CS540 - Papers Presentation
Routers

Nhat Pham

RESL@KAIST

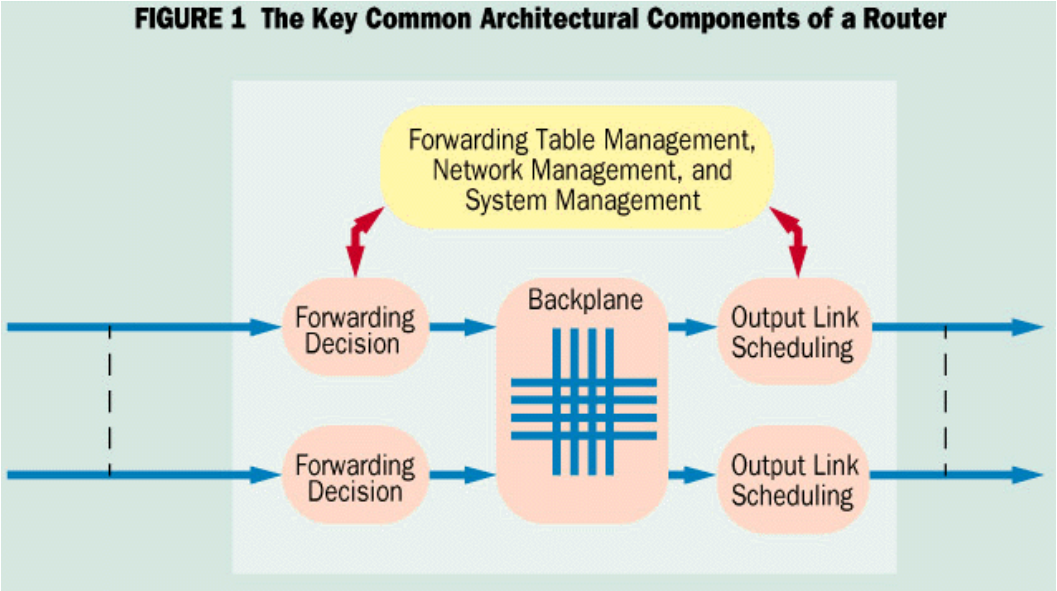
nhatphd@kaist.ac.kr

September 27, 2016

Overview

Fast Switched Backplane for a Gigabit Switched Router

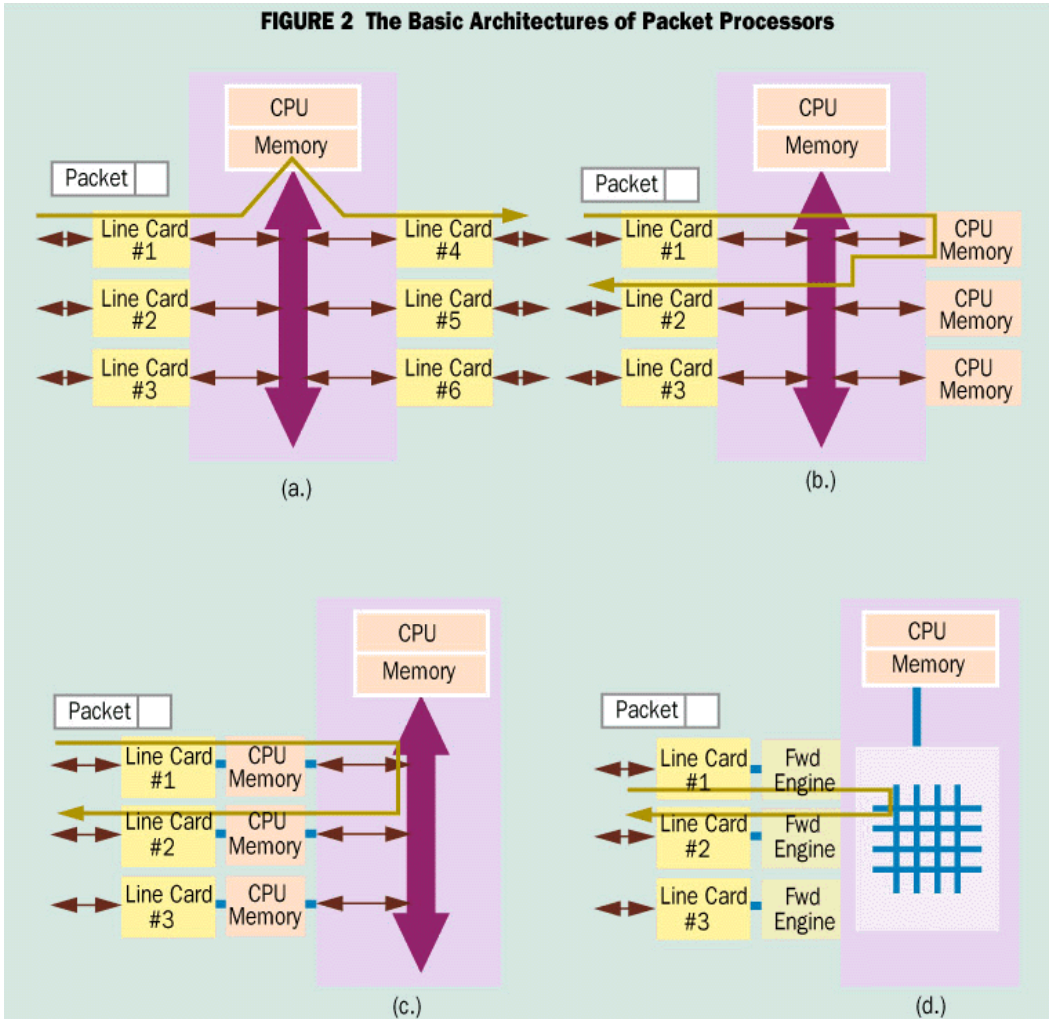
The Architecture of Internet Routers



Router functions:

- **Control functions**: performed relatively in frequently, implemented in software.
- **Datapath functions**: performed on every datagram arrives at the router. It's consists of:
  - **Forwarding decision**: datagram's destination is looked up in forwarding table. Next-hop MAC address is appended and new checksum is calculated.
  - **Backplane**: forwards datagram to its outgoing port (includes buffer queuing, packets are dropped/replaced when buffer gets full)
  - **Output-link Scheduler**: schedules datagram arrived at outgoing port to be transmitted on output link. Advanced router could have different flows or priority classes to guarantee QoS.

The Architecture of Internet Routers - Trend of Improvement



Overtime, or to enhance performance, more hardware is used in the main datapath and more parallelism is employed.

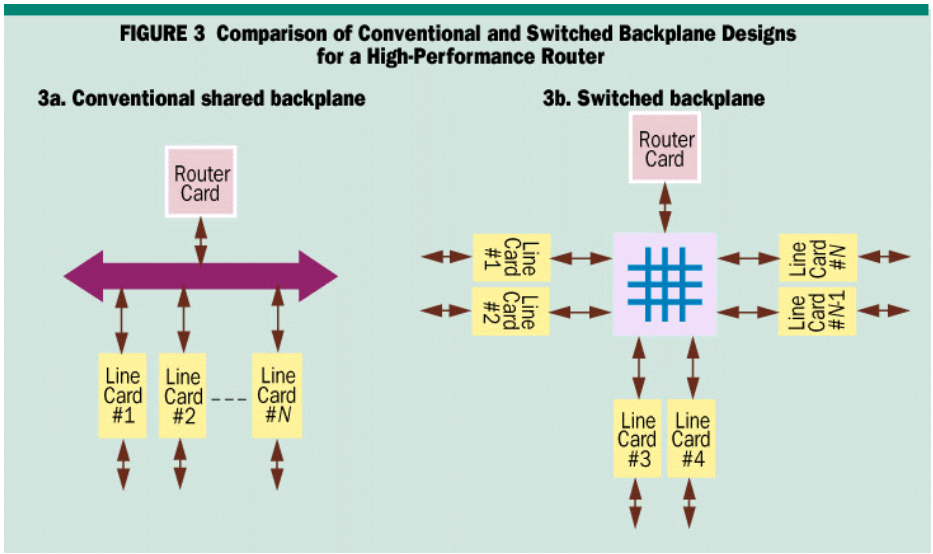
Why we need fast backplane

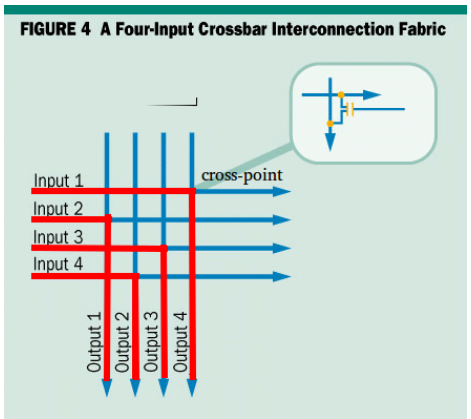
In this paper, the switched backplane design for Cisco 12000 GSR will be discussed. This design could support 16 ports simultaneously, each with a line rate of 2.4 Gbps.

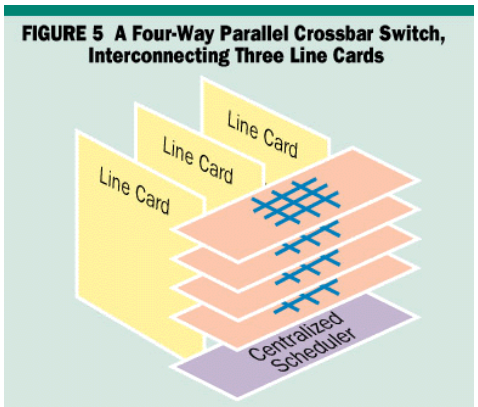
Question: Shared backplane (bus) gets congested $\rightarrow$ buffer overflows, packets get dropped $\rightarrow$ can we just make it faster?
Answer: Impractical!

- High speed bus is difficult to design.
- State-of-the-art shared buses could achieve 20Gbps (enough for 100Mbps links) while router with link-rate of 2.4 Gbps requires aggregate bandwidth of 40 Gbps (i.e. 16 line cards x 2.4 Gbps).

Switched Backplanes - Crossbar switch







Crossbar switch eliminates shared bus's limitations and enable higher throughput:

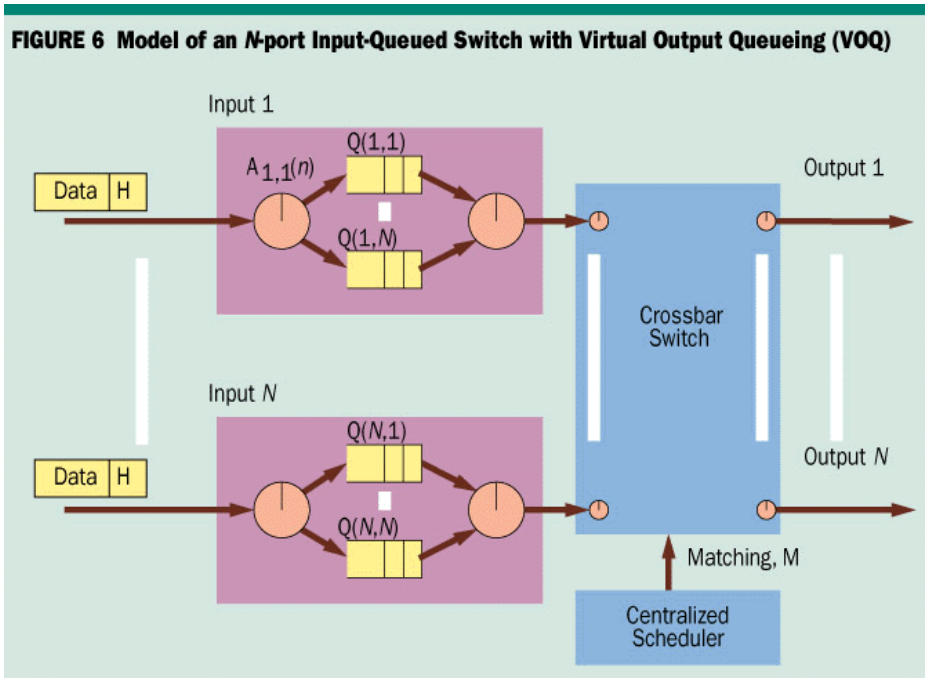
- Simple point-to-point links between line cards and central switch $\rightarrow$ higher speed (i.e. 4-10 Gbps serial links will be available soon, short point-to-point link also helps control clock skew and reduce electromagnetic interference)
- Crossbar switch could support multiple bus transactions simultaneously (i.e. internally non-blocking).
- Throughput could be further boosted by using many crossbar slices in parallel. (e.g. A four-way parallel crossbar switch could increase throughput by a factor of 4)

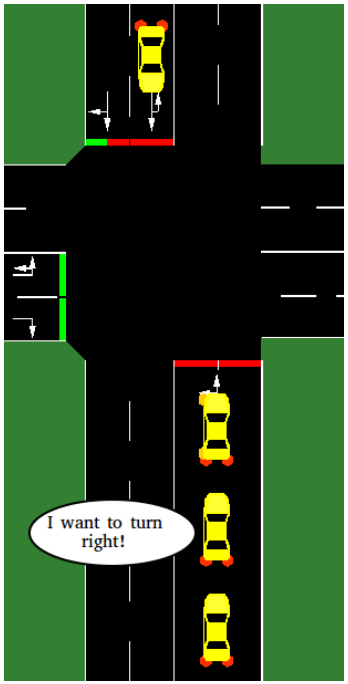
Fixed vs. Variable length packets

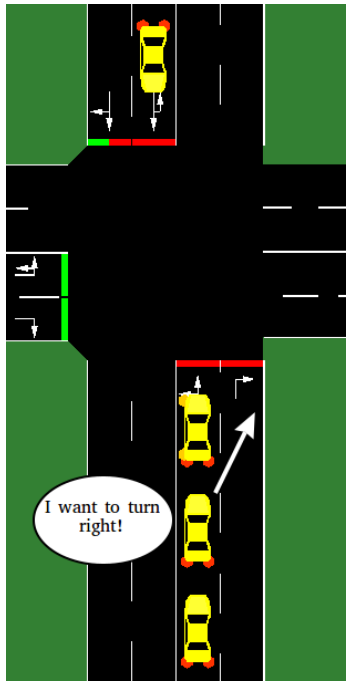
Packets could transverse across switched backplane in fixed or variable-length units (aka. cells).

- However, fixed-length cells are more efficient:
  - Easier for scheduler to examines waiting packets at the end of each time slot.
  - From hardware design perspective, simpler to design and handle fixed-length cells faster than variable ones.
- Variable-length cells make thing harder:
  - Scheduler must constantly track the busy and idle status of the outputs and schedule new input as soon as a output get idle.
  - Starvation could occur when multiple inputs require a output and each input may have packets for different outputs. (see **here**)
  - As a result, half the system's aggregate bandwidth could be used.

Input Queuing and Virtual Output Queuing







3 sources of blocking could limit performance:

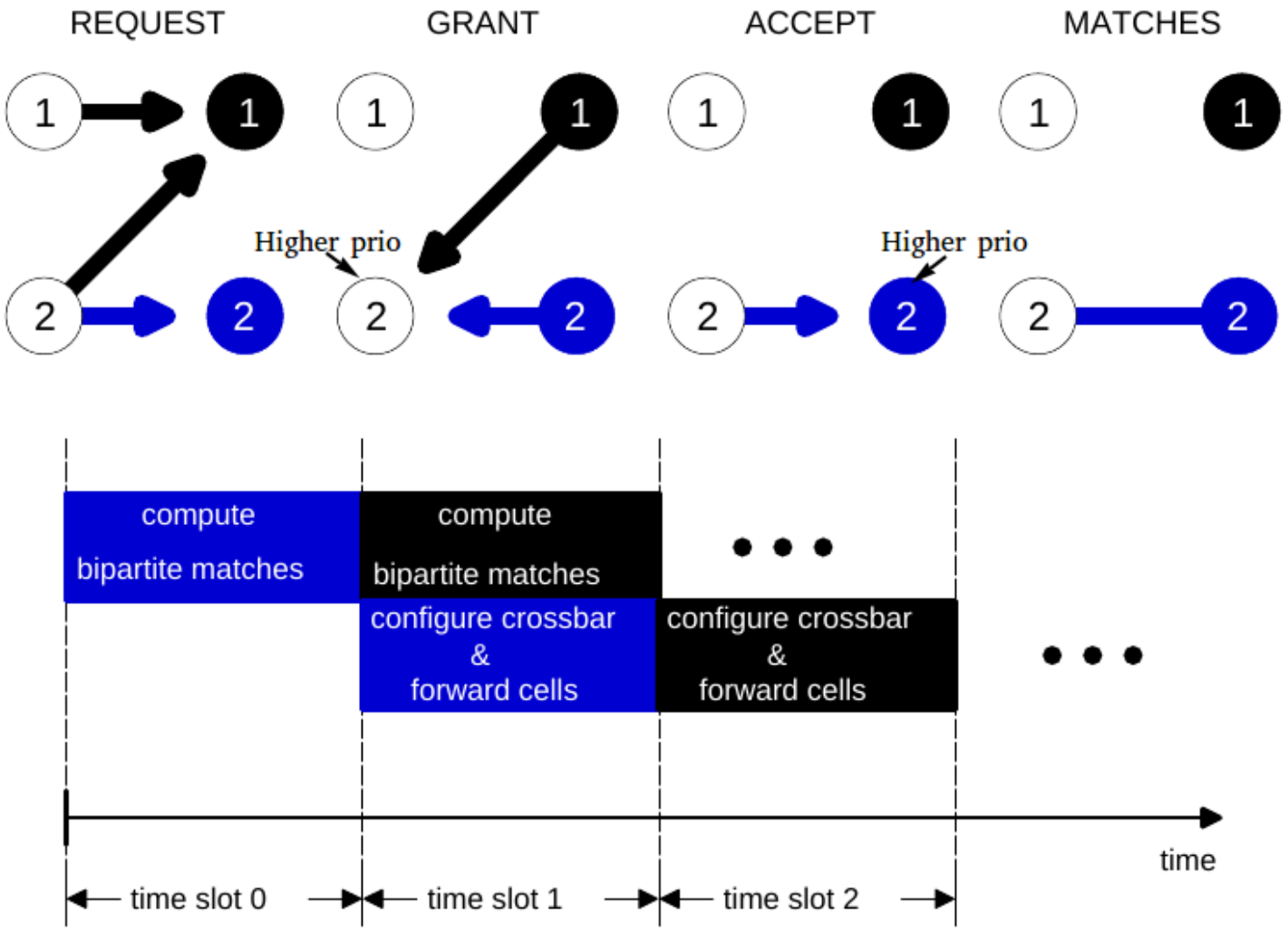
- Head of Line blocking. Could be solved by virtual output queuing.
- Input blocking and Output blocking. Don't reduce crossbar throughput but increase packet delay and make delay unpredictable. (will be discussed later)

Crossbar scheduling algorithm

Crossbar scheduling (i.e. inputs to outputs matching) algorithm should have 4 properties:

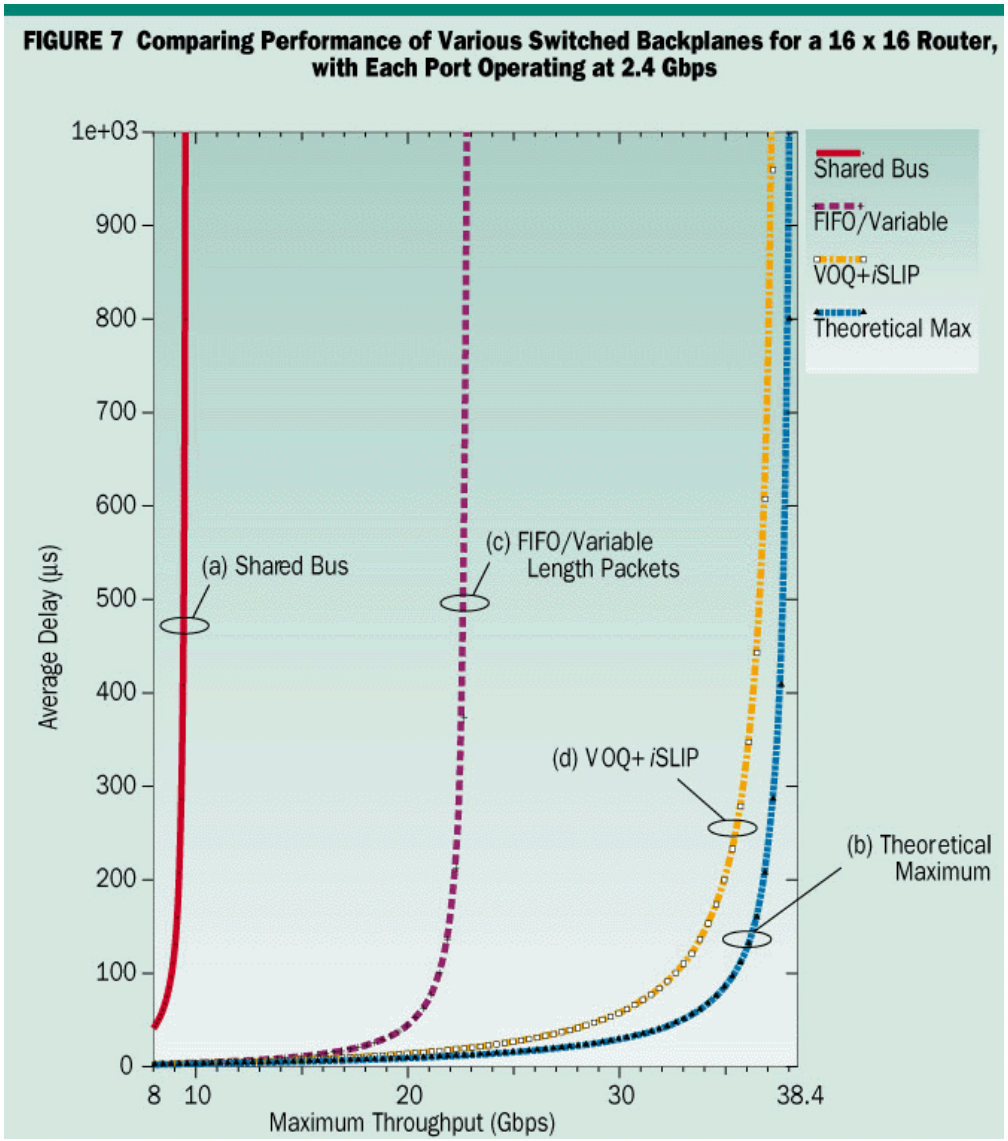
- **High Throughput**: Sustain an offered load up to 100% on each input and output.
- **Starvation-free**: no VOQ to be unserved indefinitely.
- **Fast**: must not become the performance bottleneck.
- **Simple to implement**: scheduler should be implementable in hardware.

Crossbar scheduling algorithm - iSLIP



iSLIP has 4 required properties: high throughput, starvation-free, fast and easy to implement in hardware.

Unicast traffic: Performance Comparison



- Shared bus: 600Mbps.
- FIFO/Variable-length packets: 22Gbps (i.e. 15Gbps is wasted by HOL blocking).
- VOQ+iSLIP: 38.4 Gbps.

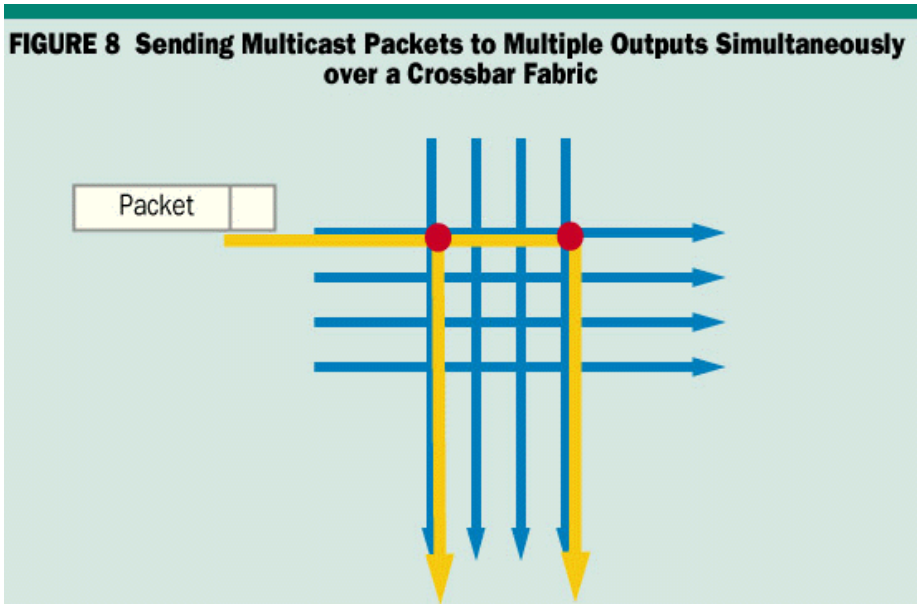
Controlling Packet Delay

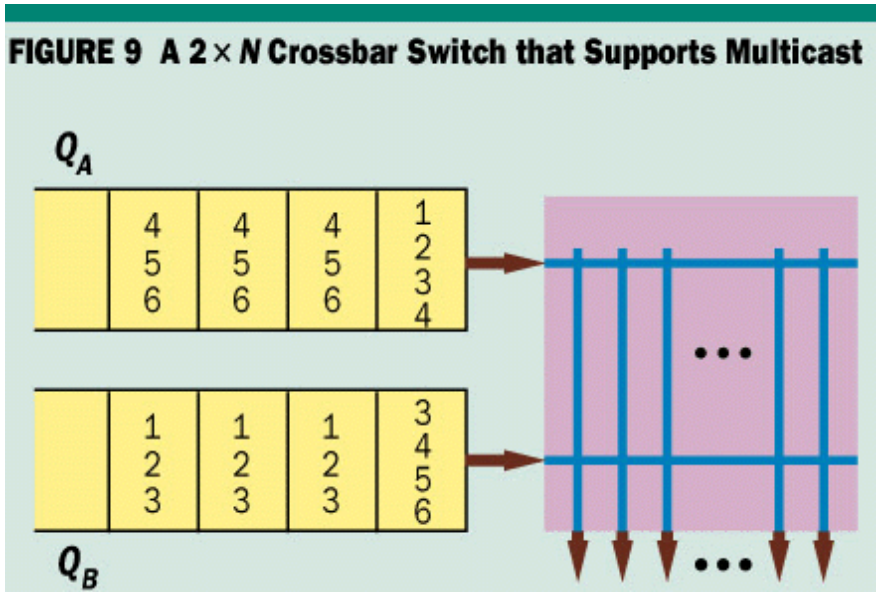
- **Input blocking**: one VOQ could be served by iSLIP scheduler at the same time, can't predict when a VOQ will be served.
- **Output blocking**: only one cell could be transmit on output link at the same time. $\rightarrow$ Make delay unpredictable. $\rightarrow$ Prioritization
- However, prioritization only works well when amount of high-priority traffic is small.

Speedup is also a way to reduce Input/Output blocking.

- Make crossbar switch much faster than line rate $\rightarrow$ reduces the delay of each cell through the switch $\rightarrow$ every cell is immediately transferred to the output port.
- Speedup of 2 is sufficient.

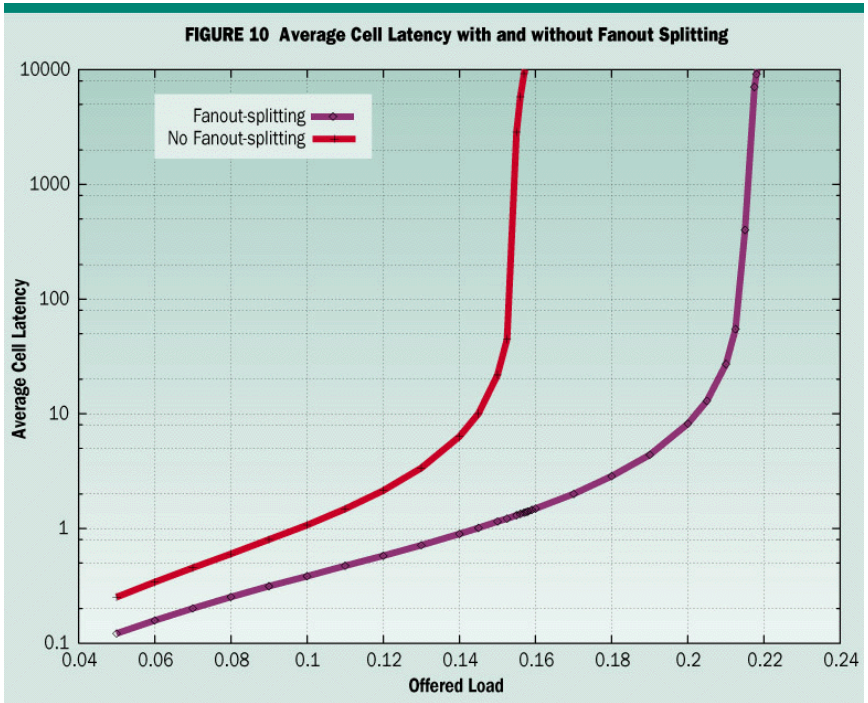
Scheduling Multicast Traffic

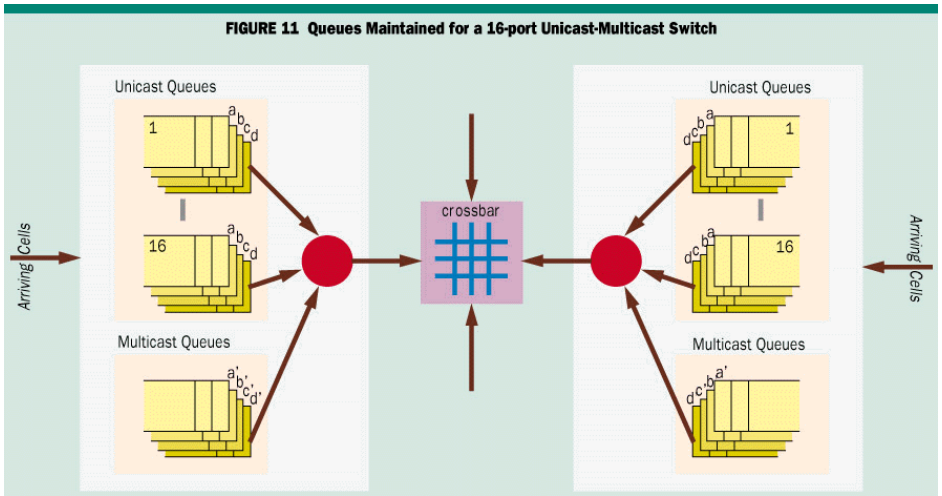




- Naive way to support multicast: replicate input cells into multiple clones. However, it consumes memory and reduces bandwidth at the same input.
- Better way is to close many cross-points of a input at the same time.
- While unicast cells are stored in VOQ, multicast cells are stored in a different FIFO of each input.

Scheduling Multicast Traffic - ESLIP





Fanout-splitting vs No Fanout-splitting

- **No Fanout-splitting**: all the copies of a cell must be sent in the same cell time.
- **Fanout-splitting**: output cells are delivered to output ports over any number of cell times. Unsuccessful ones continue to retry. $\rightarrow$ better performance!

An enhanced version of iSLIP: ESLIP was developed to combine scheduling for both unicast and multicast packets, prioritization while maintain required 4 properties (i.e. high throughput, starvation-free, fast, simple to implement)

Concluding Remarks

Summarize of Cisco 12000 GSR backplane design:

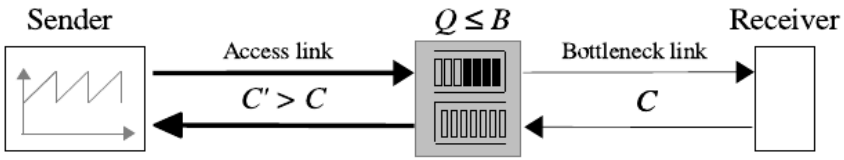
- **Fixed-Length Packets** are used instead of variable ones.
- **Virtual Output Queuing** is employed solve HOL problem.
- **Priority Levels and Speedup** are used to control delay.
- **Separate Multicast Queues** are used to support multicast packets.
- **Fanout-Splitting** is used to improve performance of multicast packets.
- **A Combined Unicast and Multicast Scheduler** is developed.

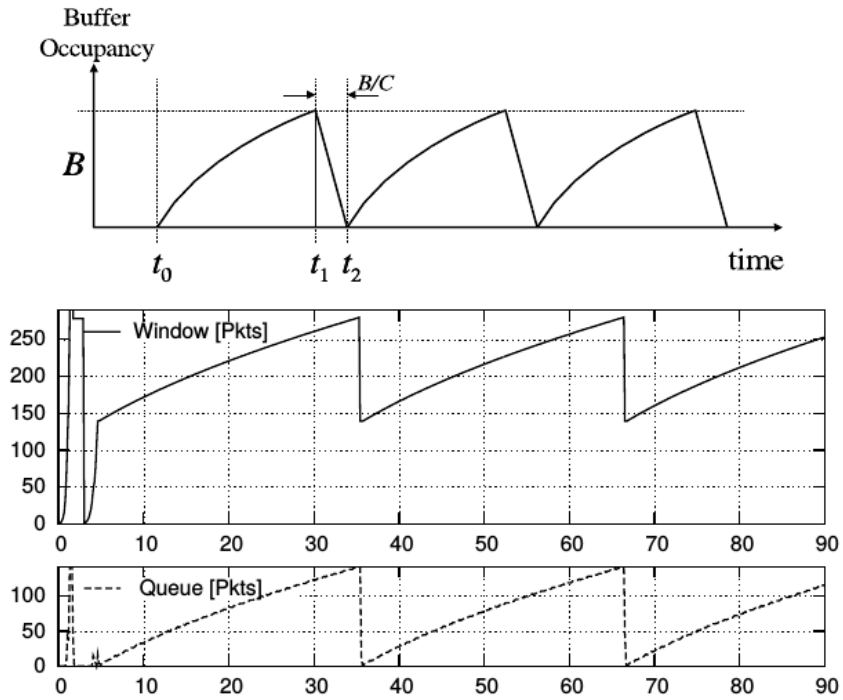
Sizing Router Buffers

Introduction and Motivation

- Well-known rule-of-thumb: $B = R\bar{T}T * C$; where, B: buffer size, $R\bar{T}T$: average round trip time, C: line-rate.
  $\rightarrow$ Buffer size will grows linearly with line rate, create huge buffer requirement (e.g. a 10Gbps linecard requires 250ms*10Gbps=2.5Gb buffer)
- The authors argues that this rule-of-thumb is incorrect and the buffer size should be: $B = R\bar{T}T * C/\sqrt{n}$, with n is number of TCP flows.
- By reducing buffer size (which could achieve 99%), we will:
  - Reduce power consumption, board space and increase density. (By using less memory chips)
  - Reduce end-to-end delays in presence of congestion. (By using faster memory such as SRAM instead DRAM)

Buffer size for a single long-lived TCP flow





- At t1, first packet gets timeout, sender reduces Wmax by 2, pauses and waits until Wmax/2 packets get ACK-ed before continue.
- Because buffer must be large enough to not go empty while sender pauses, we have: $B/C \geqslant (Wmax/2)/C \rightarrow B \geqslant Wmax/2$
- Besides, after sender resumes, it will need to send at rate $C = W/RTT$ (b/c buffer is now empty, no queuing delay). So,
  $C = W/RTT = (Wmax/2)/R\bar{T}T \rightarrow Wmax/2 = R\bar{T}T * C$.
- Thus, we will have: $B \geqslant R\bar{T}T * C$ to always keep bottleneck link busy.

When many long TCP flows share a link

Synchronized Long Flows:

- Become synchronized with each other $\rightarrow$ looks like an amplified version of one flow. $\rightarrow$ still follows $B \geqslant R\bar{T}T * C$ equation.

Desynchronized Long Flows:

- Thousand flows with different RTTs.
- As number of flows increases, amount of in-phase synchronized flows decreases. Thus, we can view as flows which is not synchronized at all. These desynchronized flows cancel out each other's queuing effect to router buffer.
- Utilization could be calculated as:

$$Util \geq erf\left(\frac{3\sqrt{3}}{2\sqrt{2}}\frac{B}{\frac{2\overline{T}_p \cdot C + B}{\sqrt{n}}}\right)$$

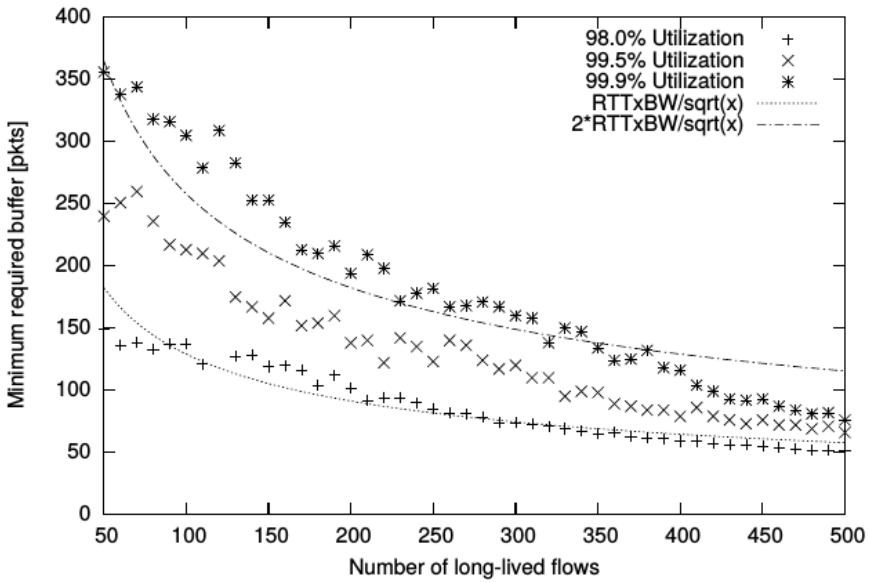
- Thus, with number of flows n = 1000, we have:

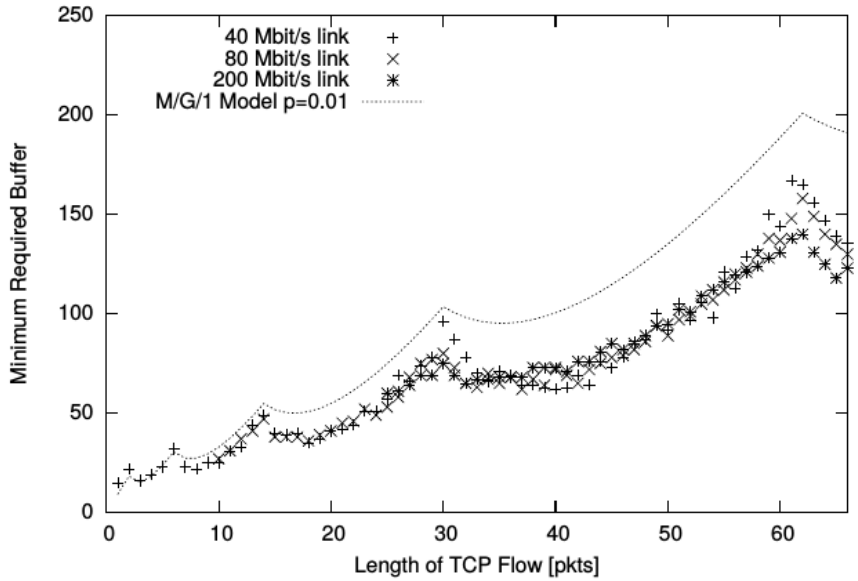
Router Buffer Size	Utilization
$B = 1 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util ≥ 98.99 %
$B = 1.5 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util ≥ 99.99988 %
$B = 2 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util ≥ 99.99997 %

Sizing the Router Buffer for short flows

- For short flows, size of buffer does not depend on line-rate, propagation delay, or number of flows but only depends on load of the link and length of the flows.
- Short-lived flows in general contributes very little to the buffering requirements. Thus, buffer size is usually determined by the number long-lived flows.

Simulation and Experimental result

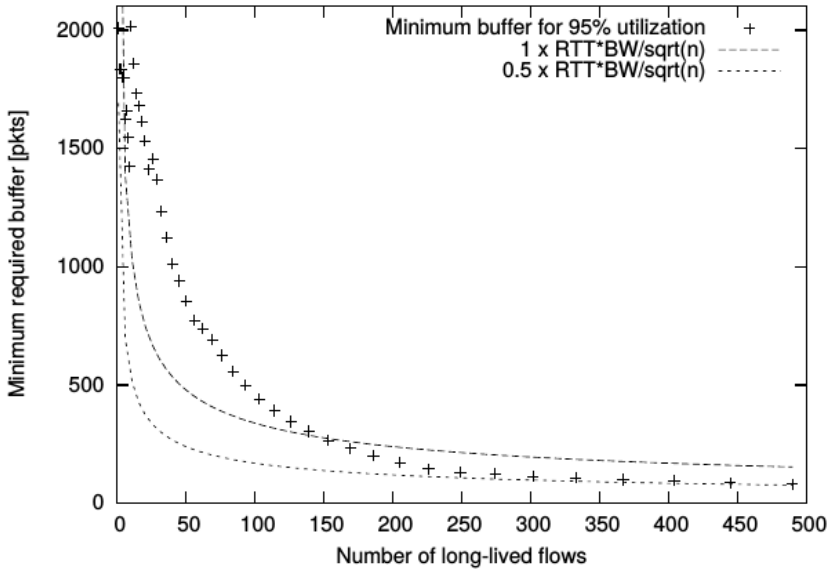


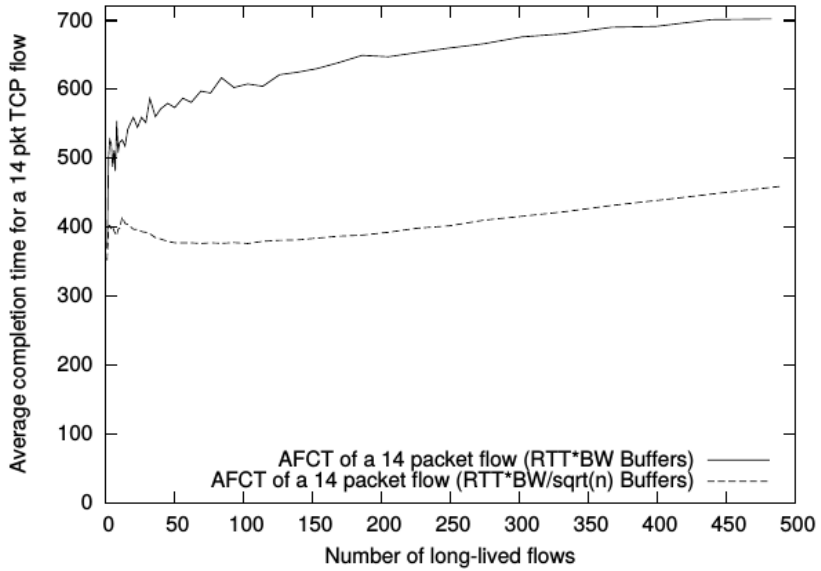


NS-2 simulation:

- When number of long live flows is small, the result doesn't hold but when number of flows is over 250, the model hold well.
- For short flows, the amount of needed buffer is not depend on number of flows, bandwidth or RTT but depends on load of the link and length of the burst.

Simulation and Experimental result



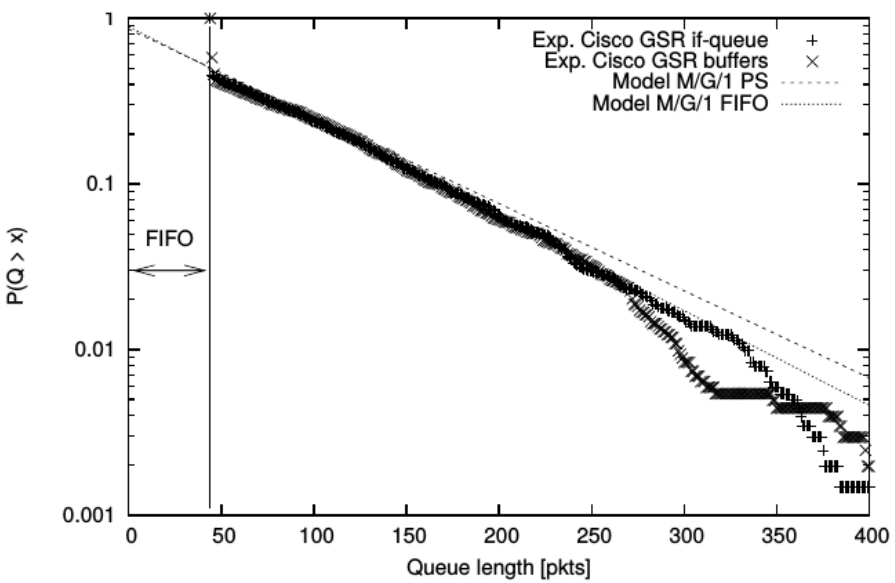


NS-2 simulation:

- With mixes of short and long-lived flows, the model still holds when number of long-lived flows increases.
- On the other hand, average flow completion time for buffer size $R\bar{T}T * C/\sqrt{n}$ is much better than buffer size $R\bar{T}T * C$ while we still achieve over 95% utilization.

Simulation and Experimental result

TCP Flows	Router Buffer $\frac{RTT \times BW}{\sqrt{n}}$	Pkts	RAM	Link Utilization (%) Model	Sim.	Exp.
100	0.5 x	64	1 Mbit	96.9%	94.7%	94.9%
100	1 x	129	2 Mbit	99.9%	99.3%	98.1%
100	2 x	258	4 Mbit	100%	99.9%	99.8%
100	3 x	387	8 Mbit	100%	99.8%	99.7%
200	0.5 x	46	1 Mbit	98.8%	97.0%	98.6%
200	1 x	91	2 Mbit	99.9%	99.2%	99.7%
200	2 x	182	4 Mbit	100%	99.8%	99.8%
200	3 x	273	4 Mbit	100%	100%	99.8%
300	0.5 x	37	512 kb	99.5%	98.6%	99.6%
300	1 x	74	1 Mbit	100%	99.3%	99.8%
300	2 x	148	2 Mbit	100%	99.9%	99.8%
300	3 x	222	4 Mbit	100%	100%	100%
400	0.5 x	32	512 kb	99.7%	99.2%	99.5%
400	1 x	64	1 Mbit	100%	99.8%	100%
400	2 x	128	2 Mbit	100%	100%	100%
400	3 x	192	4 Mbit	100%	100%	99.9%



Experimental on Cisco GSR 12410:

- For long-lived flows, model, simulation and experiment all agree that a router buffer should have a size equal to approximately $R\bar{T}T * C/\sqrt{n}$.
- Short-lived flows result is also similar to model and simulation.

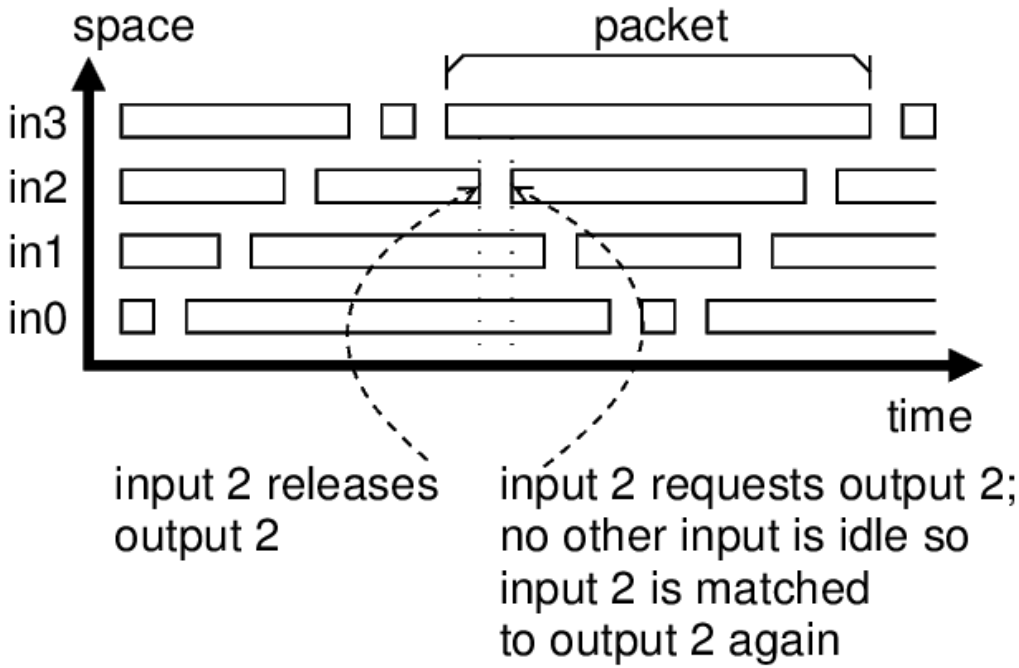
Discussion

Discussion

- The system authors proposed shows high performance that approximately to theoretical bound, but is this a scalable approach? What if using more ports, or using larger crossbar switch?
- The paper assumes the use of fixed length packets with a switched plane router due to its benefits. However, can the use of variable length packets be avoided and how can this be catered for at the same time guaranteeing good quality of service?
- The paper has stipulated the drawbacks of over buffering in the paper, but one may wonder what is the positive side of over buffering in backbone routers? What would be the impact of reducing the buffer size of a router for the future Internet?
- In a backbone router, when number of flows increases, the amount of in-phase synchronization decreases. As a result, we can consider these flows as being not synchronized at all. Why does this happen?

Appendix

Starvation scenario with variable-length units



Assume a 4 x 4 switch. Input i is initially connected to output i (for all i in [0,3]) and never switches output in the future. The figure shows the 4 flows that are constantly served; other flows with non-empty queues exist, but are never served. (**back**)

3 stages crossbars switch

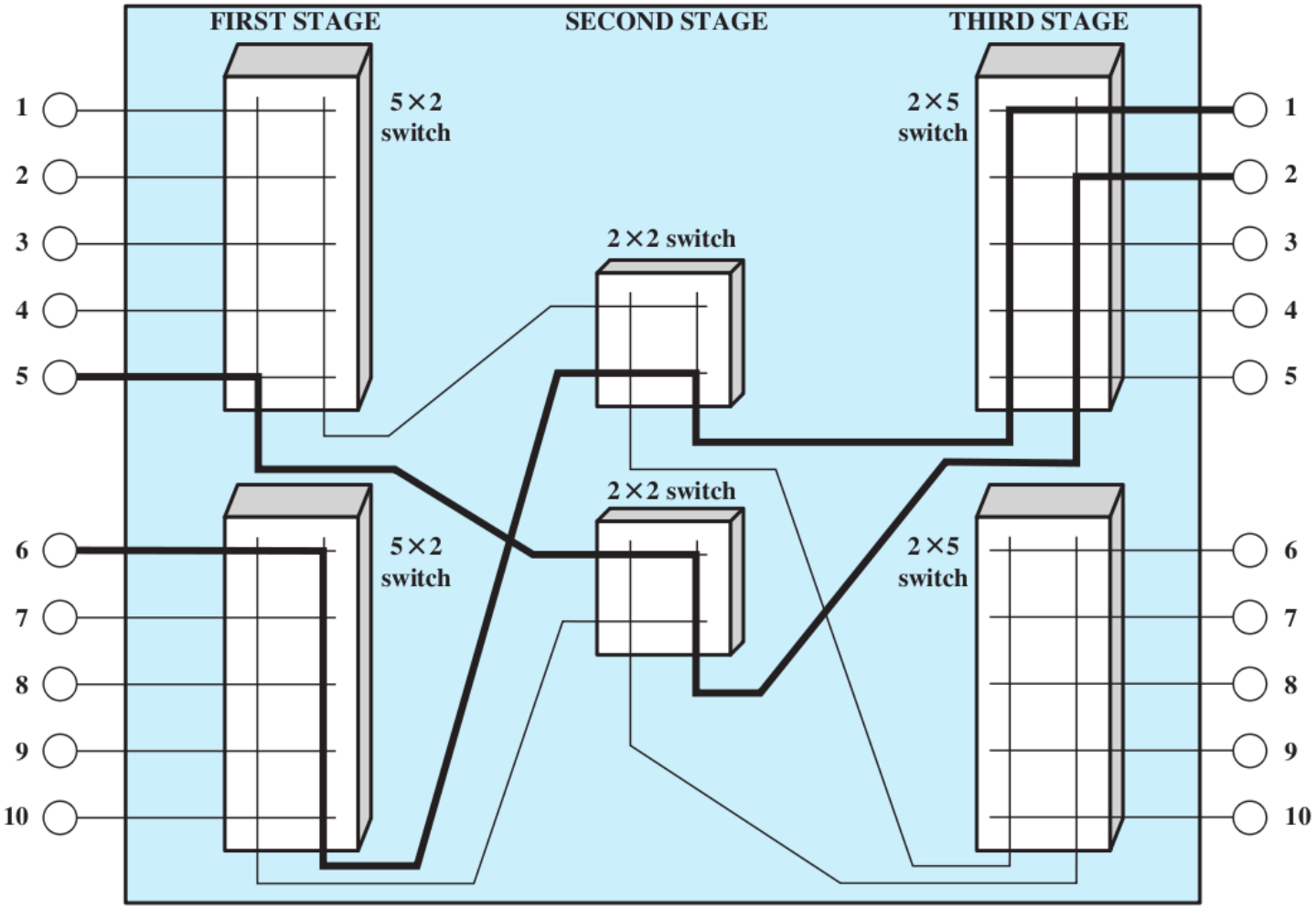
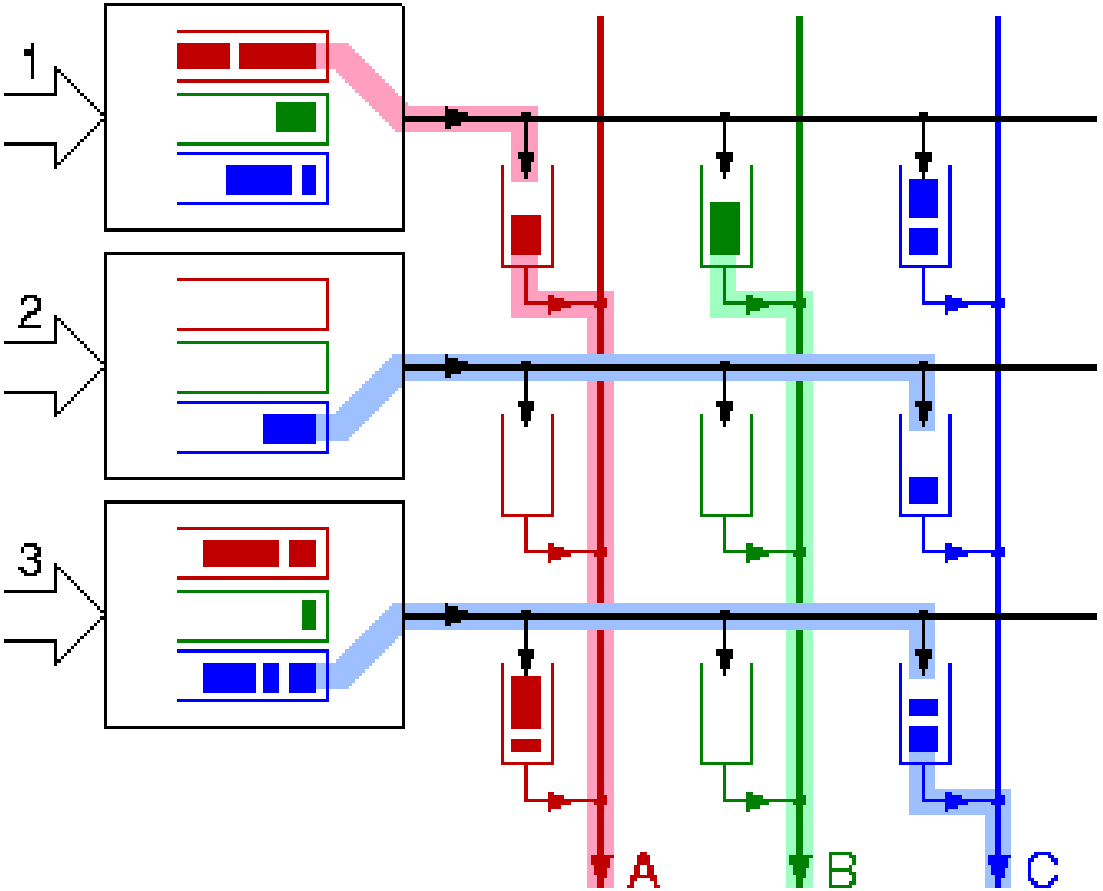


Figure 10.6 Three-Stage Space-Division Switch

Buffered cross-point



Thank you for your listening!