Embedded Communication Networks

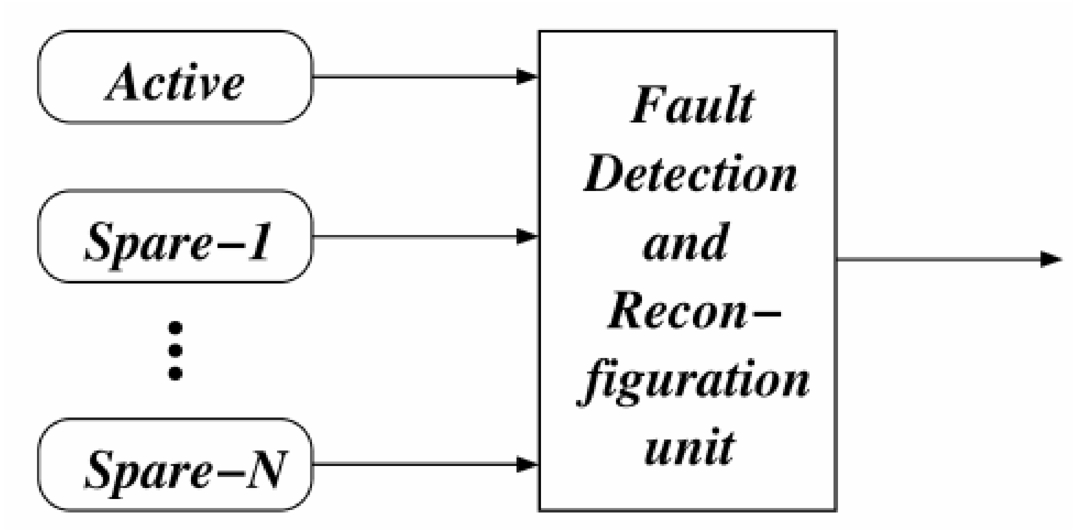
Arnab Sarkar

Advanced Technology Development Centre

IIT Kharagpur

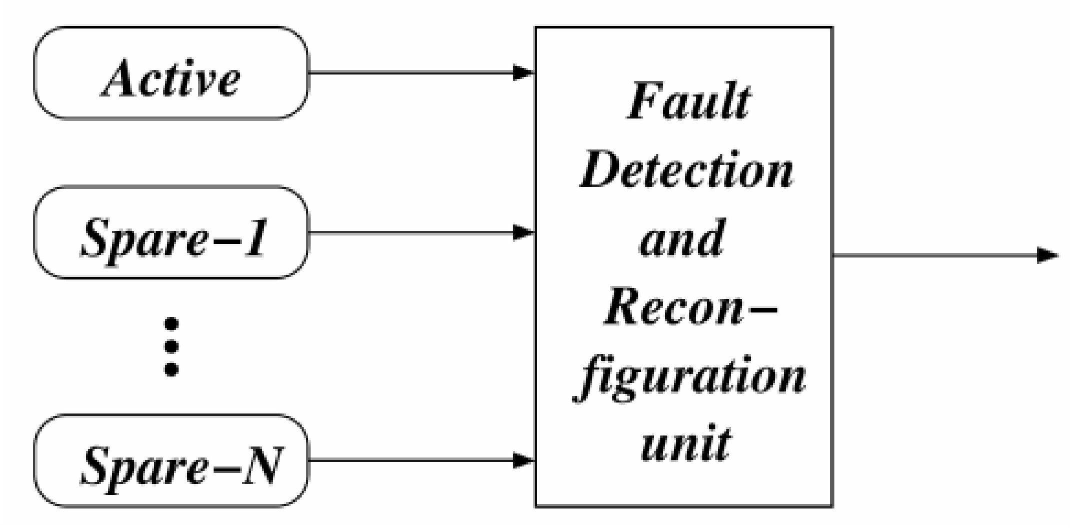
Dynamic Redundancy

- Previous variations of NMR need considerable hardware to instantaneously mask errors
- Temporary erroneous results may be acceptable if system can detect such errors and reconfigure itself by replacing the faulty module by a fault-free spare
- A dynamic redundant structure:



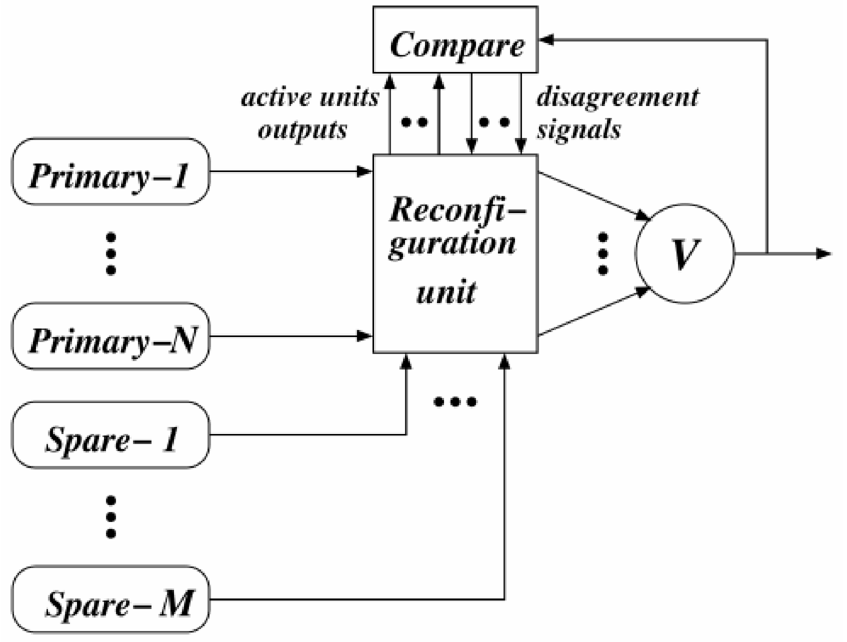
Dynamic Redundancy

- If all spare modules are active (powered) and they have the same failure rate
  - Reliability similar to a basic parallel system:
    - $R_{dynamic}(t) = R_{dru}(t)\,[1 - (1-R(t))^{N+1}]$
      - $R_{dru}(t)$ - reliability of the Detection and Reconfiguration Unit



Hybrid Redundancy

- An NMR system masks permanent and intermittent failures but its reliability drops below that of a single module for very long mission times
- Hybrid redundancy overcomes this by adding spare modules to replace active modules once they become faulty
- A hybrid system consists of a core of N processors (NMR), and M spares
- Output of primaries *compared* with *voter* to identify a faulty primary
- In case of a fault, *reconfiguration unit* replaces faulty primary with a spare

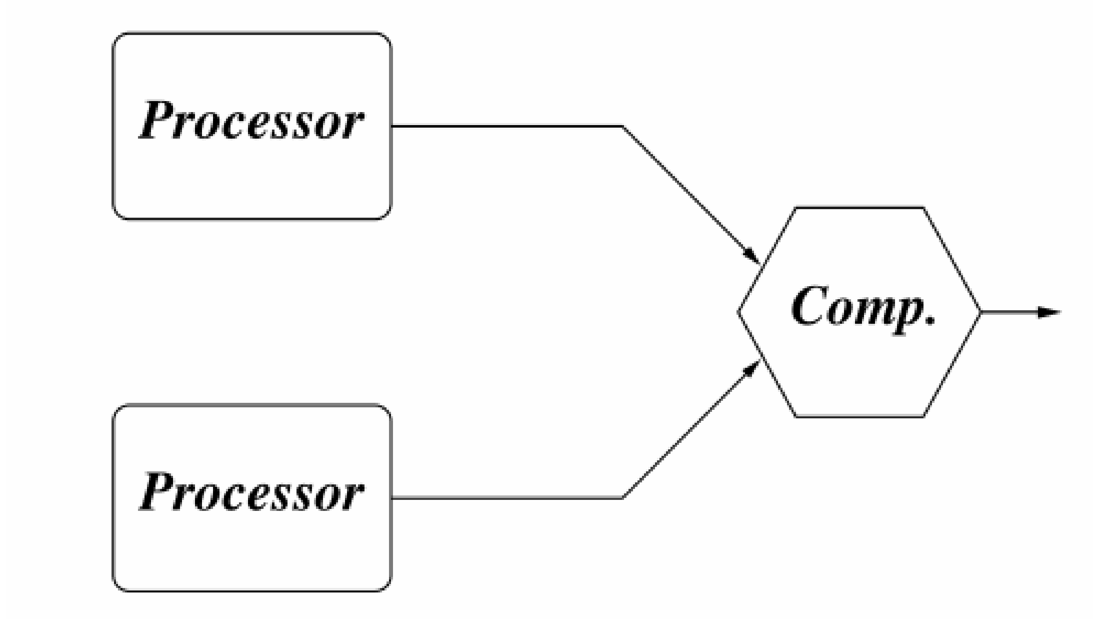


Hybrid Redundancy

- The reliability of a hybrid system with an TMR core and M spares:
  - $R_{hybrid}(t) = R_{voter}(t)\, R_{reconf}(t)\, (1 - mR(t)\, [1-R(t)]^{m-1} - [1-R(t)]^{m})$
    - m=M+3 is the total number of modules
    - $R_{voter}(t)$ and $R_{reconf}(t)$ are the reliability of voter and comparison and reconfiguration circuitry
    - Assuming that any fault in voter or comparison and reconfiguration circuit will cause a system fault
- In practice, not all faults in these circuits will be fatal: the reliability will be higher
- More accurate $R_{hybrid}(t)$ require:
  - Detailed analysis of voter and comparison & reconfiguration circuits and the ways they can fail

Duplex Systems

- Both processors execute the same task
  - If outputs are in agreement - result is assumed to be correct
  - If results are different - we can not identify the failed processor
  - A higher-level software has to decide how failure is to be handled
- This can be done using one of several methods



Duplex Systems

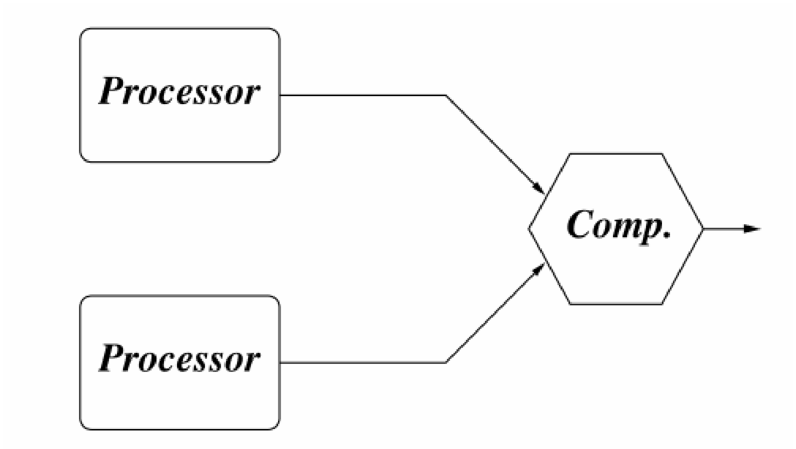
- **Acceptance Test** - a range check of each processor's output
  - Example - the pressure in a boiler must be in some known range
- We use semantic information of the task to predict which values of output indicate an error
- How should the acceptance range be picked?
- **Sensitivity** - the probability that the test will recognize an erroneous output as such
- **Specificity** - the probability that the test will identify a correct output as such
  - Narrow range - high sensitivity but low specificity
  - Wide range - low sensitivity but high specificity

Duplex Systems

- Hardware Testing: Both processors are subjected to some test
  - The processor which fails the test is identified as faulty
  - Real-life tests are never perfect
  - Test Coverage - same as test sensitivity – the probability that the test can identify a faulty processor as such
  - Test Transparency - the complement of the test coverage - the probability that the test passes a faulty processor as good
- Forward Recovery: Use a third processor to repeat the computation carried out by the duplex
  - If only one of the three processors is faulty, then the one that disagrees is the faulty one

Duplex Systems - Reliability

- Lifetime of duplex - the time until both processors fail
  - **C** - Coverage Factor - the probability that a faulty processor will be correctly diagnosed, identified and disconnected
- Rduplex(t) - the reliability of the duplex system:
  - $R_{duplex}(t) = R_{comp}(t) [R^2 (t)+2CR(t)(1-R(t)]$
    - $R_{comp}(t)$ –reliability of comparator

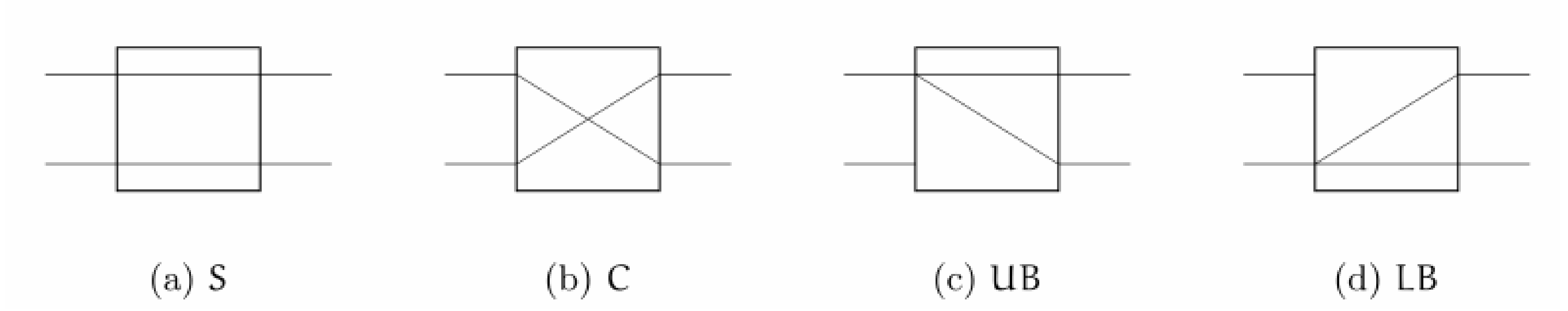


Fault-Tolerant Networks

- Multiple paths connecting the source to the destination of a message
- Spare nodes that can be switched in to replace failed units
- Fault-tolerant topologies
  - Extra-Stage Multi-Stage Networks
  - Interstitial Mesh
  - Redundant Crossbar
  - Hypercube
  - Point-to-Point Networks

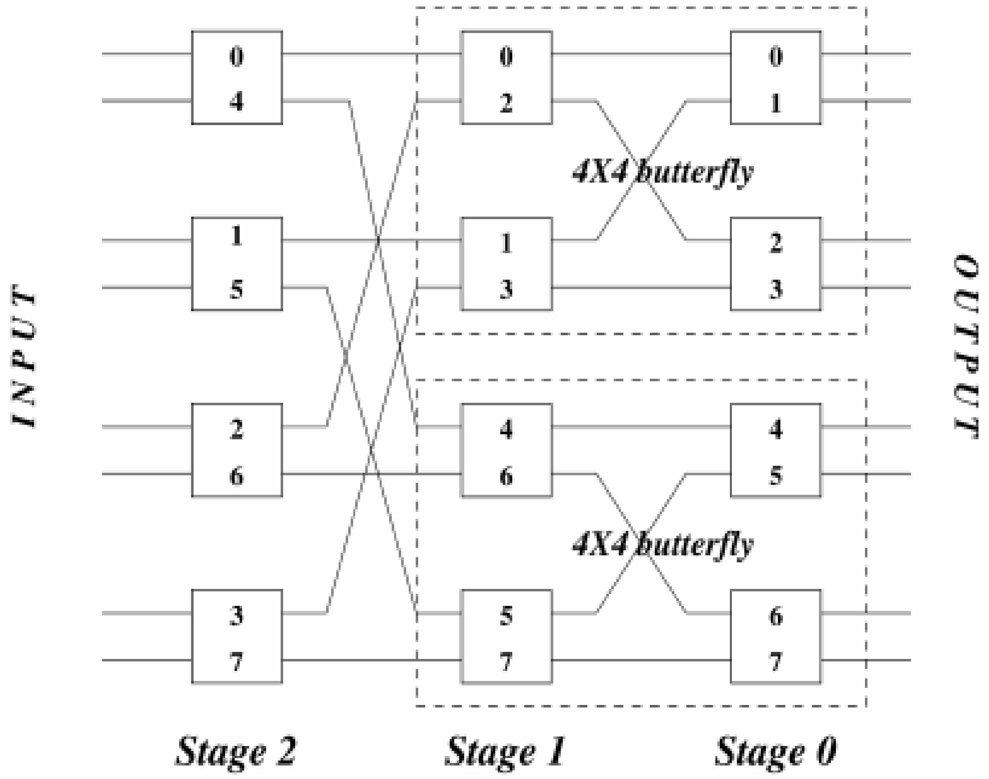
Multi-Stage Network

- Non-fault-tolerant multi-stage network (butterfly network) - typically built out of 2x2 switches - two inputs and two outputs
- Switch has four settings -
  - S - Straight - top input line connected to top output and bottom input line to bottom output
  - C - Cross - top input line connected to the bottom output and bottom input line to top output
  - UB - Upper Broadcast - top input line connected to both output lines
  - LB - Lower Broadcast - bottom input line connected to both output lines



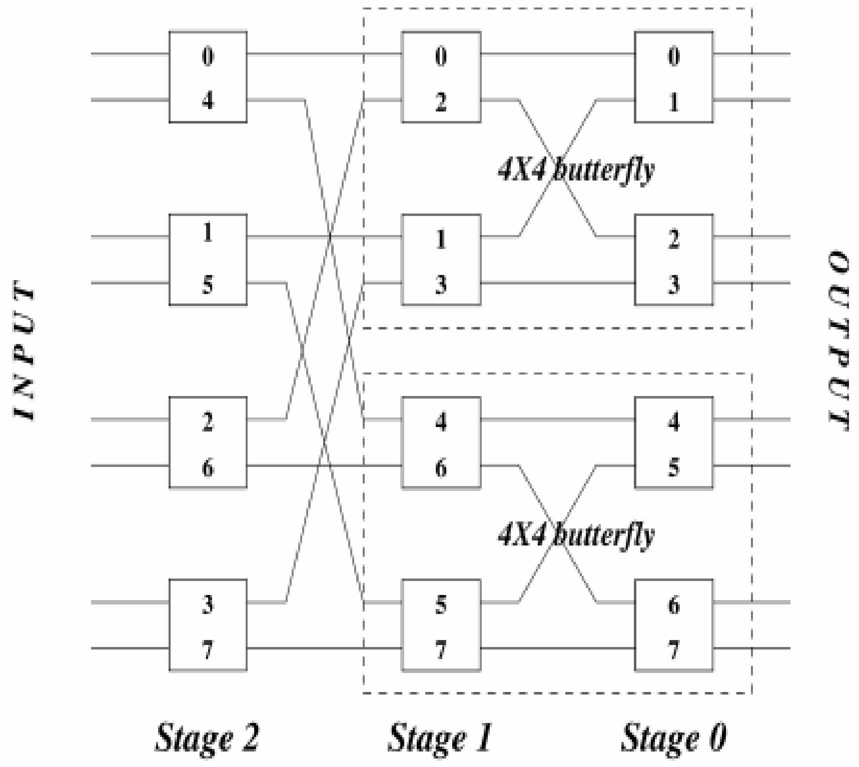
Butterfly Network

- k-stage network (k≥3)
  - 2^k inputs and 2^k outputs
  - k stages of 2^{k-1} switches each
  - Connections follow a recursive pattern from input to output
  - Input stage - top output line of each switchbox connected to the input lines of a $2^{k-1} \times 2^{k-1}$ butterfly, and the bottom output line of each switchbox connected to the input lines of another $2^{k-1} \times 2^{k-1}$ butterfly



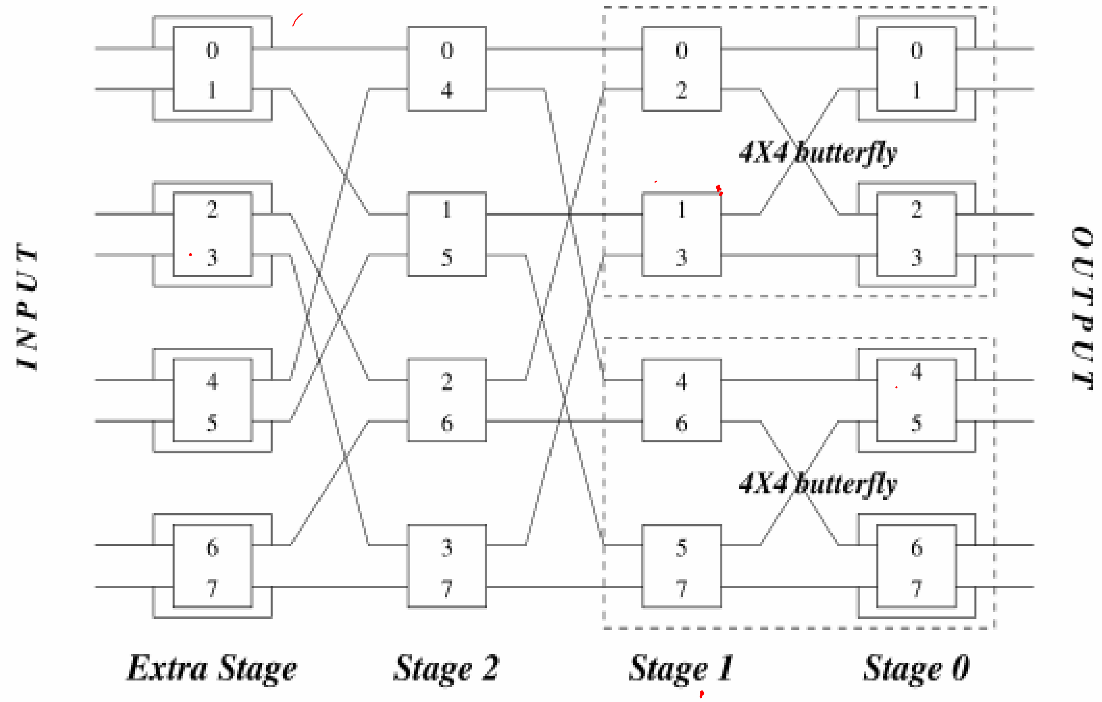
Butterfly Network - Details

- A switchbox in stage i has lines numbered 2^i apart
- Output line j of every stage goes into input line j of the following stage ($j=0,...,2^{k-1}$)
- Numbers in any box other than at the output stage are both of the same (even or odd) parity
- Butterfly is not fault-tolerant: there is only one path from any given input to a specific output
- If a switchbox in stage i fails – 2^{k-i} inputs will no longer be connected to 2^{i+1} outputs
- The system can still operate but in a degraded mode



Extra-Stage Networks - Fault Tolerant

- Extra stage – duplicating stage 0 at the input
- Bypass multiplexors around switchboxes at the input and output stages - a failed switch can be bypassed by routing around it
- Examples:
  - Stage-0 switchbox carrying lines 2,3 fails - duplicated by the extra stage - failed box is bypassed by the multiplexor
  - Switchbox in stage-2 carrying lines 0,4 fails - extra stage is set so that input line 0 is switched to output line 1 and input line 4 to output line 5 – by passing the failed switchbox



The network becomes tolerant to one switchbox failure

Measures of Resilience

- Quantify the resilience of a network or its degradation in the presence of node and link failures

- Graph-theoretic Measures
  - **Node and Link Connectivity**
    - Minimum number of nodes (links) that must be removed from the graph in order to disconnect it.
      - When a node is removed, all incident links are removed as well.
      - Higher the connectivity, more resilient the network is to faults.

    - **Diameter Stability**
      - *Distance* between a source and a destination node in a network is defined as the smallest number of links that must be traversed in order to forward a message from the source to the destination

Measures of Resilience

- Graph-theoretic Measures
  - **Diameter Stability**
    - ***Diameter*** of a network is the longest distance between any two nodes.
    - ***Diameter stability*** focuses on how rapidly the diameter increases as nodes fail
    - ***Deterministic Measure* (Persistence):** smallest number of nodes that must fail in order for the diameter to increase
      - Persistence of a cycle graph is 1: failure of just one node causes a cycle of n nodes to become a path of $n - 1$ nodes, and the diameter jumps from *floor(n/2)* to $n-2$
    - ***Probabilistic Measure***: The vector DS = $(p_{d+1}, p_{d+2}, \ldots)$, where p_{d+i} is the probability that the diameter of the network increases from d to $d+i$ as a result of faults that occur according to some given probability distribution.
      - p_∞ - probability of diameter becoming infinite (disconnected graph)

Measures of Resilience

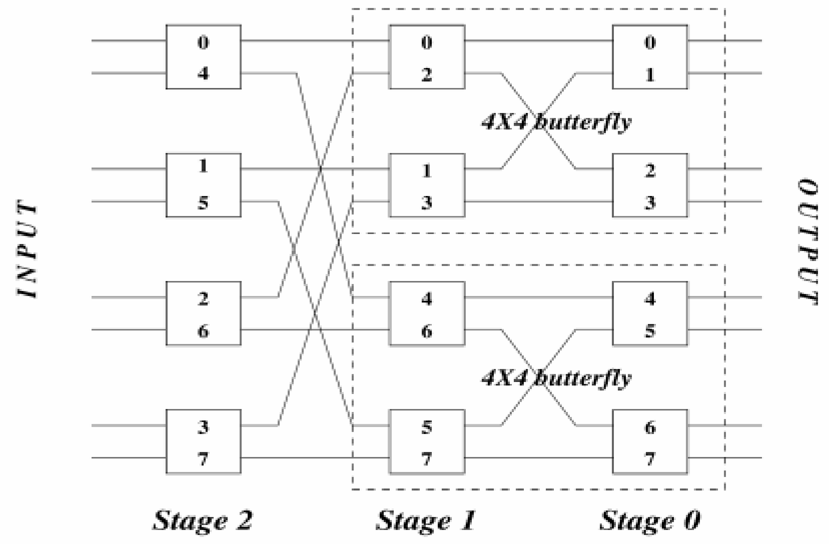
- Computer Networks Measures
  - **Reliability, R(t)**
    - Probability that all the nodes are operational and can communicate with each other over the entire time interval [0, t]
  - **Path Reliability** of a source-destination pair:
    - probability that an operational path has existed for a source–destination pair during the entire interval [0, t]
  - **Bandwidth (Meaning depends on context)**
    - Eg: Maximum rate at which messages can flow in a network
      - It may be interesting to know the expected bandwidth varies, for given failure and repair rates
  - **Connectability Q(t)**
    - Expected number at time *t* of source–destination pairs which are still connected in the presence of a failure process.
      - A more dynamic measure compared to *connectivity*

Analysis of the Butterfly Network

- *k*-stage butterfly interconnection network that connects $N = 2^k$ processors to $N = 2^k$ memory units in a shared memory architecture
- Bandwidth
  - Expected number of access requests from the processors that reach the memory modules
  - Assumptions
    - p_r: Probability of a processor requesting for memory in any cycle
    - Such requests are directed to any memory module with equal probability 1/N
    - Probability that a processor requests for a particular memory module: p_r / N
    - Each request by a processor is independent of previous requests
    - The processor generates a new independent request even if the previous request was not satisfied
  - Due to butterfly structure and uniformity assumption, all *N* outputs of a stage (say *i*), will carry memory requests with same probability $p_r^{(i)}$
  - We calculate $p_r^{(i)}$ stage by stage, starting at the inputs (processors) where, $i = k-1$, and working our way to the outputs (memories) where, $i = 0$

Analysis of the Butterfly Network

- Probability of an output of the input stage carrying memory request
  - $$p_r^{(k-1)} = \frac{p_r}{2} + \frac{p_r}{2} - \left(\frac{p_r}{2}\right)^2 = p_r - \frac{p_r^2}{4}$$
  - The requests carried by the two input lines to a switchbox are statistically independent, since the two routes they traverse are disjoint



Analysis of the Butterfly Network

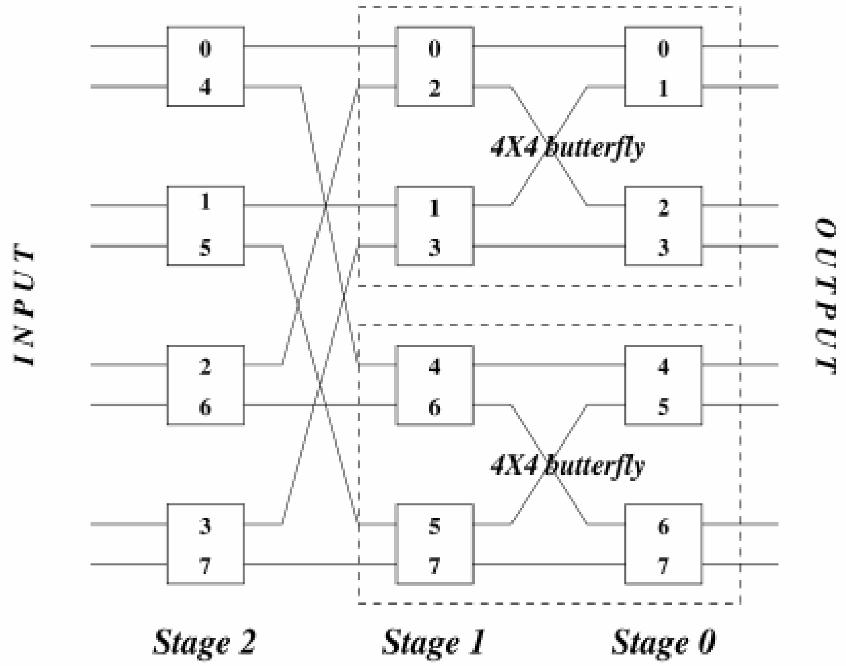
- Probability of an output of the input stage carrying memory request
  - $$p_r^{(k-1)} = \frac{p_r}{2} + \frac{p_r}{2} - \left(\frac{p_r}{2}\right)^2 = p_r - \frac{p_r^2}{4}$$
  - The requests carried by the two input lines to a switchbox are statistically independent, since the two routes they traverse are disjoint
- Probability of an output of the (i-1)th stage carrying memory request
  - $$p_r^{(i-1)} = p_r^{(i)} - \frac{(p_r^{(i)})^2}{4}$$
- Bandwidth of the network is the expected number of requests that make it to the memory end
  - $$\mathrm{BW} = N p_r^{(0)}$$

Butterfly Network with Faults

- Probability of a link being faulty: q_l
  - Probability of the link being fault-free: $p_l = 1- q_l$
- probability of switchbox failure - incorporated into incident links
  - Thus, only links can fail (Assumption)
- Prob. that a request at the input of stage ($i - 1$) will propagate from one of the corresponding inputs in stage i: $p_l \times p_r^{(i)} /2$.
  - Probability that a request at the input line at stage ($i - 1$) will propagate from any input in stage i: $p_r^{(i-1)} = p_\ell \, p_r^{(i)} - (p_\ell \, p_r^{(i)})^2/4$
- Thus, probability of the expected number of access requests from the processors that reach the memory modules in the presence of possible link failures, can be determined by recursively determining $p_r^{(0)}$: $\mathrm{BW} = Np_r^{(0)}$

Connectability

- **Connectability**: Expected number of connected processor-memory pairs in a *k*-stage, $2^k \times 2^k$ network
- There are *k+1* links and *k* switchboxes that need to be traversed in a *k*-stage network

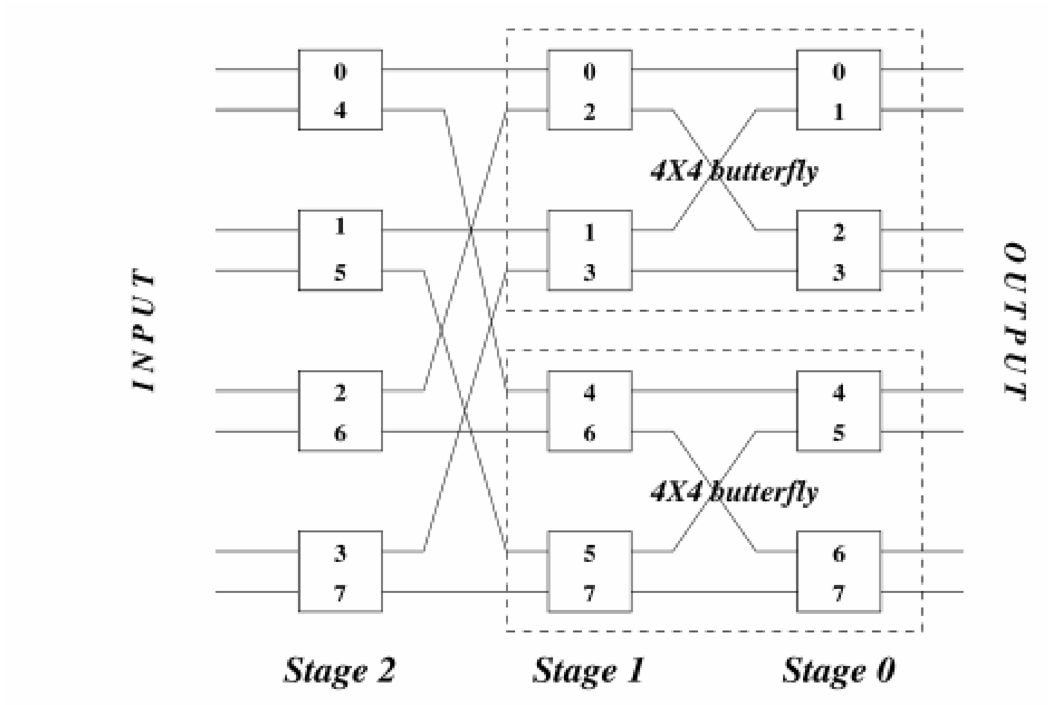


Connectability

- **Connectability**: Expected number of connected processor-memory pairs in a k-stage, $2^k \times 2^k$ network
- There are $k+1$ links and k switchboxes that need to be traversed in a k-stage network
- Probability that a switchbox fails: q_s $[p_s = 1 - q_s]$
- Probability of all links and switchboxes in a path being simultaneously fault-free: $p_l^{k+1} \times p_s^k$
- 2^{2k} input–output pairs
  - Expected number of pairs that are connected:
    - $Q = 2^{2k} \times p_l^{k+1} \times p_s^k$

Accessibility (A_c)

- A processor (memory) is accessible if it is fault-free and is connected to at least one fault-free memory (processor)
- Probability that at least one fault-free path exists from a switchbox in stage *i* to the output: $\Phi(i)$



Accessibility (A_c)

- A processor (memory) is accessible if it is fault-free and is connected to at least one fault-free memory (processor)
- Probability that at least one fault-free path exists from a switchbox in stage i to the output: $\Phi(i)$
- Probability that at least one line out of a switchbox at the output stage is functional: $\Phi(0) = 1 - q_l^2$
- Consider $\Phi(i)$ [$i > 0$]
  - A connection to the output end exists through the top link of stage-i if and only if that link is functional and the stage-(i−1) switchbox that it leads to is connected to the output end. Probability of this: $p_l\ \Phi(i\text{-}1)$
  - Therefore, $\Phi(i) = 1 - (1\text{-}\ p_l\ \Phi(i\text{-}1))^2$
- Probability that a processor can connect to the output: $p_l\ \Phi(k\text{-}1)$
- Since, there are 2^K processors: $A_c = 2^K p_l\ \Phi(k\text{-}1)$