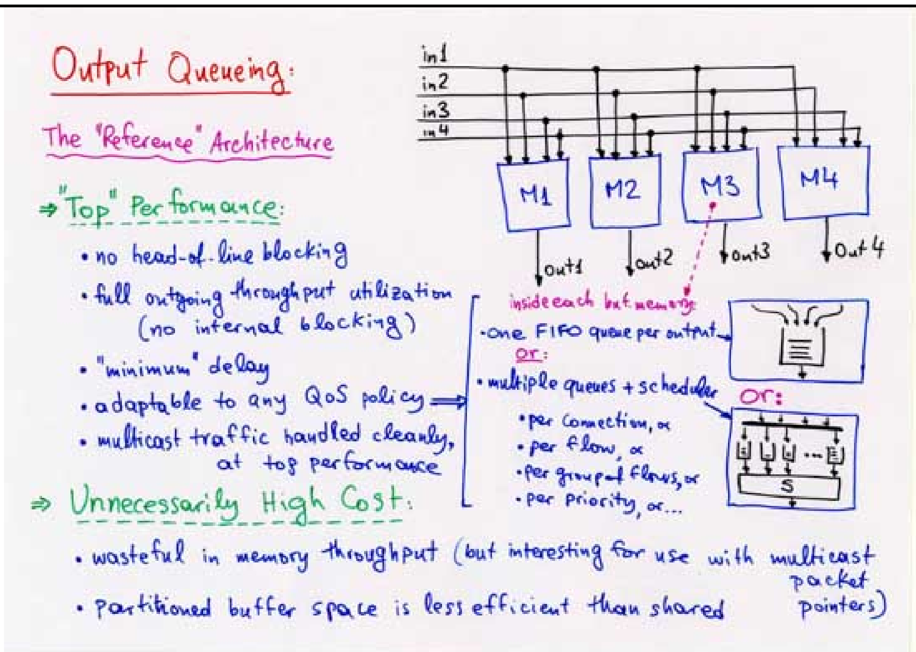
3.2 The Output Queueing / Shared Buffer family of Switch Queueing Architectures

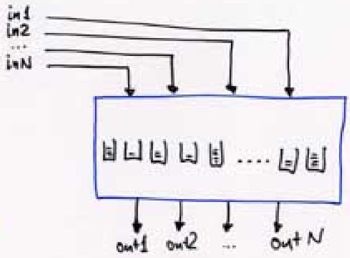
- Time Switching (this chapter):
  - Output Queueing (needlessly expensive): the reference architecture
  - Shared Buffer (improve space utilization): the best, when feasible
  - Crosspoint Queueing: high throughput, low space utilization
  - Block-Crosspoint Queueing: hybrid between shared-buffer & crossp.
- Space Switching (next chapter):
  - Input Queueing, HOL blocking, Virtual-Output Queues (VOQ)
- Combinations (next two chapters):
  - Internal Speedup
  - Switching Fabrics with Internal Buffering

 1



 2

Shared Buffer:

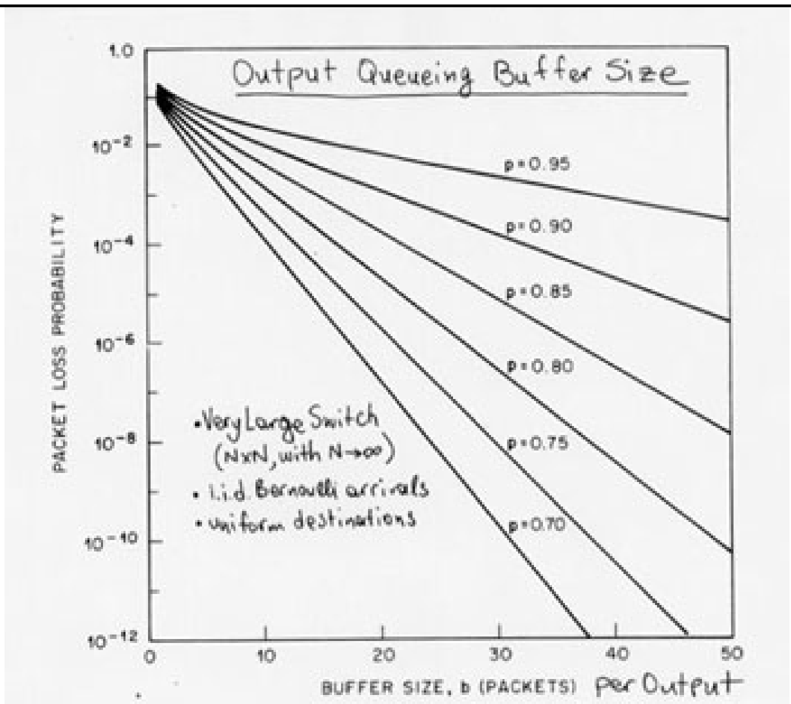


Top Performance at Low Cost for small N

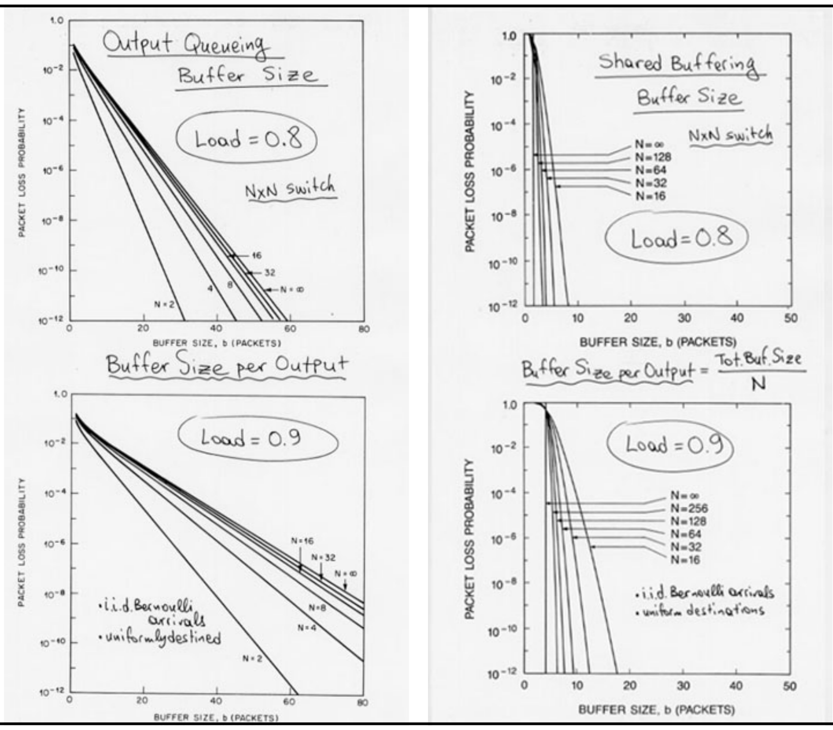
- total buffer memory throughput = 2N (versus N×(N+1) for output queueing)
- memory space is shared ⇒ better utilization
- same performance as output queueing for unicast traffic
  - multiple logical queues in a single memory, at least per output, possibly also per priority/flow)...
- for multicast packets: not enough throughput to enqueue each arriving packet into multiple (per output) queues. Hence, if fewer than 2^N multicast queues exist, some head-of-line blocking will occur in them. Interesting combination:
  - shared buffer for packet bodies
  - output queueing for queue pointers

Buffer Space Requirements

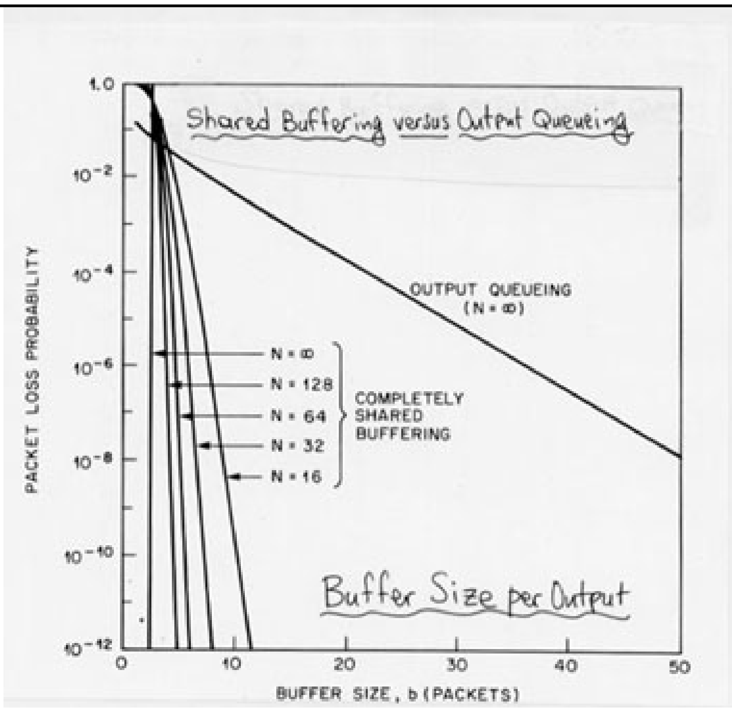
- When the incoming traffic consists of fixed-size packets from independent, identically distributed (i.i.d.) Bernoulli processes, with uniformly-distributed destination (output) ports, analysis and simulation have yielded the results plotted below.
- *Reference:* M. Hluchyj, M. Karol: "Queueing in High-Performance Packet Switching", IEEE Journal on Sel. Areas in Commun. (JSAC), vol. 6, no. 9, Dec. 1988, pp. 1587-1597
- *Attention:* results derived for i.i.d. Bernoulli (non-bursty) arrivals, with uniformly-distributed destinations (no overloaded hot-spots), are only useful for gaining a rough, first insight into the behavior of systems, but are often not representative of the real behavior of systems under real traffic!...


Output Queueing Buffer Size
PACKET LOSS PROBABILITY
1.0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
p=0.95
p=0.90
p=0.85
p=0.80
p=0.75
p=0.70
• Very Large Switch (NxN, with N→∞)
• i.i.d. Bernoulli arrivals
• uniform destinations
0 10 20 30 40 50
BUFFER SIZE, b (PACKETS) per Output

5


Output Queueing Buffer Size
Load = 0.8
NxN switch
N=2
4
8
16
32
N=∞
BUFFER SIZE, b (PACKETS)
Buffer Size per Output
Load = 0.9
• i.i.d. Bernoulli arrivals
• uniformly destined
N=16
N=32
N=∞
N=8
N=4
N=2
PACKET LOSS PROBABILITY
Shared Buffering Buffer Size
NxN switch
N=∞
N=128
N=64
N=32
N=16
Load = 0.8
BUFFER SIZE, b (PACKETS)
Buffer Size per Output = Tot. Buf. Size / N
Load = 0.9
N=∞
N=256
N=128
N=64
N=32
N=16
• i.i.d. Bernoulli arrivals
• uniform destinations

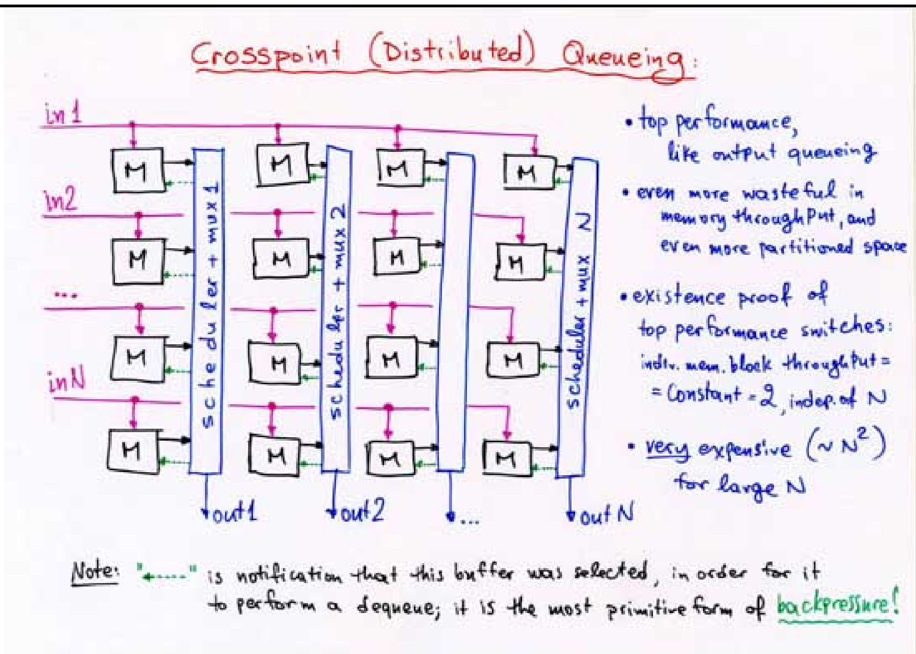
6



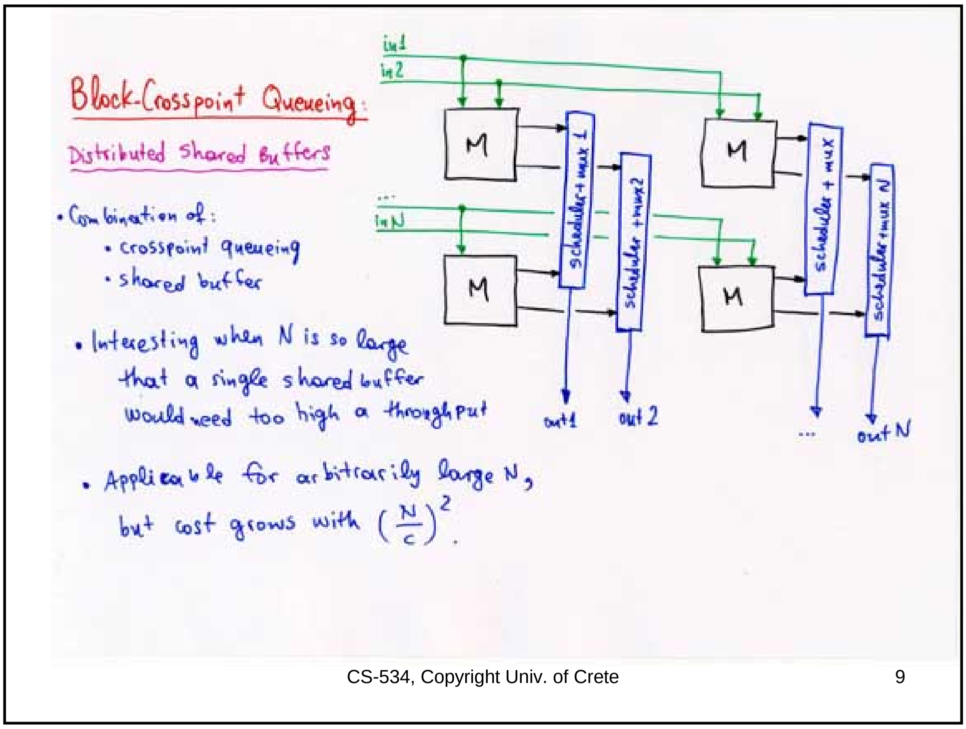
Shared Buffering versus Output Queueing
1.0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
PACKET LOSS PROBABILITY
OUTPUT QUEUEING (N = ∞)
N = ∞
N = 128
N = 64
N = 32
N = 16
COMPLETELY SHARED BUFFERING
Buffer Size per Output
0
10
20
30
40
50
BUFFER SIZE, b (PACKETS)

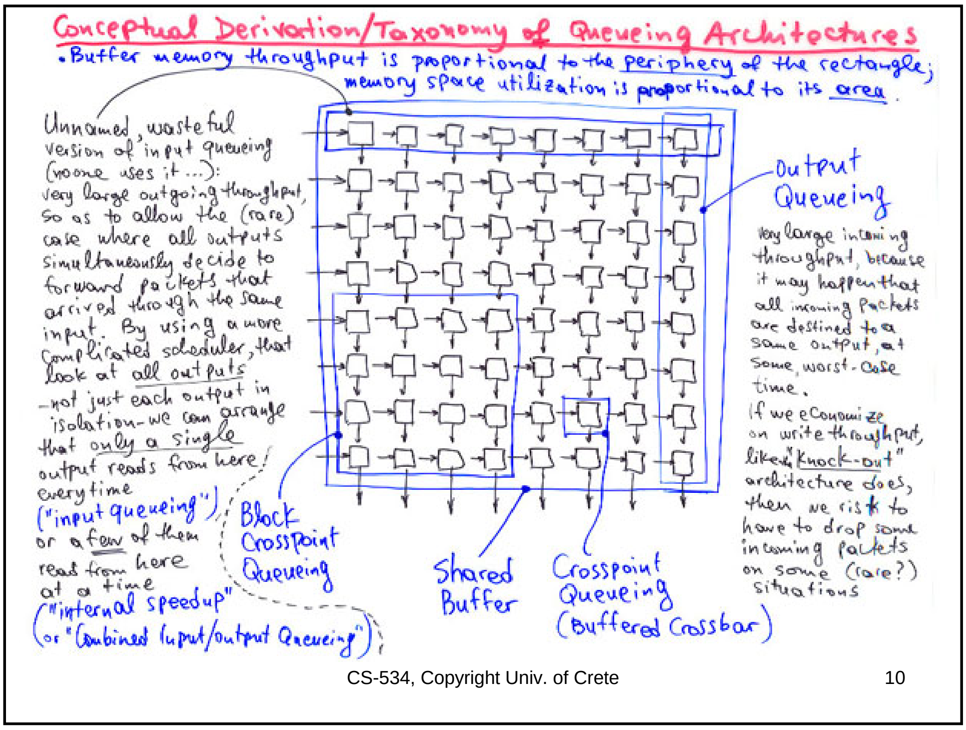
Load = 85 %

7

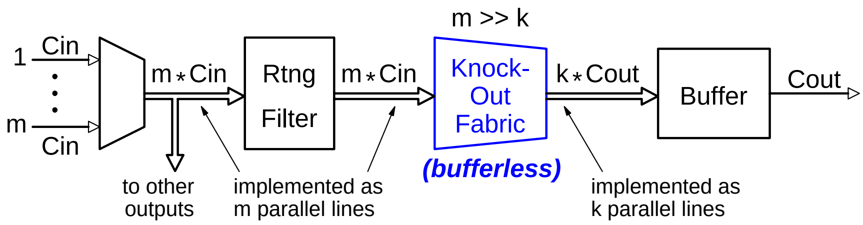


Crosspoint (Distributed) Queueing:
in 1
in2
...
inN
M
scheduler + mux 1
scheduler + mux 2
scheduler + mux N
out1
out2
...
out N
• top performance, like output queueing
• even more wasteful in memory throughput, and even more partitioned space
• existence proof of top performance switches: indiv. mem. block throughput = = Constant = 2, indep. of N
• very expensive (~ N^2) for large N
Note: "←----" is notification that this buffer was selected, in order for it to perform a dequeue; it is the most primitive form of backpressure!


Block-Crosspoint Queueing:
Distributed Shared Buffers
• Combination of:
• crosspoint queueing
• shared buffer
• Interesting when N is so large that a single shared buffer would need too high a throughput
• Applicable for arbitrarily large N, but cost grows with (N/c)²
in1
in2
inN
M
scheduler + mux 1
scheduler + mux 2
scheduler + mux
scheduler + mux N
out1
out 2
out N
CS-534, Copyright Univ. of Crete
9


Conceptual Derivation/Taxonomy of Queueing Architectures
• Buffer memory throughput is proportional to the periphery of the rectangle; memory space utilization is proportional to its area.
Unnamed, wasteful version of input queueing (noone uses it ...): very large outgoing throughput, so as to allow the (rare) case where all outputs simultaneously decide to forward packets that arrived through the same input. By using a more complicated scheduler, that look at all outputs - not just each output in isolation - we can arrange that only a single output reads from here every time ("input queueing") or a few of them read from here at a time ("internal speedup" (or "Combined Input/output Queueing"))
Output Queueing
very large incoming throughput, because it may happen that all incoming packets are destined to a same output, at some, worst-case time.
If we economize on write throughput, like the "Knock-out" architecture does, then we risk to have to drop some incoming packets on some (rare?) situations
Block Crosspoint Queueing
Shared Buffer
Crosspoint Queueing (Buffered Crossbar)
CS-534, Copyright Univ. of Crete
10

Other Variations of Output Queueing: **Knock-Out Switch**



Knock-Out Fabric:

- has m inputs and k outputs, k << m
- passes on up to k non-idle packets to its outputs
- when more than k packets arrive in the same time slot, all destined to the same output, k of them are passed and the rest are dropped
- if the traffic is uniformly destined, and k is on the order of 8 to 12, packets will rarely be dropped

See: Yeh, Hluchyj, Acampora: IEEE JSAC, October 1987, pp. 1274-1283.

11