A Network Topology With Efficient Balanced Routing

Dionysios Kountanis Vatsal Sharadbhai Gandhi
Wasim El-Hajj Ghassen Ben Brahim
email: {kountan, vsgandhi, welhajj, gbenbrah}@cs.wmich.edu
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008, U. S. A.

Abstract

In this paper a special network topology is considered in terms of how nodes should be interconnected. The considered network will be specified by a graph $G = (V, E)$, where V is the set of nodes and E is the set of links. We assume, in one case, that the set V has cardinality $|V| = k(k-1)+1$, where k is a power of a prime number. We define a function $f : V \to U \subseteq V$ such that $|U| = k-1$. For each $v \in V$ we find $f(v)$. Following this approach, E is defined by $\{(v, f(v)) | v \in V\}$. According to our scheme, any two sites can communicate by traversing at most 2 nodes regardless of the network size. Contrarily to the existing routing approaches where routing decisions are based on a large set of information duplicated at each site, the routing scheme we propose is distributed and it greatly reduces the size of the information set that should be maintained at each site.

Keywords: Network Topology, Balanced Routing, Network Congestion, Distributed Routing.

1 Introduction

The growing popularity of real-time and multimedia applications over the Internet has stimulated strong interest in improving routing protocol performance in order to achieve better Quality of Services (QoS) required by current applications.

Routing is the process of finding an efficient path between any pair of source and destination nodes in the network. The path must be chosen so that network throughput is maximized and message delay, message loss and other undesirable events are minimized. Most routing protocols aim at finding shortest paths in terms of intermediate hops between any two node pairs.

Routing protocols have two major components: the routing table and the routing engine [3], [6]. The routing table, which is available within each node is maintained and kept up-to-date by the protocol and contains information on how data should be routed from the current node to any destination node. However, the routing engine is the component responsible of generating the routing table. Usually any of the available shortest path algorithms are used such as Dijkstra, Bellman-Ford, etc.

Routing protocols can be classified into two major classes [2], [5], [1]: source routing and destination routing (next-hop routing). In source routing, a source node determines the path that a data message must take. In destination routing, each node uses its routing table to store a next hop for each destination.

In general, routing protocols adopts either of the following two approaches: Centralized routing or distributed routing functionalities [8]. With the centralized approach, each node solely makes the decision on how data should be routed. This decision is based on its knowledge of all current network resource availability. However, with the distributed approach, each node is able to make the decision about how routing should be performed based on partial view of the network resource availability.

As the network size gets larger, two major problems arise with the centralized routing protocol approaches [4], [9]. The first problem consists of a tremendous increase of the routing table size stored within each node, which makes it very difficult to maintain and becomes a protocol bottleneck. Routing protocols implementing huge routing table size usually provide low QoS to application users. The second major problem is that communication messages are required to achieve routing functionalities. These messages are considered as overhead and consume a lot of the available bandwidth over the network. This problem is aggravated as the network size gets larger.

Most of routing protocols consider a minimum number of intermediate hops when routing between source and destination pairs. This definitely reduces the communication overhead as well as the congestion over the network links.

Several distributed routing approaches have been proposed. In [6] distributed shortest-path routing algorithm based on the Ford-Fulkerson method was proposed. In the algorithm, each node maintains a routing table that lists the cost of the shortest path and the next node in the path to every other node in the network. Each node broadcasts the cost information to its neighbors either periodically or whenever path costs change. The shortest path to a given destination is then computed based on the partial network topology information maintained inside each node. In [3] a study is presented about the diameter-degree tradeoffs of current distributed hash table (DHT) algorithms.

The contribution of this paper lies in the definition of a distributed routing algorithm which improves the overall throughput of the network without adversely affecting the user requested services. The goal is perform all routing functionalities while efficiently using the available network resources. In our design, we consider the routing functionality in a special network topology with a minimum number of links which satisfy several constraints. This leads to a minimum intermediate number of hops routing while minimizing the communication overhead.

The paper is structured as follows. In section 1, we describe our network topology specifications. In section 2, we present the design of the routing engine algorithm. In Section 3, we illustrate our approach in implementing a distributed routing. In section 4, we present our simulation results.

2 Network topology design

Our design of the network topology guarantees that there is a path of a maximum of one intermediate hop between any pair of nodes. For each node in the network, a communication set is assigned such that $\forall i \forall j : 1 \leq i, j \leq N :: C_i \bigcap C_j \neq \phi$. Consequently, every pair of sites can connect directly to each other or a site acts as a connector between them. This site can be found simply by considering the intersection of the source and destination nodes' communications sets. This design, as will be detailed in the forthcoming sections, achieves minimal shortest path routing, less traffic overhead, and less routing table storage requirement. In the next subsection, we present an overview of our design, which closely follows Maekawa's algorithm used to obtain mutual exclusions of events in distributed environments [7].

2.1 Design Overview

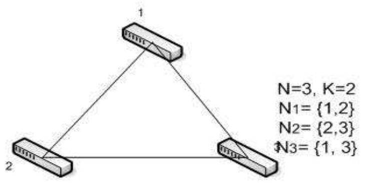
As mentioned earlier, our network topology design is based on Maekawa's sets [7] that are used to obtain mutual exclusion of events in distributed environments. Maekawa associates for each entity of the system a request set that should achieve several constraints. By analogy to his approach, we propose assiging for each network node a communication set satisfying the following constraints:

1. $(\forall i, \forall j : 1 \leq i, j \leq N :: C_i \bigcap C_j \neq \phi)$
2. $(\forall i : 1 \leq i \leq N :: N_i \in C_i)$
3. $(\forall i : 1 \leq i \leq N :: |C_i| = K)$
4. Any site N_i is contained in K number of C_is, $1 \leq i, j \leq N$.

As Maekawa observed [7], condition 1 states that any two communication sets should have at least one site in common. Condition 2 states that each site should belong to its own communication set. Condition 3 constraints the size of each set to be equal to K where $K = \sqrt{N}$ in the optimal case. This condition implies that all sites have the capability of doing the same amount of work. Condition 4 states that each site should be contained in K other sets, implying that all sites have equal responsibilities which is an important aspect in load balancing.

Conditions 1 and 2 are necessary for the correctness of the topology. Conditions 3 and 4 are desirable features of the design, which can be relaxed with no major impact on the overall network topology characteristics.

Once the above constraints are satisfied, the network topology is constructed by connecting each site with all nodes in its commumnication set. Fig. 1 depicts two such network configurations.



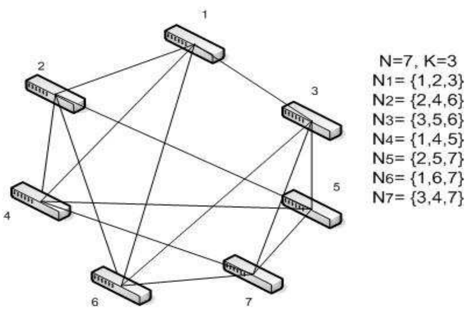


Figure 1: Example of two network topologies of size $N = 3$ and $N = 7$ nodes respectively

2.2 Communication set generation

The communication sets are the finite projective planes passing through K points. The projective planes exist only when $K+1$ is the power of a prime number. So, it is not possible to generate communication sets according to finite projective planes technique when $K+1$ is not a power of a prime number. Communication sets for such values of K can be generated using other systematic techniques. Next we illustrate the generation procedure of the communication sets for both cases: (1)$K+1$ is a prime number and (2) $K+1$ is not a prime number.

2.2.1 Case where $K+1$ is power of a prime number

If $K+1$ is power of a prime number, the communication sets are generated in polynomial time using the following algorithm:

- Consider a matrix of size $(K-1) \times (K-1)$
- Generate K groups of $(K-1)$ nonintersecting sets
  - $(K-1)$ nonintersecting rows
  - $(K-1)$ nonintersecting columns
  - $(K-2)$ of $(K-1)$ nonintersecting diagonals
  - For different diagonals, jump 1 on each row (the main diagonal), jump 2, $\cdots$, jump $(K-1)-1$
- Each number (out of the first K numbers) can be combined with each of the $(K-1)$ nonintersecting sets to produce $(K-1)$ of 1-element-intersected sets

Fig. 2 illustrates the above algorithm for a network with seven nodes $(N=7, K=3)$. The topolgy represented by these sets is illustrated in Fig. 1. It is important to notice that the complexity of the algorithm is $O(N^2)$, which is explained with the fact that for each node, either the row, column, or diagonal is traversed. An extension of the same algorithm can be used to find the sets if $K+1$ is not a power of a prime. The same matrices will be filled with nodes indices, but some nodes will be duplicated in the matrix entries. This will be presented in the next subsection.

2.2.2 Case where $K+1$ is not power of a prime number

Unfortunately, the set assignment algorithm discussed in the previous section works for N number of sites where $N = K(K-1)+1$ (K is the size of the communication sets). In this section, we present an algorithm for computing the Optimum Network Configuration with minimum number of

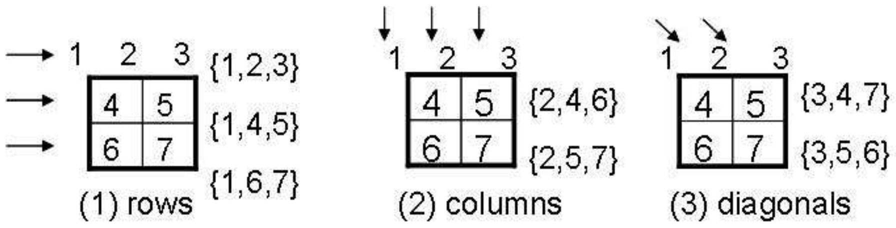


Figure 2: (1): node 1 is taken with the matrix rows. (2): node 2 is taken with the matrix columns. (3): node 3 is taken with the matrix diagonals.

links for any network size N. For the general case i.e. any number of sites, the generation of the communication sets is tricky. For $N \neq K(K-1)+1$ we find minimum $L = K(K-1)+1$ where $L > N$. So, we need to duplicate $L - N$ number of sites in order to generate the matrix used for the generation of the communication sets (as in case 1).

The following algorithm finds the number of duplications (dashes) needed for any given N. This algorithm takes as input the number of sites N, then outputs the number of dashes D and the communication set size K.

```
1. Calculate the size of communication sets $K = \lceil \sqrt{N} \rceil$ and let $L = K(K-1)+1$
2. if $(L < N)$ , then
       $K = K + 1$
3. $D = L - N$
```

For instance, if the network size is $N = 10$, then applying the above algorithm will give $K = 4$ and $L = 13$. Therefore $D = L - N = 3$ (Fig. 3).

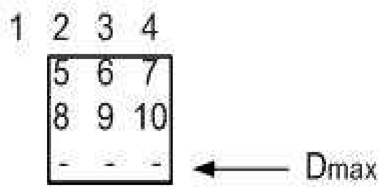


Figure 3: For N = 10, number of dashes = 3

A Brute Force algorithm duplicates all combinations of D dashes for N sites, then generates request sets for all the combinations, finally finds the minimum number of links for all the communication sets combinations. Considering this approach, the total number of combinations are N^D. However, the maximum number of dashes that we can have is $2(K-1)$. i.e. $D_{max} = 2(K-1)$, where $K = \sqrt{N}$. Therefore the complexity of the Brute Force algorithm is exponential and can be expressed $N^{f(N)}$, where $f(N) = D$.

The restriction based algorithm can further be improved in order to achieve less computation complexity. Rather than following the Brute Force technique, one dash is replaced at a time. In order to replace one dash, all values of N are considered. The value of N that minimizes the link cost is chosen to replace the dash. After replacing the dash, another dash is filled using the same procedure. This procedure continues until all dashes have been assigned.

Duplication sites for dashes are found using the following algorithm that takes as input N and K then outputs the combination of sites that should replace the dashes.

```
1:    for each $i$ from 1 to $D$
2:        for each $j$ from 1 to $N$
3:            substitute $N$ for $D_i$
4:            compute communication sets for each nodes $N_i$
5:            calculate the number of links in the network
6:        end for
7:        find the value of $N_i$ for which the number of links is minimum.
8:        restrict $N_i$ for $D_i$.
9:    end for
```

Lines 3 and Line 8 can be performed in constant time $O(1)$, line 4 has complexity of $O(k * N)$, line 5, which consists of computing the number of links for N sites takes $O(N)$ time, and line 6, which constsis of finding the minimum links has complexity $O(N)$. When considering the for loops in lines 1 and 2, the overall complexity is $O(D * N * K * N)$. Substituting D by $D_{max} = 2(K-1)$, the complexity of the algorithm is $O(N^3)$ since $k = N$.

Considering the example above, where $N = 10$, the approximation algorithm works as shown in Fig. 4.

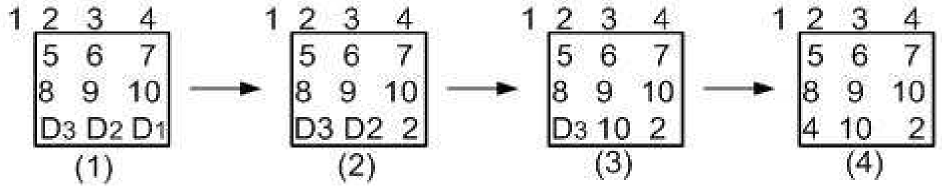


Figure 4: N=10. (1) 3 dashes need to be replaced. (2) D_1 is optimally chosen. (3) D_1 is kept, and D_2 is optimally chosen. (4) D_1 and D_2 are kept, and D_3 is optimally chosen.

2.3 Communication set assignment

The algorithms discussed in the previous section are used to generate all communication sets without knowing which set belongs to which node. In this section we propose an algorithm to assign each communication set to a corresoponding node. The algorithm takes as input the communication sets and outputs the appropriate assignments. The complexity of the algorithm is $O(N)$ because the assignment stage is performed in constant time $O(1)$ for each site N.

step 1:	node 1 gets the first set
step 2:	communication sets constructed from each row are assigned to the 2nd node of the set.
step 3:	node 2 gets the set of 2 and first column
step 4:	communication sets generated from each column are assigned to the 2nd node of the set.
step 5:	communication sets generated from each Jump-X diagonal are assigned to $(X+3)^{rd}$ node of the set

Below, we illustrate the algorithm for a network size $N = 13$.

1. Step 1 assigns sets $\{1, 2, 3, 4\}$ to site 1
2. Step 2 assigns set $\{1, 5, 6, 7\}$ to site 5, $\{1, 8, 9, 10\}$ to site 8, and $\{1, 11, 12, 13\}$ to site 11.
3. Step 3 assigns set $\{2, 5, 8, 9\}$ to site 2
4. Step 4 assigns set $\{2, 6, 9, 12\}$ to site 6 and $\{2, 7, 10, 13\}$ to site 7.
5. Step 5 assigns the jump-X diagonals as follows: Sets $\{3, 5, 9, 13\}$ to site 9, $\{3, 6, 10, 11\}$ to site 10 and $\{4, 7, 8, 12\}$ to site 8. Sets $\{4, 5, 9, 12\}$ to site 12, $\{4, 6, 8, 13\}$ to site 13 and $\{4, 7, 9, 11\}$ to site 11.

3 Network topology charecteristics

In the following section, we point out the characteristics of our network topology with communication sets constrainted by the minimum number of connecting links or the minimum communication cost.

3.1 Theorem 1

Among all the network topologies that satisfy the 4 conditions discussed in section 2.1, our algorithm produces a topology with a minimum number of communication links.

Proof:
The number of communication links in the generated network is $L = (\sum |S|) - D$, where $|S|$ represents the cardinality of a communication set and D represents the number of dashes used during the generation of the communication sets (section 2.2). The number of communication links is the summation of cardinality of all communication sets minus the number of dashes. The algorithm discussed in section 2.2.1 always generates the communication sets such that L is minimum.

Basis: For $D = 0$. This is the case when $N = K(K-1)+1$ and the communication sets generated are symmetric and the number of links generated is minimum with respect to the sets satisfying the section 2.1 conditions. Therefore $L_0 = L_{min}$.

Hypothesis: For any $D = i$, let $L_i = L_{min}$.

Induction: Prove that for $D = i+1$, L_{i+1} is minimum.

The algorithm discussed in section 2.2.2 generates the sets for any value of $D \leq j$ starting from $D = 1, 2, \cdots, j$ one at time. An each time, the algorithm takes into consideration those assignments which have minimum number of links. So we start from the basis and then move up to any number of dashes (Fig. 5).
Therefore if L_i is minimum then L_{i+1} is also minimum. There are many possible topologies which have minimum number of links but they are all isomorphic to each other.

3.2 Theorem 2

Among all the network topologies that satisfy the 4 conditions discussed in section 2.1, our algorithm produces a topology with minimum communica-

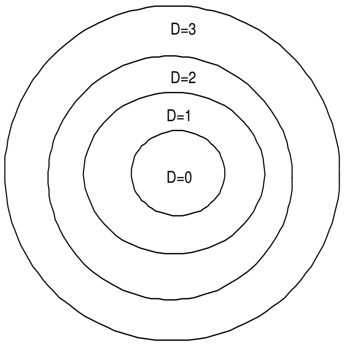


Figure 5: At each step the minimum L is chosen

tion cost.
The proof is exactly the same as the proof discussed above, but here we consider the minimum communication cost instead of the minimum number of links.

4 Distributed routing approach

A major contribution of this paper is the design of a distributed routing, which achieves a better routing performance by considerably reducing the routing table size that needs to be maintained by each node. In the followiong section, we describe how we implement a distributed routing for the generated network topology.

4.1 Routing Strategy

The distributed approach in designing a routing scheme takes advantage of the Maekawa properties that characterize our network topology. We assume initially that the original network is structured according to the design described in section 2. In this case, if a site needs to establish a path to a destination node located in its communication set then a direct connection can take place. However, if a site needs to establish a path to a destination node not in its communication set, then only one intermediate hop is needed to establish the path. In both cases, the maximum path size between any source and destination pair is composed of two hops. This can be illustrated through the following example. Given a network of size $N = 13$. Following our network topology design with communication sets $N_1 = \{1, 2, 3, 4\}$ and $N_6 = \{2, 6, 9, 12\}$ for sites 1 and 6 respectively. In this case, site 1 can directly establish a path to sites 2, 3, and 4. However in order to establish a path from site 1 to site 6, site one needs to go to the intermediate node 2, which represents the intersection of both communication sets. This design ensures a minimum routing path size.

As the network size gets larger, storing huge routing tables becomes a non desirable feature since it will consume a lot of the available storage resources as well as makes it difficult to maintain the routing tables. With our distributed approach, there is no need for the network nodes to store all network topology information. Our proposed idea is as follows: each node will maintain and store just its communication set. If a request for a path establishment between a source *s* and a destination *d* in the communication set comes in, then a direct communication can take place. However, in case a path request for a destination not in the communication set of the source node, then only the communication set of the destination set is generated on the fly and a route will be established from the source to the destination through the intersection node of the communication sets.

In the next subsection, we illustrate the distributed algorithm using pseudocode.

4.2 On the fly set generation

The basic idea behind the set communication generation is to find the communication sites of any site an the fly without the need of generating all communication sets for each node. Following this approach, in order to route from a source to a destination site, any site needs to know only the communication set of the destination site. In this case, the intersection of the source communication set with the destination communication set represents the intermediate hop that the requested path should use. The algorithm used to achieve this goal takes as input the destination site number and the communication set size (K), and is a follows:

```
1:   generate matrix $M$
2:   find index $(i^{'}, j^{'})$ for site $N$ in $M$
3:   if $(N == 1)$
4:       $S = M(i, j)$, $i = 0$, $j = 0 \rightarrow K$
5:   else if $(N == 2)$
6:       $S = M(i, j)$, $j = 0$, $i = 0 \rightarrow K$
7:   else if $(j^{'} == 0)$
8:       $S = M(i, j)$, $i = i^{'}$, $j = 0 \rightarrow K$
9:   else
10:        Depending on $i^{'}$, find the jump;
           $S = M(i, j)$, $i = 0 \rightarrow K$, $i = i * jump$, $j = 0 \rightarrow K$
11:  end if
```

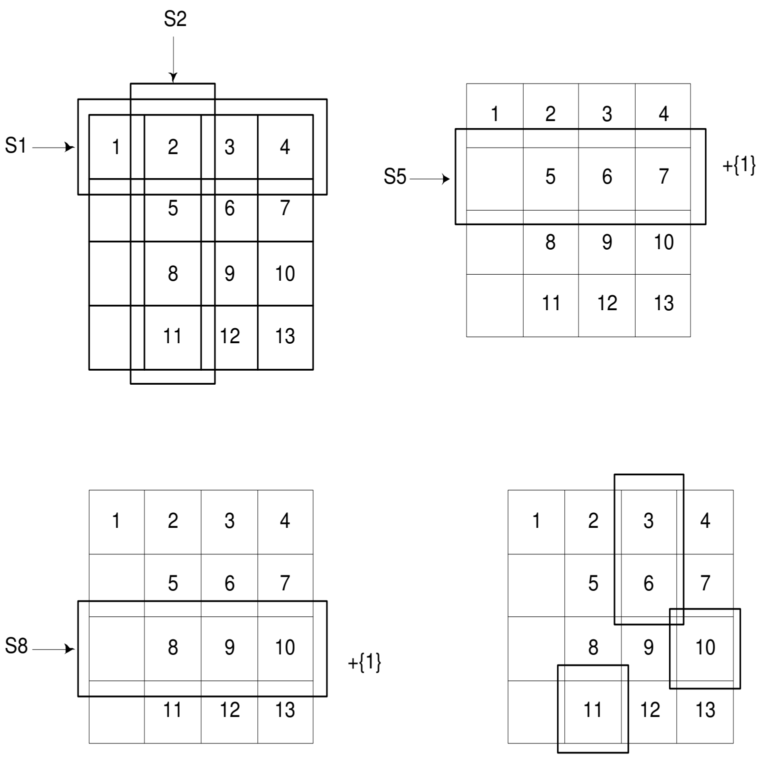


Figure 6: Automatic set generation

Fig. 6 illustrates the above algorithm. It is important to notice that the generation of the communication sets depends only on the site number. Line 2 of the algorithm finds the index of the matrix on which the site falls. If the communication set to be generated is site 1 or site 2, then the request set S consists of first row or first column of the matrix. The complexity of finding the index (i, j) is $O(1)$. To generate the communication sets has order of complexity $O(K)$. Therefore, the overall complexity of the algorithm is $O(K)$.

5 Results

We compare our design to a fully connected network. We consider as network metrices: wiring costs, routing table size, communication cost, and network quality. The following section provides the comparison results.

5.1 Routing Table Size

Usually, an $(N \times N)$ routing table is stored in every node in order to make the routing decisions. Using the suggested topology, no routing table needs to be stored on any one of the nodes. The route is computed on the fly using the algorihtm discussed earlier.

5.2 Wiring Cost

The number of wires in a completely connected network $= \frac{N(N-1)}{2}$ while the number of wires in the described topology $= N(K-1)$. This is due to the fact that every site is connected to the $K-1$ sites in its communication set.

$$Gain = \frac{\sqrt{N}+1}{2} \tag{1}$$

5.3 Communication Cost

The number of hops used in a completely connected network is $N(N-1)$ because each node can reach every other node using one hop. In the suggested topology each node can communicate with other nodes by either 1 or 2 hops. So, the total number of hops is $N\times$ (no. of 1 hop + 2(no. of 2 hops))= $N \times (2(K-1) + 2(K-1)(K-2)) = 2 \times N \times (\sqrt{N}-1)^2$.

$$Loss = 2 \times \frac{\sqrt{N}-1}{\sqrt{N}+1} \tag{2}$$

5.4 Network Quality

The quality of a network is measured using the following equation:

$$Quality = \frac{1}{d \times D \times L} \tag{3}$$

, where d is the node degree, D is the network diameter, and L is the number of links. In a completely connected network: $d = (N-1)$, $D = 1$, and $L = N(N-1)/2$. Therefore,

$$Q1 = \frac{2}{(N-1)^2 \times N} \tag{4}$$

In the suggested topology: $d = 2 * (K-1)$, $D = 2$, and $L = 2 * N * (K-1)$

$$Q2 = \frac{1}{8 \times N \times (\sqrt{N}-1)^2} \tag{5}$$

5.5 Analysis

It can be clearly seen that the gain (Eq. 1) is much bigger from the loss (Eq. 2). As N increases, the gain increases linearly (Fig. 7), while the loss is bounded by a constant (Fig. 8). Moreover, the quality of the suggested network topology is better than that of the completely connected network.

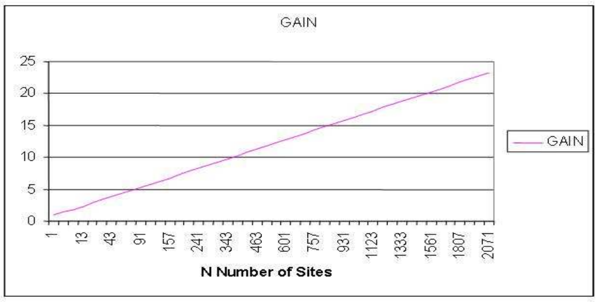


Figure 7: Gain increases linearly

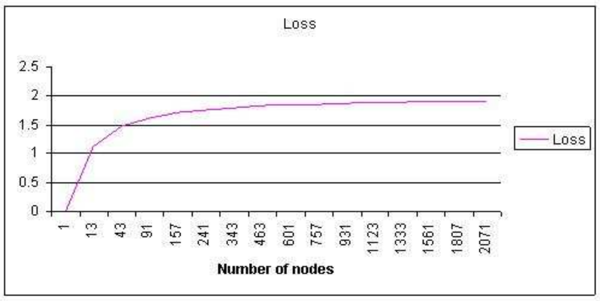


Figure 8: Loss converges to a constant

6 Conclusion

Routing protocols are considered as one of the major componenets in a distributed environment. Having a routing protocol with a good performance is very important especeially with increasaing network sizes as well as increasing demands for high QoS. In the first part of the paper we proposed a new network routing topology design with minimum number of links with a topology that follows the Maekawa request set properties. In the second part of the paper we proposed a distributed routing protocol. We illustrated by simulation that this protocol reduces both communication overhead and storage space.

References

[1] A distributed algorithm for delay-constrained unicast routing. In *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Kobe, Japan*, volume 24, no. 3, pages 848–862, 1997.

[2] C. Gavoille. Routing in distributed networks: Overview and open problems. In *ACM SIGACT News,32 (1):36-52*, March 2001.

[3] A. Kumar J. Xu and X. Yu. On the fundamental tradeoffs between routing table size and network diameter in peerto-peer networks. In *To Appear in IEEE JSAC*, November 2003.

[4] X. Jia. A distributed algorithm of delay-bounded mul-ticast routing for multimedia applications in wide area networks. In *IEEE/ACM Transactions on Networking*, pages 828–837, December 1998.

[5] D. Johnson and D. Maltz. Dynamic source routing in adhoc wireless networks. In *In T. Imielinski and H. Korth, editors, Mobile Computing*, pages 153–181, 1996.

[6] C P Low and Y J Lee. Distributed multicast routing, with end-to-end delay and delay variation constraints. In *ComputerCommunications*, volume 24, no. 3, pages 848–862, 2000.

[7] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. In *ACM Trans. Computer Systems*, May 1985.

[8] George N. Rouskas and Illia Baldine. Multicast routing with end-to-end delay and delay variation constraints. In *IEEE Journal on Selected Areas in Communications*, volume 15, no. 3, pages 346–356, 1997.

[9] A. Brodnick S. Carlson, M. Degermark and S. Pink. Small forward tables for fast routing lookups. In *In Proceedings of SIGCOMM 97 conference on application technologies*, 1997.