International Journal of Advanced Computer and Mathematical Sciences ISSN 2230-9624.
Vol 2, Issue 3, 2011, pp 126-140 http://bipublication.com

DATA COLLECTION AND DISSEMINATION IN DISTRIBUTED ENVIRONMENTS BASED ON ROBUST SPANNING TREE APPROACH

[1]Chetan S.Arage, [2]Ali A. Bagwan, [3]Pallavee A. Patil
[1]Dept. of Computer Science and Technology, Shivaji University, Kolhapur
[2]Dept. of Computer Science, Rajarshi Shahu College of Engineering, Tathawade, Pune, INDIA
[3]Dept. of Information Technology, T.K.I.E.T. Warananagar. MS. INDIA

ABSTRACT
Large-scale distributed applications are subject to frequent disruptions due to resource contention and failure. Such disruptions are inherently unpredictable and therefore robustness is a desirable property.
This work consists of description and evaluation of a robust topology for applications that operate on a spanning tree overlay network. Unlike a conventional adaptive or reactive technique, a proactive approach to robustness is attempted. This would be followed by a demonstration of its effectiveness through simulation and analysis.
Spanning trees are widely used in communication networks as a means to disseminate information from one node to all other nodes and/or to collect information at a single designated node. The robust topology itself is able to simultaneously withstand disturbances and exhibit good performance.
We demonstrated its effectiveness by simulation and comparing expected data loss and power consumption in the robust approach as opposed to existing approaches.

Keywords: Distributed computing, Graph theory, Fault tolerance, Robust, Robustness, simulation and analysis.

1. INTRODUCTION
A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A Distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.

The design and implementation of distributed computing systems has historically proved that performance is the major goal in accordance with the system perspective. Typically the objective is to optimize a criterion such as response time, makespan, or hit rate. Furthermore, the performance metrics are usually viewed from an individual perspective and may not correspond to the social optima [1].

The importance of robustness in the design of complex and distributed systems is well established [1–7]. Biological systems naturally form robust topologies that are resilient to attack [5], [6]. Distributed systems may operate in an environment that undergoes unpredictable changes causing certain system performance features to degrade. Such systems need robustness to guarantee limited degradation despite fluctuations in the behavior of its component parts or environment [8]. Social networks exhibit the small-world phenomenon in which any two members are separated by just a few acquaintances. This phenomenon can be viewed as a form of robustness since information reaches every person in the network very quickly despite the fact that some people will not pass on the information [1].

Essential focus must be given on the ability of distributed computing systems to maintain performance despite the presence of various anonymous conditions and perturbations. This should be a fundamental concept with an eye towards design of distributed systems. Particularly when network bandwidth and computational resources are shared, robustness is a desirable system property because communication delays and execution times are inherently difficult to predict. Whenever there is a choice in the design of distributed system For example, a network topology; an influence of the system's response to various disturbances in the topology of an overlay network can be achievable. There is often a trade-off for incorporating robustness into a system. However, it is needed to show that the price paid in performance loss is well worth that gained when the operating environment is unpredictable.

How to measure robustness is a subject that has not yet lead to a wide accepted metric. Several works propose different ways to measure this metric:

1) Defining the performance features that need to be robust,

2) Identify the parameter that impacts the robustness. Mostly it includes data loss, power consumption and communication delay.

S.D.Gribble [2] argued that a common design paradigm for complex systems (careful design based on a prediction of the operating environment, load, and failures that the system will experience) is fundamentally fragile. This fragility arises because the diffuse coupling of components within a complex system makes them prone to completely unpredictable behavior in the face of small perturbations.

A new approach should be considered so as to compare and contrast related work in the areas of system design and robustness. It is possible to build an application model for which the technique based on power consumption and data loss can be deployed to construct the robust topology. The form of the resulting spanning tree is compared to other spanning trees that are commonly found (FH, SP). With use of experimental analysis and simulation work, evaluation of the data loss and power consumption based on the different approaches is possible. Simulation work should consist of different applications containing various graphs. These graphs represent the different network topologies wherein data is forwarded up the tree and collected at a single node.

Hence, implementation and design technique is required for improving the robustness of a distributed system for applications that operate on a spanning tree overlay network. Spanning trees are mostly used in communication networks as a means to disseminate information from one node to all other nodes and/or to collect information at a single designated node. The defining characteristic of spanning tree topology when compared to other types of commonly seen spanning trees will show that the resulting trees perform well for multiple, conflicting metrics; the trees are robust and admit near-optimal resilience to node failures (to be proved as the optimal case) and at the same time are very efficient with respect to power consumption

2. NEED FOR ROBUST SPANNING APPROACH

2.1 Importance of topology structure

According to [5], it is suggested that mathematical subject of topology investigates objects whose characteristics are constant through distortion. Objects can be topologically equivalent while appearing physically different. As an example, any two objects formed with a simple rubber band are topologically equivalent so long as the band is not parted. A noteworthy practical analysis technique based on Topology is Kirchoff circuit analysis. Computer network topology is an extension of basic topology. This discipline examines the configuration of computer system elements and their associated interconnections. Physical Network Topology emphasizes the hardware associated with the system including workstations, remote terminals, servers, and the associated

wiring between assets. Logical Network Topology (also known as Signal Topology) emphasizes the representation of data flow between nodes, not dissimilar from graph theory analysis. The logical topography of a network can be dynamically reconfigured when select network equipment, such as routers, is available.

There might be several spanning trees possible. A minimum spanning tree would be one with the lowest total cost. The deterministic MST problem has been well studied and many efficient algorithms have been developed by many researchers [14]. There are now two algorithms commonly used, Kruskal's algorithm and Prim's algorithm for a certain network, if the edge weights of the network are fixed, then it is possible to calculate the minimum spanning tree by the algorithms.

2.2 Deterministic Minimum Spanning Tree Problem

Given a connected weighted undirected graph $G = (V; E)$, a spanning tree of that graph is a subgraph $T = (V; S)$ such that $S \mu E$, which is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. A minimum spanning tree contains a subset of the edges that forms a tree and includes every vertex, where the total weight of all the edges in the tree is minimized.

In [14] the uncertain minimum spanning tree problem in a given uncertain network considered. It is assumed that the edge costs are not precisely known and they are specified as uncertain variables. Uncertainty theory is applied to characterize

the inverse uncertainty distribution of the uncertain spanning tree under uncertain weights or costs. Three types of minimum spanning trees for uncertain network are proposed, which includes expected minimum spanning tree, alpha minimum spanning tree, and distribution minimum spanning tree. A 99-table algorithm for the uncertain minimum spanning tree problem is also presented in [14].

2.3 System Design

Several researchers have argued that robustness is a crucial property in the design and operation of distributed systems [1]–[3], [6], [7]. Gribble [2] suggests using techniques such as admission control, system introspection, and adaptive control to achieve robustness in distributed applications. These techniques are all adaptive in nature. If proactive approach is taken into consideration, same technique will be most appropriate in situations where an immediate change in the network topology is undesirable. In the physics community, Carlson and Doyle have introduced a concept for the incorporation of robustness into the design of complex systems. They have named this concept highly optimized tolerance (HOT) [7], [9].

Using examples from biology and engineering, they show how high-performance (or high-yield) systems can be made robust against disturbances for which they were designed to handle, yet can fail catastrophically when subjected to unanticipated disturbances. This is the trade-off between the need for high-performance and the increased sensitivity to randomness. In a similar fashion, proactive approach will strike a trade-off between performance and resilience to network disruptions.

2.4 Scheduling

In [10] Barlas and Veeravalli present a technique for the delivery of continuous-media documents over unreliable communication links. Progress in this area is important in order to realize the widespread adoption of video-on-demand services over the Internet. The solution presented in [10] is to deliver multiple installments of the document to the client. Each installment is transferred from a different, distributed server. A proxy at the client assembles the pieces. By making a small, adjustable portion of each installment overlap with the adjoining section, the system is able to withstand a certain amount of data loss without

compromising the continuous nature of the delivery. Similar to proactive approach, the authors of [10] assume no knowledge of where network disruptions will occur. Given the desired probability of interrupt-free playback, the authors show how to compute the appropriate amount of overlap. With respect to the divisible load scheduling problem Ghose et. al. [11] investigates probing strategies for estimation of network parameters and subsequent use of those parameters to allocate work to processors. Thus, no previous knowledge of bandwidth nor computing speeds is required. Indeed, one strategy that was investigated employs continuous probing and can adapt to changing network parameters. However, any probing strategy requires a certain amount of overhead. Furthermore, it is required to stress on the fact that the technique is purely proactive in nature as opposed to being adaptive or reactive. As such the overhead associated with measuring and adapting is not an issue [1].

2.5. Network topology and node mapping

According to Oppenheimer, Vatkovskiy and Patterson [4], to facilitate research and development in wide-area networked services, universities and companies have traditionally constructed off-the-shelf PC clusters outfitted with special network emulation software. Although hardware costs are constantly declining, system management costs are not. This fact, combined with the large number of open source developers unaffiliated with large organizations, lead us to believe that shared test bed infrastructures such as Planet Lab, which distribute management responsibility (and therefore costs) across many sites, will become increasingly popular as evaluation platforms. Because the single network topology of such a platform cannot match the varied topologies of interest to an experimenter, the experimenter will want to select the subset of available nodes that most closely matches the topology of interest for the experiment. This “topology embedding” problem becomes even more interesting when the experimenter adds per-node constraints, as when exploring the robustness of a service to node heterogeneity and/or jobs competing for CPU and memory resources Thus it is believable that scalable, robust resource discovery is an essential part of a distributed service evaluation framework [4].

K. Sohrabi, [16] presented a set of algorithms for establishing and maintaining connectivity in wireless sensor networks. The algorithms exploit the low mobility and abundant bandwidth, while coping with the severe energy constraint and the requirement for network scalability. The algorithms further accommodate slow mobility by a subset of the nodes. However, many important research questions remain, including for example bounds on the minimum energy required for network formation especially taking into account the interactions with the signal processing functions. Another issue is the extent to which the algorithms can efficiently deal with more extensive mobility in the nodes and the targets. However ,realization of sensor network s needs to satisfy the constraints introduced by factors such as fault tolerance, scalability and power consumption[15].

P. Santi [17], The goal of technique is to control the topology of the graph representing the communication links between network nodes with the purpose of maintaining some global graph property (e.g., connectivity), while reducing energy consumption and/or interference that are strictly related to the nodes' transmitting range. D. England [20] present both centralized and distributed tree construction algorithms and evaluate their effectiveness through analysis and simulation of two classes of distributed applications: data collection in sensor networks, and data dissemination in divisible load scheduling.

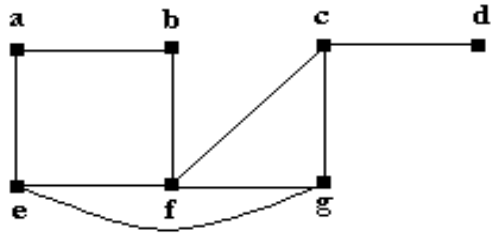
The structure (topology) of interconnection networks is very important in both parallel and distributed systems, but there is an important difference [13]

3. SYSTEM ANALYSIS

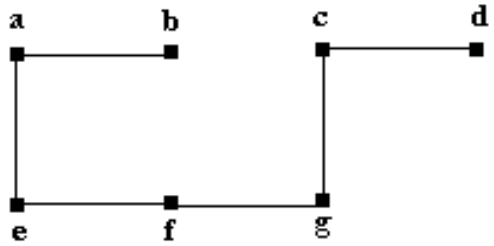
3.1 Existing System:

Spanning Trees:

For a given connected, undirected graph, a spanning tree of that graph is a subgraph which is a tree and connects all the vertices together and removes cycles. A single graph can have many different spanning trees.



(A): Graph



(B): Spanning Tree

Figure 3.1: A Graph and Its Spanning Tree

Figure 3.1 (B) indicates one of the many spanning trees of the given graph in Figure 3.1(A), by removing edges BF, CF and EG to avoid loops.

Spanning trees are widely used in communication network. Spanning trees provides redundancy without creating loops. Therefore, avoids congestion.

For a moderately sized network there exist many possible spanning trees. The most commonly seen spanning trees are the following:

- Shortest Paths
- Fewest Hops
- Minimum Weight

Shortest Paths (SP):

The distance in edge weights of the path from each node to the root node is minimal. Such a tree is efficiently constructed by Dijkstra's algorithm. MST produces trees that are very deep and "skinny." This is natural since the only criterion is edge weight and the location of the root node is not taken into consideration.

Fewest Hops (FH):

The distance in number of hops along the path from each node to the root node is minimal. This method is equivalent to SP when all edge weights are equal and, therefore, Dijkstra's algorithm may be employed. Spanning trees created by FH tend to be shallow and "fat," with the average node degree being fairly large. This is because the only criterion for cost is the distance in hops from the root with no consideration of edge weights.

Minimum Weight:

The total sum of edge weights is minimal. Such a tree can be constructed by either Kruskal's algorithm or by Prim's algorithm [12] and does not take into consideration the location of the root node.

Above mentioned spanning tress are not useful for dense network. Spanning trees created by FH tend to be shallow and fat with the average node degree being fairly large. FH minimizes the expected values of amount of data loss when a node or link fails. However it is not best choice for power consumption. The shapes of the tress produced by the SP are influenced by the distribution of edge weights, but they tend to be deeper and have smaller node degrees than FH trees [1].

3.2 Proposed System:

For dense network, question arises to which spanning tree is best for particular application. The SP, FH spanning trees are not useful for dense network. There is a need to construct spanning trees those are immune to data loss when a node or link fails and yet are able to maintain the performance i.e. Robust Spanning Tree. It is combination of path weight and hop count [1].

In this approach it is required first to reduce the density of the topology using the centralized

algorithm and then evaluation of the data loss and power consumption. This topology can be used both in centralized system and distributed system. Robustness cannot be achieved in the other existing system which is one of the desirable properties as the deliverable outcome of the experimental analysis. According to [1] analysis and simulation work will show that the spanning trees that perform best for different and even opposing metrics are constructed by considering a weighted combination of hop count and path weight as follows.

$\lambda \times$ hop count + $(1 - \lambda) \times$ path weight Eq. (1)

Where $0 \leq \lambda < 1$.

If more importance is placed on hop count, then the tree will tend to be fat and shallow. Alternatively, more importance on path weight means that the tree will be skinny and deep. The type of tree that performs best depends on the metric of interest. In order to construct trees that perform well under a wide variety of metrics, we attempt to make the tree fat near the root and skinny further away from the root. The weight λ is really a function of a node's depth in the tree. When an edge (i, j) is being considered conclusion in the tree and i is the new vertex not already in the tree, then

$\lambda i = 1 - (hi / \epsilon_1)$, Eq. (2)

Where hi is the hop count of node i from the root and ϵ_1 is the eccentricity of the root node [1].

The eccentricity of a node is the largest of the shortest paths from that node to all other nodes [1]. Eccentricity can be measured in number of hops, not path weight. Another way to think about it is that eccentricity is the depth of the deepest leaf in the SP tree. However, it is important that the eccentricity of a node is a characteristic of the underlying graph; it is not a property of the overlay network [1]. Using this measure of eccentricity in Equation 2 ensures that $0 \leq \lambda i < 1$ for all i. It also effects values for λi that are close to one when selecting nodes that are near the root, and values close to zero when selecting nodes that are further from the root. This gives the desired relative importance of hop count versus path weight in Equation 1.

3.3 Model

"Robustness is the persistence of certain specified system features, though there is presence of perturbation in the system's environment" [2]. Robust systems perform well across a wide range of operating conditions and exhibit graceful degradation under anomalous conditions [1]. The defining characteristic of robust spanning tree topology when compared to other types of commonly seen spanning trees is that the resulting trees perform well for multiple, conflicting metrics; the trees are robust.

As described in [1], there is an algorithm for constructing a robust spanning tree. For many distributed applications, the routing of data and messages takes place on a virtual overlay network that is constructed on top of the underlying physical network. For example, nodes in peer-to-peer systems are connected via the physical links in the Internet; however, a node forwards queries only to nodes in its own list of neighbors, thus forming an overlay network. Not surprisingly, the topology of such an overlay network plays a significant role in the performance and efficiency of the distributed system [1]. Herein, we address distributed systems for which the overlay network is a spanning tree, i.e., a connected network that contains no cycles. Furthermore, one particular node in the network is designated as the root node. The root node acts as a collection point for data and/or as a load origination point for the distribution of work. Here, nodes are identified by indices and the root node is always labeled with the numeral1 or numeral 0.

3.3.1 Expected Data Loss:

As far as the amount of data loss is concerned, here the failure of a node is equivalent to the failure of the link to the parent. This applies to every node in the tree except the root node, which

will not fail. The reason for omitting this possibility is that if the root node fails, then the entire data collection process is halted regardless of the topology of the overlay network. Thus there is no reason to include this possibility when comparing alternative topologies [1].

Consider a tree T with vertex set V (T) and edge set E (T). Let mi be the number of nodes in the subtree rooted at node i (including node i itself) and let qi be the probability that node i will fail. Then the expected value of data loss L given that exactly one node fails is

$$E\{L \mid \text{exactly one node fails}\} = \sum_{i \in V(T)} m_i * q_i \quad \text{Eq.(3)}$$

Where,

$$\sum_{i \in V(T)} q_i = 1. \quad \text{Eq.(4)}$$

Throughout this work it is assumed that all nodes have an equal probability of failure. The expected Value of data loss then becomes

$$E\{L \mid \text{exactly one node fails with equal probabilities}\} = 1/(n-1) * \left(\left(\sum_{i \in V(T)} m_i\right)\right) \quad \text{Eq. (5)}$$

Where,

n = |V (G)| is the number of nodes in the graph [1].

3.3.2 Power Consumption

Node consumes power when they transmit and receive data. More power is required to transmit and receive over longer distances and through obstructions. The amount of data forwarded up the tree also plays a role in power consumption since more data means more transmissions. Hence parents of large subtrees are subject to fast depletion of their power reserves. A partial offset to this situation occurs when data can be aggregated and thus fewer forwarded messages are required. However, even with data aggregation, parent nodes must still receive the data to be aggregated and also expend processor power performing the aggregation itself [1]. Furthermore, data aggregation does not reduce the number of sampled data points and so the loss of a large subtree will still result in a much smaller sample size, regardless of the extent of data aggregation. So, consider three metrics for power consumption:

- The total amount of power consumed by the network,
- The maximum amount of power consumed by a single node,
- The number of messages associated with the node that consumes the maximum amount of power.

The second and third perspectives are important since the particular node in question will be the first to deplete its power supply [1]. Ignoring data aggregation for the moment, the amount of network power consumed when all nodes send one message to the root node is the sum over all nodes of the product of the number of messages sent and the power required to transmit a single message [1]. Again let m_i be the number of nodes in the subtree rooted at node i. and let $z_{i,j}$ is the weight on the link from node i to its parent node j. Then the total network power P required to collect a single data observation is

$$P = \sum_{i \in V(T)} m_i * z_{i,j} \quad \text{Eq. (6)}$$

4. APPLICATION

Evaluation of data loss and power consumption based on the traditional approaches clearly shows the need of robustness in the system design. A small network can be represented by a graph which consists of edges and nodes. The resulting tree topology of this input graph is drawn for SP and FH respectively. It consists of following steps:

- Creation of shortest path topology for a given normally connected system by considering only the edge weights between the nodes that are connected.
- Evaluation of data loss and power consumption for SP method.
- Construction of fewest hop topology for a given normally connected system by considering the number of hops between the source and destination.

- Evaluation of data loss and power consumption for FH method.
- Comparison of the above mentioned approaches based on the data loss and power consumption

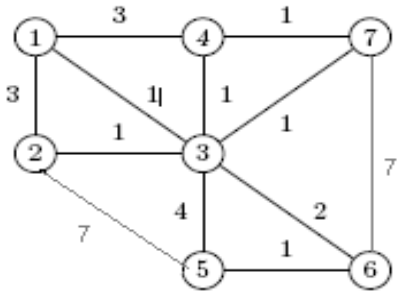


Figure 4.1: A Small Network

Figure 4.1 shows a small network that consists of seven nodes. An edge between two nodes indicates that they can communicate directly. The edge weight is the amount of power required to transmit a single message between the two nodes. A larger weight indicates a greater distance or an obstruction. Node 1 is the root node. It is the collection point to which all other nodes must route their data.

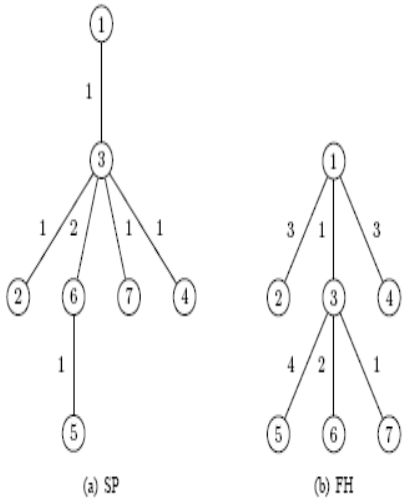


Figure 4.2: Traditional Spanning trees of small network

The expected data loss of the spanning tree in Figure 4.2(a), using Eq. (3) is

$$E\{L\} = 1/6 * [(6 + 1 + 2 + 1 + 1 + 1)] = 2.0.$$

Similarly for Figure 4.2(b), $E\{L\} = 1.5$ .In this case the FH spanning tree admits the smallest value for expected data loss. This is intuitive since the depth of the tree is as small as possible indeed the total network power for the SP tree of Figure 4.2(a) (going breadth-first through the tree), using Eq. (6) is

$$P = (6 \times 1) + (1 \times 1) + (2 \times 2) + (1 \times 1) + (1 \times 1) + (1 \times 1) = 14.$$

Similarly, P for the FH tree of Figure 4.2(b) is 17.It can be shown that SP trees will admit the minimal values for P. This is natural since the weights on the paths to the root node are smallest. SP trees use low power, but expose the network to greater possibilities of data loss when nodes fail.

5. ALGORITHM

The centralized algorithm is appropriate in situations where the node on which the algorithm runs has full knowledge of the nodes and link speeds in the underlying network [1]. This algorithm is based on Prim's algorithm [12]. Prim's algorithm begins with a single node i.e. root node and at each iteration the cheapest edge that incorporates a new vertex is selected for inclusion in the tree. The cheapest edge is simply the one with the smallest edge weight. In above algorithm the cheapest edge is computed as in Equation 1.

The complete method for generating spanning trees in this way is presented as algorithm and the shorthand identifier is RB. As per mentioned in [1], the computational complexity of such algorithm requires the eccentricity of the root node. This can be accomplished by using Dijkstra's algorithm to compute the shortest paths from the root node to every other node and then

taking the maximum distance (in number of hops) as the eccentricity.

The complexity of Dijkstra's algorithm is O (|V |^2). The process of adding edges into the partial spanning tree requires at most |V | − 1 cost computations and comparisons for every node in the graph (except the root node which is added initially). For this step, the worst-case time complexity occurs for complete graphs and is O (|V |^2). However, the average time complexity is O (a × |V |) where a represents average node degree. Thus the overall complexity of Algorithm is O (|V |^2) and is therefore good in the sense that it is bounded by a polynomial in the size of the graph [1].

Data: graph G= {V, E} with edge weight zij

Compute the eccentricity of the root node $Є_1$;

Initialize the tree with the root node only;

While *there are still vertices not yet added to the tree* do

For *every vertex* i *not in the tree* **do**

\Compute $\lambda i = 1 - (hi / Є_1)$;

Compute $\xi i = \lambda i \times hi + (1 - \lambda i) \times (\xi i + zi,j)$;

Store the minimum cost found so far;

End

Add the vertex i along edge (i, j) that achieves the minimum cost;

End

Figure 5.1: Centralized Algorithm for computing cheapest edge [1].

This algorithm takes input as a graph associated with edge weight. For this 1st compute eccentricity of root node, then calculate minimum cost for each node.Robust spanning tree created by this algorithm for the graph in figure 4.1 will be:

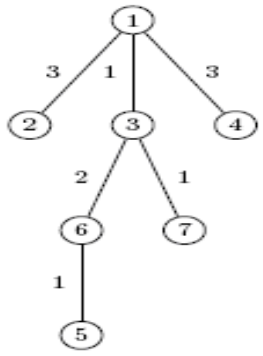


Figure 5.3: Robust spanning tree of small network

The results of Expected Data Loss and Power consumption for Robust spanning tree are:

Data loss:

$$= 1/(n-1) * ((\sum_{i \epsilon V(T)} m_i))$$

= 1/6 {1 + 4 +1 + 2 + 1 + 1}

= 1.66

Power consumption:

$$= \sum_{i \epsilon V(T)} mi * qi$$

= {(1 * 3) + (4 * 1) + (1 * 3) + (2 * 2) + (1 * 1) + (1 * 1)}

= 16

RB results for expected data loss and power consumption are intermediate between SP and FH results.

6. MODULES

6.1 Creating a Normal Topology Structure.

- In this module, there is a construction of a normal topology structure. This shows how the system nodes are interconnected between them.
- The topology structure is constructed by identifying the possible paths between the nodes that are interconnected.
- The major problem in this topology structure

is that it has redundancy problems, which shows need of the improved topology structure to be constructed.

6.2 Creation of Fewest Hop Topology.

- In this module, construction of a topology based on fewest hops method is required to be implemented.
- This topology is based on consideration of the number of hops between the source and the destination.

6.3 Creation of Shortest Path Topology.

- In this module, there is a conversion of the normal topology structure into the Shortest Path Topology structure.
- This can be constructed by considering only the edge weights between the nodes that are connected within the system.

6.4 Creation of Robust Spanning Tree Topology.

- This module is required to be implemented as per proposed system work, where construction of the topology is done using robust spanning method.
- This can be constructed by considering the advantages of the existing systems, fewest hops and shortest path respectively.

6.5 Comparison of all the three topologies

In this module, there is a comparison of the performance of all the three constructed topologies.

- Comparison is done based on the data loss and power consumption between FH SP and RB topologies.
- Here it can be seen that the Robust Spanning Tree Topology is significantly better than the other existing topologies

7. SIMULATION AND RESULTS

7.1 NS2 - Simulation Tool

There are many simulation tools available for simulation, among them NS-2 is most popular, open source simulation tool. Use of this tool, will clearly depict the comparison as well as results of traditional spanning trees and robust spanning tree for analysis.

NS2 creates a virtual environment for us. network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions [21]

7.2 Simulation Steps

- ✓ Create trees for traditional spanning trees. i.e SP, FH for the given graph.
- ✓ Then identify nature of those trees in terms of data loss and power consumption.
- ✓ Create RB tree that give benefit of both trees that is mentioned above.
- ✓ Run the application on this tree and see how network will perform.
- ✓ Analyze the behavior of network for different conditions and compare them by considering different types of graph each time .

7.3 Simulation Results

The following scenario represents the sample input graph consisting 10 nodes and its corresponding tree obtained by applying Fewest Hop Technique.

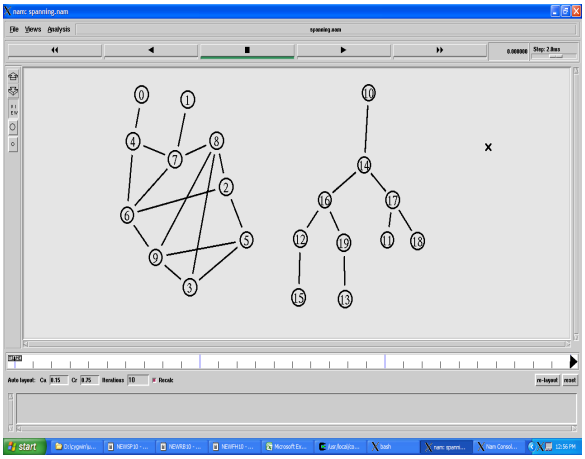


Figure 7.1: Input Graph and Resulting tree based on Fewest Hop

Figure 7.2 represents the sample input graph consisting 10 nodes and its corresponding tree obtained by applying Shortest Path Technique.

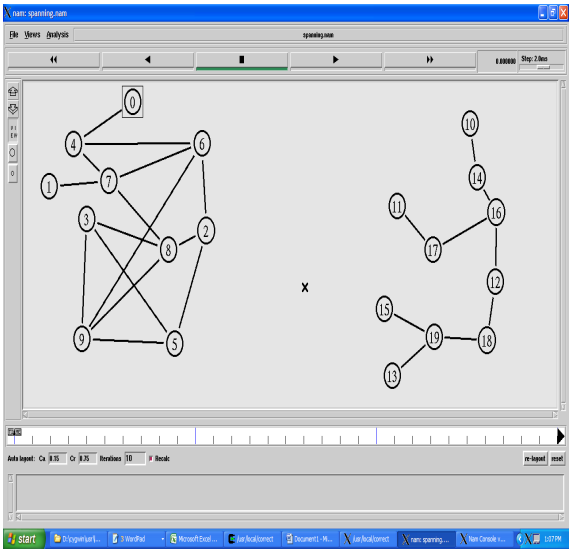


Figure 7.2 : Input Graph and Resulting tree based on Shortest Path

After executing the simulation program ,data loss and power consumption values for the resulting tree based on Shortest Path technique are obtained respectively.

For the same input graph .resulting tree based on combined approach is as shown in Figure 7.3.

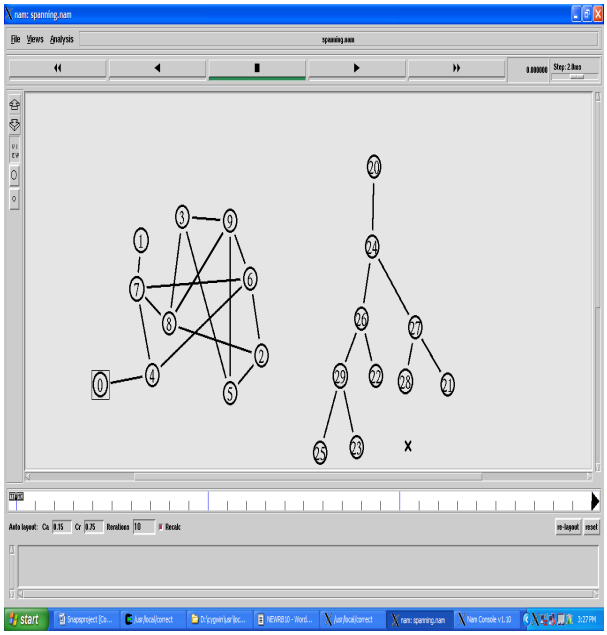


Figure 7.3: Input Graph and Resulting tree based on Robust Spanning Approach

Simulation model accepts the same graph as an input and generates the different tree structures as an outputs, applying FH ,SP,RB Method respectively .

Execution of simulation model based on above three methods calculates expected data loss and power consumed for the input graph ,in accordance with FH ,SP,RB Method .It can be easily observed that RB technique gives the intermediate values for both metrices.

8.4 Experimental Evaluation:

Using the metrics those defined for expected data loss and power consumption, it is required to evaluate the performance and robustness of different spanning trees via simulation on three categories of randomly generated networks. The three categories are

- **Sparse** : Each node has between 1 and 10 neighbors.
- **Medium:** Each node has between 20 and 30 neighbors.
- **Dense** : Each node has between 40 and 50 neighbors.

The number of neighbors is uniformly distributed in the respective ranges. For each category, it is essential to generate 10 to 15 random graphs using the method and software described in [18]. The procedure for generating a random graph is to produce the uniformly distributed degree sequence, so it is required to ensure that a graph with that degree sequence indeed exists[1] and then with the use of software of Viger and Latapy for generation of the simple[2] connected graph can be achieved. The edge weights for all three categories were uniformly distributed between 0.1 and 10. For each of the three categories, the data loss and power consumption metrics have to be computed for each of the 10 to 15 randomly generated graphs. The results presented below are the averages over these random graphs in the respective categories.

[1]is accomplished by employing the Theorem of Havel-Hakimi [12].

[2]Simple means no loops and no multiple edges.

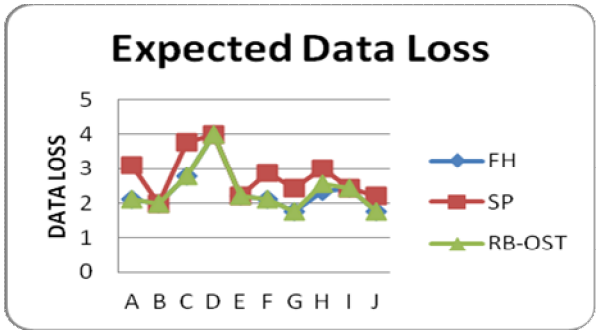


Figure 7.4: Expected data loss - Randomly Generated Sparse Networks

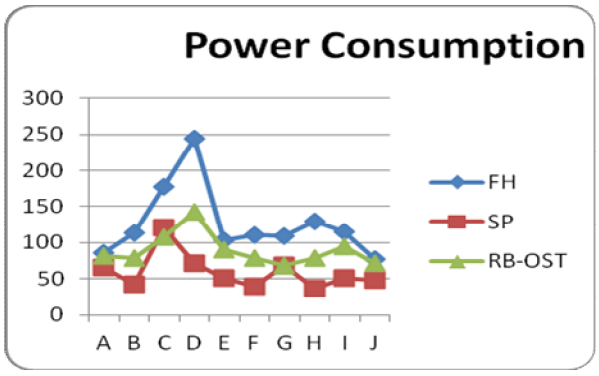


Figure 7.5: Power Consumption -Randomly Generated Sparse Networks

First category of the network is the one in which each node has between 1 and 10 neighbors, called as Sparse networks. Sparse network may have different graphical representation showing the nodes and links connecting them. It is required to achieve the results, based on traditional methods and metrics taken into consideration for the comparison. Expected data loss and power consumption can be computed for different randomly generated graphs. Input graph represents the normally connected system having 10 nodes and the corresponding links shows the structure of the network in terms of topology. Graph A to Graph J are various graphs representing the different networks of sparse category. Consider the input to the simulation experiment as Graph A showing 10 nodes and interconnection between them. The procedure for generating a random graph is to produce the uniformly distributed degree sequence, can be achieved as per software described in [19]. With the use of ns2 environment, the different graphs can be viewed showing the structure of such input topology. Later on, it is essential to convert the given normally connected topology into the topology based on traditional approaches FH, SP respectively. Each of these methods generates the tree topology for a given sparse network. The tree structure obtained in each case is useful for computing the defined metrics. With the use of equation (3) and equation (6), expected data loss and power consumption are calculated for the given input graph.

For randomly generated sparse networks the values achieved as a results show that average expected data loss in topology structure based on FH method is less than SP method. While average power consumption in tree structure based on SP method is less than FH. When combined approach is taken into the consideration, RB method gives the intermediate results compared to both traditional approaches.

Figure 7.6: Expected data loss - Randomly Generated Dense Networks

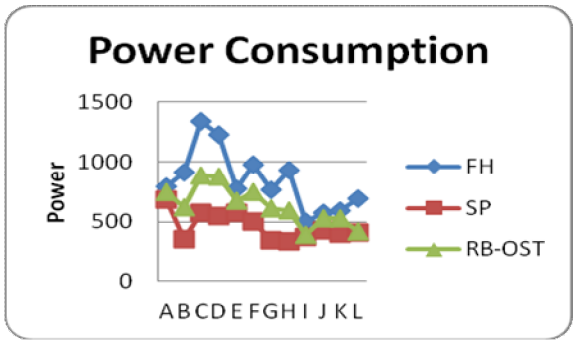


Figure 7.7: Power Consumption -Randomly Generated Dense Networks

Naturally, the FH trees exhibit the smallest depths in dense network. The robust trees are deeper than FH, but not as deep as SP. FH trees admit the smallest values for expected data loss. It is important that the expected data loss for RB trees are only slightly greater i.e. 1.95 vs. 1.92 for medium density networks and 3.20 vs. 3.18 for dense networks. Thus with respect to data loss the RB trees perform quite well.

The power consumption metrics for RB fall in between SP (the lowest) and FH (the highest). Especially for maximum power used, the RB values are closer to SP than to FH. This is evidence of the best of both worlds: data loss similar to FH trees and power consumption closer to SP trees.

A metric that combines the notions of data loss and power consumption can be viewed in the following way. The node that consumes the most power, i.e. the node that accounts for the maximum power metric, will be the first node to deplete its power supply. Let us assume that the size of the subtree rooted at this node (i.e. m_i), and hence the number of data points lost when this node ceases operation. In other words, it is the measurement of the number of messages that are forwarded by the node that first depletes its power supply [1].

In this respect, for medium and dense networks, the RB trees exhibit the smallest amount of data loss. However, the real benefit of the RB method comes from the combination of low data loss and relatively low power consumption. This can be seen in the results for expected data loss and power consumed by any one node with respect to the three network categories, as shown in figure 7.8 and figure 7.9.

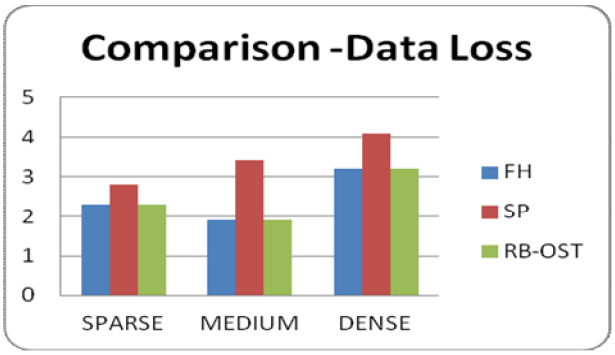


Figure 8.18: Comparison of all the three topologies based on the data loss between FH SP and RB topologies

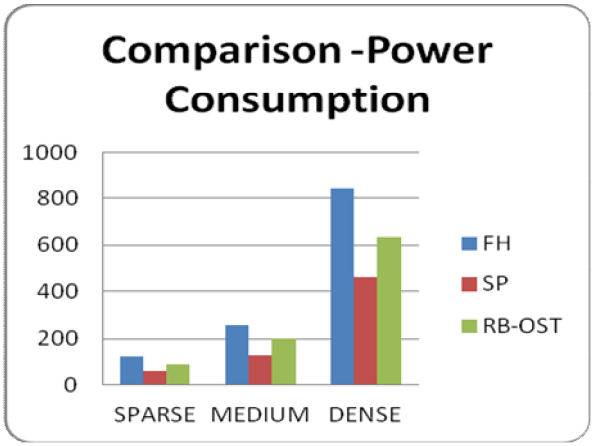


Figure 8.19: Comparison of all the three topologies based on the power consumption between FH SP and RB topologies.

8. FUTURE SCOPE

Simulation environment may not provide real characteristic of the line under consideration. An approach taken towards robustness is proactive rather than reactive. Simulation experiments and evaluation are based only on simple topology of nodes.

It is natural to ask when a node realizes that its parent has failed, why not simply choose another parent (assuming the node has multiple neighbors)? This may or may not be desirable. If there are many nodes that choose a new parent, then the properties of the tree will be unknown. If the goal is to collect a reasonable amount of data over a long period of time, then it would be better to use a topology about which prior knowledge and some statistics is available.

It seems that the permanent question is: At what point is it worth rerunning the spanning tree construction algorithm to construct a new tree? This is one subject of the future work.

9. CONCLUSION

Robustness is an important property for distributed computing systems. It measures the ability of a system to maintain performance in the presence of adverse operating conditions. This notion is important for today's large, complex high-performance computing systems because they are increasingly subject to uncertainty in load, communication latency, and even resource availability. Robust systems are able to withstand disturbances and maintain performance features. Ideally, computing systems should achieve both performance and robustness.

It is possible to present a methodology for constructing a spanning tree overlay network that exhibits robustness to network disturbances. The construction technique employs a weighted formula for hop count and path weight that changes the relative importance as the distance from the root node changes. This results in trees that perform well for a wide variety of metrics. Experiments on three sets of randomly generated graphs also show that these spanning trees achieve good performance for two opposing metrics where traditional forms of spanning trees do not. When compared to the most common forms of spanning trees, our robust trees are closest in appearance to fewest-hops spanning trees. Construction of tree using combination of SP and FH spanning tree technique, gives expected data loss closer to FH (minimum) and power consumption closer to SP.

REFERENCES

[1] Darin England, Bharadwaj Veeravalli, Jon B. Weissman, "A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments"Aug 2006.

[2] S. D. Gribble, "Robustness in complex systems," in Proceedings IEEE Eighth Workshop on Hot Topics in Operating Systems, May 2001, pp. 21–26.

[3] D. England, J. Weissman, and J. Sadagopan, "A new metric for robustness with application to job scheduling," in IEEE International Symposium on High Performance Distributed Computing 2005 (HPDC-14), July 2005, research Triangle Park, NC.

[4] D. Oppenheimer, V. Vatkovskiy, and D. A. Patterson, "Towards a framework for automated robustness evaluation of distributed services," in FuDiCo II: S.O.S. Survivability: Obstacles and Solutions, 2nd Bertinoro Workshop on Future Directions in Distributed Computing, June 2004, university of Bologna Residential Center, Bertinoro, Italy.

[5] Tomas Fencl, "Algorithm for network topology design", Doctoral Thesis, Czech Technical University in Prague, 2011

[6] R. Albert, H. Jeong, and A. L. Barabasi, "Error and attack tolerance of complex networks," Nature, vol. 406, pp. 378–382, July 2000.

[7] J. M. Carlson and J. Doyle, "Highly optimized tolerance: Robustness and design in complex systems," Physical Review Letters, vol. 84, pp. 2529–2532, 2000.

[8] S. Ali et al., "Measuring the robustness of a resource allocation," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 7, pp. 630–641, July 2004.

[9] J. M. Carlson and J. Doyle, "Highly optimized tolerance: A mechanism for power laws in

designed systems," Physical Review E, vol. 60, 1999.

[10] G. Barlas and B. Veeravalli,"Optimized distributed delivery of continuous-media documents over unreliable communication links," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 10, Oct. 2005.

[11] L. Boloni and D. C. Marinescu, "Robust scheduling of metaprograms," Journal of Scheduling, vol. 5, no. 5, pp. 395–412, 2002.

[12] D. B. West, Introduction to Graph Theory, 2nd ed. Prentice Hall, 2001.

[13] D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods. Athena Scientific, 1997.

[14] Jin Peng, Shengguo Li, "Spanning Tree Problem of Uncertain Network".

[15] I. F. Akyildiz, "A survey on sensor networks," IEEE Communications Magazine, vol. 40, no. 8, pp. 102–116, Aug.2002.

[16] K. Sohrabi, "Protocols for self-organization of a wireless sensor network," IEEE Personal Communications, pp.16–27, Oct. 2000.

[17] P. Santi, "Topology control in wireless ad hoc and sensor networks," ACM Computing Surveys, vol. 37, no. 2, pp. 164–194, June 2005.

[18] F. Viger and M. Latapy, "Efficient and simple generation of random simple connected graphs with prescribed degree sequence," in The Eleventh International Computing and Combinatorics Conference, Aug. 2005, kumming, China.

[19] B. Veeravalli and N. Viswanadham, "Subrobust solutions using integer apsproximation techniques for scheduling divisible loads on distributed bus networks," IEEE Transactions on Systems, Man, and Cybernetics, Part A, vol. 30, no. 6, pp. 680–691, Nov. 2000.

[20] D. England, "Robust design for distributed computing systems," Ph.D. dissertation, Department of Computer Science and Engineering, University of Minnesota, Twin Cities, June 2006.

[21] www.isi.edu/nsnam/ns/doc/ns_doc.pdf