3. **Time Switching, Multi-Queue Memories, Shared Buffers, Output Queueing Family**

3.1 TDM, Time Switching, Cut-Through
3.2 Wide Memories for High Thruput, Segm’tn Ovrhd
3.3 Multiple Queues within a Buffer Memory
3.4 Queueing for Multicast Traffic
3.5 Shared Buffering and the Output Q’ing Family

*Manolis Katevenis*

CS-534 – Univ. of Crete and FORTH, Greece

http://archvlsi.ics.forth.gr/~kateveni/534
3.5 Output Queueing & Shared Buffer Family

Table of Contents:

• 3.5.1 The Output Queueing Family: Review
  – Buffer Memories in Output Queueing and in Shared Buffering
  – Per-output queue structures (& per-priority, per-flow-group)

• 3.5.2 Buffer Space Requirements Analysis Results
  – Caution: uniformly-destined, Bernoulli i.i.d. traffic…

• 3.5.3 Crosspoint Queueing and Generalizations
  – Block-Crosspoint Queueing, “shape” of the block

• (other, old: knock-out switch)
Output Queueing:

The "Reference" Architecture

⇒ "Top" Performance:
  - no head-of-line blocking
  - full outgoing throughput utilization (no internal blocking)
  - "minimum" delay
  - adaptable to any QoS policy
  - multicast traffic handled cleanly, at top performance

⇒ Unnecessarily High Cost:
  - wasteful in memory throughput (but interesting for use with multicast packet pointers)
  - partitioned buffer space is less efficient than shared...
**Shared Buffer:**

Top Performance at Low Cost for small $N$

- total buffer memory throughput $= 2N$
  (versus $N \times (N+1)$ for output queuing)
- memory space is shared $\Rightarrow$ better utilization

- same performance as output queuing for unicast traffic
  - multiple logical queues in a single memory,
    at least per output, possibly also per priority/flow/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 queuing for queue pointers
3.5.2 Buffer Space Requirements: Analysis Results

• Analysis & simulation have yielded the results plotted below

• **Assuming** that the input traffic consists of packets with/from:
  – uniformly-distributed destination (output) ports,
  – independent, identically distributed (i.i.d.) Bernoulli processes,
  – fixed-size packet (cell) traffic

• **Attention:** results derived for i.i.d. Bernoulli (non-bursty) arrivals, with uniformly-distributed destination (no overloaded hot-spot output ports), are only useful for gaining a first, rough insight into the behavior of systems, but are usually not representative of the real behavior of systems under real traffic patterns!...
Output Queueing Buffer Size

Packet Loss Probability

- Very Large Switch (N x N, with N → ∞)
- i.i.d. Bernoulli arrivals
- Uniform destinations

Buffer Size, b (Packets) per Output
Load = 85%
Crosspoint (Distributed) Queueing:

- Top performance, like output queuing
- Even more wasteful in memory throughput, and even more partitioned space
- Existence proof of top performance switches: indiv. mem. block throughput = \( \text{Constant} = 2 \), indep. of \( N \)
- Very expensive (\( \sim N^2 \)) for large \( N \)

Note: "\( \ldots \)" 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 $(\frac{N}{C})^2$. 
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 (someone 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 a "knock-out" architecture does, then we risk to have to drop some incoming packets on some (rare?) situations.
**Other Variations of Output Queueing:**

**Knock-Out Switch**

![Knock-Out Switch Diagram]

**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.