Scaling Internet Routers Using Optics (Extended Version)

Isaac Keslassy, Shang-Tse Chuang, Kyoungsik Yu,
David Miller, Mark Horowitz, Olav Solgaard, Nick McKeown
Stanford University

This paper is an extended version of [1]. In conjunction with "A load-balanced switch with an arbitrary number of linecards" [2], it replaces "Architectures and algorithms for a load-balanced switch" [3].

***Abstract*— Routers built around a single-stage crossbar and a centralized scheduler do not scale, and (in practice) do not provide the throughput guarantees that network operators need to make efficient use of their expensive long-haul links. In this paper we consider how optics can be used to scale capacity and reduce power in a router. We start with the promising *load-balanced switch architecture* proposed by C-S. Chang. This approach eliminates the scheduler, is scalable, and guarantees 100% throughput for a broad class of traffic. But several problems need to be solved to make this architecture practical: (1) Packets can be mis-sequenced, (2) Pathological periodic traffic patterns can make throughput arbitrarily small, (3) The architecture requires a rapidly configuring switch fabric, and (4) It does not work when linecards are missing or have failed. In this paper we solve each problem in turn, and describe new architectures that include our solutions. We motivate our work by designing a 100Tb/s packet-switched router arranged as 640 linecards, each operating at 160Gb/s. We describe two different implementations based on technology available within the next three years.**

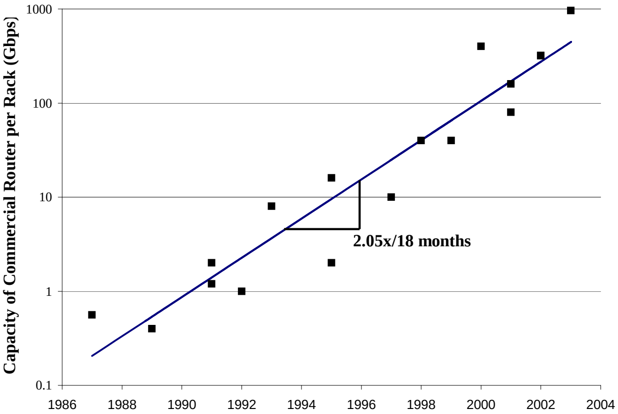


Fig. 1. The growth in router capacity over time, per unit volume. Each point represents one commercial router at its date of introduction, normalized to how much capacity would fit in a single rack. Best linear fit: 2.05-fold increase every 18 months [7].

I. Introduction and Motivation

This paper is motivated by two questions: First, how can the capacity of Internet routers scale to keep up with growths in Internet traffic? And second, can optical technology be introduced *inside* routers to help increase their capacity?

Before we try to answer these questions, it is worth asking if the questions are still relevant. After all, the Internet is widely reported to have a glut of capacity, with average link utilization below 10%, and a large fraction of installed but unused link capacity [4]. The introduction of new routers has been delayed, suggesting that faster routers are not needed as urgently as we once thought.

While it is not the goal of this paper to argue *when* new routers will be needed, we argue that the capacity of routers must continue to grow. The underlying demand for network capacity (measured by the amount of user traffic) continues to double every year [5], and if this continues, will require an increase in router capacity. Otherwise, Internet providers must double the number of routers in their network each year, which is impractical for a number of reasons: First, it would require doubling either the size or the number of central offices each year. But central offices are reportedly full already [6], with limited space, power supply and ability to dissipate power from racks of equipment. And second, doubling the number of locations would require enormous capital investment and increases in the support and maintenance infrastructure to manage the enlarged network. Yet this still would not suffice; additional routers are needed to interconnect other routers in the enlarged topology, so it takes more than twice as many routers to carry twice as much user traffic with the same link utilization. Instead, it seems reasonable to expect that router capacity will continue to grow, with routers periodically replaced with newer higher capacity systems.

Historically, routing capacity per unit volume has doubled every eighteen months (see Figure 1).[1] If Internet traffic continues to double every year, in nine years traffic will have grown eight times more than the capacity of individual routers.

Each generation of router consumes more power than the last, and it is now difficult to package a router in one rack of equipment. Network operators can supply and dissipate about 10 kW per rack, and single-rack routers have reached this limit. There has therefore been a move towards multi-

This work was funded in part by the DARPA/MARCO Center for Circuits, Systems and Software, by the DARPA/MARCO Interconnect Focus Center, Cisco Systems, Texas Instruments, Stanford Networking Research Center, Stanford Photonics Research Center, and a Wakerly Stanford Graduate Fellowship.

[1] Capacity is often limited by memory bandwidth (defined here as the speed at which random packets can be retrieved from memory). Despite large improvements in I/O bandwidths, random access time has improved at only 1.1-fold every eighteen months. Router architects have therefore made great strides to introduce new techniques to overcome this limitation.

rack systems, with either a remote, single-stage crossbar switch and central scheduler [8], [9], [10], [11], or a multi-stage, distributed switch [12], [13]. Multi-rack routers spread the system power over multiple racks, reducing power density. For this reason, most high-capacity routers currently under development are multi-rack systems.

Existing multi-rack systems suffer from two main problems: Unpredictable performance, and poor scalability (or both). Multi-rack systems with distributed, multistage switching fabrics (such as buffered Benes or Clos networks, hypercubes or toroids) have unpredictable performance. This presents a problem for the network operators: They don't know what utilization they can safely operate their routers at; and if the throughput is less than 100%, they are unable to use the full capacity of their expensive long-haul links. This is to be contrasted with single-stage switches for which throughput guarantees are known [14], [15].

However, single-stage switches (e.g. crossbars with combined input and output queueing) have problems of their own. Although arbitration algorithms can theoretically give 100% throughput,[2] they are impractical because of the complexity of the algorithm, or the speedup of the buffer memory. In practice, single-stage switch fabrics use sub-optimal schedulers (e.g. based on WFA [16] or *i*SLIP [17]) with insufficient speedup to guarantee 100% throughput. Future higher capacity single-stage routers are not going to give throughput guarantees either: Centralized schedulers don't scale with an increase in the number of ports, or with an increase in the line-rate. Known maximal matching algorithms for centralized schedulers (PIM [18], WFA [16], *i*SLIP [17]) need at least $O(N^2)$ interconnects for the arbitration process, where N is the number of linecards. Even if arbitration is distributed over multiple ASICs, interconnect power still scales with $O(N^2)$. The fastest reported centralized scheduler (implementing maximal matches, a speedup of less than two and no 100% throughput guarantees) switches 256 ports at 10Gbps [8]. This design aims to maximize capacity with current ASIC technology, and is limited by the power dissipation and pin-count of the scheduler ASICs. Scheduler speed will grow slowly (because of the $O(N^2)$ complexity, it will grow approximately with $\sqrt{N}$), and will continue to limit growth.

In summary, multi-rack systems either use a multi-stage switch fabric spread over multiple racks, and have unpredictable throughput; or they use a single-stage switch fabric in a single rack that is limited by power, and use a centralized scheduler with unpredictable throughput. If a router is to have predictable throughput, its capacity is currently limited by how much switching capacity can be placed in a single rack. Today, the limit is approximately 2.5Tb/s, and is constrained by power consumption.

Our goal is to identify architectures with predictable throughput and scalable capacity. In this paper we'll explain how we can use optics with almost zero power consumption to place the switch fabric of a 100Tb/s router in a single rack, without sacrificing throughput guarantees. This is approximately 40 times greater than the electronic switching capacity that could be put in a single rack today. We describe our conclusion that the Load-Balanced switch, first described by C-S. Chang *et al.* in [19] (which extends Valiant's method [20]), is the most promising architecture. It has provably 100% throughput. It is scalable: It has no central scheduler, and is amenable to optics. It simplifies the switch fabric, replacing a frequently scheduled and reconfigured switch with two identical switches that follow a fixed sequence, or are built from a mesh of WDM channels.

In what follows we will start by describing Chang's Load-Balanced switch architecture in Section II, and explain how it guarantees 100% throughput without a scheduler. We then tackle four main problems with the basic Load-Balanced switch that make it unsuitable for use in a high-capacity router: (1) It requires a rapidly configuring switch fabric, making it difficult, or expensive to use an optical switch fabric, (2) Packets can be mis-sequenced, (3) Pathological periodic traffic patterns can make throughput arbitrarily small, and (4) It does not work when some linecards are missing or have failed. In the remainder of the paper we find practical solutions to each: In Section IV we show how novel buffer management algorithms can prevent mis-sequencing and eliminate problems with pathological periodic traffic problems. The algorithms also make possible multiple classes of service. In Section V we show how problem (3) can be solved by replacing the crossbar switches by a fixed optical mesh — a powerful and perhaps surprising extension of the load-balanced switch. And then in Section VI we explain why problem (4) is the hardest problem to solve. We describe two implementations that solve the problem: One with a hybrid electro-optical switch fabric, and one with an all-optical switch fabric.

II. Load-Balanced Architecture

A. The Architecture

The basic load-balanced switch is shown in Figure 2, and consists of a single stage of buffers sandwiched by two identical stages of switching. The buffer at each intermediate input is partitioned into N separate FIFO queues, one per output (hence we call them virtual output queues, VOQs). There are a total of N^2 VOQs in the switch.

The operation of the two switch fabrics is quite different from a normal single-stage packet switch. Instead of picking a switch configuration based on the occupancy of the queues, both switching stages walk through a fixed sequence of configurations. At time t, input i of each switch fabric is connected to output $[(i+t) \bmod N]+1$; i.e. the configuration is a cyclic shift, and each input is connected to each output exactly $\frac{1}{N}$-th of the time, regardless of the arriving traffic. We will call each stage a *fixed, equal-rate* switch. Although they are identical, it helps to think of the two stages as performing different functions. The first stage is a load-balancer that spreads traffic over all the VOQs. The second stage is an input-queued crossbar switch in which each VOQ is served at a fixed rate.

When a packet arrives to the first stage, the first switch immediately transfers it to a VOQ at the (intermediate) input of the second stage. The intermediate input that the packet goes to depends on the current configuration of the load-balancer. The packet is put into the VOQ at the intermediate input according

[2] For example WFA [16] with a speedup of 2, MWM with a speedup of one [15].

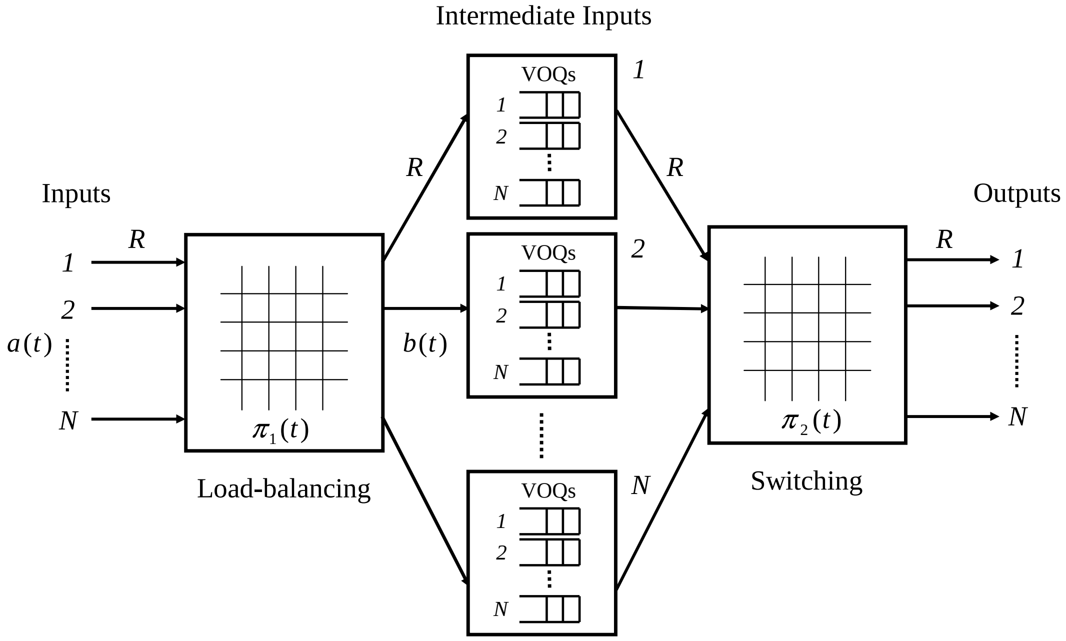


Fig. 2. Load-balanced router architecture

to its eventual output. Sometime later, the VOQ will be served by the second fixed, equal-rate switch. The packet will then be transferred across the second switch to its output, from where it will depart the system.

B. 100% Throughput

At first glance, it is not obvious how the load-balanced switch can make any throughput guarantees; after all, the sequence of switch configurations is pre-determined, regardless of the traffic or the state of the queues. In a conventional single-stage crossbar switch, throughput guarantees are only possible if a scheduler configures the switch based on knowledge of the state of all the queues in the system. In what follows, we will give an intuitive explanation of the architecture, followed by an outline of a proof that it guarantees 100% throughput for a broad class of traffic.

Intuition: Consider a single fixed, equal-rate crossbar switch with VOQs at each input, that connects each input to each output exactly $\frac{1}{N}$-th of the time. For the moment, assume that the destination of packets is *uniform*; i.e. arriving packets are equally likely to be destined to any of the outputs.[3] (Of course, real network traffic is nothing like this — but we will come to that shortly.) The fixed, equal-rate switch serves each VOQ at rate R/N, allowing us to model it as a GI/D/1 queue, with arrival rate $\lambda < R/N$ and service rate $\mu = R/N$. The system is stable (the queues will not grow without bound), and hence it guarantees 100% throughput.

Fact: If arrivals are uniform, a fixed, equal-rate switch, with virtual output queues, has a guaranteed throughput of 100%.

Of course, real network traffic is *not* uniform. But an extra load-balancing stage can spread out non-uniform traffic, making it sufficiently uniform to achieve 100% throughput. This is the basic idea of the two-stage load-balancing switch. A load-balancing device spreads packets evenly to all the inputs of a second, fixed, equal-rate switch.

Outline of proof: The load-balanced switch has 100% throughput for non-uniform arrivals for the following reason. Referring again to Figure 2, consider the arrival process, $a(t)$ (with N-by-N traffic matrix Λ) to the switch. This process is transformed by the sequence of permutations in the load-balancer, $\pi_1(t)$, into the arrival process to the second stage, $b(t) = \pi_1(t) \cdot a(t)$. The VOQs are served by the sequence of permutations in the switching stage, $\pi_2(t)$. If the inputs and outputs are not over-subscribed, then the long-term service opportunities exceed the number of arrivals, and hence the system achieves 100% throughput:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}(b(t) - \pi_2(t)) = \frac{1}{N}e\Lambda - \frac{1}{N}e < 0,$$

where e is a matrix full of 1's.

In [19] the authors prove this more rigorously, and extend it to all sequences $\{a(t)\}$ that are stationary, stochastic and weakly mixing.

III. A 100Tb/s Router Example

The load-balanced switch seems to be an appealing architecture for scalable routers that need performance guarantees. In what follows we will study the architecture in more detail. To focus our study, we will assume that we are designing a 100Tb/s Internet router that implements the requirements of RFC 1812 [21], arranged as 640 linecards operating at 160Gb/s

[3] More precisely, assume that when a packet arrives, its destination is picked uniformly and at random from among the set of outputs, independently from packet to packet.

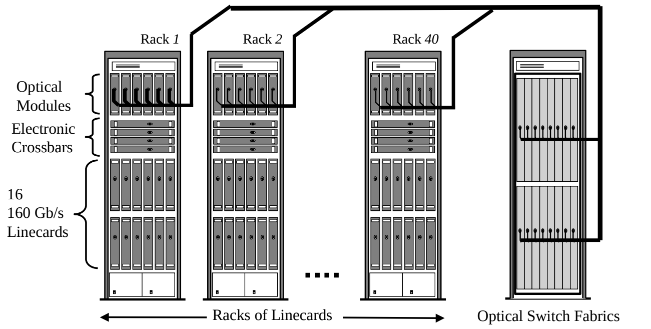


Fig. 3. Possible system packaging for a 100 Tb/s router with 640 linecards

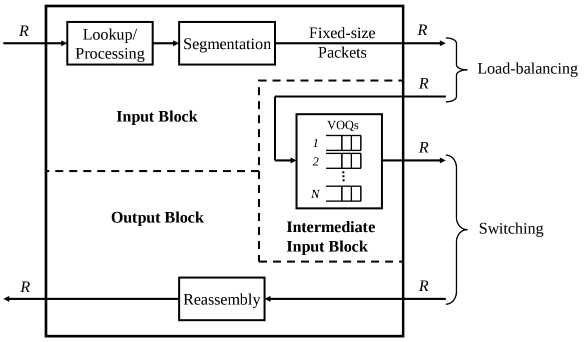


Fig. 4. Linecard block diagram

(OC-3072). We pick 100Tb/s because it is challenging to design, is probably beyond the reach of a purely electronic implementation, but seems possible with optical links between racks of distributed linecards and switches. It is roughly two orders of magnitude larger than Internet routers currently deployed, and seems feasible to build using technology available in approximately three years time. We pick 160Gb/s for each linecard because 40Gb/s linecards are feasible now, and 160Gb/s is the next logical generation.

We will adopt some additional requirements in our design: The router must have a guaranteed 100% throughput for any pattern of arrivals, must not mis-sequence packets, and should operate correctly when populated with any number of linecards connected to any ports.

The router is assumed to occupy multiple racks, as shown in Figure 3, with up to 16 linecards per rack. Racks are connected by optical fibers and one or more racks of optical switches. In terms of optical technology, we will assume that it is possible to multiplex and demultiplex 64 WDM channels onto a single optical fiber, and that each channel can operate at up to 10Gb/s.

Each linecard will have three parts: An Input Block, an Output Block, and an Intermediate Input Block, shown in Figure 4. As is customary, arriving variable length packets will be segmented into fixed sized packets (sometimes called “cells”, though not necessarily equal to a 53-byte ATM cell), and then transferred to the eventual output, where they are reassembled into variable length packets again. We will call them fixed-size packets, or just “packets” for short. The Input Block performs address lookup, segments the variable length packet into one or more fixed length packets, and then forwards the packet to the switch. The Intermediate Input Block accepts packets from the switch and stores them in the appropriate VOQ. It takes packets from the head of each VOQ at rate R/N and sends them to the switch to be transferred to the output. Finally, the Output Block accepts packets from the switch, collects them together, reassembles them into variable length packets, and delivers them to the external line. Each linecard is connected to the external line with a bidirectional link at rate R, and to the switch with two bidirectional links at rate R.

Despite its scalability, the basic load-balanced switch has some problems that need to be solved before it meets our requirements. In the following sections we describe and then solve each problem in turn.

IV. Switch Reconfigurations

A. *Fixed Mesh*

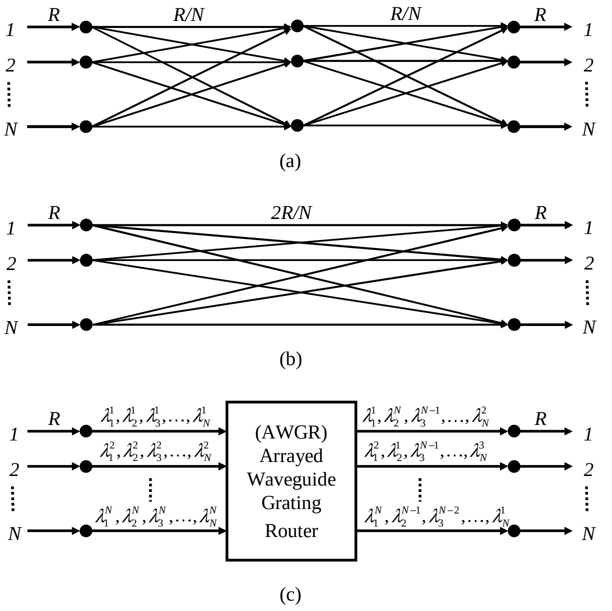
While the load-balanced switch has no centralized scheduler to configure the switch fabric, it still needs a switch fabric of size $N \times N$ that is reconfigured for each packet transfer (albeit in a deterministic, predetermined fashion). While optical switch fabrics that can reconfigure for each packet transfer offer huge capacity and almost zero power consumption, they can be slow to reconfigure (e.g. MEMS switches that typically take over 10ms to reconfigure) or are expensive (e.g. switches that use tunable lasers or receivers).[4] Below, we’ll see how the switch fabric can be replaced by a fixed mesh of optical channels that don’t need reconfiguring.

Our first observation is that we can replace each fixed, equal-rate switch with N^2 fixed channels at rate R/N, as illustrated in Figure 5(a).

Our second observation is that we can replace the two switches with a single switch running twice as fast. In the basic switch, both switching stages connect every (input, output) pair at fixed rate R/N, and every packet traverses both switching stages. We replace the two meshes with a single mesh that connects every (input, output) pair at rate $2R/N$, as shown in Figure 5(b). Every packet traverses the single switching stage twice; each time at rate R/N. This is possible because in a physical implementation, a linecard contains an input, an intermediate input and an output. When a packet has crossed the switch once, it is in an intermediate linecard; from there, it crosses the switch again to reach the output linecard.

The single fixed mesh architecture leads to a couple of interesting questions. The first question is: Does the mesh need to be uniform? i.e. so long as each linecard transmits and receives data at rate $2R$, does it matter how the data is spread across the intermediate linecards? Perhaps the first stage linecards could spread data over half, or a subset of the intermediate linecards. The answer is that if we don’t know the traffic matrix, the mesh must be uniform. Otherwise, there is not a guaranteed aggregate rate of R available between any pair of linecards. The second question is: If it is possible to build a packet switch with 100% throughput that has no scheduler, no reconfigurable switch fabric, and buffer memories operating without speedup, where does the packet switching actually take place? It takes place at the input of the buffers in the intermediate linecards —

[4] A glossary of the optical devices used in this paper appears in the Appendix.


R
R/N
R/N
R
1
2
N
1
2
N
R
2R/N
R
1
2
N
1
2
N
R
$\lambda_1^1, \lambda_2^1, \lambda_3^1, \ldots, \lambda_N^1$
$\lambda_1^2, \lambda_2^2, \lambda_3^2, \ldots, \lambda_N^2$
$\lambda_1^N, \lambda_2^N, \lambda_3^N, \ldots, \lambda_N^N$
(AWGR)
Arrayed
Waveguide
Grating
Router
$\lambda_1^1, \lambda_2^N, \lambda_3^{N-1}, \ldots, \lambda_N^2$
$\lambda_1^2, \lambda_2^1, \lambda_3^{N-1}, \ldots, \lambda_N^3$
$\lambda_1^N, \lambda_2^{N-1}, \lambda_3^{N-2}, \ldots, \lambda_N^1$
R
1
2
N
1
2
N

(a)

(b)

(c)

Fig. 5. Two ways in which the load-balanced switch can be implemented by a single fixed-rate uniform mesh. In both cases, two stages operating at rate R/N, as shown in (a), are replaced by one stage operating at $2R/N$, and every packet traverses the mesh twice. In (b), the mesh is implemented by N^2 fibers. In (c), the mesh is N^2 WDM channels interconnected by an AWGR. λ_w^i is transmitted on wavelength λ_w from input i and operates at rate $2R/N$.

the linecard decides which output the packet is destined to, and writes it to the correct VOQ.

B. When N is Large

A mesh of links works well for small values of N, but in practice, N^2 optical fibers or electrical links is impractical or too expensive. For example, a 64-port router, with 40Gb/s lines (i.e. a capacity of 2.5Tb/s) would require 4,000 fibers or links, each carrying data at 1.25Gb/s. Instead, we can use wavelength division multiplexing to reduce the number of fibers, and increase the data-rate carried by each. This is illustrated in Figure 5(c). Instead of connecting to N fibers, each linecard multiplexes N WDM channels onto one fiber, with each channel operating at $2R/N$. The $N \times N$ arrayed waveguide grating router (AWGR) in the middle is a passive data-rate independent optical device that routes wavelength w at input i to output $[(i + w - 2) \bmod N] + 1$. The number of fibers is reduced to $2N$, at the cost of N wavelength multiplexers and demultiplexers, one on each linecard. The number of lasers is the same as before (N^2), with each of the N lasers on one linecard operating at a different, fixed wavelength. Currently, it is practical to use about 64 different WDM channels, and AWGRs have been built with more than 64 inputs and outputs [22]. If each laser can operate at 10Gb/s,[5] this would enable routers to be built up to about 20Tb/s, arranged as 64-ports, each operating at $R = 320$Gb/s.

Our 100Tb/s router has too many linecards to connect directly to a single, central optical switch. A mesh of WDM channels connected to an AWGR (Figure 5(c)) would require 640 distinct wavelengths, which is beyond what is practical today. In fact a passive optical switch cannot interconnect 640 linecards. To do so inherently requires the switch to take data from each of the 640 linecards and spread it back over all 640 linecards in at least 640 distinct channels. We are not aware of any multiplexing scheme that can do this. If we try to use an active optical switch instead (such as a MEMS switch [24], electro-optic [25] or electro-holographic waveguides [26]), we must reconfigure it frequently (each time a packet is transferred), and we run into problems of scale. It does not seem practical to manufacture an active, reliable, frequently reconfigured 640-port switch from any of these technologies. And so we need to decompose the switch into multiple stages. Fortunately this is simple to do with a load-balanced switch. The switch does not need to be non-blocking; it just needs a path to connect each input to each output at a fixed rate.[6] In Section VI, we will describe two different three-stage switch fabric architectures that decompose the switch fabric by arranging the linecards in groups (corresponding, in practice, to racks of linecards).

V. Packet Mis-sequencing

In the basic architecture, the load-balancer spreads packets without regard to their final destination, or when they will depart. If two packets arrive back to back at the same input, and are destined to the same output, they could be spread to different intermediate linecards, with different occupancies. It is possible that their departure order will be reversed. While mis-sequencing is allowed (and is common) in the Internet,[7] network operators generally insist that routers do not mis-sequence packets belonging to the same application flow. In its current version, TCP does not perform well when packets arrive to the destination out of order because they can trigger un-necessary retransmissions.

There are two approaches to prevent mis-sequencing: To prevent packets from becoming mis-sequenced anywhere in the router [27]; or to bound the amount of mis-sequencing, and use a re-sequencing buffer in the third stage [28]. None of the schemes published to date would work in our 100Tb/s router. The schemes use schedulers that are hard to implement at these speeds, need jitter control buffers that require N writes to memory in one time slot [28], or require the communication of too much state information between the linecards [27].

A. Full Ordered Frames First

Instead we propose a scheme geared toward our 100Tb/s router. Full Ordered Frames First (FOFF) bounds the difference in lengths of the VOQs in the second stage, and then uses a re-sequencing buffer at the third stage.

FOFF runs independently on each linecard using information locally available. The input linecard keeps N FIFO queues — one for each output. When a packet arrives, it is placed at the tail of the FIFO corresponding to its eventual output. The basic idea is that, ideally, a FIFO is served only when it contains N or

[5] The modulation rate of lasers has been steadily increasing, but it is hard to directly modulate a laser faster because of wavelength instability and optical power ringing [23]. For example, 40Gb/s transceivers use external modulators.

[6] Compare this with trying to decompose a non-blocking crossbar into, say, a multiple stage Clos network.

[7] Internet RFC 1812 "Requirements for IP Version 4 Routers" [21] does not forbid mis-sequencing.

more packets. The first N packets are read from the FIFO, and each is sent to a different intermediate linecard. In this way, the packets are spread uniformly over the second stage.

More precisely, the algorithm for linecard i operates as follows:

1) Input i maintains N FIFO queues, $Q_1 \dots Q_N$. An arriving packet destined to output j is placed in Q_j.
2) Every N time-slots, the input selects a queue to serve for the next N time-slots. First, it picks round-robin from among the queues holding more than N packets. If there are no such outputs, then it picks round-robin from among the non-empty queues. Up to N packets from the same queue (and hence destined to the same output) are transferred to different intermediate linecards in the next N time-slots. A pointer keeps track of the last intermediate linecard that we sent a packet to for each flow; the next packet is always sent to the next intermediate linecard.

Clearly, if there is always at least one queue with N packets, the packets will be uniformly spread over the second-stage, and there will be no mis-sequencing. All the VOQs that receive packets belonging to a flow receive the same number of packets, so they will all face the same delay, and won't be mis-sequenced. Mis-sequencing arises only when no queue has N packets; but the amount of mis-sequencing is bounded, and is corrected in the third stage using a fixed length re-sequencing buffer.

B. Properties of FOFF

FOFF has the the following properties, which are proved in the Appendix.

- *Packets leave the switch in order.* FOFF bounds the amount of mis-sequencing inside the switch, and requires a re-sequencing buffer that holds at most $N^2 + 1$ packets (proof in Appendix II).
- *No pathological traffic patterns.* The 100% throughput proof for the basic architecture relies on the traffic being stochastic and weakly mixing between inputs. While this might be a reasonable assumption for heavily aggregated backbone traffic, it is not guaranteed. In fact, it is easy to create a periodic adversarial traffic pattern that inverts the spreading sequence, and causes packets for one output to pile up at the same intermediate linecard. This can lead to a throughput of just R/N for each linecard.
  FOFF prevents pathological traffic patterns by spreading a flow between an input and output evenly across the intermediate linecards. FOFF guarantees that the cumulative number of packets sent to each intermediate linecard for a given flow differs by at most one. This even spreading prevents a traffic pattern from concentrating packets to any individual intermediate linecard. As a result, FOFF generalizes the 100% throughput to *any* arriving traffic pattern; there are provably no adversarial traffic patterns that reduce throughput, and the switch has the same throughput as an ideal output-queued switch. In fact, the average packet delay through the switch is within a constant from that of an ideal output-queued switch (proof in Appendix III).
- *FOFF is practical to implement.* Each stage requires N queues. The first and last stage hold at most $N^2 + 1$ packets per linecard (the second stage holds the congestion buffer, and its size is determined by the same factors as in a shared-memory work-conserving router). The FOFF scheme is decentralized, uses only local information, and does not require complex scheduling.
- *Priorities in FOFF are practical to implement.* It is simple to extend FOFF to support k priorities using $k \cdot N$ queues in each stage. These queues could be used to distinguish different service levels, or could correspond to sub-ports.

We now move on to solve the final problem with the load-balanced switch.

VI. Flexible Linecard Placement

Designing a router based on the load-balanced switch is made challenging by the need to support non-uniform placement of linecards. If all the linecards were always present and working, they could be simply interconnected by a uniform mesh of fibers or wavelengths as shown in Figure 5. But if some linecards are missing, or have failed, the switch fabric needs to be reconfigured so as to spread the traffic uniformly over the remaining linecards. To illustrate the problem, imagine that we remove all but two linecards from a load-balanced switch based on a uniform mesh. When all linecards were present, the input linecards spread data over N center-stage linecards, at a rate of $2R/N$ to each. With only two remaining linecards, each must spread over both linecards, increasing the rate to $2R/2 = R$. This means that the switch fabric must now be able to interconnect linecards over a range of rates from $2R/N$ to R, which is impractical (in our design example $R = 160Gb/s$).

The need to support an arbitrary number of linecards is a real problem for network operators who want the flexibility to add and remove linecards when needed. Linecards fail, are added and removed, so the set of operational linecards changes over time. For the router to work when linecards are connected to arbitrary ports, we need some kind of reconfigurable switch to scatter the traffic uniformly over the linecards that are present. In what follows, we'll describe two architectures that accomplish this. As we'll see, it requires quite a lot of additional complexity over and above the simple single mesh.

A. Partitioned Switch

To create a 100Tb/s switch with 640 linecards, we need to partition the switch into multiple stages. Fortunately, partitioning a load-balanced switch is easier than partitioning a crossbar switch, since it does not need to be completely non-blocking in the conventional sense; it just needs to operate as a uniform fully-interconnected mesh.

To handle a very large number of linecards, the architecture is partitioned into G groups of L linecards. The groups are connected together by M different $G \times G$ middle stage switches. The middle stage switches are statically configured, changing only when a linecard is added, removed or fails. The linecards within a group are connected by a local switch (either optical or electrical) that can place the output of each linecard on any one of M output channels and can connect M input channels to any

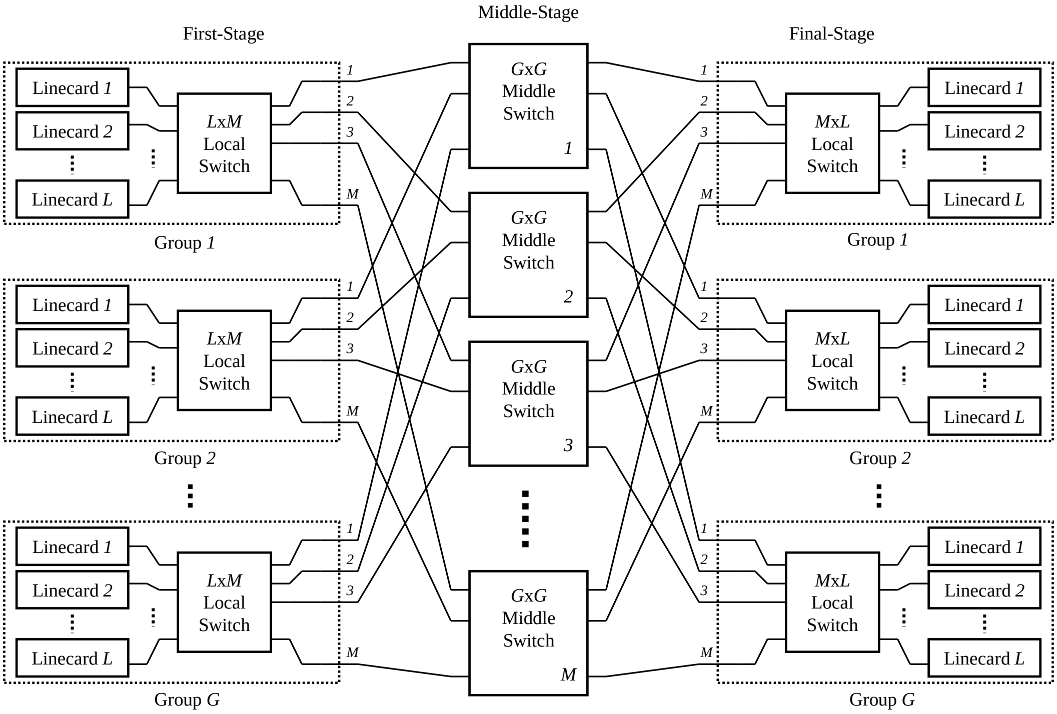


Fig. 6. Partitioned switch fabric.

linecard in the group. Each of the M channels connects to a different middle stage switch, providing M paths between any pair of groups. This is shown in Figure 6. The number M depends on the uniformity of the linecards in the groups. For uniform linecard placement, the middle switches need to distribute the output from each group to all the other groups, which requires G middle stage switches.[8] In this simplified case $M = G$, i.e. there is one path between each pair of groups. Each group sends $1/G$-th of its traffic over each path to a different middle-stage switch to create a uniform mesh. The first middle-stage switch statically connects input 1 to output 1, input 2 to output 2, and so on. Each successive switch rotates its configuration by one; for example, the second switch connects input 1 to output 2, input 2 to output 3, and so on. The path between each pair of groups is subdivided into L^2 streams; one for each pair of linecards in the two groups. The first-stage local switch uniformly spreads traffic, packet-by-packet, from each of its linecards over the path to another group; likewise, the final-stage local switch spreads the arriving traffic over all of the linecards in its group. The spreading is therefore hierarchical: The first-stage allows the linecards in a group to spread their outgoing packets over the G outputs; the middle-stage interconnects groups; and the final-stage spreads the incoming traffic from the G paths over the L linecards.

The uniform spreading is more difficult when linecards are not uniform, and the solution is to increase the number of paths M between the local switches.

Theorem 1: We need at most $M = L + G - 1$ static paths, where each path can support a rate up to $2R$, to spread traffic uniformly over any set of $n \leq N = G \times L$ linecards that are present so that each pair of linecards are connected at rate $2R/n$.

The theorem is proved formally in [2], but it is easy to show an example where this number of paths is needed. Consider the case when the first group has L line cards, but all the other groups have just one linecard. A uniform spreading of data among the groups would not be correct. The first group needs to send and receive a larger fraction of the data. The simple way to handle this is to increase the number of paths, M, between groups by increasing the number of middle-stage switches, and by increasing the number of ports on the local switches. If we add an additional path for the each linecard that is out of balance, we can again use the middle-stage switches to spread the data. Since the maximum imbalance is $L - 1$, we need to have $M = L+G-1$ paths through the middle switch. In the example given, the extra paths are routed to the first group (which is full), so now the data is distributed as desired, with $L/(L + G - 1)$ of the data arriving at the first group.

The remaining issue is that the path connections depend on the particular placement of the linecards in the groups, so they must be flexible and change when the configuration of the switch changes. There are two ways of building this flexibility. One uses MEMS devices as an optical patch-panel in conjunction with electrical crossbars, while the other uses multiple wavelengths, MEMS and optical couplers to create the switch.

B. Hybrid Electro-Optical Switch

The electro-optical switch is a straightforward implementation of the design described above. As before, the architecture is arranged as G groups of L linecards. In the center, M statically configured $G \times G$ MEMS switches interconnect the G groups. The MEMS switches are reconfigured only when

[8] Strictly speaking, this requires that $G \geq L$ if each channel is constrained to run no faster than $2R$.

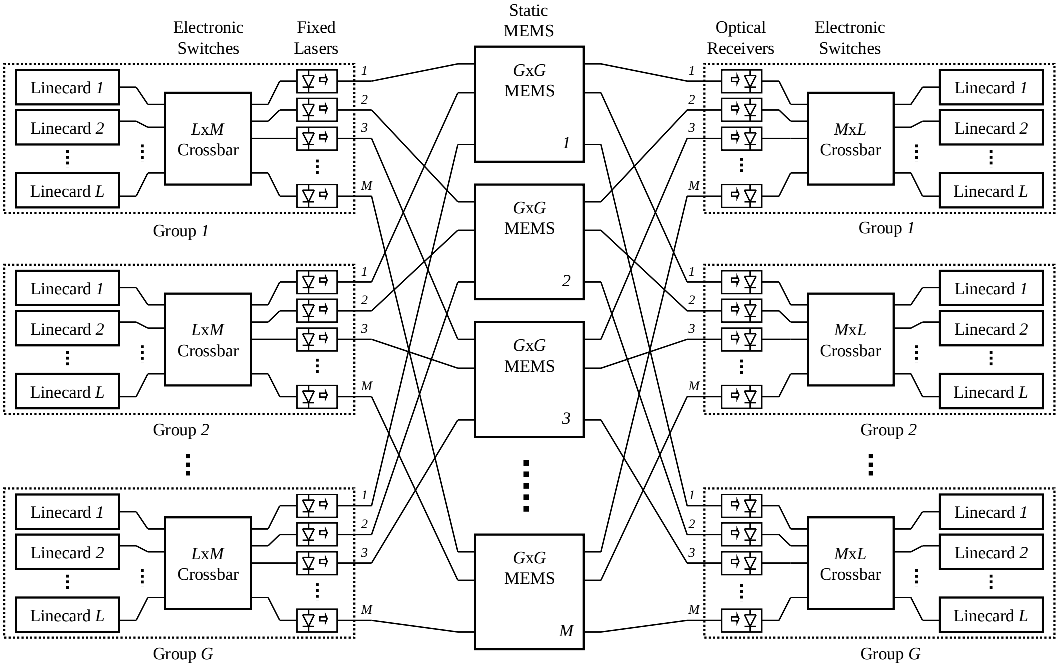


Fig. 7. Hybrid optical and electrical switch fabric.

a linecard is added or removed and provide the ability to create the needed paths to distribute the data to the linecards that are actually present. This is shown in Figure 7. Each group of linecards spreads packets over the MEMS switches using an $L \times M$ electronic crossbar. Each output of the electronic crossbar is connected to a different MEMS switch over a dedicated fiber at a fixed wavelength (the lasers are not tunable). Packets from the MEMS switches are spread across the L linecards in a group by an $M \times L$ electronic crossbar.

We need an algorithm to configure the MEMS switches and schedule the crossbars. Because the switch has exactly the number of paths we need, and no more, the algorithm is quite complicated, and is beyond the scope of this paper. A description of the algorithm, and proof of the following theorem appears in [2].

Theorem 2: There is a polynomial-time algorithm that finds a static configuration for each MEMS switch, and a fixed-length sequence of permutations for the electronic crossbars to spread packets over the paths.

C. Optical Switch

Building an optical switch that closely follows the electrical hybrid is difficult since we need to independently control both of the local switches. If we used an AWGR and wavelengths as the local switches, they could not be independently controlled. Instead, we modify the problem by allowing each linecard to have L optical outputs, where each optical output uses a tunable laser. Each of the $L \times L$ outputs from a group goes to a passive star coupler that combines it with the similar output from each of the other groups. This organization creates a large ($L \times G$) number of paths between the linecards; the output fiber on the linecard selects which linecard in a group the data is destined for and the wavelength of the light selects one of the G groups. It might seem that this solution is expensive, since it multiplies the number of links by L. However, the high line rates ($2R = 320Gb/s$) will force the use of parallel optical channels in any architecture, so the cost in optical components is smaller than it might seem.

Once again, the need to deal with unbalanced groups makes the switch more complex than the uniform design. The large number of potential paths allows us to take a different approach to the problem in this case. Rather than dealing with the imbalance, we logically move the linecards into a set of balanced positions using MEMS devices and tunable filters. This organization is shown in Figure 8. Again, consider our example in which the first group is full, but all of the other groups have just one linecard. Since the star couplers broadcast all the data to all the groups, we can change the effective group a card sits in by tuning its input filter. In our example we would change all the linecards not in the first group to use the second wavelength, so that effectively all the single linecards are grouped together as a full second group. The MEMS are then used to move the position of these linecards so they do not occupy the same logical slot position. For example, the linecard in the second group will take the 1st logical slot position, the linecard in the third group will take the 2nd logical slot position, and so on. Together these rebalance the arrangement of linecards and allows the simple distribution algorithm to work.

VII. Practicality of 100Tb/s Router

It is worth asking: Can we build a 100Tb/s router using this architecture, and if so, could we package it in a way that network operators could deploy in their network?

We believe that it is possible to build the 100Tb/s hybrid electro-optical router in three years. The system could be packaged in multiple racks as shown in Figure 3, with $G = 40$ racks each containing $L = 16$ linecards, interconnected by $L + G - 1 = 55$ statically configured 40×40 MEMS switches.

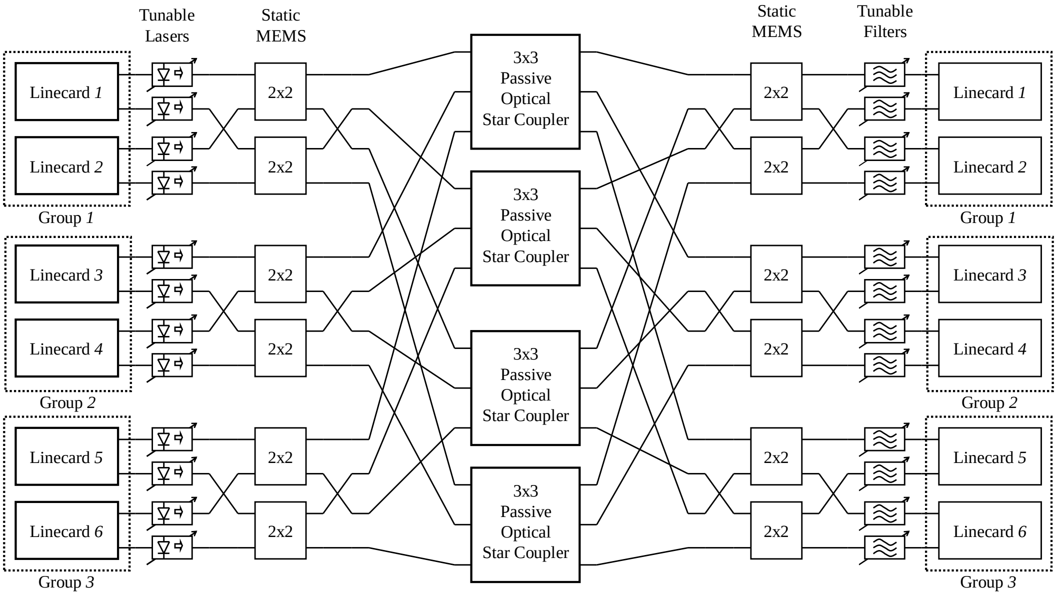


Fig. 8. An optical switch fabric for $G = 3$ groups with $L = 2$ linecards per group.

To justify this, we will break the question down into a number of smaller questions. Our intention is to address the most salient issues that a system designer would consider when building such a system. Clearly our list cannot be complete. Different systems have different requirements, and must operate in different environments. With this caveat, we consider the following different aspects.

A. The Electronic Crossbars

In the description of the hybrid electro-optical switch, we assumed that one electronic crossbar interconnects a group of linecards, each at rate $2R = 320$Gb/s. This is too fast for a single crossbar, but we can use bit-slicing. We'll assume W crossbar slices, where W is chosen to make the serial link data-rate achievable. For example, with $W = 32$, the serial links operate at a more practical 10Gb/s. Each slice would be a 16×55 crossbar operating at 10Gb/s. This is less than the capacity of crossbars that have already been reported [29].

Figure 9 shows L linecards in a group connected to W crossbar slices, each operating at rate $2R/W$. As before, the outputs of the crossbar slices are connected to lasers. But now, the lasers attached to each slice operate at a different, fixed wavelength, and data from all the slices to the same MEMS switch are multiplexed onto a single fiber. As before, the group is connected to the MEMS switches with M fibers. If a packet is sent on the n-th crossbar slice, it will be delivered to the n-th crossbar slice of the receiving group. Apart from the use of slices to make a parallel datapath, the operation is the same as before.

Each slice would connect to $M = 55$ lasers or optical receivers. This is probably the most technically challenging, and interesting, design problem for this architecture. One option is to connect the crossbars to external optical modules, but might lead to prohibitively high power consumption in the electronic serial links. We could reduce power if we could directly connect the optical components to the crossbar chips. The direct attachment (or "solder bumping") of III-V opto-electronic devices onto silicon has been demonstrated [30], but is not yet a mature, manufacturable technology, and is an area of continued research and exploration by us, and others. Another option is to attach optical modulators rather than lasers. An external, high powered continuous wave laser source could illuminate an array of integrated modulators on the crossbar switch. The array of modulators modulate the optical signal and couple it to an outgoing fiber [31].

B. Packaging 100Tb/s of MEMS Switches

We can say with confidence that the power consumption of the optical switch fabric will not limit the router's capacity. Our architecture assumes that a large number of MEMS switches are packaged centrally. Because they are statically configured, MEMS switches consume almost no power, and all 100Tb/s of switching can be easily packaged in one rack using commercially available MEMS switches today. Compare this with a 100Tb/s electronic crossbar switch, that connects to the linecards using optical fibers. Using today's serial link technology, the electronic serial links alone would consume approximately 8kW (assume 400mW and 10Gb/s per bidirectional serial link). The crossbar function would take at least 100 chips, requiring multiple extra serial links between them; hence the power would be much higher. Furthermore, the switch needs to terminate over 20,000 optical channels operating at 10Gb/s. Today, with commercially available optical modules, this would consume tens of kilowatts, would be unreliable and prohibitively expensive.

C. Fault-Tolerance

The load-balanced architecture is inherently fault-tolerant. First, because it has no centralized scheduler, there is no electrical central point of failure for the router. The only centrally shared devices are the statically configured MEMS switches,

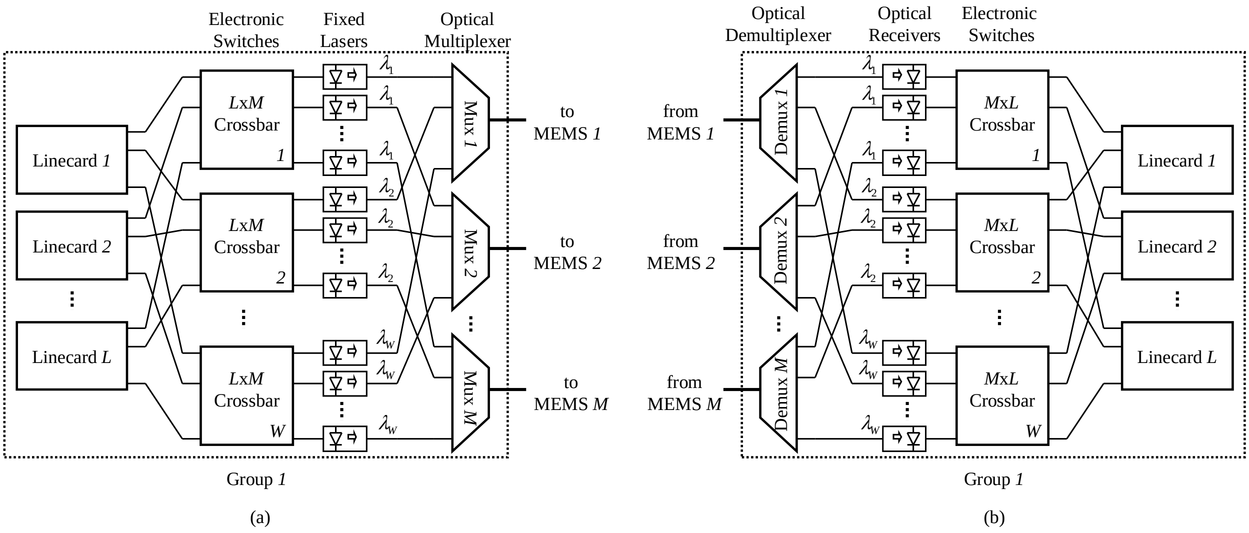


Fig. 9. Bit-sliced crossbars for hybrid optical and electrical switch. (a) represents the transmitting side of the switch. (b) represents the receiving side of the switch.

which can be protected by extra fibers from each linecard rack, and spare MEMS switches. Second, the failure of one linecard will not make the whole system fail; the MEMS switches are reconfigured to spread data over the correctly functioning linecards. Third, the crossbars in each group can be protected by an additional crossbar slice.

D. Building 160Gb/s Linecards

We assume that the address lookup, header processing and buffering on the linecards are all electronic. Header processing will be possible at 160Gb/s using electronic technology available within three years. At 160Gb/s, a new minimum length 40-byte packet can arrive every 2ns, which can be processed quite easily by a pipeline in dedicated hardware. 40Gb/s linecards are already commercially available, and anticipated reductions in geometries and increases in clock speeds will make 160Gb/s possible within three years.

Address lookups are challenging at this speed, but it will be feasible within three years to perform pipelined lookups every 2ns for IPv4 longest-prefix matching. For example, one could use 24Mbytes of 2ns SRAM (Static RAM)[9] and the brute force lookup algorithm in [32] that completes one lookup per memory reference in a pipelined implementation.

The biggest challenge is simply writing and reading packets from buffer memory at 160Gb/s. Router linecards contain 250ms or more of buffering so that TCP will behave well when the router is a bottleneck, which requires the use of DRAM (dynamic RAM). Currently, the random access time of DRAMs is 40ns (the duration of twenty minimum length packets at 160Gb/s!), and historically DRAMs have increased in random access speed by only 10% every 18 months. We have solved this problem in other work by designing a packet buffer using commercial memory devices, but with the speed of SRAM and the density of DRAM [33]. This technique makes it possible to build buffers for 160Gb/s linecards.

E. Packaging 16 Linecards in a Rack

Network operators frequently complain about the power consumption of 10Gb/s and 40Gb/s linecards today (200W per linecard is common). If a 160Gb/s linecard consumes more power than a 40Gb/s linecard today, then it will be difficult to package 16 linecards in one rack ($16 \times 200 = 3.2kW$). If improvements in technology don't solve this problem over time, we can put fewer linecards in each rack, so long as $G \times L \geq 640$. For example, we could halve the number of linecards per rack and double the number of groups. This comes at the expense of more MEMS switches ($M \geq L + G - 1$).

VIII. Conclusion

Our main conclusion is that we achieved what we set out to do: To solve the problems that allow us to design a 100Tb/s load-balanced router, with guaranteed 100% throughput under all traffic conditions. We believe that the electro-optic router we described, including switch fabric and linecards, can be built using technology available within three years, and fit within the power constraints of network operators.

To achieve our capacity requirement, optics are necessary. A 100Tb/s router needs to use multiple racks regardless of the architecture, because packets need to be processed and stored electronically, and the power consumption of the electronics is too much for a single rack. Optical links are needed to interconnect the multiple racks, but we don't have to use an optical switch fabric. A distributed switch fabric could spread the switching function over all the racks. The key problem is that a distributed switch fabric could not guarantee the system throughput. But if throughput guarantees are not needed, a distributed switch fabric (such as a torus or hypercube) might be a reasonable alternative.

[9] Today, the largest commercial SRAM is 4Mbytes with an access time of 4ns, which suggests what is feasible for on-chip SRAM. Moore's Law suggests that in three years 16Mbyte SRAMs will be available with a pipelined access time below 2ns. So 24Mbytes can be spread across two physical devices.

Providing throughput guarantees would normally require a complicated centralized scheduler to configure the switch fabric. The main advantages of the load-balanced switch are that: (1) It allows us to eliminate the centralized scheduler without sacrificing the throughput guarantees, and (2) It allows us to use a switch fabric that doesn't need to be frequently reconfigured. More precisely, in the hybrid electro-optic switch fabric only the lower-capacity local switches in each group need to be reconfigured frequently — the high capacity MEMS switches change only when linecards are added or removed. The switch fabric is essentially transparent, consumes no power, and eliminates power-hungry conversions between the electrical and optical domain. All the central switching can be packaged in a single rack.

In future, as optical technology matures, it will be possible to replace the hybrid electro-optical switch with an all-optical fabric. This has the potential to reduce power further by eliminating many electronic crossbars and serial links.

REFERENCES

[1] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard and N. McKeown , "Scaling Internet routers using optics," *ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.

[2] I. Keslassy, S-T. Chuang, and N. McKeown, "A load-balanced switch with an arbitrary number of linecards," *Stanford University HPNG Technical Report - TR03-HPNG-080102*, Stanford, CA, Aug. 2003.

[3] I. Keslassy, S-T. Chuang, and N. McKeown, "Architectures and algorithms for a load-balanced switch," *Stanford University HPNG Technical Report - TR03-HPNG-061501*, Stanford, CA, Aug. 2003.

[4] A. M. Odlyzko, "The current state and likely evolution of the Internet," *Proc. Globecom'99*, pp. 1869-1875, 1999.

[5] A. M. Odlyzko, "Comments on the Larry Roberts and Caspian Networks study of Internet traffic growth," *The Cook Report on the Internet*, pp. 12-15, Dec. 2001.

[6] Pat Blake, "Resource," *Telephony*, Feb. 2001, available at http://telephonyonline.com/ar/telecom_resource/ index.htm

[7] Router Capacity: Raw data http://www.stanford.edu/~nickm/RouterCapacity.xls

[8] N. McKeown, C. Calamvokis, S.-T. Chuang, "A 2.5Tb/s LCS switch core," *Hot Chips XIII*, Aug. 2001.

[9] K. Y. Yun, "A terabit multiservice switch," *IEEE Micro*, Vol. 21, pp. 58-70, Jan.-Feb. 2001.

[10] Alcatel, "Alcatel's packet-based digital cross-connect switch application," July 2002, available at http://www.alcatel.com.

[11] PMC-Sierra Inc., "Tiny-Tera one chip set," April 2000, available at http://www.pmc-sierra.com/pressRoom/chess.html.

[12] Juniper Networks, "The essential core: Juniper Networks T640 Internet routing node with matrix technology," April 2002, available at http://www.juniper.net/solutions/literature/ solutionbriefs/351006.pdf.

[13] W. J. Dally, "Architecture of the Avici terabit switch router," *Proc. Hot Interconnects XIII*, Aug. 1998.

[14] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. on Comm.*, Vol.47, No.8, Aug. 1999.

[15] J.G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *Proc. of the IEEE INFOCOM*, Vol. 2, pp. 556-564, Tel Aviv, Israel, March 2000.

[16] Y. Tamir and H.C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13-27, 1993.

[17] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "The Tiny Tera: A packet switch core," *Proc. of Hot Interconnects V*, Aug. 1996.

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

[19] C.-S. Chang, D.-S. Lee and Y.-S. Jou, "Load balanced Birkhoff-von Neumann switches, Part I: one-stage buffering," *Computer Comm.*, Vol. 25, pp. 611-622, 2002.

[20] L.G. Valiant and G.J. Brebner, "Universal schemes for parallel communication," *Proc. of the* $13th$ *ACM Symposium on Theory of Computation*, pp. 263-277, 1981.

[21] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, June 1995, available at http://www.faqs.org/rfcs/rfc1812.html.

[22] P. Bernasconi, C. Doerr, C. Dragone, M. Capuzzo, E. Laskowski and A. Paunescu, "Large N x N waveguide grating routers", *Journal of Lightwave Technology*, Vol. 18, No. 7, pp. 985-991, July 2000.

[23] K. Sato, "Semiconductor light sources for 40-Gb/s transmission systems," *Journal of Lightwave Technology*, vol.

[24] R. Ryf *et al.*, "1296-port MEMS transparent optical crossconnect with 2.07 petabit/s switch capacity," *Optical Fiber Comm. Conf. and Exhibit (OFC) '01*, Vol. 4, pp. PD28 -P1-3, 2001.

[25] G.I. Papadimitriou, C. Papazoglou, and A.S. Pomportsis, "Optical switching: switch fabrics, techniques, and architectures," *Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384-405, Feb. 2003.

[26] B. Pesach et al., "Free-space optical cross-connect switch by use of electroholography," *Applied Optics*, Vol. 39, No. 5, pp. 746-758, Feb. 2000.

[27] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," *Proc. of the IEEE Infocom*, June 2002.

[28] C.-S. Chang, D.-S. Lee and C.-M. Lien, "Load balanced Birkhoff-von Neumann switches, Part II: multi-stage buffering," *Computer Comm.*, Vol. 25, pp. 623-634, 2002.

[29] B. Dally, Velio Communications, "A single chip terabit switch," *Hot Chips XIII*, Stanford, Aug. 2001.

[30] A. Krishnamoorthy, "Optoelectronics flip-chip-bonded to CMOS VLSI circuits," *IMAPS Advanced Technology Workshop on Next Generation IC and Package Design*, Solvang, CA, July 1999.

[31] A. L. Lentine *et al.*, "Arrays of optoelectronic switching nodes comprised of flip-chip-bonded MQW modulators and detectors on silicon CMOS circuitry," *IEEE Photonics Technology Letters*, Vol. 8, pp. 221-223, Feb. 1996.

[32] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proc. of IEEE INFOCOM '98*, pp. 1240-1247, San Francisco, CA, April 1998.

[33] S. Iyer, R. R. Kompella, and N. McKeown, "Designing buffers for router line cards," *Stanford University HPNG Technical Report - TR02-HPNG-031001*, Stanford, CA, Mar. 2002.

[34] J. Gripp *et al.*, "Demonstration of a 1.2 Tb/s optical packet switch fabric" *27th European Conf. on Optical Comm. (ECOC) '01*, Vol. 6, Amsterdam, The Netherlands, Oct. 2001.

[35] N. Kikuchi, "Error-free signal selection and high-speed channel switching by monolithically integrated 64-channel WDM channel selector," *Electronics Letters*, Vol. 38, No. 15, pp. 823-824, July 2002.

[36] C.-S. Chang, "Performance guarantees in communication networks," *Springer-Verlag*, 2000.

APPENDIX I
GLOSSARY OF OPTICAL COMPONENTS

- *MEMS Switches* - optical equivalent of a crossbar using micromirrors to reflect optical beams from inputs to outputs and are transparent to wavelength and data-rate. Typical reconfiguration times are 1-10ms [24], [25].
- *Tunable Lasers* - lasers that can transmit light at different wavelengths. Tuning times of 10ns have been demonstrated using commercial devices [34].
- *Tunable Filters* - optical detectors that detect channel information at selected wavelengths. Devices have been demonstrated with tuning times of 10ns [35].
- *Passive Optical Star Coupler* - light from N incoming fibers is combined and broadcast to N outgoing fibers with an intrinsic $1/N$ power loss.

APPENDIX II
PROOF THAT PACKETS LEAVE THE SWITCH IN ORDER

A. Intuition on the FOFF scheme

FOFF has three distinctive features that will be used in the proof. First, for any given flow, packets leave the first stage in order. Second, this scheme is work-conserving for frames, in the sense that every N time-slots, if there is at least one full frame remaining, then N packets will be served in the next frame. Finally, the third feature is that if there is no full frame left, then flows are served in round-robin, which avoids starvation when throughput is strictly less than 1.

B. Assumptions

Let's assume that we take a snapshot of the switch every N time-slots. We will denote this period of N time-slots as *a frame slot*. During every frame slot, each input can send at most one packet to each intermediate input. Similarly, each intermediate input can send at most one packet to each output. We will assume that at each frame slot, the first input sends first all its packets to the intermediate inputs, then the second input does so, and so on. Of course, the switch needs not operate this way, but this is easier to understand conceptually.

Here is a way of implementing this order using shift registers and no memory speed-up. At time slots i, $N+i$, $2N+i$, ..., input linecard i selects an output destination. Let's assume that this output is k. Over the next N time slots, input linecard i sends packets destined to output k in a round-robin order to the N intermediate input linecards. Specifically, in time slot $i+j-1$, input linecard i sends to intermediate input linecard j a packet destined to output k. Then, the incoming packets to intermediate input linecard j are delayed by $N-j$ time slots. Similarly, in time slot $j+k-1$, intermediate input linecard j sends a packet destined to output linecard k. Once again, the incoming packets to output linecard k are delayed by $N-k$ time slots. This scheme guarantees the *frame slot* snapshot model defined above with a fixed delay of N time slots for a packet to be sent from one linecard stage to the next.

For implementation purposes, it is also possible to remove the delay elements and evaluate the queue occupancy in each linecard in staggered time slots.

Finally, by convention, we will assume that for every frame slot, the following order occurs: arrivals to the inputs, departures from the inputs, arrivals to the intermediate inputs, and so on.

C. Theorem

A feature of the FOFF algorithm is that each of the three stages can be implemented using only N queues. This is clear for the first two stages. For the third stage, the re-sequencing buffer can be implemented using at most N^2 cells arranged into N queues, one for each intermediate input. It is interesting to note that the N queues on output linecard k is an extension of VOQ k in the N intermediate input linecards. The following theorem shows how the re-sequencing buffer can be implemented with only N queues:

Theorem 3: If there are N^2 packets in the queue, the head-of-line packet of at least one of the N queues can be sent while keeping packets ordered. Therefore, we need a resequencing buffer size of at most N^2 in order for packets to leave the switch in order.

D. Notations and Proofs

Let's introduce notations and prove consecutively several lemmas before proving the theorem. In the lemmas, we will first show that inputs send approximately the same number of packets to each intermediate input. This results in an occupancy that is approximately the same in all intermediate inputs. Therefore, the mis-sequencing, which results from a difference of queue occupancy between intermediate inputs, can be bounded. This finally results in the fact that the resequencing buffer size at the output can also be bounded.

In the remainder, we will assume that there is a fixed propagation delay d between the inputs and the intermediate inputs, and similarly between the intermediate inputs and the outputs, to model the switch fabric latency. This delay d will change depending on the implementation chosen.

Let $S(i,k)$ denote the flow from input i to output k, i.e. the set of all packets going from i to k, with $1 \leq i,k \leq N$. Similarly, let $S(i,j,k)$ be the set of all packets going through input i, intermediate input j and output k, with $1 \leq i,j,k \leq N$. Obviously, $S(i,k) = \bigcup S(i,j,k)$.

As shown in Figure 10, let the couples ($A_{ijk}(t)$, $B_{ijk}(t)$), ($C_{ijk}(t)$, $D_{ijk}(t)$) and ($E_{ijk}(t)$, $F_{ijk}(t)$)) denote the cumulative number of (arrivals, departures) of packets in $S(i,j,k)$ at the end of frame slot t for input i, intermediate input j and output k. For instance, $B_{ijk}(t) - A_{ijk}(t)$ represents the number of packets from $S(i,j,k)$ stored in input linecard i. Given the definitions above, we have the following equations.

First, by causality, $A_{ijk}(t) \geq B_{ijk}(t) \geq C_{ijk}(t) \geq D_{ijk}(t) \geq E_{ijk}(t) \geq F_{ijk}(t)$.

Second, the delay d brings $C_{ijk}(t) = B_{ijk}(t-d)$ and $E_{ijk}(t) = D_{ijk}(t-d)$.

Third, in input i, arriving packets that belong to $S(i,k)$ are sent in round-robin order among all intermediate linecards, starting with intermediate linecard 1. Formally, if $A_{S(i,k)}(t)$ is the cumulative number of arriving packets that belong to $S(i,k)$, we have for all j [28]:

$$A_{ijk}(t) = \left\lceil \frac{A_{S(i,k)}(t) + 1 - j}{N} \right\rceil .$$

Therefore, for each (i,k), there exists some intermediate input j' such that: $A_{i1k}(t) = A_{i2k}(t) = ... = A_{ij'k}(t) = A_{i,j'+1,k}(t) + 1 = A_{i,j'+2,k}(t) + 1 = ...$ Similarly, since packets from a given flow are sent in order, we have a j'' such that $B_{i1k}(t) = B_{i2k}(t) = ... = B_{ij''k}(t) = B_{i,j''+1,k}(t) + 1 = B_{i,j''+2,k}(t) + 1 = ...$ And using the property above, $C_{i1k}(t-d) = C_{i2k}(t-d) = ... = C_{ij''k}(t-d) = C_{i,j''+1,k}(t-d)+1 = C_{i,j''+2,k}(t-d) + 1 = ...$ This directly proves the following lemma:

Lemma 4: For each flow, the difference in the cumulative number of arrivals to two different intermediate inputs is bounded by 1. In addition, if an intermediate input has more arrivals than another one, then its index must be lower.

Let $Q_{jk}(t)$ be the occupancy of the VOQ at intermediate input j for output k at the end of frame slot t. Given Lemma 4, the next lemma will bound the difference in occupancy for a given output between different intermediate input linecards.

Lemma 5: For all intermediate inputs j_1, j_2, output k and frame slot t: $|Q_{j_1k}(t) - Q_{j_2k}(t)| \leq N$.

Proof: Assume without loss of generality that $j_1 < j_2$. Since the link between intermediate input j_1 and output k is

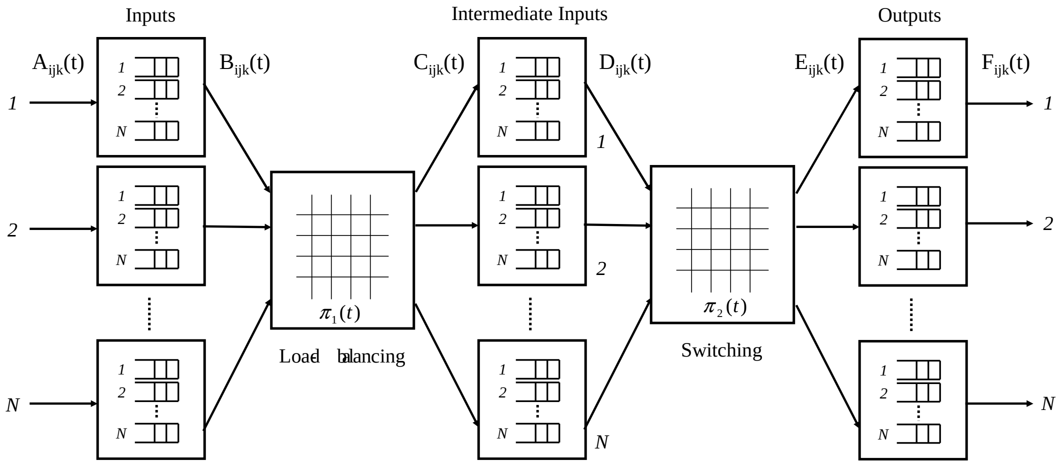


Fig. 10. Proof Notations

work-conserving at each frame slot, we have [36]:

$$Q_{j_1k}(t) = \max_{0\leq s\leq t}(\sum_i [C_{ij_1k}(t) - C_{ij_1k}(s)] - (t-s)).$$

For instance, this maximum could be reached in s':

$$Q_{j_1k}(t) = \sum_i [C_{ij_1k}(t) - C_{ij_1k}(s')] - (t-s').$$

Similarly,

$$\begin{aligned} Q_{j_2k}(t) &= \max_{0\leq s\leq t}(\sum_i [C_{ij_2k}(t) - C_{ij_2k}(s)] - (t-s)) \\ &= \sum_i [C_{ij_2k}(t) - C_{ij_2k}(s'')] - (t-s''). \end{aligned}$$

The first consequence is:

$$\begin{aligned} Q_{j_2k}(t) &\geq \sum_i [C_{ij_2k}(t) - C_{ij_2k}(s')] - (t-s') \\ &= \sum_i [C_{ij_2k}(t) - C_{ij_2k}(s')] + \{Q_{j_1k}(t) - \\ &\quad \sum_i [C_{ij_1k}(t) - C_{ij_1k}(s')]\} \\ &= Q_{j_1k}(t) + \sum_i \{[C_{ij_2k}(t) - C_{ij_1k}(t)] + \\ &\quad [C_{ij_1k}(s') - C_{ij_2k}(s')]\} \\ &\geq Q_{j_1k}(t) + \sum_i \{-1 + 0\} \\ &= Q_{j_1k}(t) - N. \end{aligned}$$

Similarly, the second consequence is:

$$\begin{aligned} Q_{j_1k}(t) &\geq Q_{j_2k}(t) + \sum_i \{[C_{ij_1k}(t) - C_{ij_2k}(t)] + \\ &\quad [C_{ij_2k}(s'') - C_{ij_1k}(s'')]\} \\ &\geq Q_{j_2k}(t) + \sum_i \{0 - 1\} \\ &= Q_{j_2k}(t) - N. \end{aligned}$$

Hence $|Q_{j_1k}(t) - Q_{j_2k}(t)| \leq N$. ■

Since packets from a given flow can traverse different intermediate input linecards, it is possible that packets will arrive to the output linecards out-of-order. The next lemma bounds the amount of mis-sequencing for a given flow.

Lemma 6: If two packets p_1 and p_2 belong to $S(i,k)$, and p_1 arrives to the switch before p_2, then p_1 will arrive to output k at most N frame slots after p_2.

Proof: If a packet p_1 from $S(i,k)$ arrives to intermediate input j_1 at t_1, it will leave the intermediate input at $t_1 + Q_{j_1k}(t_1)$ because of the work-conserving property mentioned above. Since packets of the same flow leave the first stage in order, a packet p_2 from $S(i,k)$ arriving later to input i will thus arrive to intermediate input j_2 at $t_2 \geq t_1$. As a result, p_2 will leave j_2 at

$$\begin{aligned} t_2 + Q_{j_2k}(t_2) &= t_1 + Q_{j_2k}(t_1) + \\ &\quad [t_2 - t_1 + Q_{j_2k}(t_2) - Q_{j_2k}(t_1)] \\ &\geq t_1 + Q_{j_2k}(t_1) \end{aligned}$$

(by the work-conserving property). Hence p_2 will not leave the intermediate input before $t_1 + Q_{j_1k}(t_1) + (Q_{j_2k}(t_1) - Q_{j_1k}(t_1)) \geq t_1 + Q_{j_1k}(t_1) - N$. Therefore p_1 will leave the second stage at most N frame slots after p_2 leaves. ■

Let's now prove the theorem. In the proof, we will call *head of flow* the first packet of a given flow that has not yet left the switch.

Proof: As assumed above, at output k, during a given frame slot, we first have up to N arrivals from the N intermediate inputs, and then we have up to N departures. We will of course assume that packets can only depart in order.

Let's show that whenever the number of packets queued in a given output k is strictly bigger than N^2, the output is work-conserving, i.e. the head-of-line packet of at least one of the N queues can depart while keeping packets ordered. Since arrivals cannot exceed departures when the output is work-conserving, this would clearly show that the queue occupancy of output k is bounded by N^2. As a consequence, we would only need a resequencing buffer size of at most N^2 in order for packets to leave the switch in order.

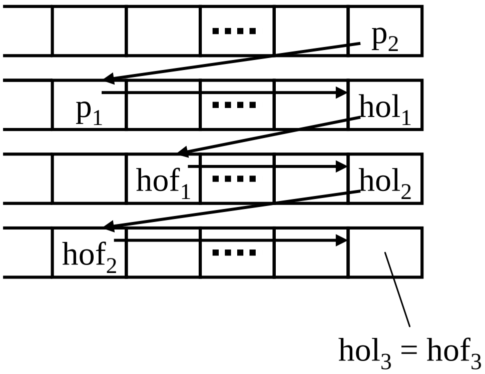


Fig. 11. View of the resequencing buffer.

Let's assume that there are more than N^2 packets queued in a given output k at frame slot t. Since at most N packets could arrive to output k in one frame slot, by the pigeonhole principle at least one of the packets queued did not arrive in the last N frame slots. Let's call this packet p_2, and let i be such that $p_2 \in S(i, k)$. Using Lemma 6, the *head of flow* for p_2 is also necessarily queued in output k (call it p_1). Therefore, there is at least one *head of flow* queued in output k if there are more than N^2 packets in output k.

Let's now show that there is at least one head of flow in the switch that is also head of line of one of the queues in output k. This will prove that the head-of-line packet of at least one of the N queues can depart while keeping packets in order.

Given the above assumption, if p_1 is head-of-line for its intermediate input j_1, then p_1 can depart. Otherwise, consider the head-of-line for j_1, hol_1, which has already arrived to output k (illustrated in Figure 11). If hol_1 is a head of flow, it can depart. Otherwise, let hof_1 be the head of the flow of hol_1, which went through some intermediate input j_2. We know that hof_1 arrived to its intermediate input no later than hol_1, because they are from the same flow. In addition, hol_1 arrived before p_1, because they were in the same intermediate input. Finally, p_1 arrived no later than p_2 (same flow). Therefore, hof_1 reached the intermediate inputs before p_2. By Lemma 5, since p_2 arrived to output k by frame slot $t - N$, hof_1 must have arrived to output k by frame slot t. Thus, hof_1 must be queued in output k.

Again, if hof_1 is the head-of-line for its intermediate input j_2, hof_1 can depart. Otherwise, consider hol_2, the head of line for hof_1. If hol_2 is head of flow, it can depart. Otherwise, consider the head of flow for hol_2, hof_2, which went through some intermediate input j_3. By repeating the same method, we can continue considering the head of flow of the head of line for successive intermediate inputs $j_1, j_2, j_3, ...$. Since each new head of flow arrived strictly before the one previously considered, the method cannot visit the same intermediate input twice. Given that there are only N intermediate inputs, this method must thus converge, and thus there is always at least one packet that can depart while keeping packets in order. ■

Appendix III
Proof of Average Packet Delay

Theorem 7: The average packet delay through the switch is within a constant of the average packet delay through a (work-conserving) output-queued switch.

The proof of this theorem will rely on several lemmas. These lemmas will consider the effect of using delay lines on the departures from a work-conserving switch, and try to compare the load-balanced switch with work-conserving switches using delay lines.

Throughout the lemmas, we will denote by $B(t)_{\{A(t)\}}$ the cumulative number of departures in a switch with cumulative arrival traffic $A(t)$. For instance, we know that in a work-conserving switch (noted WC), the cumulative number of departures $B(t)^{WC}_{\{A(t)\}}$ satisfies:

$$B(t)^{WC}_{\{A(t)\}} = \min_{0 \le s \le t}[A(s) + t - s].$$

Throughout the proof of the theorem, we will consider some arrival sequence $A(t)$. Given this arrival sequence, we will assume that the average delay through a work-conserving switch is defined and finite, and will attempt to prove that, if defined, the average delay through the load-balanced switch is within a constant of the average delay for a work-conserving switch.

Also, we will say below that two switches S^1 and S^2 are *equal* if they have the same number of packet departures for the same arbitrary packet arrivals. (Obviously, they are not necessarily emulating each other, since the departure packet order might be different.) In other words, the two switches are equal if their respective cumulative numbers of departures satisfy:

$$B(t)^1_{\{A(t)\}} = B(t)^2_{\{A(t)\}}.$$

Similarly, a switch S^1 will be said to be *better than* another switch S^2 if for any arrival traffic $A(t)$, and for any time-slot, S^1 has at least as many departures as S^2. This will be noted as

$$B(t)^1_{\{A(t)\}} \ge B(t)^2_{\{A(t)\}}.$$

For instance, a work-conserving switch is known to be better than any switch of output capacity 1.

The following lemma will prove that it is possible to permute work-conserving switches and delay lines, and still get equal systems.

Lemma 8: Consider a first system S^1 comprising a work-conserving switch followed by delay lines of delay d, noted as $S^1 = (WC, DL(d))$ (where $DL(d)$ represents a Delay Line of duration $d \ge 0$). Similarly, consider a second system S^2 comprising delay lines of delay d followed by a work-conserving switch, noted as $S^2 = (DL(d), WC)$. Then S^1 and S^2 are equal.

Proof: Let A denote the cumulative number of arrivals to each switch, with $A(0) = 0$. Let B_1 (resp. B_2) denote the cumulative number of departures from each system.

For $t \ge d$, we know that

$$B_1(t) = B(t-d)^{WC}_{\{A(t)\}} = \min_{0 \le s \le t-d}[A(s) + (t-d) - s],$$

and

$$B_2(t) = B(t)^{WC}_{\{A(t-d)\}} = \min_{0 \le s \le t}[A(s-d) + t - s].$$

But then, using $u = s - d$, $B_2(t) = \min_{-d \le u \le t-d}[A(u) + t - (u + d)] = \min_{-d \le u \le t-d}[A(u) + (t-d) - u]$. Comparing this with the expression of $B_1(t)$, we thus just need to show that for $u < 0$, $A(u) + (t-d) - u \geq A(0) + (t-d) - 0$, in order to prove that $B_2(t) = B_1(t)$. Since for $u < 0$, $A(u) = A(0) = 0$, the result follows. ■

The following lemma will help us compare tandems of switches with tandems of work-conserving switches.

Lemma 9: Consider a system composed of two switches S^1 and S^2. Assume that for any arrival traffic $A(t)$, S^1 is better than WC, i.e.

$$B(t)^1_{\{A(t)\}} \geq B(t)^{WC}_{\{A(t)\}}.$$

Also assume that S^2 behaves better than WC when their arrivals are the departures from S^1. In other words, if the departures from S^1 are written as

$$D(t) = B(t)^1_{\{A(t)\}},$$

then

$$B(t)^2_{\{D(t)\}} \geq B(t)^{WC}_{\{D(t)\}}.$$

Then the tandem (S^1, S^2) behaves better than a tandem of two work-conserving switches (WC, WC).

Proof: We assumed that $B(t)^2_{\{D(t)\}} \geq B(t)^{WC}_{\{D(t)\}}$. The first term of the inequality denotes the departures from the tandem (S^1, S^2), while the second term denotes the departures from the tandem (S^1, WC). Thus, since we know that (S^1, S^2) is better than (S^1, WC), we only need to show that (S^1, WC) is better than (WC, WC) in order to prove the lemma. However, we also know that S^1 has more departures than WC given the same arrival pattern. These departures are then used as the arrivals to the second WC in both tandems. Since the more arrivals a WC switch has, the more departures it will have, (S^1, WC) will have ore departures than (WC, WC). This proves the lemma. ■

The following lemma considers a system that is only work-conserving when its queue is above a given capacity. This can be useful, for instance, in order to analyze a system that only services packets by bursts, as in [33].

Lemma 10: Consider a system that satisfies the following two properties. First, there exists a critical queue size Q_c such that the switch is always work-conserving when at least Q_c packets are queued. Second, whenever Q packets are queued with $Q < Q_c$, the switch takes at most T time-slots to send Q packets. Then the system is better than a tandem $(WC, DL(T))$ of a work-conserving switch and a delay line of delay T.

Proof: Let $Q(t)_{\{A(t)\}}$ (resp. $Q(t)^{WC}_{\{A(t)\}}$) represent the queue size of the system (resp. of a work-conserving switch) at time-slot t under arrivals $A(t)$.

Let's prove by contradiction that $B(t)^{WC}_{\{A(t)\}} - B(t)_{\{A(t)\}} \leq Q_c - 1$ for all t. In other words, assume that t_0 is the first time-slot when this inequality is not satisfied, and let's show that t_0 cannot exist. Then $B(t_0)^{WC}_{\{A(t)\}} - B(t_0)_{\{A(t)\}} \geq Q_c$ and $B(t_0-1)^{WC}_{\{A(t)\}} - B(t_0-1)_{\{A(t)\}} \leq Q_c - 1$. Also, given the work-conserving property, $B(t_0)^{WC}_{\{A(t)\}} \leq B(t_0-1)^{WC}_{\{A(t)\}} + 1$, with equality only if there is at least one packet queued in the work-conserving system to service at time t_0. Putting all these inequalities together, $Q_c - B(t_0)_{\{A(t)\}} \leq Q_c - B(t_0 - 1)_{\{A(t)\}}$, hence $B(t_0-1)_{\{A(t)\}} \geq B(t_0)_{\{A(t)\}}$. But since the cumulative number of departures cannot decrease, these two quantities have to be equal, and therefore there had to be at least one packet queued in the work-conserving system just before the departures at time t_0. As a consequence, the queue size in the system just before the departures at t_0 is $A(t_0) - B(t_0 - 1)_{\{A(t)\}} = [A(t_0) - B(t_0-1)^{WC}_{\{A(t)\}}] + [B(t_0-1)^{WC}_{\{A(t)\}} - B(t_0-1)_{\{A(t)\}}] \geq 1 + [Q_c - 1] = Q_c$. But then the system should be work-conserving by assumption, and thus $B(t_0-1)_{\{A(t)\}} \neq B(t_0)_{\{A(t)\}}$, hence contradicting the assumption.

We now know that $B(t)^{WC}_{\{A(t)\}} - B(t)_{\{A(t)\}} \leq Q_c - 1$ for all t. In addition, we know that the difference between these queue sizes can be serviced in at most T time-slots. (Note that the packets in the difference will always be available to send because $Q(t)_{\{A(t)\}} = A(t) - B(t)_{\{A(t)\}} \geq B(t)^{WC}_{\{A(t)\}} - B(t)_{\{A(t)\}}$.) As a consequence, $B(t)^{WC}_{\{A(t)\}} \leq B(t+T)_{\{A(t)\}}$, hence $B(t-T)^{WC}_{\{A(t)\}} \leq B(t)_{\{A(t)\}}$ for all t. ■

The following lemma reminds the fact that having more packets leave earlier implies having a lower average delay.

Lemma 11: Consider an arrival traffic sequence $A(t)$, such that the average delay for a work-conserving switch having these arrivals is defined. Assume that a given switch satisfies $B(t)_{\{A(t)\}} \geq B(t)^{WC}_{\{A(t)\}}$. Then the average delay for this switch is at most the average delay for the work-conserving switch.[10]

Proof: The proof is quite straightforward. First, when defined, the average packet delay does not depend on the packet order. This is because the total packet delay until some time t is just the integral of the difference between $A(t)$ and $B(t)_{\{A(t)\}}$ (resp. $B(t)^{WC}_{\{A(t)\}}$) between 0 and t. Without loss of generality, we can thus assume that the packet order is the same in both switches. Second, packets always leave the switch earlier than (or at the same time as) the work-conserving switch. Therefore, the result follows. ■

Let's now prove Theorem 7.

Proof: First, each input of the load-balanced switch services one packet per time-slot whenever it has at least one full frame of N packets, which necessarily happens whenever it has at least $N(N-1)+1$ packets because of the pigeonhole principle. In addition, whenever it has at most $Q = N(N-1)$ packets (before arrivals), it takes at most N^2 time-slots to service at least Q packets. Therefore, using Lemma 10, any input linecard is better than a tandem of a work-conserving switch and a delay line of delay N^2. But since any input can have at most one arrival and one departure per time-slot, the work-conserving switch is just a forwarding switch. Thus any input linecard is better than a delay line of delay N^2.

Similarly, we saw before that whenever an output linecard has at least $N^2 - N + 1$ packets before arrivals, it is work-conserving. In addition, for each packet p in the output linecard which arrived at $t - d$ (where $d \geq 0$), the corresponding head-of-flow packet will arrive no later than $t - d + N^2$ to the output linecard. Since there are at most N such packets for each d, either p or a corresponding packet arrived earlier from the same

[10] Without entering into technicalities, if the average delay for the switch doesn't exist, at least a limsup exists and satisfies the same properties. In the remainder of the proofs, by *average delay*, we will mean the limsup of the average delay.

flow will be serviced by $t-d+N^2$. As a result, it is possible to see that whenever the output linecard has at most $Q = N(N-1)$ packets (before arrivals), it takes at most N^2 time-slots to service at least Q packets. As above, any output linecard is thus better than a delay line of delay N^2.

Finally, consider the system formed by all packets destined to a given output k in the intermediate linecards. Again, if this set has $N^2 - N + 1$ packets, all intermediate inputs will be non-empty (because of Lemma 5), and thus the system will be work-conserving for output k. And as above, whenever this system has at most $Q = N(N-1)$ packets (before arrivals), it takes at most N^2 time-slots to service at least Q packets.

Therefore, we can use the above results to analyze the load-balanced switch. Consider the system formed by all packets destined to a given output k in the intermediate linecards. We found above that it behaves better in the load-balanced switch than a work-conserving switch followed by a delay line of N^2. In addition, the inputs and the output k add two delay lines of delay N^2 using their respective arrivals. We can then use Lemma 9 and the fact that $B(t)^{WC}_{\{A(t)\}}$ is independent of the input to which packets arrived in a work-conserving switch, in order to prove that the load-balanced switch works better than a tandem $(DL(N^2), WC, DL(N^2), DL(N^2))$. Using Lemma 8, it thus works better than $(WC, DL(3N^2))$. Finally, using Lemma 11, if the average delay for a work-conserving switch is defined, then the average delay for the load-balanced switch is within $3N^2$. ■