows" sidebar discusses current Internet traffic patterns.

Typical Internet flows

TCP switching establishes a circuit for each application flow. To understand the feasibility and sensibility of this technology, we need an understanding of data flows in the Internet today. Because over 90 percent of Internet traffic is transmission-control protocol (TCP)—both in terms of packets and bytes—it is important to understand what a TCP flow is, and how it behaves. We have studied trace route measurements, as well as packet traces from OC-3 and OC-12 links in vBNS. Table 1 describes the typical flow in the Internet.

TCP connections usually last less than 10 seconds, carry less than 4 Kbytes of data and consist of fewer than 12 packets in each direction. Less than 0.4 percent of connections experience a route change. The typical user requests a sequence of files for downloading, and wants the fastest possible download for each file. In most cases, the requested data is not used until the file has completely arrived at the user's machine.

Table1. TCP Flows in the Internet. The figures indicate the range for the 80-percentile, the average and the median for the different links that we observed.*

Traffic characteristics	80-percentile	Average	Median
TCP flow duration in seconds	< 4-10	< 3-7	< 0.5-1.2
Packets per flow	< 12	< 10-200**	< 5-9
Bytes per flow	< 2.5-4	< 9-90**	< 0.6-1.3
Flow bandwidth (bytes/duration)	< 50-100	< 20-140**	< 8-15
Percentage of flows with retransmissions	< 7.8	< 5.6	< 4.7
Percentage of flows experiencing reroute	< 0.19	< 0.39**	< 0.02
Asymmetrical connections	Around 40 percent of the flows transmit an ACK after the FIN, that is, they acknowledge that a data packet that was sent in the other direction.		

* (The data used in the table was made available through the National Science Foundation Cooperative Agreement No. ANI-9807479, and the National Laboratory of Applied Network Research; http://moat.nlanr.net/Traces/.)

** These magnitudes present a non-negligible amount of samples with very high values, making the statistical distribution have long and heavy tails. This is why the average is higher than the 80-percentile.

Wasted capacity

Circuit switching requires circuits to be multiples of a common minimum circuit size. For example, Sonet commonly cross connects to provision circuits in multiples of STS-1 (51 Mbps). Having flows whose bandwidth is not an exact multiple wastes link capacity. There is a tradeoff between using smaller circuit granularity and the amount of state maintained by the switch.

Blocking under congestion

Circuit switching blocks packets belonging to a new flow while the new stream of data waits for a circuit to become available. If no available circuit exists, the new request waits, or gets blocked, until a circuit is free. This data flow system works differently from the link-sharing paradigm present in the Internet today, in which packets will still make (albeit slow) progress over a congested link.

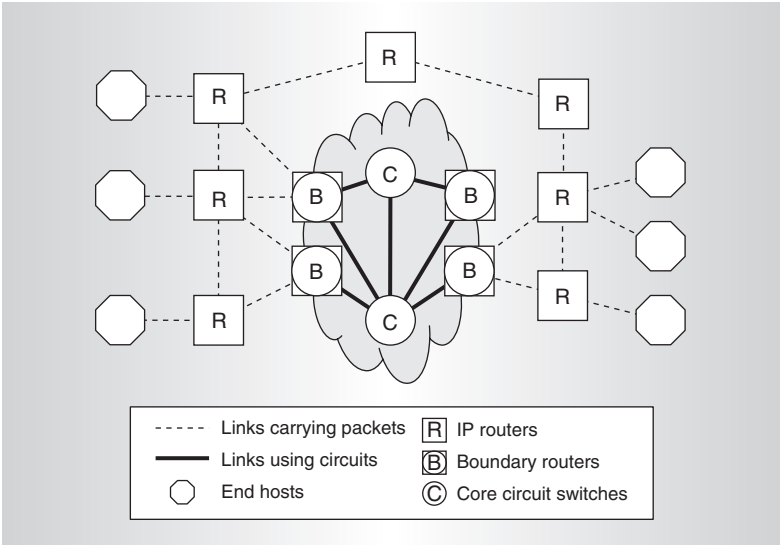

Figure 2. TCP switching represented as a cloud of circuit switching inside a packet switched network.

TCP switching in practice

TCP switching consists of establishing fast, lightweight circuits triggered by application-level flows. Figure 2 shows a self-contained TCP switching cloud inside the packet-switched Internet. The network's packet-switched portion does not change, and circuit

switches, such as Sonet cross connects make up the core of the circuit switching cloud. These circuit switches have simplified mechanisms to setup and teardown circuits. Boundary routers are conventional routers with circuit switched line cards, which act as gateways between the packet switching and circuit switching.

The first arriving packet from an application flow triggers the boundary router to create a new circuit. An inactivity timeout removes this circuit later. Hence, TCP switching maintains circuits using soft state.

In the most common case, the application flow is a TCP connection, and the first packet arriving at the boundary router is a TCP synchronization (SYN) packet. This automatically establishes the circuit, as part of the TCP connection setup handshake. By triggering the circuit establishment when the router detects the first packet—whether or not it is a TCP SYN packet—TCP switching is also suitable for non-TCP flows, and for TCP flows that experience a route change in the packet-switched network.

An examination of each step in TCP switching, following Figure 2, shows how this type of network architecture establishes a circuit end to end for a new application flow. When the boundary router detects an application flow's first packet, it examines the IP packet header and makes the usual next hop routing decision to determine the outgoing link. The boundary router then checks for a free circuit on the outgoing link (for example an empty time slot, or an unused wavelength). If one exists, the boundary router begins to use it, and forwards the packet to the first circuit switch in the TCP switching cloud. If no free circuits exist, the protocol can buffer the packet with the expectation that a circuit will become free soon, it could evict another flow, or it could just drop the packet, forcing the application to retry later. Current implementations of TCP will resend a SYN packet after several seconds, and will keep trying for up to three minutes (depending on the implementation).

If the circuit is successfully established on the outgoing link, the packet is forwarded to the next hop circuit switch. The circuit switch will detect that a previously idle circuit is in use. It then examines the first packet on the circuit to make a next hop routing decision using its IP routing tables. If a free outgoing circuit exists, it connects the incoming circuit to the outgoing circuit. From then on, the circuit switch does not need to process any more packets belonging to the flow.

The circuit establishment process continues hop by hop across the TCP switching cloud until, hopefully, the circuit is established all the way from the ingress to the egress boundary router. The egress boundary router receives packets from the circuit as they arrive, determines their next hop, and sends them across the packet-switched network toward their destination.

In its simplest form, TCP switching allows all boundary routers and circuit switches to operate autonomously. They can create circuits, and remove (timeout) circuits, independently. Obvious alternative approaches include buffering the first packet while sending explicit signals across the circuit-switched cloud to create the circuit. However, this removes autonomy and soft state, and so we prefer to avoid this method.

Our research finds that circuit switch and the boundary router complexity is minimal. The ingress boundary router performs most packet processing. It has to inject incoming packets from existing flows into the corresponding outgoing circuits, like any regular router would. Additionally the ingress boundary router must recognize the first packet in a flow, decide which outgoing link to use (routing), and then determine if the outgoing link has sufficient capacity to carry the new circuit. On the other hand, core circuit switches only need to perform processing once per flow, rather than once per packet.

Recognizing the first packet in a new flow requires the boundary router to use a four-field, exact-match classifier. When packets arrive for existing circuits, the ingress boundary router must determine to which flow and circuit the packet belongs (using the classifier). The size of the classifier depends on the number of circuits on the outgoing link. For example, an OC-192c link carrying 56-Kbps circuits requires 178,000 entries in its table.

The short life of most flows requires fast circuit establishment and tear down. Given the simplicity of the signaling in TCP switching, we believe that we can achieve this in hardware.

Table 1. Design choices in TCP switching.

Choice	Option 1	Option 2	Notes
Circuit establishment	Triggered by first packet seen in a flow (can be any packet type).	Triggered by TCP SYN packets only.	If there is a path reroute outside the TCP switched cloud, the switch will not detect the SYN packet. This is rare in practice.
Circuit release	Triggered by inactivity timeout (soft state).	Triggered by a finish (FIN) signal (hard state).	Neither option is perfect. The switch might sever connections that have asymmetrical closings or long inactivity periods.
Handling of non-TCP flows	Treats user datagram protocol (UDP) and TCP flows the same way.	Multiplex UDP traffic into permanent circuits between boundary routers.	UDP represents a small (but important) amount of traffic.
Signaling	None. Circuit establishment is implicit based on observed packets.	Explicit in-band or out-of-band signaling to establish and remove circuits.	In-band signaling requires no additional exchanges, but it is more complex.
Circuit routing	Hop-by-hop routing.	Centralized or source routing.	A centralized algorithm can provide global optimization and path diversity.
Circuit granularities	Flat, that is, all switches have the same granularity.	Hierarchical, that is, fine circuits are bundled in coarser circuits in the inner core.	A coarser granularity means that the switch can go faster, because it has to process less.

Design options

We have experimented with TCP switching networks via prototypes (in Linux) and simulation (using ns-2). We considered several variables to make design choices in our experiments. Table 1 shows some of the options.

Circuit characteristics

We assume in our experiments that the core circuit switches carry 56-Kbps circuits to match the access links of most network users. High capacity flows use multiple circuits. In our design, we use implicit signaling, that is, the arrival of a packet on a previously inactive circuit triggers the switch to route the packet and create a new circuit. Alternatively, a design could use explicit out-of-band signaling in which the first packet is sent over a separate circuit (or even a separate network) to the signaling software on the circuit switch. In this case, hardware changes to the switch are not necessary.

Flow detection

The exact-match classifier detects new flows. The classifier compares the headers of arriving packets against a table of active flows to check if the flow belongs to an existing circuit, or whether to create a new circuit. Given the duration of measured flows, we expect about 31 million lookups and 52,000 new connections per second for an OC-192c link. This is quite manageable in dedicated hardware.

We use soft state and an inactivity timer to remove connections. For TCP flows, we might remove circuits when the router detects a FIN signal, but in about 40 percent of TCP flows, an acknowledgement (ACK) packet arrives after the FIN because the communication in the other direction is still active. The timeout duration is a tradeoff between efficiency in bandwidth and signaling; if the circuit is timed out too quickly, the bandwidth becomes free for another flow, but a new circuit must be established again by the boundary router when the inactive flow resumes. Our simulations suggest that a 60 second timeout value will reliably detect flows that have ended (which is similar to results for IP switching).[6] But the cost of using such a long timeout value is high because the circuit remains unused for long time. We are exploring dynamic timeout values and policies for circuit reuse, such as having a new flow take over the circuit of the

least-recently-active flow. By default, all flows receive a circuit of the same size (56 Kbps in our prototype and simulations). Even though the boundary router can allocate different numbers of circuits to different flows, these techniques are beyond the scope of the basic operation of TCP switching.

Experimentation

We have implemented an ingress boundary router as a kernel module in Linux 2.4 on a 1-GHz Pentium III. We measured the increased forwarding delay for each packet. Regular IP forwarding takes 17 µs, whereas in TCP switching forwarding a packet in the boundary router takes 17 to 25 µs. In our nonoptimized software, the circuit setup time is approximately 57 µs, fast enough to handle new connection requests of an OC-48c link. We expect these numbers to drop dramatically if we implement part of our software in dedicated hardware.

Approaches

Recently several researchers have described the integration of IP and circuit switching in the Internet core. Three main approaches define signaling mechanisms that will add dynamism to the circuit linking process in Sonet/SDH circuits: Generalized Multiprotocol Label Switching,[1] Optical Internetworking Forum,[2] and Optical Domain Service Interconnect.[3] These three working groups provide the control mechanisms for managing circuits, and vendors define how to monitor traffic, what triggers circuit establishment, and how to allocate bandwidth.

Two architectures try to address this decision-making. Optical burst switching[4] queues packets up to a threshold and then establishes a circuit with an explicit connection release time (also known as a burst). Veeraraghavan et al.[5] define an end-to-end, circuit-switched network that is parallel to the packet-switched Internet. This model only transmits large file transfers through the circuit-switched network.

Our approach differs in that TCP switching (usually) piggybacks the creation of a circuit on the setup phase of a TCP connection. In this respect, TCP switching is similar to IP switching[6] in which user flows triggered the establishment of asynchronous transfer mode virtual circuits.

The Internet's current over provisioning of link capacity and the difficulty in building high-performance packet switches point out the need for research into alternatives. One obvious approach is to use very high capacity circuit switches that already incorporate new optical technology. While these switches replace Sonet switches, it is not yet clear how the network should control them. Research into answering this question is still far from complete. TCP switching provides one path to exposing circuits to IP. Our experiments suggest that circuit switching yields similar response times for network users and that TCP switching is relatively easy to implement in boundary routers and core switches. MICRO

Acknowledgments

We thank Byung-Gon Chu for his contributions to the implementation of the TCP Switch in Linux.

References

1. A. Banerjee et al., "Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques," *IEEE Comm. Magazine*, Jan. 2001, pp. 144-151.
2. G. Bernstein, B. Rajagopalan, and D Spear, "OIF UNI 1.0: Controlling Optical Networks," white paper, Optical Internetworking Forum, Mar 2001; http://www.oiforum.com/ (current Jan. 2002).
3. A. Copley, "Optical Domain Service Interconnect (ODSI): Defining Mechanisms for Enabling On-Demand High-Speed Capacity from the Optical Domain," *IEEE Comm. Magazine*, Oct. 2000, pp. 168-174.
4. M. Yoo, C. Qiao, and S. Dixit, "Optical Burst Switching for Service Differentiation in the Next-Generation Optical Internet," *IEEE Comm. Magazine*, Feb. 2001, pp. 98-104.
5. M. Veeraraghavan et al., "Architectures and Protocols that Enable New Applications on Optical Networks," *IEEE Comm. Magazine*, Mar. 2001, pp. 118-127.
6. P. Newman, G. Minshall, and T. Lyon, "IP Switching: ATM Under IP," *IEEE/ACM Trans. Networking*, vol. 6, no. 2, Apr. 1998, pp. 117-129.
7. A.M. Odlyzko, "Data Networks are Mostly Empty and for Good Reason," *IT Profes-*

sional, vol. 1, no. 2, pp. 67-69, Mar/Apr 1999.
8. M. Heinzl, "Operators of Fiber-Optic Networks Face Capacity Glut, Falling Prices," *The Wall Street J.*, Oct. 19, 2000.
9. C. Labovitz, A. Ahuja, and F. Jahanian, "Delayed Internet Routing Convergence," *Proc. ACM Sigcomm*, ACM Press, New York, 2000, pp. 175-187.
10. *ANSI standard T1.105.01-2000, Synchronous Optical Network (Sonet): Automatic Protection Switching*, Am. Nat'l Standards Inst, New York, Mar. 2000.
11. D. Neilson et al., "Fully Provisioned 112×112 Micromechanical Optical Crossconnect with 35.8Tb/s Demonstrated Capacity," *Proc. Optical Fiber Conf.*, 2000.
12. Agilent Technologies, *Agilent Technologies Photonic Switching Platform*, white paper; http://www.agilent.com/cm/photonicswitch/white_papers/wp_prelim.shtml (current Jan. 2002).
13. B. Pesach et al., "Free-Space Optical Cross-Connect Switch by use of Electroholography," *Applied Optics*, vol. 39, no. 5, Feb. 2000, pp. 746-758.
14. A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair-Queuing Algorithm," *Proc. ACM Sigcomm*, ACM Press, New York, 1989, pp 1-12.
15. 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 Trans. Networkin*g, vol. 1 no. 3, Jun. 1993, pp. 344-357.
16. M. Shreedar and G. Varghese, "Efficient Fair Queuing using Deficit Round Robin," *Proc. ACM Sigcomm*, ACM Press, New York, 1995, pp. 231-242.

Pablo Molinero-Fernández is a PhD candidate in electrical engineering at Stanford University. His research interests include circuit and packet switching for high-speed networks, Internet traffic analysis and reliable multicast. Molinero-Fernández has an MS in electrical engineering from Stanford University, a Telecommunications Engineer degree from the Technical University of Madrid (UPM) and the National School of Higher Education in Telecommunications in Paris (ENST-Paris), and an MS in physics from the National Distance-Learning University (UNED), Madrid.

Nick McKeown is a professor of electrical engineering and computer science at Stanford University. His research interests include the architecture, analysis, and design of high-performance switches and Internet routers, IP lookup and classification algorithms, scheduling algorithms, Internet traffic analysis, traffic modeling, and network processors. McKeown has a PhD from the University of California at Berkeley in electrical engineering and computer sciences. He is the Robert Noyce Faculty Fellow at Stanford, a Fellow of the Alfred P. Sloan Foundation, and recipient of a Career award from the National Science Foundation. He received the 2000 IEEE Rice Award for the best paper in communications theory. He is a senior member of the IEEE and serves as an editor for the *IEEE Transactions on Communications* and *ACM/IEEE Transactions on Networking*.

Direct questions and comments about this article to Pablo Molinero-Fernández, Computer Systems Lab, Stanford University, Gates Building (room 342), 353 Serra St., Stanford, Calif., 94305; molinero@stanford.edu.