Simplified Layering and Flexible Bandwidth with TWIN

Indra Widjaja and Iraj Saniee
Bell Laboratories, Lucent Technologies
Murray Hill, NJ 07974, USA
{iwidjaja,iis}@research.bell-labs.com

ABSTRACT

This paper describes a novel network architecture with simplified layering, called Time-domain Wavelength Interleaved Networking (TWIN), that scales end-to-end bandwidth granularity flexibly up to the wavelength capacity. In TWIN, all packet and complex processing functions are pushed to the network edge such that the network core only has to deal with an optical forwarding layer. Furthermore, by avoiding fast optical switching and optical buffering in the core through scheduling fast-tunable lasers and buffering packets at the edge, TWIN effectively makes the network act like a switch. We examine distributed network scheduling for this architecture and show its performance via analysis and simulation. We also explore other research issues that are unique in TWIN.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*packet-switching networks, circuit-switching networks*

General Terms

Design, Performance

Keywords

Network, switch, simplified layering, bandwidth granularity, network scheduling

1. INTRODUCTION

Service providers typically face a multitude of network technologies in their infrastructures, for example, TDM networks for the transport of voice traffic, and IP, Frame Relay and ATM networks for data traffic. There is a growing interest in the community to reduce capital and operational costs by eliminating network duplication and making the infrastructure simpler. Internet backbone traffic volume already exceeded long-distance voice volume by a factor of two or more at the end of 2003 [1] and continues to grow at a higher rate. As a result, packet-based technology such as IP/MPLS is increasingly regarded as the "converged" network capable of supporting multiple types of services on a common platform. Work on various adaptation and control protocols (e.g., see [2] for Ethernet over IP/MPLS, [3] for ATM over IP/MPLS and [4] for frame relay over IP/MPLS)) is currently under way to enable convergence at the packet layer.

Converged networks have different interpretations in different contexts. For the purpose of this paper, we define a converged network as one that is capable of supporting the following services: (1) various types of asynchronous/bursty traffic such as IP, Ethernet, Frame Relay and ATM), (2) various types of synchronous/TDM traffic such as T-1, T-3 and OC-3 leased lines, and (3) flexible bandwidth demands; e.g., from a few Mb/s to the wavelength capacity. It is well known that a packet network is suitable for carrying asynchronous traffic with arbitrary bandwidth. Although synchronous services can also be supported on a packet network by means of circuit emulation [5][6], such services are more cost-effectively provided by a circuit-switching network.

A packet network typically consists of edge/access routers that provide *external* interfaces to clients and core/backbone routers that provide high-speed forwarding functions for transit traffic *within* the network (for example, see Fig. 8 in [7] and Fig. 1 in [8]). In this paper, the network core refers to the part of the network that solely consists of core nodes (core routers) and the network edge to the part that solely consists of edge nodes (edge routers). The network core is traditionally constructed using multiple layers of network elements. For example, core routers in different points of presence (POPs) are traditionally interconnected through a circuit-switching (SONET) network which in turn may run over a wavelength-division-multiplexing (WDM) network, resulting in Packet over SONET over WDM. As the capacity of a router port has increased to very high speed, currently at 10-40 Gb/s, it appears increasingly attractive to eliminate SONET network elements and interconnect core routers directly through point-to-point WDM links, resulting in Packet over WDM.

Because the cost per bit of a high-speed router port is considerably higher than that of a SONET port[1] and much of the traffic (about 70% or so) through a core router port is for transit rather than for local drop, Sudipta *et al.* [9] quantify

SIGCOMM'04 Workshops, Aug. 30+Sept. 3, 2004, Portland, Oregon, USA.

[1]Based on representative industry pricing [9] and also the fact that the linecard of a typical transport switch is simpler than that of a packet switch [10].

that the capital cost for packet over WDM can be lowered if core router ports are bypassed by TDM pipes for transit traffic. From a different perspective, Molinero-Fernández *et al.* [10] argue for using packet switching at the network edge and circuit switching in the core. Unfortunately, these solutions require insertion of a SONET network below a packet network, which would unavoidably increase network complexity and operational cost[2]. Thus, we arrive at a dilemma when "Packet over Something" is used as a converged network. On the one hand, we would like to interconnect packet core routers directly to reduce network layering. On the other hand, because packet processing at each core router along the path is a costly proposition, we would like to bypass core router ports for transit traffic. But this inevitably results in increased network layering.

In this paper, we argue that a cost-effective converged network can be constructed by implementing a *passive* WDM layer in the network core. Such a network, called TWIN [11], is fundamentally different from current approaches where some type of processing (packet switching or TDM switching) occurs at each node. One difficulty with a passive WDM layer is that it becomes challenging to support flexible bandwidth demands, as required in a converged network, since wavelength capacity is often much higher than a typical end-to-end service demand between two edge nodes. For this reason, optical circuit switching that establishes end-to-end wavelengths between edge nodes is usually not cost-effective. Our key contribution bridges the gap between end-to-end demands and wavelength capacity *without* the need for fast optical switching and optical buffering, which are currently impractical, or OEO conversion, which results in multi-layering. We show that a streamlined converged network can be realized by exchanging fast switching in the network core with fast wavelength interchange at the network edge, thereby rendering *the network able to act effectively like a giant and distributed Birkhoff-von Neumann switch whose input/output ports are spread across the edge nodes and whose fabric is spread across the core nodes.* We show that making the network act like a switch has other advantages, including simpler scheduling and control.

2. TWIN - NETWORK AS A SWITCH

Our goal of designing a network that realizes a cost-effective converged infrastructure and acts like a switch is guided by the following key principles:

- *Thin-layered and simple core.* In a fully utilized network core, the per-unit bandwidth cost of an optical forwarding path is generally lower than that of a TDM forwarding path, which in turn is generally lower than that of a packet forwarding path. Thus, a thin-layered core in principle can only be realized cost effectively if the forwarding path stays in an optical domain. Otherwise, there is always an opportunity to bypass some transit ports by introducing another layer. The core further becomes simple if it only employs passive devices with no data-path processing to carry out forwarding functions for transit traffic. To emulate a Birkhoff-von Neumann switch [12], the core needs to behave like a simple crossbar fabric connecting different input/output ports; that is, data travels transparently in the core without buffering.

[2]The study in [9] and [10] does not consider operational cost.

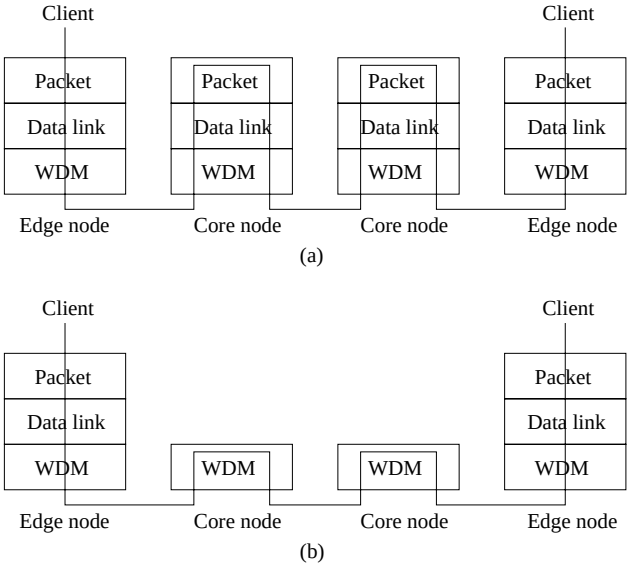

Figure 1: Protocol stacks for a converged network: (a) packet network, (b) TWIN. The edge node in TWIN acts like an input/output port of a Birkhoff-von Neumann switch while the core nodes, viewed together, act like a non-blocking fabric.

- *Intelligent edge.* Control and processing are consolidated at the edge nodes where OEO conversion and data processing have to take place irrespective of the network core. To emulate the input/output ports of a Birkhoff-von Neumann switch, each edge node contains buffers and transmits data to other edge nodes based on a given schedule.

Fig. 1 highlights the difference of the data paths in a packet network versus TWIN. Thin-layering in the core is key to a cost-effective architecture. Like a packet network, TWIN can also perform multiplexing and demultiplexing (switching) of traffic streams in a core node but without resorting to higher-layer processing above WDM. It is worth noting that while optical packet switching (e.g., [13][14]) and optical burst switching (e.g., [15][16]) can also keep the payload of each data unit at the WDM layer in network core, they have to resort to layer-3 processing for each header, and thus still require complex processing in the core. Furthermore, they also require optical nodes that are capable of fast reconfiguration on a per-packet or per-burst basis and optical buffering to store contending packets/bursts, both facing technological barriers.

Fig. 2 illustrates the basic TWIN systems architecture which consists of multiple POPs, each being served by *edge node(s)* and a *core node*. Multiple core nodes may also be employed for extra redundancy. An edge node has two types of interfaces: client interface and network interface. The client interface provides *external* connectivity services to clients (IP routers, Ethernet switches, SONET cross-connects, etc.) in different networks. With appropriate control plane, we envision an edge node that can offer a variety of connectivity services including layer-3 and layer-2 VPN services, access to public Internet and leased-line services. The network interface of an edge node is directly connected to a core node by a fiber. Transmission between edge nodes

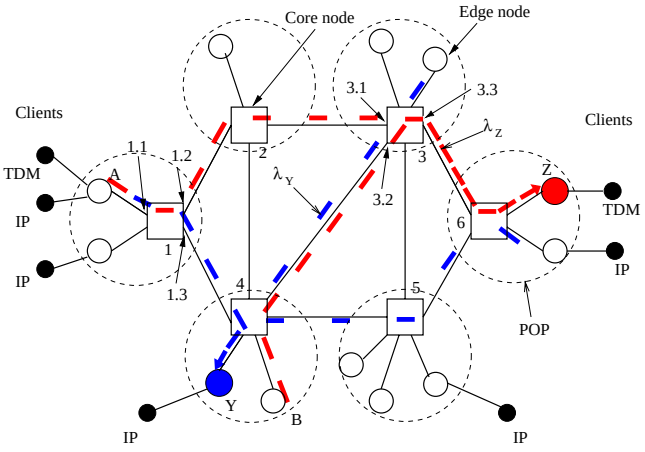

Figure 2: A TWIN architecture consists of edge nodes that provides external interfaces to clients and core nodes that forwards traffic in a transparent optical domain.

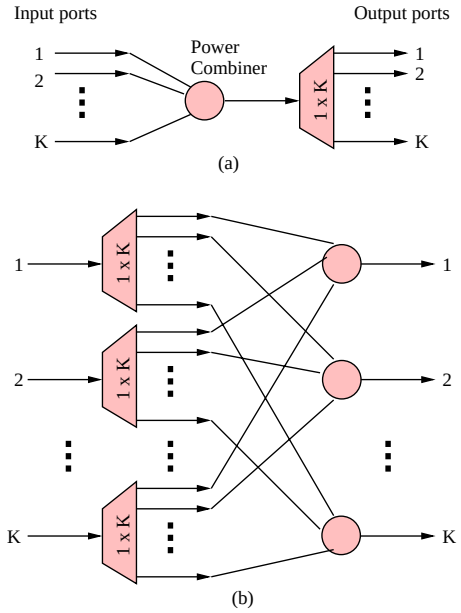

Figure 3: Structure of a core node: (a) routing based on incoming wavelength, (b) routing based on incoming wavelength and input port.

across the core nodes, which is purely optical, is enabled by a fast-tunable laser on the transmitting side of a network interface and a burst-mode receiver on the receiving side. The core nodes are also directly connected by fibers. The function of a core node is to *passively steer* optical signals from its inputs to its outputs based on either the wavelength of the signal, or the wavelength of the signal and the input the signal comes from. A core node is optically transparent and can be implemented simply by $1 \times K$ wavelength-selective switches [17][18] and power combiners (see Fig. 3). The purpose of a combiner is to enable merging of signals of the same wavelength to the same output.

An ingress edge node aggregates incoming protocol data units from various clients, encapsulates them into bursts[3], provides burst buffering, and uses a fast-tunable laser[4] for burst transmission. An egress edge node essentially performs the reverse functions by demodulating the received optical signal via a burst-mode receiver, decapsulating each burst into respective client protocol data units, and delivering them to appropriate clients. A natural question is: how do bursts sent by a source (ingress edge node) arrive at the intended destination (egress edge node)? In the simplest form, each destination j is assigned a unique address λ_j. When a source has data to send to destination j, the source will tune its laser intermittently to wavelength λ_j to transmit the data. The bandwidth allocated to a source-destination pair can be easily controlled by the duty cycle of the tunable laser. Since each core node can be configured to steer an incoming wavelength to its intended output, the route taken by an optical signal can be determined by appropriate configuration of the core nodes. Fig. 2 shows an example where routes forming multipoint-to-point trees rooted at destinations Y and Z have been provisioned. The configuration of the core nodes generally stays at very long time scales; for example, reconfiguration is needed when there is a failure and different backup routes are used.

[3]TWIN protocol data unit is called a burst, which is of duration of a few μs.

[4]A transmitter with a multifrequency laser can switch wavelengths in sub-nanoseconds [19].

To see how *fast* optical switching (multiplexing and demultiplexing) in the network core can be emulated through fast tuning of wavelength at the network edge, first consider the case where both sources A and B are transmitting bursts to destination Z. As shown in Fig. 2, these sources use the same wavelength (address) λ_Z on the bursts. It can also be seen that fast tuning of lasers at both sources A and B emulates fast multiplexing of bursts from two incoming ports 3.1 and 3.2 to the same outgoing port 3.3 at core node 3. Coordination between sources A and B is of course required if burst collision is to be prevented, and TWIN relies on scheduling to prevent collision (discussed in Sec. 3). Now consider the case when source A is simultaneously transmitting bursts to destination Y using another wavelength λ_Y. It can be seen that fast tuning between wavelengths λ_Y and λ_Z at source A emulates fast demultiplexing of bursts from an incoming input 1.1 to two separate outgoing outputs 1.2 and 1.3 at core node 1. The bandwidth allocated to (A, Y) and (A, Z) is flexible and can be easily controlled by the duty cycle of the corresponding wavelength. The only constraint is that the total bandwidth out of an ingress edge node and into an egress edge node is less than the capacity of a wavelength, which is also equivalent to the constraint in a Birkhoff-von Neumann switch. Further, destination trees ensure that if there is no collision at the destination, then there will be no collision anywhere else in the network core. This implies that the network core behaves like a *nonblocking* crossbar fabric, except with significant propagation delays. This property simplifies TWIN scheduling as it only needs to track the states of the edge nodes, but *not* the core nodes. In summary, *we can view a TWIN network as a giant and distributed Birkhoff-von Neumann switch whose input/output ports are spread across the edge nodes and whose fabric is spread across the core nodes.*

3. NETWORK SCHEDULING

One critical challenge in TWIN is to design scheduling to arbitrate burst transmission and maximize resource utilization. *Switch* scheduling for an input-queued crossbar switch with a negligible delay between input and output ports has been investigated extensively in the past (e.g., [20], [21], [22]). In switch scheduling, the scheduler knows the instantaneous state information of each input/output port, and can compute the schedule for each packet transmission. In contrast, TWIN has to deal with *network* scheduling where significant propagation delays may exist among sources and destinations. In TWIN, the scheduler can only observe a delayed version of the state information.

3.1 Centralized versus Distributed

In general, TWIN scheduler can be implemented in a centralized or distributed fashion. With centralized scheduling, there is one control point (a backup is needed for redundancy) that computes the schedules for all source-destination pairs based on the knowledge of the traffic demand matrix and the propagation delay between each node pair. Centralized scheduling is thus suitable for handling synchronous traffic, which results in quasi-static schedules. With distributed scheduling, a control point is located at each destination. Each control point independently computes the schedules for the sources that are transmitting to it. Centralized scheduling generally should be able to compute better schedules while distributed scheduling is inherently faster and thus more suitable for handling asynchronous traffic.

3.2 Scheduling Cycles

Because of propagation delays, a network scheduler in TWIN may have to recompute schedules at a much longer time scale than a traditional switch scheduler. To facilitate a scheduler that recomputes schedules infrequently, burst transmission in TWIN is organized by repetitive *scheduling cycles* of duration T_c per cycle. In general, recomputations are only needed when there are changes in bandwidth demands. A (scheduling) cycle consists of B *time slots*, each of duration T_s. Each time slot can be used to carry one burst. Adjacent bursts are inter-spaced by a *guard time* of duration T_g to take into account time-of-day synchronization errors and other implementation factors. Each burst consists of an overhead of duration T_o and a payload of duration $T_p = T_s - T_g - T_o$. An important part of the overhead supplies a bit-synchronization preamble so that a burst-mode receiver can perform frequency and phase synchronization to the transmitting bit stream. The above parameters are designed to meet a satisfactory trade-off among efficiency, bandwidth granularity, and buffer requirements.

In general, the scheduling cycles at all sources and destinations cannot be aligned since propagation delays can be arbitrary. Implementation, however, is simplified if the scheduling cycles are aligned at all destinations and all nodes are synchronized to a common time-of-day clock, which can be derived from a GPS-based timing reference (GPS-based synchronization can limit time-of-day errors to within 100 ns [23]). This allows a schedule for a given source-destination pair to be described simply by the slot number(s) the burst(s) transmitted by the source are to arrive at the destination in *subsequent cycles*. Once the source knows its schedule and propagation delay to the destination, the source can easily determine when to tune its laser to the destination. For synchronous traffic, this also implies that one schedule can be sufficient for the entire duration of the communications between a source and a destination since the slot number(s) occupied by the bursts can remain static in each cycle. For asynchronous traffic, a new schedule needs to be reissued when bandwidth demand changes. The frequency of reissuing schedules is limited by the time it takes to request bandwidth change and receive a new schedule.

3.3 Scheduling with Non-Zero Propagation Delays

To further illustrate the difference between switch scheduling and network scheduling, consider a TWIN network with N edge nodes, and let δ_{ij} denote the fixed propagation delay from source i to destination j $(i, j = 1, 2, \cdots, N)$. Suppose that the traffic demand of d_{ij} bursts is to be scheduled for each node pair (i, j). Let B^*_δ denote the minimum timespan of the schedule (the minimum number of time slots needed to meet all demands) with propagation delay matrix $\{\delta\}$. When $\{\delta\} = \{0\}$, as in switch scheduling, the minimum timespan is given by

$$B^*_0 = \max\{\max_i \sum_j d_{ij}, \max_j \sum_i d_{ij}\}$$

Unfortunately, B^*_0 is generally unachievable when propagation delays are non-zero. Fig. 4 shows a simple example illustrating this point. Let us suppose that a total of four bursts need to be scheduled with $d_{13} = d_{14} = d_{23} = d_{24} = 1$. From the equation above, $B^*_0 = 2$. To verify, we first consider the case where all propagation delays are zero. The scheduler can use the following schedules (Fig. 4b): node 1 transmits to node 3 and node 2 transmits to node 4 in time slot 1, then node 1 transmits to node 4 and node 2 transmits to node 3 in time slot 2. Thus, two time slots are sufficient to schedule all bursts in this case. Now assume that the propagation delay between node 1 and node 3 is one time slot while the others are zero (in Fig. 4a, delay(1,3)=1). If node 1 transmits to node 3 in time slot 2, then the burst will arrive in time slot 3 and at least three time slots are required. Thus, suppose that node 1 transmits to node 3 in time slot 1 (Fig. 4c). The burst will then arrive at node 3 in time slot 2. In time slot 1, node 2 can also transmit to node 3, also arriving at node 3 in time slot 1. Next, node 1 transmits to node 4 in time slot 2. To prevent collision at node 4, node 2 has to wait and can only transmit to node 4 in time slot 3. It can be easily concluded that three time slots are necessary and sufficient to transmit all bursts, and thus $B^*_\delta \neq B^*_0$ in this case.

3.4 Distributed Scheduler

The design and performance of a centralized scheduler have been discussed in [24]. In this section, we explore an approach for distributed scheduling.

A possible approach is to have sources independently *request* for transmission bandwidth (say, in units of time slots per cycle) to destinations which subsequently perform independent scheduling and *grant* time slots. Request and grant messages may be transmitted out-of-band on a separate control channel, or may be piggybacked on data bursts if such a channel is available. The request-grant approach is well known in practice when there is only one destination as in Cable Modem (one CMTS) [25] or EPON (one OLT) [26] for example. However, when multiple independent destina-

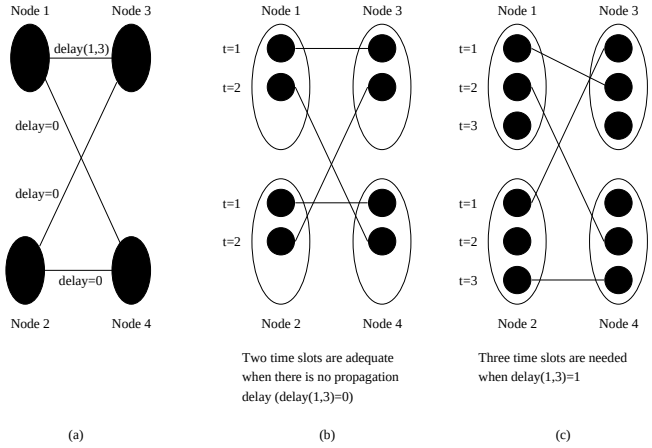

Figure 4: An example of scheduling with zero and non-zero propagation delays.

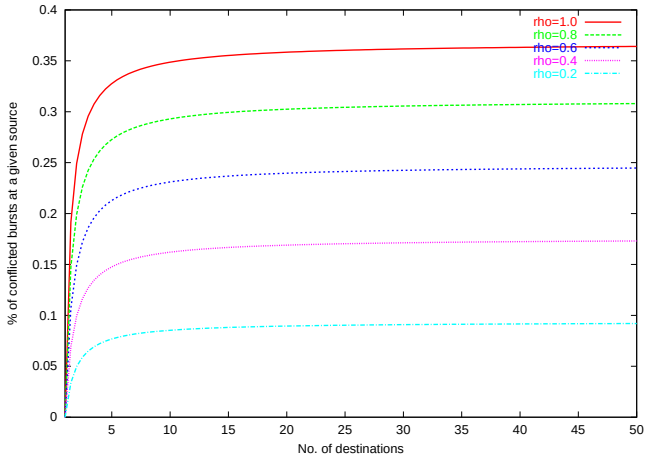

Figure 5: Average number of burst transmissions blocked versus number of sources.

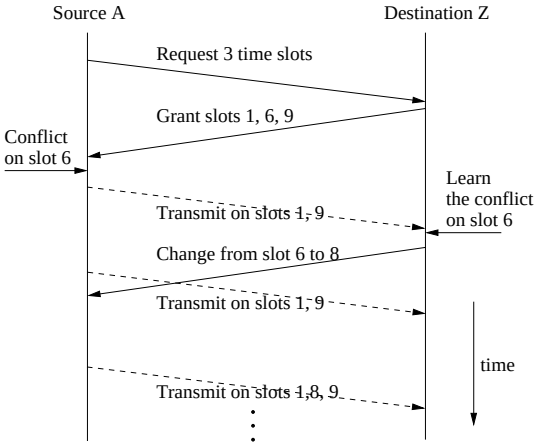

Figure 6: Example of the operation of the scheduler with learning.

tions are involved as in TWIN, a source may receive multiple grants that call for it to transmit simultaneously, thus creating a *conflict*. When there is a conflict, the source can only choose one destination in TWIN as there is only one tunable laser per source.

Unpredictable load. The moments of the number of transmissions (bursts) that are carried or blocked can be derived using a recursive scheme via an occupancy model. Here, we provide an analytical result for the blocking probability P_b. Suppose a given source sends a request to each destination j $(1 \leq j \leq N)$ and receives a corresponding grant to transmit d_j bursts in the next scheduling cycle. Let $D = \sum_{j=1}^{N} d_j \leq B$ be the total number of bursts to be transmitted by the source. Since each destination computes its schedule independently, the grants received from different destinations may cause transmission conflicts. In the following, we state the blocking probability result due to conflicts. The proof is given in Appendix A.

THEOREM 1. *The blocking probability due to conflicts is given by*

$$P_b = 1 - \frac{B}{D} + \frac{B}{D} \prod_{j=1}^{N} (1 - \frac{d_j}{B})$$

When $d_j = B/N$ for $1 \leq j \leq N$ and $D = B$, then $P_b = (1 - \frac{1}{N})^N \to 1/e$ as $N \to \infty$.

Fig. 5 plots the blocking performance for different numbers of bursts with cycle time of $B = 150$ time slots. We vary the number of sources N and the number of bursts requested per source d so that $\rho = Nd/B$ remains fixed at 1, 0.8, 0.6, 0.4 and 0.2. When $Nd = 150$, we observe that the worst-case blocking probability is about 0.36 for large N, or the maximum throughput achievable is about 0.64.

Quasi-static load. When a burst transmission granted by a given destination is blocked due to a source conflict, the source can simply refrain from transmitting on its assigned time slot. Thus, the destination can *learn* about a source conflict if it detects that an assigned time slot does not contain a burst after a round-trip propagation delay[5]. Learning takes advantage of the fact that bandwidth demand per cycle has smaller fluctuations as traffic volume increases [27]. Fig. 6 illustrates an example of learning. Here, a given destination grants available time slots 1, 6 and 9 (chosen randomly) to the source and slot 6 causes a conflict (because the source already uses it for transmission to another destination). The source thus only transmits bursts on time slots 1 and 9 on subsequent cycles. When the destination learns about the conflict on slot 6, it selects a new time slot (e.g., 8) and reissues a grant. If another conflict reoccurs, the destination may reissue another grant.

Instead of having the source refrain from transmitting on its assigned time slot when there is a conflict, an alternative is for the source to bump existing transmission to a different destination that causes the conflict. The rationale for doing this is to allow for more flexible schedules through rearrangement. To this end, we adopt randomization for bumping to limit disruption. Specifically, when a source receiving a

[5]For robustness, if there is no conflict but the source does not have data to send, the source needs to send "dummy" bursts on the assigned time slots.

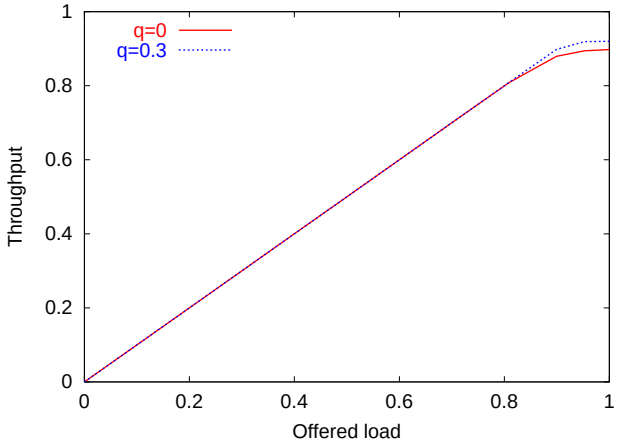

Figure 7: Throughput versus offered load under moderate propagation delays.

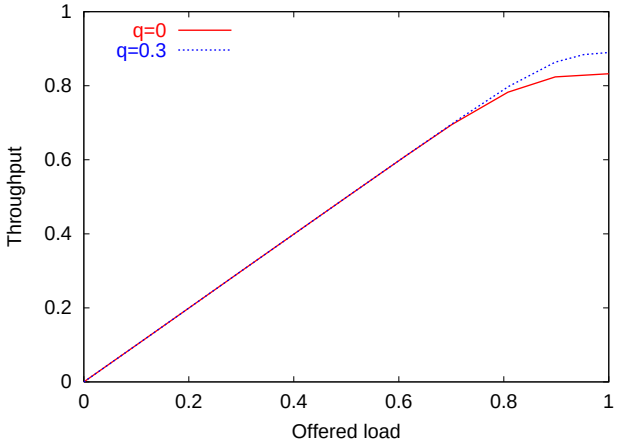

Figure 8: Throughput versus offered load under large propagation delays.

grant from destination j causes a conflict as existing transmission is already used for destination j', the source stops transmission to destination j' and begins transmission to destination j with probability q. With probability $(1-q)$, the source simply refrains from transmitting bursts to destination j. In either case, the respective destination detects a source conflict and will reassign a different time slot to the source. Randomization techniques have been applied successfully in the context of switch scheduling [28][29].

We present some preliminary results on the performance of the distributed protocol using simulation (a more thorough investigation will be covered elsewhere). In the following, we use a 10-node network and assume that $B = 500$ time slots and $T_s = 10\mu$s. The end-to-end distance δ_{ij} between two edge nodes is independently and uniformly distributed in (x_L, x_U). If optical signal propagates at the speed of 200,000 km/sec, then a distance of 2 km will incur a delay of one time slot. We assume a uniform traffic model where $E[d_{ij}] = \bar{d}$ for all $i \neq j$. To model quasi-static traffic, we use a simple Markov process whereby each d_{ij} independently alternates in two states corresponding to two different rates with an average dwelling time of 200 cycles in each of the two states. We further assume a bufferless model so that a request that is not successfully granted is lost.

Fig. 7 plots the throughput versus offered load when $x_L = 10$ km and $x_U = 30$ km so that the average distance is 20 km (metro network). The offered load denotes the normalized average number of time slots requested and the throughput denotes the normalized average number of time slots successfully granted. Note that the throughput tracks the offered load under moderate load, but eventually levels off to about 0.92 with $q = 0.3$ (or 0.89 with $q = 0$) under heavy load. An interesting observation is that scheduling with source bumping outperforms the one without, indicating the advantage of rearrangement.

Fig. 8 shows the corresponding plot when $x_L = 1000$ km and $x_U = 3000$ km so that the average distance is 2000 km (long-distance network). Notice that the benefit of source bumping is also clearly demonstrated for large delays.

We note that there are still many issues in distributed scheduling that are worth investigating. For example, it is interesting to investigate further the dynamics of the learning protocol with real traffic. Others may include treatment with multi classes of traffic, fairness among different flows (source-destination pairs), and interaction with higher-layer protocols (e.g., TCP). In a multi-class traffic scenario, one approach may be to allow high-priority traffic to bump low-priority traffic, but not vice versa. Fairness may be implemented so that a fairness controller (e.g., similar to the idea in [30]) computes the number of time slots to be granted to each source. For some services, the scheduler may need to guarantee a certain minimum rate for each flow.

4. OTHER ISSUES

4.1 Scalability

In the basic TWIN network with N edge nodes, each destination is assigned a unique wavelength with a capacity of C b/s. Thus, from the capacity side, a TWIN network can carry a total NC b/s of ingress and egress traffic. Recent record experiment on a fiber optic link can achieve a capacity of 6.4 Tb/s (160 wavelengths each at 42.7 Gb/s) with a capacity-distance product of 20 Pbit/s-km [31]. Based on the fundamental limit [32], however, the information capacity estimate of a fiber link with spectral efficiency of 3 b/s/Hz is even much higher at about 150 Tb/s. Thus, the capacity of TWIN can be expected to grow much further with continued improvement in transmission technology.

To check the demand side, we quote from Andrew Odylzko [33]:

> *"The upper estimate [...] for U.S. Internet backbone traffic at year-end 2002, 140 PB/month, is equivalent to only about 420 Gb/s on average. Even if we make allowances for a considerably higher than usual 1.9 ratio of peak hour to average traffic on backbones, we find that it would take only 800 Gb/s of capacity to transport all the traffic on all U.S. backbones, and that much capacity can now comfortably fit on a single fiber!"*

Since a single network or subnetwork generally carries a significantly smaller portion of the entire U.S. Internet backbone traffic and since transmission technology can be expected to continue to improve, we believe that the capacity

of basic TWIN for practical purposes would be sufficient in the foreseeable future. Nevertheless, it is also worthwhile to investigate scaling of TWIN to be able to meet traffic demand in the very long term.

There are several approaches to scale the size of TWIN. One approach is to partition the network into multiple islands. Each island uses a unique set of wavelengths but different islands can reuse the same set of wavelengths. The islands may be naturally bounded by the reach constraint of optical transmission. When client traffic needs to traverse from one island to another, OEO conversion is needed at the border of these islands. Therefore, it is desirable to minimize the traffic that crosses different islands. Given traffic demand information, one problem of interest may be optimal partitioning of a TWIN network into islands to minimize OEO conversion or to maximize the "1-hop" carried load. Another approach to scaling is is to take advantage of multiple-fiber strands that are commonly found in a bundle and run the same wavelengths on different fibers.

4.2 Survivability

Service provider networks are usually designed with survivability so that a failure in the network will only cause disruption locally and for a short period of time. Survivability may be provided through path protection. In 1+1 path protection, traffic is copied on both the *working* and *protection* paths and the destination selects the best path. In 1:1 path protection, traffic is sent to the working path during normal operation. When a failure occurs on the working path, traffic is automatically switched to the protection path. Path protection typically doubles the capacity required in a network.

1:1 protection can be implemented in TWIN by configuring multipoint-to-point trees with redundancy. Tree redundancy can be created using edge-disjoint trees, arc-disjoint trees, or multi-trees [34]. An important issue in TWIN is the impact of protection on scheduling efficiency. Here, a scheduler may want to *precompute* schedules for both working and protection paths jointly and take into account possible sharing of time slots. Preliminary results show substantially improved efficiency due to sharing [24]. Precomputed schedules also allow for faster restoration time.

4.3 Control Plane

A TWIN control plane is needed to automatically discover new core and edge nodes, assign wavelengths to destinations, setup a tree for each destination, perform ranging to determine propagation delays between edge nodes, provision bandwidth within the network, and other relevant functions such as restoration. Control messages may be carried in a separate data communications network (DCN) whose links may utilize wavelength channels that are dedicated for each pair of neighboring nodes. Once the above tasks have been completed, the advantage of TWIN should become apparent as the network acts like a switch. For example, bandwidth provisioning in TWIN only needs to be accounted for at sources and destinations, but not at core nodes. This implies that TWIN can be used to easily provision the hose model for VPN [35] without worrying about the internal of the network. Unlike connection-oriented networks such as MPLS, there are no virtual circuits to maintain in TWIN. As a result, there is also no traffic engineering issue inside the network.

5. CONCLUSION

In this paper, we have shown how a cost-effective network (TWIN) with simplified layering can be constructed by exchanging fast switches in the network core with fast-tunable lasers at the network edge. Key to our proposal is the elimination of expensive transit ports and processing in the network core that are both common in today's networks. In TWIN, the network acts like a giant and distributed Birkhoff-von Neumann switch with input/output ports spread across the edge nodes and passive fabric spread across the core nodes. However, TWIN's distributed switch differs from a normal switch in that propagation delays from input ports to output ports may be significant. To the best of our knowledge, network scheduling that explicitly takes into account non-zero propagation delays has not been addressed prior to TWIN. We have presented some results on distributed scheduling and discussed several research issues. We believe that the concept of "network as a switch" has a potential of not only reducing network layering that exists in today's service provider network infrastructures, but also simplifying many of the provisioning and bandwidth-management issues.

6. REFERENCES

[1] A. Odlyzko, "Pricing and architecture of the Internet: historical perspectives from telecommunications and transportation," available at http://www.dtc.umn.edu/∼odlyzko.

[2] L. Martini et al., "Encapsulation methods for transport of Ethernet frames over IP/MPLS networks," IETF work in progress, Apr. 2003.

[3] L. Martini et al., "Encapsulation methods for transport of ATM over IP and MPLS networks," IETF work in progress, Apr. 2004

[4] C. Kawa et al., "Frame relay over pseudo-wires," IETF work in progress, Feb. 2004.

[5] ATM Forum, "Circuit emulation service - interoperability specification," AF-SAA-0032.000, Sep. 1995.

[6] A. Malis et al., "SONET/SDH circuit emulation over packet (CEP)," IETF work in progress, Apr. 2004.

[7] N. Spring, R. Mahajan, D. Wetherhall and T. Anderson, "Measuring ISP topologies with rocketfuel," in *Proc. SIGCOMM 2002*, Pittsburgh, Aug. 2002.

[8] C. Fraleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network Magazine*, pp. 6-16, Nov. 2003.

[9] S. Sudipta, V. Kumar and D. Saha, "Switched optical backbone for cost-effective scalable core IP networks," *IEEE Comm. Magazine*, pp. 60-70, Jun. 2003.

[10] P. Molinero-Fernandez, N. McKeown and H. Zhang, "Is IP going to take the world (of communications)?," *HotNets-I*, Princeton, Oct. 2002.

[11] I. Widjaja, I. Saniee, R. Giles and D. Mitra, "Light-core and intelligent edge for a flexible, thin-layered, and cost-effective optical transport network," *IEEE Comm. Magazine*, pp. S30–S36, May 2003.

[12] C. Chang, W. Chen and H. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. IEEE INFOCOM 2000*, Aprl. 2000.

[13] S. Yao, S. Dixit and B. Mukherjee, "Advances in photonic packet switching: an overview," *IEEE Comm. Magazine*, pp. 84-94, Feb. 2000.

[14] T. S. El-Bawab and J. Shin, "Optical packet switching in core networks: between vision and reality," *IEEE Comm. Magazine*, pp. 60-65, Sep. 2002.

[15] J. Turner, "Terabit burst switching," *J. High Speed Networks*, vol. 8, no. 1, pp. 3-16, Jan. 1999.

[16] C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an optical Internet," *J. High Speed Networks*, vol. 8, no. 1, pp. 69-84, Jan. 1999.

[17] D. M. Marom et al., "Wavelength-selective 1x4 switch for 128 WDM channels at 50 GHz spacing," in *Proc. OFC 2002*, postdeadline paper FB7, Los Angeles, 2002.

[18] D. M. Marom, "Wavelength selective $1 \times K$ switches for transparent optical networks," in *Proc. SPIE 2002*, Vol. 4907, Oct. 2002.

[19] M. Kauer et al., "16-channel digitally tunable packet switching transmitter with sub-nanosecond switching time," in *Proc. ECOC 2002*, paper 3.3.3, 2002.

[20] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. IEEE INFOCOM 1998*, Mar. 1998.

[21] N. McKeown, "The iSLIP scheduling algorithm for input-queue switches," *IEEE Trans. on Networking*, vol. 7, pp. 188-201, Apr. 1999.

[22] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM 2000*, Apr. 2000.

[23] http://www.symmetricom.com

[24] K. Ross, N. Bambos, K. Kumaran, I. Saniee, I. Widjaja, "Scheduling bursts in time-domain wavelength interleaved networks," *IEEE J. Selected Areas in Communications*, vol. 21, no. 9, pp. 1441-1451, Nov. 2003.

[25] CableLabs, "Data-over-cable service interface specifications - radio frequency interface specification," SP-RFIv2.0-I01-011231, Dec. 2001.

[26] IEEE, "Draft amendment to carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," IEEE Draft P802.3ah/D3.1, Feb. 2004.

[27] C. Fraleigh, F. Tobagi and C. Diot, "Provisioning IP backbone networks to support latency sensitive traffic," in *Proc. IEEE INFOCOM 2003*, Apr. 2003.

[28] T. E. Anderson, S. S. Owicki, J. B. Saxe and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[29] M. W. Goudreau, S. G. Kolliopoulos and S. B. Rao, "Scheduling algorithms for input-queued switches: randomized techniques and experimental evaluation," in *Proc. IEEE INFOCOM 2000*, Apr. 2000.

[30] D. Katabi, M. Handley and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. SIGCOMM 2002*, Pittsburgh, 2002.

[31] B. Zhu et al., "6.4-Tb/s (160x42.7 Gb/s) transmission with 0.8 bits/s/Hz spectral efficiency over 32x100 km of fiber using CSRZ-DPSK format," in *Proc. OFC 2003*, postdeadline paper PD19, Georgia, 2003.

[32] P.P. Mitra and J.B. Stark, "Nonlinear limits to the information capacity of optical fiber communications," *Nature*, vol. 411, pp. 1027-1030, Jun. 2001.

[33] A. Odlyzko, "Internet traffic growth: sources and implications," *Proceedings of SPIE 2003*, vol. 5247, 2003.

[34] M. Médard, S. G. Finn, R. A. Barry and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. on Networking*, vol. 7, no. 5, Oct. 1999.

[35] N.G. Duffield, A. Greenberg, P. Mishra, K.K. Ramakrishnan and J.E. van der Merwe, "A flexible model for resource management in virtual private networks," in *Proc. SIGCOMM 1999*, Boston, Oct. 1999.

APPENDIX

A. PROOF OF THEOREM 1

Consider a particular source. Define d_j to be the number of time slots requested by the source to destination j, where $D = \sum_{j=1}^{N} d_j$. Upon receiving a request, destination computes the schedule and grants d_j time slots to the source. Let us focus on a particular slot at the source, and let $p = Pr\{\text{a slot receives no grants for transmission}\}$. Then,

$$p = \prod_{j=1}^{N} \frac{\binom{B-1}{d_j}}{\binom{B}{d_j}} = \prod_{j=1}^{N} \left(1 - \frac{d_j}{B}\right) \quad (1)$$

Thus, the mean number of empty slots, $m_0 = Bp$. The average blocking can be given by

$$P_b = (D - (B - m_0))/D. \quad (2)$$

From (1) and (2), we obtain

$$P_b = 1 - \frac{B}{D} + \frac{B}{D} \prod_{j=1}^{N} (1 - \frac{d_j}{B}).$$