BANYAN-BASED SWITCHES

HIGH PERFORMANCE SWITCHES AND ROUTERS
Wiley
H. JONATHAN CHAO and BIN LIU
Instructor: Mansour Rousta Zadeh

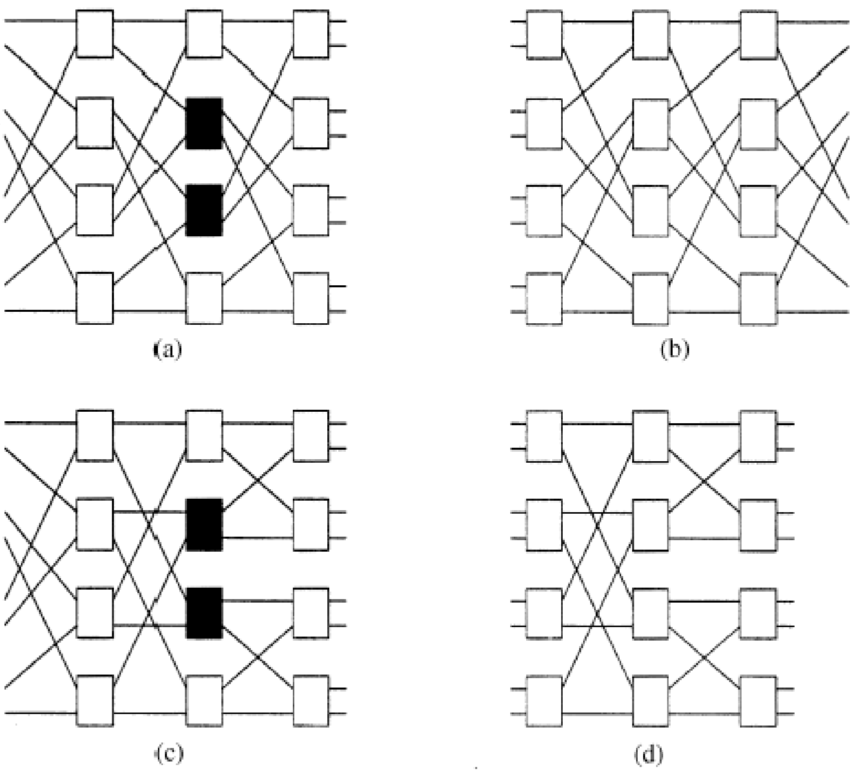
INTRODUCTION

Multistage Network

- First in circuit switched telephone networks
- To aim nonblocking with less crosspoints than a crossbar

Banyan network

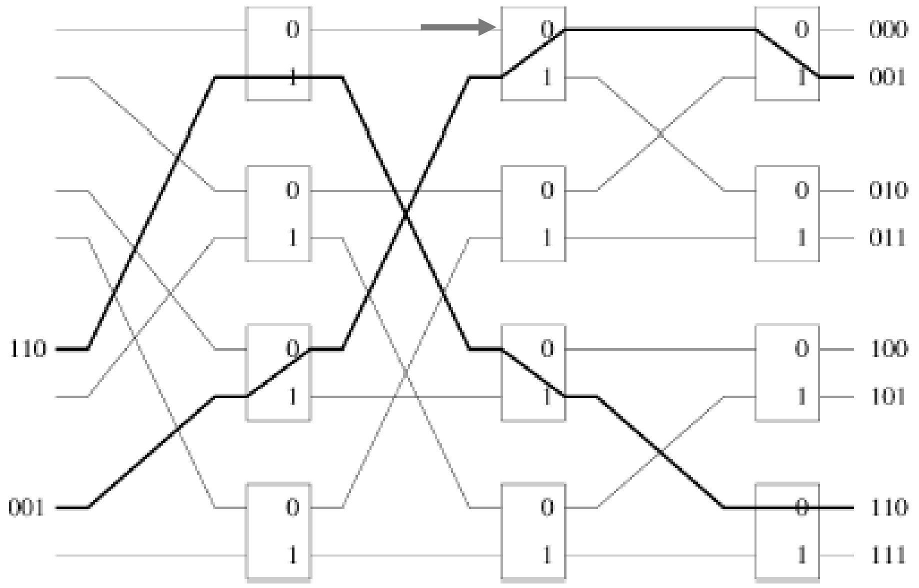
- Exactly one path from any input to any output
- 4 classes
  a) Shuffle-exchange (Omega)
  b) Reverse shuffle-exchange
  c) Narrow-sense banyan
  d) Baseline



INTRODUCTION

Banyan network principal properties

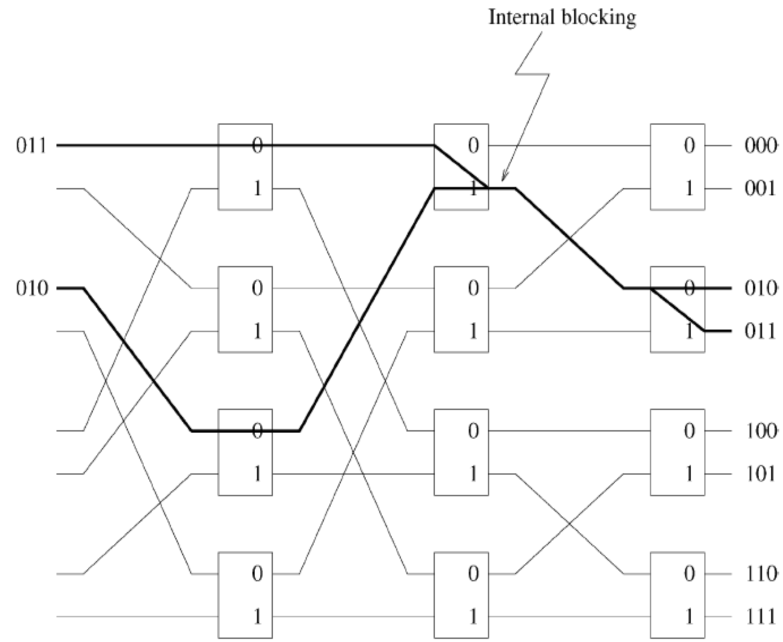
- Stages = n = $\mathrm{Log}_2 N$,
  Nodes per stage = N/2
- Self routing: one bit check in each step
- Regularity: Attractive for VLSI implementation

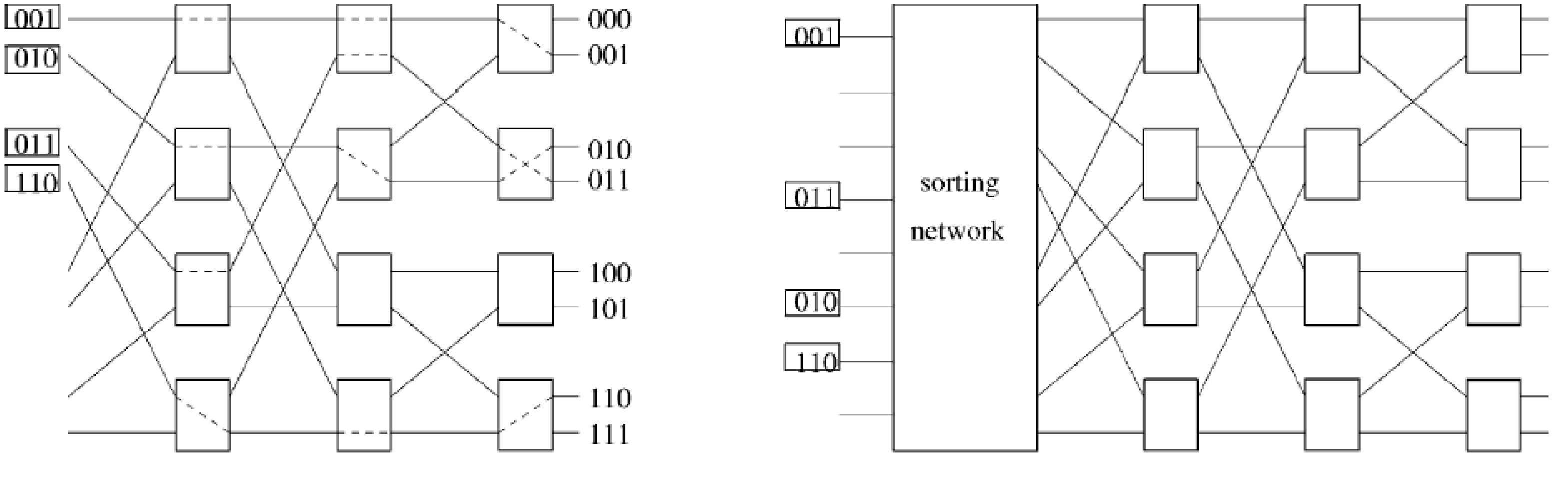


INTERNAL BLOCKING

Internal blocking

- Cell is lost due to the contention on a link inside the network
- Can not occur in banyan networks if
  - There is no idle input between any two active inputs
  - Destination address of cells are sorted
- The need for sorting network





(a) (b)

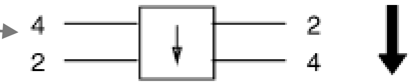
BATCHER SORTING NETWORK

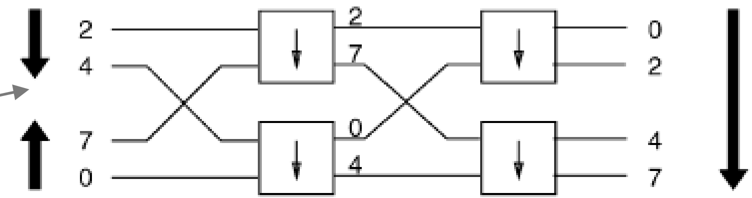
Merge network

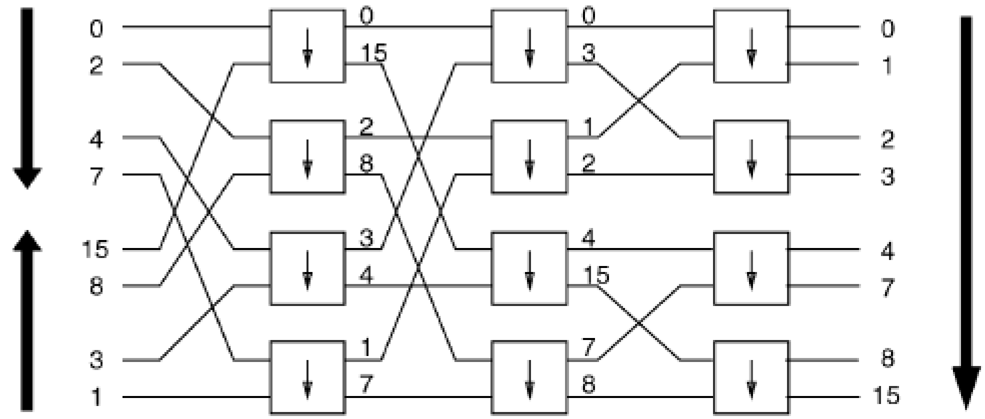
- Consists of 2*2 sorting elements
- Partial sort
- Merge2
- Merge4
- Merge8
- MergeN contains $(N\mathrm{Log}_2N)/2$ SEs

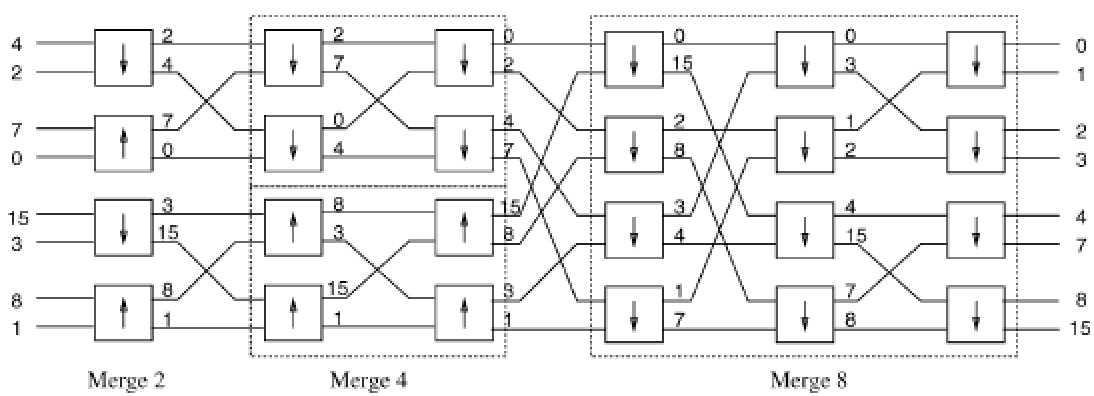
Batcher sorting network

- A series of merge networks
- 8*8: merge2 → merge4 → merge8
- N*N batcher network
  - $(N\mathrm{Log}_2N)(\mathrm{Log}_2N+1)/4$ SEs
  - $1+2+\ldots+\mathrm{Log}_2N = (\mathrm{Log}_2N)(\mathrm{Log}_2N+1)/2$ Stages

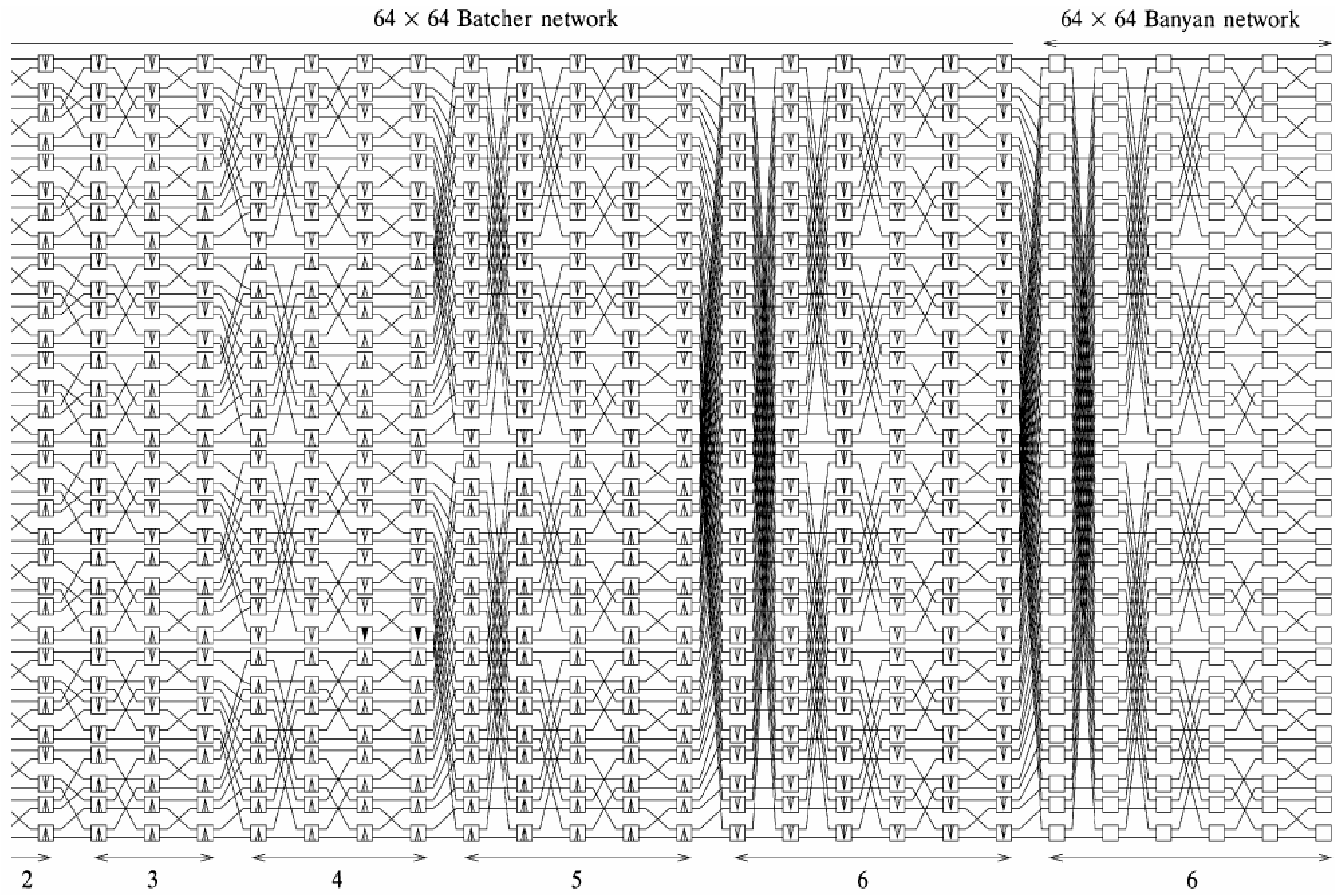








BATCHER SORTING NETWORK



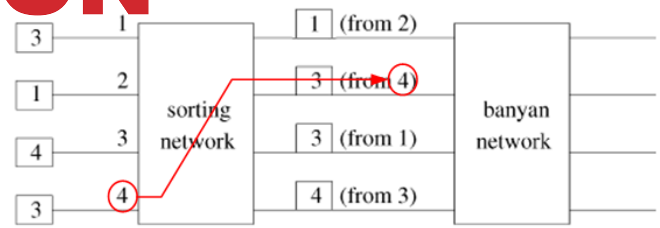
OUTPUT CONTENTION RESOLUTION

Output contention resolution algorithms

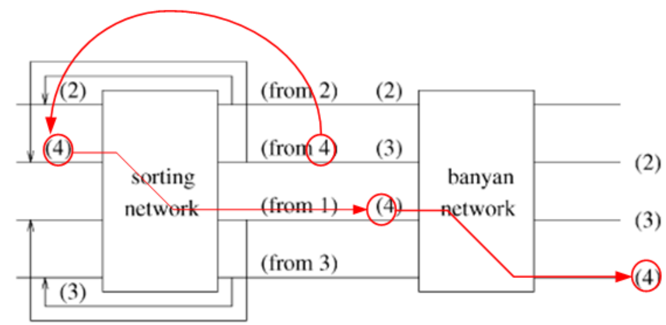
- Three phase method
- Ring reservation method

Three phase method

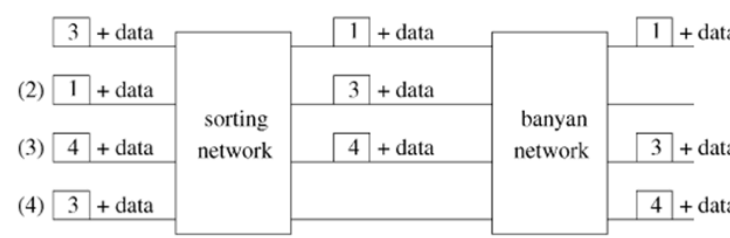
- phases
  - Request
  - Acknowledge
  - Data



(a) Phase I: Send and resolve request
- Send source-destination pair through sorting network
- Sort destination in non-decreasing order
- Purge adjacent requests with same destination

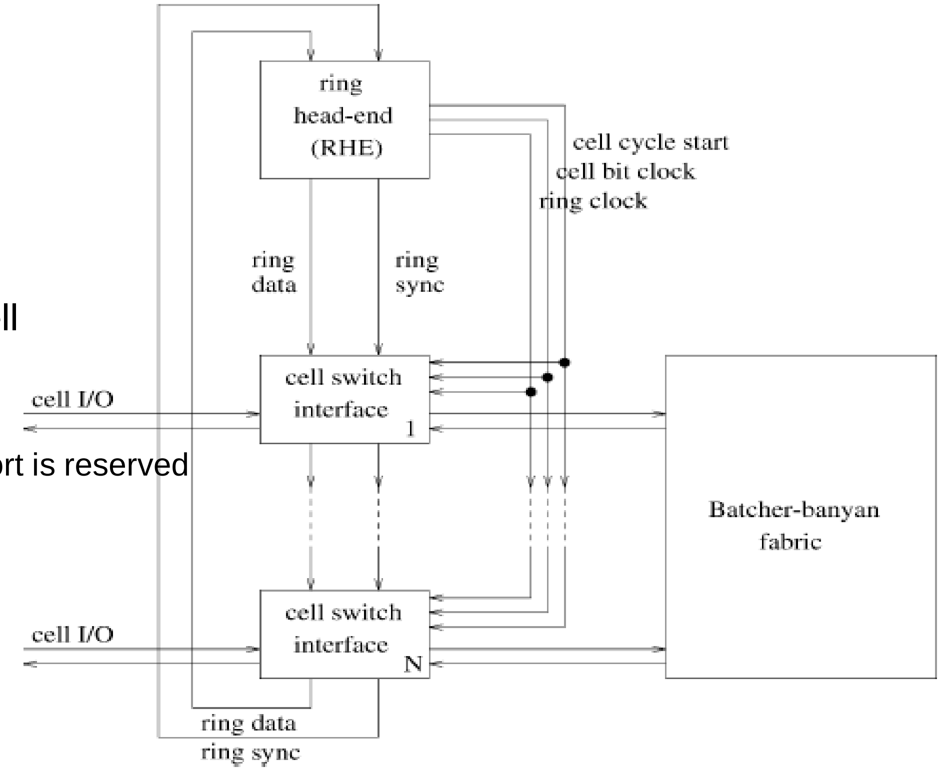


(b) Phase II: Acknowledge winning ports
- Send ACK with source to ports winning contention
- Route ACK through Batcher-banyan network



(c) Phase III: Send with data
- Acknowledged ports send cells through Batcher-banyan network
- Cells not acknowledged are buffered and retry in the next slot

OUTPUT CONTENTION RESOLUTION



Ring reservation method

- A batcher-banyan fabric
- A ring head end (RHE)
- N switch interfaces

Each switch interface

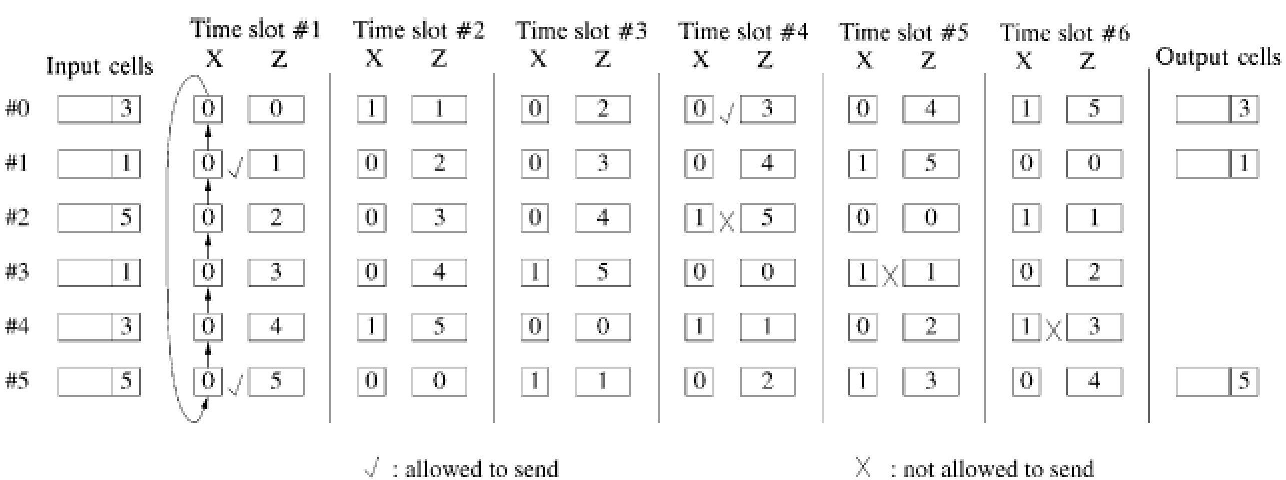
- Buffering input and output cells
- Competition for reserving output port of the cell
- A counter in each switch interface
    - Reset by RHE control signals
    - Incremented at each time slot
    - Match with output port number → output port is reserved

Advantages

- Fairness
- Can be used for all switch types

Disadvantage

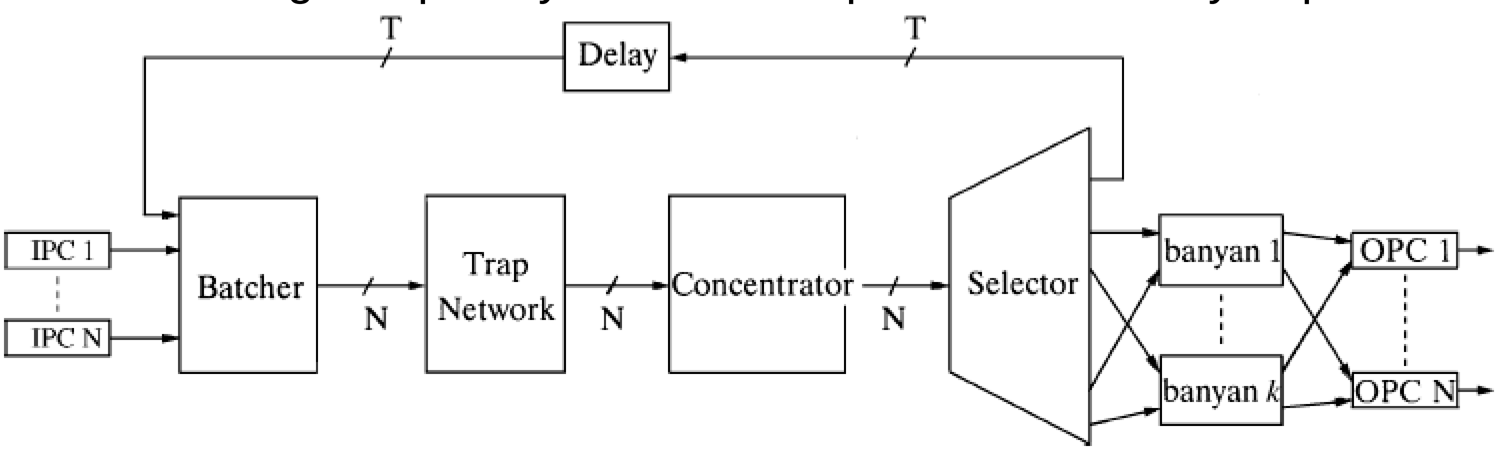
- N time slots for each arbitration



THE SUNSHINE SWITCH

The Sunshine switch

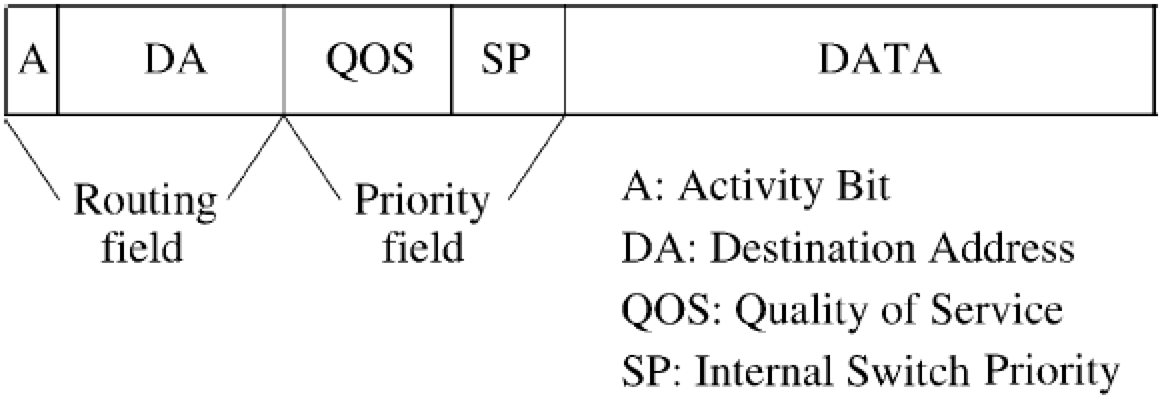
- A batcher sorting network
- k banyan networks in parallel
- More than one path to each destination (k paths)
- Recirculating queue
  - T paths in the figure
  - If more than k cells have the same output port
  - Excess cells are recirculated with a delay
- Batcher network
  - Sorts in the order of port & priority
  - The highest priority cell for each port is selected by trap network



THE SUNSHINE SWITCH

The sunshine switch

- Control Header
  - Is added to each cell by IPC
  - Two parts
    - Routing part
      - A: cell activity (non-emptiness)
      - DA: destination address
    - Priority part
      - QoS: quality of service (priority of cell)
      - SP: switch priority (assigned by switch to compensate the recirculation delay)



DEFLECTION ROUTING

Deflection

- Two cells contend at a node
- One of them will be routed incorrectly

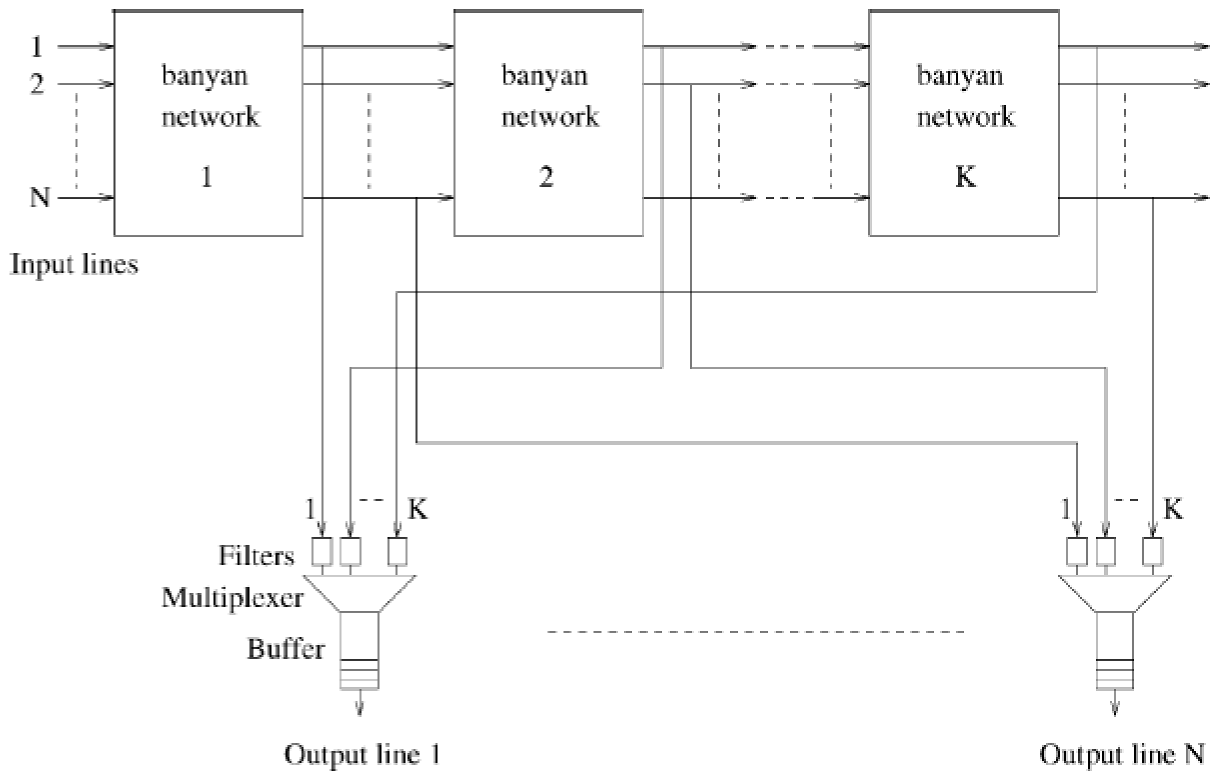
Works on deflection routing

- Tandem banyan switch
- Shuffle-exchange network with deflection routing
- Dual shuffle exchange network with error-correcting routing

TANDEM BANYAN SWITCH

Tandem banyan switch

- Chain of K banyan networks
- Deflected cells continue into the next banyan network
- Correctly routed cells go to output buffers
- One added banyan network → one order of magnitude reduction in deflections



TANDEM BANYAN SWITCH

Tandem banyan switch

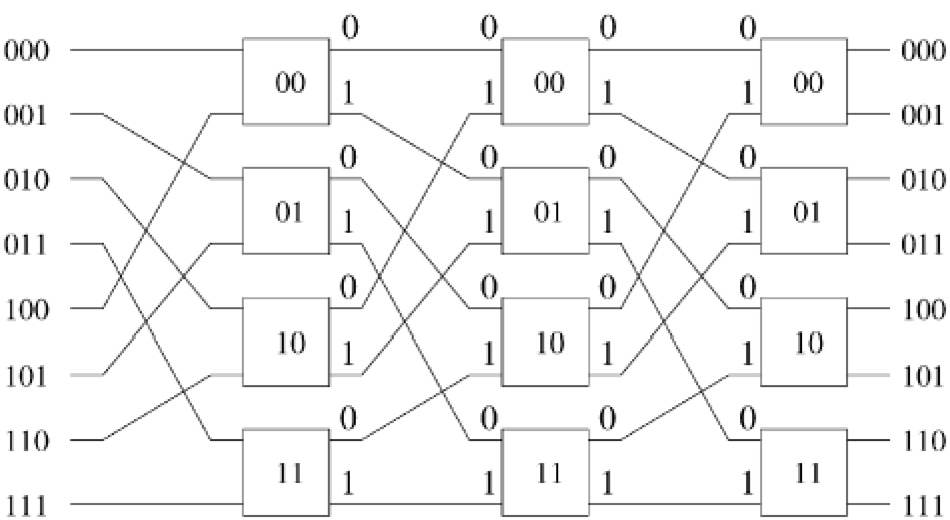
- Switching header
    - Activity bit: a
    - Conflict bit: c
    - Priority field: P
    - Address field: D
- For two cells 1, 2 in the same stage s:

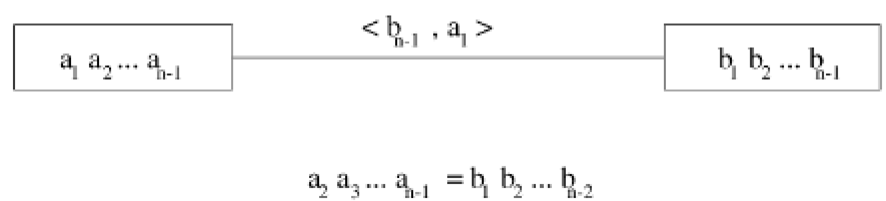
1. If $a_1 = a_2 = 0$, then take no action, i.e., leave the switch in the present state.
2. If $a_1 = 1$ and $a_2 = 0$, then set the switch according to d_{s1}.
3. If $a_1 = 0$ and $a_2 = 1$, then set the switch according to d_{s2}.
4. If $a_1 = a_2 = 1$, then:
   (a) If $c_1 = c_2 = 1$, then take no action.
   (b) If $c_1 = 0$ and $c_2 = 1$, then set the switch according to d_{s1}.
   (c) If $c_1 = 1$ and $c_2 = 0$, then set the switch according to d_{s2}.
   (d) If $c_1 = c_2 = 0$, then:
      i. If $P_1 > P_2$, then set the switch according to d_{s1}.
      ii. If $P_1 < P_2$, then set the switch according to d_{s2}.
      iii. If $P_1 = P_2$, then set the switch according to either d_{s1} or d_{s2}.
      iv. If one of the cells has been misrouted, then set its conflict bit to 1

SHUFFLE-EXCHANGE NETWORK WITH DEFLECTION ROUTING

N*N shuffle-exchange network (SN)

- Stages = n = $\log_2 N$
- SEs per stage = N/2
- SE labels: numbers with n-1 bits length
- SE input (output) labels: 1 bit numbers
- How SE forwards cells
  - The cell with a 0 in i'th destination address bit goes to output 0
  - The cell with a 1 in i'th destination address bit goes to output 1
- Network connections:
  - Consider binary lebels represented in form $(a_1a_2\ldots)$
  - Output a_n of node $X=(a_1a_2\ldots a_{n-1})$ → input a_1 of node $Y=(a_2a_3\ldots a_n)$
  - This link is represented as $<a_n,a_1>$





SHUFFLE EXCHANGE NETWORK WITH DEFLECTION ROUTING

The path from input to output

- Is determined by:
  - Source address $s_1 s_2 \ldots s_n$
  - Destination address $d_1 d_2 \ldots d_n$

$$
\begin{aligned}
S = s_1 \cdots s_n
&\xRightarrow{\langle -, s_1 \rangle} (s_2 \ldots s_n) \xRightarrow{\langle d_1, s_2 \rangle} (s_3 \ldots s_n\, d_1) \\
&\xRightarrow{\langle d_2, s_3 \rangle} \cdots \xRightarrow{\langle d_{i-1}, s_i \rangle} (s_{i+1} \ldots s_n d_1 \ldots d_{i-1}) \\
&\xRightarrow{\langle d_i, s_{i+1} \rangle} \cdots \xRightarrow{\langle d_{n-1}, s_n \rangle} (d_1 \cdots d_{n-1}) \\
&\xRightarrow{\langle d_n, 0 \rangle} d_1 \ldots d_n = D.
\end{aligned}
$$

- An example: S=001 and D=101

$$001 \xrightarrow{<-,0>} 01 \xrightarrow{<1,0>} 11 \xrightarrow{<0,1>} 10 \xrightarrow{<1,0>} 101$$

$$S \xrightarrow{<-,s_1>} s_2 s_3 \xrightarrow{<d_1,s_2>} s_3 d_1 \xrightarrow{<d_2,s_3>} d_1 d_2 \xrightarrow{<d_3,0>} D$$

- SEs the cell passes
  - An string $s_2 \ldots s_n d_1 \ldots d_{n-1}$
  - An (n-1) bits window shifting one bit in each stage from left to right

SHUFFLE EXCHANGE NETWORK WITH DEFLECTION ROUTING

State of the traveling cell

- Pair (R,X)
- R: Current routing tag
- X: Label of SE which cell resides
- First state: ($d_n \ldots d_1$, $s_2 \ldots s_n$)
- State transition (self routing algorithm):

$$\underset{\text{input label } x_n}{(r_1 \ldots r_k, x_1 x_2 \ldots x_{n-1})} \xRightarrow{\text{exchange}} \underset{\text{output label } r_k}{(r_1 \ldots r_{k-1}, x_1 x_2 \ldots x_{n-1})}$$

$$\xRightarrow[\langle r_k, x_1 \rangle]{\text{shuffle}} (r_1 \ldots r_{k-1}, x_2 \ldots x_{n-1} r_k).$$

- Routing bit is removed after each transition
- Final state: ($d_1 \ldots d_{n-1}$)

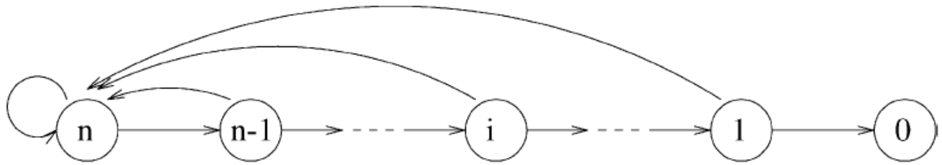
Deflected cells

- Routing tag is reset to $d_n \ldots d_1$
- Network is extended to have more than n stages

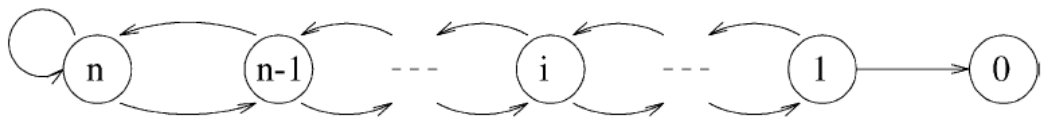
DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

SN with deflection routing (the previous scheme)

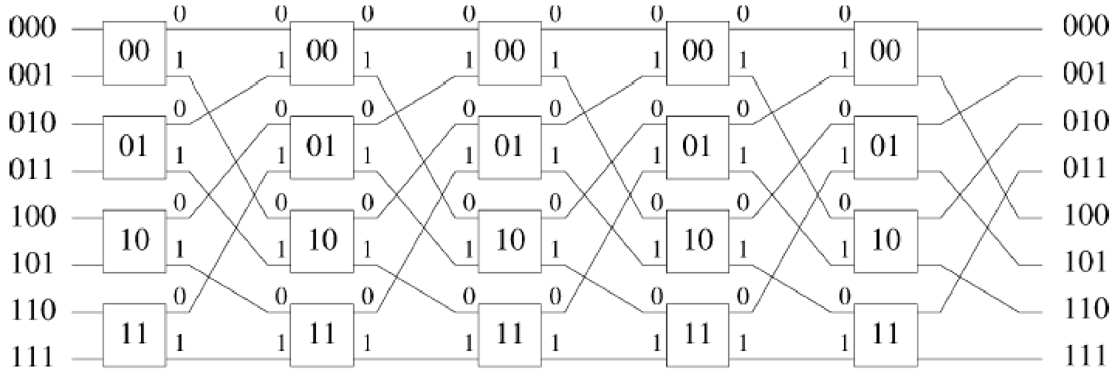
- Highly inefficient: Routing must be restarted for deflected cell:



- Desired network behavior:



- Dual shuffle-exchange network
  - A shuffle-exchange network (SN)
  - An unshuffle-exchange network (USN)
    - Mirror of SN network
    - An 8*8 example of USN:



DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

USN: mirror image of SN

Similar rules as SN:

- SE connections:

$$a_1 a_2 \ldots a_{n-1} \xrightarrow{\langle b_1 , a_{n-1} \rangle} b_1 b_2 \ldots b_{n-1}$$

$$a_1 a_2 \ldots a_{n-2} = b_2 b_3 \ldots b_{n-1}$$

- Paths from inputs to outputs:

$$
\begin{aligned}
S = s_1 \ldots s_n \\
&\xRightarrow{\langle - , s_n \rangle} (s_1 \ldots s_{n-1}) \xRightarrow{\langle d_n , s_{n-1} \rangle} (d_n \, s_1 \ldots s_{n-2}) \\
&\xRightarrow{\langle d_{n-1} , s_{n-2} \rangle} \ldots \xRightarrow{\langle d_{i+2} , s_{i+1} \rangle} (d_{i+2} \ldots d_n s_1 \ldots s_i) \\
&\xRightarrow{\langle d_{i+1} , s_i \rangle} \ldots \xRightarrow{\langle d_2 , s_1 \rangle} (d_2 \ldots d_n) \\
&\xrightarrow{\langle d_1 , 0 \rangle} d_1 \ldots d_n = D.
\end{aligned}
$$

  - SEs the cell passes
    - An string $d_2 \ldots d_n s_1 \ldots s_{n-1}$
    - An (n-1) bits window shifting one bit in each stage from right to left
- Traveling cell state diagram:
  - Initial state: $(d_1 \ldots d_n , s_1 \ldots s_{n-1})$
  - Final state: $(d_1 \ldots d_n)$
  - Transition:

$$\underset{\text{input label } x_n}{(r_1 \ldots r_k , x_1 x_2 \ldots x_{n-1})} \xRightarrow{\text{exchange}} \underset{\text{output label } r_k}{(r_1 \ldots r_{k-1} , x_1 x_2 \ldots x_{n-1})}$$

$$\xRightarrow[\langle r_k , x_{n-1} \rangle]{\text{unshuffle}} (r_1 \ldots r_{k-1} , r_k x_1 \ldots x_{n-2})$$

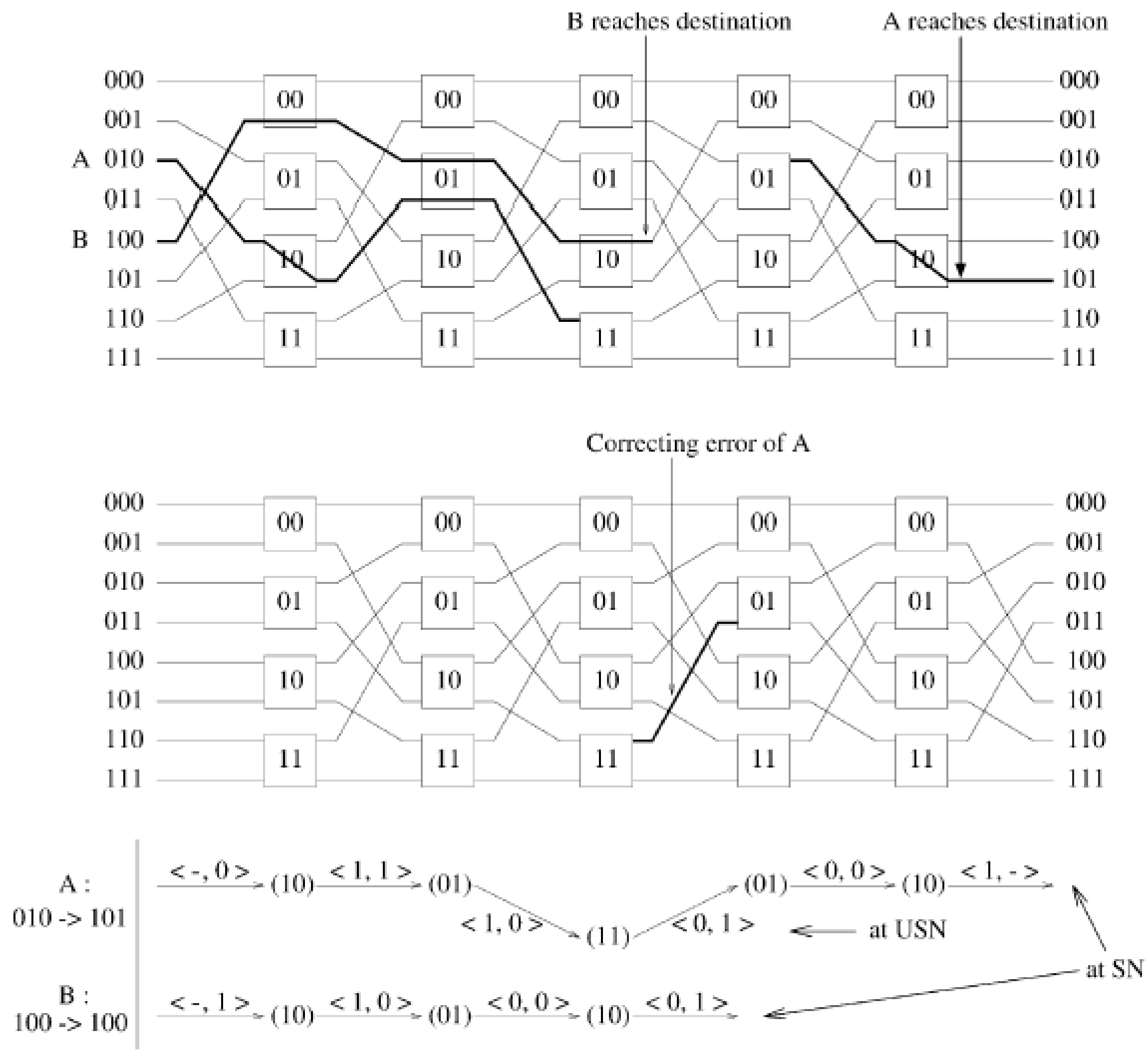
DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

Consider a USN overlaid on a SN

USN can undo what SN performs

Deflected cell can return to the state before deflection

Example:

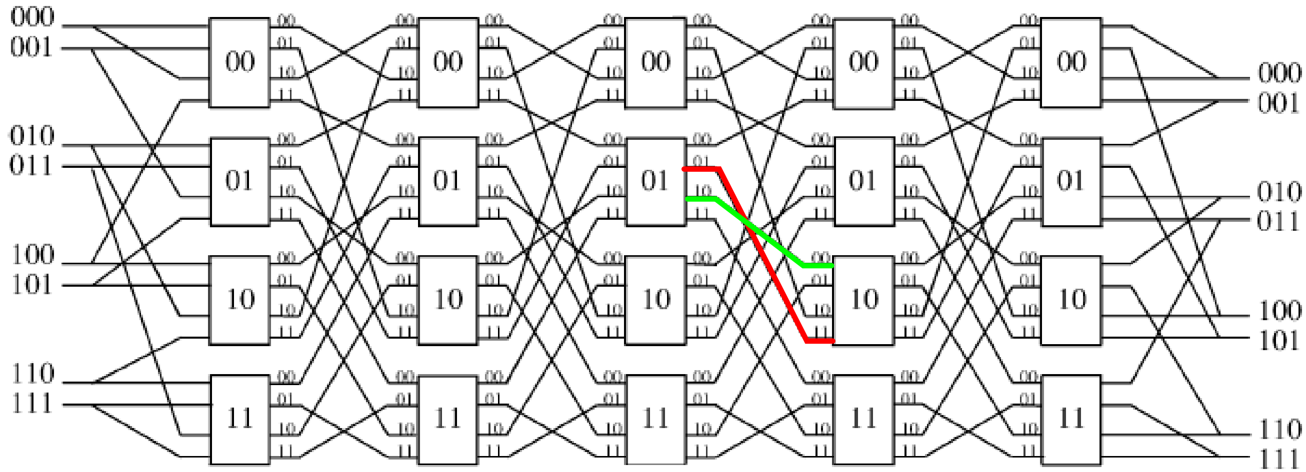


DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

Error correction procedure

- Cell state: $(r_1 \ldots r_k , x_1 \ldots x_{n-1})$
- Cell should be sent via link $<r_k,x_1>$ to the next stage
- It is deflected to link $<r'_k , x_1>$
- It goes to node $(x_2 \ldots x_{n-1} r'_k)$ in the next stage
- Error correction procedure does not remove the bit r_k
- Instead, it attaches bit x_1 to the routing tag
- New state: $(r_1 \ldots r_k x_1 , x_2 \ldots x_{n-1} r'_k)$
- Then the cell is moved to the USN
- It will be sent via link $<x_1 , r'_k>$
- It will return to the state before deflection: $(r_1 \ldots r_k , x_1 \ldots x_{n-1})$
- If cell is deflected in the USN, it can be corrected in SN in a similar way

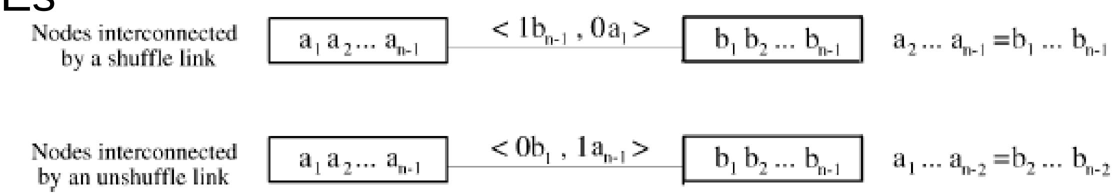
DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING



Merging SN and USN

- SN: 2*2 SEs
- USN: 2*2 SEs
- Merged (dual SN): 4*4 SEs

Labeling



- Inputs: 00..11
- Outputs: 00..11
- Unshuffle links: <0a,1b>
  - Connect outputs 10 or 11 to inputs 00 or 01 of the next stage
- Shuffle links: <1a,0b>
  - Connect outputs 00 or 01 to inputs 10 or 11 of the next stage

Connections

- Consider two nodes $A=(a_1 \ldots a_{n-1})$, $B=(b_1 \ldots b_{n-1})$
- They are connected via unshuffle link $<0b_1,1a_{n-1}>$ if $a_1 \ldots a_{n-2}=b_2 \ldots b_{n-1}$
- They are connected via shuffle link $<1b_{n-1},0a_1>$ if $a_2 \ldots a_{n-1}=b_1 \ldots b_{n-2}$

DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

States and transitions

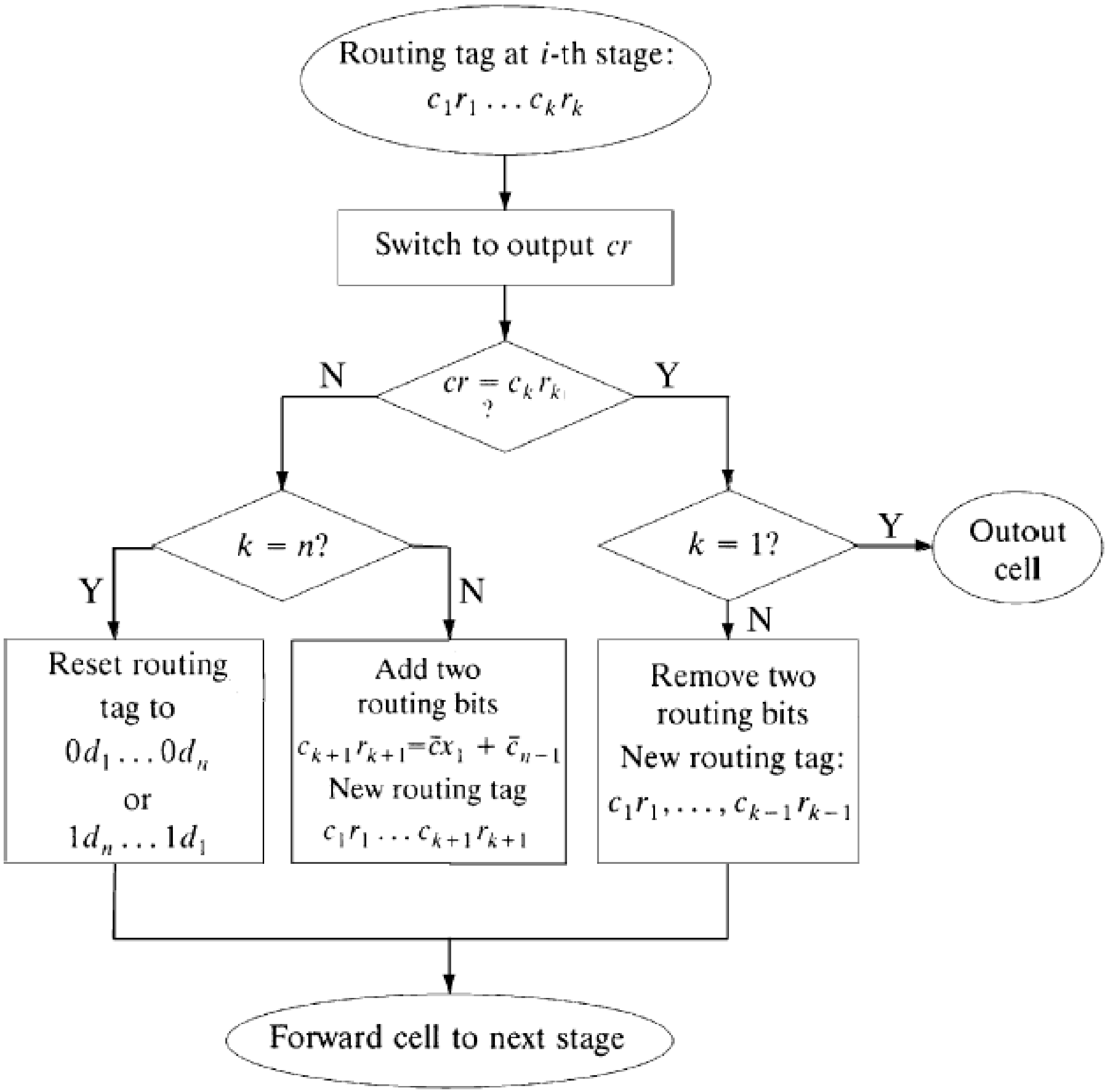
- Two routing bits required at each stage
- Destincation: D=$d_1 \dots d_n$
- Initial state:
    - $0d_1 \dots 0d_n$ (Starting from USN)
    - $1d_n \dots 1d_1$ (Starting from SN)
- Cell state: $(c_1r_1 \dots c_kr_k\ ,\ x_1 \dots x_{n-1})$
- Transitions:
    - $c_k=0$ → unshuffle link
    - $c_k=1$ → shuffle link
    -

$$(c_1 r_1 \dots c_k r_k, x_1 \dots x_{n-1})$$

$$\begin{cases} \xrightarrow{\langle 0 r_k, 1 x_{n-1} \rangle} & (c_1 r_1 \dots c_{k-1} r_{k-1}, r_k x_1 \dots x_{n-2}) \quad \text{if } c_k = 0, \\ \xrightarrow{\langle 1 r_k, 0 x_1 \rangle} & (c_1 r_1 \dots c_{k-1} r_{k-1}, x_2 \dots x_{n-1} r_k) \quad \text{if } c_k = 1. \end{cases}$$

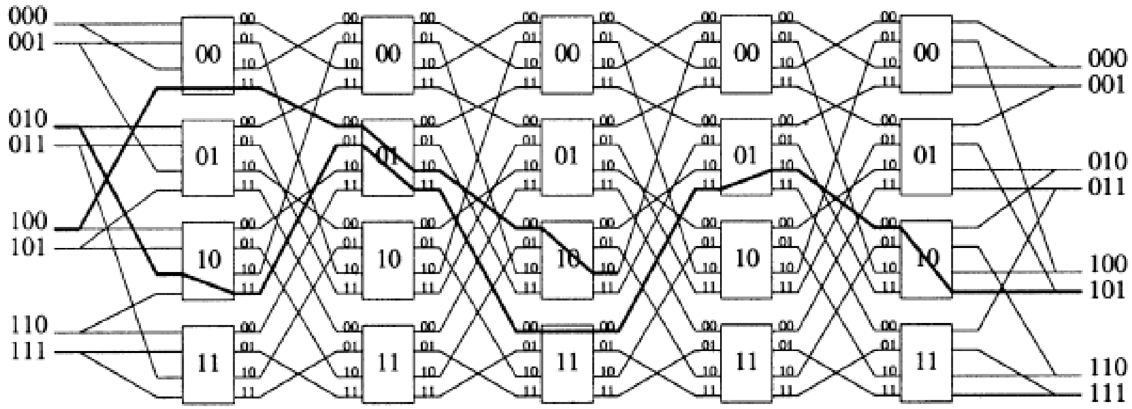
DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

The algorithm



DUAL SHUFFLE-EXCHANGE NETWORK WITH ERROR-CORRECTING ROUTING

An example
(The previous example)

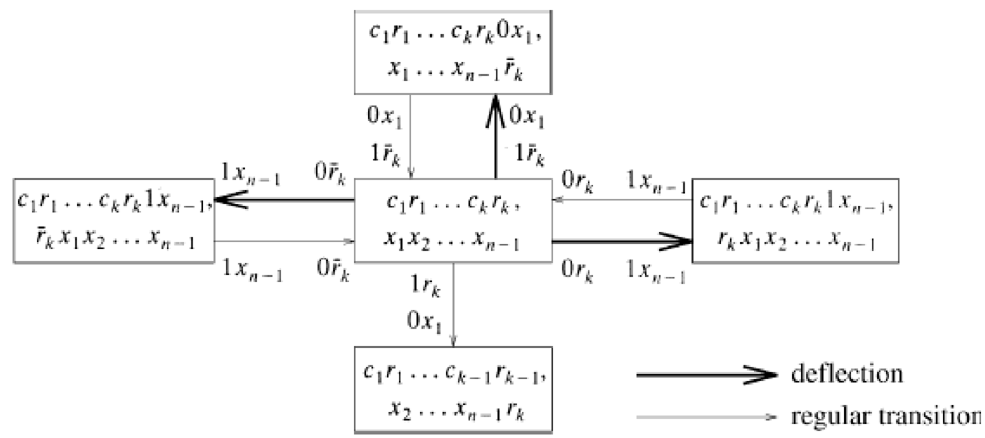


State transition of cells A and B

A: 010 -> (111011, 10) -> (1110, 01) -> (111000, 11) -> (1110, 01) -> (11, 10) -> 101 (error; error correction)

B: 100 -> (101011, 00) -> (1010, 01) -> (10, 10) ->100

State diagram



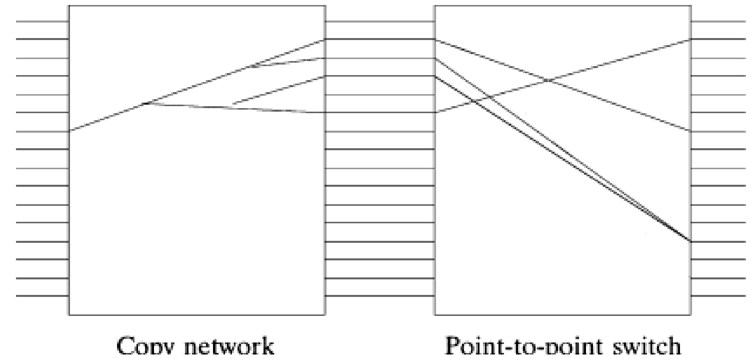
Transitions for deflection

$$(c_1r_1\dots c_kr_k, x_1\cdots x_{n-1}) \begin{cases} \xRightarrow{\langle 0r,1x_{n-1}\rangle} (c_1r_1\dots c_kr_k1x_{n-1}, rx_1\dots x_{n-2}) & \text{if } c_kr_k\neq 0r,\\ \xRightarrow{\langle 1r,0x_1\rangle} (c_1r_1\dots c_kr_k0x_1, x_2\dots x_{n-1}r) & \text{if } c_kr_k\neq 1r.\end{cases}$$

MULTICAST COPY NETWORKS

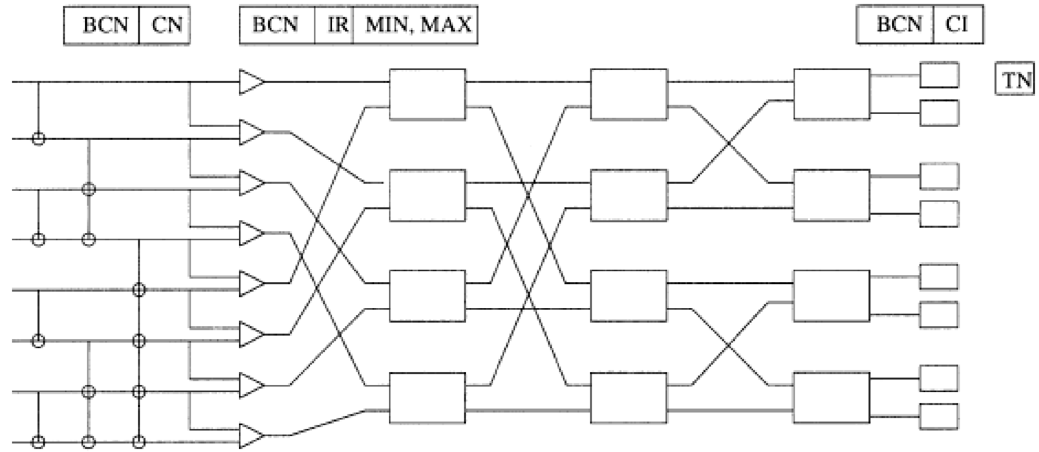
Point-to-multipoint communications

- A copy network (replicates cells)
- A point-to-point switch



The copy network components

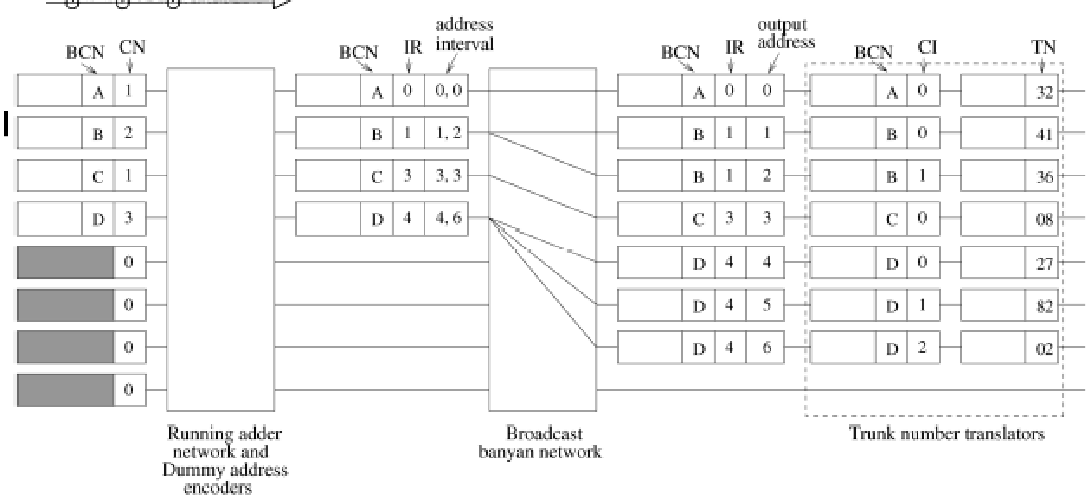
- Running address network (RAN)
- Dummy address encoder (DAE)
- Broadcast banyan network
- Trunk number translator



Acronyms in figure:

- CN: Number of copies
- IR: Index reference
- CI: Copy index
- BCN: Broadcast channel numbe

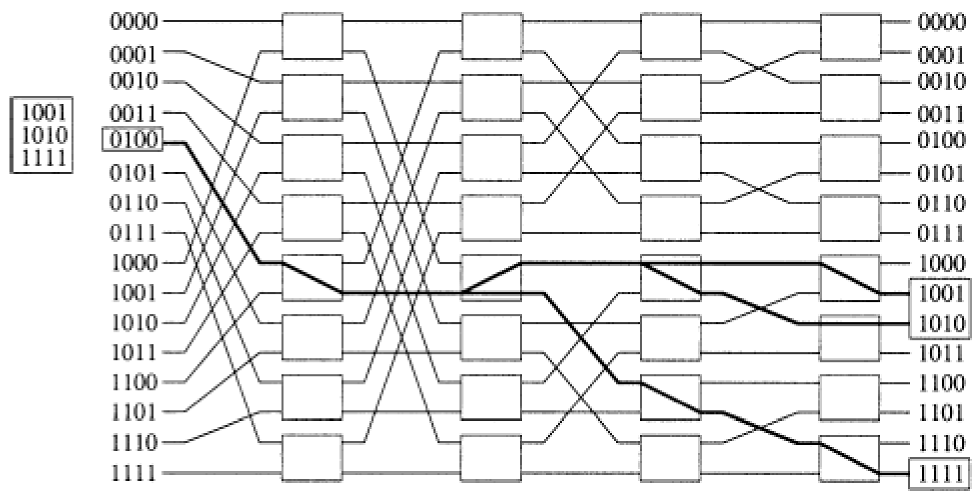
An example

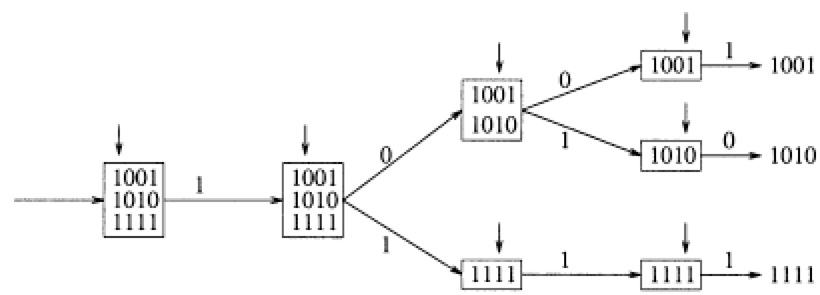


MULTICAST COPY NETWORKS

Broadcast banyan network

- SEs cam replicate cells
- 3 possibilities (2 bits needed in cell header):
  - Cell goes to output 1
  - Cell goes to output 1
  - Cell is replicated to both outputs
- Generalized self routing algorithm
  - Current bit of all destination addresses are checked at each stage
  - All zero → cell goes through output 0
  - All one → cell goes through output 1
  - Some zero, some one → cell is replicated
  - An example





MULTICAST COPY NETWORKS

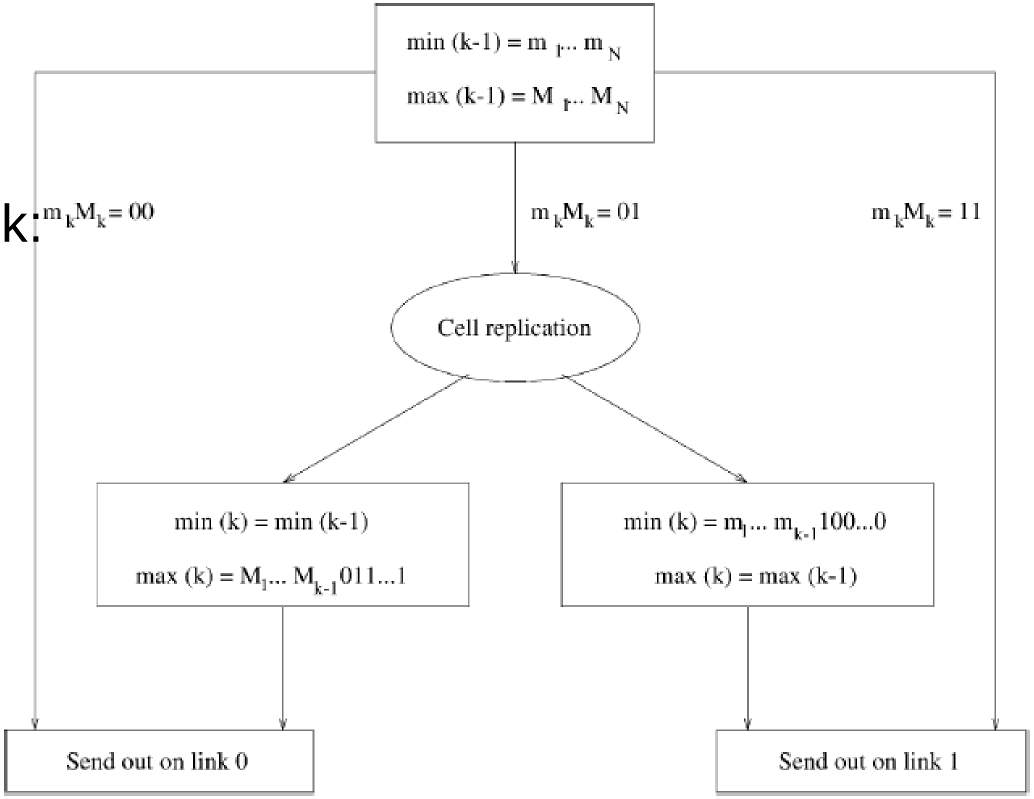
Broadcast banyan network

- Generalized self routing algorithm
  - Problems
    - A variable number of destination addresses
      - Must be recorded in cell header
      - Must be checked at SEs at each stage
      - Need to be processed for cell header modifications
    - Cell path forms a tree → more blocking
  - Solution
    - Boolean interval splitting algorithm

MULTICAST COPY NETWORKS

Boolean interval splitting algorithm

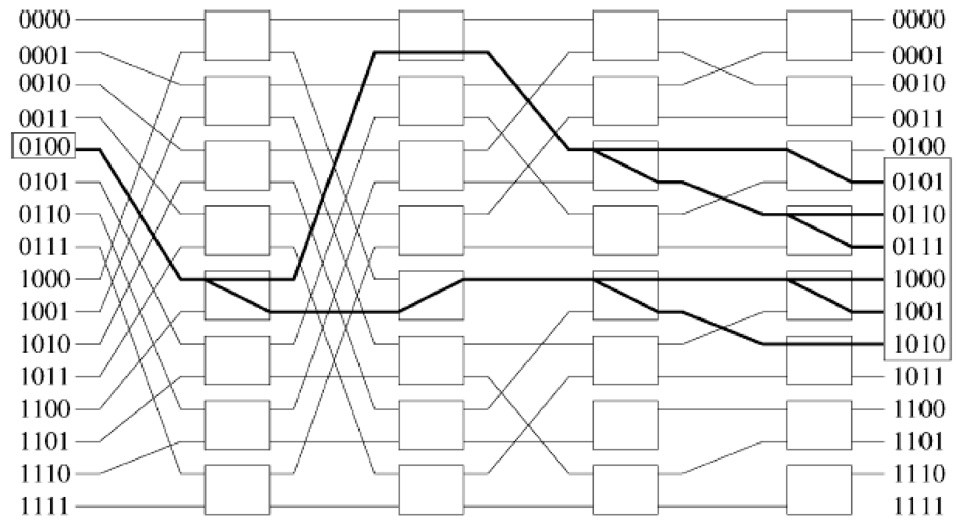
- Interval: a set of N-bits numbers between min and max
- Interval represents a set of contiguous destination addresses
- Stage k
  - $\min(k-1) = m_1 \ldots m_k$
  - $\max(k-1) = M_1 \ldots M_k$
  - Self routing at stage k:

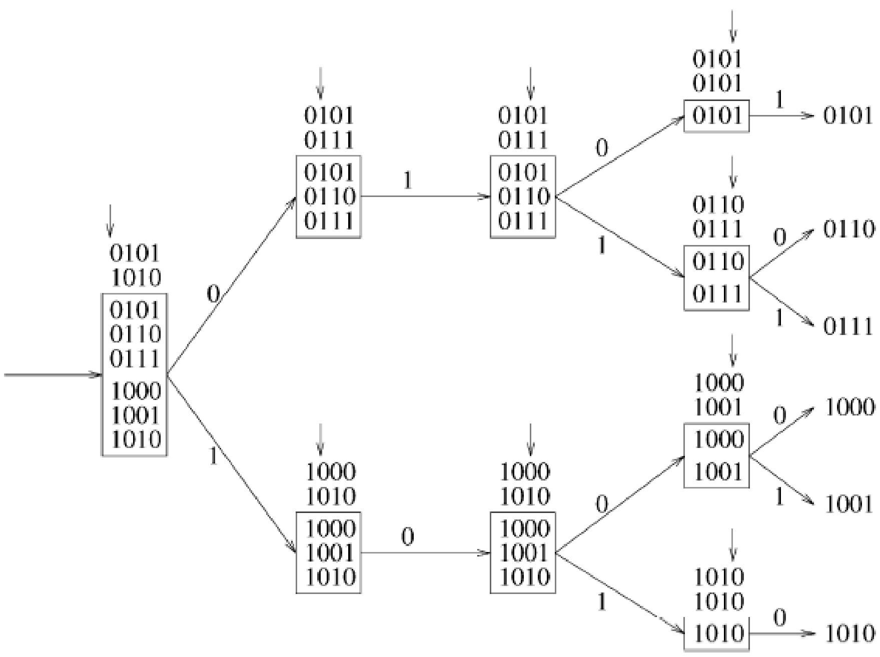


MULTICAST COPY NETWORKS

Boolean interval splitting algorithm

- An example





MULTICAST COPY NETWORKS

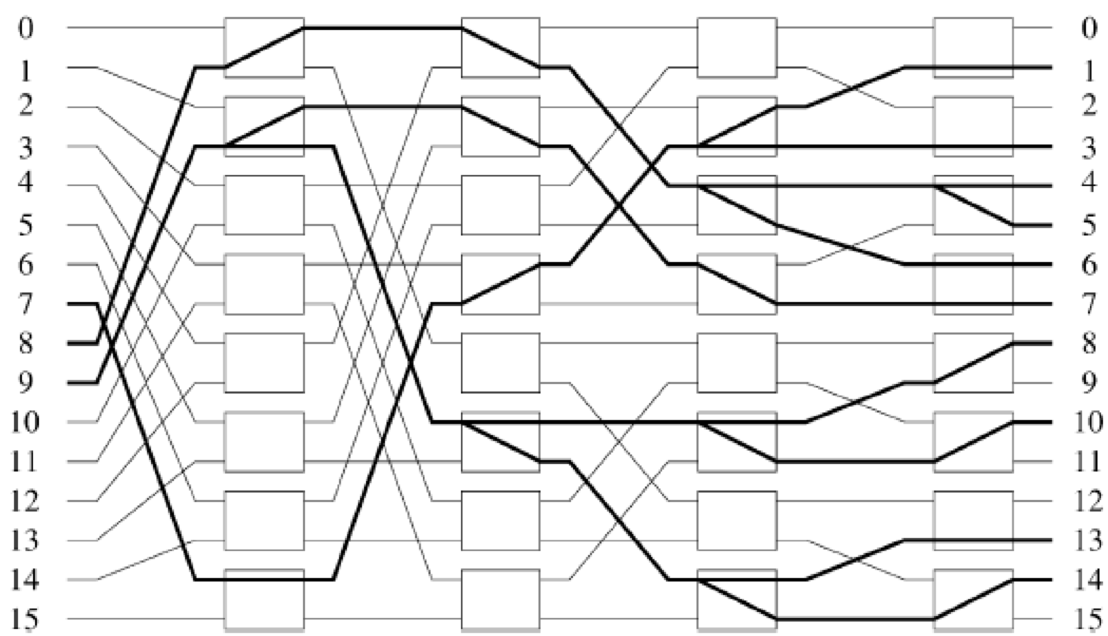
Broadcast banyan network is nonblocking when input cells are

- Monotonic
  - Output port sets are sorted
- Concentrated
  - No idle inputs exists between active outputs
- An example

$x_1 = 7 \quad Y_1 = \{1,3\}$

$x_2 = 8 \quad Y_2 = \{4,5,6\}$

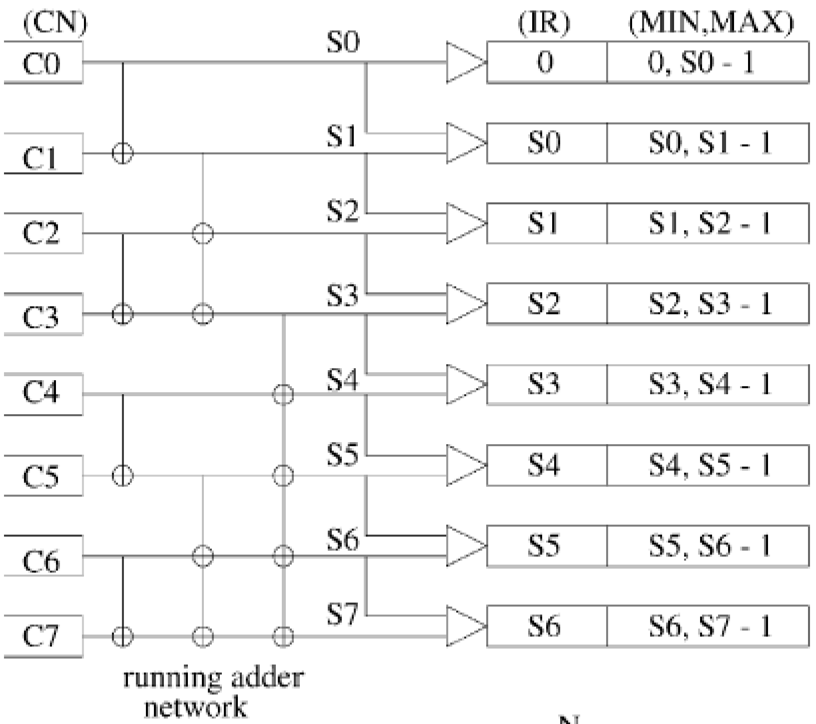
$x_3 = 9 \quad Y_3 = \{7,8,10,13,14\}$

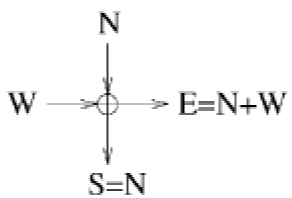


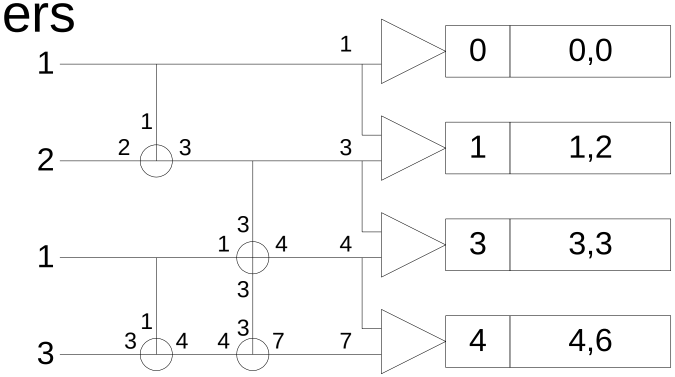
MULTICAST COPY NETWORKS

RAN, DAE

- Encoding process
  - RAN, DAE transform copy numbers into a set of monotonic addresses
- Decoding process
  - TNT determines the real destinations of copies
- RAN, DAE
  - $(N/2)\mathrm{Log}_2N$ adders
  - Log_2N stages
- An example



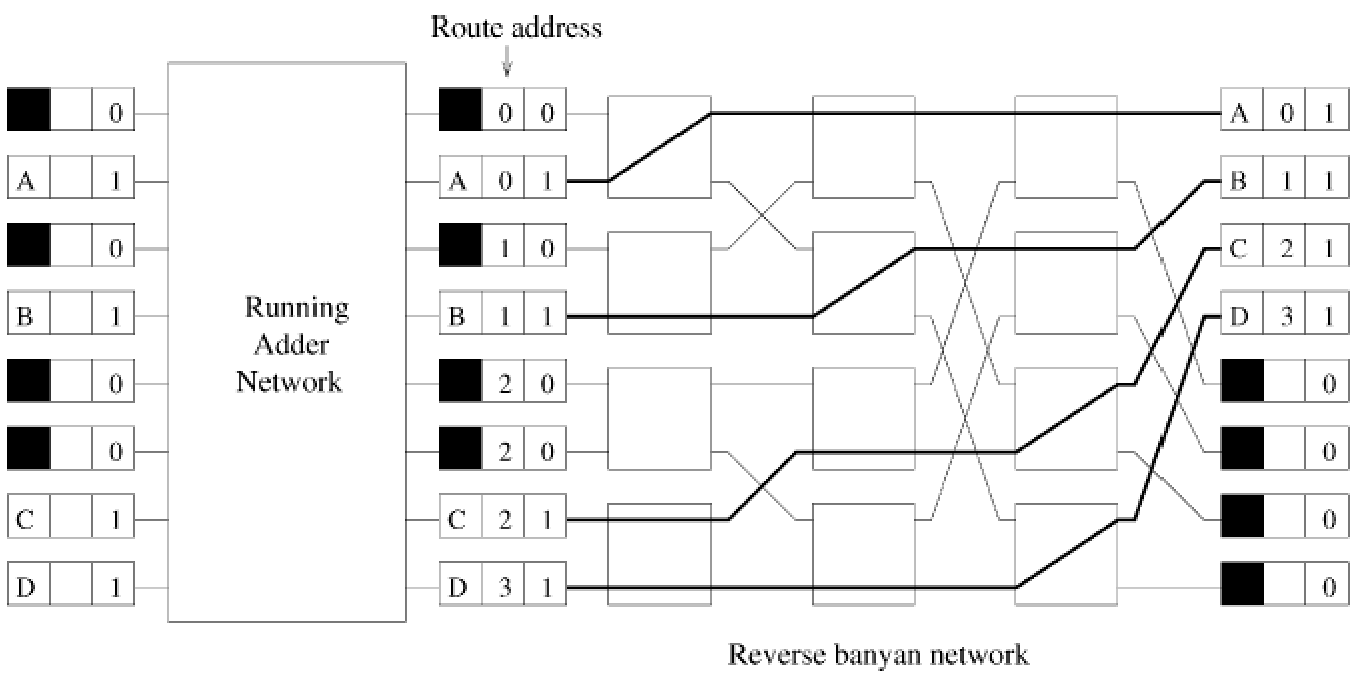




MULTICAST COPY NETWORKS

Concentration

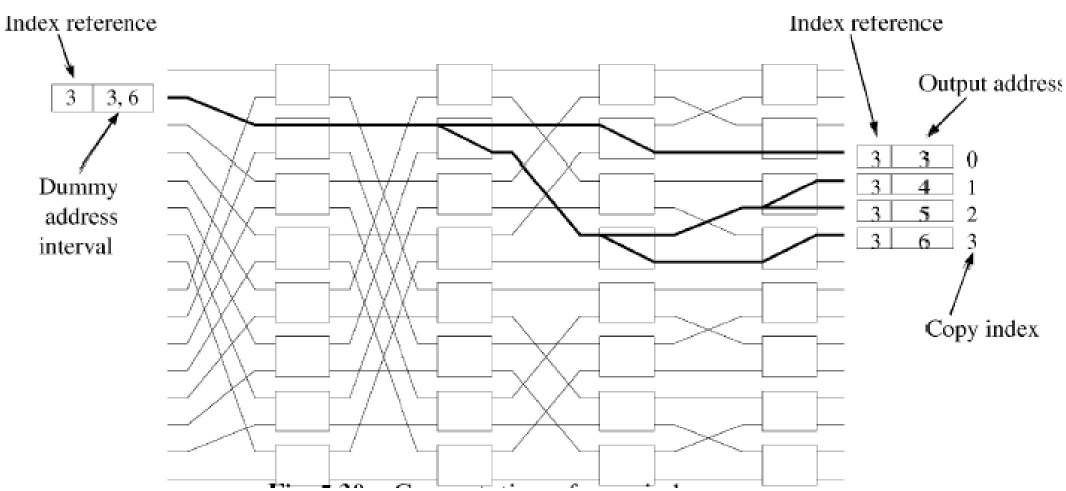
- Eliminate idle inputs between active outputs
- Must be done before broadcast banyan network
    - Berfor RAN, or
    - After DAE
- A reverse banyan network (RBN) can be used

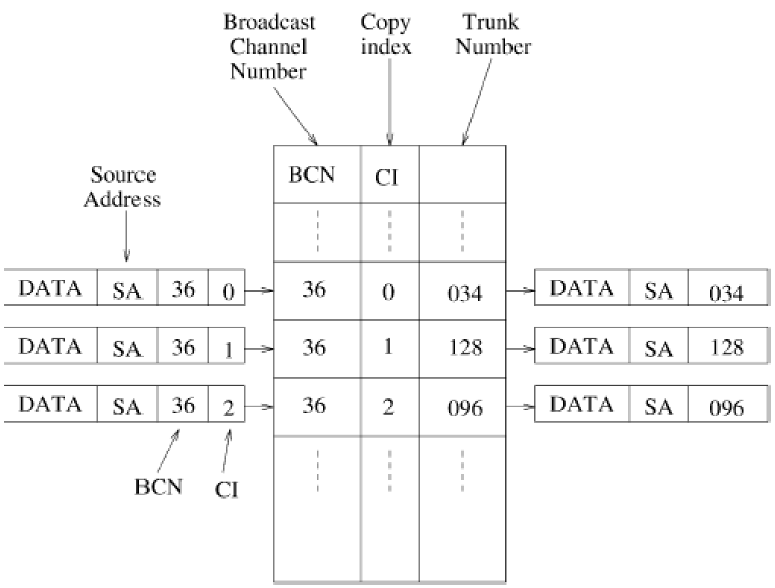


MULTICAST COPY NETWORKS

Decoding process

- When cell leaves broadcast banyan network,
  - The interval in the header is only one address
  - This address = min = max
  - Copy index = this address – index reference
  - TNT assigns the actual address to each copy
    - A simple table lookup
    - Search key: BCN and CI

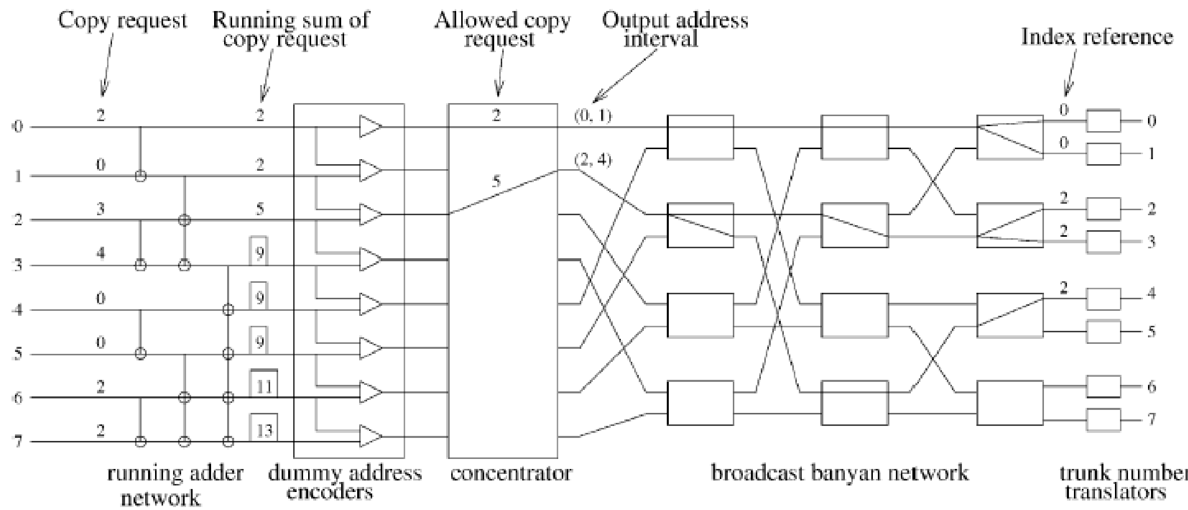


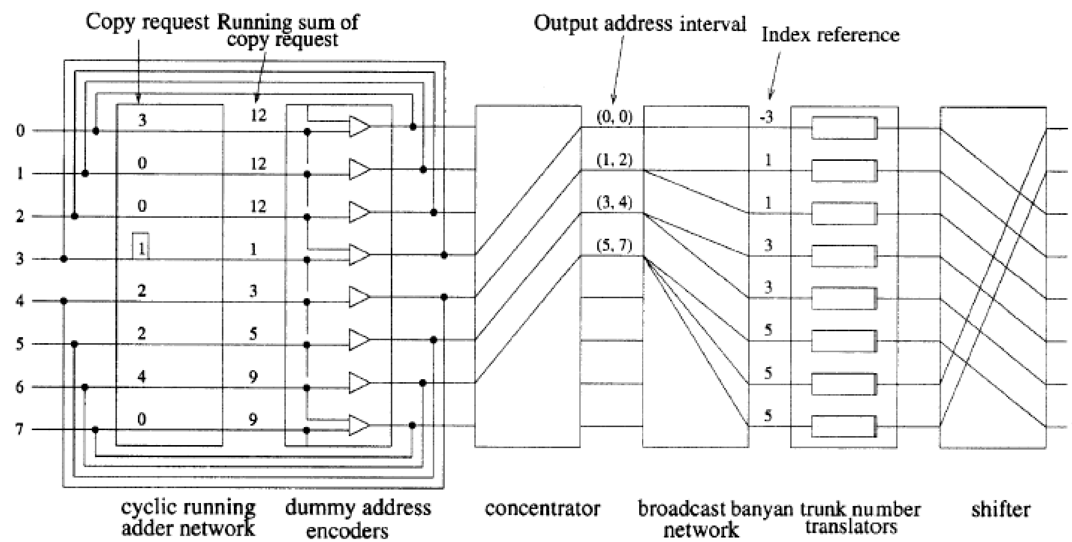


MULTICAST COPY NETWORKS

Overflow

- Copy network may not be able to do all copy requests
- An example
- Overflow problems
  - Performance
  - Unfairness



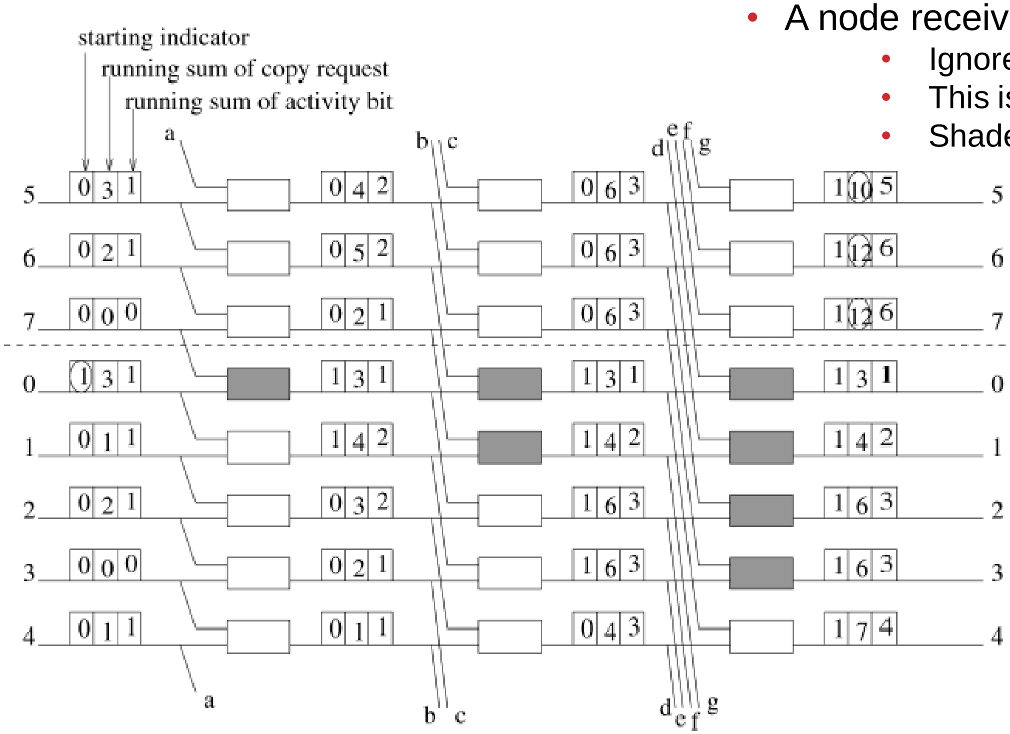


- Unfairness problem
  - Lower numbered inputs will have less overflow
  - Solution: CRAN instead of RAN
    - Adaptively changes RAN sum starting point

MULTICAST COPY NETWORKS

CRAN

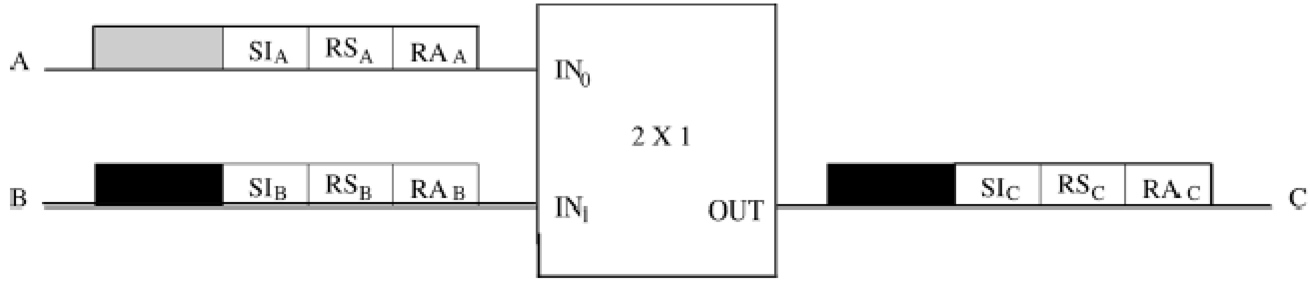
- Cell header fields
  - Starting indicator (SI)
  - Running sum (RS)
  - Routing address (RA)
- Initial values
  - SI: nonzero only for the starting point
  - RS: the number of copies
  - RA: 1 if port is active, otherwise 0
- At output, as the result:
  - RA has the running sum over activity bits
- A node receiving SI=1
  - Ignores its links
  - This is propagated
  - Shaded node in the figure



MULTICAST COPY NETWORKS

CRAN

- Header modification in a node



- The next starting point
  - No overflow → same as the previous point
  - Overflow → the first port facing the overflow
- RS updating

$$SI_0 = \begin{cases} 1 & \text{if } RS_{N-1} \leq N, \\ 0 & \text{otherwise.} \end{cases}$$

$$SI_i = \begin{cases} 1 & \text{if } RS_{i-1} \leq N \text{ and } RS_i > N, \\ 0 & \text{otherwise,,} \end{cases}$$

- SCN: starting copy number
  - Is sent back via feedback paths to input ports
  - So they know how many copies to serve

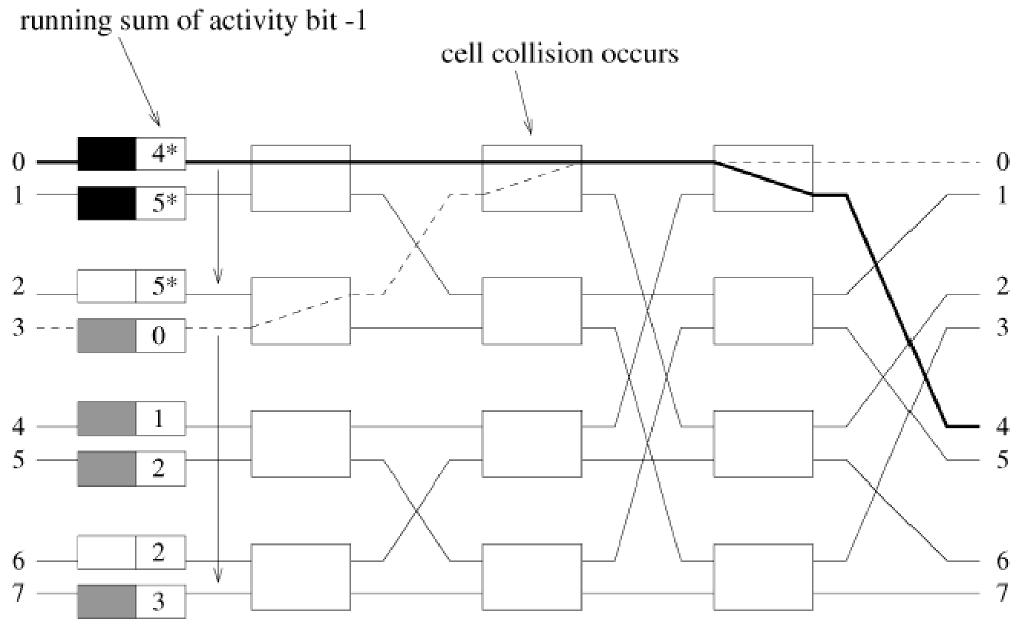
$$SCN_0 = RS_0,$$

$$SCN_i = \begin{cases} \min(N - RS_{i-1}, RS_i - RS_{i-1}) & \text{if } RS_{i-1} < N \\ 0 & \text{otherwise} \end{cases}$$

MULTICAST COPY NETWORKS

Concentration problem

- The starting point in CRAN may not be port 0
- Internal collisions may occur in reverse banyan network (RBN)



- Solution: an additional RAN before RBN