Reliability Evaluation Techniques

- ***Reliability*** refers to the degree of tolerance against errors and component failures in a system.

- A reliable system prevents loss of information even in the event of component failures.

- The multiplicity of storage devices and processors in a distributed system allows the maintenance of multiple copies of critical information within the system and the execution of important computations redundantly to protect them against catastrophic failures.

  E.g. if one of the processors fails, the computation can be successfully completed at the other processor, or if one of the storage devices fails, the information can still be retrieved from the other storage device.

- Distributed computing systems have a potential for higher reliability since a few parts of the system can be down without interrupting the jobs of the users who are using the other parts of the system. This is in contrast to the centralized system, where a failure of the central processor results in the entire system shutting down.

  E.g. if a workstation of a distributed computing system that is based on the workstation-server model fails, only the user of that workstation is affected.

- The advantage of higher reliability is an important reason for use of distributed computing systems for mission-critical applications whose failure may be disastrous.

Ways to represent reliability

There are two ways to represent reliability:

1. Network point of view
2. Application point of view

Network Point of View

- Given the link and node reliability for a specific network topology, determine the probability of the network becoming partitioned.

There are two popular parameters used to evaluate network reliability:

1. Source-terminal reliability (STR)
   This is the probability that a path exists between a given source node, *s*, and a given destination node, *d*.

2. Computer Network Reliability (CNR)
   This is the probability that a path exists between **any** source node and **any** destination node (i.e. the probability that any node in the network can communicate with any other node in the network).

Application Point of View

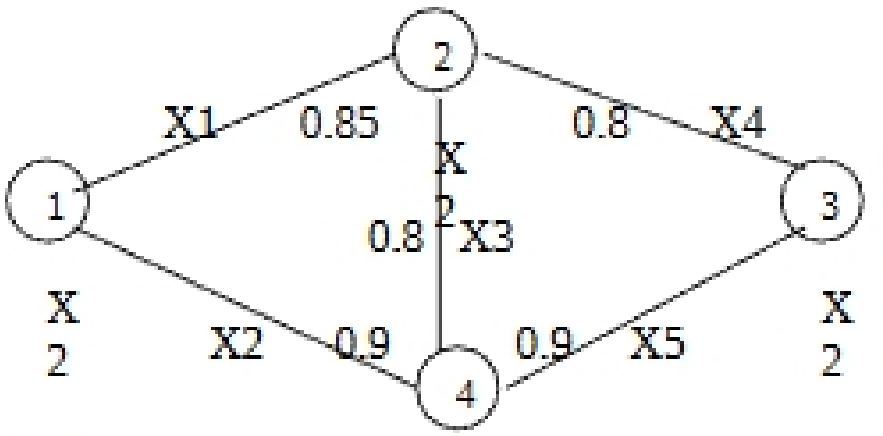
Given the link and processing node reliabilities, and the distribution of programs as well as data files, determine the probability of successfully completing a specific application.

A popular parameter used to evaluate reliability from the application execution standpoint is:

Distributed Program Reliability (DPR)
This is the probability that a program can successfully execute in a distributed computing system (i.e. the program can always access the data files it needs to successfully execute).

Computing the STR

Consider the network topology given below:



Assumptions:

1. Each link is full-duplex when working.
2. Each link can is either in a working state or in a failed state.
3. The reliability of every possible link is known.
4. The nodes are perfectly reliable (this assumption is for mathematical convenience only. The failure of a node can be easily modeled by assuming the failure of all links that connect to the node that has failed).

Step 1:

Determine all possible paths from source node 1 to destination node 3. The possible paths (in increasing order of cardinality) are: X1 X4, X2 X5, X1 X3 X5, and X2 X3 X4. These are depicted below in order:

X1 X4 X2 X5 X1 X3 X5 X2 X3 X4

Step 2:

Reliability Expression Generation by making the terms disjoint:

Term 1: X1 X4 ⟶ P1 P4

Term 2: XI X4 (these are both not present in the second path X2 X5)

Z(X1 X4) = X1' + X1 X4'

X2 X5 (X1' + X1 X4') ⟶ P2 P5 (Q1 + P1 Q4)

Term 3: X4 + X2

Z(X4 + X2) = X4' X2'

X1 X3 X5 X4' X2' ⟶ P1 P3 P5 Q4 Q2

Term 4: X1 + X5 + X1 X5 ⟶ X1 (1 + X5) + X5 ⟶ X1 + X5