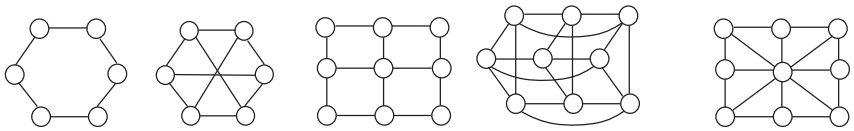
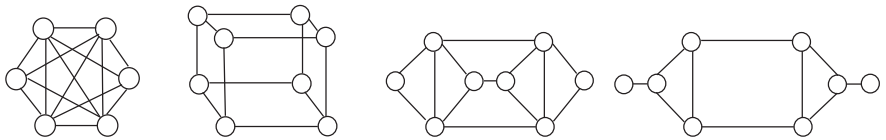
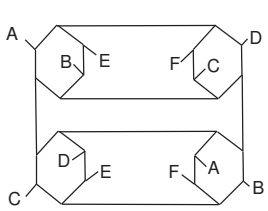
Static and Dynamic Networks: What are MINs?



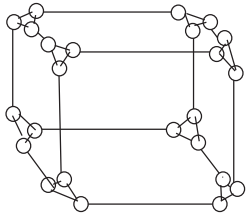
Cycle Chord Graph 2-d Mesh 2-d Torus Diagonal Mesh



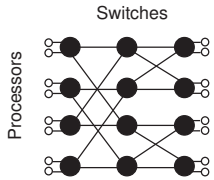
K6 Binary Hypercube Binary De Bruijn Shuffle Exchange



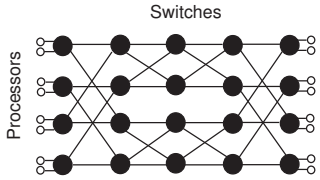
Star Graph (N=4!)



Cube-Connected Cycles



Butterfly Network

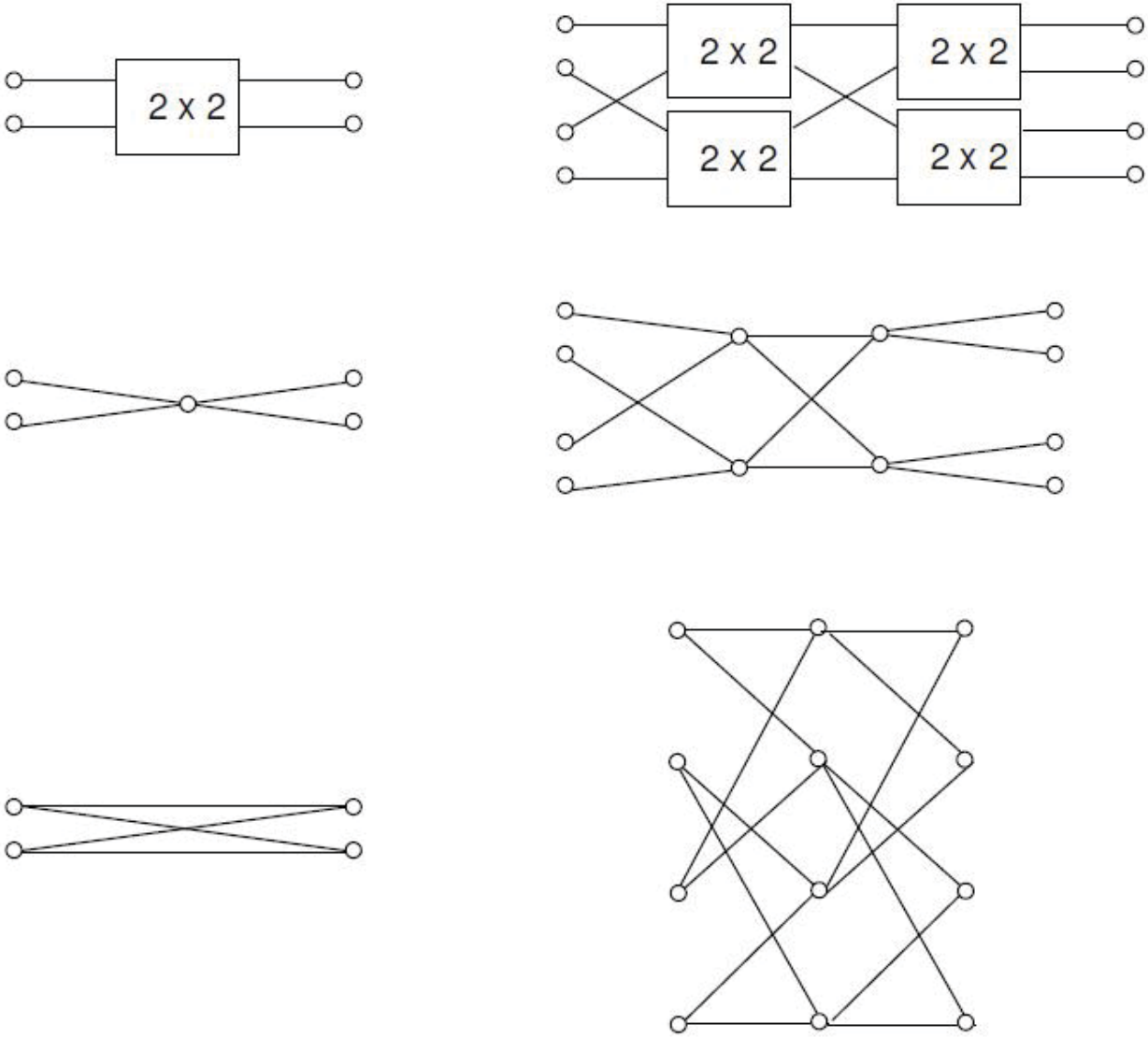


Benes Permutation Network

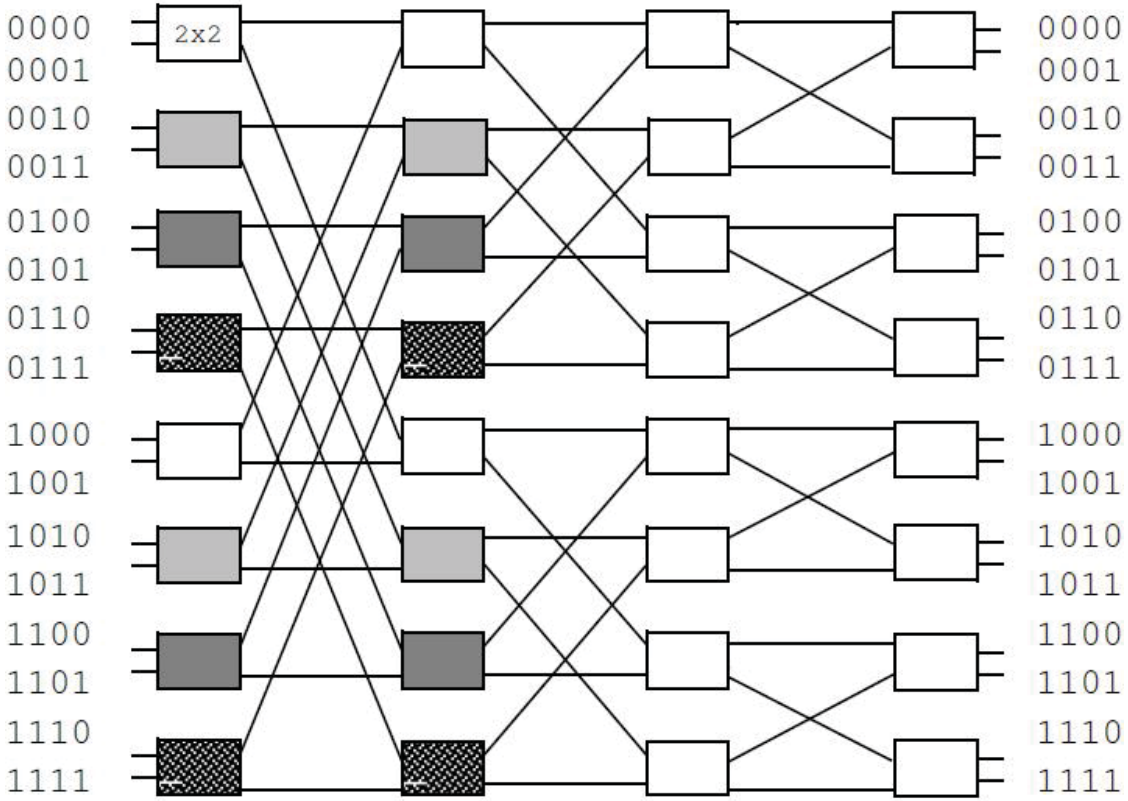
Classification of Multistage Networks



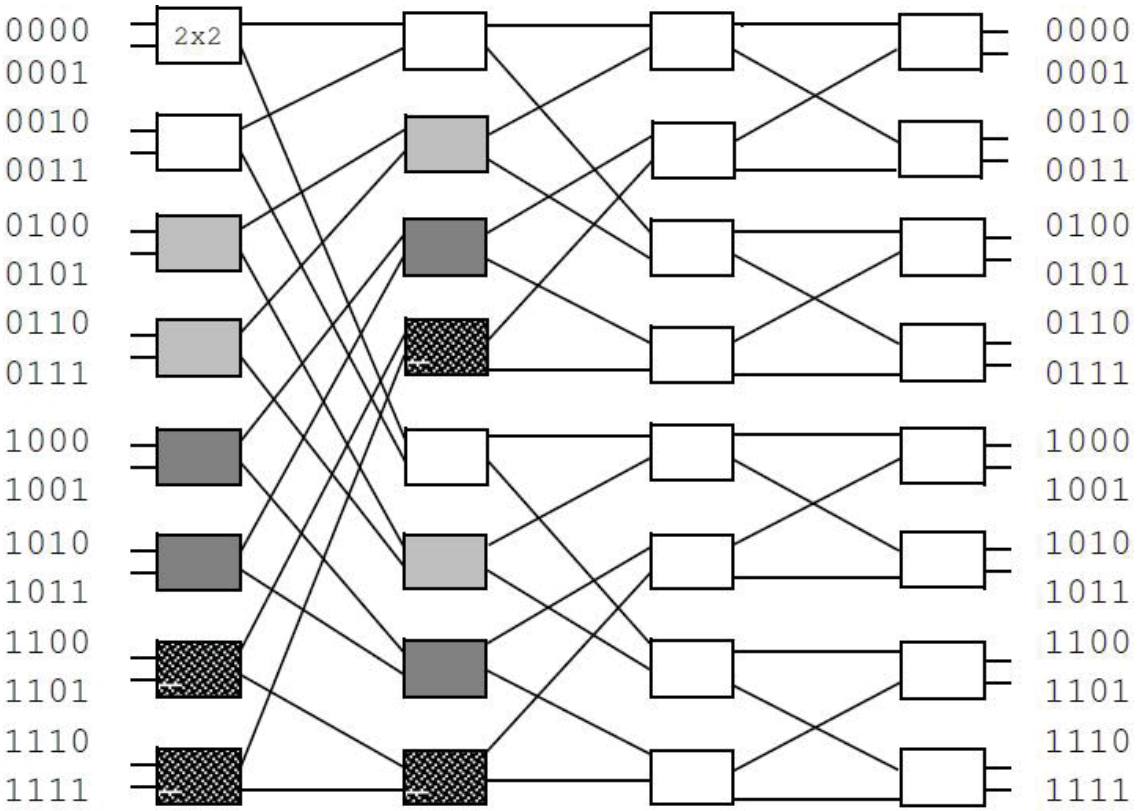
Representation of Multistage Networks



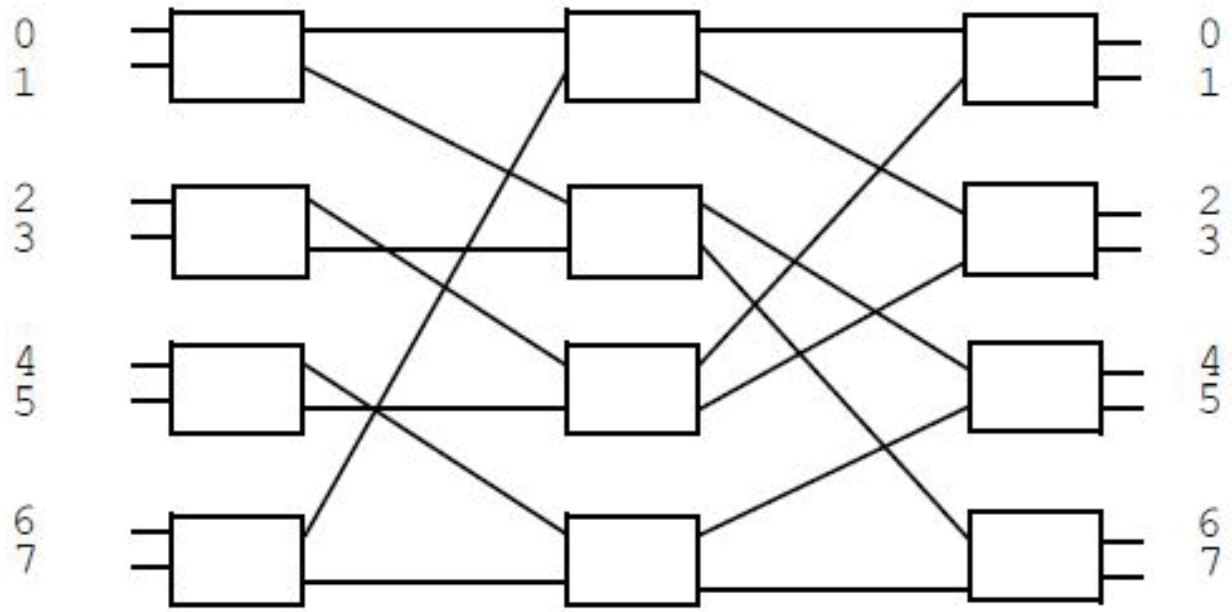
Blocking Networks - Butterfly



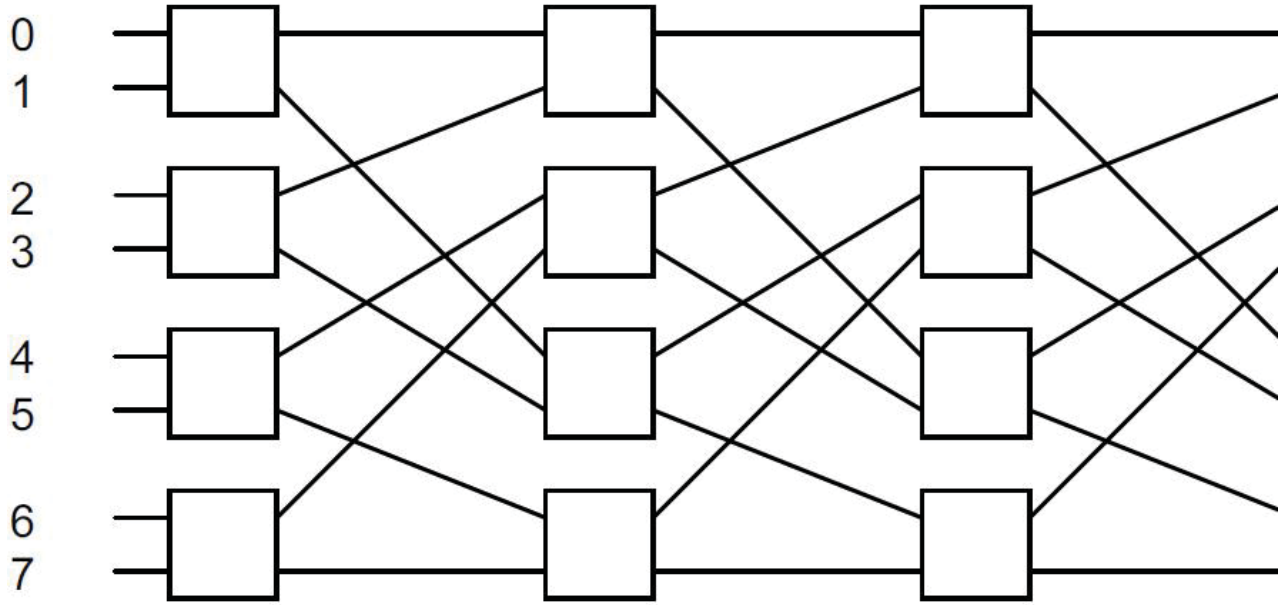
Blocking Networks - Baseline



Blocking Networks - Omega



Blocking Networks - Inverse Omega



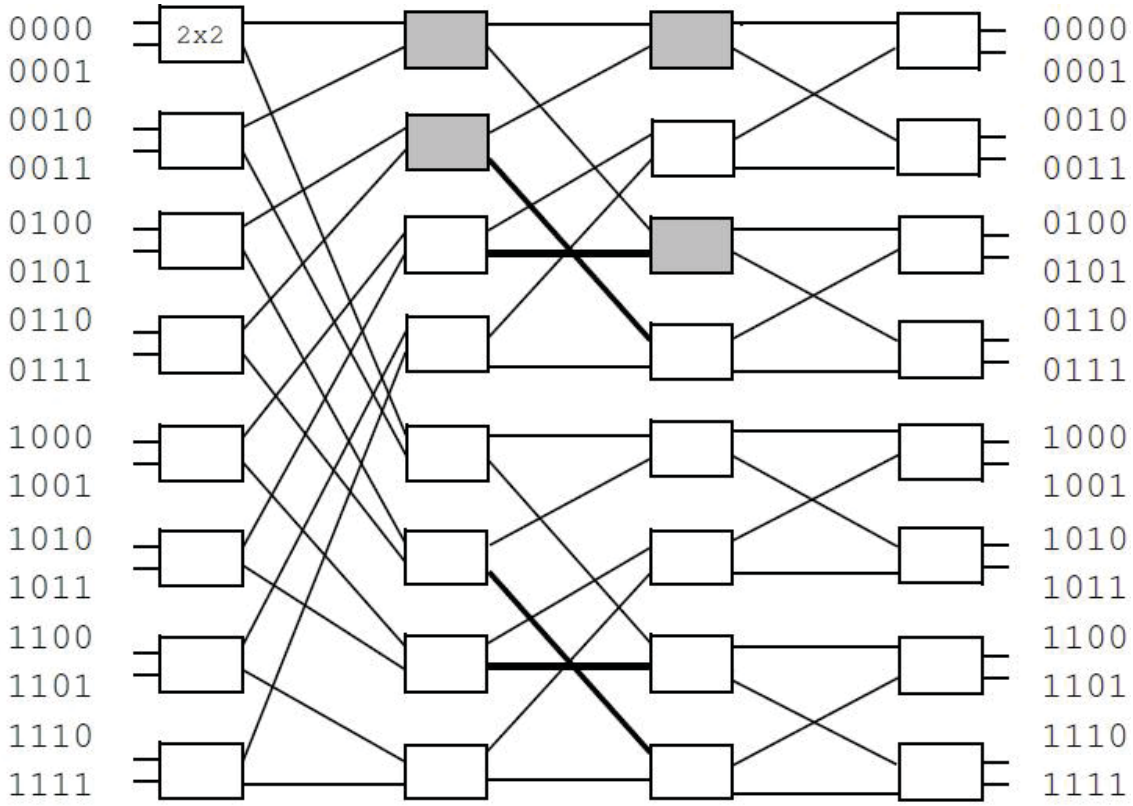
Δ-Networks

A network is called a Δ-network iff it consists of

- *a single switch (e.g. an $n \times n$ crossbar),* **OR**
- *one level of $n \times n$ switches, followed by n-disjoint Δ networks, Δ_1, Δ_2, $\ldots$, Δ_n. Furthermore, the buddy property must hold; input nodes in the network are connected to the first stage switches and output j of each switch in the first level is connected to an input node in Δ_j.*

Notice that, Δ-networks are not always topologically equivalent!

Δ-Networks



Network lacks distributivity & unique-path!

Δ^2 Networks

A Δ^2-network is a Δ-network from both directions.

Examples of Δ^2-networks: butterfly, shuffle-exchange (omega), inverse omega, baseline, reverse-shuffle, and indirect cube.

Two networks are **topologically equivalent** *if they can be obtained from the other by relabeling switches and input/output terminals.*

Two networks are **functionally equivalent** *if they can realize the same set of permutations.*

All Δ^2-networks are topologically equivalent, i.e., any two Δ^2-networks of degree n and dimension k, with $N = n^k$ inputs are isomporphic.

Nonblocking & Rearrangeable Networks

A strictly nonblocking network can always connect an idle pair of terminals without disturbing calls already in progress.

A wide-sense nonblocking network must suitably choose new routes (preprocessing).

A rearrangeable network may need to reassign new routes for existing ones in order to connect the idle pair. This is sometimes called a universal network.

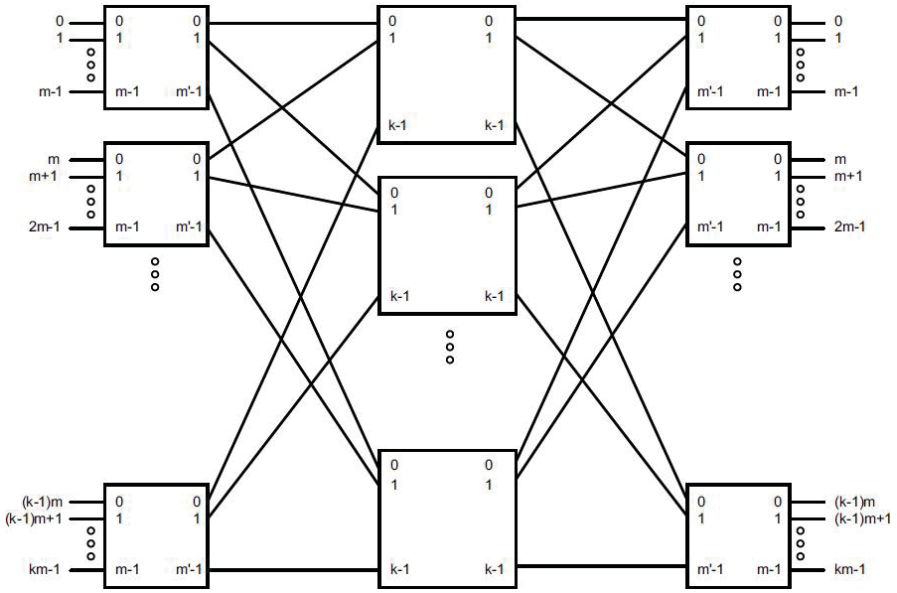
A blocking network does not realize all permutations.

Clos Networks - Definition

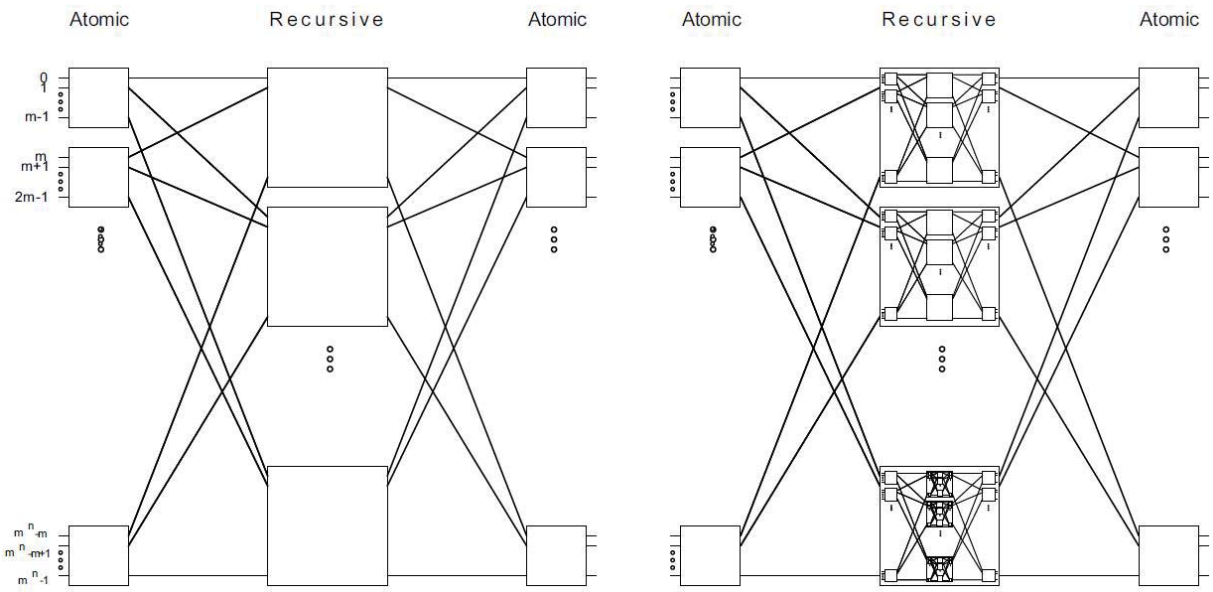
A strictly nonblocking Clos network of size N is defined as follows:

- *N/n switches of size $n \times m$ at Stage I*
- *m switches of size $\frac{N}{n} \times \frac{N}{n}$ at Stage II*
- *N/n switches of size $m \times n$ at Stage III*
- *Clos distributes one link to each switch in the middle stage. The j^{th} output of each switch is connected to j^{th} switch of next stage.*

Clos Network - General Construction

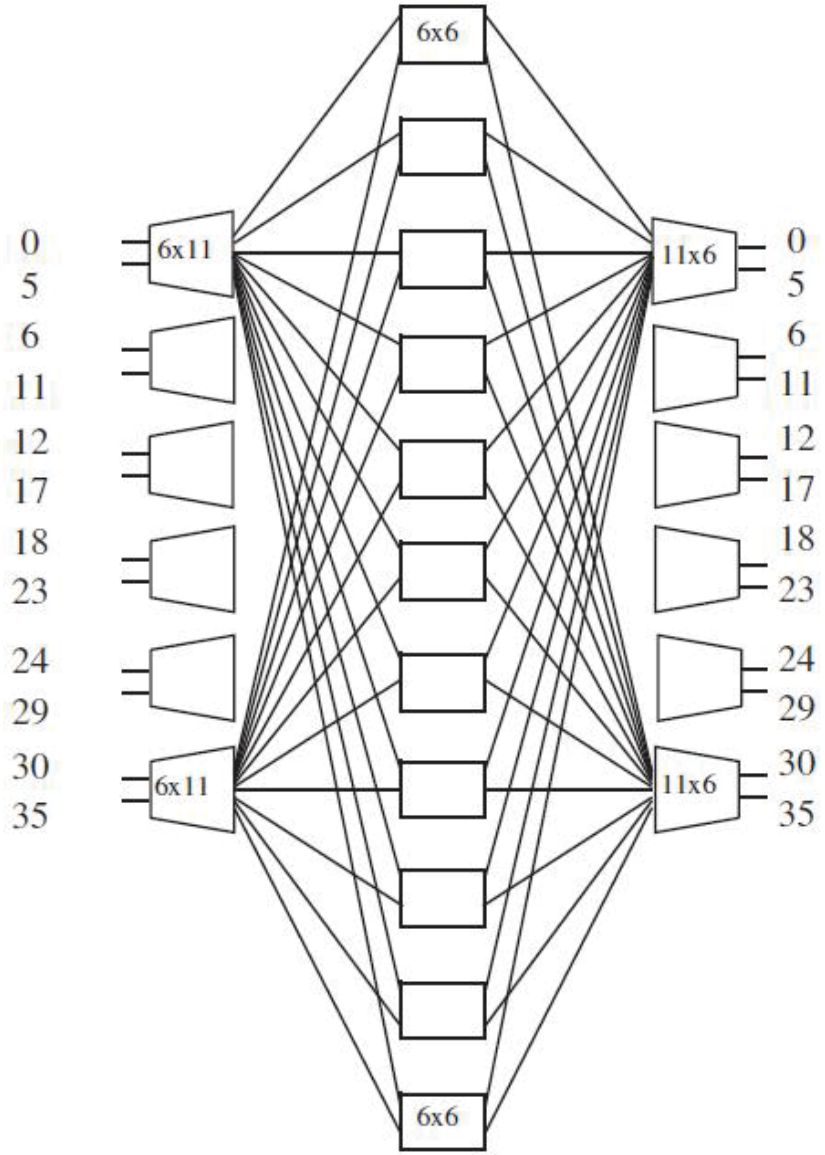


Clos Network - General Construction



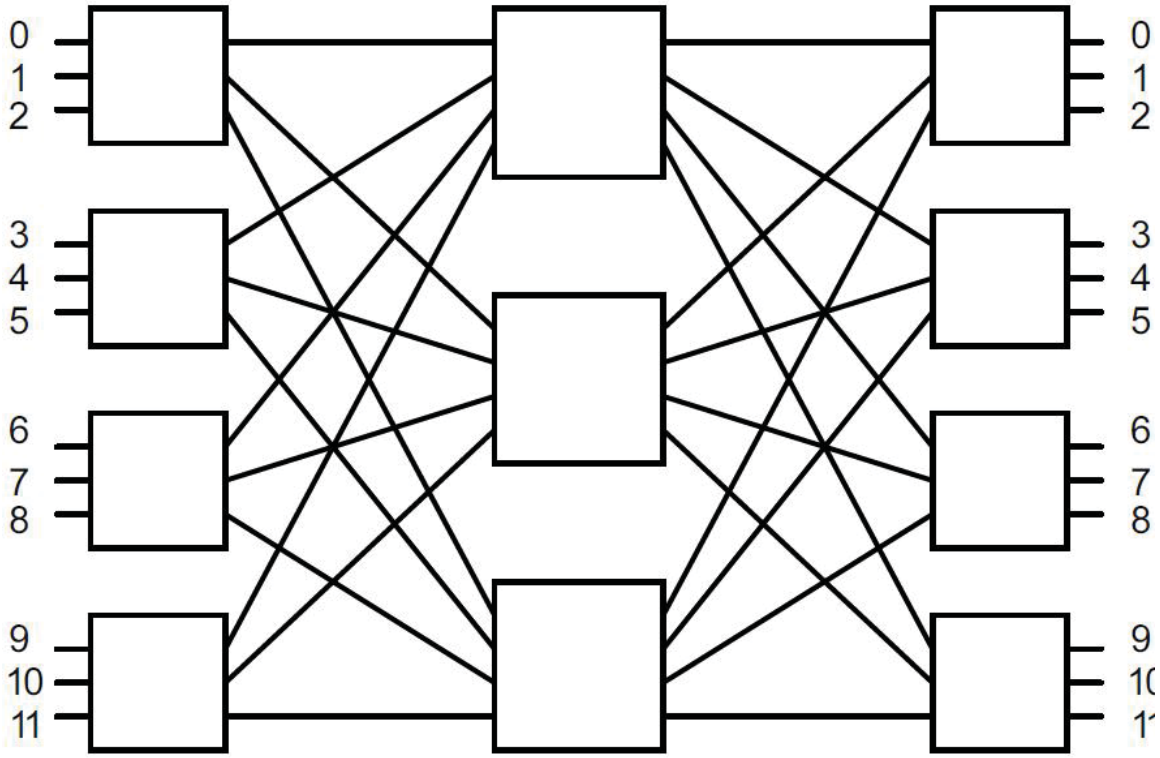
Clos Networks - Example

As an example, consider $n = 6$, $k = 6$, *and* $m \geq 11$.

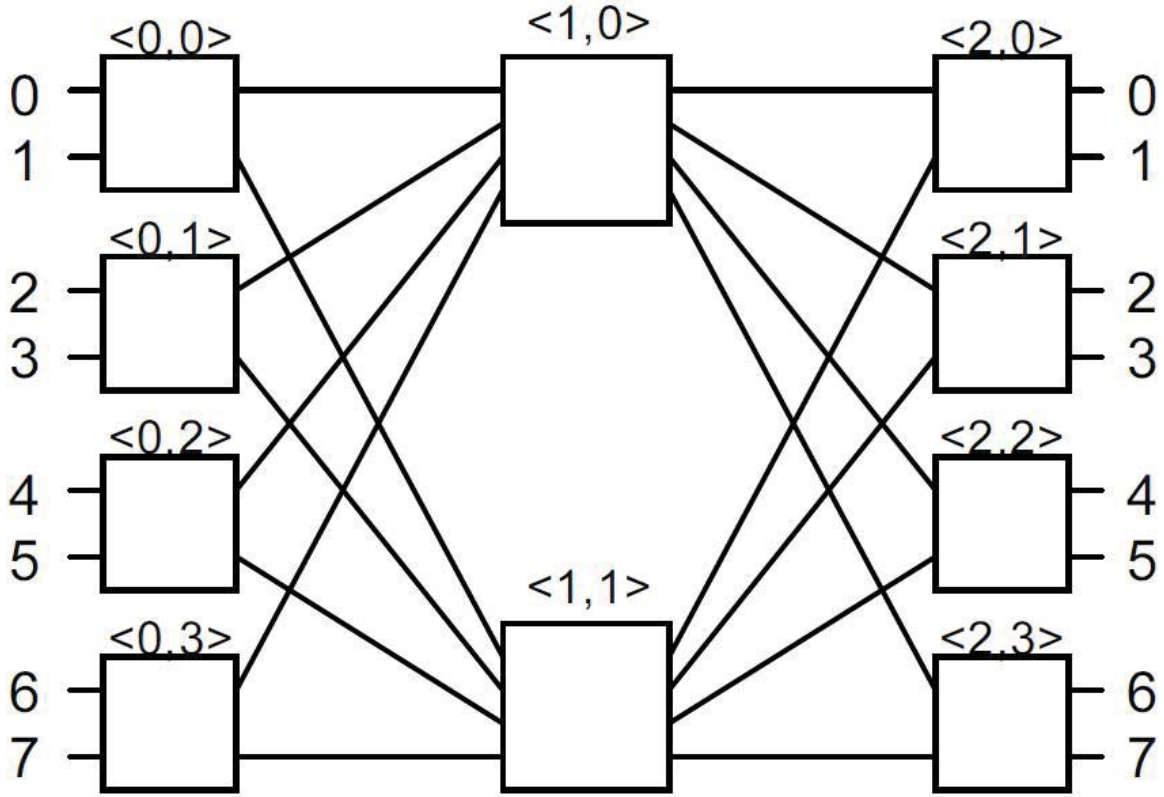


Clos Networks - Example

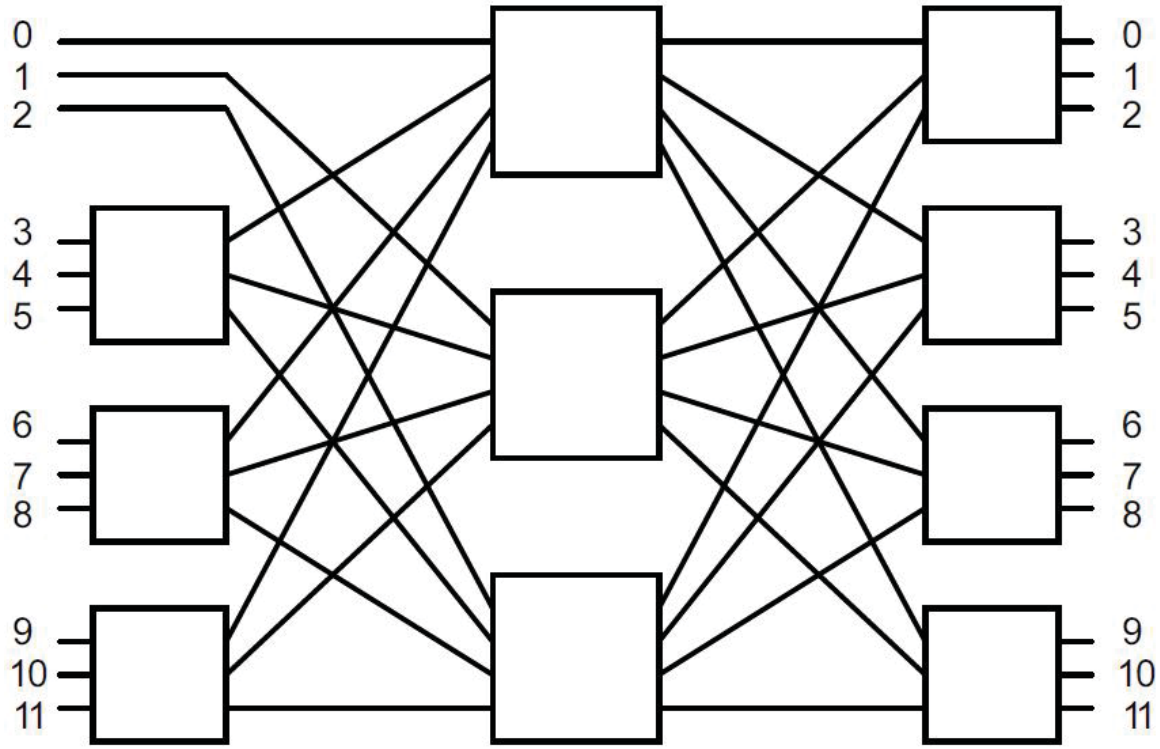
As an example, consider $n = 3$, $k = 4$, *and* $m = 3$.



Clos Networks - Example (Low Cost)



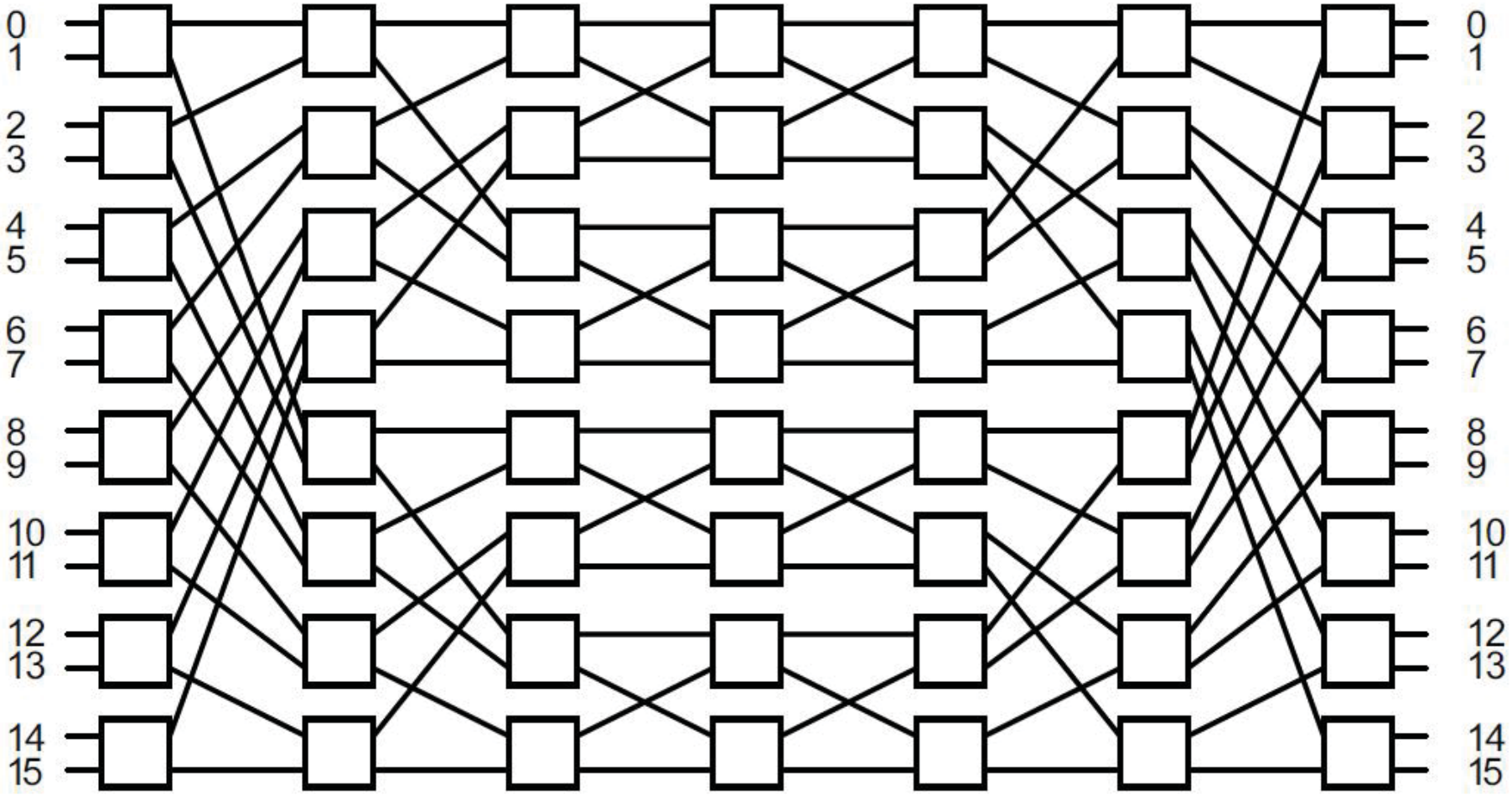
Clos Networks - Example (Low Cost)



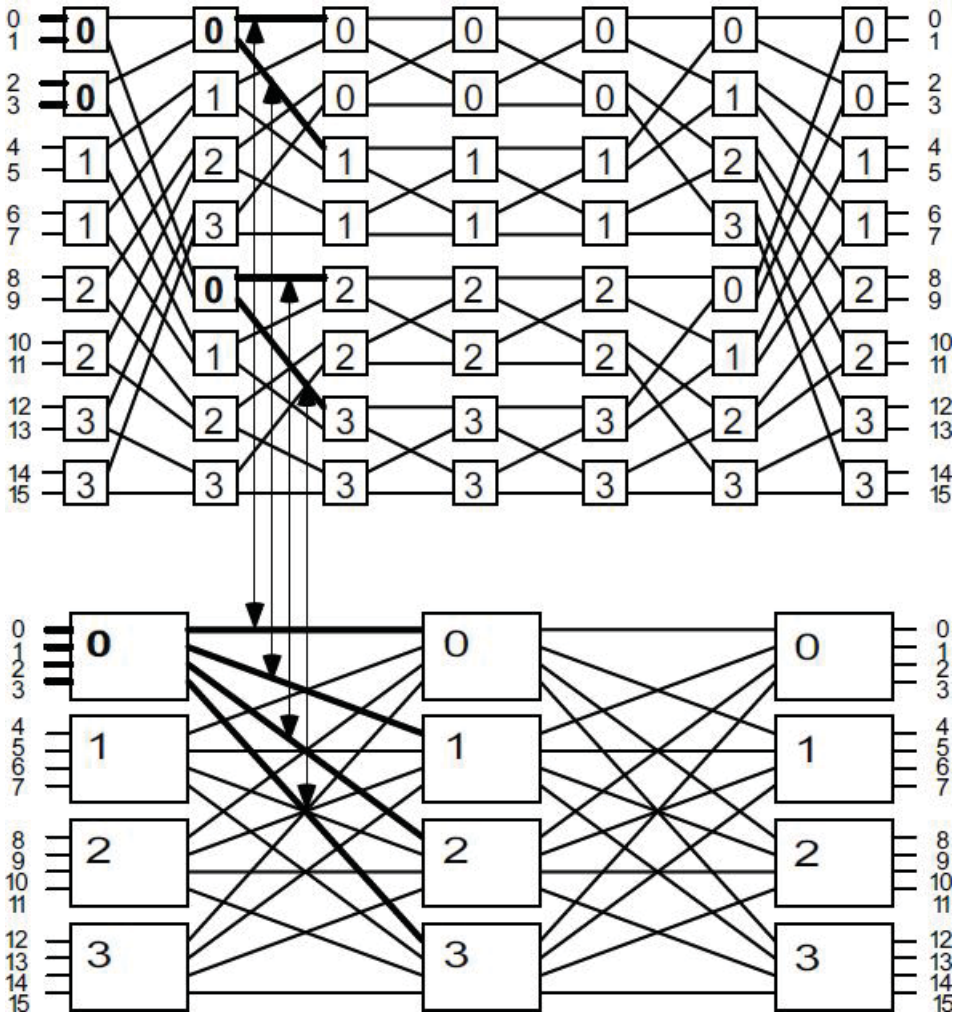
Clos Networks - Properties

- *Every 3-stage Clos network with* $m < n$ *is blocking. Some inputs may be lost.*
- *Every 3-stage Clos network with* $m \geq n$ *is rearrangeable. Beneš network obtained by selecting* $N = n \cdot r$*, and more often* $r = 2$*. The number of switches is* $O(NlogN)$ *which is asymptotically optimal.*
- *Every 3-stage Clos network with* $m \geq 2n - 1$ *is strictly nonblocking.*
- *Recursive extension to* $2t+1$*,* $t = 1,\ 2,\ \ldots$ *stages has* $O(N^{t+1})$ *crosspoints. Optimal choices for* n *satisfy* $N = O(n^{t+1})$*. Compare this with the* N^2 *crosspoints of an* $N \times N$ *crossbar.*

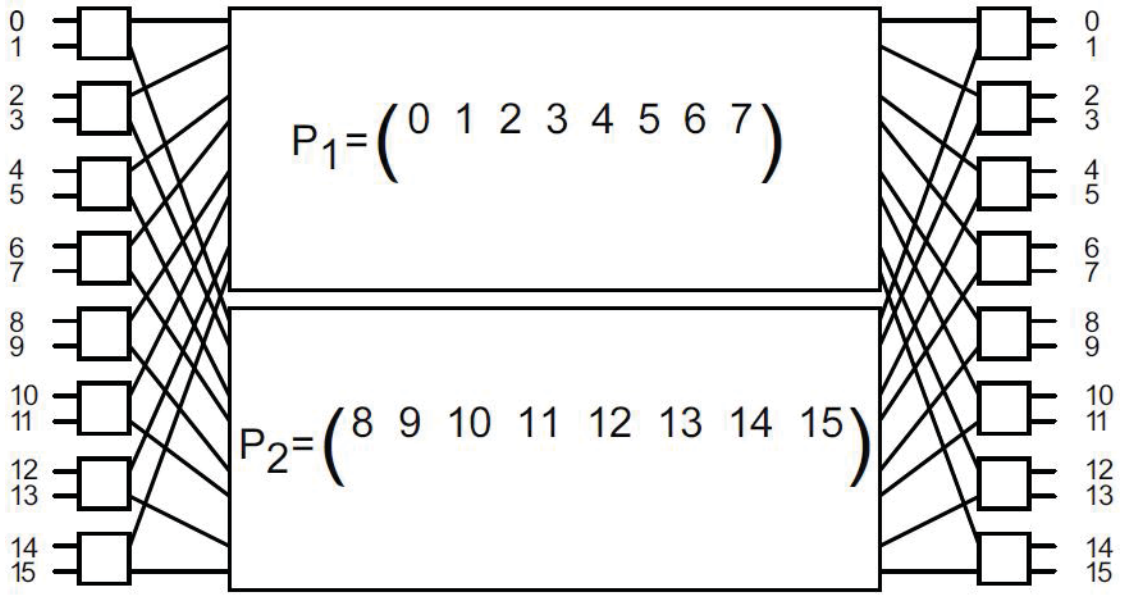
Benes Network - Rearrangeable



Benes Network - Rearrangeable/Large Switch

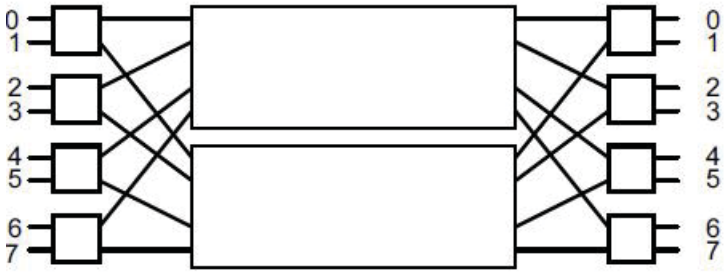


Benes Network - Waksman Algorithm (Recursive)

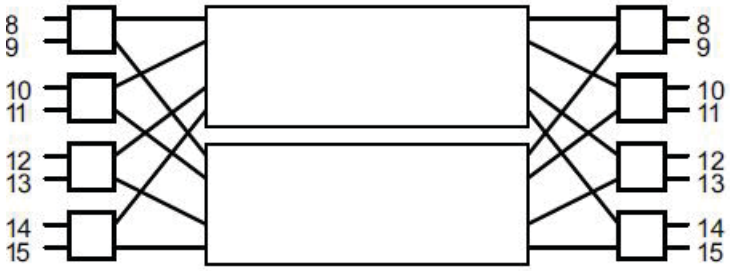


$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 5 & 15 & 12 & 7 & 3 & 0 & 8 & 11 & 9 & 6 & 13 & 14 & 10 & 2 & 1 & 4 \end{pmatrix}$$

LARGE Benes Network - Waksman Algorithm (Recursive)

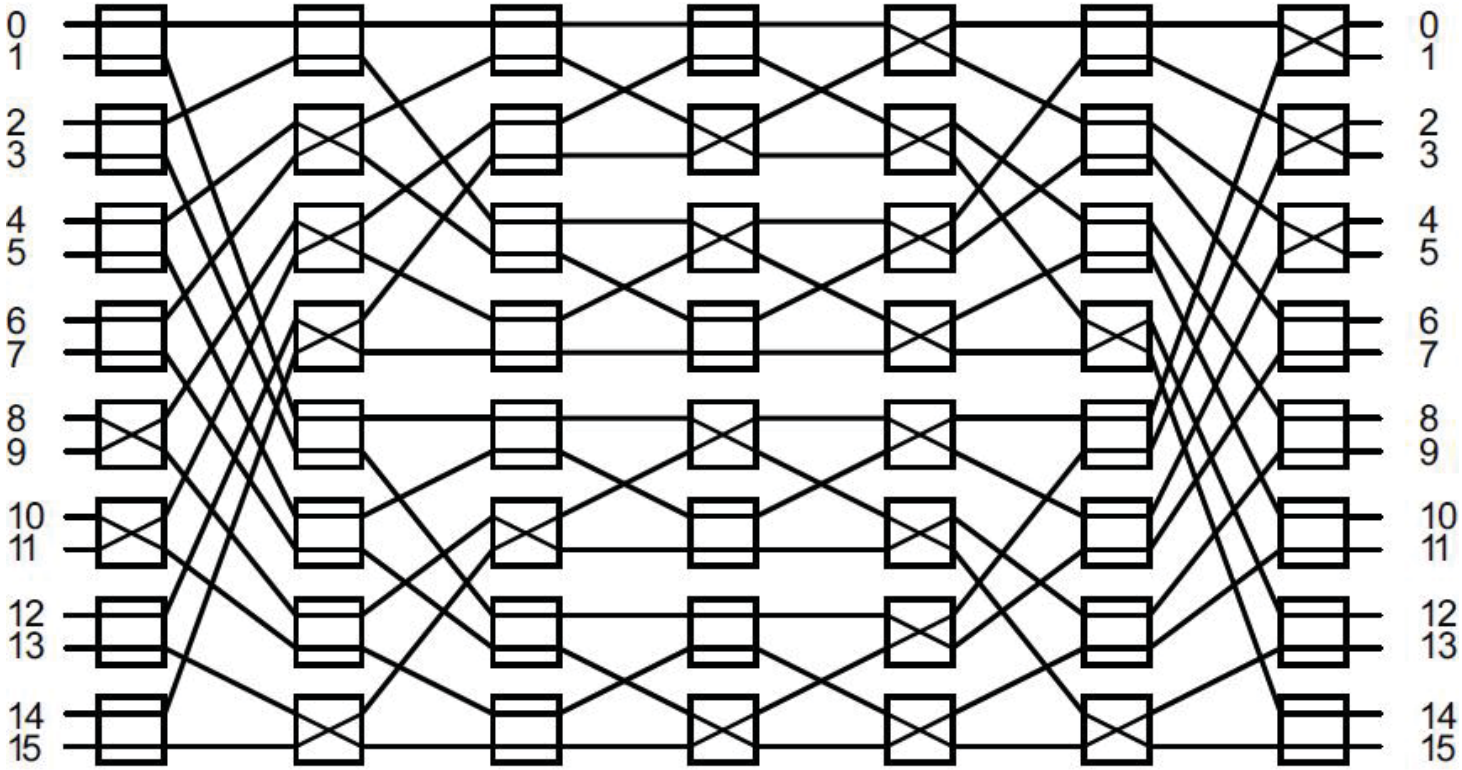


$$P_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 6 & 1 & 4 & 3 & 7 & 5 & 0 \end{pmatrix}$$



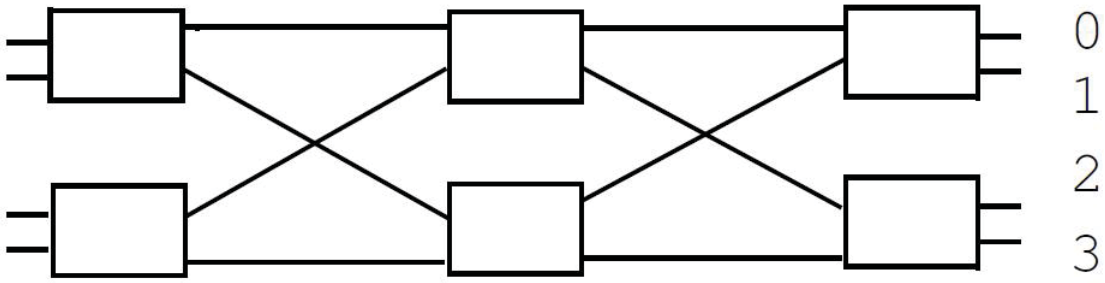
$$P_2 = \begin{pmatrix} 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 15 & 11 & 8 & 13 & 12 & 14 & 9 & 10 \end{pmatrix}$$

Benes Network - Waksman Algorithm (Final)

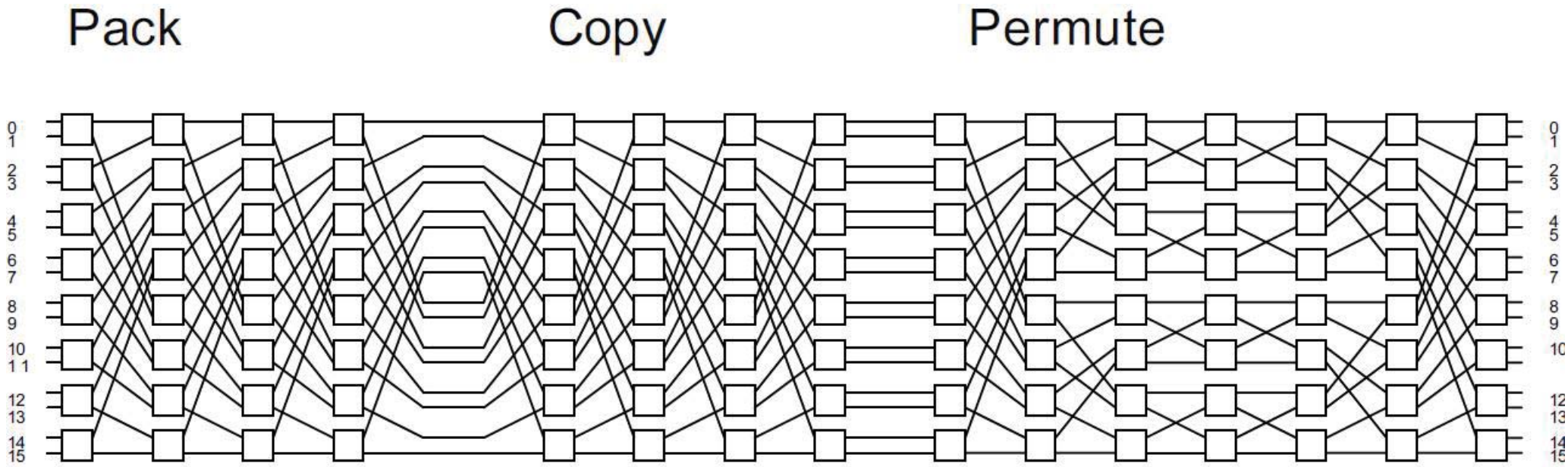


$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 5 & 15 & 12 & 7 & 3 & 0 & 8 & 11 & 9 & 6 & 13 & 14 & 10 & 2 & 1 & 4 \end{pmatrix}$$

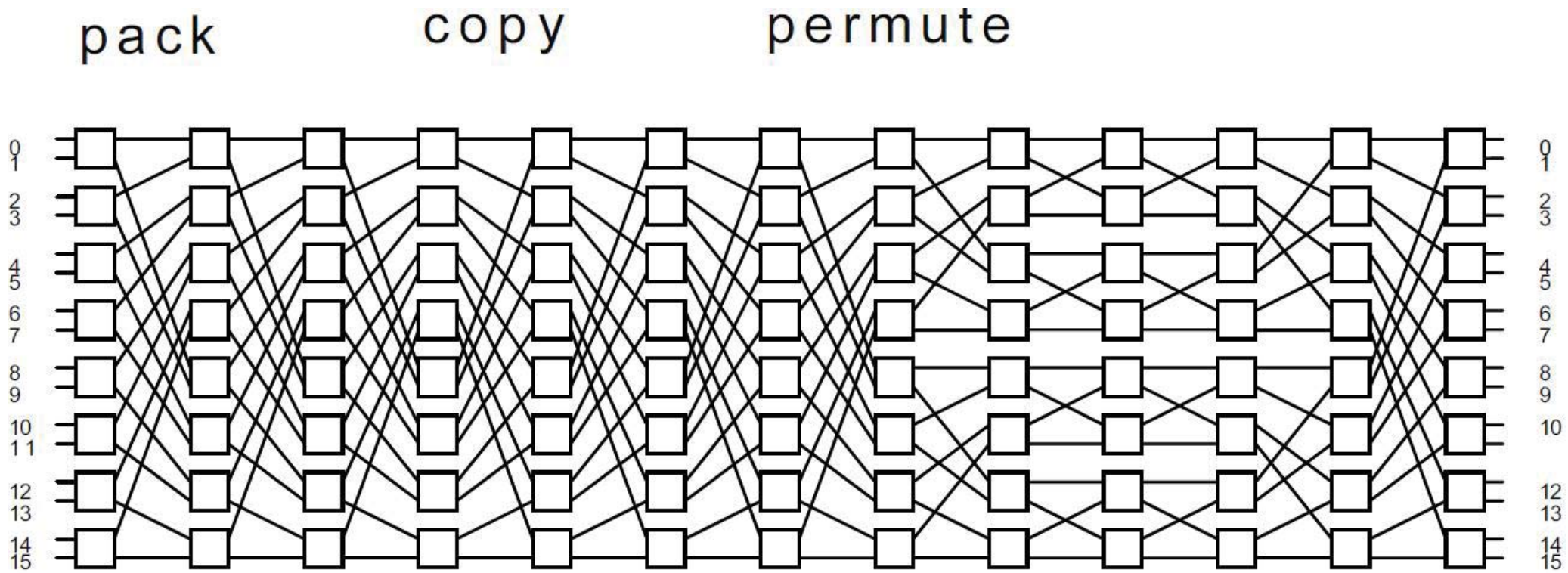
Strictly Nonblocking - Non-Clos



General Concetrator - Offman



General Concentrator - Offman (less cost)



Cost-Effective Network Scalability

1. **Realizability - Packaging**
   - *Topology and number of processors*
   - *Bisection bandwidth*
   - *Optical fiber between boards (clock speed)*
   - *Degree vs. pin-count (Technology)*
   - *Wire density limited by Technology*
   - *Memory bandwidth limits parallelism*
   - *Multilevel cache, multivalued memory*
   - *Multithreading, prefetching*
   - *Software overheads, e.g. context switch*

2. **Efficient Programmability**
   - *Data placement (graph embeddings)*
   - *Communication primitives (adaptivity)*
   - *Synchronization (atomic ops, combining)*
   - *Parallel applications (load balancing)*
   - *Software (operating system, compiler)*

3. **Serviceability**
   - *Structure regularity increases reliability*
   - *Fault recovery (checkpoints, diagnostics)*

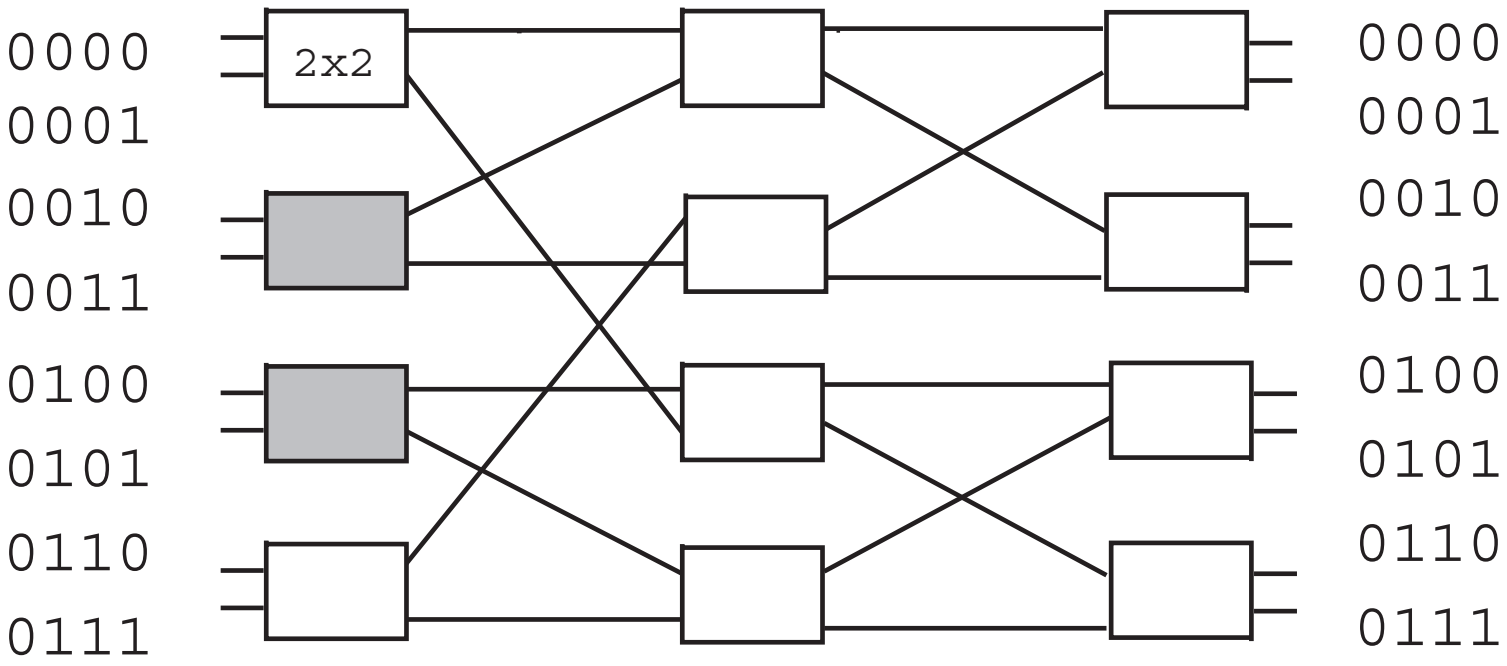
Scalability Concepts

- **Non-technical Use**
  1. *Enough sockets to plug processors?*
  -. *What about processor cards, performance?*
  2. *Linear speedup on application benchmarks?*
  -. *But what if constant of linearity is small?*

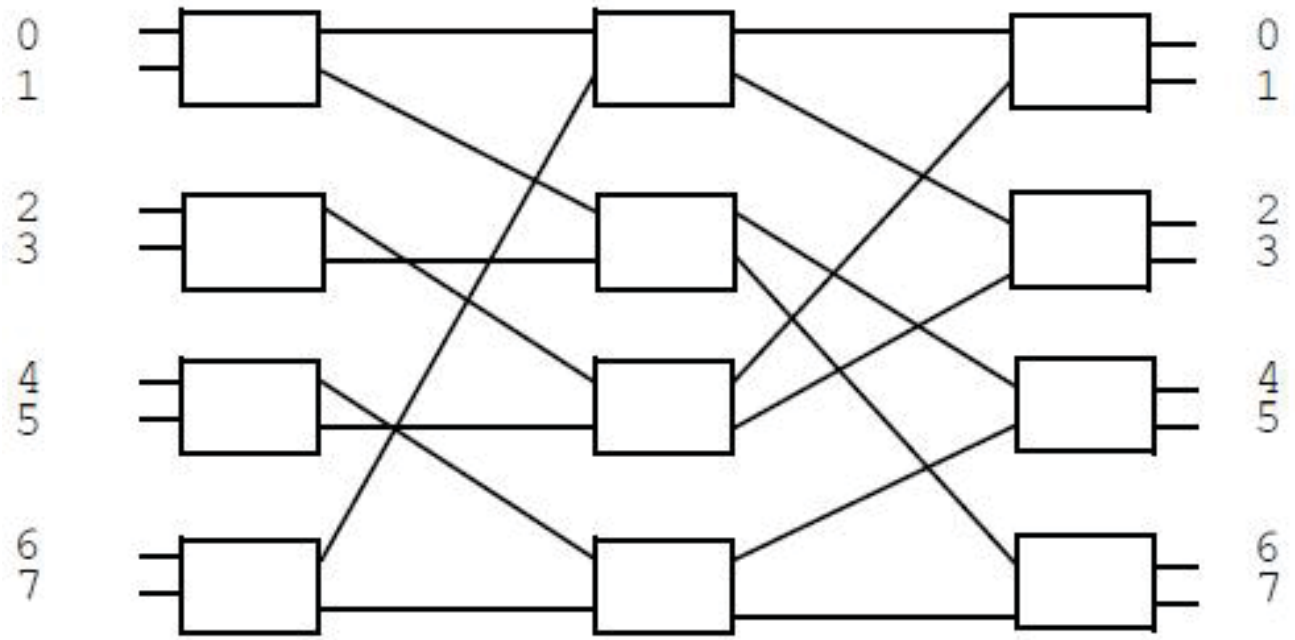
- **Computer Science Use**
  1. *System can grow by replication (e.g. 100-fold)*
  2. *Cost scales well with network size (as $N \cdot logN$)*
  3. *All aspects of system performance scale (load)*

- **Business View - Investment Protection**
  1. *Anticipated business growth*
  2. *Safety factor (new application demands)*

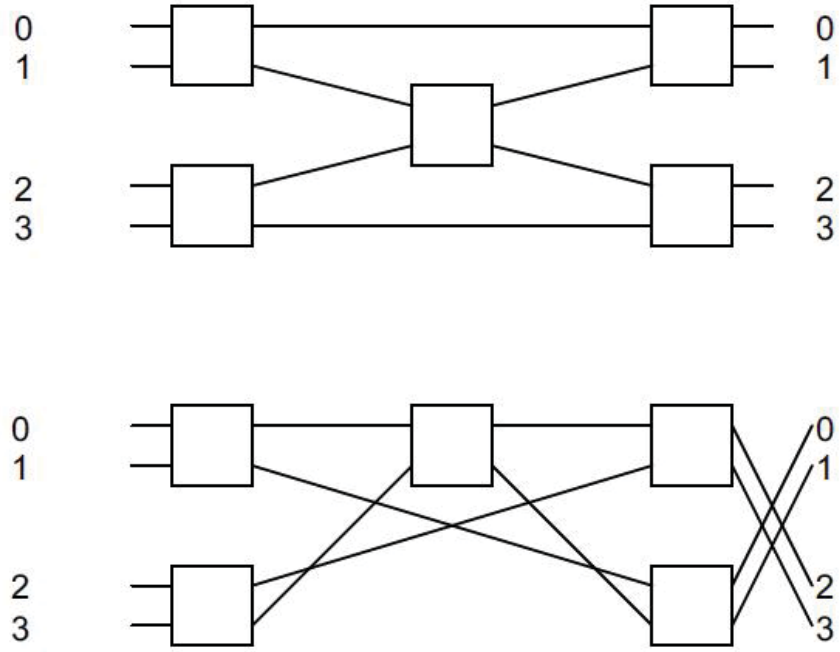
Problem with this Network?



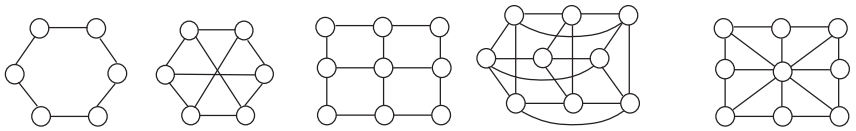
Properties of this Network (MFA)?



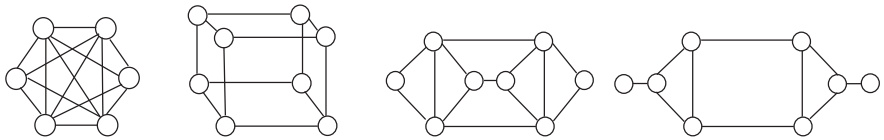
Topological and/or Functional Equivalence?



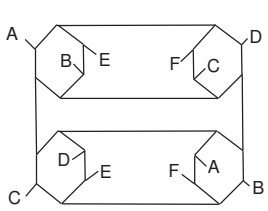
Static and Dynamic Networks: Analytical Models (Network Calculus)



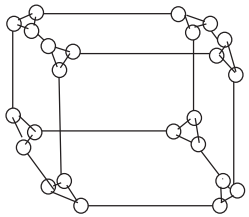
Cycle Chord Graph 2-d Mesh 2-d Torus Diagonal Mesh



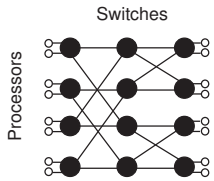
K6 Binary Hypercube Binary De Bruijn Shuffle Exchange



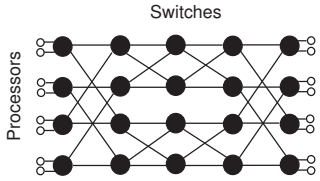
Star Graph (N=4!)



Cube-Connected Cycles

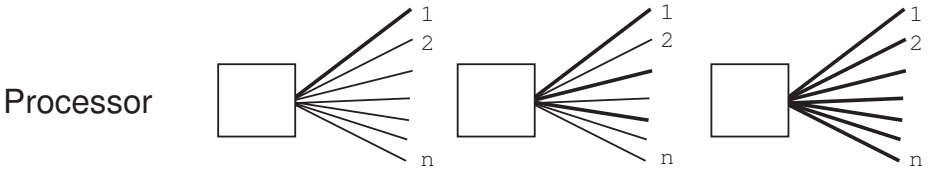


Butterfly Network



Benes Permutation Network

COMMUNICATION MODELS



- 1-port (SIMD)
- k-port (k=3) (DMA-bound)
- All-port (MIMD)



- Half Duplex vs Full Duplex

- Constant Delay vs Linear Delay

  T=1 unit

  $T = Ts + L\,P$

  Ts = Startup
  L = Link bandwidth
  P = Packet size excl. header

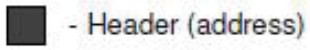
  VLSI model. Time vs. chip-area tradeoffs.

- Routing Mode

  store-forward, circuit switching,

  wormhole, vct, mad-postman strategy

Routing Mode



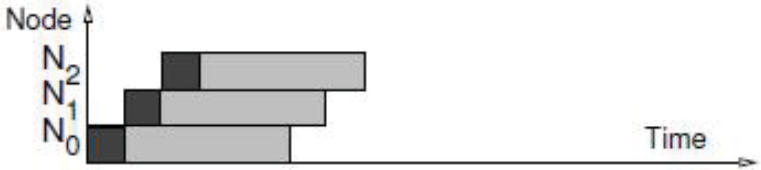
- Store-Forward (telegram, early computers)
  - No established path between source and destination



- Circuit Switching (telephony: low utilization, long message/path)
  - Establish path, acknowledge, transmit, remove path



- Wormhole Routing (fast inexpensive routers, small buffers)
  - When head moves entire packet moves (worm-like)



- Virtual Cut Through
  - Entire packet can pile up at a node (enough queuing)
- Mad-Postman Strategy for Virtual Networks (2D mesh/torus)
  - Packet sent to adjacent node in VN before header received

Performance of Routing Models

Let L be the packet size in flits, W the flit size in bits, T the time to transmit one bit to a neighbor, D the channels traversed, and R the transmission rounds (circuit switching).

Routing Mode	**Latency**
Store-Forward	$L \cdot W \cdot T \cdot D$
Wormhole Routing	$W \cdot T \cdot (L + D)$
Virtual Cut-Through	$W \cdot T \cdot (L + D)$
Circuit Switching	$\approx c_1 R \cdot T + c_2 \cdot L \cdot W \cdot D$
Mad Postman	$L \cdot W \cdot T + k \cdot T \cdot D$

For virtual cut-through no deterioration for blocked messages (unlike wormhole). *Performance efficiency depends on packet size (L) vs. network diameter (D).*

Dead address flits in mad postman strategy cause traffic overhead. *Normally $0 < k \leq 1$, while $k = 1$ for synchronous transmission.*

Mad Postman vs. Virtual Cut-Through

- *Transpose* 32×32 *mesh, packet=27bits, data=8bits.*

Routing Mode	**Latency (steps)**
Virtual Cut-Through	$(570, 664)$ *for* $(25, 100)\%$ *sparsity*
Mad Postman (min 89 steps)	$(154, 664)$ *for* $(25, 100)\%$ *sparsity*

Network throughput is more important?

Deflection Routing

In **hot-potato routing** *(called deflection routing) there are* **no buffers at intermediate nodes**. *Packets continue to move, possibly towards a wrong direction, until they ultimately reach their destinations.*

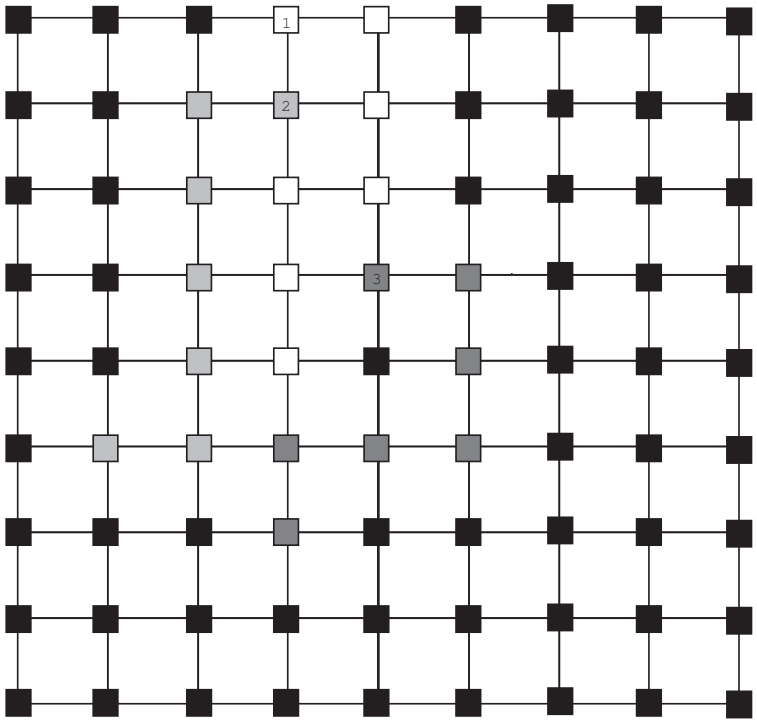
Deflection routing is used in optical networks!

Optical delay lines are expensive, and opto-electronic conversion for electronic buffering is slow.

COMMUNICATION ALGORITHMS

- Online Algorithm (communication pattern known on the fly)
  - Random pattern
  - Common pattern (BPC, Omega permutation, Broadcast)
- Off-line Algorithm (communication pattern known in advance)
  - Precompute the 'best' route for packets
- Oblivious - Adaptive - Randomized Algorithm
  - Oblivious (a packet route is independent of other packets)
  - Randomized algorithm routes all packets with high probability
    Sherwood- type randomization
- Deadlock Free Routing
  - Acyclic channel graph is sufficient to avoid deadlock

ROUTING ALGORITHMS

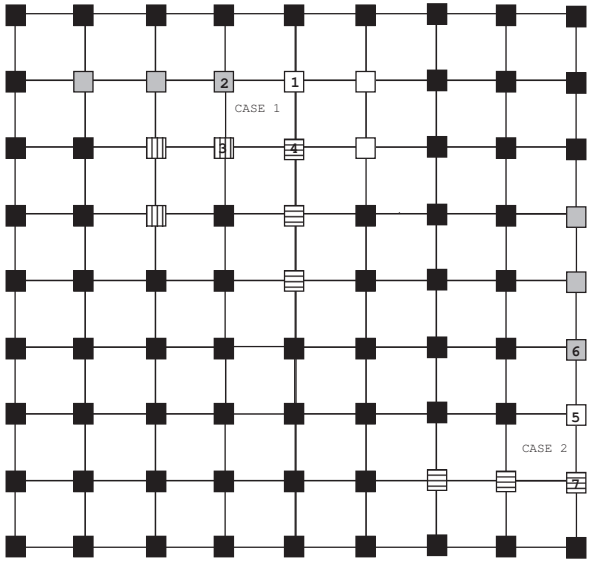


OBLIVIOUS ROUTING - PATH DOES NOT DEPEND ON REMAINING ROUTES

RANDOMIZED ROUTING - PACKET MAKES RANDOM CHOICES (Las Vegas)

ADAPTIVE ROUTING - PATH DEPENDS ON THE TRAFFIC

Deadlock - Livelock - Starvation



1. Deadlock is a cyclic dependency of ungranted requests for buffer/channel resources. Example: case 1, with clockwise routing and (1,3) (3,1) (2,4) (4,2) source-destination pairs. Routing should not introduce deadlock hazards. Also parallel programmer's responsibility.

2. Livelock: packet never reaches its destination.

3. Starvation: processor never handles incoming/outgoing packet. Causes serialization. Example: case 2, packet 7 never reaches its destination 5 because packet 6 is always preferred.

Deadlock-Free Routing

Store-Forward - Virtual Cut-Through

Define partial order on buffer classes. Use structured buffer pool assignment to packets. Need long buffers and suffers from long delays. Flow control per packet.

Deterministic Wormhole Routing

Dimension order routing. Examples: E-cube, (X,Y) mesh-routing, etc. No fault tolerance and poor worst-case performance. Flow control per flit.

Deadlock-Free Adaptive Wormhole Routing

Partition physical channels C_i to virtual channels (VC) C_{ij}. Adaptivity is defined by $R \subset VC \times destination \times VC$. Let the channel dependency graph $G = (C_{ij}, E)$, where E contains an edge when there is a link from C_{ij} to C'_{ij}.

Deadlock-free routing corresponds to a cycle-free channel dependency graph.

Techniques for Deadlock Avoidance

1. Use dimension-reversal numbers (0 ... r). When the new VC is smaller we increment the dimension reversal number. When it is equal to r, perform dimension order routing.

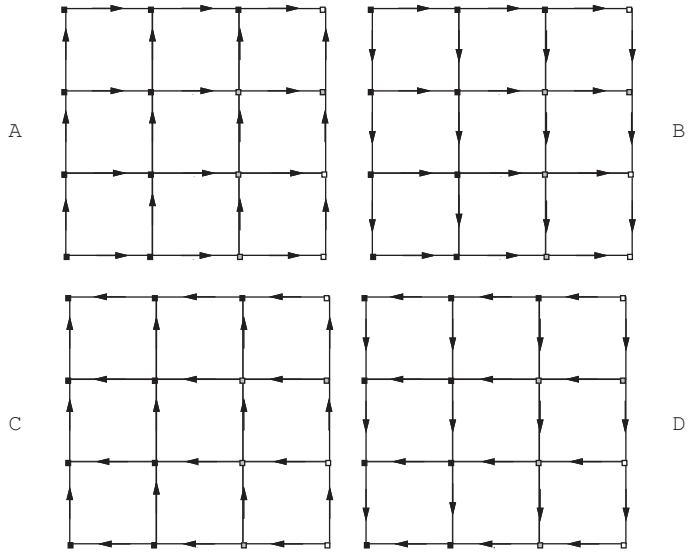
2. Allocate adaptive and deterministic virtual channels. Initially all packets use adaptive channels. Each packet reverts to deterministic routing whenever it must wait for a "smaller" virtual channel.

3. Use virtual networks to break cycles. Packets are divided depending on their destination to different networks. One buffer is allocated per each virtual network at each node. Examples: Mesh, Torus, etc (long diameter networks).

4. Tear off VCs for all packets which have not reached their destination within a given time delay. This method requires padding packets traveling a distance smaller than the network diameter. Test packets can use flow control paths.

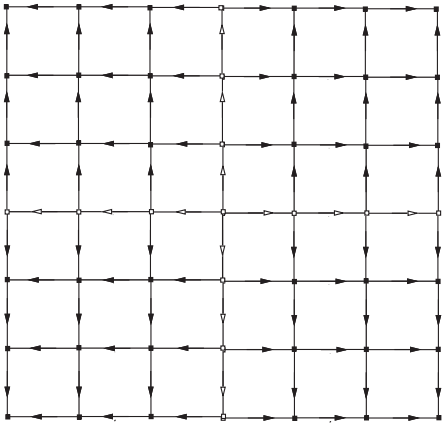
5. Avoid turns by using west first, north last or negative first routing strategy. Virtual circuits are not needed.

Virtual Networks in Mesh & Torus



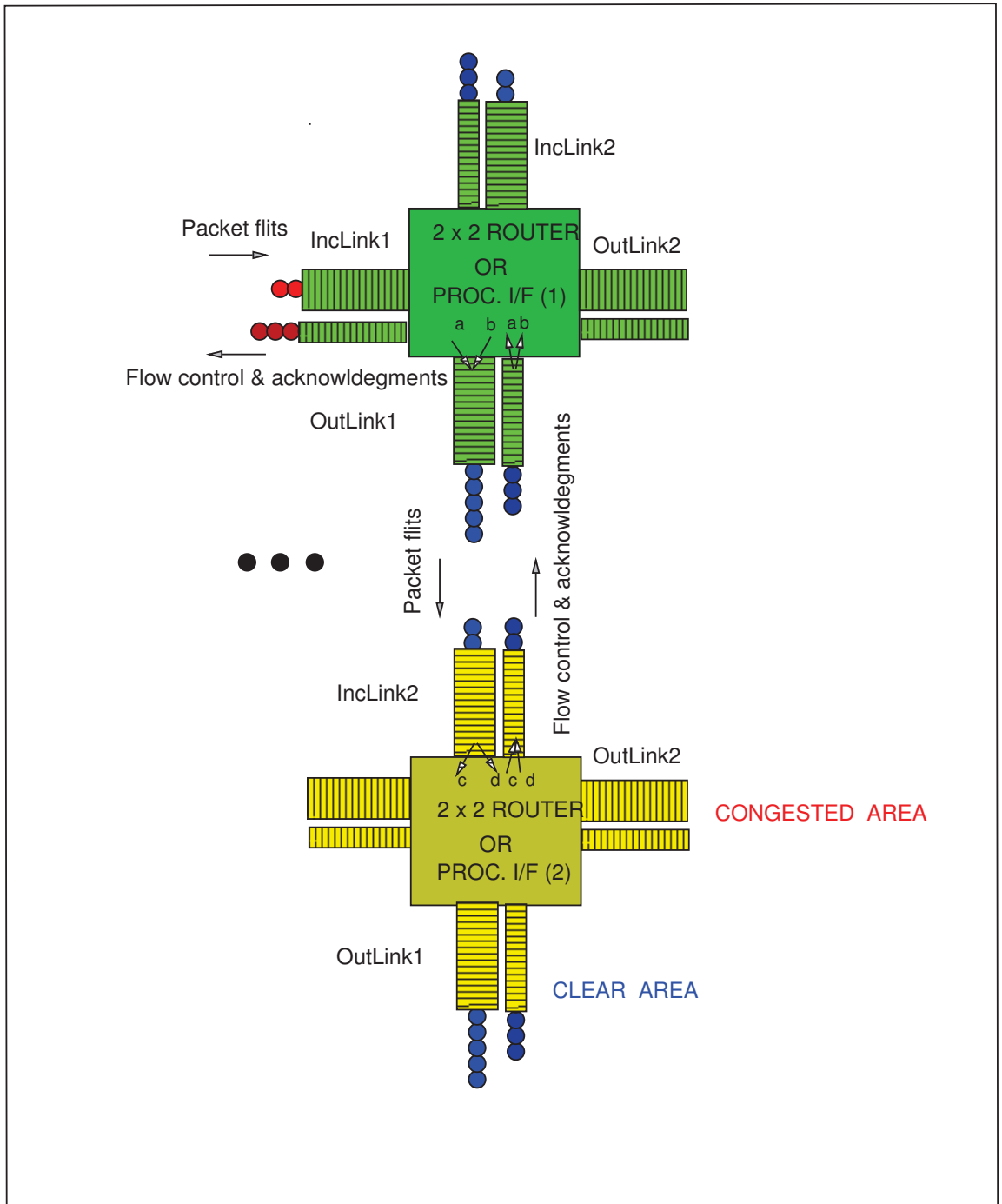
1. *For 2-dimensional mesh, virtual networks B&C (or A&D) can be combined without introducing any cycles. Thus, only 2 buffers per node are needed.*
2. *For d-dimensional mesh, similar technique yields 2d buffers per node.*
3. *For d-dimensional tori we can use shortest path routing within the appropriate mesh Virtual Network.*
3. *Separate channels necessary for request & replies.*

Virtual Networks for Multicast



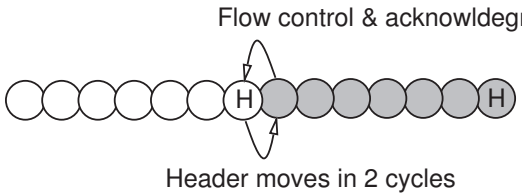
Multicasting using (X-Y) routing is deadlock free! Partition the physical network into deadlock-free subnetworks and multicast using minimum spanning tree.

USING VCs TO AVOID HEAD-OF-LINE BLOCKING

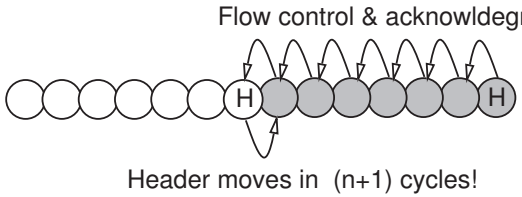


WHEN DOES THE HEADER MOVE?

DING & BHUYAN: MOVE ONLY IF NEXT BUFFER IS NONFULL



JENQ'S MODEL: MOVE ALSO IF NEXT BUFFER FULL AND SIMULTANEOUSLY PACKET TRANSMITTED OUT OF THIS BUFFER



FOR LONG PACKETS AND PATHS THE FLOW CONTROL PROPAGATION TIME BECOMES SIGNIFICANT MAKING JENQ'S MODEL INEFFICIENT

Performance Measures

- **Switch throughput S: average # packets routed through the output port of the switch per clock cycle.** *For switches without replicated/dilated ports, S is the probability that an output port is active.*
- **Switch latency L: average packet delay through the switch, measured from header arrival to departure time of the last packet bit.**
- **Network throughput S': average # packets routed through an output port of a last stage switch per clock cycle.**
- **Network latency L': average packet delay, measured from header arrival to departure of last packet bit from the network.** *Sum of switch latencies plus processor input buffer arbitration time.*

Network throughput is normalized by maximum possible throughput (bisection bandwidth).

Bandwidth & Latency Losses

- **Maximum link bandwidth.**
- **Acknowledgments and/or flow control packets consume bandwidth.** *For the STC104 a) packets and flow control share unidirectional links, while b) packets, ack and flow control share bidirectional links.*
- **Congestion consumes link bandwidth.**
- **User-space to user-space measured bandwidth (software overhead - context switch).**

SP1 maximum link bandwidth 40-50 MB/sec.
Sustained bandwidth ≈ 5 MB/sec.

SP1 node to node hardware latency 1 μ sec.
Application to application latency 30 μsec.

Network bw & latency loss can be severe!

Ways to Hide Network Latency

- **Overlap communication and computation.**
- **Multithreading.**
- **Lower cache miss ratio, weaker consistency model.**
- **Virtual processors.**
- **More computations per node.**
- **Data redundancy.**

No way to hide network throughput!

Earlier S'/L' used as performance metric. Nowadays, after PRAM, X-PRAM, and BSP, the latency overhead model ($\log P$) defines gap G as the minimum time interval allowed between injecting two packets into the network; gap is related to throughput.

$$G = \frac{N_{proc}}{S'}$$

Better performance metrics are application driven.

Packet Fields



- **Header.** *Destination and sometimes source address. Data length for variable size packets. Other path-select information.*
- **Data.** *Data and/or memory address (virtual memory). Performance related information: transaction ID and type, epoch counters. Ack vs. non-ack, blocking vs. non-blocking requests. Switch control and testing.*
- **CRC.** *Header and data error detection/correction bits.*
- **Packet termination** *for variable size packets.*

Primitive Instructions

Lowest level parallel instruction primitives.

- *Remote read or DMA. Header, address and CRC.*
- *Reply to remote read or DMA. Header, data and CRC.*
- *Remote write. Header, address, data and CRC.*
- *Reply from remote write. Header and CRC.*
- *Synchronization primitives. Header, address, data & CRC. Fetch & add, compare & swap, fetch & store.*
- *Reply from synchronization. Header, data & CRC.*
- *Performance-related. E.g. Remote enqueues.*

Higher level software tools.

- *I/O drivers.*
- *Distributed shared memory (ShMem).*
- *Standard message passing interface (MPI).*
- *Parallel languages (Cuda, Star-P, F-90, HPF, C*, FORTRAN D, CRAFT, MP Fortran, etc).*
- *Clustering services, multicore operating systems.*

Priority Messages

- *Some packets may be labeled as high priority packets by setting a specific bit in the packet header.*
- *By providing separate buffers for different priority packets it is possible to allow a network to select high priority packets to bypass the buffers and thus get delivered first.*
- *In a simple non-preemptive implementation, additional buffers (for each priority level) are needed to store high priority messages when several high priority packets arrive are destined to the same output port.*

Network throughput is normalized by maximum possible throughput (bisection bandwidth).

Router Design - Buffer Space

- **Retransmissions vs. back-pressure.** *Either a cell-loss probability model controls buffer overflow (ATMs), or all traffic directed to a full buffer is stalled (tickets). With packet retransmissions, in order packet delivery from the same source (transaction completion) is not guaranteed.*
- **Single multibuffer vs. fully-connected buffer.** *When packets arrive, can simultaneous data buffer writes occur?*
- **Multiport vs. nonmultiport buffer memory.** *When packets depart, can simultaneous data buffer reads occur?*
- **Statically vs. dynamically allocated buffer.** *Can each allocated buffer serve multiple input streams?*
- **Is the switch fair to all packets? Is head-of-line blocking avoided.** *Use of virtual circuits, one for each source-destination pair.*

Remarks - Router Design

The effect of nonmultiport buffers on performance *(due to nonsimultaneous transmission of packets arriving at the same time and leaving from different output ports)* **can be minimized.**

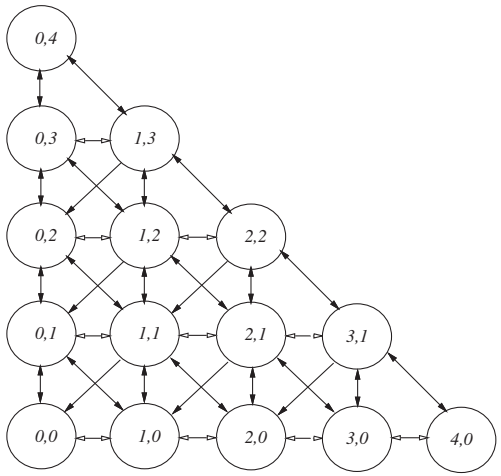
Input buffering saturates at 0.58 of maximum throughput for both the retranmission, and back-pressure models.

A dynamically-allocated fully-connected switch has the highest throughput. *However, building such an $n \times n$ switch, requires complex hardware data structures (pointers), especially for $n > 2$.*

Network Simulation Assumptions

- **Continuous routing.** *Dynamic traffic to random destinations with constant generating and hotspot rate.*
- **E-cube shortest-path routing.** *Dimensions are checked in increasing order.*
- **Similar buffer capacity per switch.** *For the* 128*-node hypercube, each SIDB* buffer holds $128^2/8^2 = 256$ packets, since the *CDB* switch provides 128^2 virtual circuits (one per each source-destination pair).
- **Fairness.** Sending 9 flits out of the 9-flits-per-packet *CDB* switch takes as much time as sending one flit out of the one-flit-per-packet *CDB* (or *SIDB*) switch.
- **Sample size.** Repeat experiments, increasing the time window ($\approx$ 5000 clock cycles), until steady state is reached.

Dynamically-Allocated Fully-Connected Buffer



Input ports utilize **more than** *their share of total buffer space.*

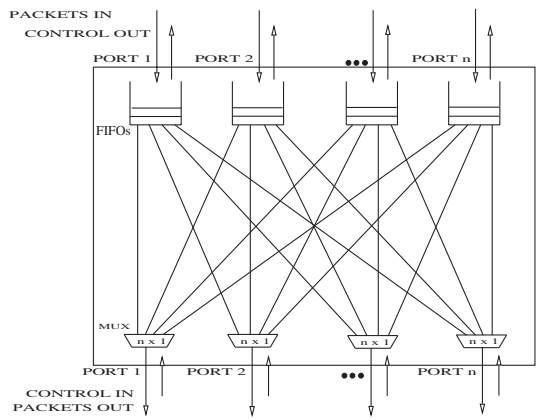
Application to network interface design.

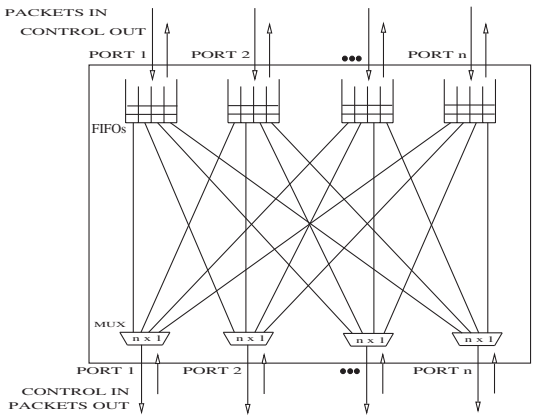
Memory, processor and router share a bus.

Memory and processor (integrated router) share bus.

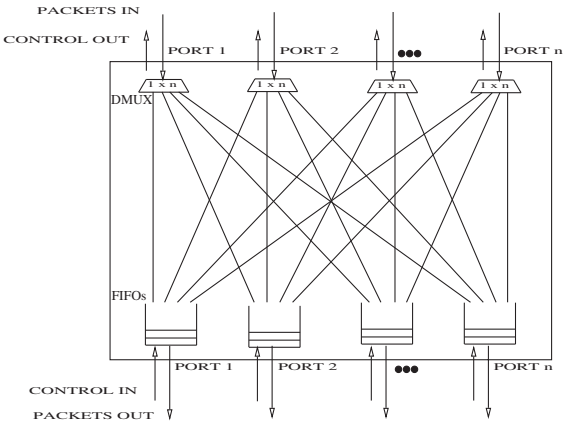
Processor and memory (integrated router) share bus.

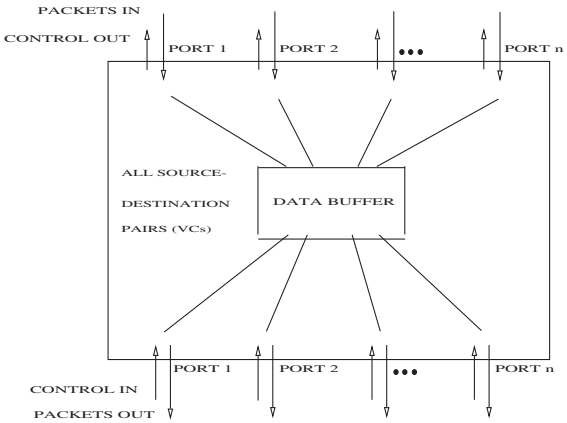
Buffer Allocation for Switches





A 4×4 input buffer switch and a 4×4 split input buffer switch.





A 4×4 output buffer switch and a 4×4 central buffer switch.

Router Model Assumptions

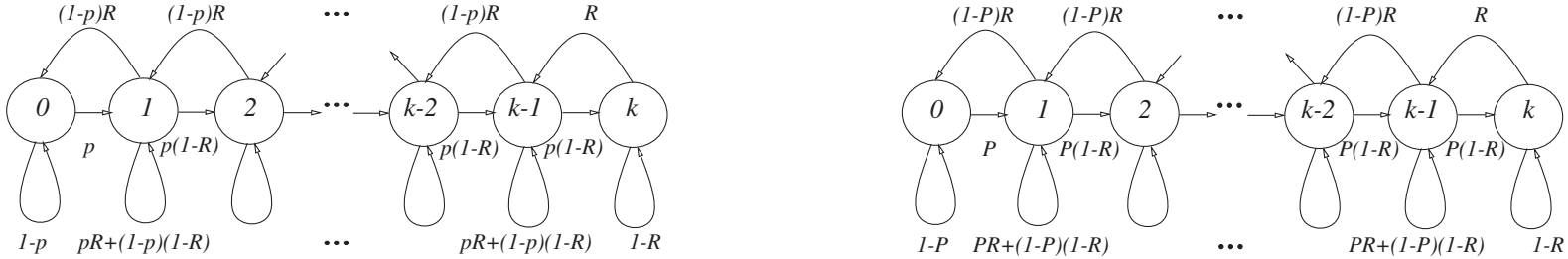
- **Fixed size packets.**
- **Packet transmitted only if there is free buffer at subsequent stage.**
- **Switch startup delay time for buffering/prerouting is one time unit.**
- **Similar buffer storage capacity.**
- **Ignoring software/synchronization overhead.** *Latency and throughput are not computed from user-space to user-space.*

Performance Model for Packet Routers

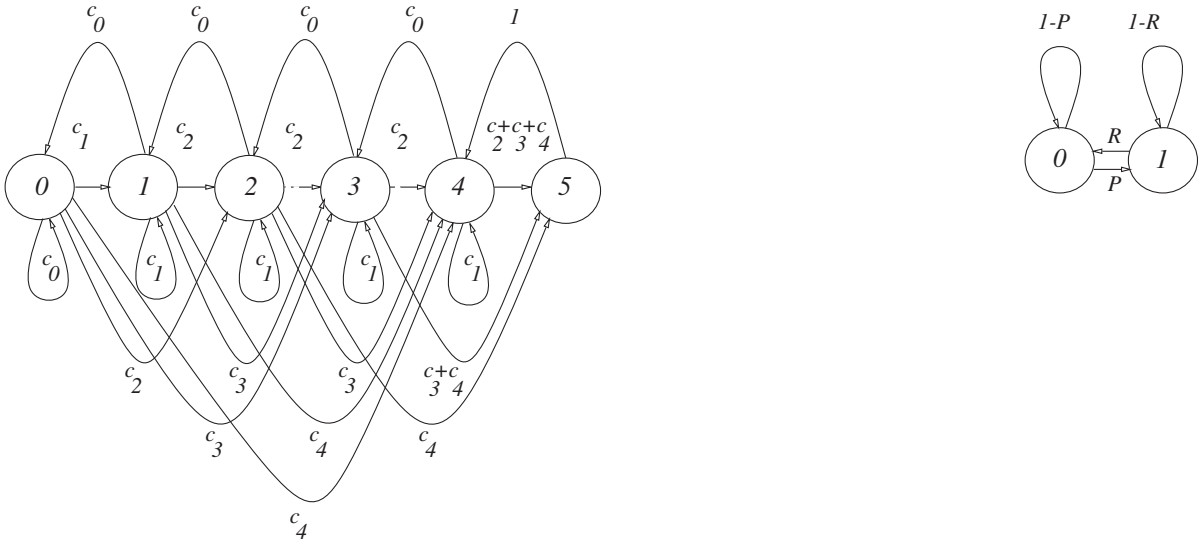
Example: Continuous packet routing.

- **Initially all buffers are empty.**
- **Constant packet generating rate p applied at each input port.**
- **Packets headed to random output ports.**
- **Input rate p reduced when buffers become full (may drop to zero).**

Markov Models for Queue Size



Transitions for a) IDB $n \times n$, and b) SIDB $n \times n$ switches.



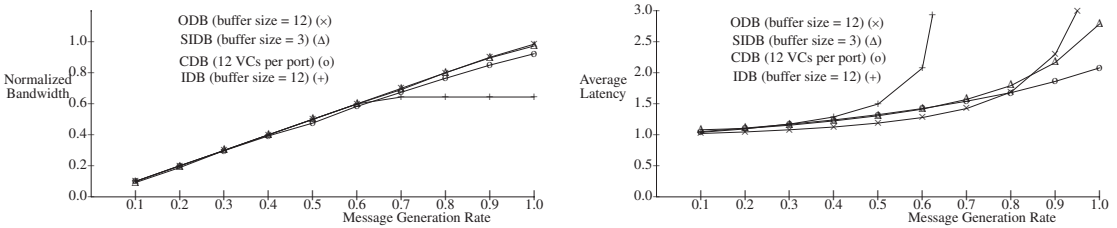
Transitions for a) ODB 4×4 ($k = 5$), and b) CDB $n \times n$ switches.

Switch Throughput and Latency

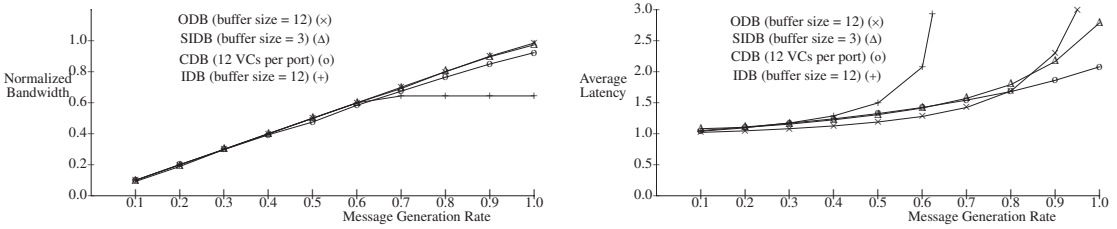
Input Buffering	Split Input Buffering
$S = 1 - (1 - \frac{1-\pi_0}{n})^n$ $R = \frac{S}{1-\pi_0}$ $L = \frac{1-\pi_0}{R \cdot \sum_{j=1}^{k} \pi_j / j}$	$S = 1 - \pi_0^n$ $R = \frac{S}{(1-\pi_0)n}$ $L = \frac{1-\pi_0}{R \cdot \sum_{j=1}^{k} \pi_j / j}$
Output Buffering	**Central Buffering**
$S = 1 - \pi_0$ $L = \frac{1-\pi_0}{\sum_{j=1}^{k} \pi_j / j}$	$S = 1 - {\pi_0}^{\lambda}$ $R = \frac{S}{(1-\pi_0)\lambda}$ $L = 1/R$

The above R is the probability that a packet at the front of its buffer moves forward.

Switch Performance Comparisons

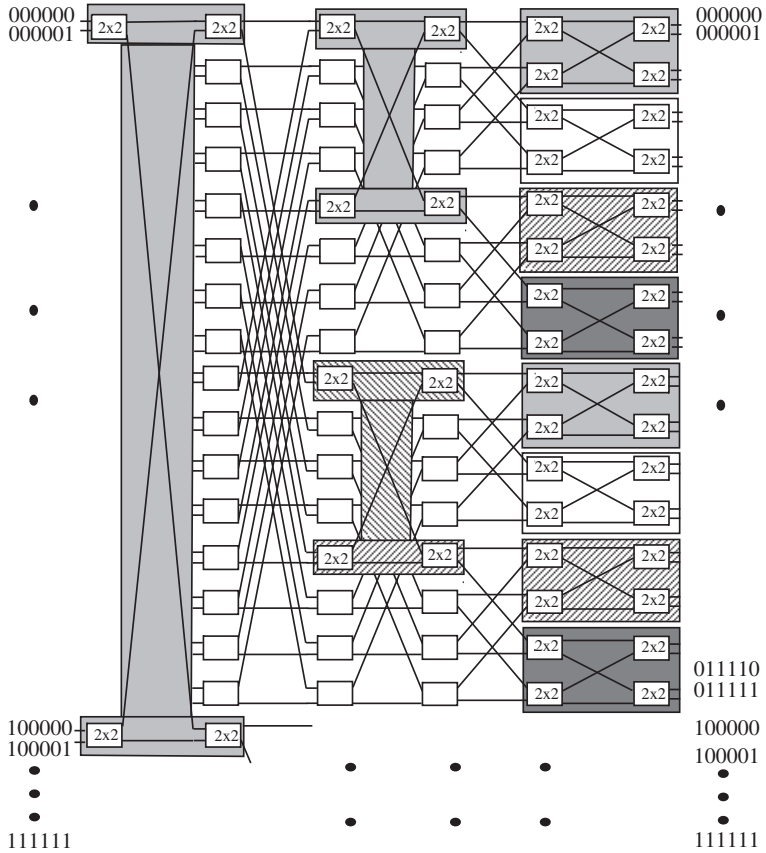


Throughput and latency vs. input rate for 4×4 switches.



Throughput and latency vs. input rate for 16×16 switches.

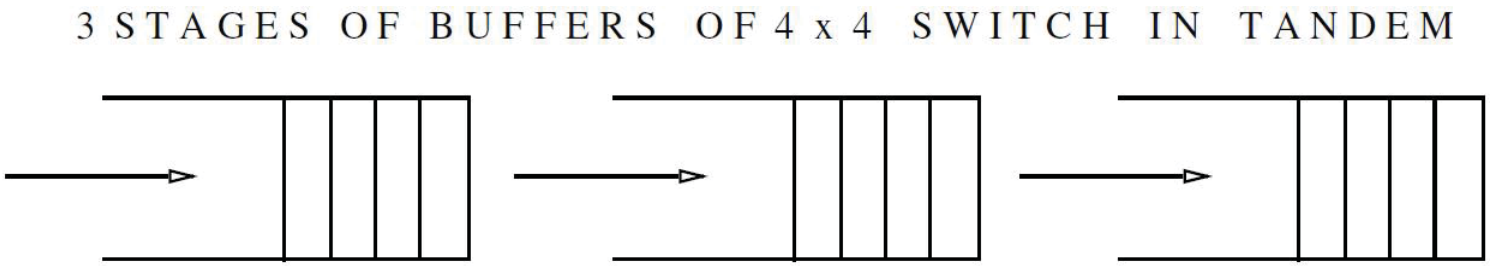
Network Decomposition - Larger Switches



A 64-input unique-path network reconfigured with 4×4 switches.

Analysis of Unique-Path Networks

Buffers assigned to different paths are independent (unique-path, symmetric traffic).



Queuing model for 64-input network.

Buffers on a given network path are interdependent, since a packet moves forward only if the next buffer is not full. *Models are adjusted by a) reducing the probability of a packet moving forward (R), by multiplying R with the probability of a nonfull buffer at the next stage, and b) mapping throughput rate at one stage to input rate at the next stage. Iterative method until convergence of latency, throughput, queue size values.*

Network Performance

$Butterfly\ Network$	$Latency,\ L$	$Throughput,\ S$
$N = 4^5$	31.83	0.993
$N = 32^2$	6.79	0.508

- **The network configured with smaller switches has higher throughput and latency.**
- **If switches were of similar technology, the network configured with smaller switches would be better.** *For example, constant pin-count constraints imply that the throughput of a* 4×4 *is 8 times as much and its latency is approximately 8 times smaller than the latency of a* 32×32 *switch.*
- **Adjustments are needed for actual routers.**

Model Example: STC104 32×32 vs. Telegraphos I 4×4 switch

- **One/two byte headers (10/20 bit) vs. one byte headers (8 bit).**
- **Message body 320 bits with 64 bits parity vs. 64 bits with five CRC bits.**
- **Four termination bits vs. no termination bits.**
- **Switch startup time** $500\ nsec$ **vs.** $160\ nsec$**.**
- **Link delay per bit** $10\ nsec$ **vs.** $5\ nsec$**.**
- **Input buffering (buffer size one) vs. central data buffering.**

Actual throughput and average latency of switch:

C104 32×32 **switch**	**Telegraphos** 4×4 **switch**
$S = \frac{S \cdot 256 bits}{3740 nsec} = 85.33 \cdot S\ Mb/sec$	$S = \frac{S \cdot 59 bits}{480 nsec} = 122.92 \cdot S\ Mb/sec$
$L = 3740 \cdot L\ nsec$	$L = 480 \cdot L\ nsec$

Extensions of Continuous Routing

- **Retransmissions (cell-loss probability).**
- **Hot-spots in links and/or processors.**
- **Broadcasting, multicasting, or combining switches.**

VLSI Complexity Model

Let A be the chip area, n be the problem size, and T be the time required to perform a computation on a VLSI chip. For certain computations, $A \times T^2 \geq O(f(n))$.

The bound can be considered as a **memory bound on chip area** *A, an* **I/O bound on volume** *AT, or most commonly a* **bisection communication bound** *$\sqrt{A}\ T$. The bisection represents the maximum amount of information exchange between two halves of the chip.*

With multilayer chips the bisection is $(AT)^{\frac{2}{3}}$.

VLSI Layout

- *VLSI complexity aspects. For certain computations, if s is the problem size, then $AT^2 = f(s)$, for a 2-D chip, and $V^2 \cdot T^3 = f'(s)$ for a 3-D chip [Th80, Ul84].*
- *VLSI design aspects. For a given bisection bandwidth and pin-out count (between 256-512), maximize the clock speed and wire length within the chip.*

Planar layouts for the $N = 2^{2n}$ node binary hypercube.

- *As $2^n \times 2^n$ mesh, with Gray code numbering. Better only for certain communication patterns, like FFT.*
- *As $2^n \times 2^n$ mesh, with row-major numbering. Smaller layout area. Similar wire length along any hypercube dimension.*

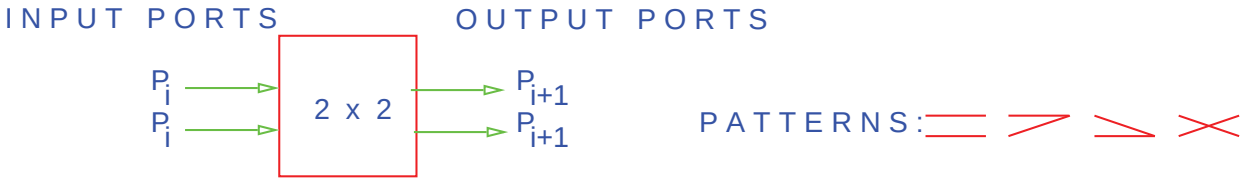
VLSI Layout - Signal Delay

The communication/processing speed is often determined by the worst-case signal delay, although density and yield play an important role.

Signal delay models.

- *Capacitive model. The signal delay of a wire of length l is proportional to $logl$.*
- *Resistive model. The signal delay of a wire of length l is proportional to l^2. This can be reduced to l if repeaters are provided, or if the speed of light is the limiting factor.*
- *Constant delay model. Signal delay is determined by parameters other than the wire length.*

Circuit Switching Models



Using independence, retransmissions on $n \times n$ switch.

$$P_{i+1} = 1 - [1 - \frac{P_i}{n}]^n$$

For $N/2 \cdot \log N$ switch network $\rightarrow$ recurrence equations.

Extension to nonuniform traffic (m_0 to hot spot).

$$m_0 + (k-1) \cdot m_1 + (n-k) \cdot m_2 = 1$$
$$P_{i+1} = 1 - [1 - P_i \cdot m_0] \cdot [1 - P_i \cdot m_1]^{k-1} \cdot [1 - P_i \cdot m_2]^{n-k}$$

The combinatorial power of multistage networks can be approximated using the blocking probability P_b.

$$C_N \approx 1 - P_b^n$$

Circuit Switching - BBN Butterfly

Butterfly Network	*Technique*	*Throughput*
$N = 2^9$	*single data*	28 %
$N = 2^9$	*duplex data*	48 %
$N = 2^9$	*quadruplex data*	73 %
$N = 2^9$	4×4 *switches*	41 %
$N = 4^4$	*collapsing stages*	37 %
$N = 4^5$	*collapsing stages*	32 %