Switching
Nick McKeown

“High-speed switch scheduling for local-area networks”
[Tom Anderson, Susan Owicki, James Saxe, Chuck Thacker. 1993]
Context

Tom Anderson
At the time: DEC SRC (Palo Alto)
Professor of CS, University of Washington
Previously: UC Berkeley, EECS

Susan Owicki
At the time: DEC SRC (Palo Alto)
Before that: Prof of EE & CS, Stanford
Today: Marriage and Family Therapist, Palo Alto

James B. Saxe
At the time: DEC SRC (Palo Alto)
After that: Compaq and HP Labs

Chuck Thacker (d. 2017)
At the time: DEC SRC (Palo Alto)
Before that: Xerox PARC (“Alto”)
After that: Microsoft
2010 Turing Award Winner

At the time the paper was written…
• WWW was new, and Internet traffic was growing fast
• Fastest Ethernet networks ran at 100Mb/s
• Lots of interest in building faster switches and routers
• Lively debate about an alternative to the Internet, called “ATM”
But first...
A few words about packet queues…

\[ R = \text{line rate.} \]
\[ \text{e.g. 100M bit/s, 10Gb/s} \]

**Packet buffer**

**Observation:** With one arrival “line” at the same rate, the queue is always empty (or at most one store-and-forward packet). The arrival process is “bounded” by \( R \).

**Q:** For any “load” \( \lambda \leq 1 \), what arrival pattern leads to the most customers in the queue?

**Q:** For any “load” \( \lambda \leq 1 \), what arrival pattern leads to the most customers in the queue?

**Observation:** The arrival rate is “bounded” by \( R \) on average.
Different cases for $\lambda = 1$

1. line 1
   - Q: How big does the buffer need to be?

2. line 1
   - Q: How big does the buffer need to be?

3. line 1
   - Q: How big does the buffer need to be?

Observation: For a given arrival rate, in order to know the queueing delay, we need to know the pattern (or “process”) of arrivals.
A switch, or router, with \( N \) “ports”. Each port runs at rate \( R \) b/s. We say the “switching capacity” is \( N \times R \) b/s.
An output-queued (OQ) switch

Properties of an OQ switch
• All buffering takes place at the output.
• Output queues must be able to write packets at rate $N \times R$.

Consequences
• “Work conserving”: Whenever there is a packet in the system, its output is busy sending a packet. No unnecessary idling.
• Average delay is minimized.
• But memory bandwidth limits the switching capacity.
Traffic Matrix

Traffic matrix, $\Lambda = [\lambda_{i,j}]$

$\lambda_{i,j}$ is the fraction of traffic from input $i$ to output $j$

For example:

$\Lambda = \begin{bmatrix}
0.1 & 0.2 & 0.2 & 0.4 \\
0.2 & 0.3 & 0.1 & 0.1 \\
1.0 & 0.0 & 0.0 & 0.0 \\
0.1 & 0.4 & 0.3 & 0.1 \\
\end{bmatrix}$

Note that the row (input) sum: $\sum_j \lambda_{i,j} \leq 1$, $\forall i$

Non-oversubscribed TM:

Total traffic rate to each output is $\leq 1$

$\sum_i \lambda_{i,j} \leq 1$, $\forall j$

and still: $\sum_j \lambda_{i,j} \leq 1$, $\forall i$

Uniform Traffic Matrix:

$\Lambda = \lambda \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}$

where: $\lambda \leq 1/N$
OQ Switches and “100% Throughput”

If we send traffic according to any non-over-subscribed traffic matrix to an OQ switch (with infinite buffers) then the output rates correspond to the column sums.

\[ \text{i.e. The traffic rate at output } j = R \sum_i \lambda_{i,j} \leq R \]

Put another way, an OQ switch can “keep up” with any reasonable traffic matrix we throw at it.

We often say an OQ switch can “sustain 100% throughput”.

Q: What happens if the buffers are finite?
An input-queued (IQ) switch

Properties of an IQ switch
- All buffering takes place at the input.
- Input queues only need to be able to write packets at rate $R$ (instead of $N \times R$).

Consequences
- Can build a switch $N$ times faster.
- But, a packet can be held up by packet ahead destined to a different output.
- Hence an IQ switch is not “work conserving”. It can unnecessarily idle.
- May not achieve “100% throughput”.
- Average delay is not minimized.
Head of Line Blocking
Head of Line Blocking

IQ switch with uniform traffic matrix, \( \lambda \leq 1 \)

**Observation**: HOL Blocking means we lose 42% of the switching capacity

\[ E(d) = \frac{1}{2} \left( \frac{2 - \lambda}{1 - \lambda} \right) \]

Poisson arrivals:
\[ \lambda \leq 2 - \sqrt{2} \approx 58\% \]

Karol '87
What does the “58%” result mean?

\[
\lambda, \mu \leq 1
\]

OQ switch

\[
\lambda R, \mu R
\]

IQ switch uniform TM, Poisson

\[
\lambda R, 0.58 R
\]
Virtual Output Queues (VOQs)
Basic idea

With a VOQ, a packet cannot be held up by a packet in front of it, destined to a different output.

Q: With VOQs, does/can 58% become 100% throughput?

IQ switch uniform TM, Poisson

IQ switch with VOQs Any TM, Any arrivals

\[ \lambda R \rightarrow \text{Arrival rate} \]
\[ 0.58 R \rightarrow \text{Departure rate} \]
100% Throughput

Reminder: “100% throughput” is equivalent to
For a non over-subscribing traffic matrix, queues don’t grow without bound.
\[ \mu \geq \lambda \] for every queue in the system.

Observations:
1. Burstiness of arrivals does not affect throughput
2. For a uniform Traffic Matrix, solution is trivial!
An input-queued (IQ) switch with VOQs and a crossbar

Observation: scheduling is equivalent to choosing a permutation.
$N^2$ VOQs

bipartite request graph

bipartite match

e.g. “maximum size match”
crossbar
Crossbar schedule

Fixed cycle of permutations:

$\lambda \leq 1$, therefore arrival rate $\leq$ departure rate.
True for all VOQs, therefore 100% throughput for uniform TM schedule.
100% throughput for uniform traffic

Four (trivial) algorithms for a uniform traffic matrix:

1. Cycle through permutations in “round-robin” (i.e. previous slide).
2. Each time, randomly pick one of the permutations in (1).
3. Each time, pick a permutation uniformly and at random from all possible N! permutations.
4. Wait until all VOQs are non-empty, then pick any algorithm above.
Quick recap so far
An input-queued (IQ) switch

Properties of an IQ switch
- All buffering takes place at the input.
- Input queues only need to be able to write packets at rate $R$ (instead of $N \times R$).

Consequences
- Can build a switch $N$ times faster.
- HOL Blocking: a packet can be held up by packet ahead destined to a different output.
- Hence an IQ switch is not “work conserving”. It can unnecessarily idle.
- May not achieve “100% throughput”.
- Average delay is not minimized.
Head of Line Blocking

IQ switch with uniform traffic matrix, $\lambda \leq 1$

**Observation:** HOL Blocking means we lose 42% of the switching capacity

Poisson arrivals:

$\lambda \leq 2 - \sqrt{2} \approx 58\%$

Karol '87

Poisson arrivals:

$E(d) = \frac{1}{2} \left( \frac{2 - \lambda}{1 - \lambda} \right)$
100% throughput easy for uniform traffic

Four (trivial) algorithms for a uniform traffic matrix:
1. Cycle through permutations in “round-robin”.
2. Each time, randomly pick one of the permutations in (1).
3. Each time, pick a permutation uniformly and at random from all possible N! permutations.
4. Wait until all VOQs are non-empty, then pick any algorithm above.
Q: So why did the authors need Parallel Iterative Matching (PIM)?

Because in practice, arrivals are **not** uniform.
(If we know the matrix, we can **still** create a cycle of permutations to serve every VOQ at the rate in the traffic matrix).

In practice we don’t know the traffic matrix.

Hence, PIM....
Parallel Iterative Matching

A maximal bipartite match

Iteration 1:

Request

Grant

Accept

Iteration 2:

Q: Are we done?
Q: Is a larger match possible?
PIM Properties

1. Inputs and outputs make decisions independently and in parallel.
2. Guaranteed to find a maximal match in at most $N$ iterations.
3. Typically completes in much fewer than $N$ iterations.

Q: How large is a maximal match compared to a maximum match?

A maximal match is guaranteed to be at least half the cardinality (size) of a maximum match.
Parallel Iterative Matching

Note log scale

Simulation
16-port switch
Uniform traffic matrix
Parallel Iterative Matching

Simulation
16-port switch
Uniform traffic matrix

- IQ + FIFO
- VOQ + Maximum Size Match
- Output Queued

PIM with one iteration
Parallel Iterative Matching

- **IQ + FIFO**
- **VOQ + Maximum Size Match**
- **Output Queued**

Simulation
16-port switch
Uniform traffic matrix

- PIM with one iteration
- PIM with four iterations
How many PIM iterations should be run?
Parallel Iterative Matching

*Number of iterations*

Consider the $n$ requests to output $j$

\[
\begin{align*}
\text{Requesting inputs receiving no other grants} & \quad k \quad \text{\{all requests to } j \text{ are resolved} \\
\text{Requesting inputs receiving other grants} & \quad n-k
\end{align*}
\]

\[
\begin{align*}
\text{w.p.} & \quad \frac{k}{n}, \text{ at most } k \text{ remain unresolved} \\
E[\text{Num unresolved requests}] & \leq \frac{k}{n} \cdot 0 + \left(1 - \frac{k}{n}\right) \cdot k \\
& \leq \frac{n}{4}, \text{ because } (1 - a) \cdot a \leq \frac{1}{4}, \text{ when } a < 1
\end{align*}
\]

Therefore, $3/4$ of all requests are resolved each iteration.

(It follows that the number of iterations $\leq \log_2 N + \frac{4}{3}$)
Known methods for non-uniform traffic

1. 100% throughput is now known to be theoretically possible with:
   - IQ switch, with VOQs, and
   - An arbiter to pick a permutation to maximize the total matching weight (e.g. weight is VOQ occupancy)

M, Walrand and Anantharam, 1996
Observation: give preference to longer VOQs
Leads to 100% throughput for any traffic matrix.

Choose matching $M$ that maximizes $\sum_{i,j \in M} L_{i,j}$

“maximum WEIGHT match”
Known methods for non-uniform traffic

2. It is practically possible with:
   - IQ switch, VOQs, all running *twice as fast* (i.e. choose and transfer two cells per cell time)
   - An arbiter running a *maximal* match (e.g. PIM)

**Intuition:** Because maximal match is at least half the size of a maximum match, running twice as fast compensates for it.
Known methods for non-uniform traffic

3. 2 switch stages with a fixed schedule of permutations!
A 2-stage Load-balancing switch

Fixed cycle of permutations

Fixed cycle of permutations

N² VOQs

Intuition: If uniform traffic is so easy, can I make non-uniform traffic “sufficiently uniform”?
A 2-stage Load-balancing switch

Deceptively simple but works for non-uniform traffic!

Q: Where is the switching taking place?
Q: Can packets be mis-sequenced?
End.