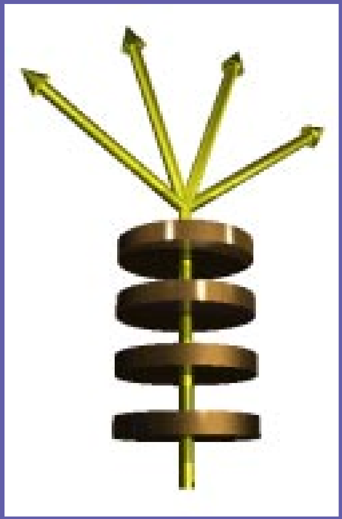


High Performance Switching and Routing

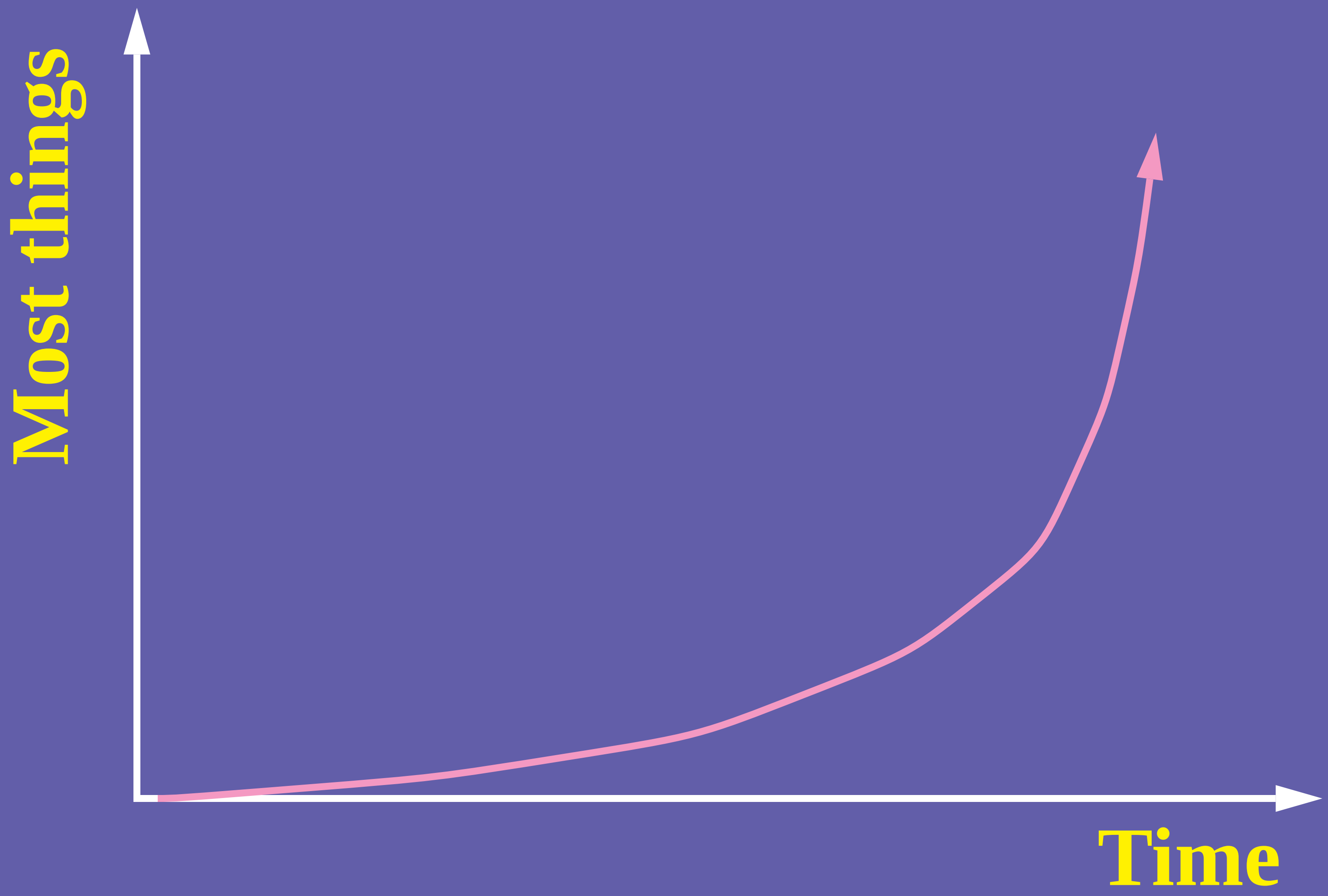
Nick McKeown

Assistant Professor of Electrical Engineering and Computer Science

nickm@stanford.edu
http://www.stanford.edu/~nickm

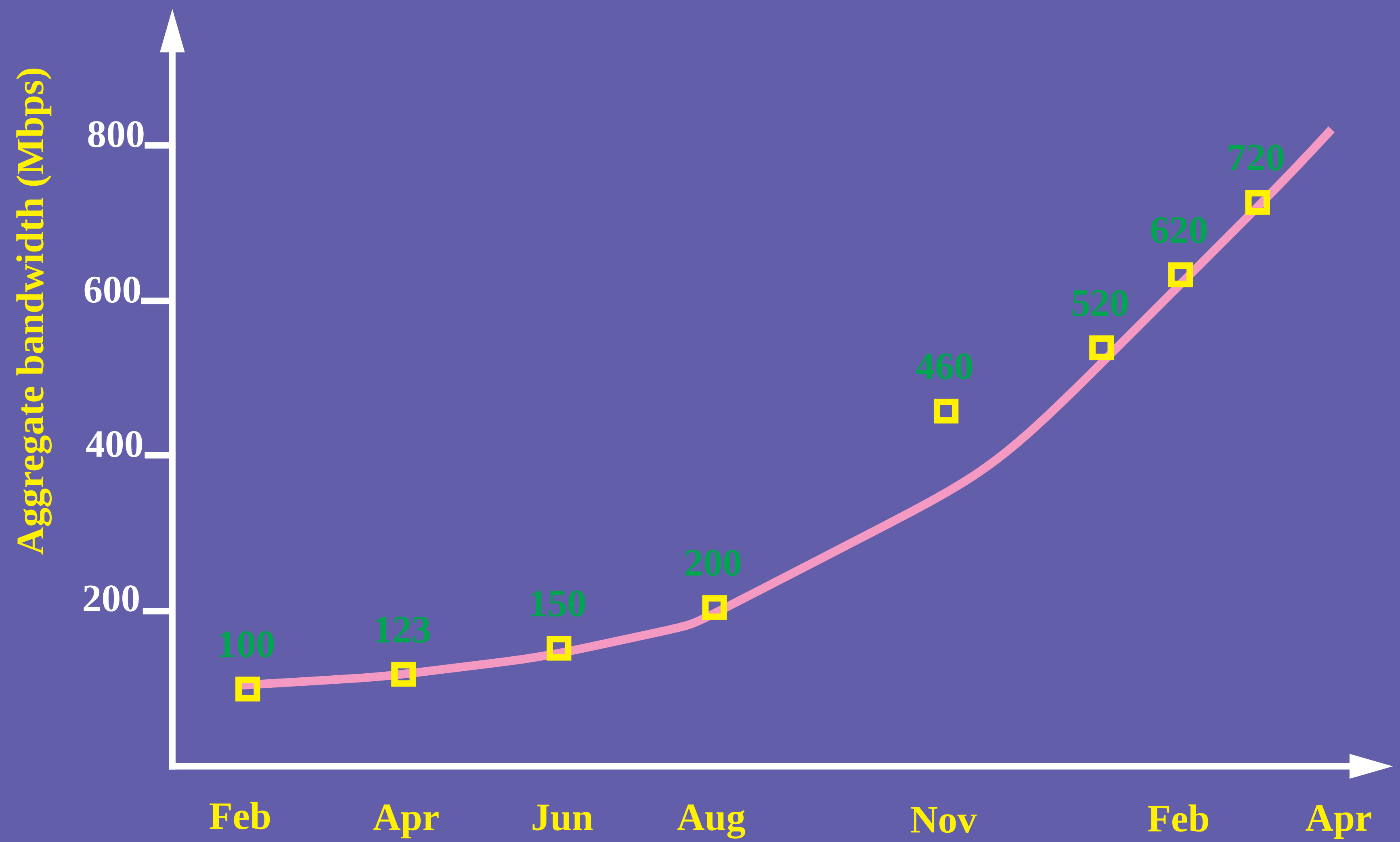
1. The Demand for Bandwidth
2. The Shortage of Switching/Routing Capacity
3. The Architecture of Switches and Routers
4. Some (of our) solutions

What's the Problem?



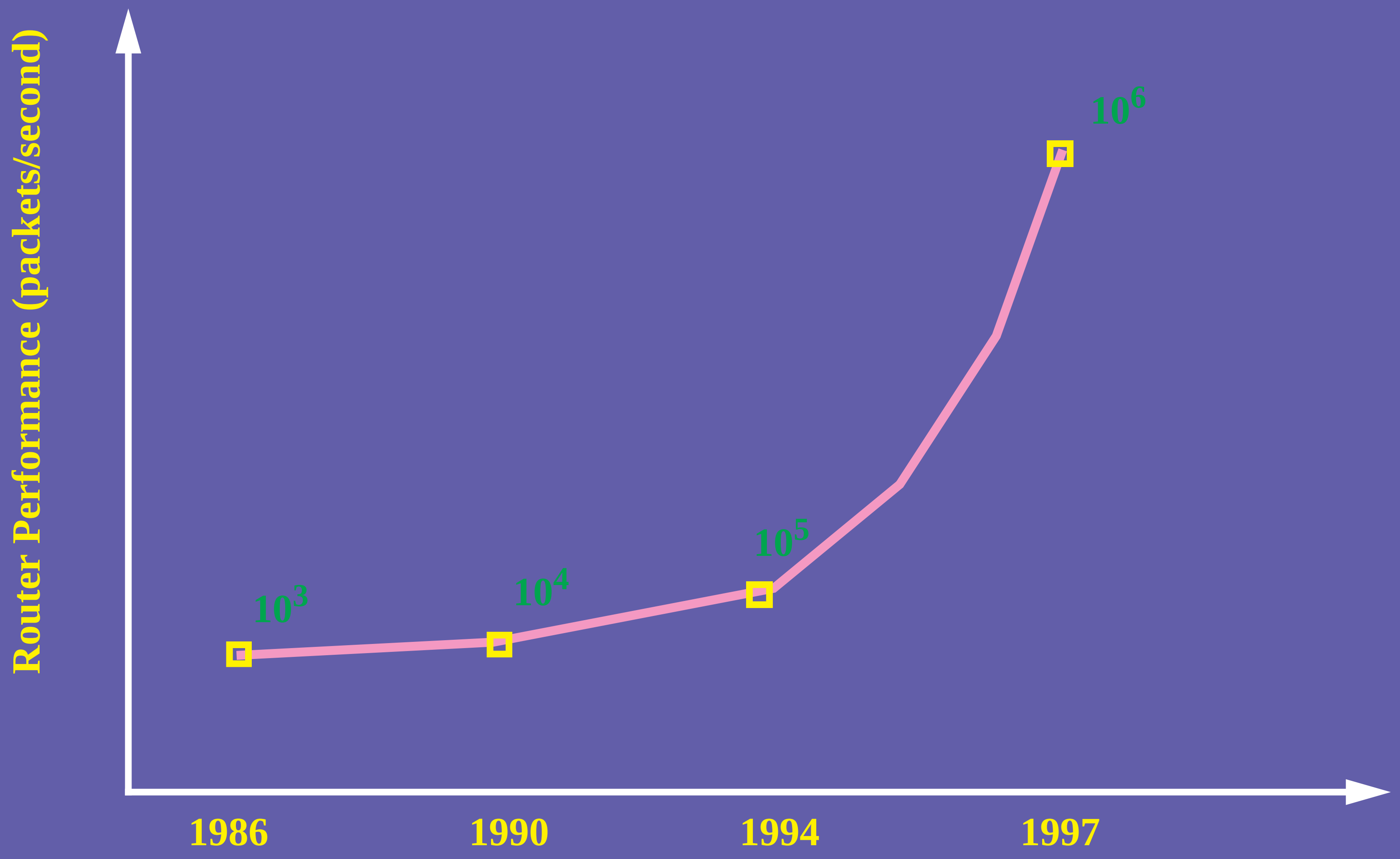
The demand

The San Jose NAP



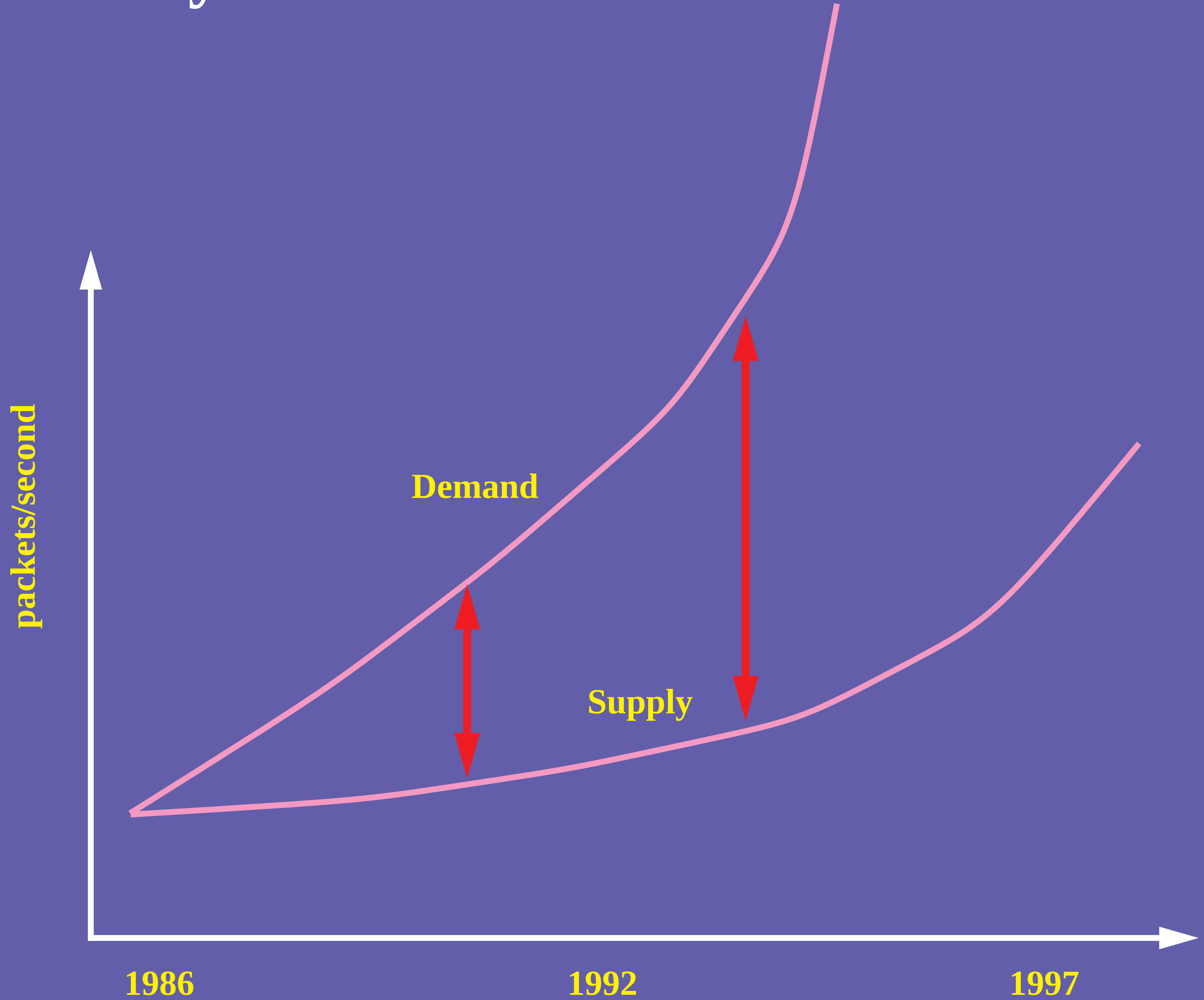
Source: http://www.mfsdatanet.com/MAE/west.stats.html

The supply



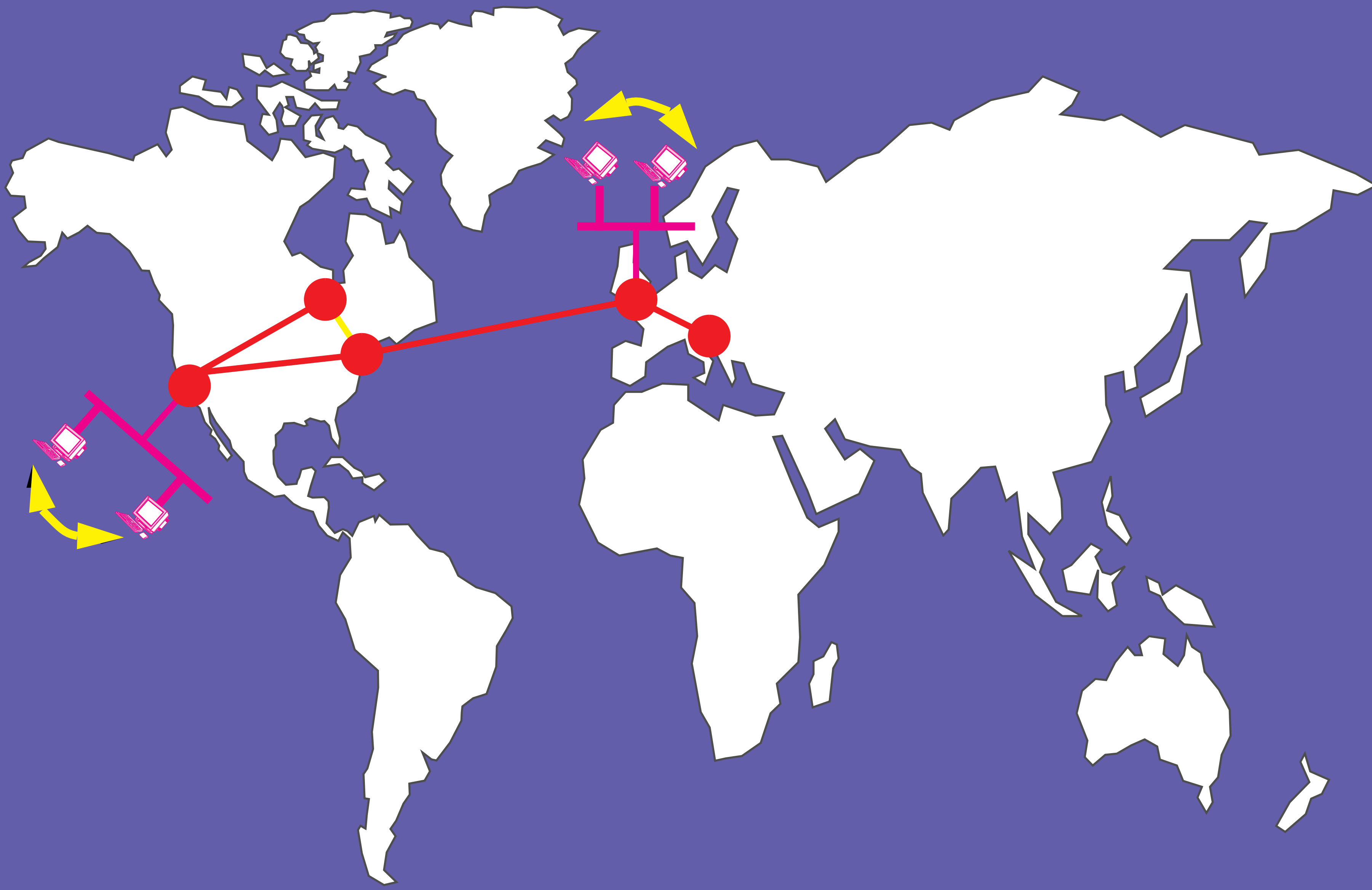
Router Performance (packets/second)
10^3
10^4
10^5
10^6
1986
1990
1994
1997

Why we need faster switches/routers



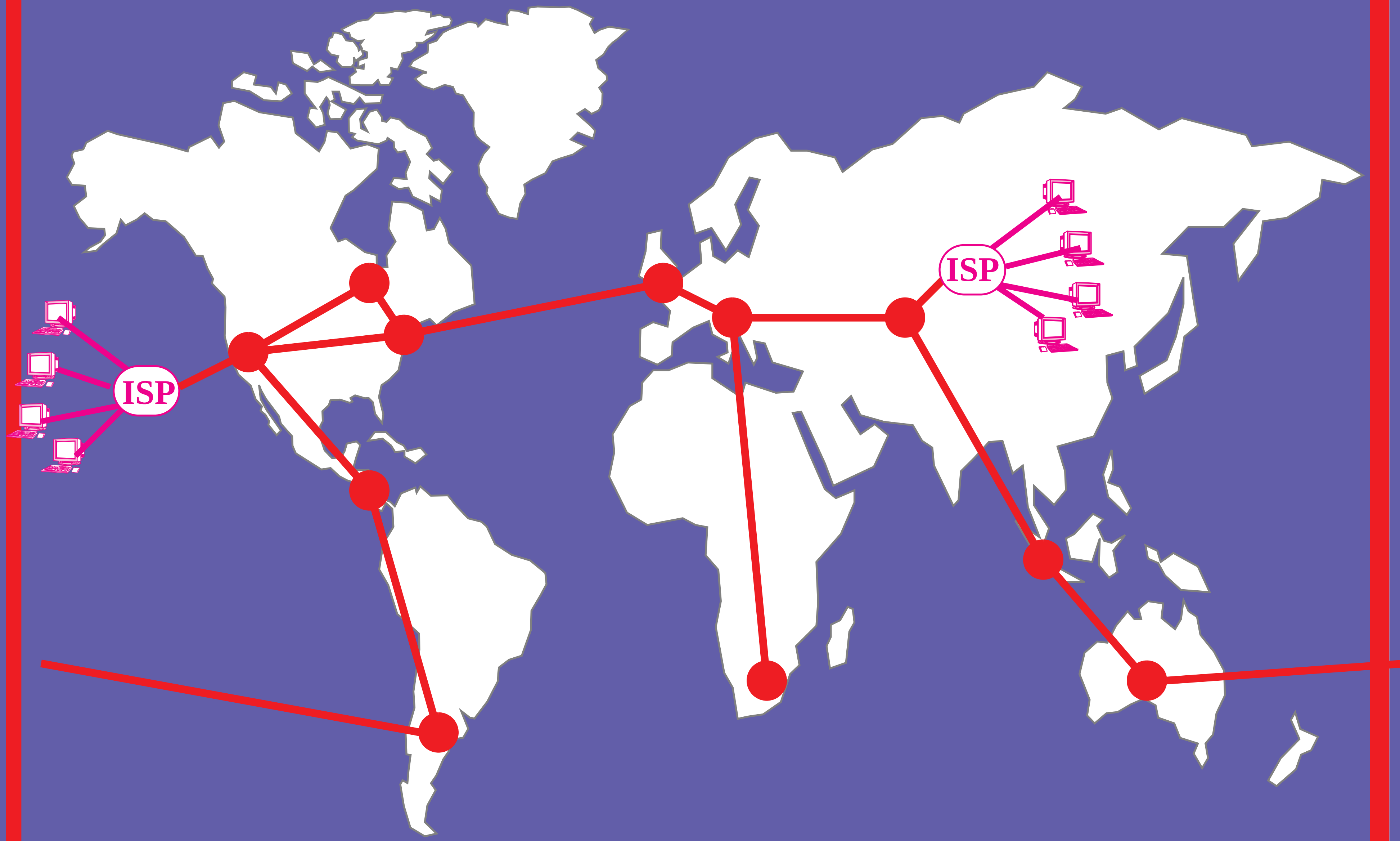
Traffic Inversion

10 years ago

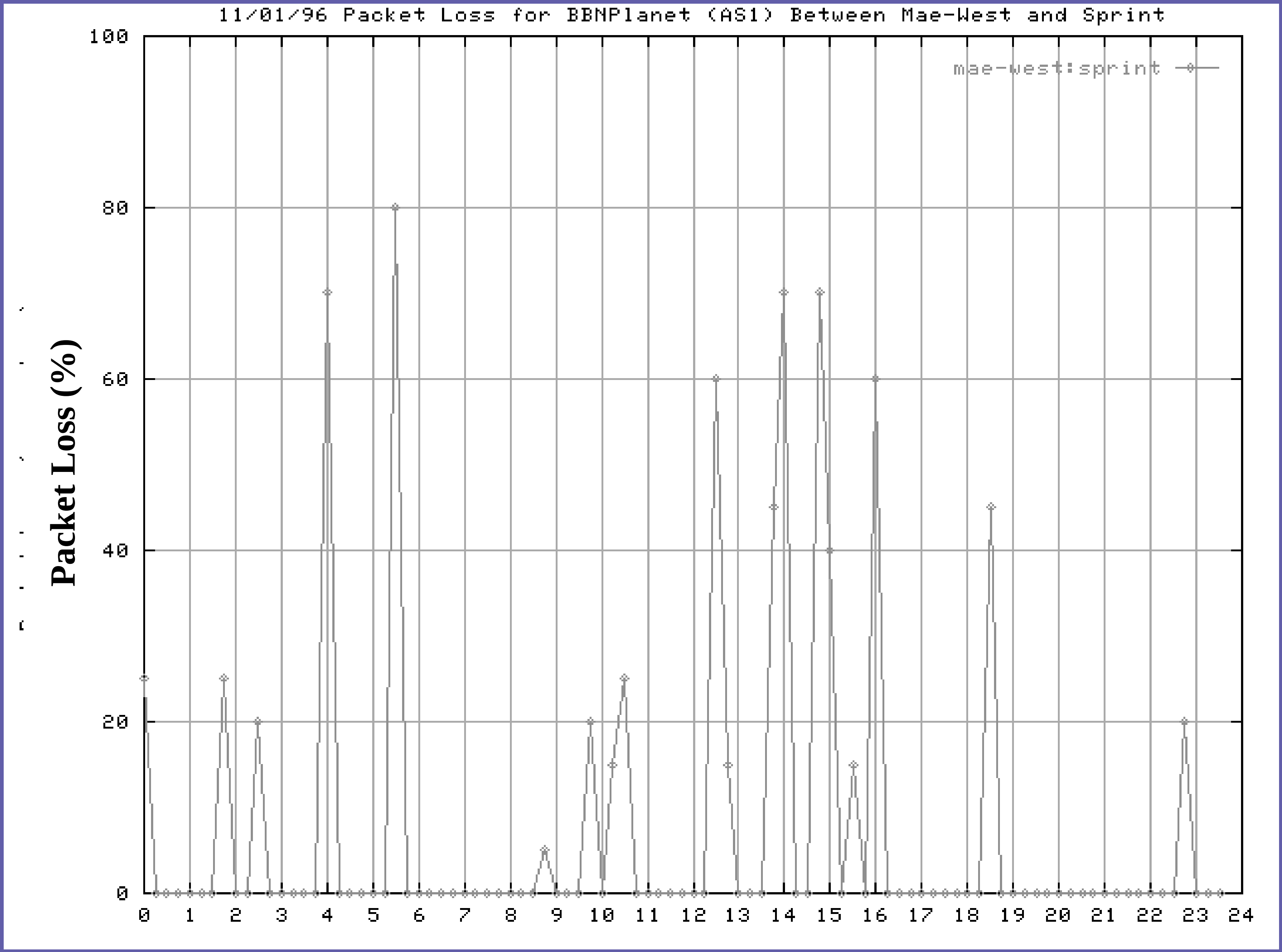


Traffic Inversion

Today



Why is this a problem?



November 1st, 1996

1. The Demand for Bandwidth

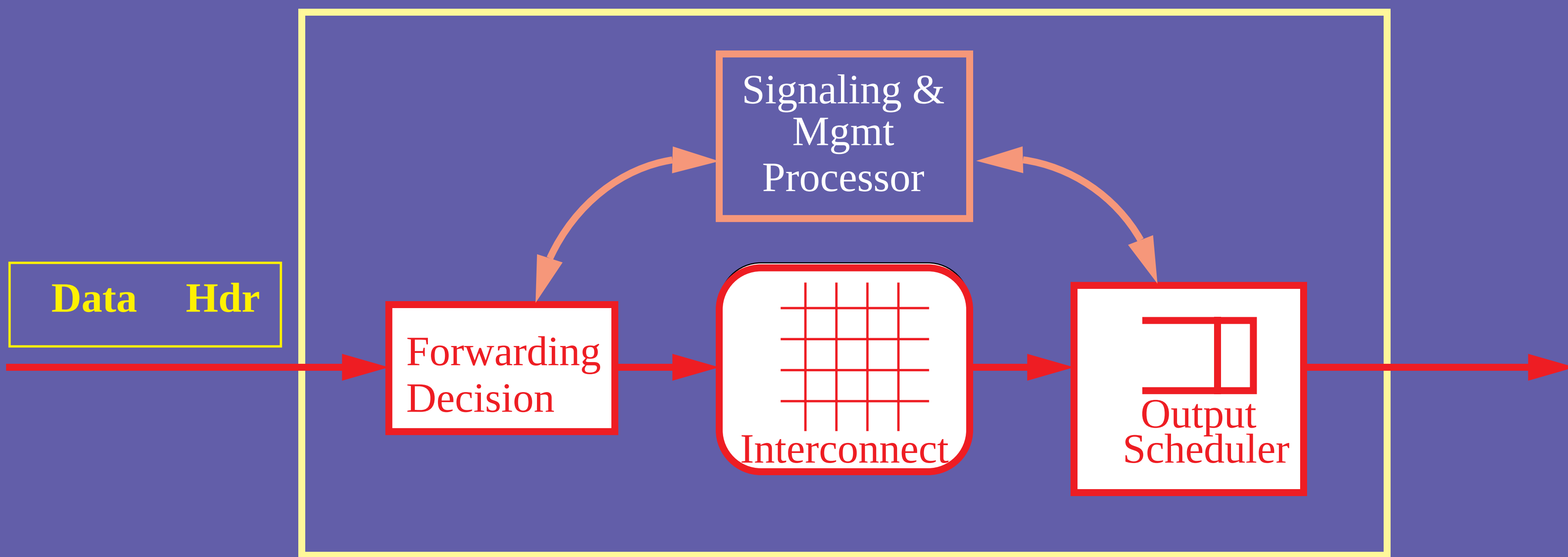
2. The Shortage of Switching/Routing Capacity

3. The Architecture of Switches and Routers

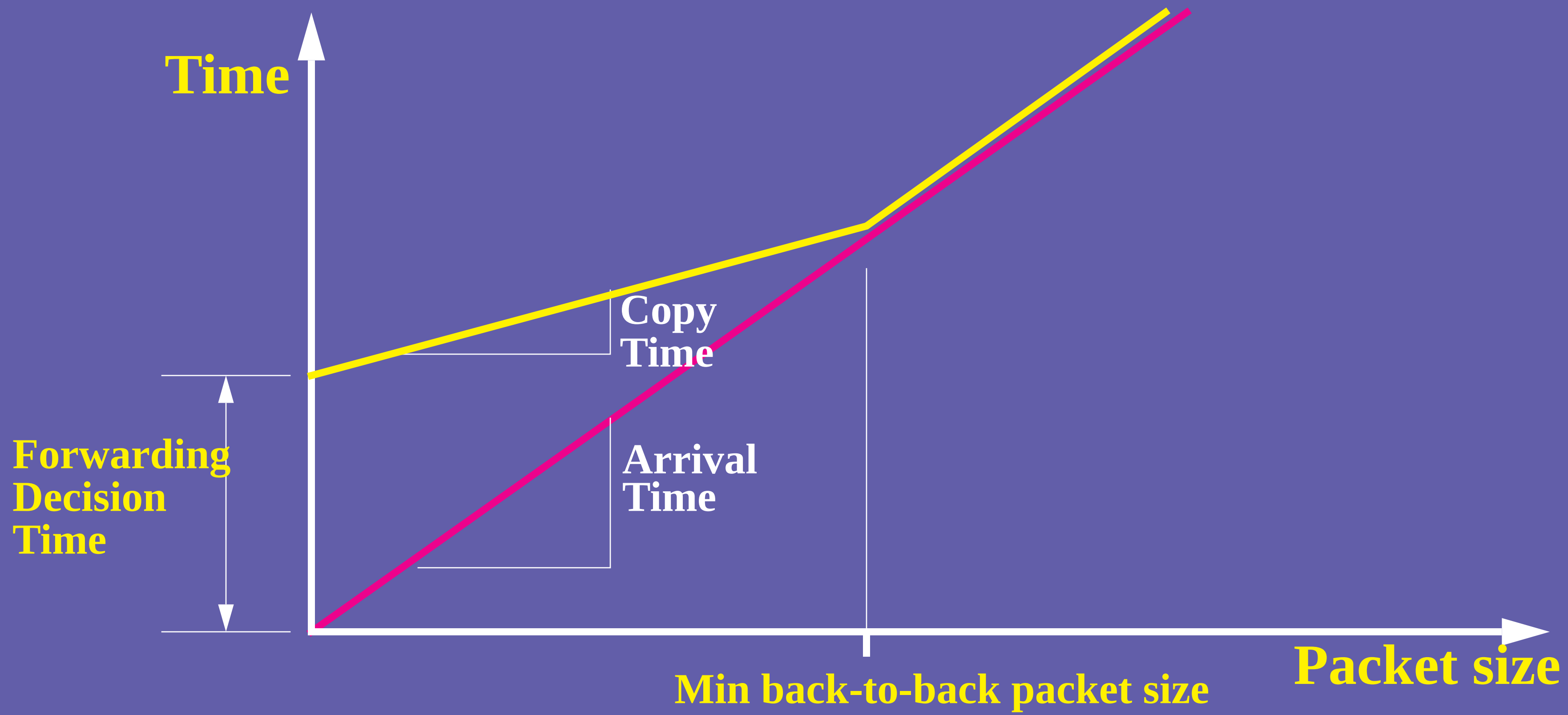
4. Some (of our) solutions

The Architecture of Switches and Routers

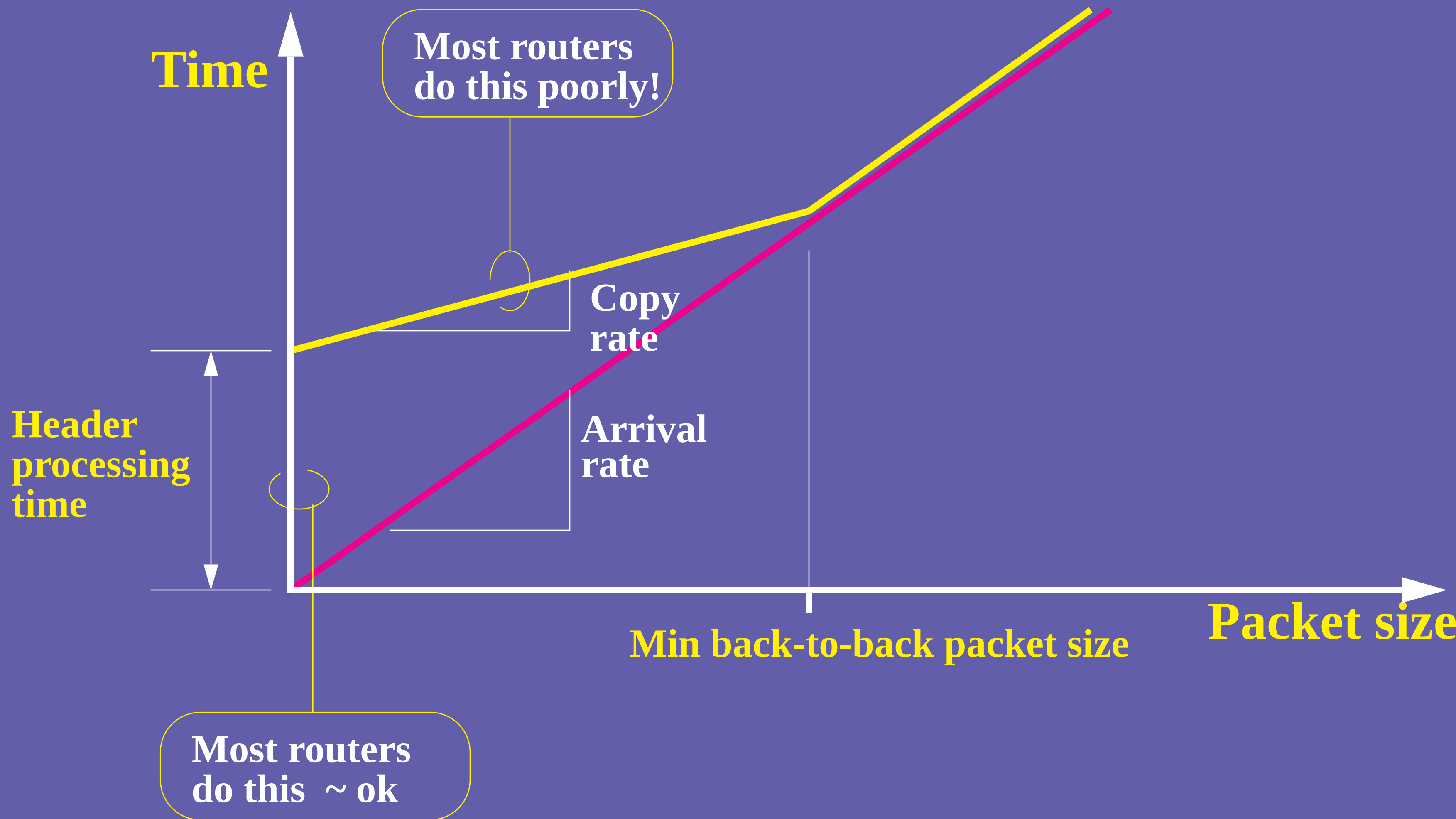
Generic Packet Switch:
(e.g. IP Router, ATM Switch, LAN Switch)



Performance of IP Routers

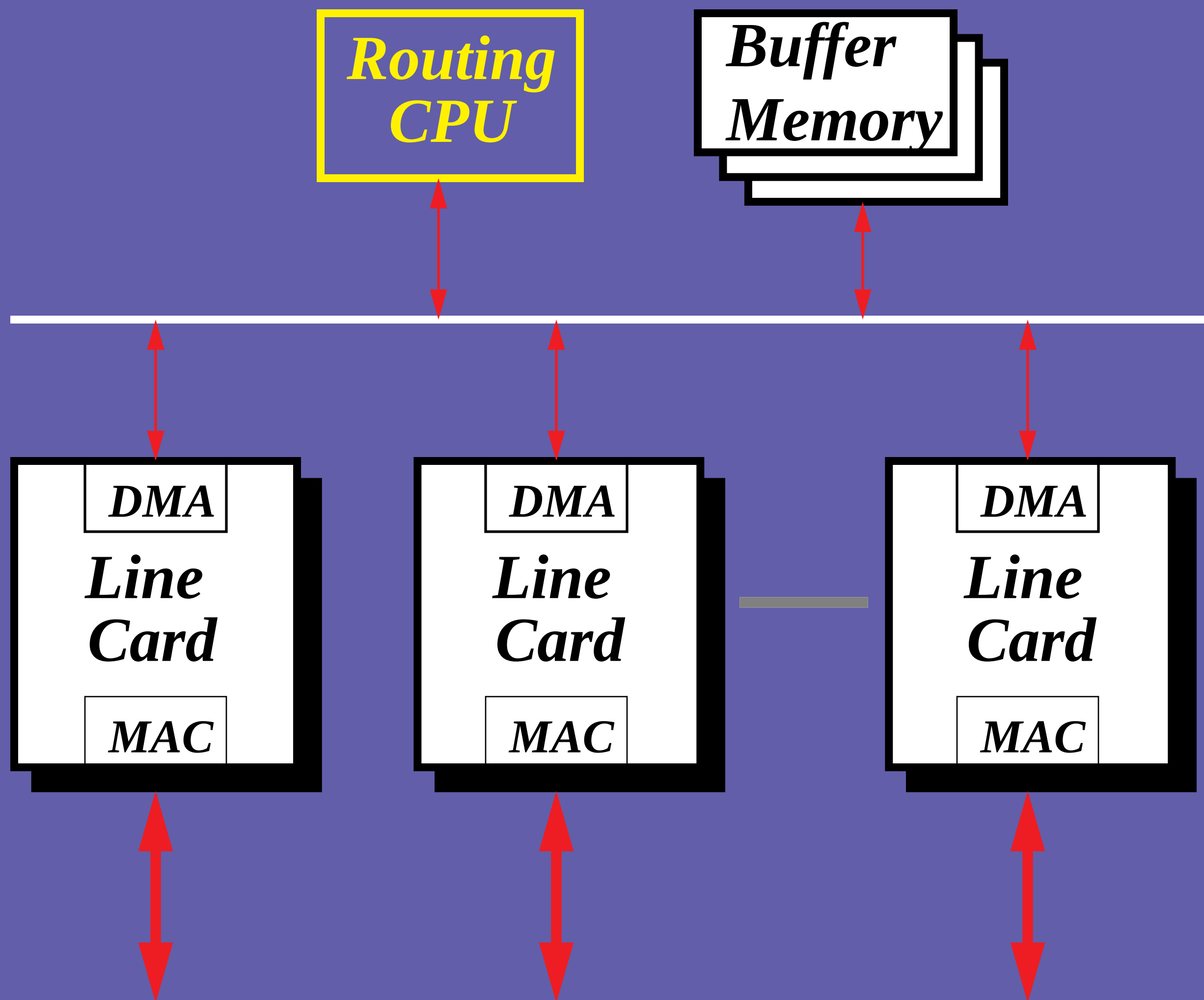


Performance of IP Routers



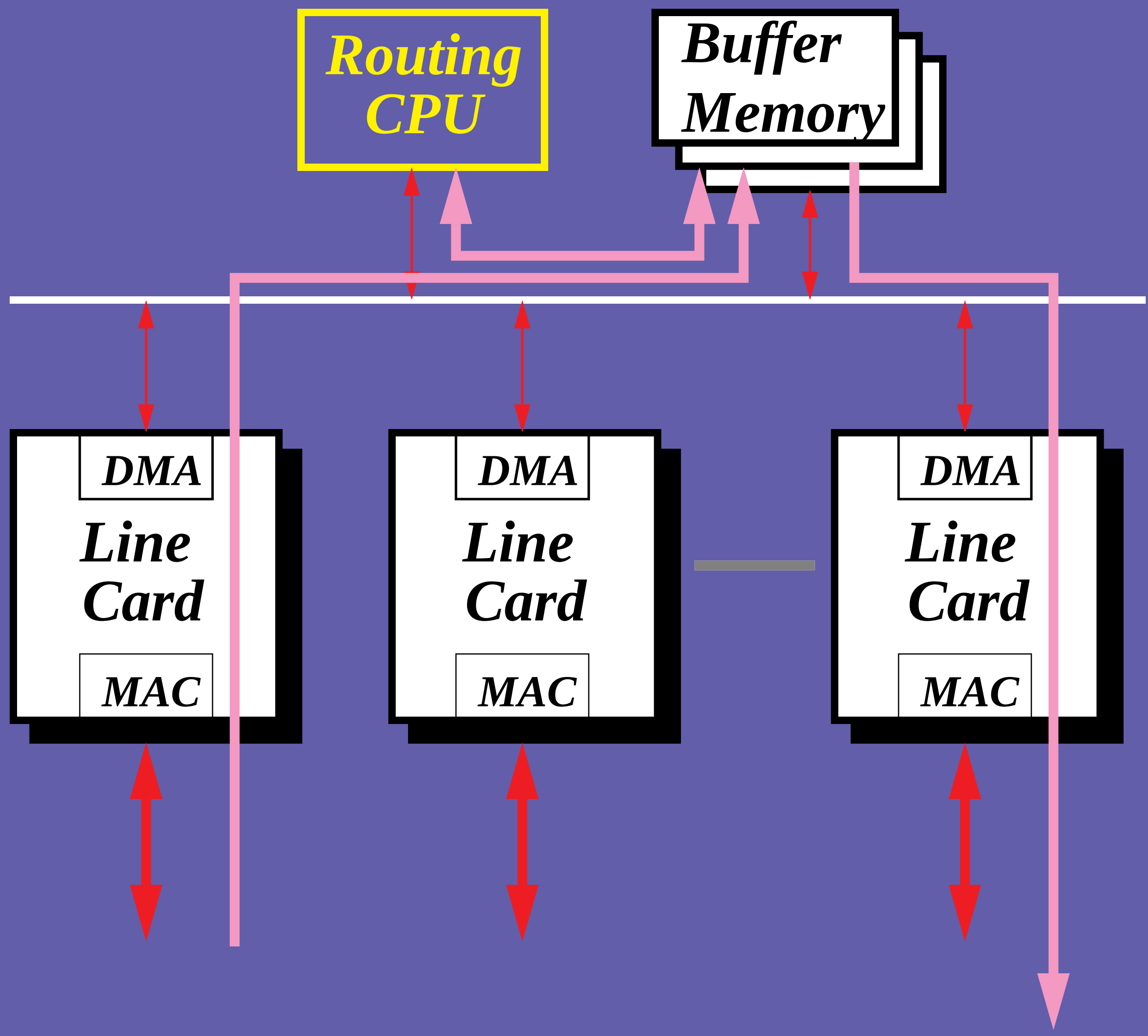
The Evolution of Routers

The first shared memory routers



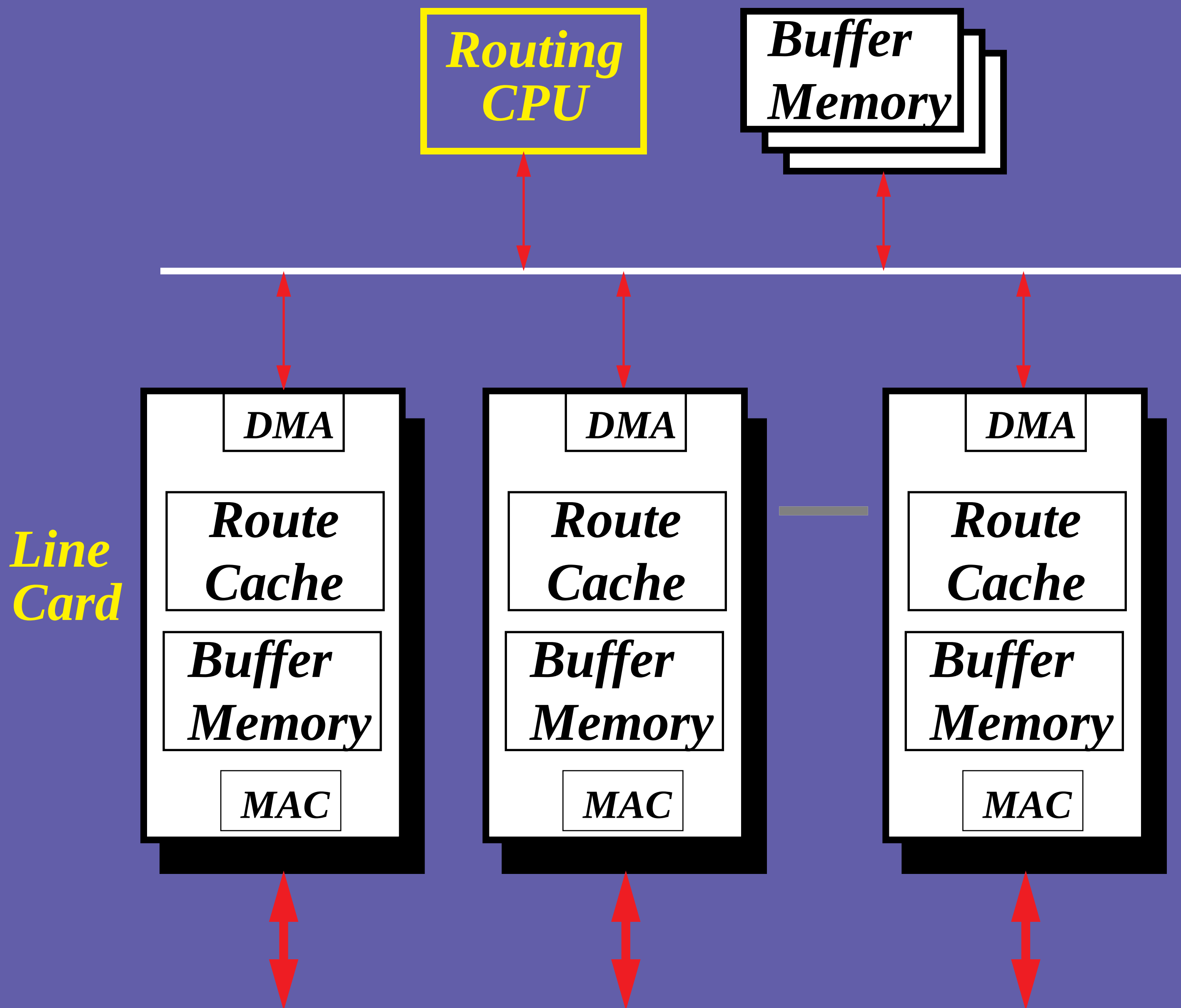
The Evolution of Routers

The first shared memory routers



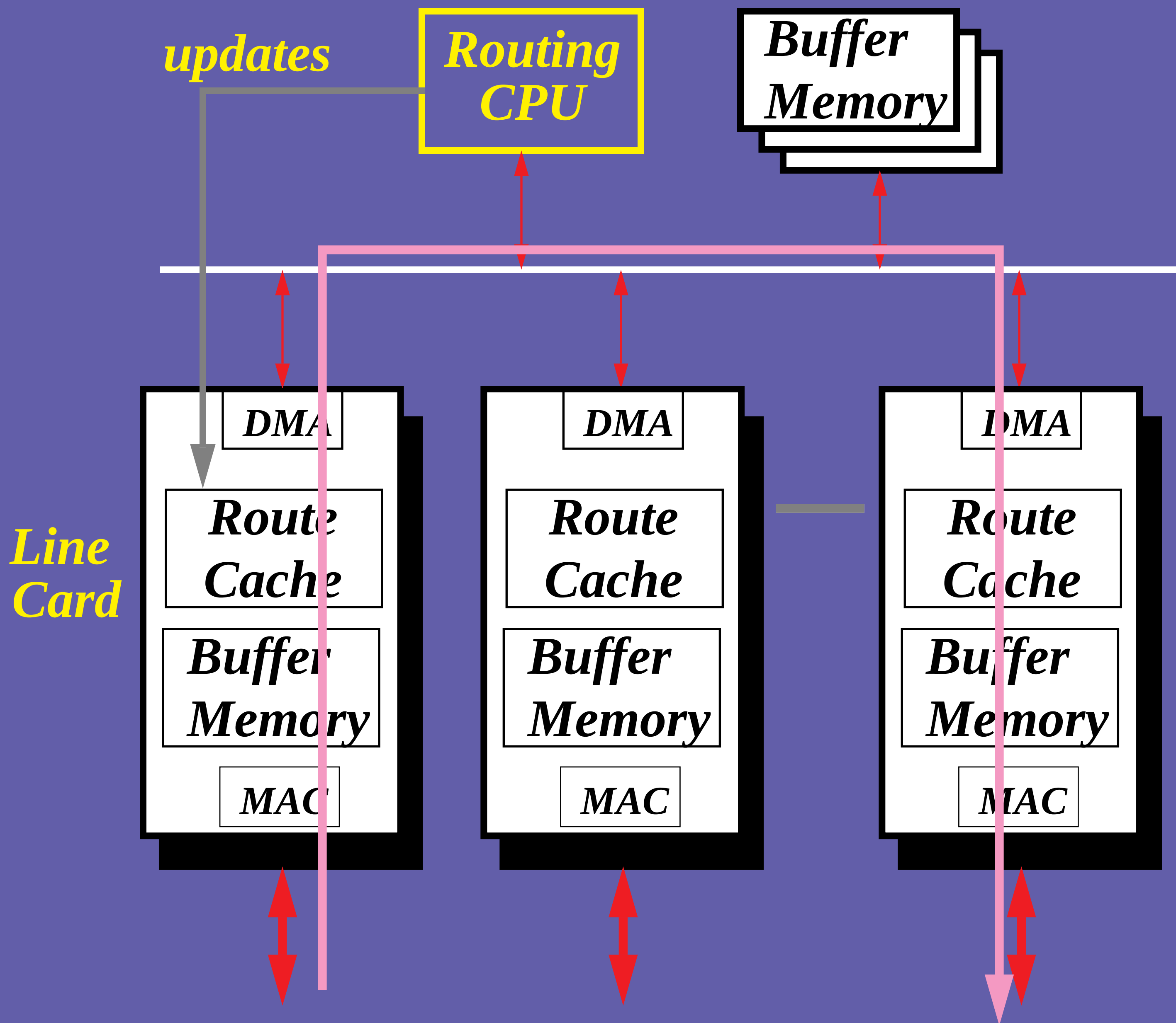
The Evolution of Routers

Reducing the number of bus copies



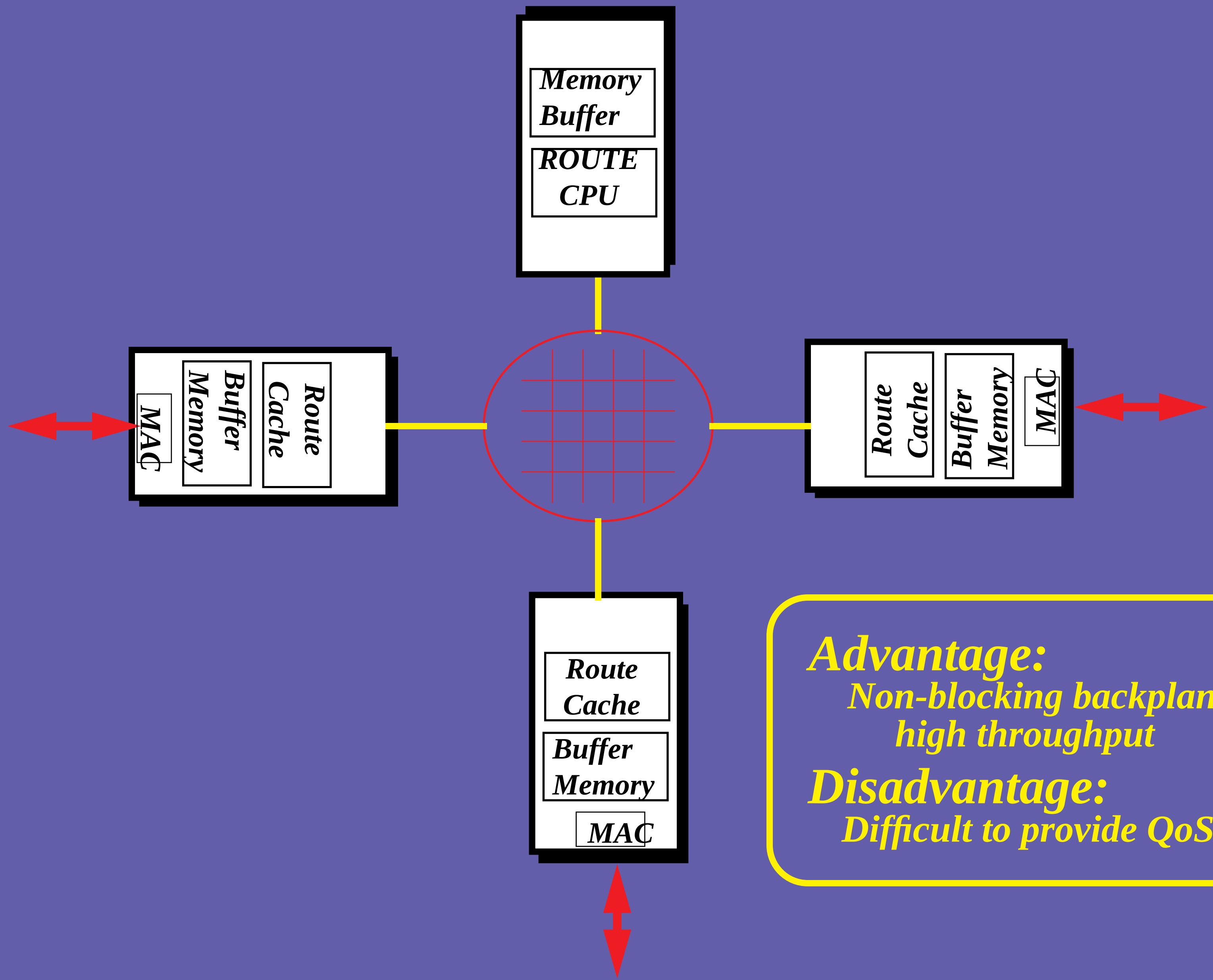
The Evolution of Routers

Reducing the number of bus copies



The Evolution of Routers

Avoiding bus contention



Advantage:

Non-blocking backplane—high throughput

Disadvantage:

Difficult to provide QoS

1. The Demand for Bandwidth

2. The Shortage of Switching/Routing Capacity

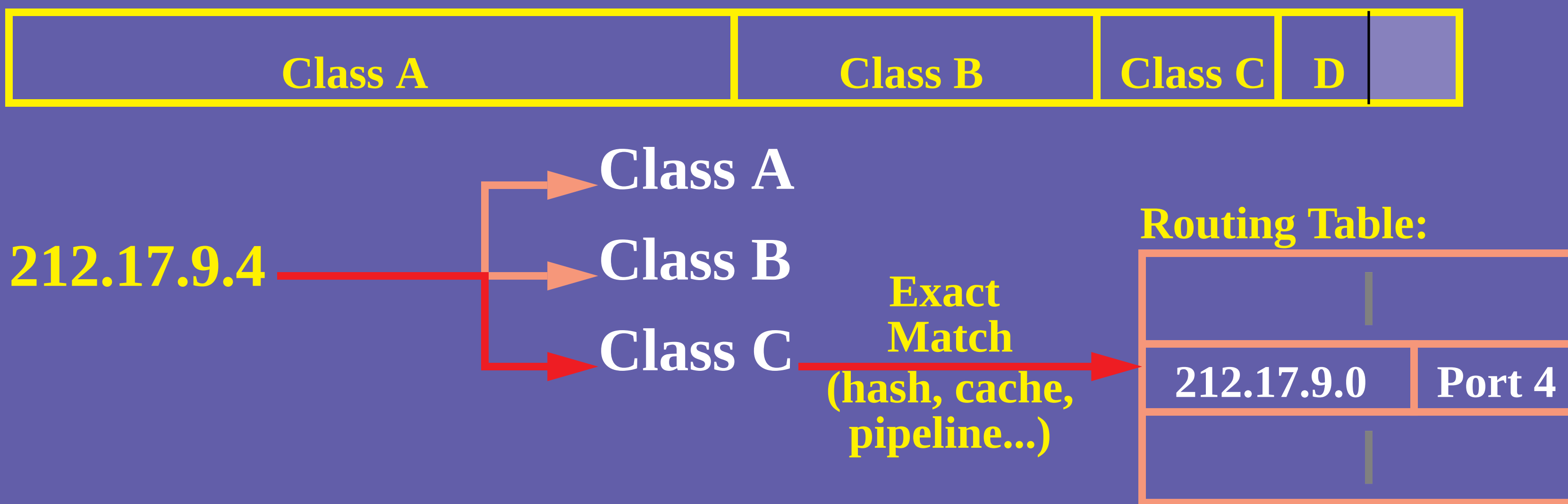
3. The Architecture of Switches and Routers

4. Some (of our) solutions

Some (of our) Solutions

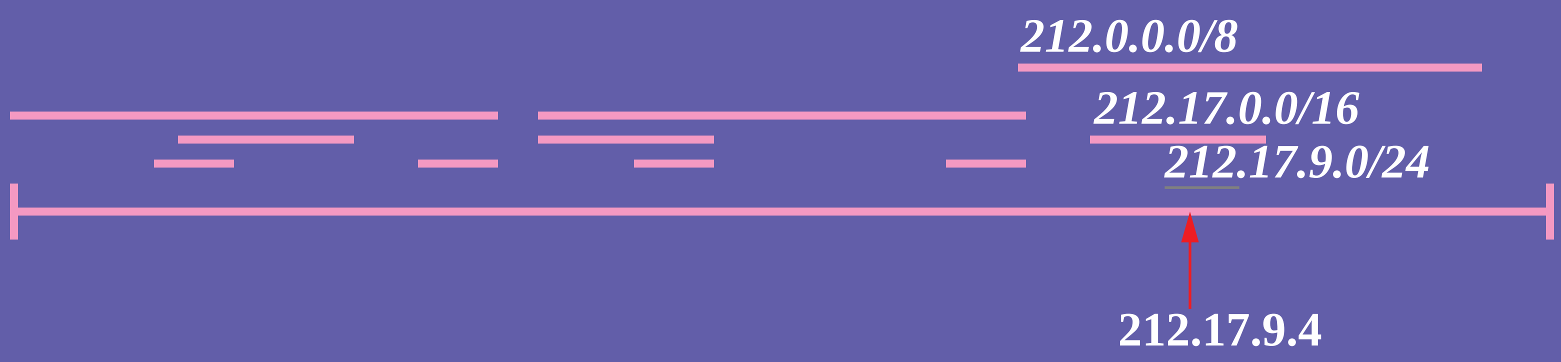
1. **Accelerating Forwardng Decisions:**
   - Longest-matching prefixes
2. **Interconnections: Switched Backplanes**
   - Input Queueing
     - Theory
     - Unicast
     - Multicast
   - Fast Buffering
   - Speedup
   - The *Tiny Tera* Project

Routing Lookups



Routing Lookups with CIDR (“supernetting”)

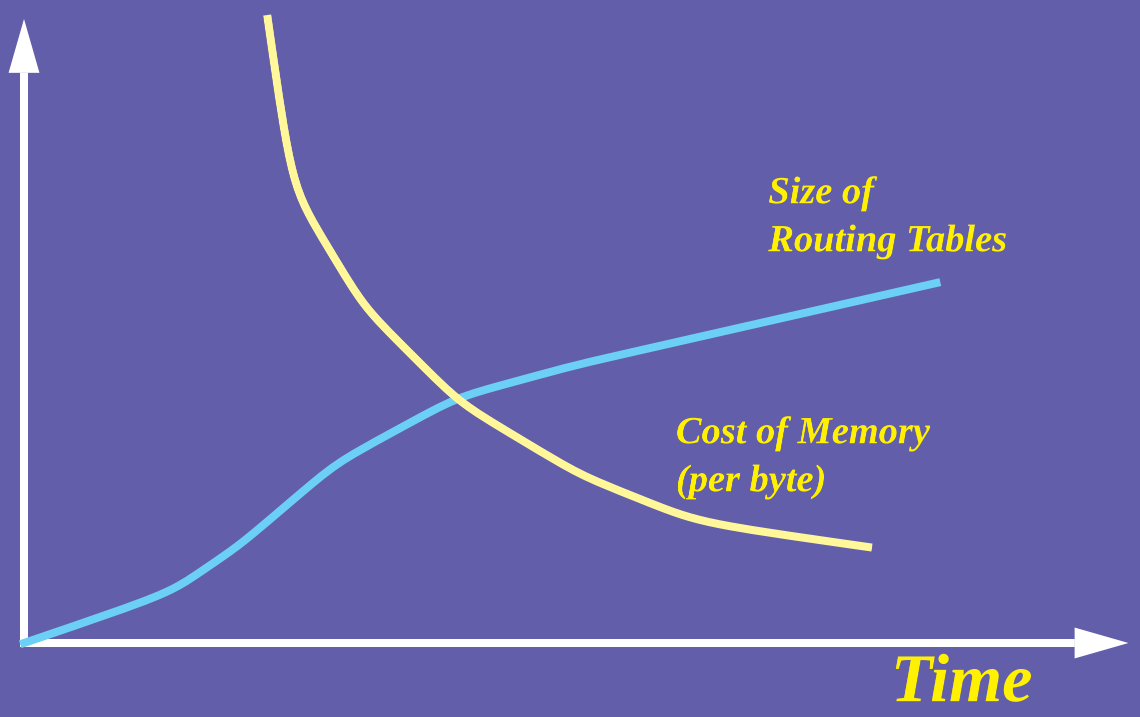
CIDR uses “longest matching prefix” routing:



Hashing, caching and pipelining are hard!

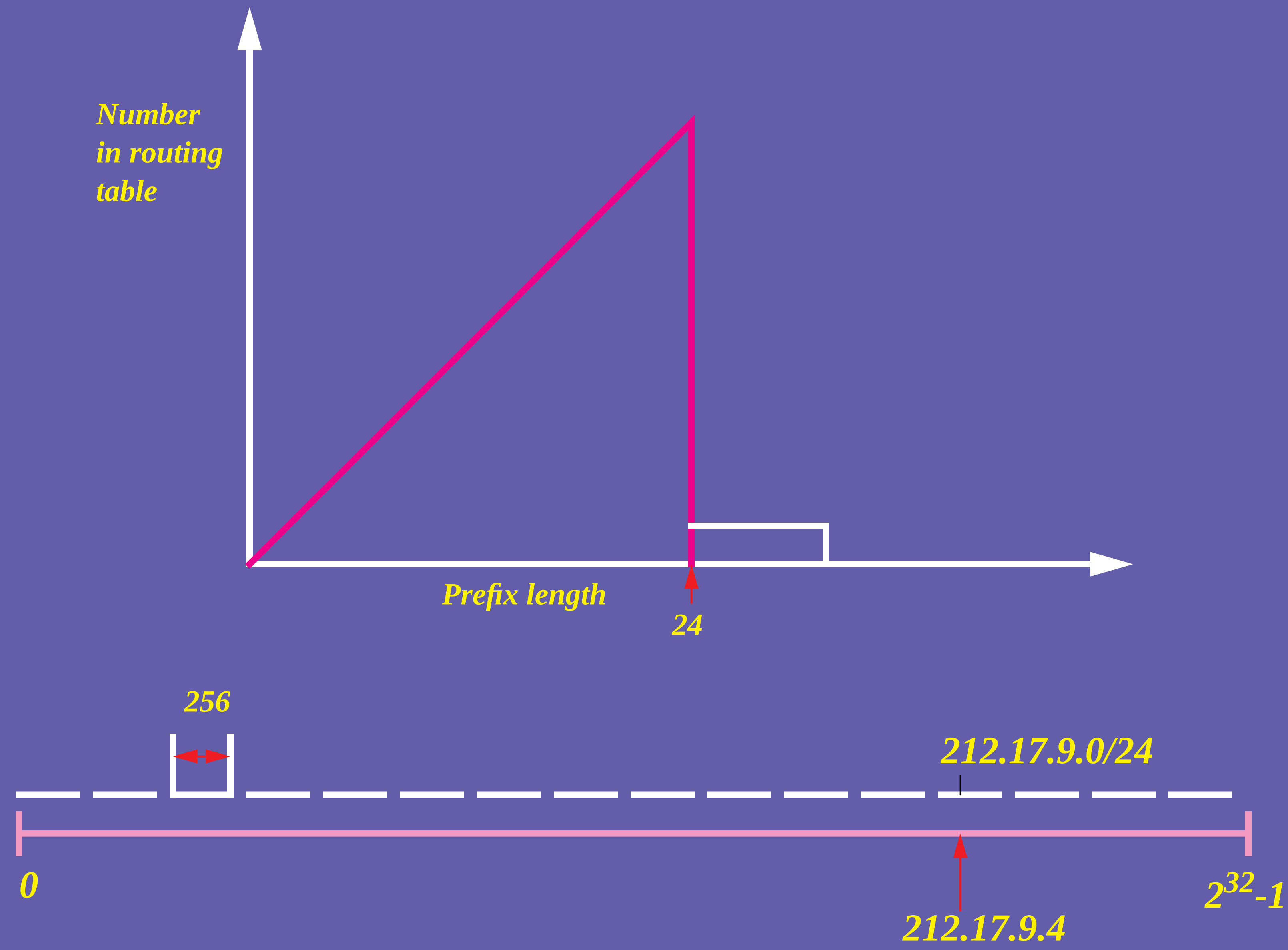
Perform Lookups Faster

Observation #1:

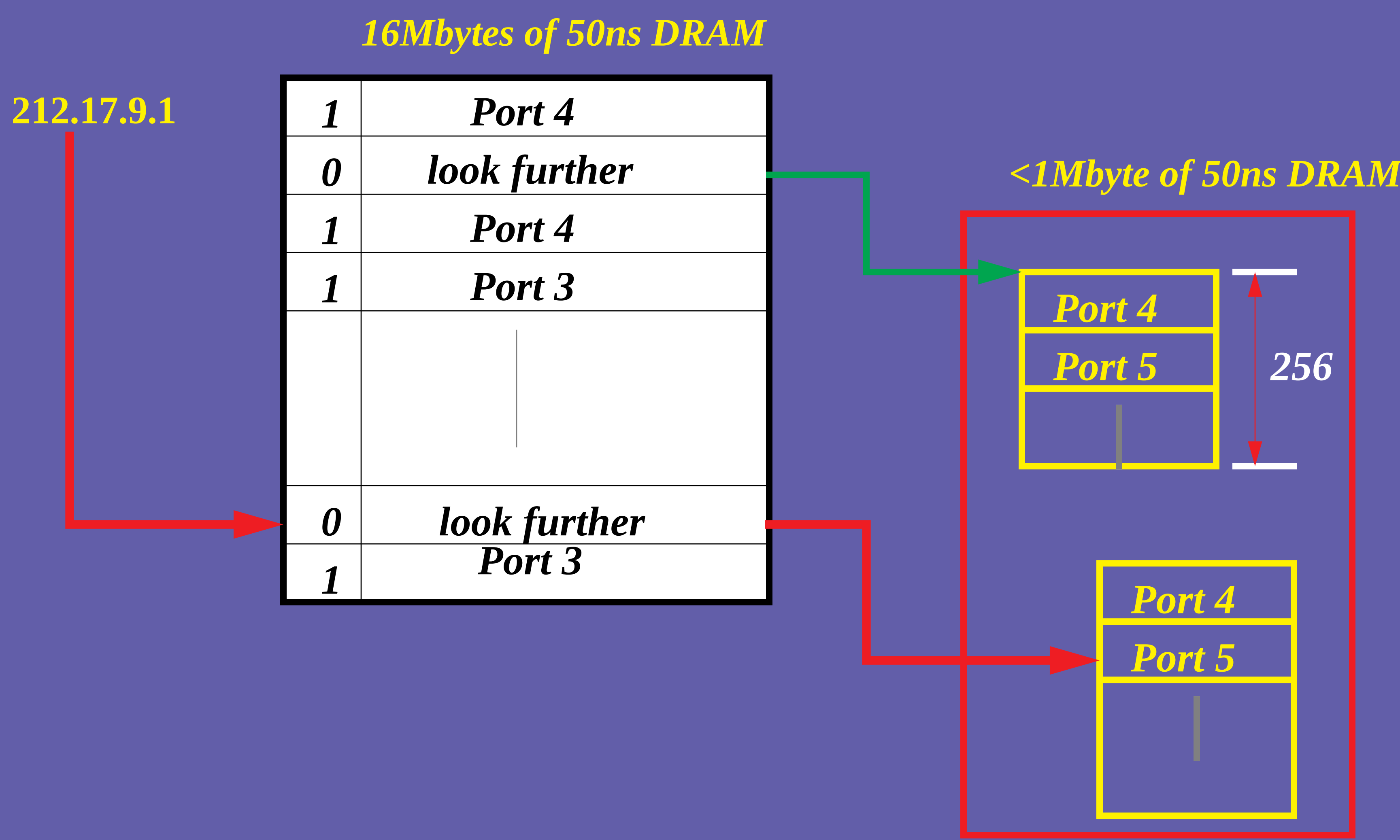


Performing Lookups Faster

Observation #2:



20 million lookups per second



1. Accelerating Forwardng Decisions:

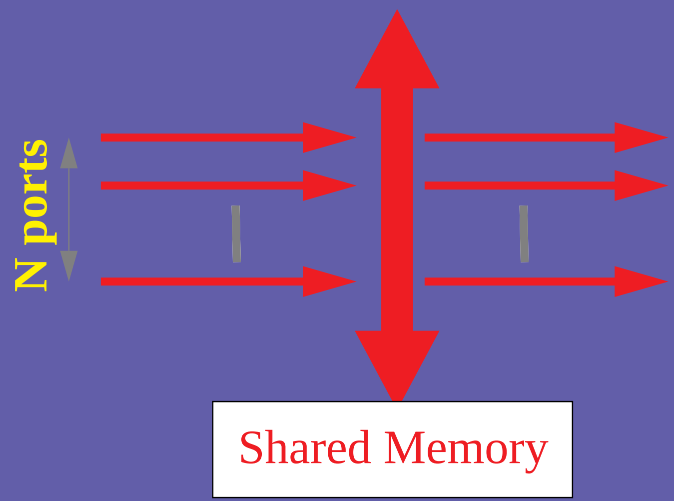
- Longest-matching prefixes

2. Interconnections: Switched Backplanes

- Input Queueing
  - Theory
  - Unicast
  - Multicast
- Fast Buffering
- Speedup
- The *Tiny Tera* Project

Should we use shared memory or input-queueing?

Shared Memory:



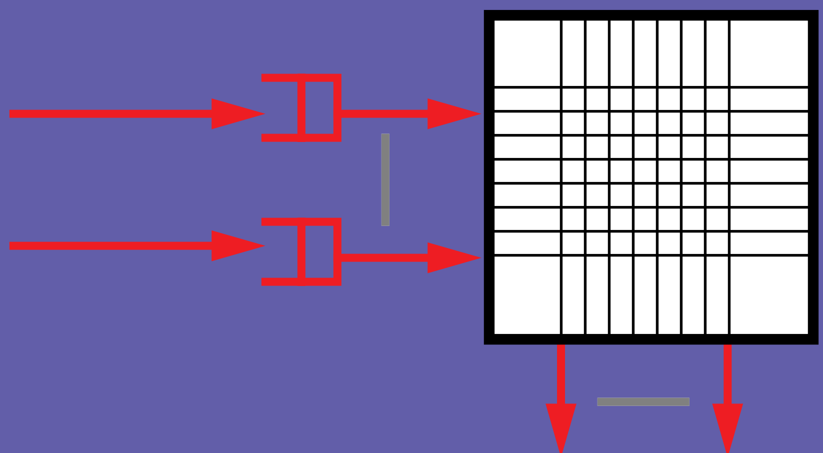
Advantages:

Highest Throughput.
Possible to control packet delay.

Disadvantages:

N-fold internal speedup

Input Queueing:



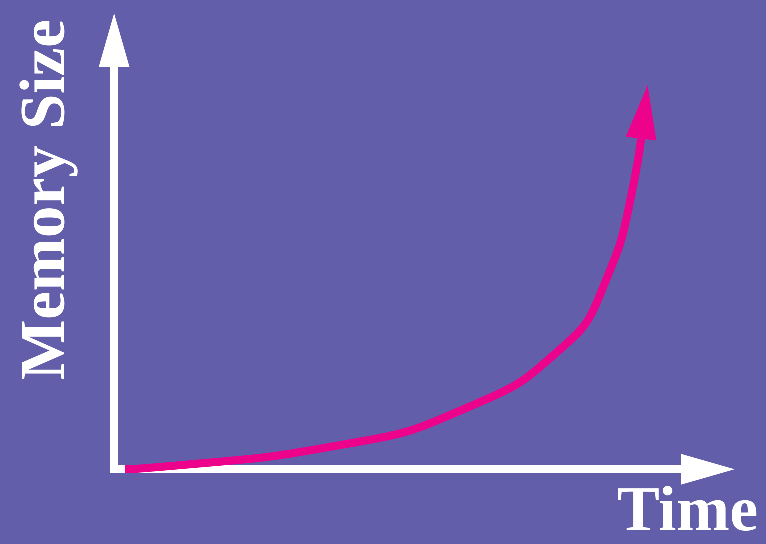
Advantages:

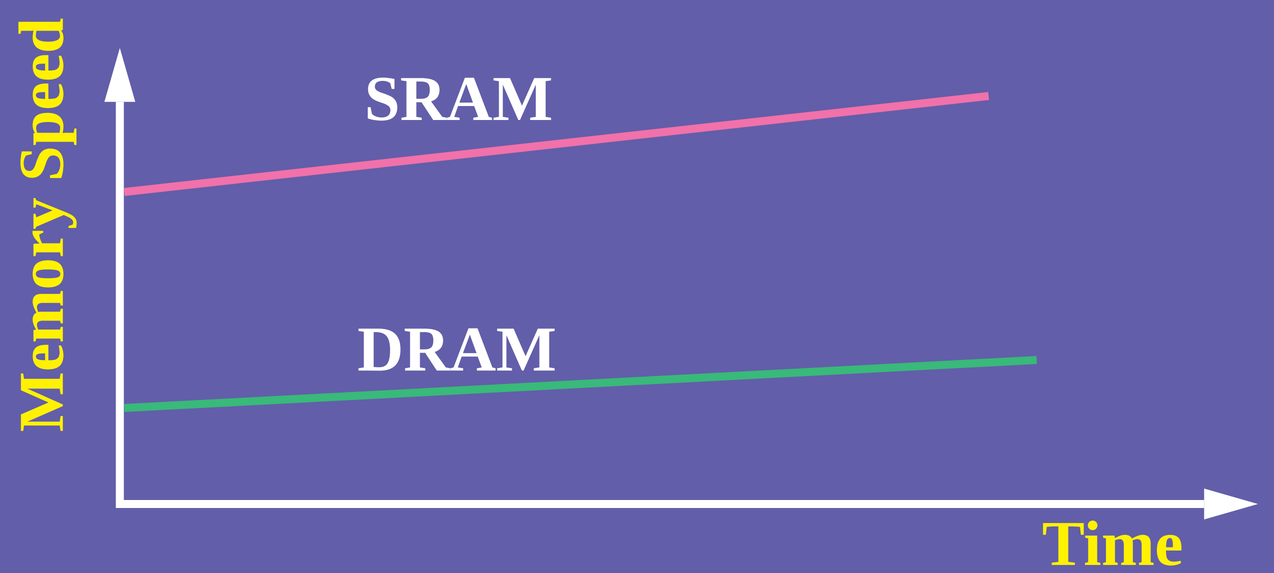
Simplicity
High Bandwidth

Disadvantages:

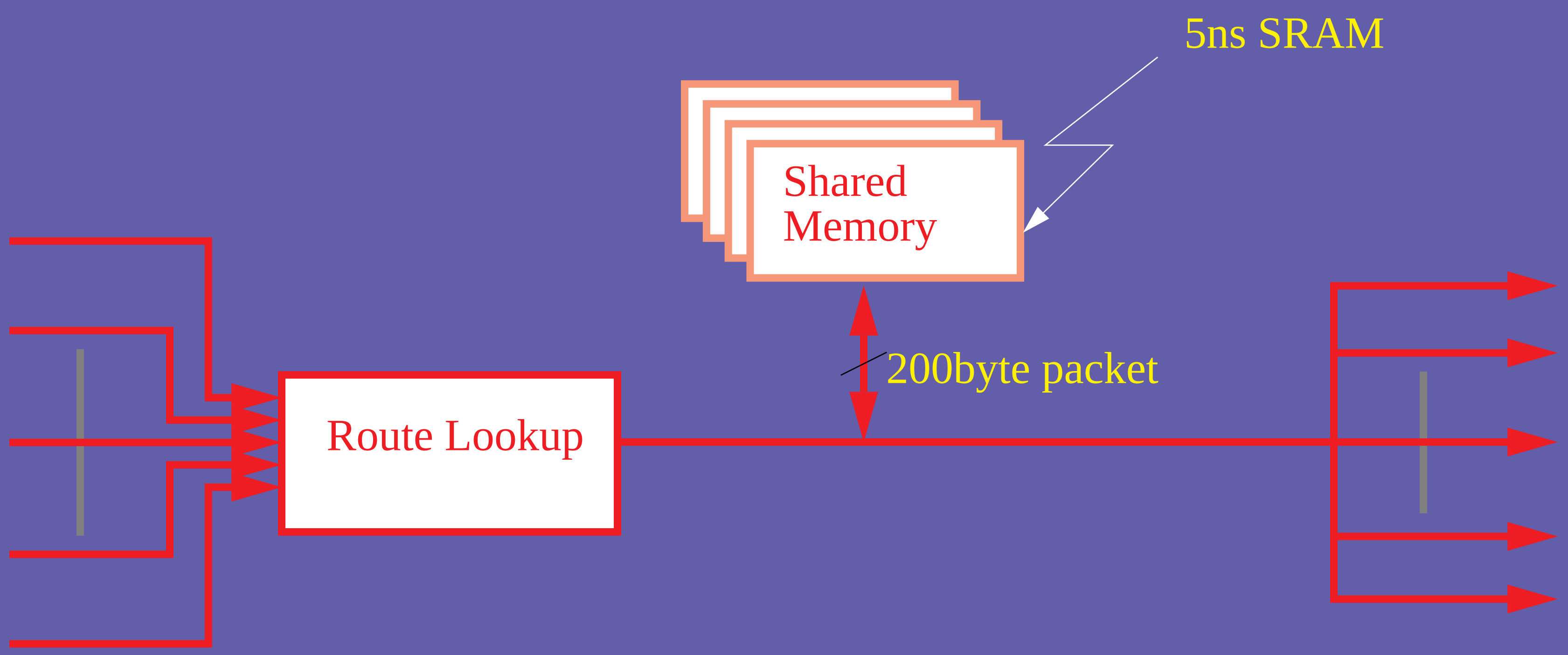
HOL Blocking
Less efficient
Difficult to control packet delay.

Memory Bandwidth





An aside: How fast can shared memory operate?



How fast can a 16 port switch run with this architecture?

$$5ns \text{ per packet} \times 2 \text{ memory operations per cell time}$$
$$\Rightarrow \text{aggregate bandwidth is } 160Gb/s$$

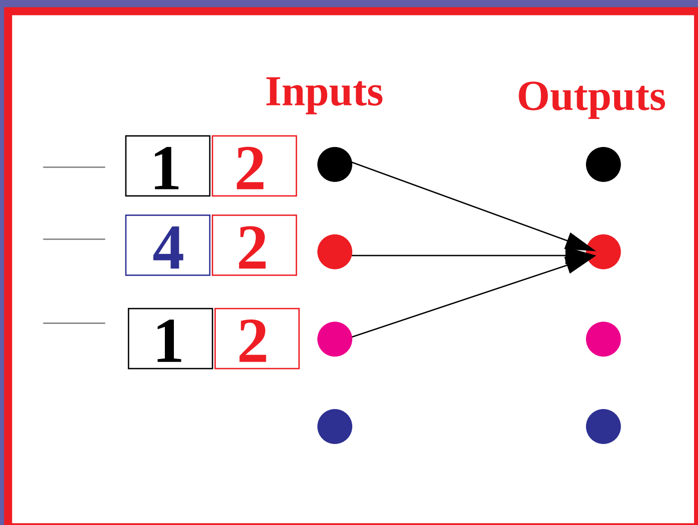
Should we use shared memory or input-queueing?

Because of a *shortage of memory bandwidth*, most multigigabit and terabit switches and routers use either:

1. Input Queueing, or
2. Combined Input and Output Queueing.

Head of Line Blocking

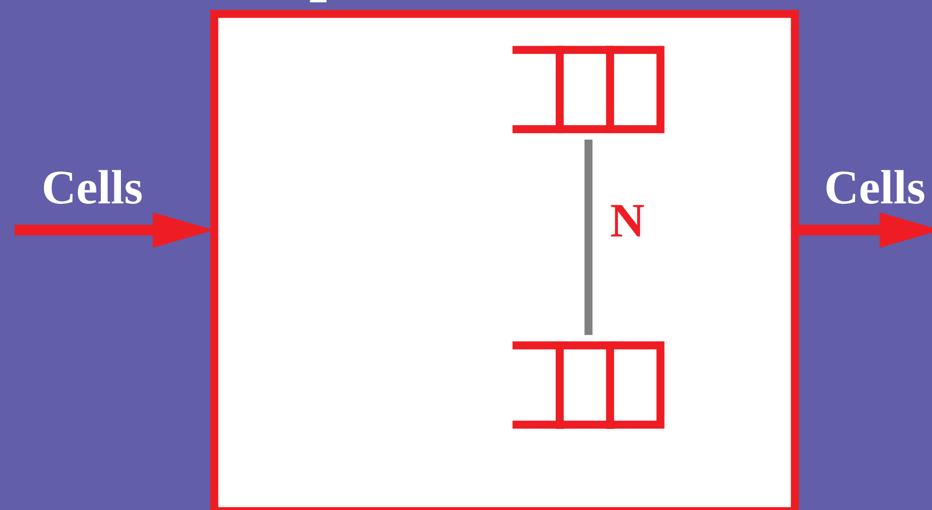
The Problem



$$\rho_{max} = 2 - \sqrt{2} = 58\%$$

A Solution....

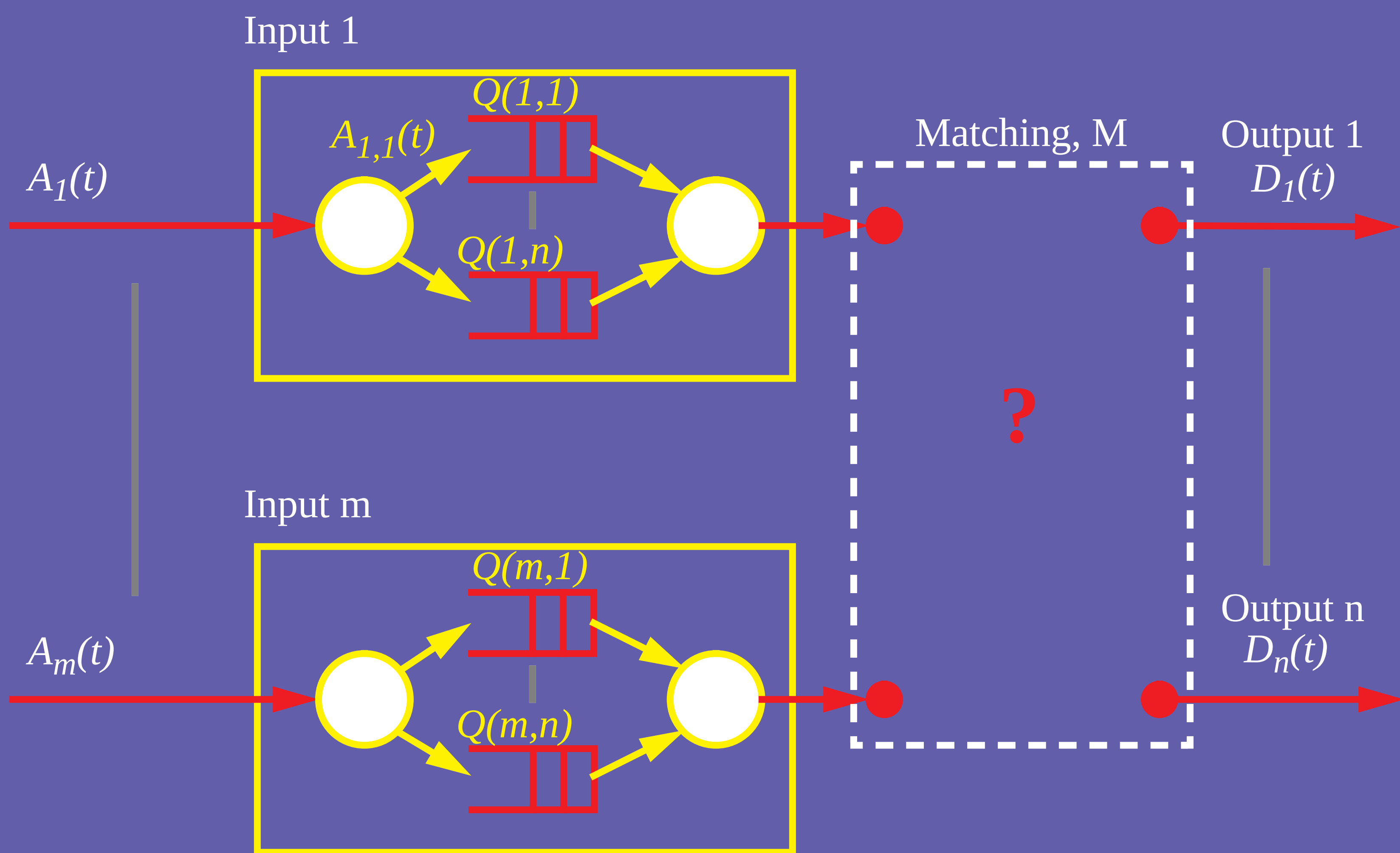
Input Cell Buffer



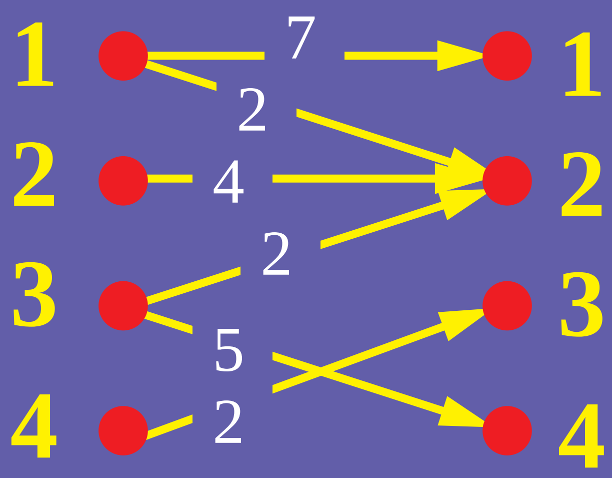
"Virtual Output Queueing"

$$\rho_{max} = 100\%$$

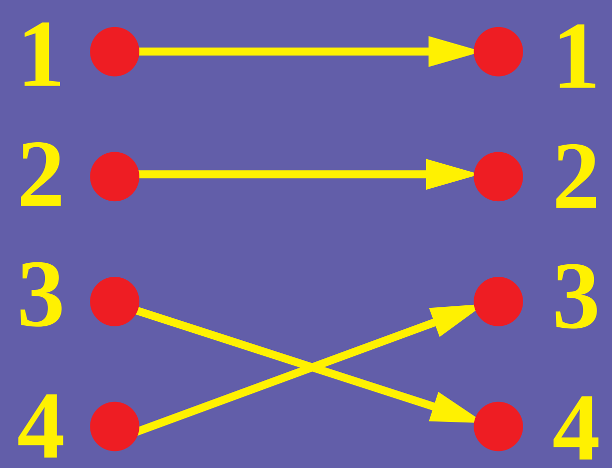
...but requires scheduling...



....which is equivalent to graph matching



Request Graph



Bipartite Matching

(Weight = 18)

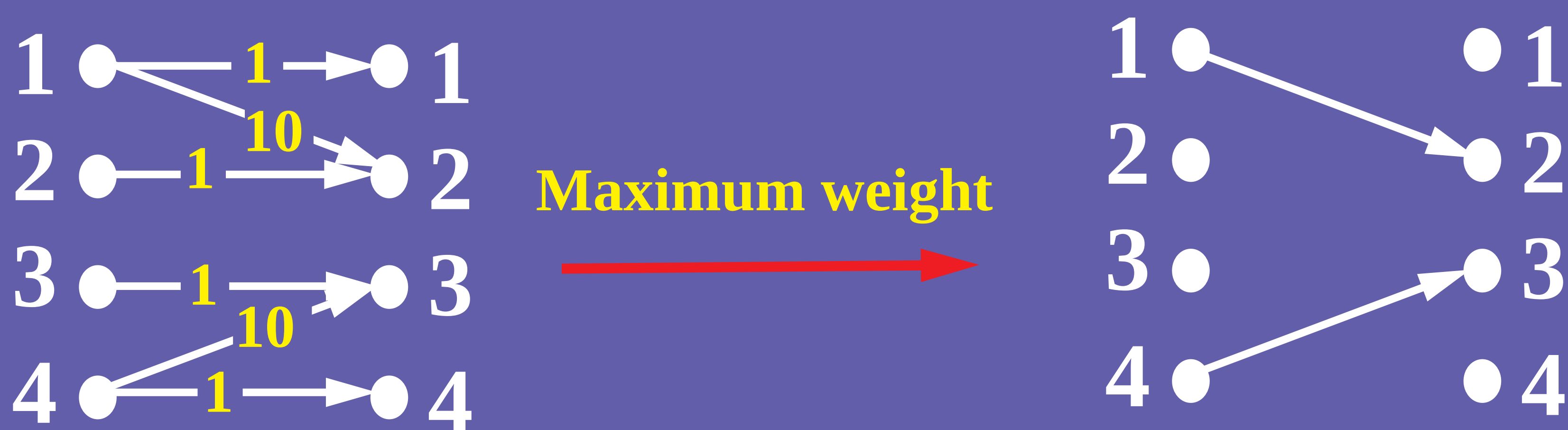
Practical Algorithms

1. ***i*SLIP**	— Weight = 1 — Iterative round-robin — Simple to implement	*Simple, fast, efficient*
2. ***i*LQF**	— Weight = Occupancy	*Good for non-uniform traffic. Complex!*
3. ***i*OCF**	— Weight = Cell Age	
4. **LPF**	— Weight = Backlog	*Good for non-uniform traffic. Simple.*

Achieving 100% Throughput

Longest Queue First & Oldest Cell First

Weight = { **Queue Length**, **Waiting Time** } ⇒ **100%**



Theorems

Theorem:

Both LQF and OCF can achieve 100% throughput for independent traffic both uniform and non-uniform.

Proof:

$$E\left[\sum_{i,j} L_{i,j}(n)\right] < \infty, \forall n \quad \overset{\text{Def}}{\Rightarrow} \quad \text{100\% throughput}$$

$$\Uparrow$$

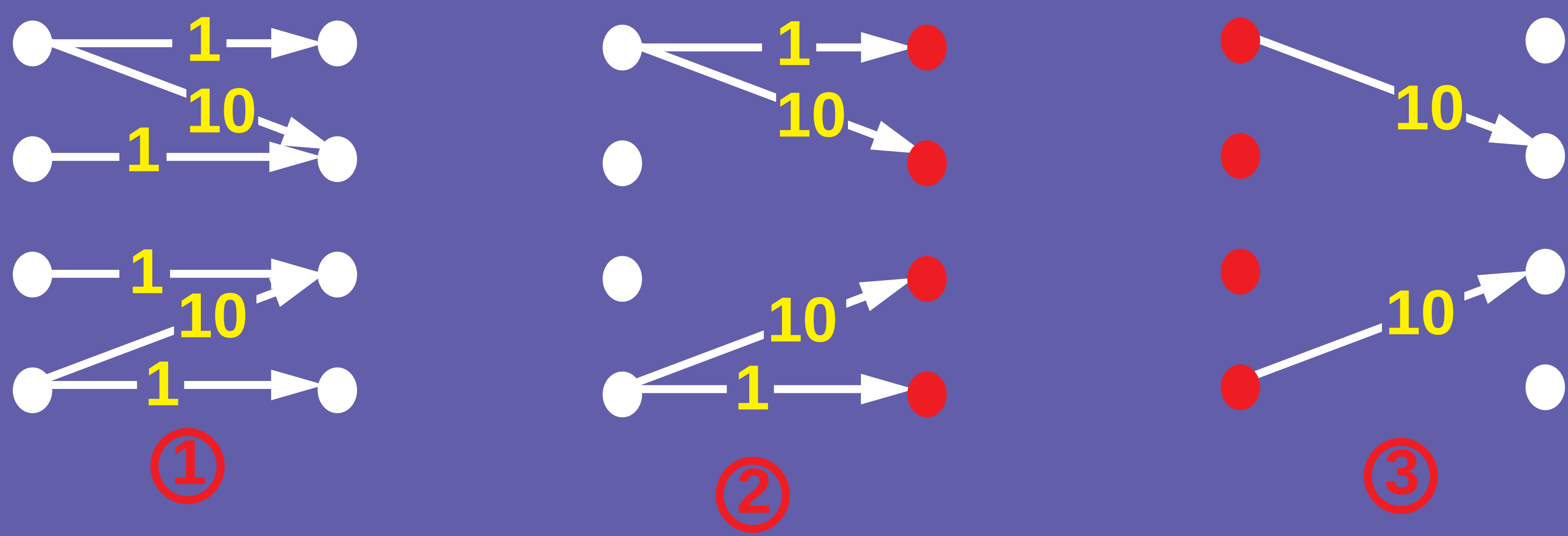
Lyapunov Stability Criterion:

$$\mathrm{E}\,[V(\underline{L}(n+1)) - V(\underline{L}(n)) \mid \underline{L}(n)] \le 0, \forall |\underline{L}(n)| > k$$

http://tiny-tera.stanford.edu/~adisak/research.html

Approximating LQF and OCF

iLQF&iOCF



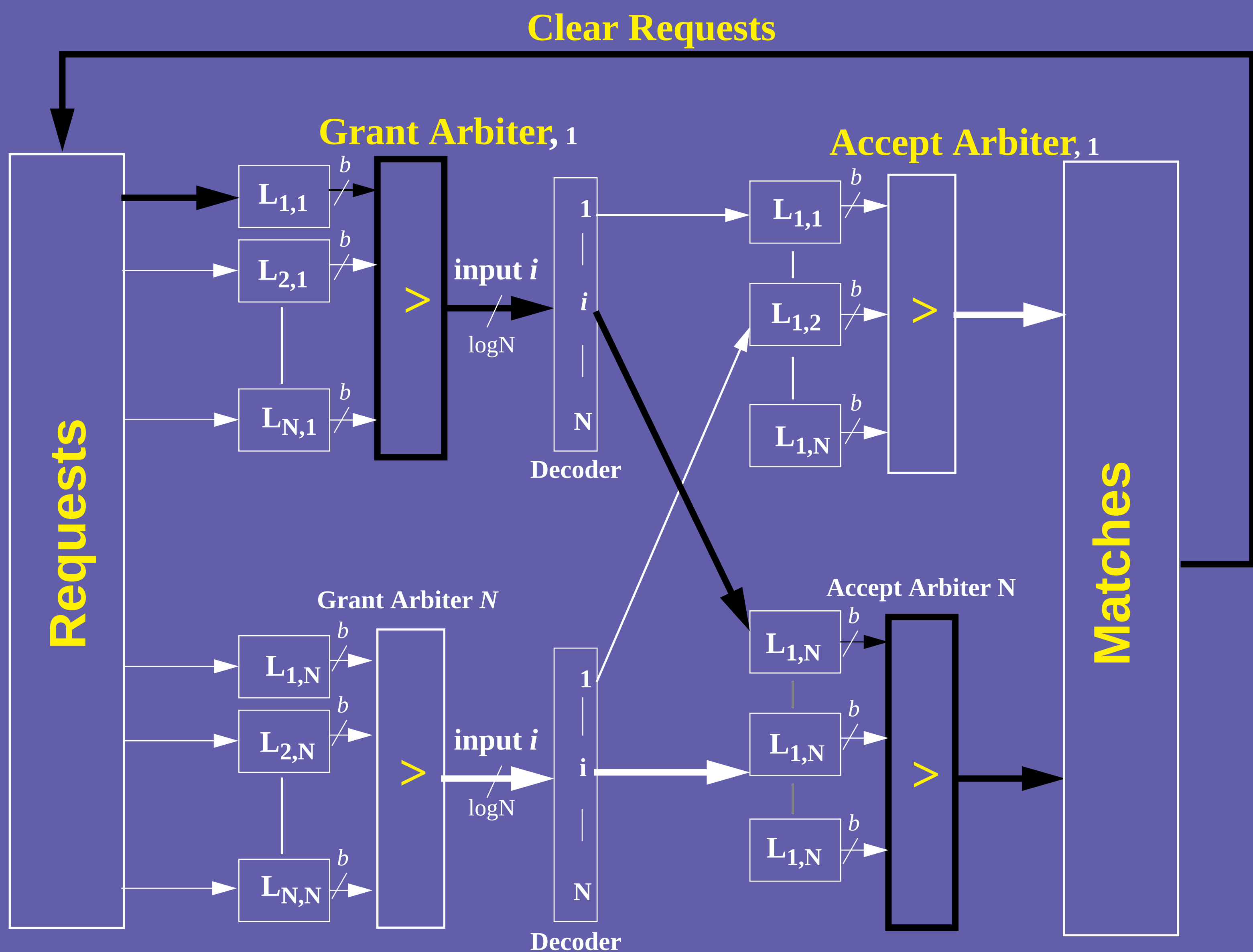
Iteration steps

Step 1. ***Request***

Step 2. ***Grant to the largest request***

Step 3. ***Accept grant to the largest request***

*i*LQF and *i*OCF
Problem is in Comparators



Solution to *Complexity Problem*

- Longest Port First (LPF)
- Oldest Port First (OPF)

Advantages

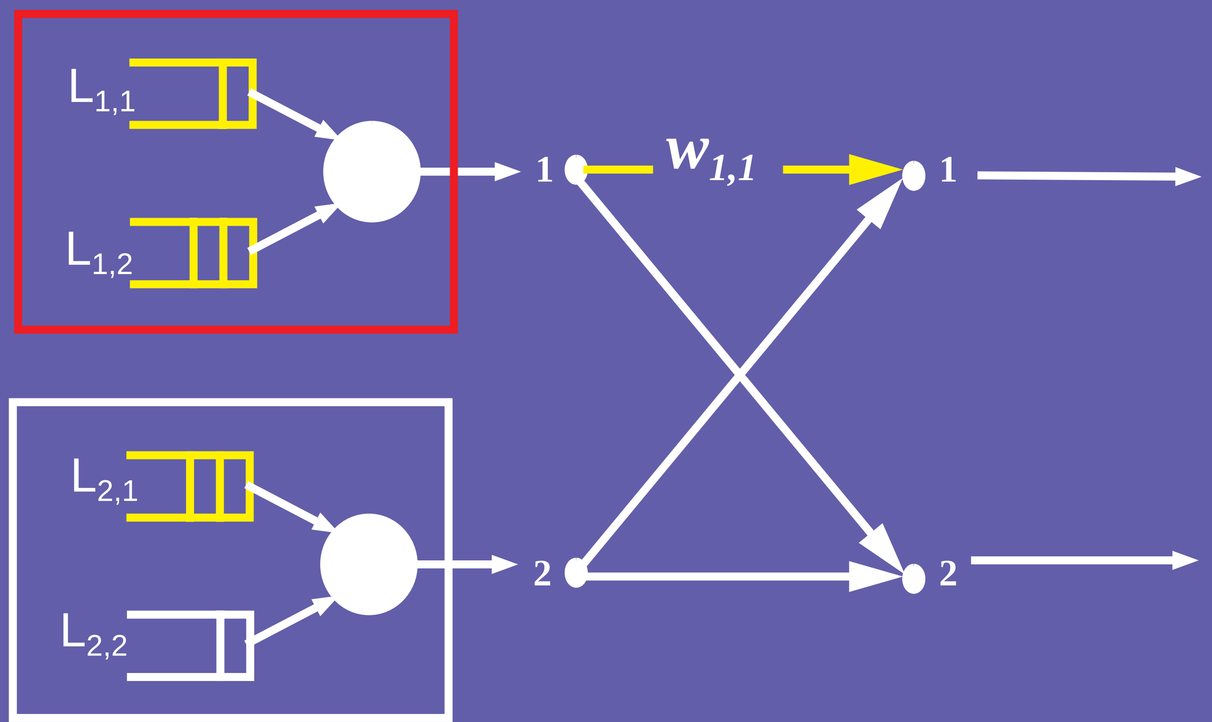
— SIMPLER.

- Can use maximum size matching — $O(N^{2.5})$.

— FASTER.

- Move magnitude comparator out of the critical path.
- Lends itself well to pipelining.

LPF Algorithm
Using Port Occupancy



$$w_{i,j} = \sum_j L_{i,j} + \sum_i L_{i,j}$$

i.e. $w_{1,1} = \underbrace{L_{1,1} + L_{1,2}}_{\text{Input occupancy}} + \underbrace{L_{1,1} + L_{2,1}}_{\text{Output occupancy}}$

Input occupancy Output occupancy

On The Theorems

Theorem:

An LPF match is of both maximum weight and maximum size.

Theorem:

LPF can achieve 100% throughput for independent traffic both uniform and non-uniform.

Proof: $V(\underline{L}(n)) = \underline{L}^T(n) T \underline{L}(n)$

$$\mathrm{E}\,[V(\underline{L}(n+1)) - V(\underline{L}(n)) \mid \underline{L}(n)] \le 0, \forall |\underline{L}(n)| > k$$

Presorting Inputs & Outputs

2	20	0
17	0	8
0	0	1

Weight request →

	19	20	9
22	1 41	1 42	0 31
25	1 44	0 45	1 34
1	0 20	0 21	1 10

	19	20	9
22	1 41	1 42	0 31
25	1 44	0 45	1 34
1	0 20	0 21	1 10

Permute →

	20	19	9
25	0 45	1 44	1 34
22	1 42	1 41	0 31
1	0 21	0 20	1 10

	20	19	9
25	45 (0)	44 (1)	34 (1)
22	42 (1)	41 (1)	31 (0)
1	21 (0)	20 (0)	10 (1)

Remove Weights →

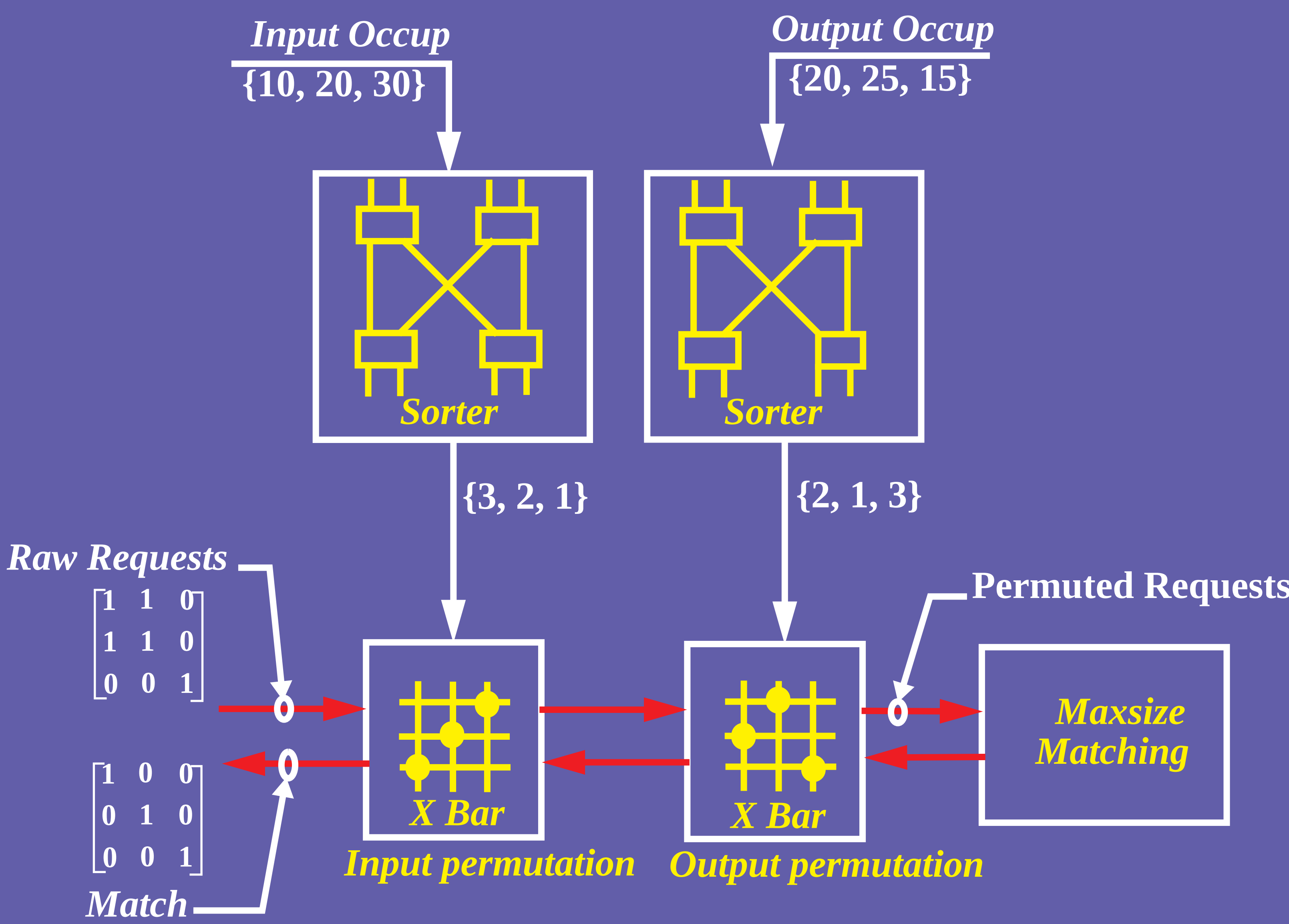
0	1	1
1	1	0
0	0	1

0	1	1
1	1	0
0	0	1

Matching →

0	1	0
1	0	0
0	0	1

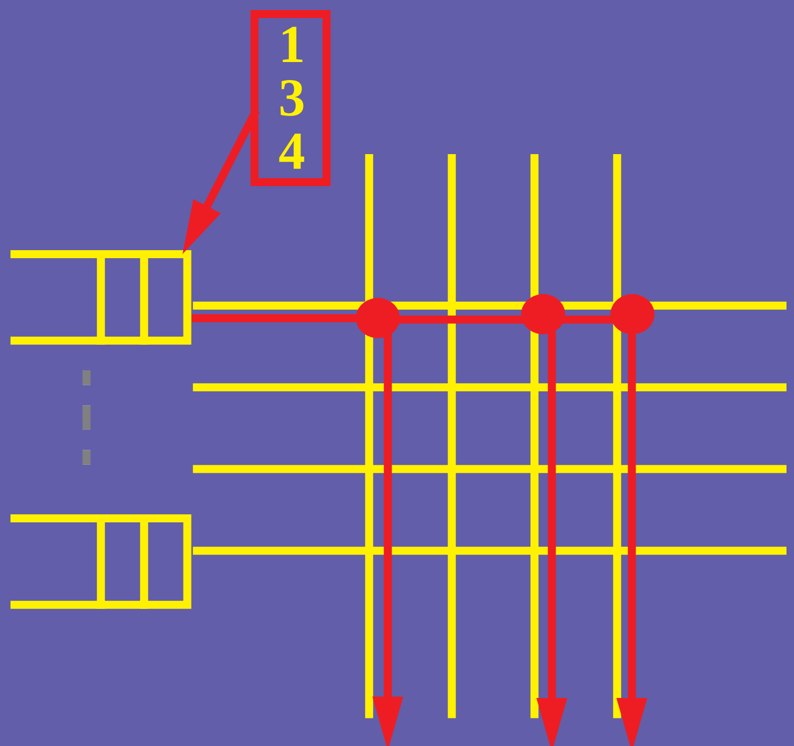
Implementation



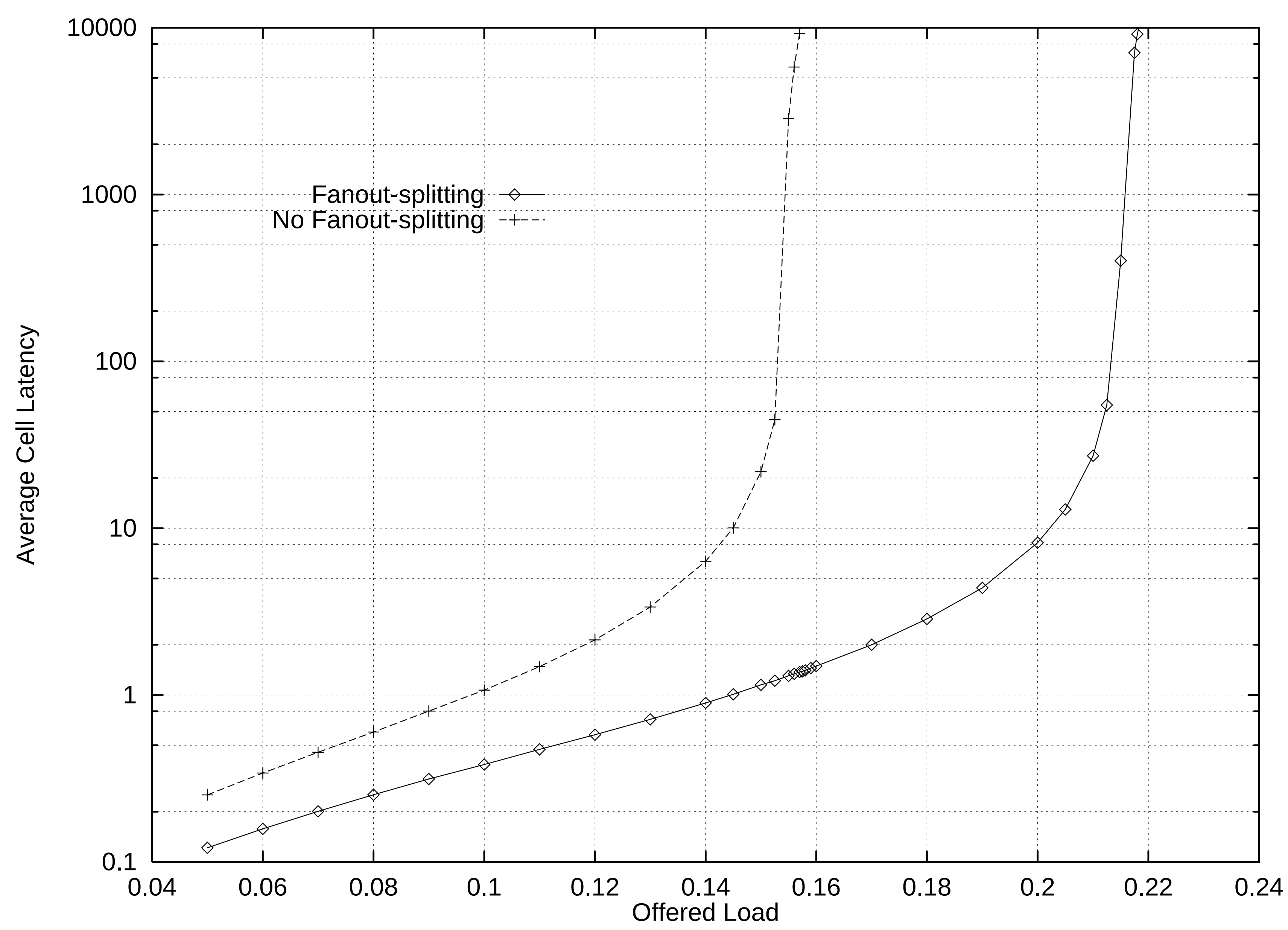
Multicast Traffic

Queue Architecture

1. Making use of the crossbar
2. Why treat multicast differently?
3. Why maintain a single FIFO queue?
4. Fanout-splitting



Fanout-Splitting



Multicast Traffic

1. ***Residue Concentration***

2. ***Tetris-based schedulers***

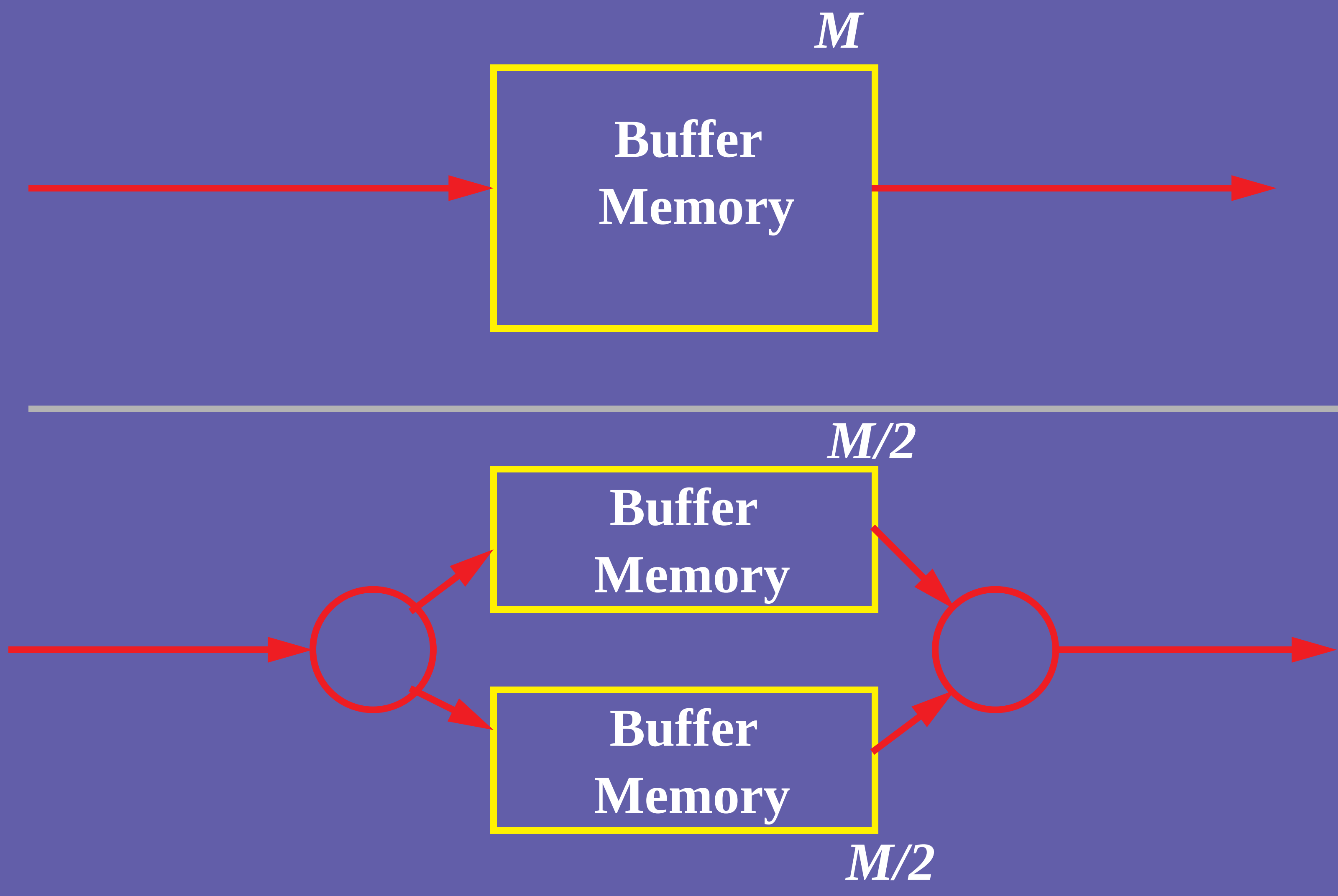
1. Accelerating Forwardng Decisions:

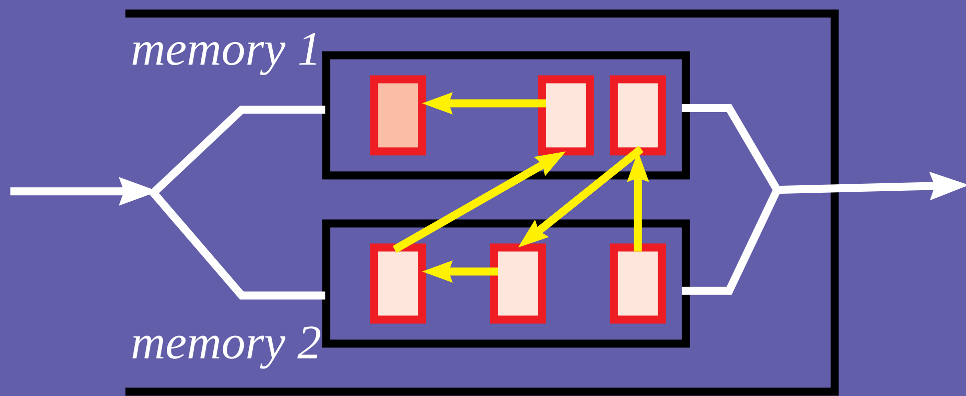
- Longest-matching prefixes

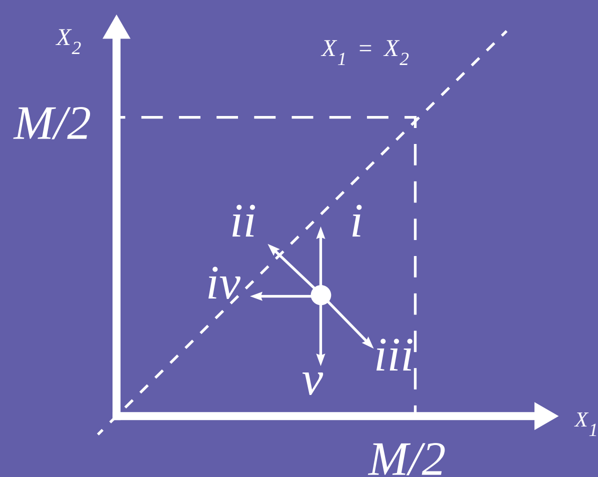
2. Interconnections: Switched Backplanes

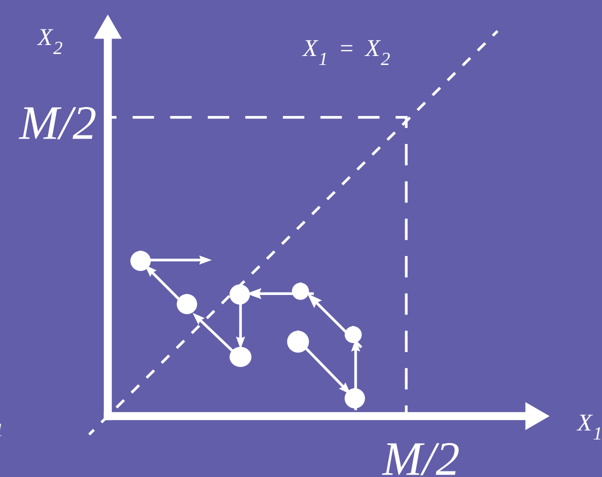
- Input Queueing
  - Theory
  - Unicast
  - Multicast
- Fast Buffering
- Speedup
- The *Tiny Tera* Project

Fast Buffering
Ping-pong Memory

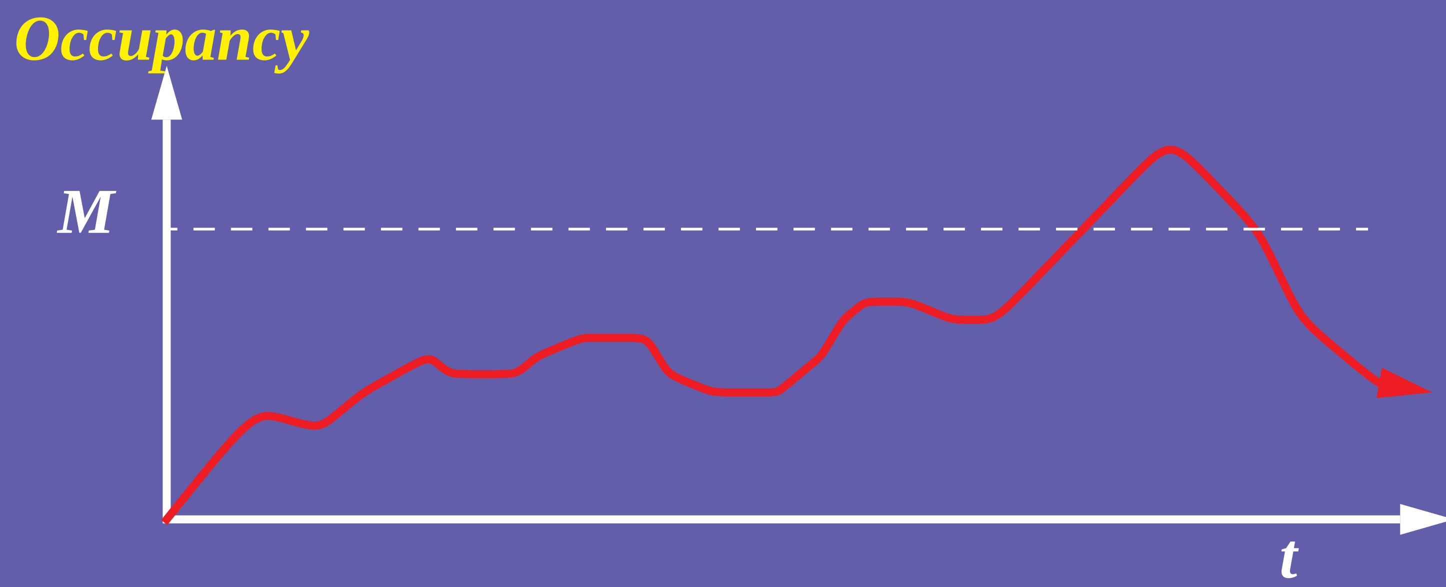


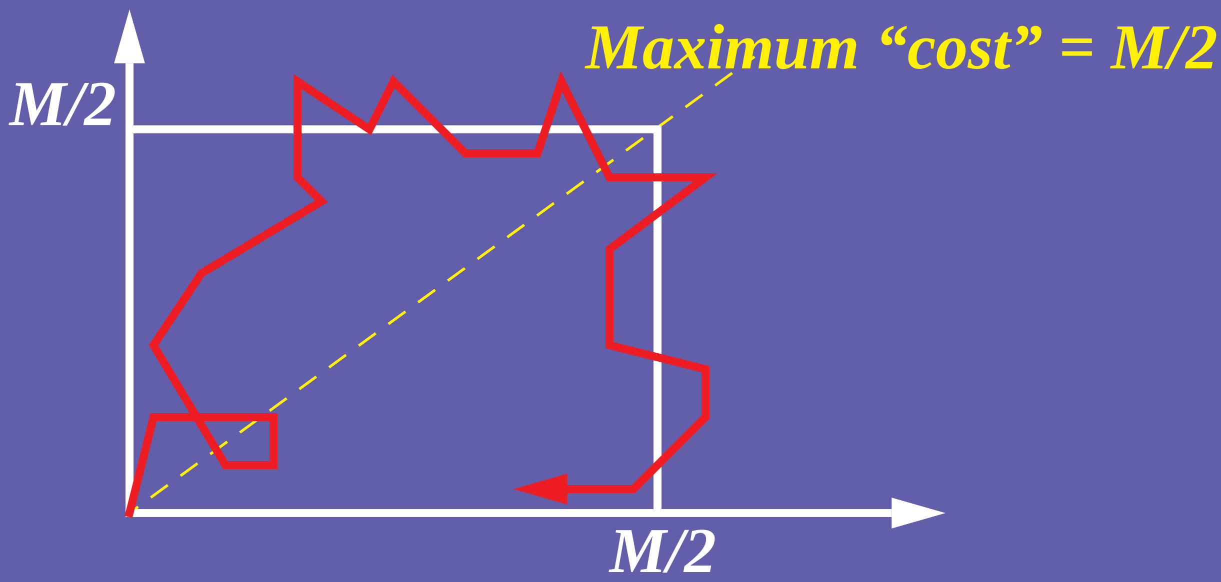

memory 1
memory 2


x2
x1 = x2
M/2
ii
i
iv
iii
v
x1
M/2

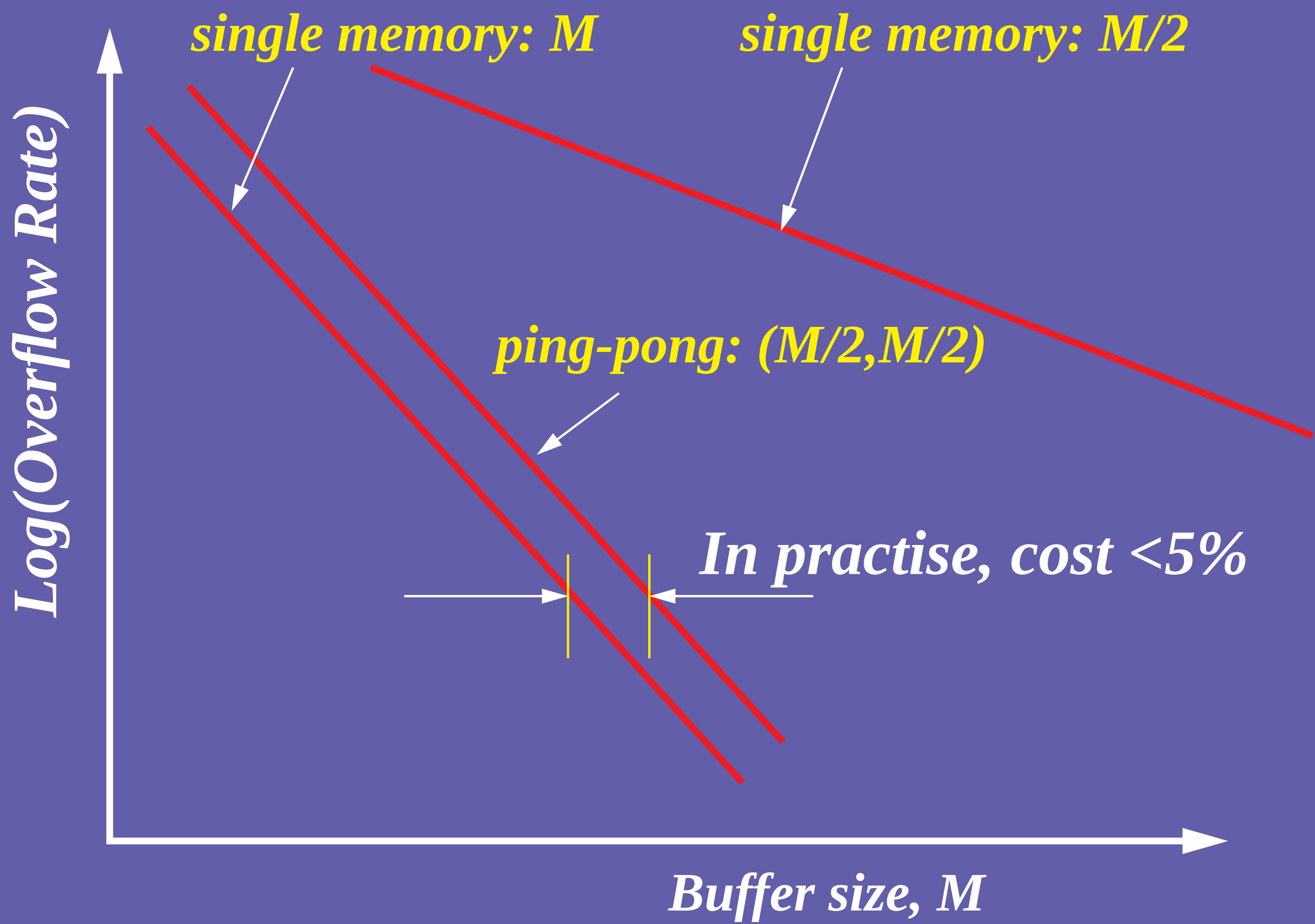

x2
x1 = x2
M/2
x1
M/2

Fast Buffering
Ping-pong Memory





Fast Buffering
Ping-pong Memory



Some Results
Input Queued Switch

$$\textit{Wastage Factor},\ \omega(R) \equiv \frac{M(R) - \tilde{M}(R)}{M(R)}$$

- $\omega(R)$ decreases with M
- $\omega(R)$ decreases with burstiness
- $\omega(R)$ decreases with load
- $\omega(R)$ decreases with number of ports

1. Accelerating Forwardng Decisions:

- Longest-matching prefixes

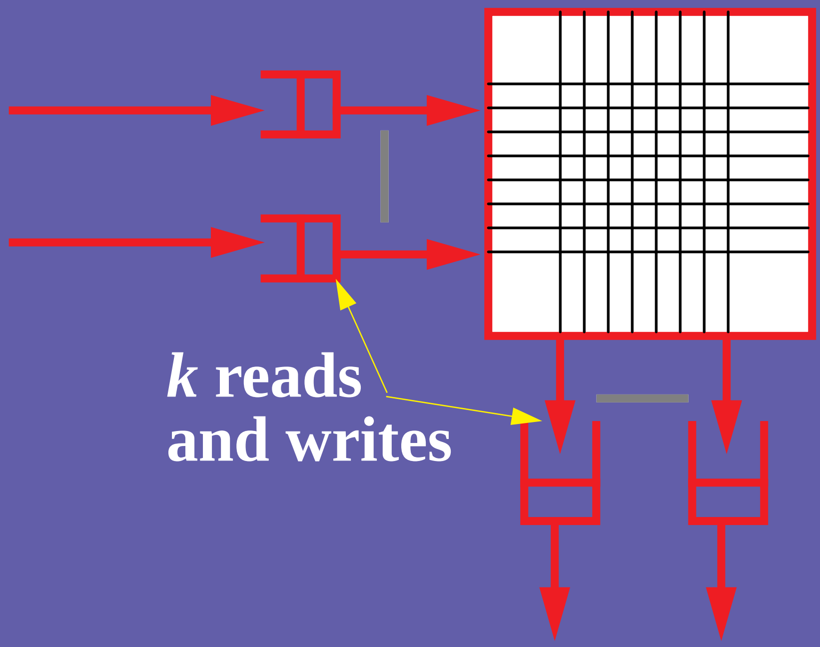
2. Interconnections: Switched Backplanes

- Input Queueing
  - Theory
  - Unicast
  - Multicast
- Fast Buffering
- Speedup
- The *Tiny Tera* Project

Matching Output Queueing with Input- and Output- Queueing

How much speedup is enough?

Combined Input- and Output-Queueing:



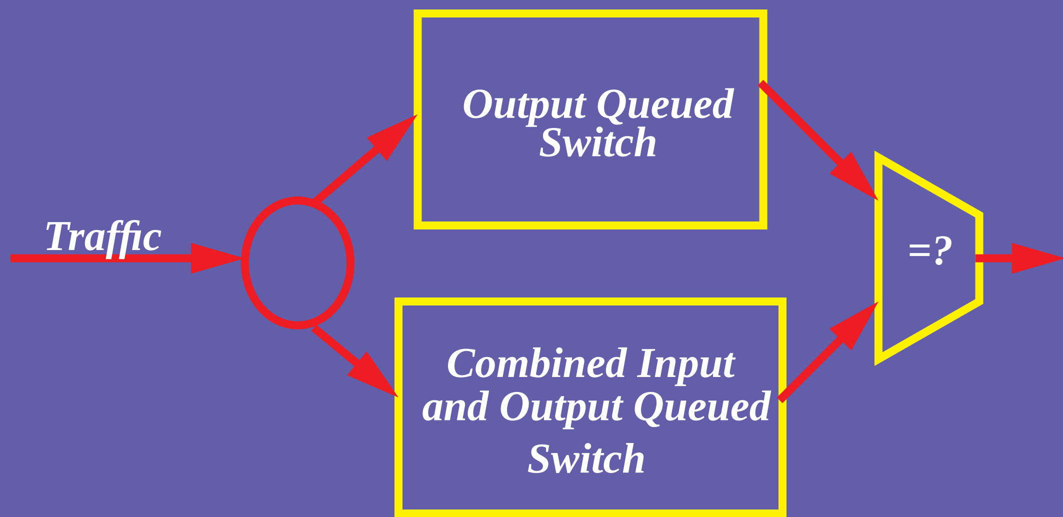
Matching Output Queueing with Input- and Output- Queueing

How much speedup is enough?

Conventional wisdom suggests:

A speedup $k = 2 - 4$ ***leads to high throughput***

Matching Output Queueing with Input- and Output- Queueing



Fact *To match output queueing, with FIFO input queues:*

$k = N$ *is necessary and sufficient.*

Fact *To match output queueing, with virtual output queues:*

$k = \left(2 - \frac{1}{N}\right)$ *is necessary and sufficient.*

1. Accelerating Forwardng Decisions:

- Longest-matching prefixes

2. Interconnections: Switched Backplanes

- Input Queueing
  - Theory
  - Unicast
  - Multicast
- Fast Buffering
- Speedup
- The *Tiny Tera* Project