Evolution of Packet Switching Architectures : from Gigabit Switches to 100 Terabit Switches

Cheng-Shang Chang

Institute of Communications Engineering
National Tsing Hua University
Hsinchu 300, Taiwan, R. O. C

Outline

- Motivation
- The Switch Architecture
  - Shared Memory Switches
  - Shared Medium Switches
  - Input-buffered Switches
  - Load Balanced Birkhoff-von Neumann Switches
- Stanford's implementation for 100 terabits/sec optical router
- Conclusions

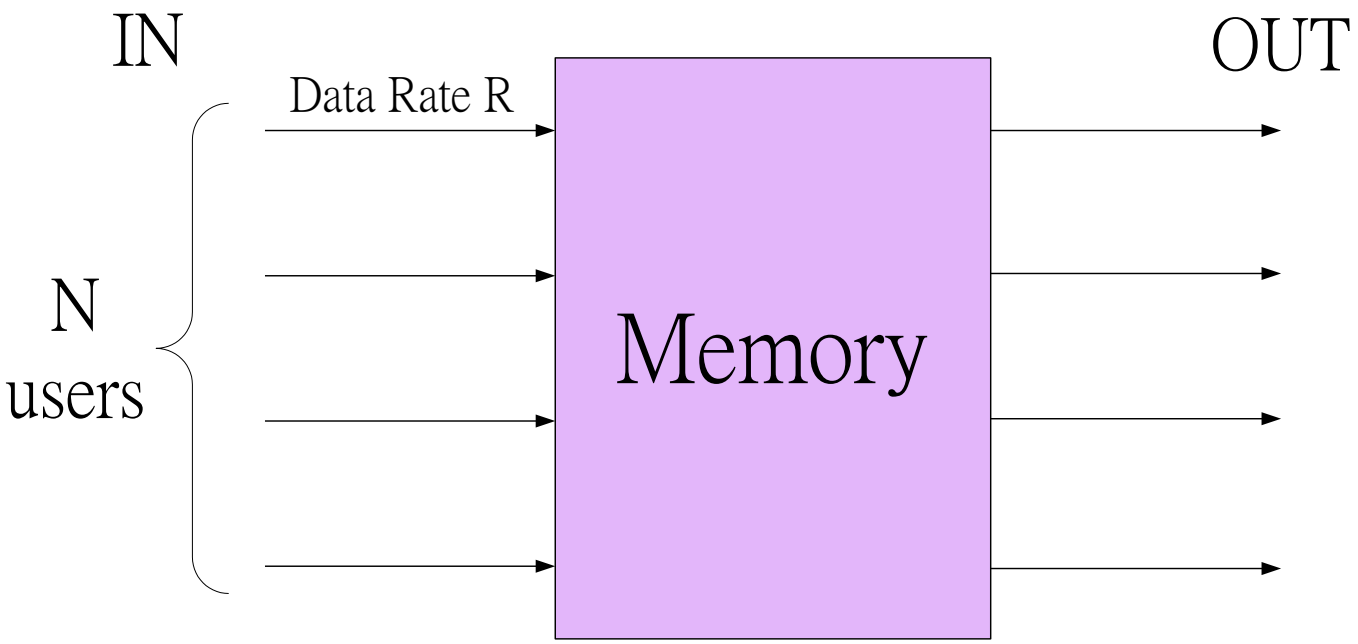
Motivation

- A switch is a network element with multiple input ports and output ports
- M $\times$ N switch: M input ports and N output ports
- Basic functions:
  - Table lookup
  - Message copying

Fundamental Problem

- The speed of light is much faster than the speed of electrons

Shared Memory Switch



Memory Speed $\geq 2 \cdot N \cdot R$

Shared Memory Switch

- For an N×N shared memory switch, there are N write operations for the N input ports and N read operations for the N output ports per time slot.
- The memory access speed must be at least 2N × link speed.
- Scalability problem

Memory Speed

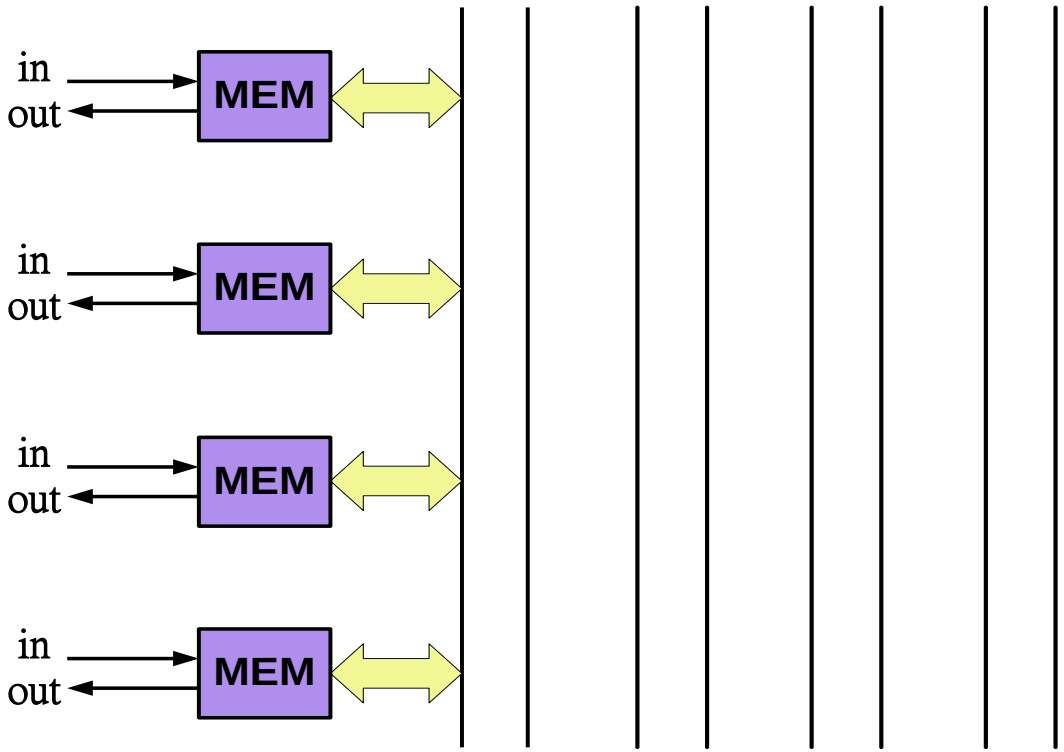
- The memory access time for the current DRAM is roughly 10 ns (nano second)
- For a packet of 64 bytes (512 bits), the memory access speed of a shared memory switch is roughly 51.2 Gbits/sec.
- If the line speed is 2.48 Gbits/sec (OC48), then a shared memory switch can support up to 10 input/output ports.

Commercial Products

- **CheetahSwitch™** Source: http://www.accton.com.tw
  - Store-and-Forward
  - Full Aggregate bandwidth:16Gbps

- **DES-6300** Source: http://www.dlink.com.tw
  - Backplane switch fabric bandwidth: 32Gbps

Shared Medium Switch

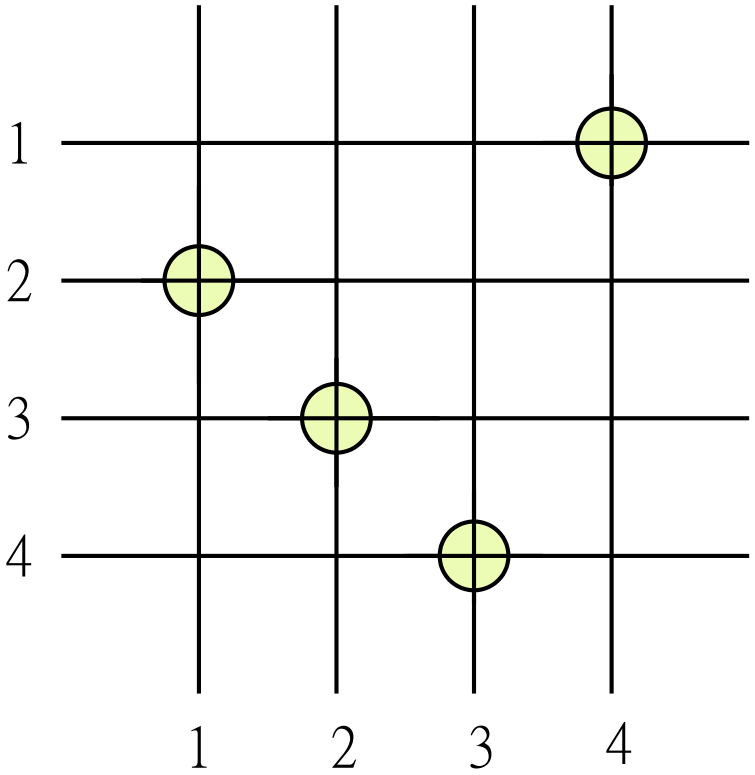


High speed bus Memory Speed $\geq N \cdot R$

Adding more buses Number of buses = Number of users

Memory Speed $\geq R$

Conflicts



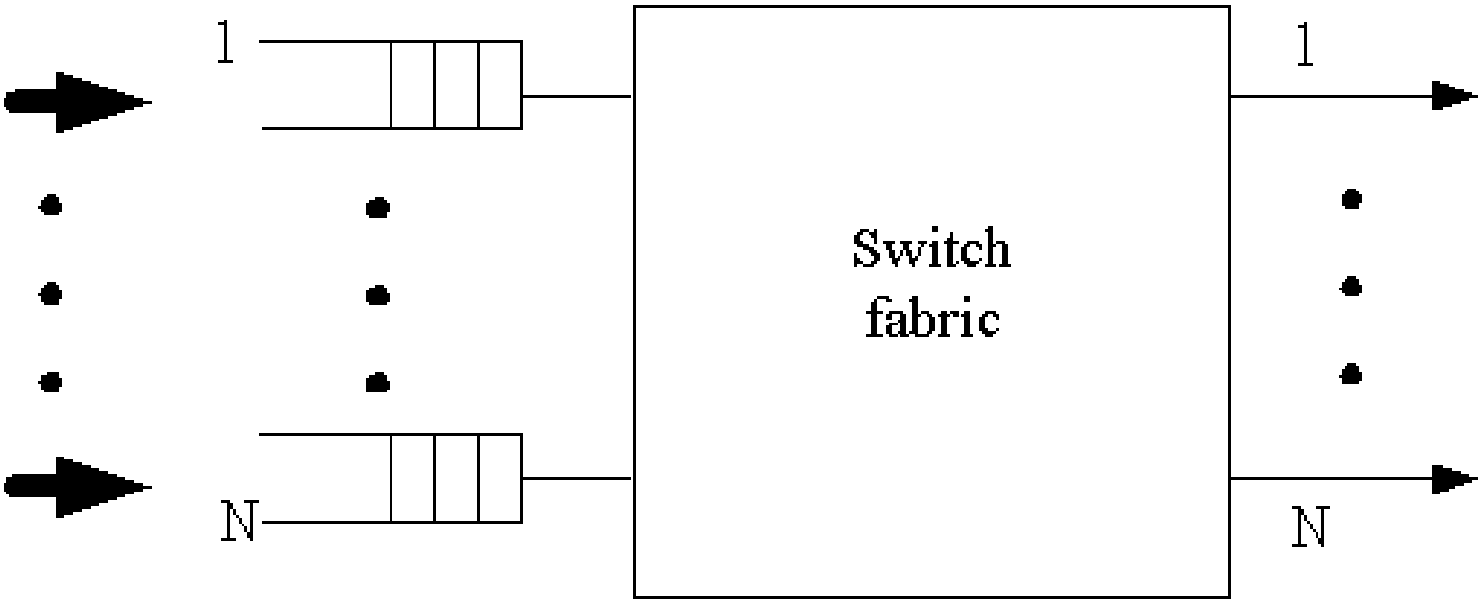
Permutation Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Input-buffered Switches

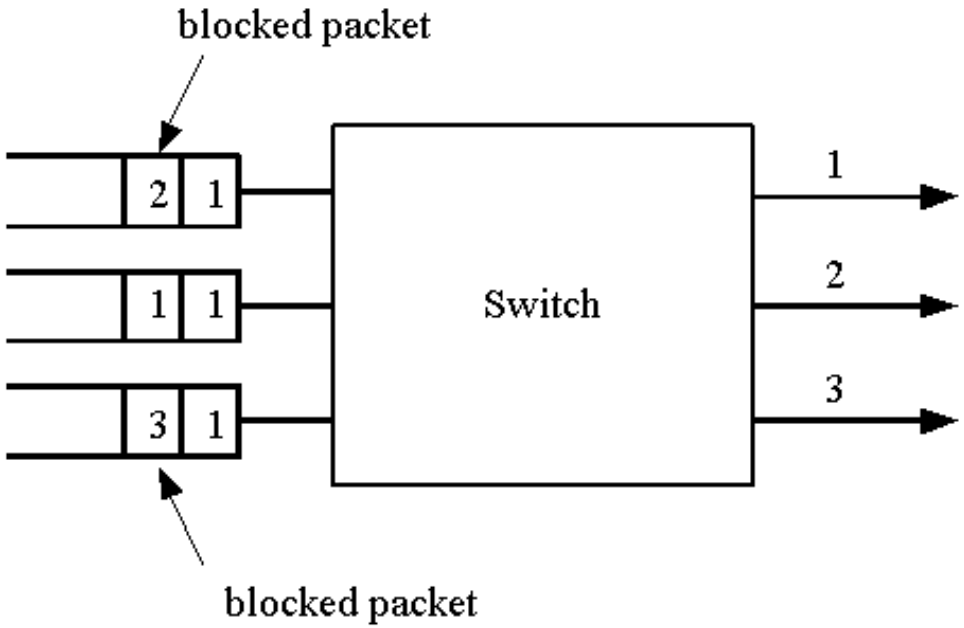
- If two or more input ports would like to transmit a packet to a particular one output port, only one of them is allowed to do so and the rest of them need to buffer their packets at the input ports.
- Need a queue at each input port.

Input-buffered Switches

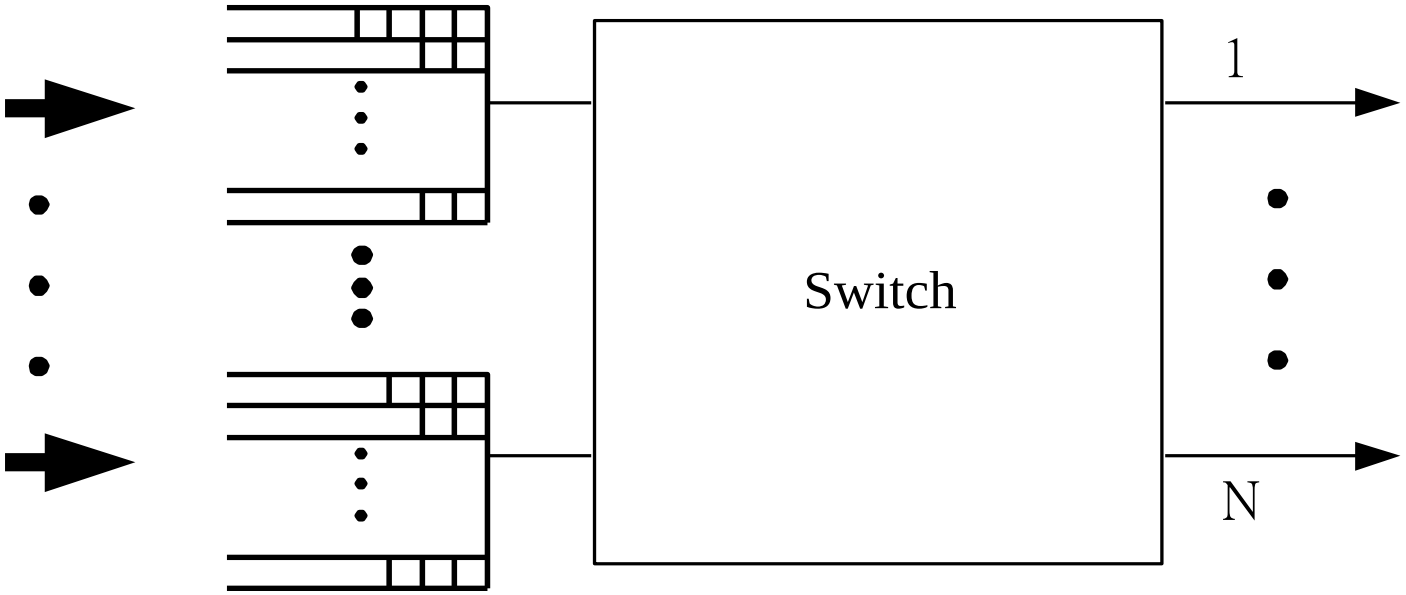


Head-of-line Blocking

- An input-buffered switch with each input maintaining a single First In First Out (FIFO).

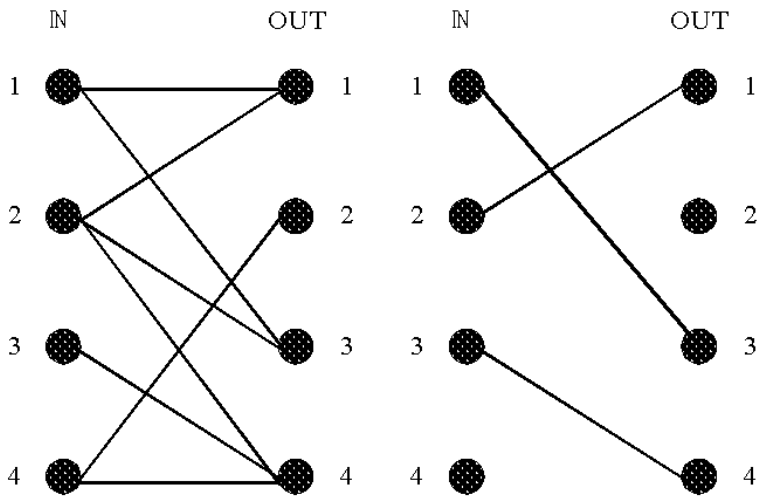


Solution: Virtual Output Queueing (VOQ)



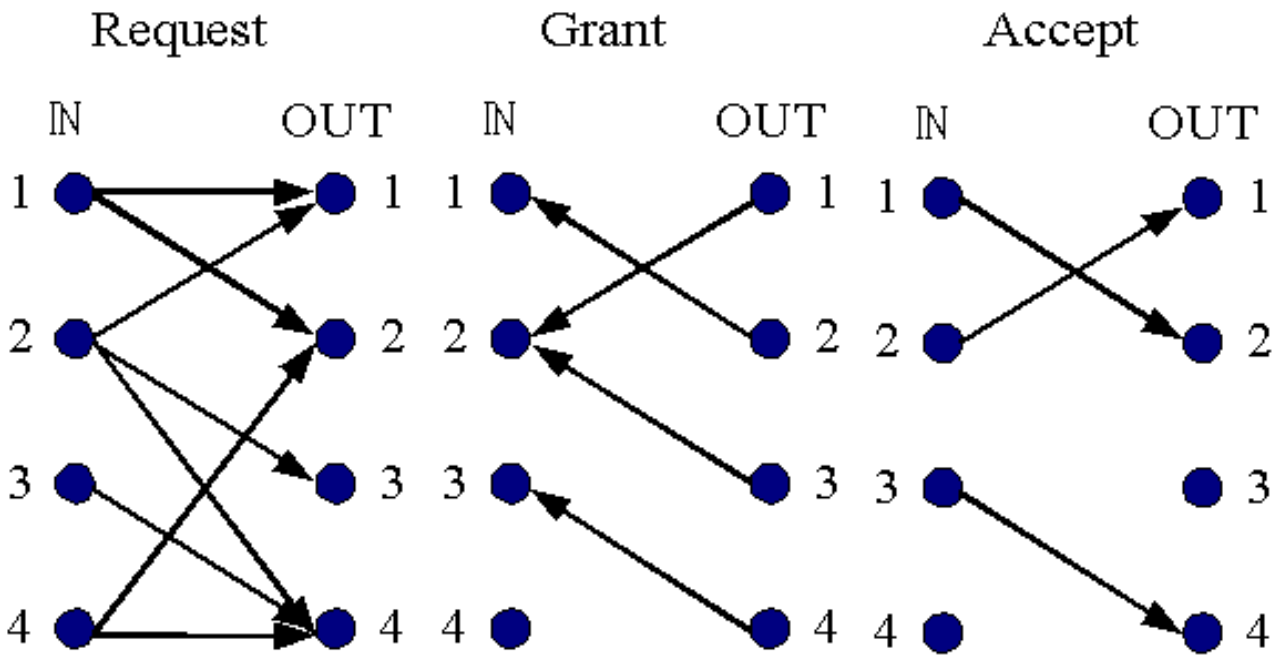
Matching

- The HOL packets need to be chosen under the following two constraints:
  - No more than one packets can be from the same input port.
  - No more than one packets can be sent to the same output port.



Parallel Iterative Matching (PIM)

- Step 1. Request. Each unmatched input sends a request to every output for which it has a non-empty VOQ.
- Step 2. Grant. If an unmatched output receives any requests, it grants to one by randomly selecting a request uniformly.
- Step 3. Accept. If an input receives a grant, it accepts one by randomly selecting a grant uniformly.


Request
IN
OUT
1
2
3
4
1
2
3
4
Grant
IN
OUT
1
2
3
4
1
2
3
4
Accept
IN
OUT
1
2
3
4
1
2
3
4

Variants of Matching

- *i*SLIP
- Wave front arbitration
- Maximum weighted matching
- DRRM
- Stable matching in combined input/output queueing (CIOQ)

Overheads

- Communication overhead: one has to gather the information of the buffers at the inputs.
- Computation overhead: based on the gathered information, one then applies a certain algorithm to find a matching.
- Scalabilty problem: if we use a single bit to indicate whether a VOQ is empty, then we have to transmit N bits from each input (to a central arbiter or to an output) in every time slot.

Cisco 12000 Series Routers

- Input-buffered switch with matching
- Up to 320 Gbps switching fabric capacity
- N=16 (16 input/output ports)

Source: http//www.cisco.com

Juniper T640 Router

- 640 Gbps of throughput
- 40 Gbps per slot
- N=16?

Source: http://www.juniper.net

Throughput Problem of Matching

- R= $\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix}$
- Convex combination of permutation matrices
- $R \le \sum_i \phi_i P_i$, $\sum_i \phi_i = 1$, $0 \le \phi_i \le 1$

  P_i : permutation matrix

No Overbooking Conditions

- Convex combination of permutation matrices are doubly stochastic matrices
  - $\sum_i r_{ij} \le 1$: the total rate to a particular output is not greater than 1
  - $\sum_j r_{ij} \le 1$: the total rate from a particular input is not greater than 1
- Question :
  Is any doubly stochastic rate matrix achievable ?

Birkhoff-von Neumann switch

- An input-buffered switch with VOQ.
- Use the following algorithm to provide uniform rate guarantees for each input-output pair.
- Algorithm 1 (von Neumann 1953)
  Transform the rate matrix (a doubly substochastic matrix) into a doubly stochastic matrix.
- Algorithm 2 (Birkhoff 1946) Decompose the doubly stochastic matrix as a convex combination of permutation matrices.
- Algorithm 3 Use the PGPS algorithm as the scheduling policy for the decomposition.

Birkhoff-von Neumann switch

- No communication overheads.
- (Memory complexity) The number of permutation matrices is $O(N^2)$.
- (On-line scheduling complexity) The complexity of on-line scheduling is $O(\log N)$.

- Drawbacks: (i) need to know the rate matrix to begin with, and (ii) memory complexity does not scale.

Example

Algorithm 1

$$R=\begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.6 \\ 0.3 & 0.1 & 0.4 & 0.8 \\ 0.1 & 0.5 & 0.2 & 0.8 \\ 0.7 & 0.8 & 0.7 & \end{bmatrix} \Rightarrow \begin{bmatrix} 0.6 & 0.2 & 0.1 & 0.9 \\ 0.3 & 0.1 & 0.4 & 0.8 \\ 0.1 & 0.5 & 0.2 & 0.8 \\ 1 & 0.8 & 0.7 & \end{bmatrix} \Rightarrow \begin{bmatrix} 0.6 & 0.3 & 0.1 & 1 \\ 0.3 & 0.1 & 0.4 & 0.8 \\ 0.1 & 0.5 & 0.2 & 0.8 \\ 1 & 0.9 & 0.7 & \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0.6 & 0.3 & 0.1 & 1 \\ 0.3 & 0.2 & 0.4 & 0.9 \\ 0.1 & 0.5 & 0.2 & 0.8 \\ 1 & 1 & 0.7 & \end{bmatrix} \Rightarrow \begin{bmatrix} 0.6 & 0.3 & 0.1 & 1 \\ 0.3 & 0.2 & 0.5 & 1 \\ 0.1 & 0.5 & 0.2 & 0.8 \\ 1 & 1 & 0.8 & \end{bmatrix} \Rightarrow \tilde{R}=\begin{bmatrix} 0.6 & 0.3 & 0.1 & 1 \\ 0.3 & 0.2 & 0.5 & 1 \\ 0.1 & 0.5 & 0.4 & 1 \\ 1 & 1 & 1 & \end{bmatrix}$$

Example

Algorithm 2

$$\tilde{R} = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.3 & 0.2 & 0.5 \\ 0.1 & 0.5 & 0.4 \end{bmatrix} = 0.2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.4 & 0.3 & 0.1 \\ 0.3 & 0 & 0.5 \\ 0.1 & 0.5 & 0.2 \end{bmatrix}$$

$$= 0.2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.4 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0.3 & 0.1 \\ 0.3 & 0 & 0.1 \\ 0.1 & 0.1 & 0.2 \end{bmatrix}$$

$$\tilde{R} = 0.2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.4 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Example

Algorithm 3

- virtual time:

(slot 0) 5, 2.5 , 5 , 10, 10 $\Rightarrow$(slot 1) 5, 5 , 5, 10, 10

$\Rightarrow$ (slot 2) 10, 5 , 5 , 10, 10 $\Rightarrow$ (slot 3) 10, 7.5, 5, 10, 10

$\Rightarrow$ (slot 4) 10, 7.5, 10, 10, 10

$\Rightarrow$ (slot 10) 15, 12.5, 15, 20, 20

packets : 2 : 4 : 2 : 1 : 1

Recap

1. What is the main problem of shared memory (output-buffered) switches?

Memory access speed.

2. Can FIFO queues achieve the fundamental limits (100% throughput) in input-buffered switches? Why?

No. Head-of-line (HOL) blocking.

3. How do people solve the HOL blocking problem in input-buffered switches?

Use Virtual Output Queueing (VOQ).

Recap

4. What do you have to do in input-buffered switches with VOQ?

Need to find and schedule matchings.

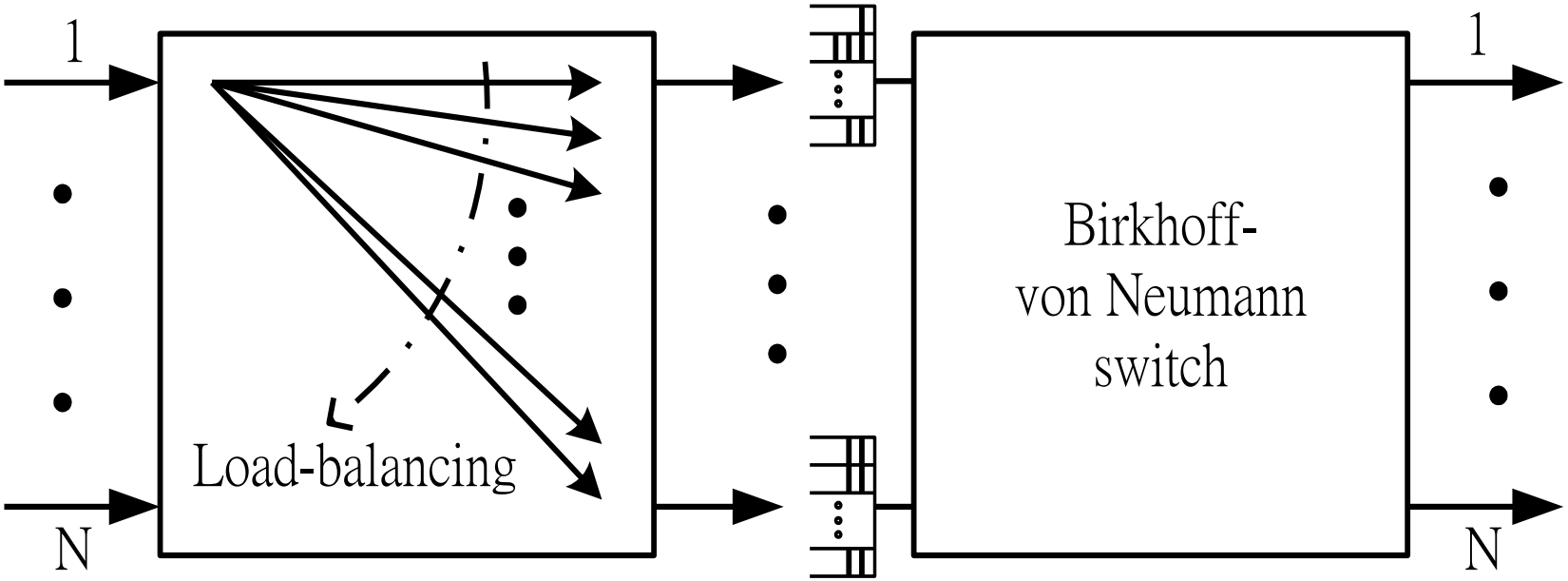
5. Why are the scalability problems for the switches that need to find matchings?

Communication and computation overheads.

6. Can the Birkhoff-von Neumann switches achieve the fundamental limits (100% throughput) in input-buffered switches? Why?

Yes. It uses the rate information to find the right decomposition.

Load Balanced Birkhoff-von Neumann Switch



The first stage performs load balancing

The second stage performs switching for load balanced traffic

The First Stage (Load Balancing)

- The first stage is a unbuffered crossbar switch with periodic connection patterns generated from a one-cycle permutation matrix.
- Packets arriving at the first stage at time t are switched instantly to the second stage according to the connection pattern.
- The first stage makes the input traffic to the second stage uniform (cf. randomization in Valiant 1982).

The Second Stage (Switching)

- The second stage is a Birkhoff-von Neumann switch.
- The second stage runs with a sequence of periodic connection patterns generated from a one-cycle permutation matrix (as in the first stage).
- The period is equal to the number of input/output ports.

One-cycle Permutation Matrix

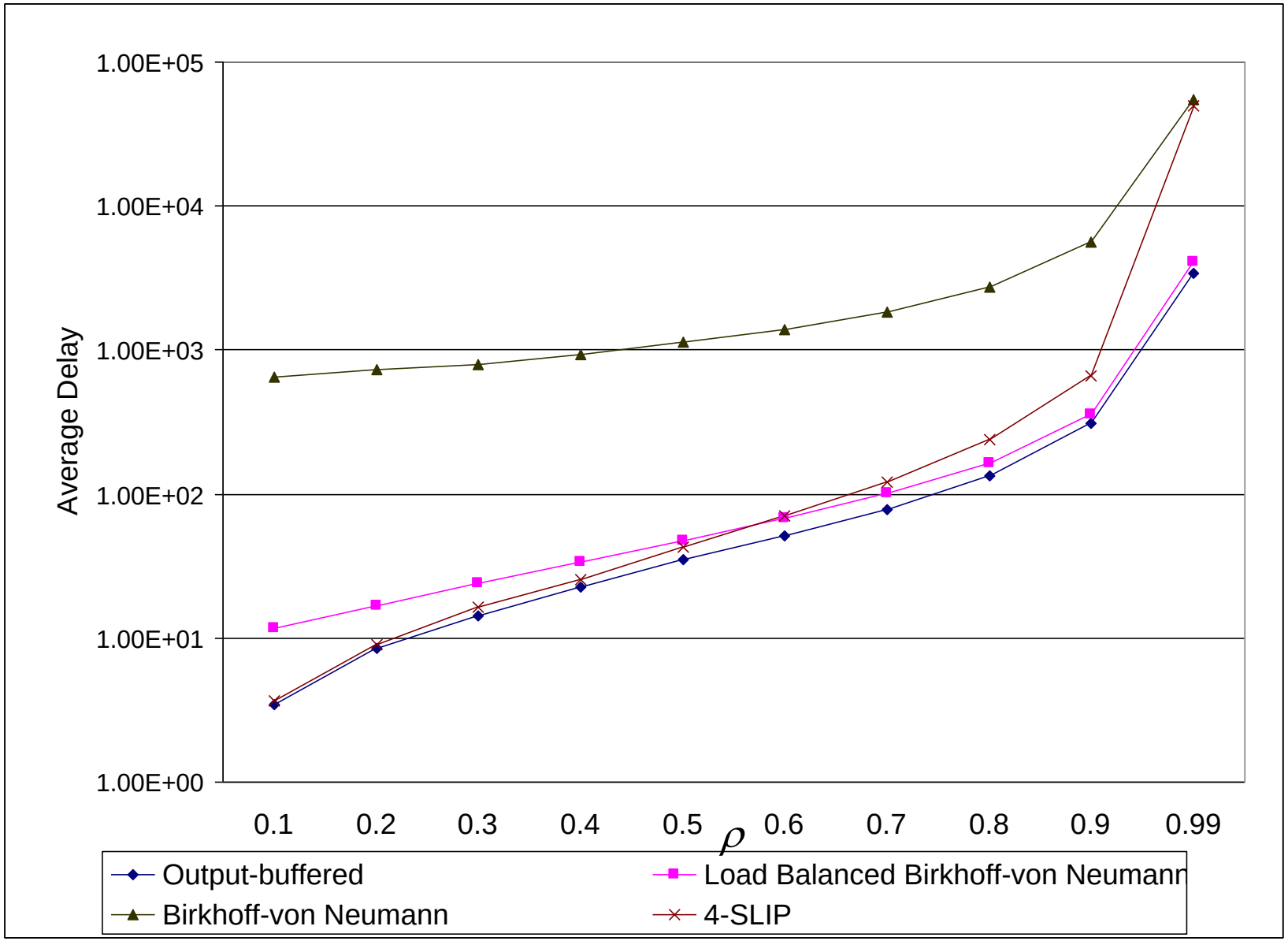
- Let P be any one-cycle $N \times N$ permutation matrix.
- Assign $P_k = P^k$ and $\phi_k = 1/N$ for $k = 1, \cdots, N$ in the second switch.
- As P is a one-cycle permutation matrix, P^N is the identity matrix, and the PGPS-like algorithm in the Birkhoff-von Neumann switch is simply periodic with period N.
- For example, when N=4:

$$p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad p^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

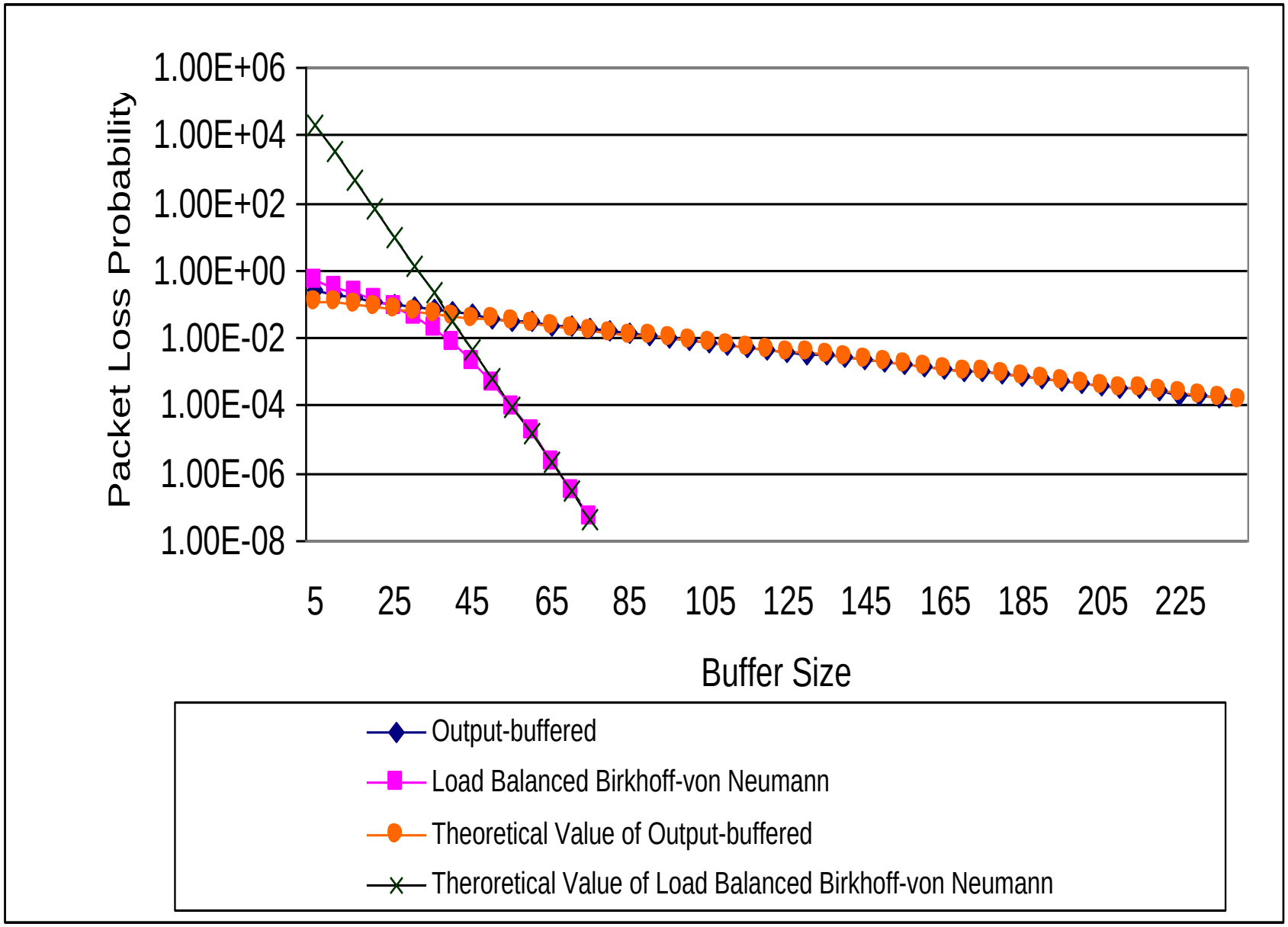
$$P^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad P^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Advantages

- Scalability: the on-line complexity is O(1).
- Low hardware complexity.
- 100% throughput.
- Low average packet delay in heavy load and bursty traffic.
- Efficient buffer usage.

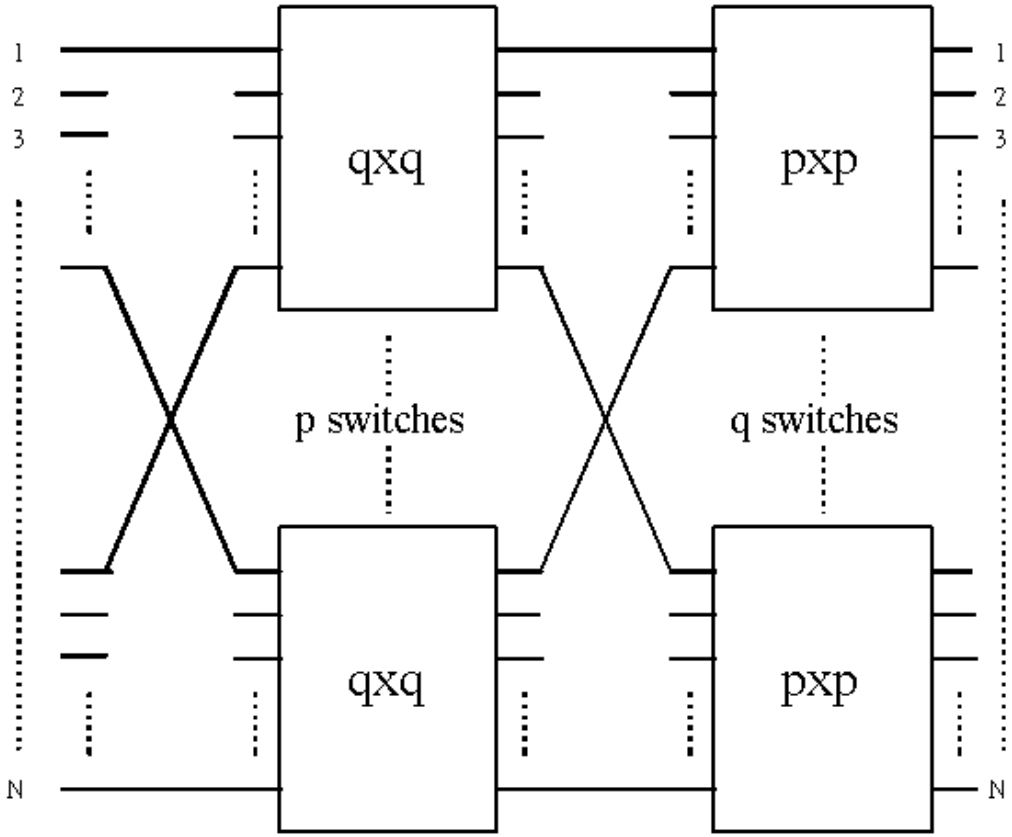


Uniformly Pareto bursty traffic, N=16

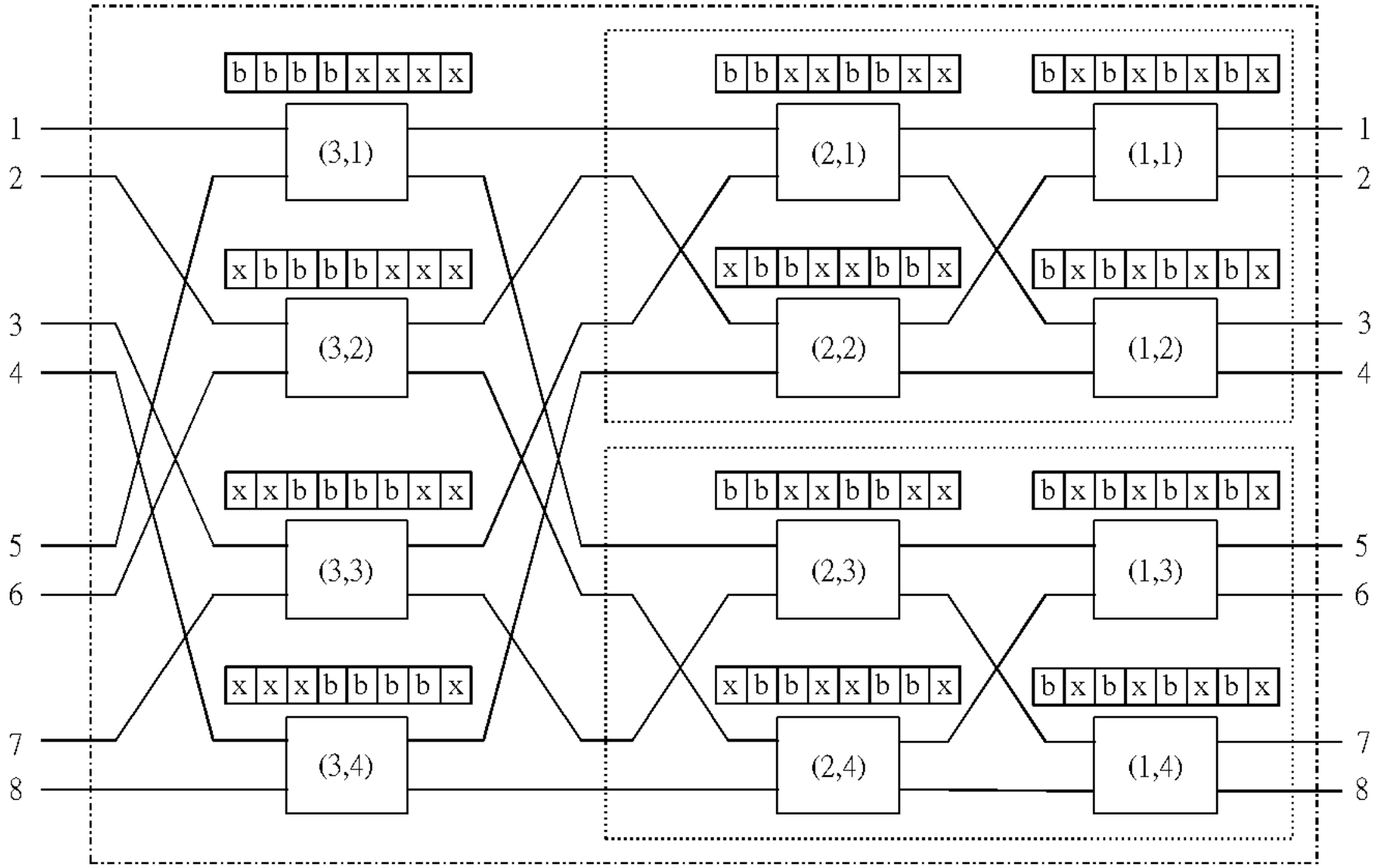


$$N = 16,\ \rho = 0.8,\ \theta^{*} = 0.4575$$

Recursive construction of the switch fabrics



An 8×8 switch fabric via 2×2 switches



Load Balanced Birkhoff-von Neumann Switch with One-stage Buffering

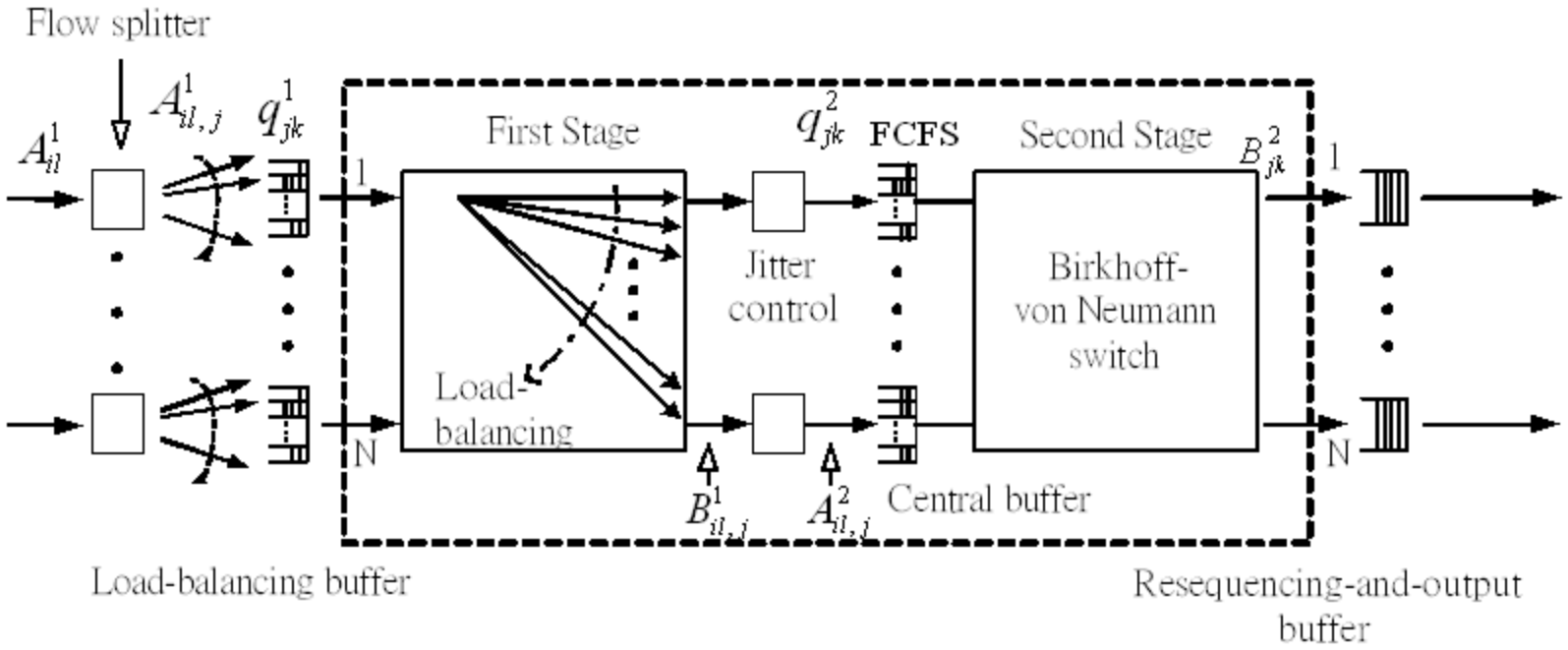
- On-line computational complexity for scheduling reduced from *O(logN)* to *O(1)*.
- Low average packet delay in heavy load and bursty traffic.
- More efficient buffer usage comparing with output-buffered switches.

➢ Main drawback :

Output traffic may be *out of sequence*.

➢ Solution: Add load balancing buffer and resequencing buffer.

Load Balanced Birkhoff-von Neumann Switch with Multi-stage Buffering



- Packets from the same flow are split in the round-robin fashion to the N virtual output queues and scheduled under FCFS policy.

Load Balanced Birkhoff-von Neumann Switch with Multi-stage Buffering (cont.)

- The resequencing-and-output buffer after the second stage keeps packets in sequence , and stores packets waiting for transmission from the output links.

➢ Main results of the approach:

- The end-to-end delay for a packet through the multi-stage buffering switch is bounded when comparing with the delay of an output-buffered switch.
- The load balancing buffer is bounded.
- 100% throughput for multicasting flows with fan-out splitting at the central buffer.

References

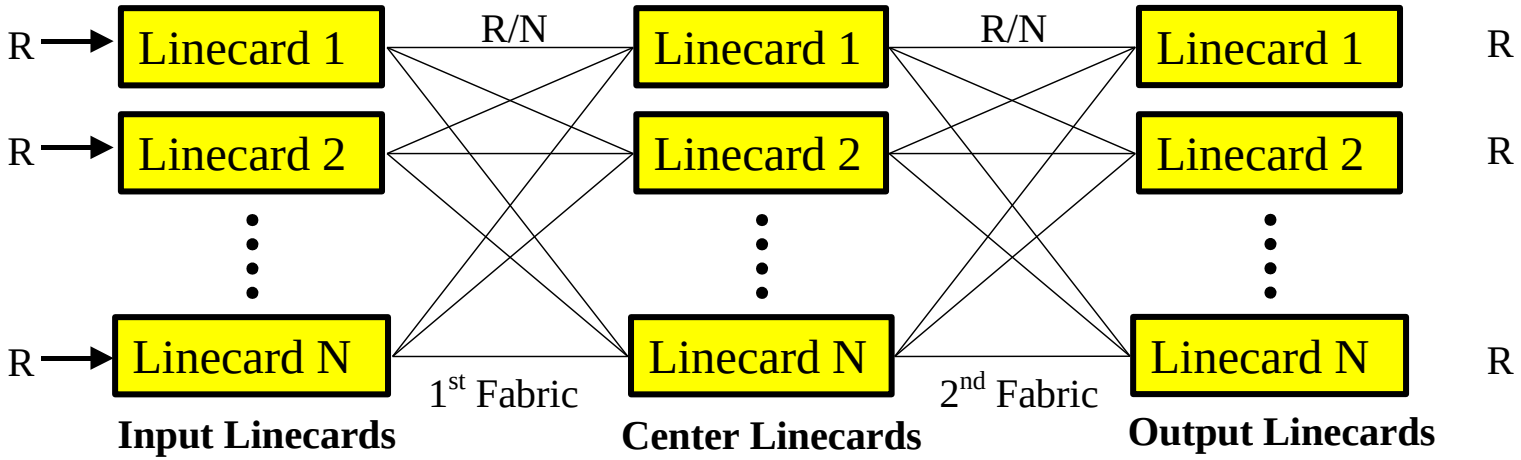
- C.S. Chang, W.J. Chen and H.Y. Huang, ``On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," IEEE IWQoS'99, pp. 79-86, London, U.K., 1999
- C.S. Chang, W.J. Chen and H.Y. Huang, ``Birkhoff-von Neumann input buffered crossbar switches,'' IEEE INFOCOM2000, pp. 1614-1623, Tel Aviv, Israel, 2000.
- C.S. Chang, D.S. Lee and Y.S. Jou, ``Load Balanced Birkhoff-von Neumann Switches, Part I: One-stage Buffering,“ Computer Communications, Vol. 25, pp. 611-622, 2002.
- C.S. Chang, D.S. Lee and C.M. Lien, ``Load Balanced Birkhoff-von Neumann Switches, Part II: Multi-stage Buffering, “Computer Communications, Vol. 25, pp. 623-634, 2002.
- C.S. Chang, D.S. Lee and C.Y. Yue, "Providing Guaranteed Rate Services in the load balanced Birkhoff-von Neumann switches," Proceedings of IEEE INFOCOM, 2003.
- S. Iyer and N. McKeown, ``Making parallel packet switch practical," Proc. IEEE INFOCOM 2001, Anchorage, Alaska, U.S.A.

Stanford's Implementation (100 Terabit Switches)

Source: http://tiny-tera.stanford.edu/~nickm/talks

Load Balanced Birkhoff-von Neumann Switches

Two fabrics each performing a cyclic shift

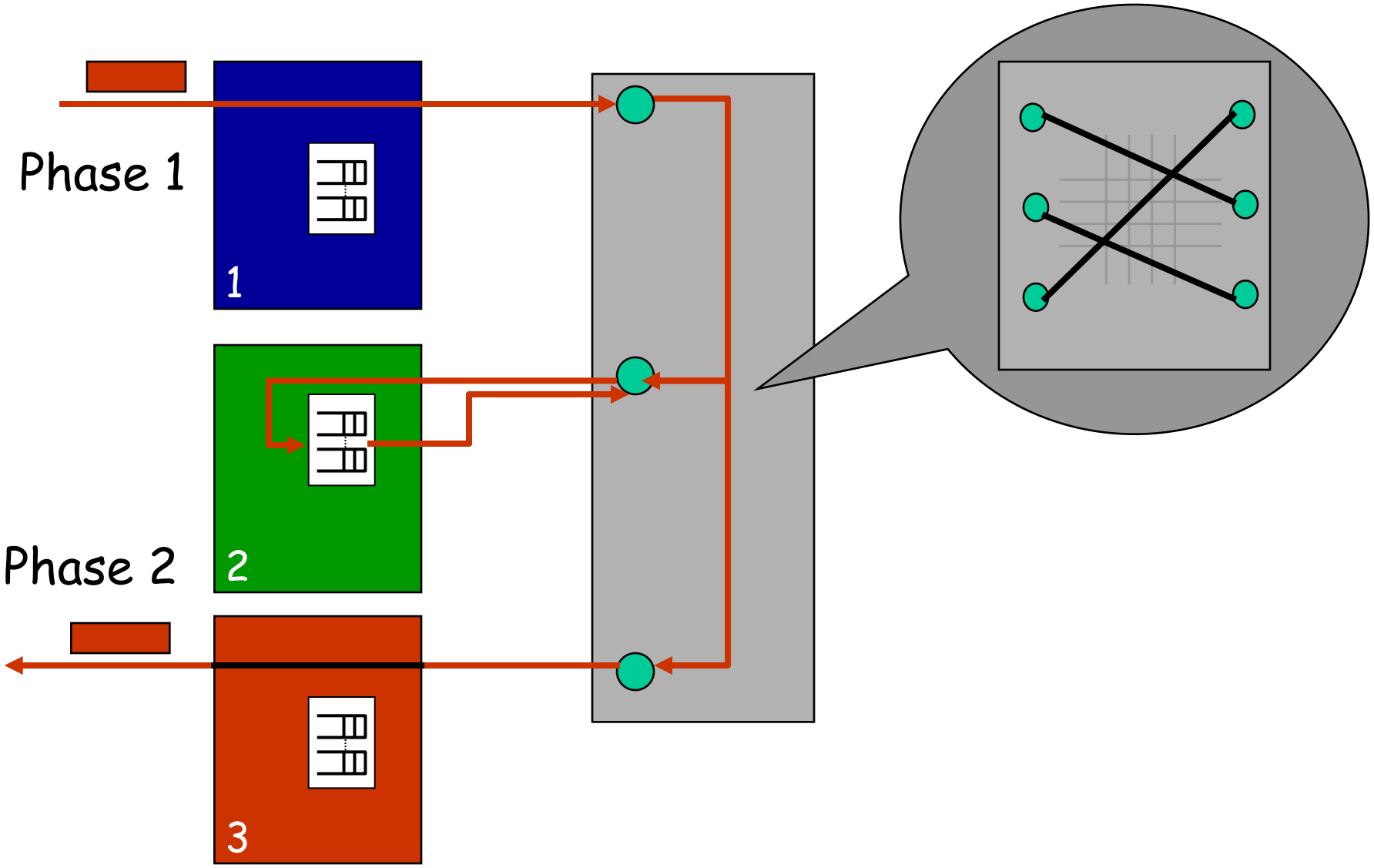


The algorithm presented by C.S. Chang et al.

REQUIREMENTS FOR THE ROUTER

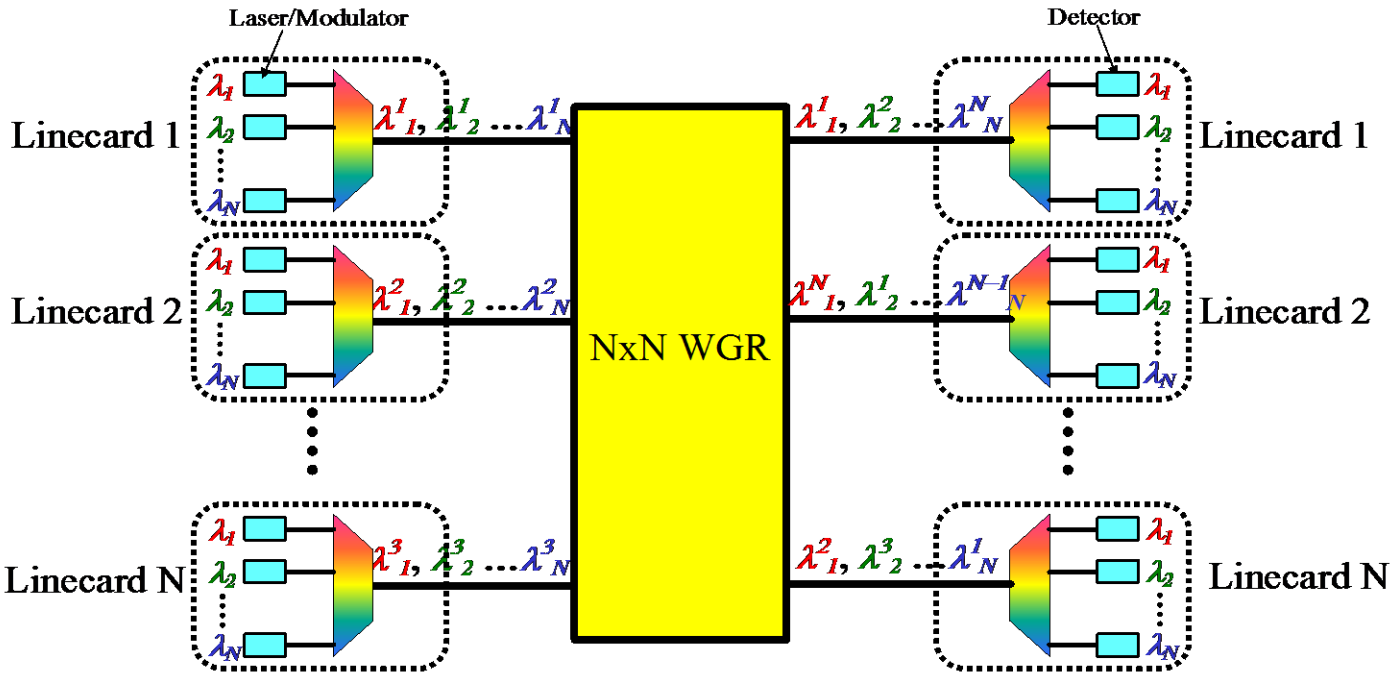
- An aggregate bandwidth of 100 Tb/s
- Having 625 linecards each running at a 160 Gb/s data rate (4 times OC768)
- Combine the two fabrics into one fabric with twice the aggregate bandwidth
- For each linecard to send 320Gb/s of data uniformly to all linecards

An optical two-stage switch

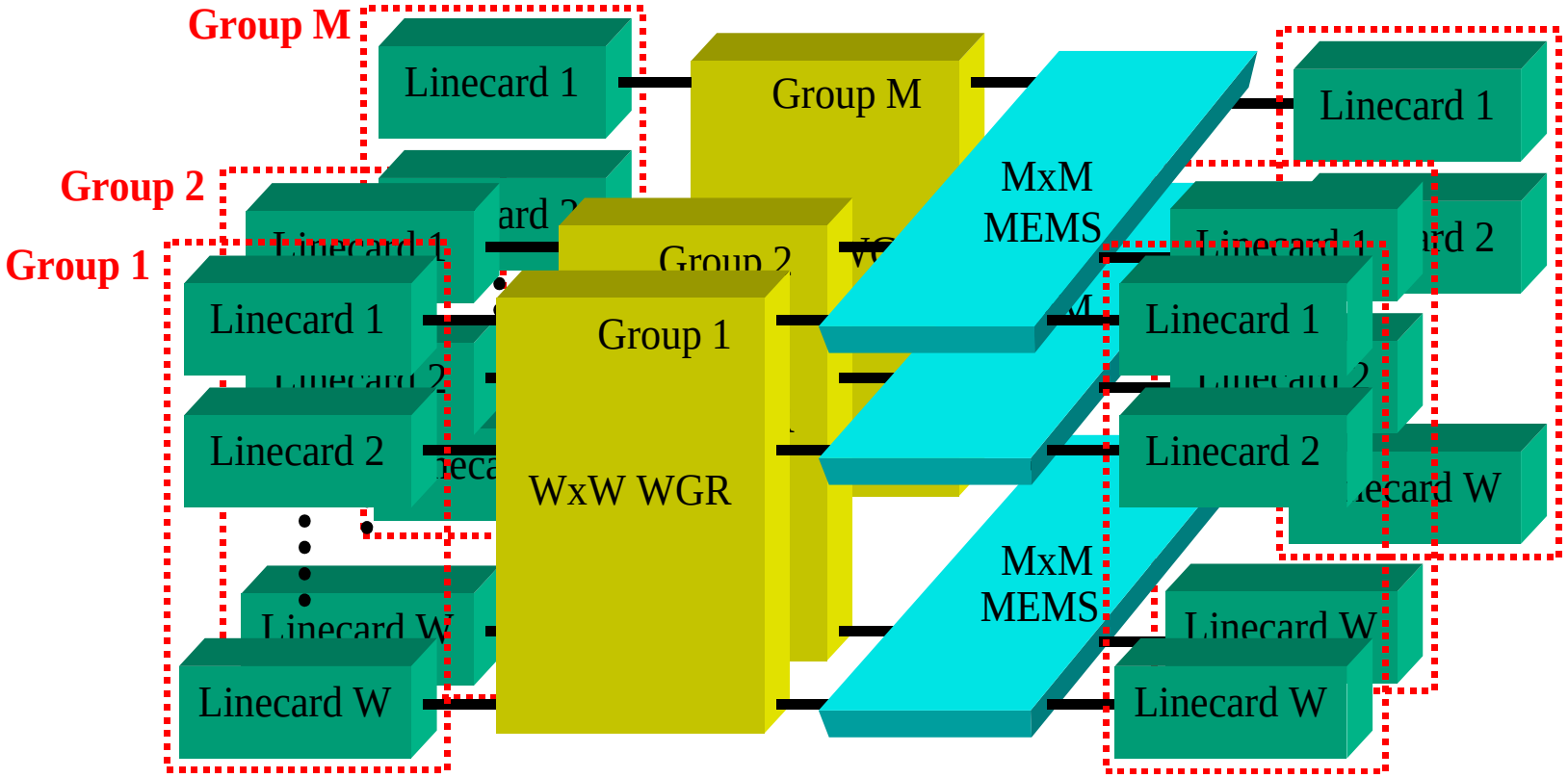


Basic Design

- Each fixed lambda laser carries 1/Nth of the data rate.
- Problem?

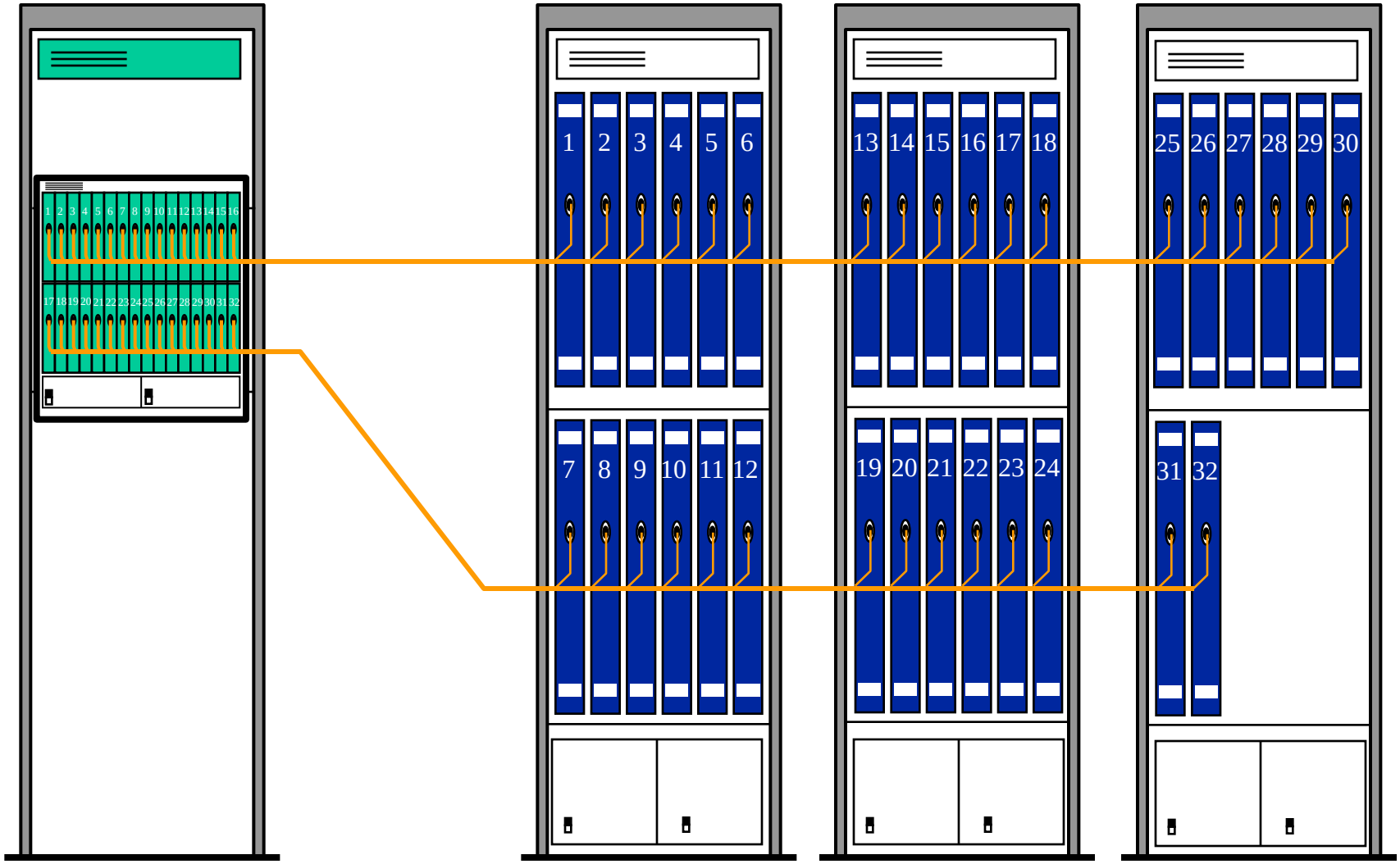


(Cont.)


Group M
Group 2
Group 1
Linecard 1
Linecard 2
Linecard W
Group M
Group 2
Group 1
WxW WGR
MxM
MEMS
MxM
MEMS
Linecard 1
Linecard 1
Linecard 2
Linecard W
Linecard W

Fourth-Generation Switches/Routers

Clustering and Multistage



THE END

- More information: http://www.ee.nthu.edu.tw/~cschangc
- 教育部顧問室通訊科技教育改進計畫 (高速交換原理與實作)
- Thanks for your attention.