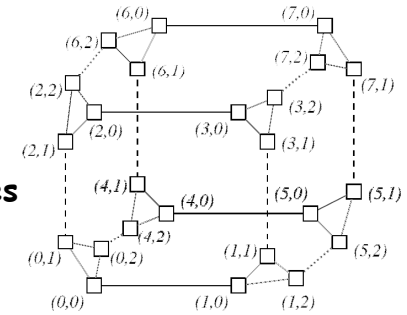
FAULT TOLERANT SYSTEMS

http://www.ecs.umass.edu/ece/koren/FaultTolerantSystems

Part 13 - Networks – 3

Chapter 4: Network Fault Tolerance

Cube-Connected Cycles Networks

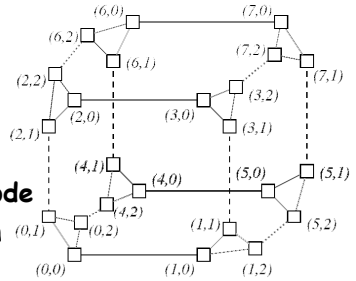


- Hypercube :
  - multiple paths between nodes
  - low diameter
  - cost - a high node degree
  - n ports - new node design required when size of network increases
- Cube-Connected Cycles (CCC) Network :
  - Keeps degree of a node fixed at 3 or less (for any n)
- Example: A CCC network that corresponds to the H3 hypercube
  - Each node of degree 3 in H_3 is replaced by a cycle consisting of 3 nodes

General CCC Network

- Each node of degree n in the hypercube Hn is replaced by a cycle containing n nodes where the degree of every node in the cycle is 3
- The resulting CCC(n,n) network has $n \cdot 2^n$ nodes
- We may have a CCC(n,k) network with $k \cdot 2^n$ nodes - each cycle includes k≥n nodes with the additional k-n nodes having a degree of 2
- The extra nodes of degree 2 have a very small impact on the network properties
- We will restrict ourselves to k=n

Labeling CCC Nodes



- CCC node label - (i;j)
  - i - an n-bit binary number - label of corresponding hypercube node
  - j (0≤j≤n-1)- position of node within cycle
- Two nodes (i;j) and (i';j') are linked by an edge in the CCC if and only if
  - either i = i' and j - j' = ± 1 mod n (link along the cycle)
  - or j = j' and i differs from i' in precisely the jth bit (dimension-j edge in the hypercube)
- Example: Nodes 0 and 2 in H_3 connected through a dimension-1 edge that corresponds to the edge connecting nodes (0,1) and (2,1)

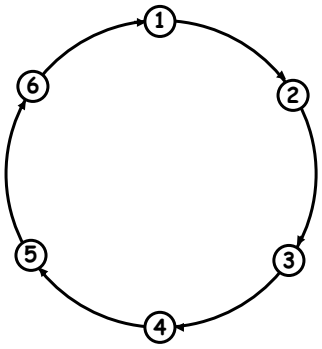
Hypercube vs.CCC

- CCC has lower degree of nodes compared to hypercube
- CCC has a higher diameter than hypercube
- Hypercube has a diameter of n
- The CCC(n,n) has a diameter of

$$2n + \left\lfloor \frac{n}{2} \right\rfloor - 2 \approx 2.5n$$

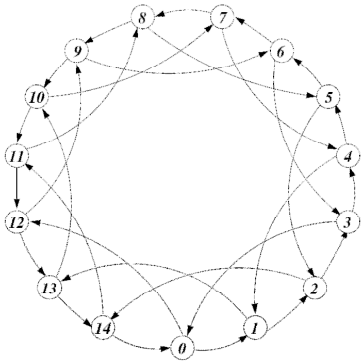
- Routing of messages in the CCC is more complicated than in the hypercube
- Fault tolerance of the CCC is higher - failure of a single node in the CCC is similar to a single faulty link in the hypercube
- No closed form expression for reliability of CCC

Loop Networks



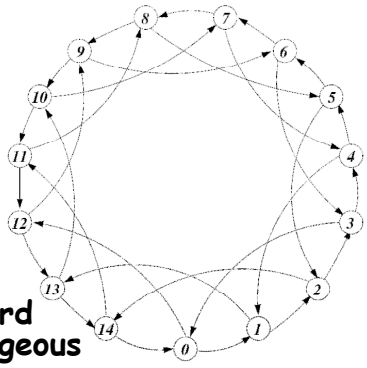
- The cycle topology (also called loop network) that is replicated in the CCC network can serve as an interconnection network
- Advantages:
  - simple routing algorithm
  - small node degree
- Disadvantages:
  - an n-node unidirectional loop has a diameter of n-1 - an average of n/2 intermediate forwarding nodes per message
  - unidirectional loop network not fault tolerant – a single node or link failure disconnects the network

Chordal Networks



- Reduce diameter and improve fault tolerance of a unidirectional loop network by adding extra links - called chords
- Each node has an additional backward link connecting it to a node at a distance s, called the skip distance
- Node i ($0 \le i \le n-1$) has a forward link to node $(i+1) \bmod n$ and a backward link to node $(i-s) \bmod n$
- Degree of every node is 4 for any value of n
- Example: 15-node chordal network with a skip distance of 3

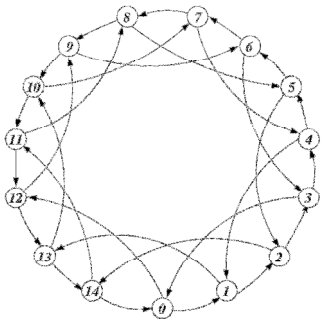
Minimizing Diameter of Chordal Networks



- Choice of s affects diameter D
- Looking for s that minimizes D
- Diameter - longest distance from source to destination - depends on routing
- Assumption: Routing uses backward chords as long as this is advantageous
- b - number of backward chords used
- bs - number of nodes skipped
- b' – maximum value of b
- b's +b' ≥ n

$$b' = \left\lfloor \frac{n}{s+1} \right\rfloor$$

Minimizing Diameter – Cont.



- We may need to add a maximum of **s-1** forward links to **b'**

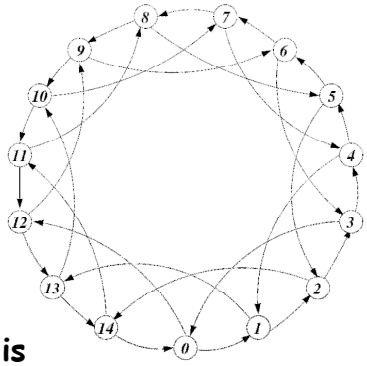
$$D = \left\lfloor \frac{n}{s+1} \right\rfloor + (s-1)$$

- For most values of **n**, $s = \lfloor \sqrt{n} \rfloor$ minimizes **D** and

$$D_{opt.} \approx 2\sqrt{n} - 1.$$

- **Example**:
  - n=15 ;
  - optimal s = 3
  - minimal D = 5

Reliability of Chordal Networks



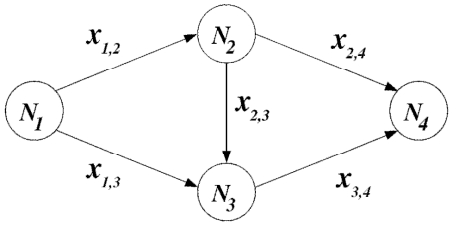
- Exact reliability calculation - complicated
- Instead - calculate number of paths between the two farthest nodes in network
- If the number of these paths is maximized - reliability is likely to be high
- The minimum length between the two farthest nodes is **b'+s-1**
- The number of paths of length **b'+s-1** consisting of **b'** backward chords and **(s-1)** forward links is $\binom{b'+s-1}{s-1}$

Increasing Reliability of Chordal Networks

- **s** that maximizes $\binom{b'+s-1}{s-1}$ is $s = \lceil \sqrt{n} \rceil$
- However, for most values of **n**, $s = \lfloor \sqrt{n} \rfloor$ yields the same number of paths
- **Conclusion:** In most cases, the value of **s** that minimizes the diameter also maximizes the number of alternate paths
- $s = \lfloor \sqrt{n} \rfloor$ should be selected in order to improve the reliability of the network

Reliability of Point-to-Point Networks

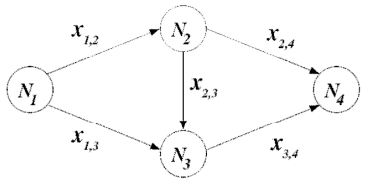
- Not necessarily a regular structure - often more than one path between any two nodes
- **Path (or Terminal) Reliability** - the probability that there exists an operational path between two specific nodes, given the probabilities of link failures



- **Example** - calculating the path reliability for the source-destination pair N_1 - N_4

Path Reliability - Example I

- Three paths from N_1 to N_4
  - $P_1=\{X_{1,2},X_{2,4}\}$
  - $P_2=\{X_{1,3},X_{3,4}\}$
  - $P_3=\{X_{1,2},X_{2,3},X_{3,4}\}$



- $P_{i,j}$ $(q_{i,j})$ - probability that link $X_{i,j}$ is good (faulty)
- Probability of node failure - incorporated into failure probability of outgoing links
- Events {path P_i is operational} must be modified to an equivalent set of mutually exclusive events - otherwise some cases will be counted more than once
- Mutually exclusive events - (I) P_1 is up ; (II) P_2 is up and P_1 is down ; (III) P_3 is up and both P_1 and P_2 are down

$$R_{N_1,N_4} = p_{1,2}p_{2,4} + p_{1,3}p_{3,4}[1 - p_{1,2}p_{2,4}] + p_{1,2}p_{2,3}p_{3,4}[q_{1,3}q_{2,4}]$$

Calculating Path Reliability - General

- Path reliability of a network with m paths $p_1,\ldots,p_m$ from source Ns to destination Nd
- E_i $(\overline{E}_i)$ - event in which P_i is operational (faulty)
- R_{NsNd} = Prob{Operational Path Exists} = Prob{ $\cup$ Ei }
- m events decomposed into mutually exclusive events -

$$E_1 \cup E_2 \cup \cdots \cup E_m = E_1 \cup (E_2 \cap \overline{E_1}) \cup (E_3 \cap \overline{E_1} \cap \overline{E_2}) \cup \cdots \cup (E_m \cap \overline{E_1} \cap \overline{E_2} \cap \cdots \cap \overline{E_{m-1}})$$

and

$$R_{N_s,N_d} = \text{Prob}\{E_1\} + \text{Prob}\{E_2 \cap \overline{E_1}\} + \cdots + \text{Prob}\{E_m \cap \overline{E_1} \cap \overline{E_2} \cap \cdots \cap \overline{E_{m-1}}\}$$

or using conditional probabilities

$$R_{N_s,N_d} = \text{Prob}\{E_1\} + \text{Prob}\{E_2\}\text{Prob}\{\overline{E_1} \mid E_2\} \cdot + \cdots + \text{Prob}\{E_m\}\text{Prob}\{\overline{E_1} \cap \overline{E_2} \cap \cdots \cap \overline{E_{m-1}} \mid E_m\}$$

Calculating Conditional Probabilities

$$\text{Prob}\{\overline{E_1} \cap \cdots \cap \overline{E_{i-1}} \mid E_i\} = \text{Prob}\{\overline{E_{1|i}} \cap \cdots \cap \overline{E_{i-1|i}}\}$$

$\overline{E_{j|i}}$ - the event in which P_j is faulty given that P_i is operational

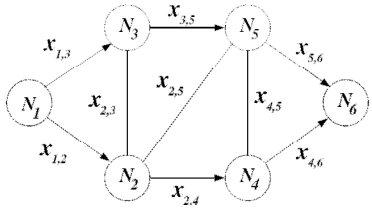
- To identify the links which must fail so that E_i occurs but not E_j, conditional sets are used

$$P_{j|i} = P_j - P_i = \{x_k \mid x_k \in P_j \text{ and } x_k \notin P_i\}$$

- At least one link in $P_{j|i}$ needs to fail

Path Reliability - Example II

- This six-node network has 9 links - 6 uni-directional and 3 bi-directional
- All paths from N1 to N6 -



$P_1 = \{x_{1,3}, x_{3,5}, x_{5,6}\}$
$P_2 = \{x_{1,2}, x_{2,5}, x_{5,6}\}$
$P_3 = \{x_{1,2}, x_{2,4}, x_{4,6}\}$
$P_4 = \{x_{1,3}, x_{3,5}, x_{4,5}, x_{4,6}\}$
$P_5 = \{x_{1,3}, x_{2,3}, x_{2,4}, x_{4,6}\}$
$P_6 = \{x_{1,3}, x_{2,3}, x_{2,5}, x_{5,6}\}$
$P_7 = \{x_{1,2}, x_{2,5}, x_{4,5}, x_{4,6}\}$
$P_8 = \{x_{1,2}, x_{2,3}, x_{3,5}, x_{5,6}\}$
$P_9 = \{x_{1,2}, x_{2,4}, x_{4,5}, x_{5,6}\}$
$P_{10} = \{x_{1,3}, x_{2,3}, x_{2,4}, x_{4,5}, x_{5,6}\}$
$P_{11} = \{x_{1,3}, x_{2,3}, x_{2,5}, x_{4,5}, x_{4,6}\}$
$P_{12} = \{x_{1,3}, x_{3,5}, x_{2,5}, x_{2,4}, x_{4,6}\}$
$P_{13} = \{x_{1,2}, x_{2,3}, x_{3,5}, x_{4,5}, x_{4,6}\}$

- Paths are ordered from shortest to longest

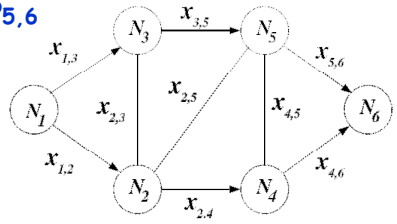
Calculating Path Reliability for Example II

- The first term for the reliability equation is Prob{E1} = P1,3 P3,5 P5,6
- To calculate the second term in the reliability equation - the conditional set is used

$$P_1 = \{x_{1,3}, x_{3,5}, x_{5,6}\}$$

$$P_2 = \{x_{1,2}, x_{2,5}, x_{5,6}\}$$

$$P_{1|2} = P_1 - P_2 = \{x_{1,3}, x_{3,5}\}$$



- At least one of the links in this set must fail so that P1 is faulty given that P2 is operational
- The second term in the probability equation - P1,2 P2,5 P5,6(1 - P1,3 P3,5)

Example II - Cont.

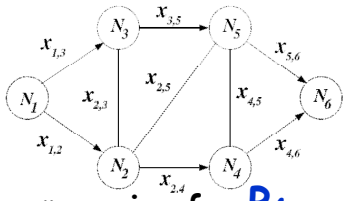
- For calculating other terms in the sum - intersection of several conditional sets must be considered

$$P_1 = \{x_{1,3}, x_{3,5}, x_{5,6}\}$$

$$P_2 = \{x_{1,2}, x_{2,5}, x_{5,6}\}$$

$$P_3 = \{x_{1,2}, x_{2,4}, x_{4,6}\}$$

$$P_4 = \{x_{1,3}, x_{3,5}, x_{4,5}, x_{4,6}\}$$

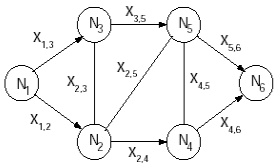


- Calculating the fourth term - expression for P4 - the conditional sets are: P1|4 = {X5,6}; P2|4 = {X1,2 ,X2,5,X5,6}; P3|4 = {X1,2 ,X2,4}
- P1|4 is included in P2|4 - if P1|4 is faulty, so is P2|4 - P2|4 can be ignored
- The fourth term in the reliability equation - P1,3 P3,5 P4,5 P4,6 (1 - P5,6) (1 - P1,2 P2,4)

Example II – Cont.

- Calculating the third term:

$$P_1 = \{x_{1,3}, x_{3,5}, x_{5,6}\}$$
$$P_2 = \{x_{1,2}, x_{2,5}, x_{5,6}\}$$
$$P_3 = \{x_{1,2}, x_{2,4}, x_{4,6}\}$$



$$P_{1|3} = \{x_{1,3}, x_{3,5}, x_{5,6}\},\ P_{2|3} = \{x_{2,5}, x_{5,6}\}$$

- The event both $P_{1|3}$ and $P_{2|3}$ are faulty needs to be divided into disjoint cases:
- (I) $X_{5,6}$ is faulty
- (II) $X_{5,6}$ is operational and both $X_{1,3}$ and $X_{2,5}$ are faulty
- (III) $X_{5,6}$ and $X_{1,3}$ are up, and $X_{3,5}$ and $X_{2,5}$ are faulty
- Resulting expression for third term

$$p_{1,2}p_{2,4}p_{4,6}[q_{5,6} + p_{5,6}q_{1,3}q_{2,5} + p_{5,6}p_{1,3}q_{3,5}q_{2,5}]$$

- Remaining terms - calculated similarly
- Path reliability is the sum of all thirteen terms

Alternative Calculation of Path Reliability

- A given number of components (links)
- Each component can be up or down
- We need to calculate the probability that certain combinations of the components are all up
- In the last example - 9 links with 2^9 states
- Probability of each state obtained by multiplying 9 factors of the form $p_{i,j}$ or $q_{i,j}$
- We add up the probabilities of all the states in which a path from node N1 to node N6 exists
- The sum is the path reliability $R_{N1,N6}$

Fault-Tolerant Routing

- Objective: get a message from source to destination despite a subset of the network being faulty
- Basic idea: if no shortest or most convenient path is available because of failures, reroute message through other paths to destination
- Focus on unicast routing - a message is sent from a source to just one destination
- Multicast - copies of a message sent to a number of nodes - is an extension of the unicast problem

Classification of Routing Algorithms

- Centralized routing
  - A central controller knows the network state – faulty links or nodes, congested links - and selects path for each message
  - Variation: Source of message specifies the route for that message
- Distributed routing
  - No central controller – each intermediate node decides to which node to send it next
- Unique routing –
  - One path for each source-destination pair
- Adaptive routing
  - Path selected according to network conditions (congestion)
- Implementing fault tolerance in centralized routing
  - A centralized router that knows the state of each link
  - Can use graph-theoretic algorithms to determine all paths
  - Secondary considerations (load balancing, number of hops) can be used to select the path

Routing in Injured Hypercubes

- Routing algorithm must be modified to route around the faulty nodes or links
- Basic idea - list the dimensions along which the packet must travel, and traverse them one by one
- As edges are traversed and are crossed off the list
- If, due to a link or a node failure, the desired link is not available - another edge in the list, if any, is chosen for traversal
- If packet arrives at some node to find all dimensions on its list down - it backtracks to the previous node and tries again

Formal Routing Algorithm - Notations

- TD - list of dimensions that the message has traveled on - in order of traversal; TD^R - in reversed order
- $\oplus_{i=1}^{k}$ - exclusive-or operation carried out k times, sequentially
- Example - $\oplus_{i=1}^{3} a_i$ means $(a_1 \oplus a_2) \oplus a_3$
- D - destination, S - source, d=D⊕S (⊕ - bitwise exclusive-or operation on corresponding bits of D and S)
- SC(A) - set of nodes visited if we travel on each of the dimensions listed in set A
- Example - at node 0010 - SC(1,3)={0000,1000}

Notations - Cont.

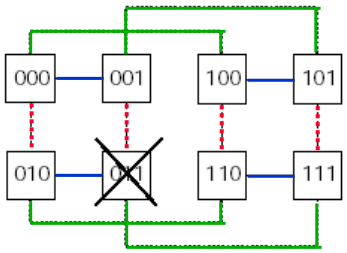
- e_n^i - n-bit vector consisting of a 1 in the i-th bit position and 0 everywhere else
- Example - e_3^2 = 100
- Packets are assumed to consist of
  - (I) d; d=D⊕S
  - (II) Message being transmitted (the "payload")
  - (III) List of dimensions taken so far - TD
- θ - append operation
  - TD θ x - append x to the list TD
- transmit(j) - send packet (d ⊕ e^j, message, TD θ j) along the j-th-dimensional link from the present node

Routing Algorithm for Injured Hypercubes

```
If (d == 0···0)
    the destination has been reached. Exit.
else
    for j = 0 to (n − 1) step 1 do {
        if ((d_j == 1) && (j'th dimension link from this node is nonfaulty)
            && (e_n^j ∉ SC(TD^R)) {
            TRANSMIT(j)
            exit
        }
    }
end if
if (there is a non-faulty link not in SC(TD^R))
    Let h be one such link.
else {
    g = max{m : ⊕_{i=1}^{m} e^{TD^R}(i) == 0···0}
    if (g==number of elements in SC(TD)) {
        A path does not exist.
        exit
    }
    else
        h = (g + 1)' st element in TD^R.
    end if
    TRANSMIT(h)
end
```

Example of Routing in H_3

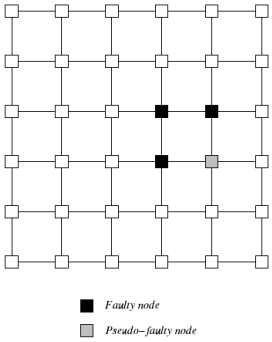
- H_3 with faulty node 011
- node 000 wants to send a packet to 111
- At 000, d=111 - sends the message out on dimension-0, to node 001
- At 001, d = 110 and TD=(0) - attempts dimension-1 edge - impossible
- Bit 2 of d is also 1 - checks and finds that the dimension-2 edge to 101 is available - message is sent to 101 and then to 111
- Exercise - What if both 011 and 101 are down?



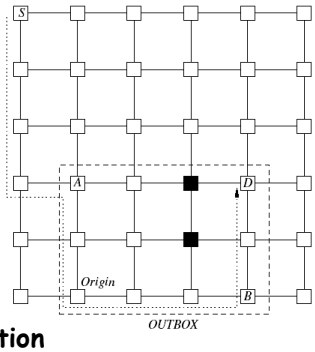
Dimension-0 edges
Dimension-1 edges
Dimension-2 edges

Origin-Based Routing in Mesh

- Depth-first strategy:
  - No advance information about faulty nodes
  - Backtracks if it arrives at a dead-end
- If faulty regions are known in advance - no backtracking is necessary
- Topology - two-dimensional NxN mesh with at most N-1 failures
  - Can be extended to meshes of dimension ≥3 and more than N-1 failures
- Assumptions:
  - All faulty regions are square, if not, additional nodes are declared to have pseudo faults
  - Each node knows the distance along each direction to the nearest faulty region in that direction
  - One node defined as the origin
  - Origin chosen so that its row and column do not have any faulty nodes
    - » possible since there are no more than N-1 failures



Origin-Based Routing Algorithm - Definitions



- Sending a message from node S to node D
- Definitions:
- IN-path - edges that take the message closer to the origin
- OUT-path – takes the message farther away from the origin (either may be empty)
- Outbox is the smallest rectangular region that contains both the origin and the destination
- V is a safe node with respect to D and a set of faulty nodes F if
  - V is in the outbox for D
  - there exists a fault-free OUT-path from V to D
- Diagonal band for D - all nodes V in the outbox such that

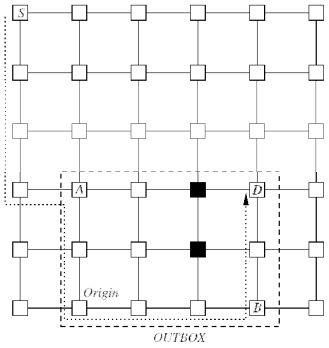
$$x_V - y_V = x_D - y_D + e \qquad e \in \{-1, 0, 1\}$$

- Once we get to a safe node, there exists an OUT-path from that node to D
- Each step along an OUT-path increases the distance to the origin - the message cannot be traveling forever in circles

Origin-Based Routing Algorithm

- Three phases :
- Phase 1 : The message is routed on an IN path until it reaches the outbox, at node U
- Phase 2. : Compute the distance from U to the nearest safe node and compare to the distance to the nearest faulty region in that direction
- If the safe node is closer than the fault, route to the safe node; otherwise, continue to route on the IN links
- Phase 3.Once the message is at a safe node U, if there is a safe nonfaulty neighbor V that is closer to the destination, send it to V ; otherwise, U must be on the edge of a faulty region
- In such a case, move the message along the edge of the faulty region toward the destination D, and turn toward the diagonal band when it arrives at the corner of the faulty square

Origin-Based Routing - Example



- Routing a message from node S at the northwest end of the network to D
- The message first moves along the IN links, getting closer to the origin
- It enters the outbox at node A
- Since there is a failure directly east of A, it continues on the IN links until it reaches the origin
- Then it continues, skirting the edge of the faulty region until it reaches node B
- At this point, it recognizes the existence of a safe node immediately to the north and sends the message through this node to the destination