Terabit Switching and Routing Algorithms

Aleksandra Smiljanić

Informatika, Jevrejska 32,
Belgrade University, Kralja Aleksandra 73,
11000 Belgrade, Yugoslavia

ABSTRACT

The Internet is growing, and terabit routers are needed. We will discuss two scalable architectures, and their performances: the packet switch with input buffers based on a cross-bar fabric, and the packet switch based on a Clos fabric and load balancing. Both architectures provide non-blocking, and, therefore agile admission control for unicast and multicast traffic. They will be also shown to provide rate and delay guarantees to the sensitive applications.

Keywords: Packet switches, Internet routers, control algorithms, scheduling, performance

1. INTRODUCTION

In conventional telephone networks, the bandwidth demand was predictable and it was slowly changing. Users demanded low bit-rate services, and their traffic was groomed at the edge of the network into the higher bit-rate circuits and transported through the network. However, on the Internet, the traffic pattern is not predictable. Also, the applications require a wide range of bit-rates and have a large variety of holding times. Traffic grooming becomes difficult to implement in this environment. Instead, the Internet routers were build to transport packets based only on their destination ports, not requiring the global synchronization or circuit setups. The bandwidth allocation within a packet switch changes with the traffic pattern, and individual traffic streams do not have to be groomed but are switched on a packet-by-packet basis.

When a packet arrives to the router, its IP destination address is read and the router output port to which it will be switched is determined based on this IP address. The packet is stored in the appropriate buffer, and either transmitted through the router at a certain point of time, or dropped in the case of congestion.

First generations of Internet routers had output buffers. A packet coming to some input is broadcast to all outputs. Only the output buffer for which the packet is bound stores the incoming packet in question. The output buffers are able to store incoming packets from all inputs simultaneously, and so their throughputs are equal to the router throughput. Consequently, the router throughput is limited by the buffer maximum throughput. Also, the multiple output buffers are significantly overbuilt to pass the peak bit-rate N times larger than the line bit-rate that they pass on average. In a router with the shared buffer, all packets are stored in a single buffer. In this way, the number of high-throughput buffers is minimized. On the other side, the shared buffer should be able to store the larger number of packets and, consequently, the router throughput may be more severely limited. In both architectures, a separate scheduler will determine the transfer of packets at each output port. As a result, sophisticated scheduling of packets can be implemented to provide rate and delay guarantees to the demanding applications, and a fair service of the best-effort traffic. Both architectures are inherently non-blocking since they pass all the traffic as long as it does not overload the outputs.

In this paper, we will focus on the performance of two more scalable architectures for the Internet routers: packet switches with input buffers, and Clos packet switches based on load balancing.

It has been recognized (by DEC, Cisco, and many research groups) that the packet switches with input buffers provide higher switching capacities, because they require low throughput buffers and the cross-bars that scale better than the shared buffers.[1, 5, 7, 14, 16, 17, 26] Packets are stored into the virtual output queues based on their output ports. The property of a cross-bar is that it transmits at most one packet from some input router

Email: aleks@ieee.org

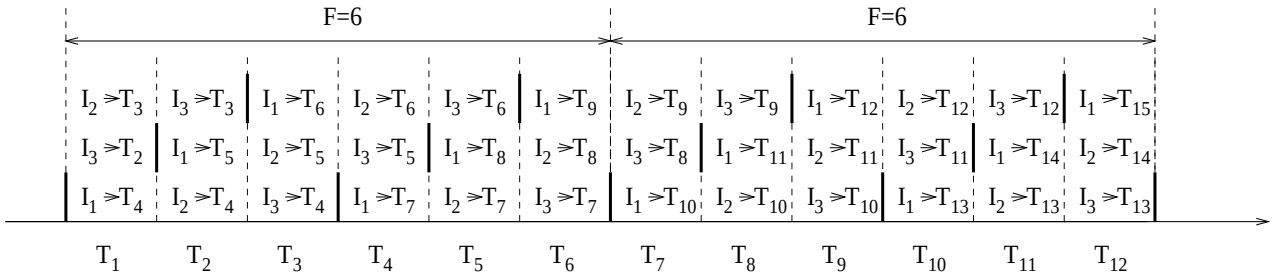


Figure 1. Pipelining of the sequential scheduling protocol in a 3 × 3 switch.

port and to some output router port at any point of time. Given the outstanding packets, and the cross-bar property, the central controller schedules packets through the router. The scheduling task is more complex than in the case of the routers with output or shared buffers because it must consider all incoming packets which do not only compete with the packets bound for the same output ports, but also with the packets sourced by the same input ports. We will discuss in the later section various scheduling algorithms that have been proposed to configure the cross-bars in packet switches with input buffers. It will be shown that the practical algorithms can provide rate and delay guarantees to the most sensitive applications in this architecture.

The size of a single-hop cross-bar fabric is still limited by the technology, and the fabrics available on the market do not exceed the terabit capacity. Terabit optical packet-switched cross-bars have been demonstrated in the research labs of various system vendors, but still require costly technology. On the other side, a multihop fabric such as Clos network provides the higher capacity by using the smaller switching elements (SE). When the traffic load is balanced over the switches in a middle stage, all the traffic would get through the fabric, as long as the switch outputs are not overloaded.[2,27] The scalability of this architecture is enhanced because it requires neither synchronization on a cell-by-cell basis across the fabric, nor the centralized scheduler. The delay that packets experience through the Clos switch depends on the number of flows that are separately balanced. We examined the maximum fabric utilization under which a tolerable delay is provided for various load balancing algorithms,[22–25] and will present the general formula for this utilization in terms of the number of flows that are balanced. It will be shown that the algorithms which balance flows with sufficiently coarse granularity provide both high fabric utilization and delay guarantees to the most sensitive applications. Since no admission control is performed within the switch, the fast traffic-pattern changes can be accommodated in the proposed scalable architecture.

2. PACKET SWITCHES WITH INPUT BUFFERS

Packet switches with input buffers drew a lot of attention in the last decade due to their scalability, and were discussed extensively in the literature.[1,5,7,14,16,17,21,26] If packets are stored at the switch inputs, the buffer throughput is equal to the port bit-rate and is not a bottleneck as in the packet switches with output or shared buffers. Packets, then, pass a cross-bar fabric that is non-blocking if the right scheduling algorithm is applied.

Optical packet-switches cross-bars have been demonstrated in several research labs.[20] The optical cross-bars may reduce the power required for packet switching, and the compactness of the equipment; also, they may enhance the reliability of the equipment. The NEC and Alcatel research groups have demonstrated the optical cross-bars based on semiconductor optical amplifiers (SOAs) and array waveguide gratings (AWG). The Lucent research group built an optical cross-bar where fast tunable lasers attached to AWG route packets by tuning to the appropriate wavelengths. Company Chiaro developed the waveguide arrays that steer the light depending on the applied voltages, and used them to construct high-capacity cross-bars. The capacity of electronic cross-bars increase with the development of the chip-to-chip transmission technologies. The electronic cross-bars with capacities exceeding 200Gbps are offered by various companies: AMCC, Mindspeed, Vitesse etc.

The scheduler design is another challenge of high-capacity packet switches with input buffers. It was realized that FIFO queues in the input buffers imply the head-of-line (HOL) blocking that may cause the switch throughput to fall to the port capacity for some particular packet arrival pattern, and that the scheduler should consider

all requested outputs by each input when scheduling packets in order to avoid the HOL blocking.[1] Later on, it was shown that the maximal matching scheduling algorithms ensure the 100% utilization of a cross-bar with the speedup of two.[7,16,21] It was also shown that the maximal matching algorithms provide rate and delay guarantees through cross-bars. Two practical maximal matching algorithms based on a pipeline technique have been proposed in Ref. 16,26. They provide satisfactory performance to the delay sensitive applications. But, the fairness that they provide to the best-effort traffic should be further improved.

2.1. Implementation of Sequential Greedy Scheduling

The sequential greedy scheduling (SGS) protocol determines when the packets are switched from the input buffers through a cross-bar fabric.[16,17] SGS is a maximal matching algorithm that is simple to implement. Inputs choose outputs one after another in a pipeline fashion. Packets are stored in different queues according to their destinations, so that the information about any queue status (empty or non-empty) and its heading packet is readily obtained. Such an input buffer organization is often referred to as a buffer with virtual output queueing (VOQ).[1,14] Packets are split into cells, and at most one cell is transmitted from any input, and to any output in each time slot. The cross-bar configuration in one time slot is calculated in multiple earlier time slots, and multiple schedules are calculated in each time slot.

Figure 1 shows the time diagram for pipelining where in each time slot, only one input selects an output for a particular time slot in the future. If $\mathrm{I}_i \rightarrow \mathrm{T}_k$ is assigned to some time slot T_j, it means that input I_i reserves an output for time slot T_k, and this reservation is made during time slot T_j. Bold vertical lines enclose a calculation of the cross-bar configuration, which lasts N time slots in the given example. Here N denotes the number of input and output ports. In the more general case, in any time slot multiple inputs might select outputs for some future time slot, or it might take multiple time slots for an input to select an output for a future time slot.

Time is further divided into frames comprising a fixed number of time slots, F (as shown in Figure 1). Input-output pairs are guaranteed the negotiated numbers of time slots (credits) per frame. Before some input chooses an output for the first time slot within a frame, its counters are set to the negotiated numbers of credits, $c_{ij} = a_{ij},\ 1 \leq j \leq N$. Only the queues with positive counters would compete for service, and whenever a queue is served its counter is decremented by 1. After inputs schedule packets from the queues with positive counters, they might schedule packets from the remaining queues in the same pipelined fashion.[16,17] In this way, the best-effort traffic can be accommodated if there is some bandwidth left after the higher priority traffic is served.

The pipelined sequential greedy scheduling algorithm is easy to implement, and it scales well with the increasing number of ports and the decreasing packet transmission time. An advantage of the proposed protocol is that it requires communication only among adjacent input modules, and, consequently, the simple scheduler implementation, as shown in Figure 2. Also, by using pipelining, the requirements on the speed of electronics are relaxed.

2.2. Performance of the Cross-bar Run by a Maximal Matching Algorithm

Definition: A maximal matching algorithm finds a maximal subgraph of a bipartite graph in which a node degree is at most one. Here the maximal subgraph (called maximal matching) is the one to which no edge can be added while keeping a node degree to be at most one.

In our case, if there is a packet from some input to some output, a maximal matching algorithm will schedule either the input or the output in question. If we assume the opposite (that there is a packet from some input to some output and neither the input nor the output are scheduled), a bipartite subgraph that corresponds to this schedule would not be maximal because an edge can be added between the nodes corresponding to these input and output, and their degrees become one (and are at most one). Note that SGS is the maximal matching algorithm.

The following theorem has been previously proven[16,17,21]:

THEOREM 2.1. *A maximal matching algorithm ensures a_{ij} time slots per frame for unicast traffic to input-output pair $(i,j),\ 1 \leq i,j \leq N$, if the following condition holds:*

$$\sum_m a_{im} \leq \frac{F+1}{2},\ \sum_m a_{mj} \leq \frac{F+1}{2}. \tag{1}$$

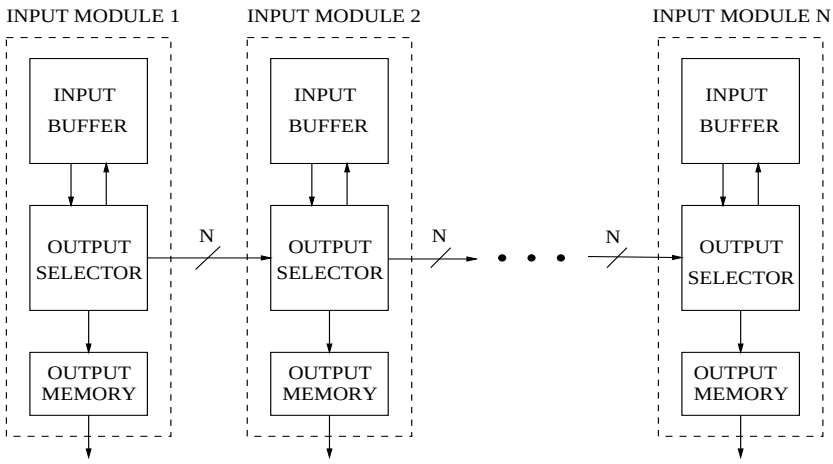


Figure 2. Central controller implementing sequential greedy scheduling protocol.

According to Theorem 2.1, the traffic passes the fabric in the next policing interval as long as the utilization of a port capacity is 50%. The policing interval equals F cell time slots. Note that policing is necessary for rate and delay guarantees to be provided. For example, input 1 negotiated the bit-rate of 10Mbps to output 3, the policing interval is $F = 10^4$ cell time slots, and the port bit rate is 10Gbps. Then, input 1 will send at most one high-priority cell per frame to output 3.

Scheduling of multicast packets is an issue that can be addressed in different ways. If the controller transmits a packet when all multicast outputs are idle, the packet may experience an unacceptable delay. If a multicast packet is replicated at the input port and sent separately to different multicast outputs, the separate transmissions of the multicast packets may clog the input port. One way to transfer multicast sessions through a cross-bar without blocking is to forward them multiple times through the cross-bar fabric.[18, 19] Namely, an input port would transmit a multicast packet only to one multicast output, and this multicast output would forward the packet to P other multicast outputs. Each multicast output forwards the packet to P outputs that have not received the packet yet until all outputs receive the packet. The following theorem holds:

THEOREM 2.2. *A maximal matching algorithm ensures a_{ij} time slots per frame for multicast traffic to input-output pair (i, j), $1 \le i, j \le N$, if the following condition holds:*

$$\sum_m a_{im} \le \frac{F+1}{P+2}, \quad \sum_m a_{mj} \le \frac{F+1}{P+2}, \tag{2}$$

where P is the forwarding fan-out.

According to Theorem 2.2, the traffic passes the fabric within $\log_P ((P-1) \cdot (N+1) + 1)$ policing intervals as long as the utilization of a port capacity is $1/(P+2)$ or less.

The switch capacity that can be utilized equals $C = N \cdot B/(P+2)$. There is an obvious trade-off between granularity of bandwidth reservations $G = B/F$ and packet delay $D = S \cdot F \cdot T$, where B is the port bit rate, and T is the packet transmission time. Assuming that $P = 2$, $B = 10$Gb/s, $T = 50$ns and $F = 10^4$, the granularity of bandwidth reservations is $G = 1$Mbps, and the packet delay is $D = 5$ms. As the frame length is increased, the granularity is refined, but the packet delay is prolonged. However, the packet delay does not increase significantly with the switch size.

3. CLOS PACKET SWITCHES BASED ON LOAD BALANCING

Clos circuit switch has been proposed by Clos in 1953 at Bell Labs.[6] Figure 3 shows the connections between switching elements (SE) in a symmetric Clos three-stage switch. This interconnection rule is: the xth SE in

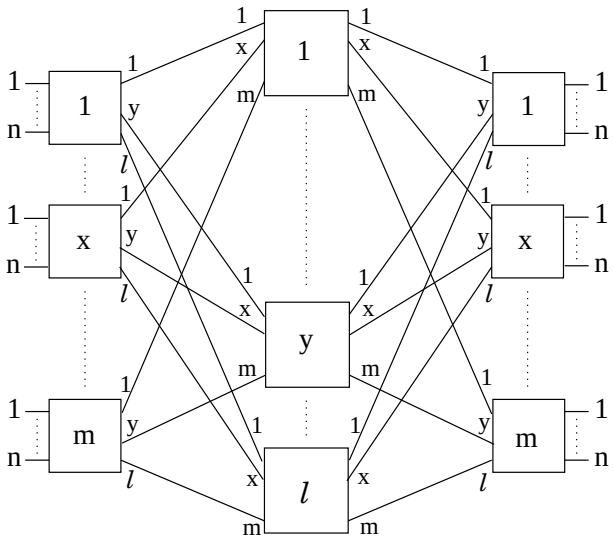


Figure 3. Clos switching fabric

some switching stage is connected to the xth input of each SE in the next stage.[6, 8, 9, 13] It is assumed that all connections have the same bandwidths. In Figure 3 and here on, n denote the number of external ports per SE, m the number of input and output SEs, and l the number of center SEs. The algorithms for the circuit setup have been derived for various Clos fabric complexities. However, these algorithms cannot be readily implemented in Clos packet switches, where SEs should be reconfigured in each cell time slot. Here, packets are split into cells of a fixed duration, which is typically 50ns (64 bytes at 10Gb/s). First, all SEs should be synchronized on a cell-by-cell basis. Then, implementation of the algorithm that rearranges connections on a cell-by-cell basis in SEs of a rearrangeable non-blocking Clos switch would be prohibitively complex.[8] So, the Clos fabric with the larger switching fabric is needed for a non-blocking packet switch. A scheduling algorithm that would provide non-blocking in this Clos packet switch would require the higher processing complexity than its counterpart designed for a cross-bar switch.[16, 17] Few heuristics have been proposed to configure SEs in Clos packet switches without assessment of their blocking nature.[12, 15]

A Clos packet switch in which the traffic load is balanced across SEs provides non-blocking, i.e. with sufficiently large buffers it passes all the traffic if the outputs are not overloaded.[2, 27] There is a buffering in each stage of the architecture, and the SEs in the heading stages are balancing packets over the SEs in the next stages. Turner showed that the architecture is non-blocking if the traffic of each multicast session is balanced over the SEs in a Benes packet switch.[27] Here, the multicast session carries the information between end users in the network. The implementation of load balancing in Clos packet switches is simple: there is no need for the high-capacity shared buffers or cross-bars, there is no need for the cell-by-cell synchronization across the fabric, and there is no need for the centralized scheduler. The non-blocking property of these switches is an attractive feature: it simplifies the network design because the traffic passes the fabric as long as the output ports are not overloaded, and it enables distributed admission control which can follow the fast traffic-pattern changes typical for the Internet. Namely, when reserving the bandwidth, an input port has to check if the output port has enough capacity, or a user has to check if its destination user has enough capacity to receive data.

Load balancing has been proposed in other architectures such as parallel plane switches (PPS) or Birkhoff-von Neumann switches,[3, 10, 11] that were recently shown to be equivalent.[11] In these architectures, packets from each line card are balanced over the output queued switches in the middle stage.[3, 10] This architecture is a special case of Clos packet switches with only one external port per SE, $n = 1$.

Delay sensitive traffic is a significant part of the Internet traffic. It was shown that the tolerable delay is guaranteed for the fabric utilization that may unacceptably decrease with the increasing number of flows that are separately balanced.[22–25] However, certain load balancing algorithms were proven to provide the tolerable delay even to the most sensitive applications in high-capacity switches.

3.1. Implementation of Load Balancing Algorithms in the Clos Packet-Switches

Obviously, when cells reach the center SEs (SEs in the second stage), they are further routed according to their output addresses. So, load balancing can be only performed at the input SEs (SEs in the first stage). We will discuss four different load balancing algorithms. They differ according to the definition of the flows that are separately balanced.[22] For example, in one definition a flow comprises cells sourced by some input and bound to the same output; in another definition a flow comprises cells sourced by some input and bound to the same output SE (SE in the third stage); or a flow comprises cells sourced by the same input SE and bound to the same output; or a flow comprises cells sourced by the same input SE and bound to the same output SE etc.

In the first load balancing algorithm, input $i,\ 0 \leq i < N$, has N different counters associated with different outputs, $c_{ij},\ 0 \leq j < N$. Here $N = nm$ is the number of switch input and output ports. A cell arriving to input i and bound for the jth output will be marked to be transmitted through the c_{ij}th output of its SE, i.e. to be transmitted through the c_{ij}th center SE. Then, the counter in question is incremented modulo l, namely $c_{ij} \leftarrow (c_{ij} + 1) \bmod l$. In the second load balancing algorithm, input $i,\ 0 \leq i < N$, stores the smaller number of counters, m, associated with different switch output SEs, $c_{ij},\ 0 \leq j < m$. A cell arriving to input i and bound for output SE j will be marked to be transmitted through the c_{ij}th output of its SE. Then, the counter in question is incremented modulo l. In the third load balancing algorithm, input SE $i,\ 0 \leq i < m$, stores N different counters associated with different outputs, $c_{ij},\ 0 \leq j < N$. A cell arriving to input SE i and bound for the jth output will be marked to be transmitted through the c_{ij}th output of its SE. Then, the counter in question is incremented modulo l. In the fourth load balancing algorithm, input SE $i,\ 0 \leq i < m$, stores the smaller number of counters, m, which are associated with different switch output SEs, $c_{ij},\ 0 \leq j < m$. A cell arriving to input SE i and bound for output SE j will be marked to be transmitted through the c_{ij}th output of its SE. Then, the counter in question is incremented modulo l.

3.2. Performance Analysis of Load Balancing Algorithms

The following theorem has been previously proven.[22]

THEOREM 3.1. *Non-blocking is provided in a Clos packet switch based on the load balancing without the fabric speedup.*

Traffic of each individual flow is balanced independently across the SEs. If there are many flows that simultaneously transmit cells across some SE, their data will experience long delay. Many applications, e.g. voice and video, require rate and delay guarantees. We will assess the lowest maximum utilizations for which a tolerable delay is guaranteed to the sensitive applications. We will assume that the delay sensitive traffic is policed at either the edge of the network, or at the switch ports. Also, SEs are non-blocking and transfer the policed traffic within one frame period. These features hold for the SE with the shared buffers, and the SEs using cross-bars with the speedup of two and run by the maximal matching algorithms.[16, 17, 20, 26]

Eeach input-output pair is guaranteed a specified number of time slots per frame, for example a_{ij} time slots are guaranteed to input-output pair $(i, j),\ 0 \leq i, j < N$, while each port can be assigned at most F_u time slots per frame, i.e.

$$\sum_k a_{ik} \leq F_u,\ \sum_k a_{ki} \leq F_u. \tag{3}$$

We assume a coarse synchronization in a switch, i.e. that at some point of time the input ports schedule cells belonging to the same frame, and the SEs in each stage schedule packets that have arrived in the previous frame. Since all cells of a frame are guaranteed to pass the switch stage within the next frame, the resequenced cells may be at most F cells apart. Consequently the resequencing buffer size is F cells, and the total cell delay through a three-stage Clos packet switch equals:

$$D = 4FT_c. \tag{4}$$

Formulas for the switch utilization and the fabric speedup for which the specified tolerable delay is guaranteed are enclosed in the following sections. Their derivations are beyond the scope of this paper, and have been, in part, previously published.[22–24]

3.2.1. Switch Utilization

In the previous papers, we calculated the number of cells per frame sent from a given input SE through a given center SE ($F'_c \leq F$) in terms of F_u, and then the maximum utilization of the connections from input to center SEs (F_u/F). Because of the symmetry, utilization is the same for the connections from center to output SEs. Note that all theorems hold in large switches where $l > 10$.

THEOREM 3.2. *Maximum utilization of any internal link in the fabric under which all cells pass it within designated frames is:*

$$\max\left(0, S - \frac{lN_f}{nF}\right) \leq U_a \leq \min\left(1, S - \frac{l(N_f - n)}{nF}\right). \tag{5}$$

where N_f is the number of flows that are passing through some internal link of the fabric.

Note that Theorem 3.2 holds for a Benes network with an arbitrary number of stages.[2,27]

The maximum utilization in Theorem 3.2 is calculated when different flows bound for the same SE are not properly synchronized, so they might send cells within a given frame starting from the same center SE. Alternatively, equal numbers of flows are balanced starting from different center SEs in each frame. For example, flow g of SE_{1i} resets its counter at the beginning of a frame to $c_{ig} = (i + g) \bmod l$. Or, flow g bound to SE_{3k} resets its counter at the beginning of a frame to $c_{kg} = (k + g) \bmod l$.

THEOREM 3.3. *In the algorithms where balancing of different flows is synchronized, maximum utilization of any internal link in the fabric under which all cells pass it within designated frames is:*

$$U_r = \begin{cases} S - \frac{lN_f}{2nF} & F \geq \frac{lN_f}{nS} \\ \frac{nS^2F}{2lN_f} & F < \frac{lN_f}{nS}. \end{cases} \tag{6}$$

Note that Theorem 3.3 provides the maximum utilization when both balancing of flows sourced by an input SE, and balancing of flows bound for an output SE are synchronized. This assumption will hold in all considered algorithms.

3.2.2. Switch Speedup

Signal transmission over the fibers connecting distant routers may require the most complex and costly hardware. Therefore, it is important to provide the highest utilization of the fiber transmission capacity. For this reason, switching fabrics with the speedup have been previously proposed. Namely, the internal links of the fabric have the higher capacity than the external links:

$$S = \frac{nmR}{mlR_c} \geq 1, \tag{7}$$

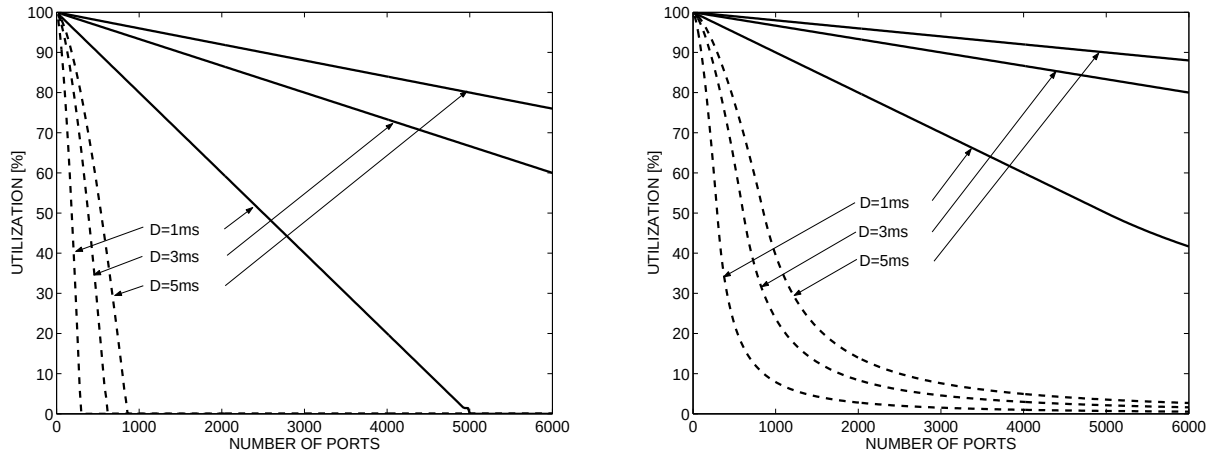
where R is a bit-rate at which data is transmitted through the fibers, and R_c is a bit-rate at which data is transmitted through the fabric connections.

THEOREM 3.4. *The speedup S required to pass all incoming packets with a tolerable delay when the counters are asynchronous is:*

$$1 + \frac{l(N_f - n)}{nF} \leq S_a < 1 + \frac{lN_f}{nF}, \tag{8}$$

and the speedup when counters are synchronized is:

$$S_r \geq \begin{cases} 1 + \frac{lN_f}{2nF} & F \geq \frac{lN_f}{2n} \\ \sqrt{\frac{2lN_f}{nF}} & F < \frac{lN_f}{2n}. \end{cases} \tag{9}$$



(a) Asynchronous counters (b) Synchronized counters

Figure 4. Switch utilization: solid curves represent the algorithm in which inputs balance flows bound for output SEs, and the algorithm in which input SEs balance flows bound for outputs; dashed curves correspond to the algorithm in which inputs balance flows bound for outputs.

3.3. Performance of Load Balancing Algorithms

It can be observed from formulas (5,6,8,9) that the performance of a load balancing algorithm depends on the number of flows that are separately balanced.

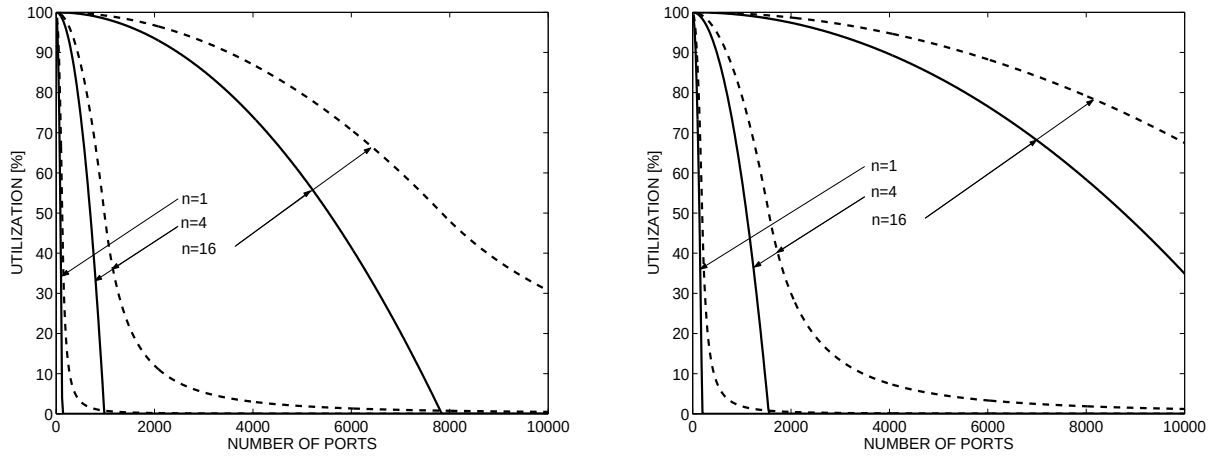
First we will assume that the Clos packet switch comprises identical $n \times n$ SEs, i.e. that $n = m = l = \sqrt{N}$. In the first algorithm, $N_f = nN$, because any input SE sources nN flows, and each of N inputs balances n flows bound for any output SE. In the second algorithm $N_f = N$, because any input SE sources $n^2 = N$ flows, and each of N inputs balances one flow for any output SE. In the third algorithm, $N_f = N$ because any input SE sources N flows, and each of n input SEs balances n flows for any output SE. In the fourth algorithm, $N_f = n$ because any input SE sources n flows, and each of n input SEs balances one flow for any output SE. In order to increase the efficiency of the load balancing algorithms, the frame length should be increased. On the other side, the cell delay is proportional to the frame length. So, the maximum frame length will be determined by the delay that could be tolerated by the applications such are interactive voice and video. Assume that the maximum delay that can be tolerated by interactive applications is D, and the cell time slot duration is T_c, then

$$F \leq \frac{D}{4T_c}. \tag{10}$$

Under the assumption of no speedup, i.e. $S = 1$, we obtain the maximum utilizations by substituting $F = D/(4T_c)$ and N_f for the corresponding balancing algorithms in formulas (5) and (6). Speedups are obtained when N_f is substituted in formulas (8) and (9).

One way packet delay that can be tolerated by interactive applications is around 150ms, but only 50-60ms of this allowed delay can be budgeted for queueing. In order to provide flexible multicasting, the ports should forward packets multiple times through a packet switch, and the packet delay is prolonged accordingly.[2, 18, 19, 27] Consequently, the tolerable delay through one packet switch should be well below 10ms.

Figure 4 shows the fabric utilization decreases as the switch size is increasing for various tolerable delays (T_c =50ns). The solid curves represent the second and the third algorithm ($N_f = N$), while the dashed curves correspond to the first algorithm ($N_f = nN$). One can see that the efficiency of the first balancing algorithm might decrease unacceptably as the switch size is increasing. For example, the utilization of a fabric with 1000 ports drops below 10% for a tolerable delay of 3ms. On the other side, for the same tolerable delay, utilization of a fabric with 4000 ports is 80% if the second or the third load balancing algorithm is applied. We note that the efficiency of the first load balancing algorithm is improved when the counters are synchronized, but, it is



(a) T_c =50ns (b) T_c =20ns

Figure 5. Fabric utilization for the algorithm in which input SEs balance flows bound for output SEs: solid curves represent the performance with the asynchronous counters; dashed curves correspond to the performance with the synchronized counters.

still low in the large switches where cells bound for the particular output are spread equally across the center SEs. For example, the utilization of a fabric with 1000 ports drops below 30% for a tolerable delay of 3ms and T_c = 50ns, and again drops below 10% in a switch with 4000 ports. Efficiency of the second and the third load balancing algorithm is improved too. For the same tolerable delay, utilization of a fabric with 4000 ports is 90%.

The fourth algorithm requires the large number of SEs with the shared buffers of a limited size, $l = m > n$. From equations (5,6,8,9) follows that the performance degrades as l/n increases, but the smaller number of flows that are balanced in the fourth algorithm may compensate for this degradation. Figure 5 shows the fabric utilization decrease with the number of ports for various numbers of ports per SE. The solid curves represent the algorithms with no synchronization of the counters, while the dashed curves correspond to the algorithms in which the counters are synchronized. Utilization is above 50% in a switch with 10^4 ports when the number of ports per line card is $n = 16$ and T_c = 20ns.

The first load balancing algorithm requires the speedups larger than 2 and 10, in order to provide the delay less than 3ms through a switch with 1000 and 4000 ports, respectively. The required speedup is somewhat reduced when the counters are synchronized. Speedup close to 1 is needed when the counters are synchronized and any other load balancing algorithm is applied. When the fourth algorithm is applied the number of ports per line card should be as large as 16, for the efficient performance.

Initially, it was proposed that the end-to-end sessions are separately balanced in a switch.[27] In that case $N_f \geq nN$, and consequently the performance is poorer than of the first algorithm which does not perform well itself. On the other side, the algorithms perform well when the sessions are groomed into the coarser flows that are then balanced separately. The tolerable delay would be provided in the large switches that comprise either the cross-bars or the shared buffers. The switch performance is significantly improved when the counters are synchronized. At the expense of a minor increase of the algorithm complexity, the fabric utilization is significantly increased. As a result, the switch consuming the smaller space and power would provide the given high capacity.

4. SUMMARY

Packet switches based on the cross-bar fabrics and the corresponding schedulers can support high switching capacities. Maximal matching algorithms schedule packets through a cross-bar fabric with the speedup of two without blocking them, i.e. they pass all the traffic that does not overload the outputs. Even higher capacity are envisioned in non-blocking Clos packet switches based on load balancing. Clos packet switches do not require high-capacity cross-bars, synchronization on a cell-by-cell basis, or a centralized scheduler. The fabric is well utilized and the tolerable delay is guaranteed when the number of balanced flows is reduced. Both architectures

provide rate and delay guarantees to the most sensitive applications such are interactive voice and video. Their non-blocking nature allows a distributed admission control which can follow the fast changes of traffic patterns on the Internet.

Acknowledgements

I thank to Miloš Petrović for his comments.

REFERENCES

1. 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, November 1993, pp. 319-352.
2. T. Chaney, J. A. Fingerhut, M. Flucke, J. S. Turner, "Design of a gigabit ATM switch," *Proceedings of INFOCOM 1997,* vol. 1, pp. 2-11.
3. C. S. Chang, D. S. Lee and C. M. Lien, "Load balanced Birkhoff-von Neumann switches, Part II: multi-stage buffering," *Computer Communications,* vol. 25, pp. 623-634, 2002.
4. C. S. Chang, D. S. Lee and C. Y. Yue, "Providing guaranteed rate service in the load blanaced Birkhoff-von Neumann switches," *Proceedings of INFOCOM 2003.*
5. H. J. Chao, "Saturn: A terabit packet switch using dual round-robin," *Proceedings of GLOBECOM 2000,* pp. 487-495.
6. C. Clos, "A study of non-blocking switching networks," *Bell Systems Technology Journal,* vol. 32, 1953, pp. 406-424.
7. G. Dai, and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE INFOCOM 2000,* pp. 556-564.
8. J. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Press 1990.
9. F. K. Hwang, *The mathematical theory of nonblocking switching networks,* World Scientific, 1998.
10. S. Iyer, and N. McKeown, "Analysis of the Parallel Packet Switch Architecture," *IEEE/ACM Transactions on Networking,* vol. 11, no. 2, April 2003, pp. 314-324.
11. I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, N. McKeown, "Scaling Internet routers using optics," *Proceedings of ACM SIGCOMM 2003.*
12. T. McDermott, and T. Brewer, "Large-scale IP router using a high-speed optical switch element," *OSA Journal on Optical Networking, www.osa-jon.org,* July 2003, pp. 229-241.
13. W. Kabacinski, C. T. Lea, G. Xue, "50th anniversary of Clos networks," *IEEE Communication Magazine*, vol. 41, no. 10, October 2003, pp. 26-64.
14. N. McKeown *et al.*, "The Tiny Tera: A packet switch core," *IEEE Micro*, vol. 17, no. 1, Jan.-Feb. 1997, pp. 26-33.
15. E. Oki, Z. Jing, R. Rojas-Cessa, H. J. Chao, "Concurrent round-robin-based dispatching schemes for Clos-network switches," *IEEE/ACM Transactions on Networking,* vol. 10, no. 6, December 2002, pp. 830-844.
16. A. Smiljanić, "Flexible bandwidth allocation in terabit packet switches," *Proceedings of IEEE Conference on High Performance Switching and Routing,* June 2000, pp. 233-241.
17. A. Smiljanić, "Flexible bandwidth allocation in high-capacity packet switches," *IEEE/ACM Transactions on Networking,* April 2002, pp. 287-293.
18. A. Smiljanić, "Scheduling of multicast traffic in high-capacity packet switches," *IEICE/IEEE Workshop on High-Performance Switching and Routing,* May 2002, pp. 29-33.
19. A. Smiljanić, "Scheduling of multicast traffic in high-capacity packet switches," *IEEE Communication Magazine,* November 2002, pp. 72-77.
20. A. Smiljanić, "Focus Issue: High-Capacity Packet-Switched Fabrics," *OSA Journal on Optical Networking, www.osa-jon.org,* July 2003, pp. 229-241.
21. A. Smiljanić, "Bandwidth Reservations by Maximal Matching Algorithms," *IEEE Communication Letters,* March 2004, pp. 177-179.
22. A. Smiljanić, "Performance of load balancing algorithms in Clos packet switches," *Proceedings of IEEE Workshop on High Performance Switching and Routing,* April 2004, pp. 304-308.

23. A. Smiljanić, "Performance of load balancing algorithms in Clos packet switches," *Invited Presentation at Stanford Workshop on Load-Balancing,* May 2004.
24. A. Smiljanić, "Load balancing algorithms in Clos packet switches," *Proceedings of IEEE International Conference on Communications,* June 2004.
25. A. Smiljanić, "High Performance Routers," *invited paper at joint Optoelectronic and Communication Conference and International Conference on Optical Internet,* Yokohama, Japan, July 2004.
26. Y. Tamir, and H. C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Transactions on Parallel and Distributed Systems,* vol. 4, no. 1, 1993, pp. 13-27.
27. J. S. Turner, "An optimal nonblocking multicast virtual circuit switch," *Proceeding of INFOCOM 1994,* vol. 1, pp. 298-305.