A design for an Internet router with a digital optical data plane

Joe Touch[*a], Joseph Bannister[b], Stephen Suryaputra[a], Alan E. Willner[c]
[a]USC/ISI, 4676 Admiralty Way, Marina del Rey, CA, USA 90292-6695;
[b]The Aerospace Corporation, 2310 E. El Segundo Blvd., El Segundo, CA, 90245-4609;
[c]Dept. of EE-Systems, USC, 3740 McClintock Ave., Los Angeles, CA 90089-2565

ABSTRACT

This paper presents a complete design for an optical Internet router based on decomposing the steps required for IP packet forwarding. Implementations of hopcount decrement and header matching are integrated with a simulation-based approach to variable-length packet merging that avoids recirculation, resulting in an all-optical data plane. A method for IPv4 checksum computation is introduced, and this and previously designed components are extended from binary to higher-density (multiple bits per symbol) encodings. The implications of this design are considered, including the potential for chip-level and system integration, as well as the requirements of basic optical processing components.

Keywords: Internet router, optical packet switching, digital optical packet processing, optical Internet

1. INTRODUCTION

A router with an optical data plane is needed to support the high-speed Internet. Optical packet switching can support the Internet's multiplexing efficiency and flexibility, while avoiding costly and complex conversion from optical formats required for high-speed, long-distance transmission. The operation of an Internet router can be reduced to its four basic core functions – hopcount decrement, header lookup, checksum calculation, and packet merging. This paper presents our experience implementing or simulating each function and the potential for future integration to a complete device.

1.1 The need for optical routing

A high-speed Internet requires per-packet sharing and optical transmission. High-efficiency capacity sharing requires packet-level processing. Packets are variable-length messages and the packet-level multiplexing results in an effective capacity gain of 3-10 times larger, *i.e.*, a 10Gb/s link shared using packet multiplexing can support tens to hundreds of users with 1Gb/s links, because Internet traffic is bursty and packet-based sharing is efficient. This efficiency occurs for packet multiplexing and is undermined by sharing on longer timescales, *e.g.*, for packet bursts or persistent channels.

High-speed, long-distance data transmission is necessarily optical. Electronic transmission is limited to roughly 10Gb/s over 10m. Longer distances result in reduced data rates but higher rates result in reduced distances, so optical transmission is required when aggregate capacity exceeds 10Gb/s. That requirement is driven out towards the aggregation edge by increasing user access rates.

An Internet router is a device with multiple inputs and outputs that shares capacity on a per-packet basis[1]. Assuming its inputs and outputs are necessarily optical, an optical router would avoid complex and costly OEO conversion, Routers are composed of a control plane and a data plane; the control plane exchanges information with other routers to ensure that packets make progress towards their destination, and the data plane uses a packet switch to direct incoming packets as indicated by their header information and control plane configuration (a process known as 'forwarding' when it applies to IP datagrams). In this paper, we focus on developing an optical data plane to natively switch optically encoded packets and we assume a conventional electronic control plane for management operations.

1.2 Some history

The Internet is the most recent stage of an evolution in communication architecture that began with telephony. Early network architectures used circuits, a dedicated physical path between endpoints. The need for more flexible sharing led to the development of virtual circuits, in which a circuit is emulated by dedicating fractions of resources over multiple paths. In the mid-1990's, the Internet's increasing need for flexibility and more efficient sharing led to the development

[*] touch@isi.edu, http://www.isi.edu/touch

of Tag Switching[2] (also known as Flow Switching[3] and more recently as Multiprotocol Label Switching, MPLS[4]), in which groups of packets are handled as a set using configurations that are deployed in advance. The final step in this evolution towards increasingly shorter multiplexing timescales culminated in the development of electronic packet switching that was capable of electronic transmission rates towards the turn of the last century; this native electronic packet switching is the basis of electronic Internet routers.

Optics is currently recapitulating this evolution. It began with dedicated light paths and evolved to virtual light paths using multiplexing wavelengths (WDM), with wavelength conversion to avoid the need for global coordination of wavelength assignment. Optical Burst Switching[5] (another variant is known as Terabit Burst Switching[6]) is the optical corollary of Tag Switching, configuring paths for small groups of adjacent packets with the same destination, effectively providing very short timescale circuits. Optical packet switching is the obvious conclusion of this evolution, in which resources are assigned for each message, which is why it is the basis of our effort.

The alternative would be to optimize the network configuration to support less dynamic, adaptive, and efficient sharing. This requires data exchange patterns that are both known in advance and are stable over long periods. Such patterns occur in datacenters and other researchers are developing optical circuit systems for that environment[7]. We prefer to focus on the more general, most flexible solution of optical packet switching, which would support not only these known, long-scale traffic patterns but unknown, rapidly shifting patterns prominent elsewhere throughout the Internet.

1.3 Our assumptions

We focus on the Internet router as the device that shares capacity between interconnected optical links. As a result, we assume optical links that do not introduce their own, distinct sharing mechanism - *i.e.*, they are individual, directional channels with no further partitioning by wavelength, phase, polarization, *etc*. As with electronic Internet channels, we assume that these dimensions may be used to encode information, *e.g.*, symbols that represent multiple bits, but that they are not otherwise used to partition packets within a link.

Similarly, we assume point-to-point (*i.e.*, unicast) links. Shared links require a media sharing arbitration mechanism; this arbitration is more efficient when coordinated within a packet switch, which is why Ethernet evolved from shared links to unicast links between packet switches.

We avoid assumptions on the size of packets, other than that they conform to Internet requirements. IPv4 packets (also called 'datagrams') are a minimum of 20 bytes, but are typically exhibit a bimodal distribution at 40 and around 1500 bytes[8]. IPv6 packets (these too are called 'datagrams') are a minimum of 40 bytes and typically exhibit a bimodal distribution at 60 and around 1500 bytes[9]. Packets on a link are assumed to have bytes in the order specified in the IPv4 or IPv6 formats – header bytes arrive in the order that they are defined in the IP standards. We also do not assume adjacent (or non-adjacent) packet sequences with the same destination, *i.e.*, we do not assume typical flow or burst sizes. We want a solution that works in the absence of such flow properties, without advance setup.

We assume an optical encoding compatible with optical processing, because conversion between transmission and processing encodings is as complex and costly when both are optical as when converting from optical to electrical, and we want to avoid that cost and complexity. Our preliminary implementations utilize binary encodings (*e.g.*, on-off keying, RZ encoding, or binary phase-shift keying (BPSK); in Section 7 we address extending these designs to higher-density, multiple bit symbols (*e.g.*, N-PSK) for more efficient long-distance transmission.

2. INTERNET ROUTER ARCHITECTURE

Internet routers are typically electronic, even when interfaced to optical links. They switch incoming packets towards their destination, a process known as "forwarding" when it refers to Internet packets (datagrams), as shown in Figure 1 (left). A forwarding engine (or 'forwarder') examines the packet header, matching it to entries in a forwarding table. That table is determined manually by the operator or by exchanges with the control planes of adjacent routers. The packet passes through a switching fabric, configured based on the result of this lookup. At various locations in this process, packets are queued to avoid backlogs due to slow header processing, switch fabric availability, output port contention, or to adjust between internal transfer speeds and incoming and outgoing line data rates.

Although Internet routers may include a number of additional capabilities, such as filtering and proxying (relaying) high level protocols, the core operation of a router is composed of four independent functions in sequence: forwarding lookup, hopcount decrement, checksum computation, and packet multiplexing (Figure 1, right)[10].

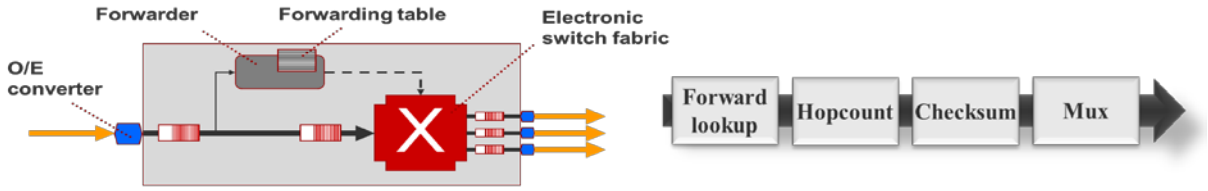

Figure 1. Basic data (thick lines) and control (thin) in an Internet router (left, color online), and internal sequence (right).

Packet forwarding lookup, sometimes called 'header lookup', matches the header destination address to an internal forwarding table, whose entries are patterns. IP datagrams are matched in this table using "longest prefix" matching. Forwarding entry patterns consist of a sequence of consecutive zeroes and ones, followed by "don't cares" (indicated as 'x'), sometimes indicated with the "don't cares" as zeroes followed by a "mask length" indicating the number of non-don't care bits (*e.g.*, 11xxxxxx is denoted as 11000000/2). All matches start at the high-order bit and proceed to lower-order bits. Longer matches take precedence over shorter matches, *e.g.*, a packet destined for address 11010111 that matches both 11000000/2 (*i.e.*, 11xxxxxx) and 11010000/4 (*i.e.*, 1101xxxx) would select the latter. IPv4 addresses are 32 bits long, but typically patterns are no longer than 24 bits inside the network (*vs*. at the user edge); IPv6 addresses are 128 bits long, but typically patterns are 56 bits or less. Routers in the core of the Internet can have several hundred thousand distinct patterns and packets not matching any patterns are silently dropped.

The hopcount is a field in the header of Internet packets. In IPv4, it is called "Time-to-Live" (TTL), which originally indicated the number of seconds a packet was allowed to live in the network. In IPv6, the field has been renamed "Hop Limit" to more accurately reflect what it tracks – the maximum number of routers traversed (in IPv4, every router is required to decrement the TTL by at least one even when a packet takes less than one second to forward). In both cases, the hopcount field is decremented by at least one (typically exactly one) at every router; when the count reaches zero the packet is discarded. The hopcount field avoids the potential for packets to accumulate and thus disable a network; an 'immortal' packet that is misrouted in a loop might interfere with new packets entering. By ensuring that all packets 'die' after some finite time, Internet architects protect the network from transient routing loops and ensure that any network can be cleaned of 'zombie' packets by shutting down network inputs, rather than requiring the entire network to be disconnected and/or rebooted. Although it may seem wasteful to lookup a packet that might be then dropped because its hopcount is exceeded, it is more likely that a packet would be dropped because of a forwarding lookup failure, so this step often happens second in the sequence. In addition, a packet might be destined to the router itself (*e.g.*, to its control plane), in which case the hopcount would not be decremented because the packet does not strictly traverse the router; this decision cannot be made until after the forwarding step.

IPv4 packets include a checksum that validates the correctness of the header[8]. The IP checksum is validated and recomputed at each hop using the Internet checksum algorithm - it is a 16-bit, one's complement sum of the data in the header[11]. This checksum is also used to validate the entire contents of packets at other protocol layers, notably TCP and UDP. The one's complement sum is the same as a two's complement sum whose carry-out (from the high-order bit) is wrapped around as the carry-in to the low-order bit. The IP checksum changes whenever any field of the packet header changes, *e.g.*, when the hopcount (TTL) field is decremented. IPv6 does not include a separate header checksum; it assumes that the IP header is sufficiently protected by checksums at other layers (*e.g.*, link)[9]. Again, although it may seem wasteful to do forwarding lookup and hopcount decrement on a packet whose checksum is invalid, such cases are sufficiently rare that it is effective to process the unmodified header (to validate the checksum) and modified header (to recompute the new checksum) at the same time as the third step in the sequence.

Finally, a router needs to deal with output port contention, when two packets seek the same output port at the same time. This is a multiplexing function, needed to manage sharing of the output resources. It necessarily requires either delaying (preferably) or discarding some of the packets competing for the same resource. The fewer discarded packets, the higher the efficiency of the multiplexing function.

Each of these functions can be challenging to convert to optical processing in different ways. Forwarding lookup is difficult because of the length of the patterns and the number of concurrent patterns being matched. Hopcount decrement and Internet checksum validation and recomputation are both difficult because they require mathematical digital optical processing. Multiplexing is difficult because it requires delaying packets and light can be difficult to delay. However, for

each of these challenges, we have developed a candidate approach that has been validated either by experimentation or simulation as presented in Sections 3–6. A final challenge involves integrating these steps and composing them in sequence, which is addressed in the discussion of feasibility presented in Section 7.

3. OPTICAL FORWARDING

Optical packet forwarding requires performing a longest prefix match on a potentially very large number of forwarding table patterns, each of which could be 24–64 bits long. This process is typically simplified in modern routers, so that the forwarding table uses only fixed-length entries, replicating entries where necessary to represent shorter patterns. As a result, longest-prefix matching is often implemented as fixed-length pattern match, which can be implemented optically a number of correlators (Figure 2, left)[12]. The number and length of the correlators can be managed by compressing the forwarding table to represent the most frequently used entries, in which several hundred thousand entries can be sufficiently represented by approximately 100 entries (Figure 2, right)[13].

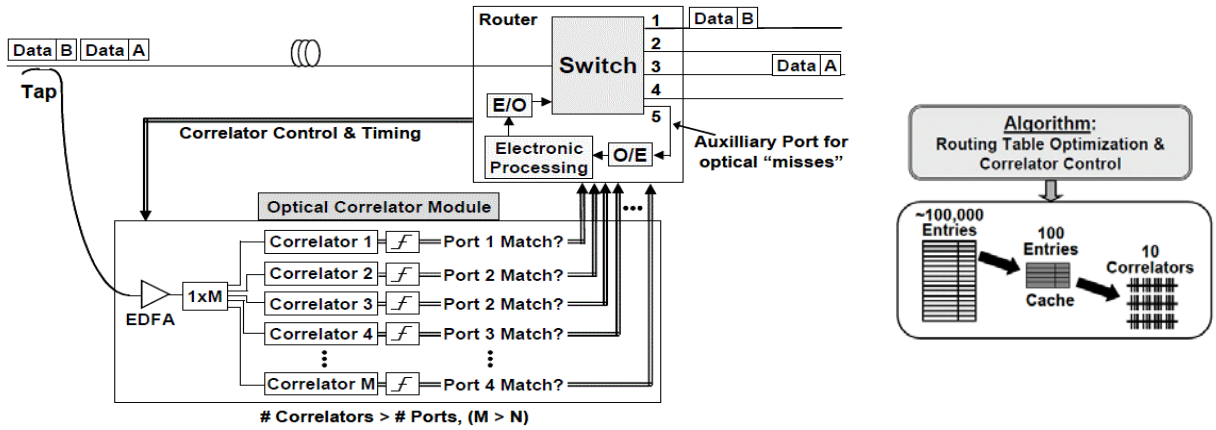

Figure 2. Optical header lookup as a pattern matching operation (left); distilling full routing to 10 correlators (right).

Those entries indicate the egress port and next-hop link layer address for each matched destination address[1]. For networks whose links are point-to-point, as we assume, the next-hop link layer address may be absent, thus the resulting table can often be compressed by considering entries to the same egress port. Analysis of the table in this manner frequently yields a very small number of patterns (*e.g.*, <10) of a very small number (*e.g.*, 5–8) of 0/1 (non-don't care) bits and this pattern might not start with the highest bit and its 0/1 values may not be adjacent as with the original IP forwarding table patterns[14,15,16]. The resulting set of patterns may be more feasible to implement as a small set of correlators that each examine a small number of bits. Such a compressed table may not match 100% of incoming packets, but an optical router can be coupled with a conventional electronic router in parallel, the latter to handle packets not matching this small number of compressed optimized patterns (Figure 2)[17].

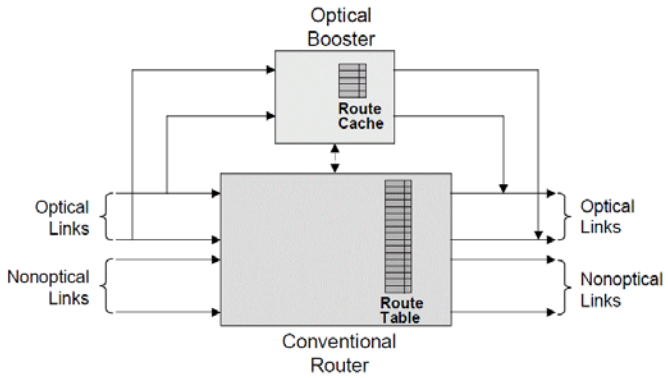

Figure 3. A fast optical booster that runs in parallel with a conventional (slow, electronic) router; packets matching the booster cache are discarded by the conventional router. Nearly all packets are processed on the fast optical path, but a few – whose destination addresses are not cached – must be forwarded on the slow electronic path.

We implemented a configurable header correlator consistent with this approach and measured its performance using 10Gb/s NRZ encoded packet headers (Figure 4). The data rate was slower than desired due to the limitations of available equipment; faster data rates can be easier to implement due to the smaller size of the correlator device. Our correlators were able to match only "1" values in the NRZ encoded data, so we needed to match the "1" values and "0" values separately (don't care values were ignored and not matched). The "1" values were matched against the incoming data, and the "0" values were matched against the complement of the incoming data. Threshold detectors combined the individual bit match signals to determine when the entire pattern matched and then the correlator outputs were combined to control a 2×2 optical switch.

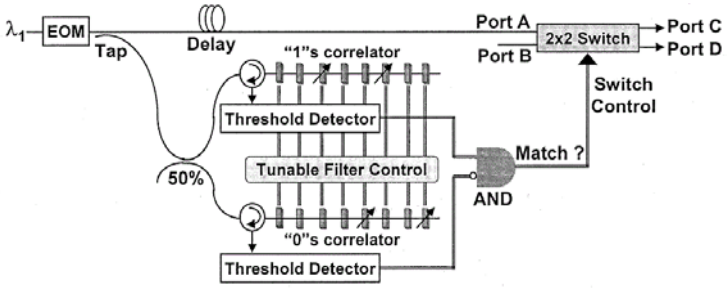

Figure 4. Optical header lookup as a pattern matching operation, combining 1's match (top) and 0's match (bottom).

The correlators were implemented using 0.5cm micro-heaters deposited on 1cm centers on a Fiber Bragg grating (FBG) (Figure 5, left), where the micro-heaters could be controlled to vary the reflectivity (Figure 5, right). The correlators were tuned to match two different header patterns (1010 and 1000), and the system successfully switched headers based on their value (Figure 6). Packet payloads would follow the same path as the header by latching the output of the correlator, either electronically or optically, until the beginning of the next packet. A delay in the data path allows the header match to be completed in time to configure the 2×2 switch to affect the corresponding packet header (Figure 4).

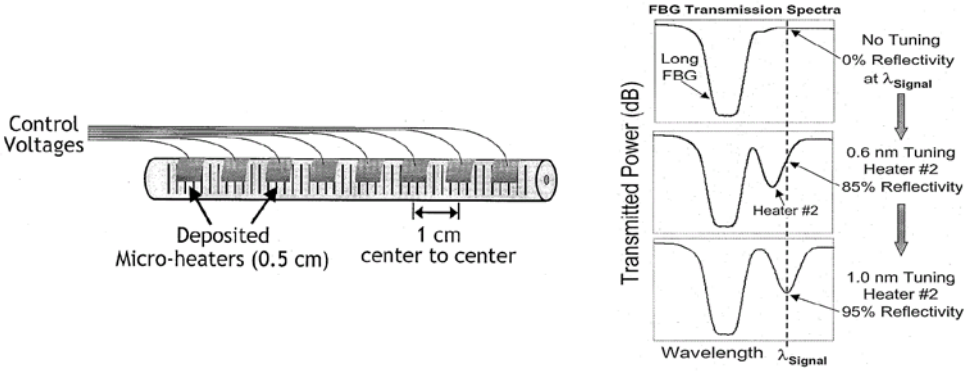

Figure 5. Correlator implemented as sequence of micro-heaters (left); heater tuning impact on transmitted power (right).

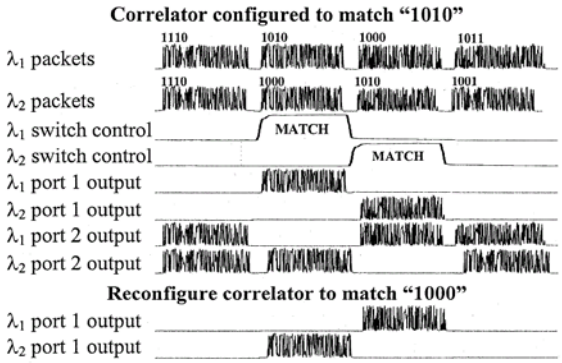

Figure 6. Experimental optical header correlator results, showing actual match signals and ability to reconfigure.

4. OPTICAL HOPCOUNT DECREMENT

Hopcounts are represented as 8 bit unsigned numbers in both IPv4 and IPv6[8,9]. The value must be decremented by at least one at each router traversed by a packet, but in most cases it is decremented by exactly one. If the value reaches zero, the packet is silently discarded (Figure 7).

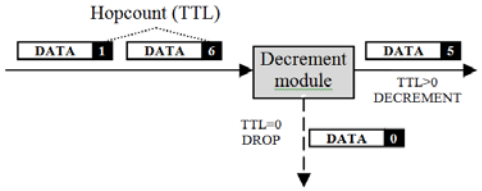

Figure 7. Hopcount effect on processed packets, showing decrement and drop if zero.

Full, parallel implementations of 8-bit unsigned arithmetic would be too complex to implement using digital optical processing. Fortunately, a simple serial algorithm can perform unsigned decrement on values of any bit length. Figure 8 demonstrates this algorithm (left), which starts with the low-order bit, inverts the '0' bits until and including the first '0' bit, then copies the remaining bits[18]. The equivalent schematic (Figure 8, right) requires only three simple elements – a bit inverter, a 2×2 switch, and a set-reset flip-flop (S/R FF). An S/R FF has two inputs, Set and Reset, and one output "Q"; the inputs and outputs begin in the '0' state. When the Set input changes to a '1', the output changes to a '1' and stays there – regardless of Set input changes – until the Reset changes to a '1' (which resets the output to a '0').

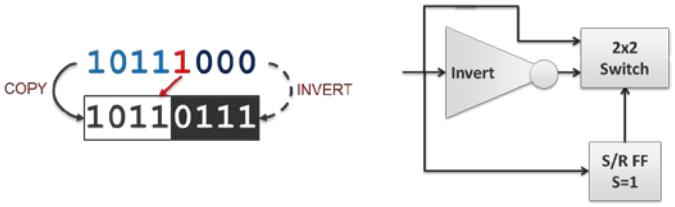

Figure 8. Hopcount decrement as a serial process (left, color online), combining inversion and copy (right).

We implemented the schematic in Figure 8, right, as shown in Figure 9, left[18]. The data signal was 10Gb/s NRZ, the same as for forwarding lookup as described in Section 3, again due to limitations of available equipment. Data inversion was accomplished using a semiconductor optical amplifier (SOA) powered by a CW pump and an optical circulator to create the conjugate of the NRZ signal. A pair of periodically poled lithium niobate (PPLN) waveguides creates the optically controlled 2×2 switch. The S/R FF is emulated using an electronic device, whose output and output complement drive modulators that concurrently enable and disable the pair of PPLNs. The experiment trace (Figure 8, right) shows the two PPLNs driven in complementary fashion from the low-order bit of the hopcount up to and including the first '1' to invert the data input, then swapping configurations to copy the data input to the output. The 1×2 switch separates dropped packets (when a '1' is not seen) from correctly decremented packets.

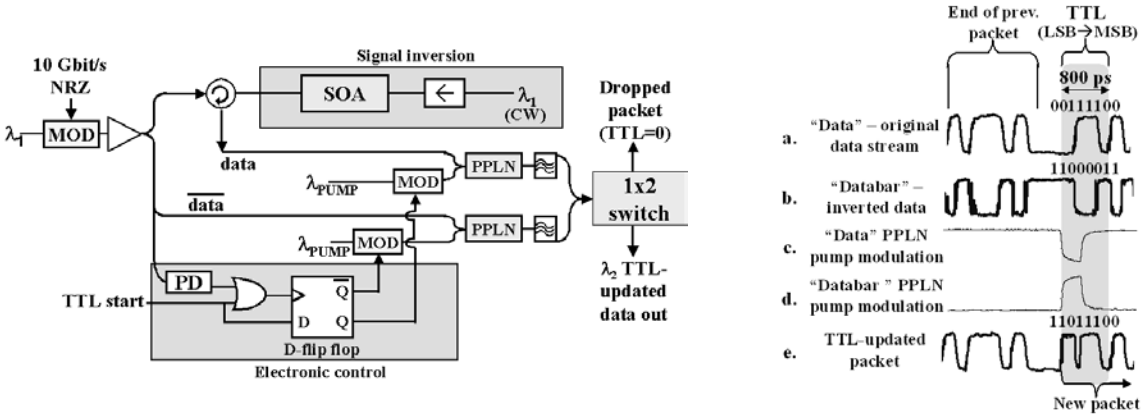

Figure 9. Optical hopcount decrement implementation, using an electronic S/R FF (using an electronic data or "D" FF).

The most significant deficiency of the experiment is that an electronic S/R FF was used to emulate an optical device. Optical S/R FFs have been developed[19], but currently represent an additional level of implementation complexity. We do we expect to incorporate them in future implementations.

5. OPTICAL CHECKSUM

Routers processing IPv4 packets are required to validate the IP checksum and to update its value to accommodate header field changes (*e.g.*, hopcount decrement). Although the shift to IPv6 is already underway, it is unclear when (if ever) the shift will affect most of Internet traffic. Further, the Internet checksum used as an IP header checksum is also used in other protocols such as TCP and UDP, and it needs to be updated if the packet payload or certain header fields are changed, such as for network address translation. Even if optical routers never process IPv4 packets and never update TCP or UDP checksums, optical checksums could be useful for link error checking, so exploring the IP checksum provides an opportunity to evaluate the potential to convert a simple checksum calculation to optical processing.

The IP checksum is a one's complement 16-bit sum of the 16-bit words of the IP header[11]. It is commonly implemented using commonly available two's complement adders, in which the high-order carry-out is wrapped around and used as the low-order carry-in. When implemented in hardware, this carry wraparound results in a symmetric device, *vs*. the asymmetry of a two's complement system in which carries propagate from low to high bit only.

Our team previously developed a 1.26Gb/s IP checksum using a simple programmable logic device (PLD)[20]. The device was composed of four 4-bit full adders, whose carries and data are cascaded in a cycle (Figure 10, left). For an optical version, we propose using a single full adder with 16 bits of recirculation delay for both the data and carry (Figure 10, right). This variant may also require data regeneration ('R' in the figure), depending on the number of 16 bit words being summed. For most IPv4 packets, the header is 20 bytes long, requiring 10 16-bit words, where the partial sums and carries cycle 10 times. Our team has developed a proprietary variant of this design that requires only four cycles to reduce the signal degradation that occurs during each cycle.

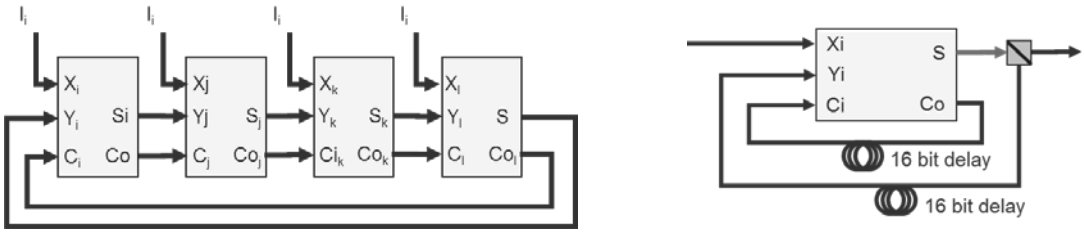

Figure 10. A single full adder implements the Internet checksum, using delayed recirculation with regeneration.

Implementation of this design requires the development of an optical one-bit full adder. Other teams have developed such a device for binary RZ encoded data[21], and we are currently planning on implementing the IP checksum based on existing binary adders.

6. OPTICAL MULTIPLEXING

The final phase of router packet processing is multiplexing, where packets whose destinations have been determined, hopcounts have been decremented, and checksums have been updated are switched towards the output port indicated byh the forwarding lookup. The basic high-level structure of a router includes input and output stages that correspond to input and output links, connected by a switching fabric (Figure 11, left). One implementation variation of that structure adds a demultiplexer to each input stage and a multiplexer to each output stage, so that the switching fabric is replaced by a fully connected internal mesh (Figure 11, center). In this architecture, output port contention is handled in the output multiplexer, where each output port's multiplexing function is independent (Figure 11, right).

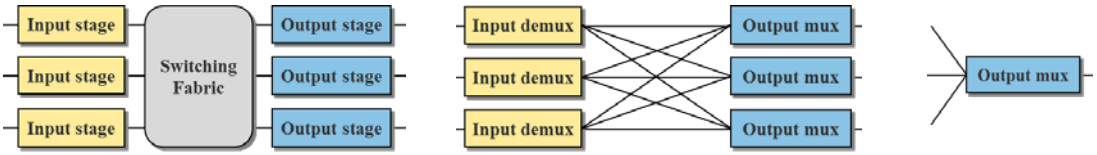

Figure 11. Basic structure of a packet switch (left), simplified to a demux-mux (center), and a separate output mux (right).

We chose this architecture for our optical router for two reasons. First, it lets us focus on the multiplexing operation of each output independently, rather than considering that an internal function of the switching fabric. Second, the output multiplexer itself is useful as a traffic aggregator. It does result in a potentially less scalable design, where N inputs and outputs (assuming a symmetric capacity configuration) require N^2 internal bandwidth, and N output multiplexers each with N inputs, *i.e.*, N^2 merging elements. Other more scalable designs can reduce this to N internal bandwidth and $N*log^2N$ merging elements, but can require more complex packet merging functions.

In the demux-mux architecture, we can examine packets destined to each output port independently[22]. Figure 12, left, shows a set of packets arriving at each of four input ports, shaded to represent their desired output port (based on lookup in the forwarding table). The numbers inside each packet indicate its length (in arbitrary units). Figure 12, right, shows only those packets destined for output port #3; in the demux-mux design, these are the set of packets that the output multiplexer needs to coordinate.

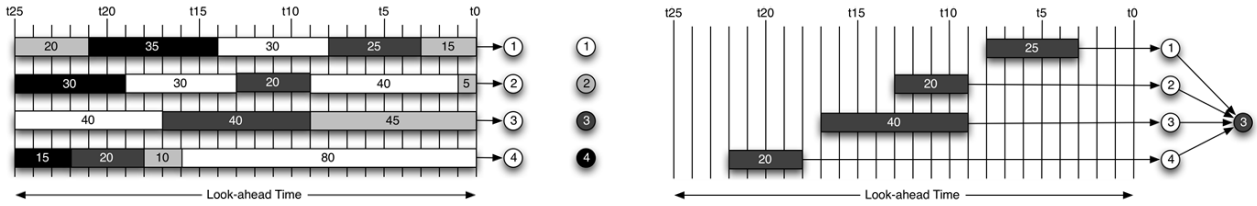

Figure 12. Length of the sequence of mutually overlapping packets (the contention set) for various traffic distributions.

Note that some packets overlap (the one of length 20 and the one of length 40, both starting at time *t9*), whereas other packets do not overlap with that set. We call the set of packets that overlap the *contention set*. We measured the size of this set for generated traffic based on Internet characteristics[23]. Figure 13 shows various types of such traffic and the cumulative probability of a set having a given number of packets; for most distributions, most of the contention sets include 10 or fewer packets. This suggests that the output multiplexer need not coordinate arbitrarily long sequences of overlapping packets, but that natural gaps occur sufficient that only a few packets need to be scheduled at a time. This presents the opportunity for batch processing, which could enable a slower electronic scheduler to manage groups of packets. We call this batch scheduling interval the *lookahead time*.

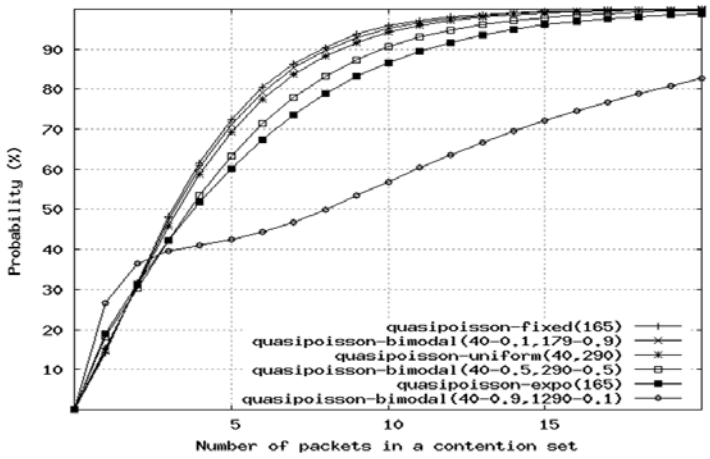

Figure 13. Length of the sequence of mutually overlapping packets (the contention set) for various traffic distributions.

First, it is useful to determine the limits of the effect of such batch processing. Within each batch, only a subset of packets may succeed in being multiplexed while others may need to be dropped, *e.g.*, if the average output link capacity is exceeded for a long enough period of time. We performed extensive simulations to determine the difference between arbitrary (*e.g.*, random) drop of contending packets *vs.* intelligent drop, the latter where the smallest number of the shortest packets is dropped[23]. Our simulation configuration considered a 32×32 switch under various input loads from 10% to 100% per port activity and a variety of traffic patterns from Poisson (emulating highly multiplexed traffic, as is typical close to the core) to Pareto (emulating highly bursty traffic, as is typical closer to the edge). The simulator exhaustively considered every possible combination of packet drops, looking arbitrarily far into the incoming packet

stream, determining the best choice in every case. We called this the "precognition switch", because it performs as if it knows what's coming. Drops alone were shown to be insufficient, achieving only 65% efficiency, where even the most intelligent precognition approach was only 3-5% more efficient than arbitrary random drops (60%).

Efficient multiplexing thus requires time-shifting some of the packets, to better utilize the gaps between arrivals to compensate for overlapping arrivals[22]. Electronic packet switches accomplish this time-shift using queues. Queues are first-in, first-out buffering devices that help time-shift and sometimes reorder groups of packets. Electronic queues provide first-in, first-out behavior, but are typically implemented less like waiting lines (*e.g.*, ticket queues at the movie theater) and more like parking lots. Arriving packets are placed into determined locations based on the packet type and arrival time. They remain parked in this parking lot until they are scheduled for release, in a process that is more like a valet retrieving the desired car than people buying tickets (Figure 14, left). Common mechanisms for electronic queuing place these parking lots at the input stage, called Virtual Output Queuing (VOQ), where packets are parked in separate groups destined for each output, such that packets are released to the output stage with no overlap, so the output passively merges the result. The two common electronic VOQ algorithms are iSLIP[24] and Parallel Iterative Matching (PIM)[25]; these provide the metrics against which we can evaluate optical queuing.

Although electronic packets can be parked in random access memory (RAM), optical packets cannot be 'stopped' in this manner. Light can be slowed down, but not stopped. As a result, we consider a different approach to queuing, in which packets merge much as do cars on merging roads. Packets come together where they (or an external party) detect potential contention, and one car 'hits the brakes' so the cars merge passively without slowing down (Figure 14, right).

Figure 14. Electronic queues (left, as in VOQ, color online) use random-access storage, where incoming packets are "parked" until removed (*e.g.*, by a valet). Optical queues (right, color online) can only slow down to avoid collisions, more like highway merging.

The merging approach, based on gradual and selective slowdown, can be accomplished using a switched delay line (SDL) (Figure 15, left inset). An SDL is a sequence of switches that select between a fast and a slow path, which can be implemented in optics using waveguides with different refractive indices or of different length. Our approach assumes a small number (*e.g.*, 10) of switches in series, with the same delay possible at each switch. This sequence of SDLs is sometimes known as a 'staggered delay line'[26], which behaves similar to a 'conveyor queue'[27].

Our output multiplexer consists of two sets delay lines – one fixed followed by one switched – of the same length (Figure 15, left)[22,28,29]. The fixed delay provides access to the headers of a batch of incoming packets, one line from each of the input demultiplexers. While in this *lookahead* region, packet headers are sent to an electronic control that determines which packets to slow down and which to drop. The control configures the SDLs in the *shift* region, and the packets are time-shifted to avoid contention. Each input line is allocated its own *lookahead* and *shift* regions, which together form a linear delay, so packets from a given input are never reordered. The resulting set of shifted packets is combined passively onto the output link. Figure 15, right, shows this behavior for a set of incoming packets while in the *lookahead* region (top) and after being aligned in the *shift* region (bottom). We call this the 'Tetris' switch, named after the popular video game, because it is similar to the packing of blocks in the game when viewed horizontally[28].

Batch processing of sets of contending packets in the *lookahead* region enables our approach to support a variety of prioritization algorithms, which "pack" the output port similar to classical bin packing algorithms. This prioritization also supports different types of packet quality-of-service (QoS), in which header priority tags or other properties (address, destination, protocol type) can determine which packets are shifted first, which should be dropped or not dropped, and which can wait for available capacity (*e.g.*, network control messages, traffic that exceeds assigned capacity, packets assigned to guaranteed or reserved flows, and background capacity, respectively). For our measurements, we use a default algorithm in which packets are scheduled in the order they arrive, attempting to retain the relative arrival order.

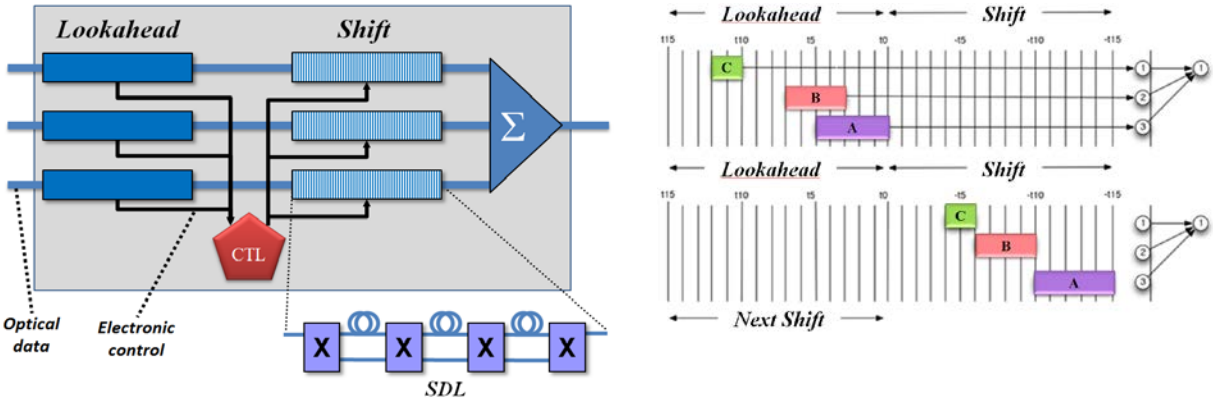

Figure 15. Our proposed packet merging architecture (left, color online), composed of a lookahead region, a configurable switched delay line shift region, and a passive optical coupler. Packets in the lookahead region are batch scheduled with shifts in the shift region configured to avoid overlap (right, color online).

We measured the effectiveness of this configuration in a 32×32 switch using simulations based on traffic properties closer to the Internet edge, where we believe packet switching will have the most significant impact[22]. Such traffic is more bursty (Pareto) and thus more likely to benefit from statistical multiplexing gains. We measured the effective throughput of the switch under a variety of loads from 10% to 100%, using *lookahead* and *shift* buffer regions of varying sizes from 1500 bytes to 15000 bytes, equivalent to 1-10 large IP packets. We assume the *lookahead* and *shift* region buffer sizes match, largely to simplify the number of variables considered.

Our results at 100% load are shown in Figure 16, both as absolute measures (left) and relative to VOQ iSLIP (right). We compared two electronic VOQ algorithms, iSLIP[24] and PIM[25], forward-shift, backward-shift, and no queuing (unbuffered). In backward shift (inspired by an NICT architecture[30]), the SDL default is to select the fast path and packets are slowed down (take the longer path) when desired. In our proposed forward shift, the SDL default path is to select the slow path and packets are effectively sped-up (take the shorter path) when desired[28,29]. The results demonstrate that optical SDLs can achieve up to 95% of the efficiency of electronic VOQ with buffers as short as 6000 bytes (the equivalent of 4 large IP packets)[22]; this result agrees with the new understanding that efficient Internet routing can be achieved with a comparatively small amount of buffering, rather than the conventional wisdom of a full round-trip bandwidth-delay product's worth[31]. The results also clearly demonstrate the difference between backward and forward shift approaches, where forward shift is much more efficient especially for smaller buffer regions.

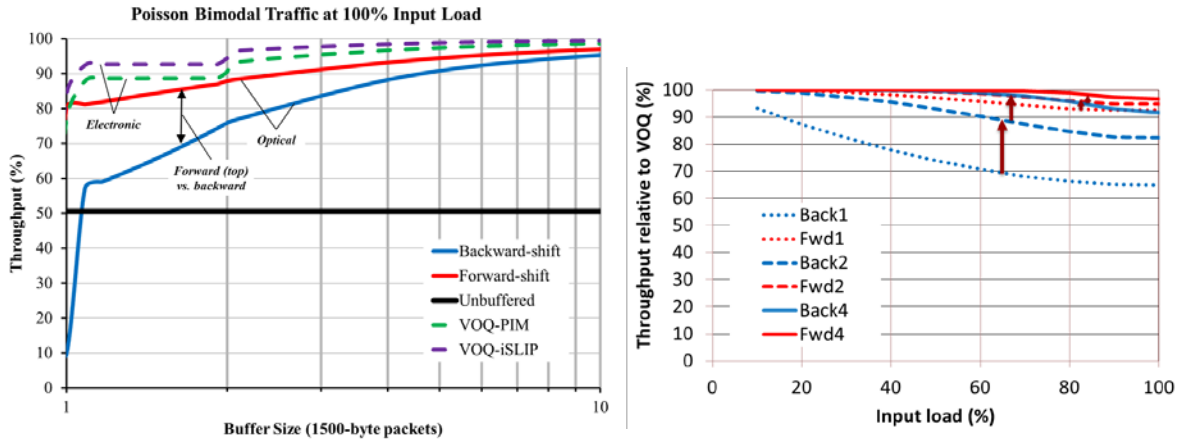

Figure 16. Forward shift (top solid curve, color online) achieves throughput efficiency approaching that of electronic routers and can be over 93% efficient (95% as efficient as electronic VOQ show as dashed curves, right curve, color online) with as few as 6000 bytes of SDL buffering. Forward shift is much more effective than backward shift for most nontrivial buffer sizes (over 1070 bytes).

Figure 16, right, shows the difference between backward and forward shifting under input loads varying from 10-100% compared to VOQ iSLIP[22]. It compares 1, 2, and 4 large IP packets' worth of buffering and shows that forward shifting is much more efficient under all loads than backward shifting assuming the same amount of buffering. It also shows that forward shift achieves at least 95% of the efficiency of VOQ iSLIP under all loads.

7. RELATED WORK

Two large surveys focus on high-performance router design[32] and optical routers[26]. High-performance router designs tend to focus on increased electronic parallelism, which does not support all-optical packet processing needed when communication speeds and distances necessitate optical transmission modulation formats. Many optical approaches focus on the expectation of reduced energy, where some attempt to refute that claim[26]. Our approach makes no assertions as to the potential for energy savings for optical approach; we assume that the primary benefit of an optical packet switch is avoiding the complexity and cost inherent in conversion from necessary optical transmission formats to electronic formats for switching.

A number of groups have attempted to develop optical packet switches. Most designs fall into two categories: recirculation and burst switching. Recirculation systems handle output port collision by sending colliding packets to a set of fiber delay lines (FDLs) of different delay, where those packets are returned to the input for a new attempt at switching[33,34,35,36]. Recirculation causes undesirable packet reordering. Some switches handle only fixed-sized packets that arrive aligned (synchronously)[34], whereas others recirculate variable-length packets into FDLs of different length[36] or have input ports with one FDL staging area for each output port[37]. Our design shifts packets while in transit and each input is assigned its own SDL for each output, so it never recirculates them, thus avoiding reordering.

Optical Burst Switching (OBS)[5] and Terabit Burst Switching (TBS)[6] schedule groups of adjacent packets with the same destination as if they were short circuits configured while the packets are in flight. They assume both that these groups can be created at the network edge, typically using large electronic buffers, and that these brief-duration circuits avoid contention while in transit, sometimes by scheduled coordination of burst emission. We consider the delay introduced in gathering sufficient packets for a burst and the global coordination required to avoid burst contention untenable approaches to handling dynamic Internet traffic patterns.

Some groups have recently revived more use of more conventional optical circuits as a complement to packet switching. This approach can be very effective when traffic patterns are known sufficiently in advance, both - enough to mask large circuit setup delays to offset the dead time during circuit reconfiguration and involving enough data to warrant a dedicated optical channel. This is rarely the case for Internet traffic in general, but can be used for provisioning links between Internet routers in the Internet core (where traffic is large enough), especially over long timescales. Known traffic patterns are also a hallmark of many data center applications, where hybrid use of optical circuits can complement packet switching[7].

Some groups have explored so-called 'subwavelength' fast optical switching to handle packets. These often focus on the switching mechanism itself, especially using wavelength conversion mechanisms. In most such designs, the header is assumed available out-of-band concurrent with the packet and is handled electronically[38]. Our approach uses an all-optical data plane, including optical packet processing, and assumes a focus on switching rather than wavelength partitioning.

The closest optical packet switch architecture to our approach is the NICT design[30]. Their architecture, like ours, supports variable-length packets without converting them internally to fixed-length packets. They too avoid the issue of wavelength conversion, focusing on a single-channel per port approach. Their design uses a fixed set of FDLs of different lengths to provide different delays, rather than our SDL approach. Our backward-shift configuration was inspired by their use of no delay as default and to add delay only when needed. Our forward-shift approach was inspired by the Tetris game, specifically the use of the "space bar" to "drop" a block into place rather than waiting for it to drop more slowly

8. DISCUSSION

This exploration demonstrates the need for an optical data plane for Internet routers and the viability of optical processing. For each component, including forwarding, hopcount decrement, checksum computation, and multiplexing, we have demonstrated either through experiment or simulation the feasibility of a relatively simple approach.

Practical optical header matching will depend on the ability to efficiently compress the forwarding table into a very small number of cached entries, each of which considers a fairly small number of bit values, and there is evidence this may be feasible[14,15,16]. The alternative would require both very large scale integration to support larger, more complete forwarding tables, as well as the ability to efficiently optically replicate packet headers to support a large number of concurrent correlators.

Internet hopcount decrement is the closest to already being practical in optics. A more complete solution would require integration of an optical S/R FF, but the current approach is otherwise sufficiently simple and direct. It would additionally benefit from being extended to support multibit encodings, such as N-PSK or QAM, to natively support common high speed, long-distance transmission formats. This extension changes the inverter to a "-1" transform ('-1' is the same as inversion for binary values, but '-1' is more complex than conjugate generation) and changing the S/R FF from a binary device (Figure 17, left) to one that treats the Set input as either '0' or nonzero (Figure 17, right).

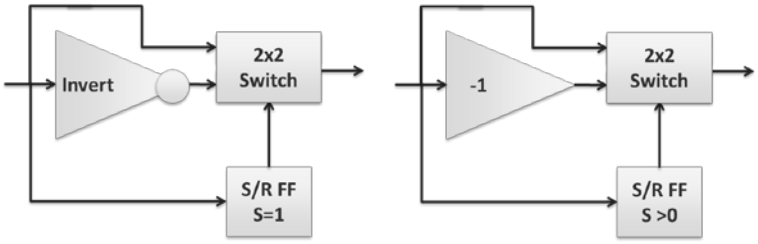

Figure 17. Extending hopcount decrement from binary (left) to multibit data (right).

The Internet checksum may be challenging to implement in optics, but shows the promise of support for other checksum approaches, especially those that might be more directly tailored to optical processing. The two impediments for checksum development are signal integrity and extension to multibit formats. The checksum currently requires the partial carry and partial sum to be recirculated *K* times for a *K*-word input (for *K* recirculations, with one final recirculation to allow the carry to propagate through the final sum). This may be practical for the 20-byte default IPv4 header, but would be difficult to extend to support packet rewriting that requires updating the transport layer (TCP, UDP) checksum over the full packet body. We are in the process of developing a proprietary mechanism that effectively folds the feedback loop, reducing the number of recirculations from O(*K*) to O(*log(K)*).

The checksum also needs to be extended from binary to multibit encodings. Although binary half- and full-adders have been demonstrated[21], multibit adders, *e.g.*, for N-PSK encodings, are currently limited to modulus arithmetic[39]. Extending these to a full-adder requires concurrent carry-out generation, resulting in a half-adder, and composition of multibit half-adders to create a multibit full adder. Our team has explored a potential approach to N-PSK carry generation, in which incoming signal phases are halved, added, then squeezed and shifted to create the corresponding carry-out value (Figure 18, left to right)[40]. Two binary half-adders can be combined, joining the carry-outs in an OR gate (Figure 19, left), but it takes three multibit half-adders to create a multibit full-adder (Figure 19, right).

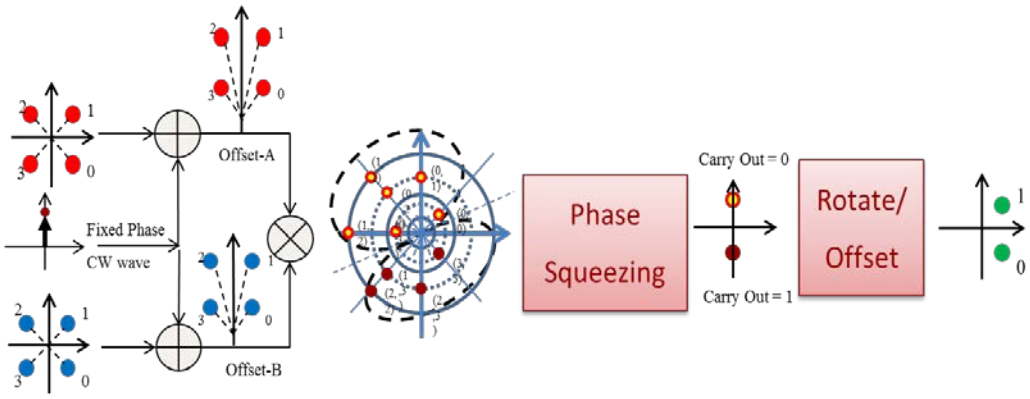

Figure 18. Proposed approach to compute carry-out of a PSK-encoded signal using phase halving and phase squeezing.

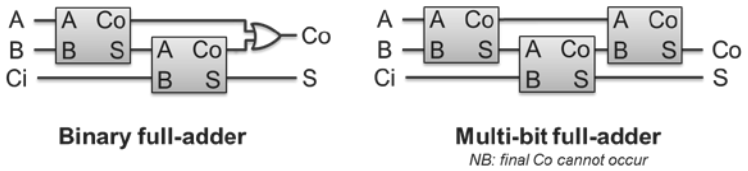

Figure 19. A binary full-adder from half-adders, using an additional OR gate (left); the equivalent for multibit data (right).

Sufficiently efficient multiplexing can be implemented in optics using as few as 6000 bytes of optical delay. At low speeds, *e.g.*, 10Gb/s RZ encoding, this can be unwieldy (2.2km), but as symbol rates and encoding density increase, this decreases quickly. At 40Gsymbols/s using 8-PSK, 6000 bytes of delay is only 121m; at 1Tsymbols/s the same delay requires only 5m. It will be challenging to implement this delay in 10 equal units with as many low-loss switches in series. The switches are reconfigured each batch, so existing ~1ns active device switching times should be sufficient.

The major challenges are integration and regeneration. A practical optical data plane would need to be implemented as a very small number of integrated devices, each of which would require dozens of active components – far beyond what is currently practical, but definitely what is expected in the near term. Regeneration is required to compose these separate functions into a practical processing pipeline, to avoid signal degradation that would increase the data error rate. Our team, in conjunction with Fujitsu, has also been developing approaches to phase regeneration that we hope might make such composition practical for N-PSK signals in the future[41].

The relationship between transmission formats and processing complexity also affects the feasibility of this approach. Current transmission formats are optimized for robust transmission, but not necessarily for ease of digital processing. Our "Optical Turing Machine"[42] project is beginning to explore whether other multibit encoding formats might concurrently support high-speed transmission while more feasibly enabling digital optical packet processing functions.

9. CONCLUSIONS

We show that the transition to native optical IP packet processing is inevitable, driven by the need for increased transmission capacity and avoiding format conversion, even if it might not be motivated by power reduction compared to electronic implementations. We have demonstrated the potential for developing an Internet router with an optical data plane by showing the feasibility of each of its component steps. We previously validated implementations of forwarding lookup via header matching and hopcount decrement using 10Gb/s RZ optical packets. We show the design and feasibility of an optical Internet checksum based on existing optical full-adders, as well as optical packet multiplexing of Internet traffic based on forward-shift SDLs, both using simulations. We also show the extension of optical processing from binary to multibit encodings, to support even higher optical transmission formats. The composition of these components is a true optical Internet router, and its implementation depends more on advancement in optical integration and regeneration than on any inherent impediment of efficiently optically processing IP packets.

10. ACKNOWLEDGEMENTS

The authors would like to acknowledge the feedback provided by Mohammed Chitgarha at USC, Michal Lipson and her students at Cornell University, Christos Papadopoulos and his students at Colorado State University, and George Kesidis and his students at Penn State, all of whom endured early versions of presentations resulting in this work. This work is partly supported by NSF CIAN grant # 0812072. "Tetris" is a registered trademark of The Tetris Company, LLC.

REFERENCES

[1] Baker, F., (Ed.), "Requirements for IP Version 4 Routers," RFC 1812, ISSN 2070-1721 (1995).
[2] Rekhter, Y., Davie, B., Katz, D., Rosen, E., and Swallow, G., "Cisco Systems' Tag Switching Architecture Overview," RFC 2105, ISSN 2070-1721 (1997).
[3] Newman, P., Minshall, G., and Lyon, T., "IP Switching — ATM Under IP," IEEE/ACM Trans. Networking, 6(2), 117-129 (1998).
[4] Rosen, E., Vishwanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, ISSN 2070-1721 (2001).

[5] Qiao, C., and Yoo, M., "Optical burst switching (OBS) - a new paradigm for an optical Internet," J. High Speed Networks, 8(1), 69-84 (1999).
[6] Turner, J., "WDM Burst Switching for Petabit Data Networks," Proc. OFC, 2, 47-49 (2000).
[7] Porter, G., Strong, R., Farrington, N., Forencich, A., Sun, P-C., Rosing, R., Fainman, Y., Papen, G., and Vahdat, A., "Integrating microsecond circuit switching into the data center," Proc. ACM SIGCOMM, 447-458 (2013).
[8] Postel, J., (Ed.), "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, ISSN 2070-1721 (1981).
[9] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, ISSN 2070-1721 (1998).
[10] Touch, J., "Components developed for all-optical Internet router," SPIE Newsroom, DOI: 10.1117/2.1200809.1294 (2008).
[11] Braden, R., Borman, D., and Partridge, C., "Computing the Internet Checksum," RFC 1071, ISSN 2070-1721 (1988).
[12] Hauer, M., McGeehan, J., Kumar, S., Touch, J., Bannister, J., Lyons, E., Lin, C., Lau, A., Lee, H., Starodubov, D., and Willner, A., "Optically-Assisted Internet Routing Using Arrays of Novel Dynamically Reconfigurable FBG-Based Correlators," IEEE/OSA Journal of Lightwave Technology, Special Issue on Optical Networks, 21(11), 2765-2778 (2003).
[13] Bannister, J., Touch, J., Kamath, P., and Patel, A., "An Optical Booster for Internet Routers," Proc. Eighth International Conference on High Performance Computing, 339-413 (2001).
[14] Degernark, M., Brodnik, A., Carlsson, S., and Pink, P., "Small Forwarding Tables for Fast Routing Lookups," Proc. ACM Sigcomm, 3-14 (1997).
[15] Waldvogel, M., Varghese, G., Turner, J., and Plattner, B., "Scalable High Speed IP Lookups," Proc. ACM Sigcomm, 25-36 (1997).
[16] Talbot, B., Sherwood, T., and Lin, B., "IP Caching for Terabit Speed Routers," Proc. IEEE Globecom, 1565-1569 (1999).
[17] Bannister, J., Touch, J., Kamath, P., Patel, A., McGeehan, J., and Willner, A., "A Method to Forward Internet Packets Without Conversion from an Optical to an Electronic Format," U.S. Patent #7369766, (2008).
[18] McGeehan, J., Kumar, S., Gurkan, D., Motaghian Nezam, S., Bannister, J., Touch, J., and Willner A., "All-Optical Decrementing of a Packet's Time-To-Live (TTL) Field and Subsequent Dropping of a Zero-TTL Packet," IEEE/OSA Journal of Lightwave Technology, Special Issue on Optical Networks, 21(11), 2746-2752 (2003).
[19] Clavero, R., Ramos, F., Martinez, J., Marti, J., "All-Optical Flip-Flop Based on a Single SOA-MZI," IEEE Photonics Tech. Letters, 17(4), 843-845 (2005).
[20] Touch, J., and Parham, B., "Implementing the Internet Checksum in Hardware," RFC 1936, ISSN 2070-1721 (1996).
[21] Scaffardi, M., Ghelfi, P., Lazzeri, E., Poti, L., and Bogoni A., "Photonic Processing for Digital Comparison and Full Addition Based on Semiconductor Optical Amplifiers," IEEE J. Sel. Topics in Quantun Electronics, 14(3), 826-833 (2008).
[22] Touch, J., Suryaputra, S., Bannister, J., and Willner, A. "Optical packet switch using forward-shift switched-delay lines," Proc. OECC-PS, 1-2 (2013).
[23] Suryaputra, S., Touch, J., and Bannister, J., "The Case of a Precognition Optical Packet Switch," Proc. IEEE High-Speed Networks Workshop, 1-6 (2009).
[24] McKeown, N., and Anderson, T., "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches," Computer Networks and ISDN Systems, 30(24), 2309-2326 (1998).
[25] Ge, N., Hamdi, M., and Letaief, K., "Efficient scheduling of variable-length IP packets on high-speed switches," Proc. GLOBECOM, 2, 1407-1411 (1999).
[26] Tucker, R., "The Role of Optics and Electronics in High-Capacity Routers," J. Lightwave Tech., 24(12), 4655-4673 (2006).
[27] Coffman, E., Gelenbe, E., and Gilbert, E., "Analysis of a conveyor queue in a flexible manufacturing system," ACM SIGMETRICS Perform. Eval. Rev., 14(1), 204-223 (1986).
[28] Suryaputra, S., Touch, J., and Bannister, J., "The Tetris Switch," USC/ISI Tech. Report, ISI-TR-662 (2009).
[29] Touch, J., Bannister, J., and Suryaputra, S., "Packet Switch with Separate Look Ahead, Computation, and Shift Phases," U.S. Patent #8306047, (2012).
[30] Harai, H., and Murata, M., "High-speed buffer management for 40 Gb/s-based photonic packet switches" M. Trans. on Networking, 14(1), 191-204 (2006).

[31] Enachescu, M., Ganjali, Y., Goel, A., McKeown, N., and Roughgarden, T., "Routers with very small buffers" Proc. IEEE INFOCOM, (2006).
[32] Chao, H., "Next generation routers," Proc. IEEE, 90(9), 1518-1558 (2002).
[33] Singh, Y., Kushwaha, A., Bose, S., "Exact and approximate analytical modeling of an FLBM-based all-optical packet switch," IEEE/OSA J. Lightwave Tech, 21(3), 719-726 (2003).
[34] Haas, Z., "The 'Staggering Switch': An Electronically Controlled Optical Packet Switch," J. Lightwave Tech., 11(5/6), 925-926 (1993).
[35] Liew, S., Hu, G., and Chao, H-J., "Scheduling Algorithms for Shared Fiber-Delay-Line Optical Packet Switches - Part I: The Single-Stage Case," J. Lightwave Technology, 23(4), 1586 (2005).
[36] Hunter, D.K., Cornwell, W., Gilfedder, T., Franzen, A., and Andonovic, I., "SLOB: a switch with large optical buffers for packet switching," J. Lightwave Tech., 16(10), 1725-1736 (1998).
[37] Shiramizu, Y., Oda, J., and Goto, N., "All-optical autonomous first-in--first-out buffer managed with carrier sensing of output packets," Optical Engineering, 47(8), 085006-085008, (2008).
[38] Blumenthal, D., Olsson, B-E., Rossi, G., Dimmick, T., Rau, L., Masanovic, M., Lavrova, O., Doshi, R., Jerphagnon, O., Bowers, J., Kaman, V., Coldren, L., and Barton, J., "All-Optical Label Swapping Networks and Technologies," J. Lightwave Tech., 18(12), 2058-2075 (2000).
[39] Wang, J., Nuccio, S., Yang, J-Y., Wu, X., Bogoni, A., and Willner, A., "High-speed addition/subtraction/complement/doubling of quaternary numbers using optical nonlinearities and DQPSK signals," Optics Letters, 37(7), 1139-1141 (2012).
[40] Ziyadi, M., Khaleghi, S., Chitgarha, M., Touch, J., and Willner, A., "Digital Optical Computation," Poster at OIDA Datacenter Workshop, (2012).
[41] Yang, J-Y., Ziyadi, M., Aksaka, Y., Khaleghi, S., Chitagarha, M., Touch, J., and Sekiya, M., "Investigation of Polarization-Insensitive Phase Regeneration Using Polarization-Diversity Phase-Sensitive Amplifier," Proc. ECOC, 1-3 (2013).
[42] Touch, J., "Optical Turing Machine," (2013). http://www.isi.edu/otm