ABSTRACT

TREE BASED RELIABLE TOPOLOGY FOR DISTRIBUTING LINK STATE INFORMATION

by

Ram Narayan Krishnan

Finding paths that satisfy the performance requirements of applications according to link state information in a network is known as the Quality-of-Service (QoS) routing problem and has been extensively studied. However, distributing link state information may introduce a significant protocol overhead on network resources. In this thesis, the issue on how to update link state information efficiently and effectively is investigated. A theoretical framework is presented, and a high performance link state policy that is capable of minimizing the false blocking probability of connections under a given update rate constraint is proposed. Through theoretical analysis, it is shown that the proposed policy outperforms the current state of the art in terms of the update rate and higher scalability and reliability.

TREE BASED RELIABLE TOPOLOGY FOR DISTRIBUTING LINK STATE INFORMATION

by

Ram Narayan Krishnan

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

August 2003

APPROVAL PAGE

TREE BASED RELIABLE TOPOLOGY FOR DISTRIBUTING LINK STATE INFORMATION

Ram Narayan Krishnan

Dr. Nirwan Ansari, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, and
Computer and Information Science, Associate Chair for Undergraduate Studies, NJIT

Dr. Roberto Rojas-Cessa, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Ram Narayan Krishnan

Degree: Master of Science

Date: August 2003

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ, 2003

- Bachelor of Technology in Computer Science & Engineering
  Pondicherry Engineering College, Pondicherry, India, 2001

Major: Computer Engineering

To my family and friends, who with their love and support have encouraged me throughout my graduate studies.

ACKNOWLEDGMENT

From the formative stages of this thesis to the final draft, I owe an immense debt of gratitude to my advisor, Dr. Nirwan Ansari. His sound advice and careful guidance were invaluable during the course of my research for this thesis. At the same time, I am also highly indebted to Dr. Edwin Hou and Dr. Roberto Rojas-Cessa for participating in the thesis committee and providing valuable suggestions for improvement.

I would also like to take this opportunity to thank Gang Cheng for his support and advice at various stages of this thesis. His suggestions on numerous occasions helped me define the scope of my thesis and find a better solution to the problem I faced.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

Link State Protocol is the standard and conventional intra-domain routing protocol. Each router in the network advertises its link state information to its adjacent routers by what is known as Link State Advertisement (LSA in OSPF). LSAs are normally cost parameters, which might be a measure of anything like bandwidth, delay or for instance any traffic engineering property. These costs are manually assigned to each link in the network. Any change in the cost will necessitate flooding the entire network again, thus causing a ripple effect. While some cost parameters like delay and bandwidth for one link might appear to remain constant, it drastically changes when the network is perceived globally. For example, in QoS (Quality of Service) routing, when bandwidth is allocated and de-allocated on a particular path, the cost parameters keep changing dynamically forcing to start flooding time and again. This problem magnifies when the network has to be scaled. The network remains under-utilized since much time is spent on network stabilization and convergence.

Assuming Router A changes its cost (Figure 1.1), it is observed that Router D receives LSAs from Routers A, B and C, although D would discard the LSA that was received later. Thus, on a large network, most of the LSAs are discarded as duplicates. This is a serious impediment having a direct impact on scalability as the amount of overhead involved might increase exponentially in a densely connected network.

The topology in Figure 1.2 (numerical values over the links indicate the costs) is used for the purpose of all the forthcoming discussion.

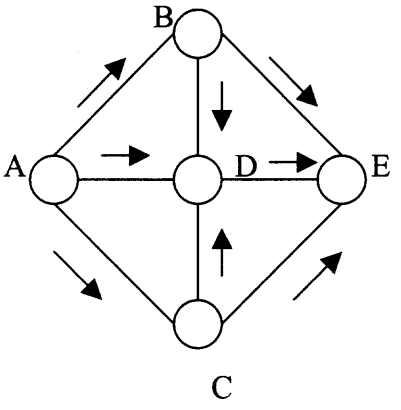

Figure 1.1 Sample Network Topology.

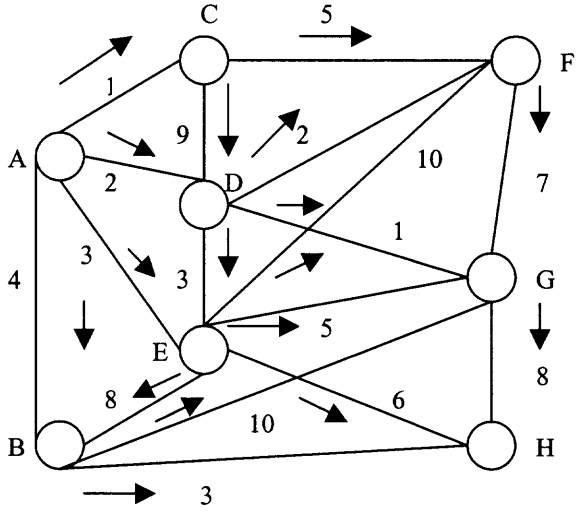

Figure 1.2 Network Topology Showing Link Weights Used for Analysis.

1.2 Background Information

Multiple and diverse applications with various quality of service (QoS) requirements are expected to be supported by the broadband integrated services network. Accordingly, a key issue is to select feasible paths that satisfy these (QoS) requirements. This problem is known as QoS routing [1].

1.2.1 QoS Routing

Consumer applications such as streaming live videos, packetized voice, multiplayer games, and Worldwide Web-based shopping, are now commonplace. Businesses also depend on the network for providing electronic storefronts, support and service to their customers, and a means to conduct day-to-day operations. The applications that deliver these new services introduce new traffic characteristics and impose new requirements on network performance, reliability, and availability. Yet, the Internet's fundamental service consists only of a packet delivery system that makes no promise regarding reliability, timeliness, or in-order delivery. The network follows a "best-effort" paradigm in which all packets are treated identically, regardless of the user application. This 'one model for all' architecture cannot carry on for long due to the proliferation of the above mentioned applications. Supporting these new types of applications requires more sophisticated mechanisms for link scheduling, buffer management, and route selection, all of which play an important role in meeting the new demands on the network. Examples of QoS include guarantees on network delay, throughput, or loss, for either individual application flows, or groups of flows.

1.2.2 Conventional Routing

Link-state protocol (e.g., OSPF or IS-IS) is the dominant type of IGP used in the current Internet. Each link in the network is manually assigned a preconfigured cost that may reflect the capacity or delay, for example. Each router in the network distributes information about the cost of its incident links to all other routers in the domain. Since the costs are relatively static quantities, this link-state information need not be distributed frequently, thus limiting the control and computational overheads. Using the received information, each router computes the shortest path to every other node in which the distance is in terms of the link costs. Routers recompute paths relatively infrequently, for example, only when new link-state information is received. These routes are stored in a next-hop forwarding table so that when a packet arrives, the router simply looks up the destination in the table and forwards the packet on the corresponding interface.

1.2.3 Providing QoS (Dynamic Routing)

While the conventional intra-domain routing described above is relatively simple, and exhibits low overhead, it offers little flexibility in managing network traffic. At best, an ISP may set link costs according to some notion of an expected traffic pattern such that traffic is distributed evenly throughout the network. However, when the volume of traffic between particular points shifts unexpectedly, the network load may become significantly imbalanced, leading to poor performance and utilization. These fluctuations may arise, for example, due to variations in user demand and changes in the network configuration, including failures or reconfigurations in the networks of other service providers. Network providers rely on coarse measurement tools to discover performance problems in the

network, and when the problem requires adjustment of the network configuration, the ISP typically has to manually reroute traffic.

These challenges have spurred increased interest in dynamic routing as a tool for managing network traffic and providing QoS guarantees. By selecting paths based on link utilization, dynamic routing responds to long and short timescale fluctuations in the traffic pattern, and automates the process of redirecting traffic. In doing so, it balances the network load and improves the overall network utilization. Furthermore, choosing routes based on resource availability rather than static link weights provides the ability to satisfy per-flow QoS requirements and improve application performance.

Despite these potential advantages, however, most backbone networks still employ static link state routing (e.g., based on routing protocols such as OSPF). Unlike static routing, load-sensitive routing algorithms require accurate and frequently distributed link-state information to make good routing decisions. Dynamic routing is particularly sensitive to link-state staleness. When excessive staleness occurs, out-of-date information leads routers to direct most traffic to a seemingly attractive path while an alternative path lies under-utilized. A new update arriving to correct the view causes the router to redirect all traffic to the underutilized path, reversing the roles of the routes. Frequent distribution of link-state information prevents oscillation, but runs the risk of flooding the network with control traffic. Similar issues apply to route computation. Accurate route selection requires that routers compute paths using the latest link-state information frequently (usually with more sophisticated and complex algorithms), which incurs higher computational overhead.

Recent research has focused primarily on three areas of dynamic and QoS routing. Theoretical work proposes new algorithms for QoS routing that optimize multiple QoS metrics (e.g., delay and throughput), or compute multicast routing trees subject to QoS requirements. Performance evaluation work compares the performance of several route selection algorithms, most often under specific network and traffic configurations. Finally, protocol development efforts consider issues such as link-state distribution policies, path set-up mechanisms, and integration into existing intra-domain routing protocols.

In order to guarantee convergence of a link state routing protocol, it is vital to ensure that link state Process Data Units or PDU (Link State Advertisements or LSAs in the case of OSPF) are delivered to all routers within the flooding scope limits. The scope can be an area or the whole AS depending on the protocol and the type of the link state PDU. The method used by link state protocols to achieve this implies that a) PDUs are transmitted reliably between any pair of routers, and b) whenever a new PDU is received, it is sent across all interfaces other than the one it was received on.

To fulfill the first requirement, link state routing protocols keep retransmitting new PDUs to the neighbors that have not acknowledged reception. As an example, in OSPF, a link state retransmission list is maintained for every neighbor of each interface. When an LSA is sent through an interface, it is put on the retransmission list of every neighbor associated with this interface and is removed from it only after the neighbor has acknowledged reception of the LSA.

Thus, in general, two issues are critical to QoS routing: state distribution and routing strategy [2]. As already discussed, routing strategy is used to find a feasible path,

which meets the QoS requirements; this has been extensively studied in the literature [3]-[8]. State distribution addresses the issue of exchanging the state information throughout the network and can be further decomposed into two sub-problems: when to update and how to disseminate the link state information. In this thesis research, the state distribution, especially on the latter sub-problem, is focused. A number of research works have also been reported on when to disseminate the link state information [9]-[12], which is, however, beyond the scope of this research.

1.2.4 Link State Flooding

Many existing link-state routing protocols recommend that link state information should be disseminated by simply flooding or flooding-like approaches. As a result, they possess the advantage of robustness, i.e., on the cases of link failures and node failures, link state information is still reachable to all nodes as long as the network is connected. On the other hand, because of the poor scalability of flooding, a large update interval has to be adopted in order to reduce the protocol overhead on network resources. For instance, a link disseminates its state information every 30 minutes in OSPF. Consequently, because of the highly dynamic nature of link state parameters, the link state information known to a node is often outdated. Hence, the effectiveness of the QoS routing algorithms may be degraded significantly. Moreover, distributing link state information by flooding also unnecessarily wastes network resources.

1.2.5 Driving Force

If multiple links connect two routers, flooding of new information will cause considerable overhead of link bandwidth and CPU time spent by the protocol. Consider an example shown in Figure 1.3 [11].

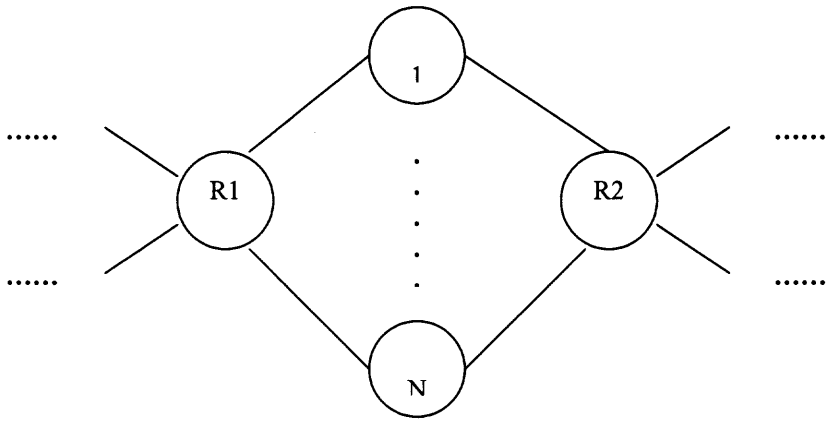

Figure 1.3 Illustration of Flooding Overhead.

When a new PDU is received at R1 from its LAN segment, it is stored in the database of R1 and flooded through all of R1's interfaces. Since flooding presumes sending the new PDU over all interfaces except from the one it was received, routers end up doing the following:

1) R1 sends not one, but N copies of the new PDU to R2.
2) Only the first copy of the PDU is actually installed in R2's data structure, but link bandwidth and CPU cycles are spent to transmit and process all N copies.
3) Furthermore, when R2 receives the first copy of the LSA and installs it, it floods back to R1 N-1 copies of it, again spending extra bandwidth and CPU time.

4) If R1 receives an acknowledgment from R2 on some links, but not from others, it will keep retransmitting unacknowledged LSAs though they are already in R2's LSDB.

The solution described in this document provides a technique to minimize the overhead that link state routing protocols cause in the described situation and use link bandwidth more efficiently.

1.3 Brief Survey of Existing Solutions

Zinin and Shand's [11] proposed idea is to move the flooding algorithm from the per-interface to per-neighbor basis. The technique is generic for all protocols utilizing reliable flooding and is based on the observation that the ultimate goal of the flooding algorithm is not to send link state PDUs over all interfaces, but to deliver them to all routers in the network. To implement this optimization, it is necessary to maintain a list of neighbors within an area. Whenever a new neighbor is discovered on an interface belonging to the area, the corresponding interface neighbor data structure is linked to the corresponding element in the list of neighbors. Based on the information in the list of neighbors, as well as on the type of interfaces they use, interfaces within the area are marked either flooding-active or flooding-passive. The process of election of flooding-active interfaces takes into consideration the costs of interfaces, giving preference to faster interfaces. Multi-access interfaces need special treatment since they may be (usually are) associated with more than one neighbor. However, if such an interface connects only two routers, it still may be marked as flooding-passive. Whenever the number of entries in the list or state of the adjacency in the list changes, the interface election algorithm is rerun. Note

that since the flooding paradigm is changed from the per-interface to per-neighbor basis, PDU retransmission is not performed for a specific neighbor on a specific interface, but is instead done for a specific neighbor in general, and it is enough to receive a single acknowledgment on any interface for sending router to stop retransmitting.

Kleinrock and Kamoun's [13] solution to this problem is to use the spanning tree of the given topology. Figure 1.3 is the minimum spanning tree of the topology in Figure 1.2 computed using Prim's algorithm [14].

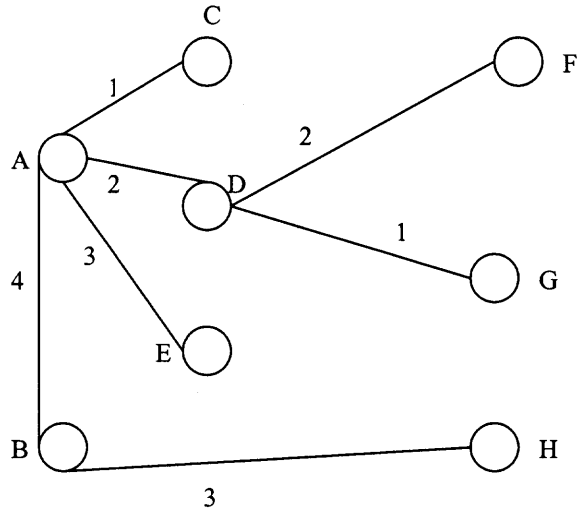

Figure 1.4 Minimum Spanning Tree for Distributing Link State Information.

Since a spanning tree is used, there are no redundant LSAs. Although it reduces the overhead by eliminating redundant LSAs, it makes the network less reliable. When any link goes down, the whole network might get divided into many forests. For instance, if the link between routers A and D goes down, the network becomes two separate trees. Any changes originating at routers A, B, C or E cannot be communicated to routers D, F

and G and vice-versa. Thus, the objective is to propose a more reliable and robust solution to this problem while guaranteeing minimum overhead.

Bellur and Ogier [15] proposed Topology Broadcast based on Reverse Path Flooding (TBRPF) algorithm to broadcast topology information to all nodes of a communication network. The basis for their algorithm is Reverse Path Flooding, in which messages generated by a given source are broadcast in the reverse direction along the directed spanning tree formed by the shortest paths from all nodes to the source. The major difference is that it uses minimum-hop trees (based on number of hops) instead of shortest-path trees (based on link costs) which results in less frequent changes in the broadcast trees since the topology of the network does not change that rapidly. Apparently, this disregard to the link costs is totally unfavorable for cost-effective routing.

Cain [16] used Reverse Path Flooding again in Fast Link State Flooding Algorithm. The convergence time of the network using conventional flooding is affected due to various aspects like time for failure detection, the flooding time itself, and the hold-down time (wait time to receive multiple LSAs before starting to calculate the shortest paths). This algorithm uses the data forwarding paths of the routers to send the Link State Updates (LSU). If a router detects a failure in one of its links, it sends a fast LSU to the flooding address. When the router whose direct link was cut from the source router receives the packet, it detects that it did not receive the packet on the interface that is closest to the source and hence floods the LSA using conventional flooding. This approach uses a combination of their algorithm with the existing conventional flooding and concentrates more on faster convergence of the network.

Chousdhury, Maunder and Sapozhnikova [17] discussed fast detection of failures. The proposed idea shows that faster hello exchanges (for detection of existence of each router), fast flooding and more frequent shortest path calculations reduces the scalability and stability of network as the hello exchange packets cannot be distinguished from other less critical packets by the receiving router. It proposes that marking such critical packets like the hello exchange would enable the router to queue them separately and prioritize them, thus improving network convergence time and stability.

Miyamura, Kurimoto and Aoki [18] proposed an improved solution to this problem. The algorithm first finds the spanning tree of the given topology. It then adds a new link to every node, which has a degree of one in the spanning tree but had a degree greater than one in the original topology (the new link to be added is determined based on its cost).

Most of the approaches for providing reliability do not make sure that the cardinality of the set of minimum edge cut of the entire topology is at least two. This is an essential condition to provide an alternate path for an LSA transfer through an alternate router if the other router is busy processing some other request or the router itself is temporarily unavailable for some reason. Thus, what is required is that at least two edges need to be cut to split the network into two or more islands. This is an essential condition to be satisfied for the topology to be robust.

CHAPTER 2

PROPOSED SCHEME

2.1 Objective

Intuitively, one would find a minimum Hamilton Circuit [14] from the given graph and use the path to send the LSAs. Hamilton Circuit is a cycle that starts at a particular source and passes through every vertex of a graph exactly once and reaches the source at the end. The major disadvantage of this approach is that finding a minimum Hamiltonian Circuit in a given graph is NP-complete, and hence would not be practical to use for fast convergence in real time networks. Also, the topology would not be robust.

In this section, a simple scheme that makes the network robust while reducing LSA related overhead as much as possible is proposed to enhance scalability [19]. The spanning tree (Figure 1.4) of the given topology in Figure 1.2 has already been derived. Now, a new graph is formed by removing the links present in the tree shown in Figure 1.4 from the original topology as shown in Figure 2.1. Note that some nodes (routers) may be left out due to this operation and they are discarded, as "don't cares" (node A in this case).

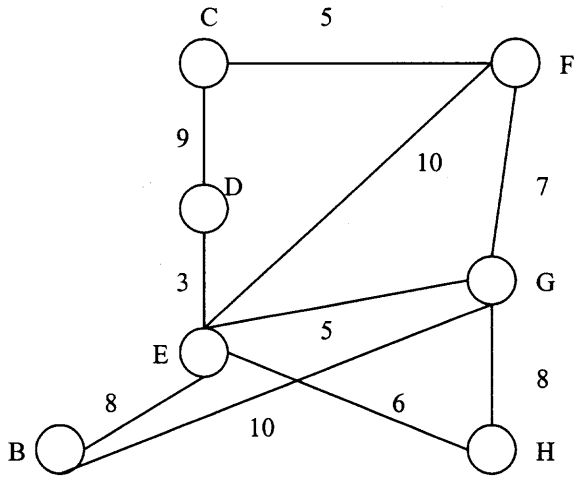

Figure 2.1 Intermediate Graph.

A spanning tree of this graph in Figure 2.1 is found again, which is shown in Figure 2.2. The final graph shown in Figure 2.3 is obtained by combining the two spanning trees obtained in Figure 1.4 and 2.2. Now, Figure 2.3 is the topology that is used to establish adjacencies for each node. Note that the degree of every node in this graph is greater than one (for every node that had a degree greater than one in the original topology in Figure 1.2) and the minimum edge cut of the graph is greater than or equal to 2. Another important thing to be noted is that when the Intermediate Graph need not be a single graph. It could be multiple graphs. So, the spanning tree of each of those Intermediate Graphs should be found and combined with the first spanning tree in Figure 1.4. This approach is referred to as Reliable Graph from Multiple Spanning Trees (RGMST).

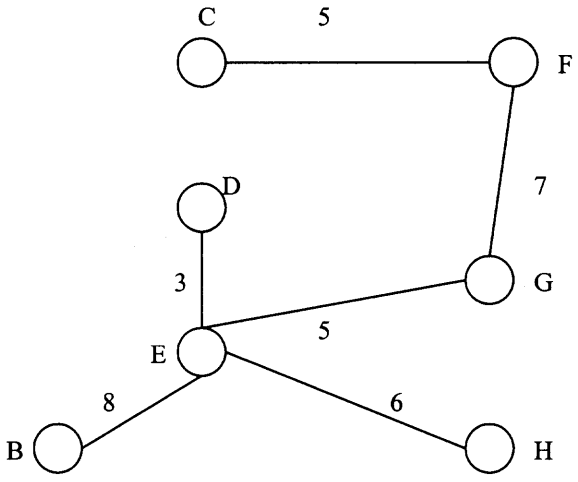

Figure 2.2 Spanning Tree of the Intermediate Graph.

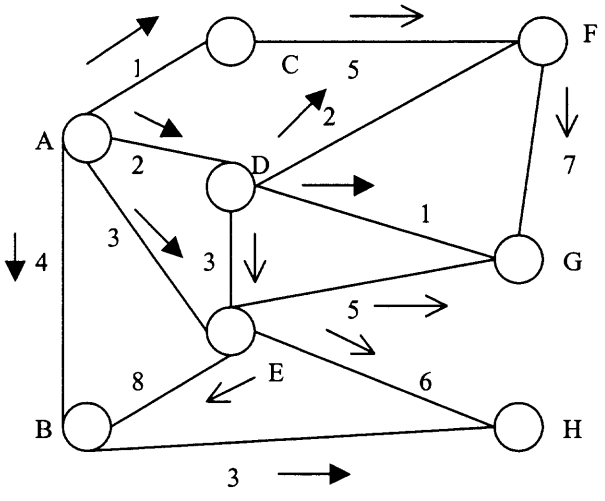

Figure 2.3 Final Graph to Establish Node Neighbors.

2.2 Algorithm RGMST

1. Find the minimum spanning tree (T1) of the original topology, G1
2. Remove each edge present in T1 from G1. This gives a new graph G2
3. Find the minimum spanning tree (T2) of the graph G2

4. Combine the spanning trees T1 and T2 which gives G. (A simple procedure is to add each link present in T2 to T1)
5. Graph G is the target graph which is used to establish adjacencies for each node

Now, the way flooding occurs in this graph is discussed. Each node sends out LSAs to its adjacent nodes, say first as per the spanning tree in Figure 1.4 and then again as per the spanning tree in Figure 2.2. Assume that the origination of an LSA is node A. Consider node D. At some time t, node D first sends LSA to nodes F and G according to the adjacency established in Figure 1.4. Next, it sends LSA to node E according to the adjacency established in Figure 2.2. If no adjacency exists for a node in one of the spanning trees, obviously no LSAs are sent to that node in its spanning tree. For example, node A does not have adjacent nodes in the second spanning tree (Figure 2.2) and hence no LSAs are sent.

2.3 Complexity Analysis

The graph can be represented by two ways [14]:

1. Adjacency Lists- Each node has a linked list associated with it. The list contains the set of adjacent nodes of this node.

2. Adjacency Matrix- If a link is present between two nodes, the corresponding value in the row-column is set to 1 else to 0.

The complexity of the algorithm varies as per the way the graph is represented. Assuming Prim's algorithm is used to construct the minimum spanning tree. Adjacency Matrix representation is used, and hence the complexity would be of O ($e\log(n)$) (e is the number of edges, n is the number of nodes).

Suppose E and N are the sets of edges and nodes in the given graph G, and E_i and N_i (i>=1) the sets of edges and nodes in one or more of the intermediate graphs. The complexity of step 1 of our algorithm is $E\log N$, and that of step 3 would be $E_i\log N_i$ (i varies from 1 to n, where n is the number of intermediate graphs). Hence, the complexity of our algorithm RGMST would be O ($E\log N$) where E is the max (E_i) (i varies from 1 to n) and N is max(N_i) (from the basic algorithm theory that if $T_1(n)=f(n)$ and $T_2(n)=g(n)$, then $T_1(n)+T_2(n)=O(\max(f(n),g(n)))$. Thus, the complexity of RGMST remains pretty much comparable to that of a regular spanning tree algorithm for dense graphs (where E is very large).

2.4 Proof of the RGMST

In this section, the proposed solution is defined and proved by mathematical analysis.

2.4.1 Definitions

Minimum Edge Cut: Given a graph (G, E), the set minimum edge cut is defined as a set with the least number of edges, E_m (E_m is a subset of E), such that only when we remove all the edges in E_m from the graph G, G can be split into two or more sub-graphs.

Reliable Graph: A reliable graph is defined as a Graph G_r with $c(E_m) \geq 2$ where $c(E_m)$ is the cardinality of its minimum edge cut.

2.4.2 Problem Formulation

Given a network topology G with a cardinality of the set of minimum edge-cut greater than or equal to two, Algorithm RGMST produces a topology with the minimum edge cut of cardinality of at least two.

2.4.3 Proof

Let Eg be the minimum edge cut set of the given graph G. Let $Eg=\{e_1, e_2, ..., e_n\}$. Thus, $c(Eg)=n$, $n \geq 2$ (this is the basic assumption). The validity of the proposed solution is established by proof by contradiction. Hence, assume that the solution does not hold the formulated problem. Thus, $c(E_m)=1$ (E_m is the minimum edge cut of the final topology in the proposed solution); arbitrarily assume two nodes m and n that are not directly connected, and m is a member of G_1 and n is a member of G_2. Since $c(E_g)=n$, the given graph G can be drawn as shown in Figure 2.4. G can be split into two parts as G_1 and G_2 connected by all the edges in E_g. Thus, if the edges $e_1, e_2, ..., e_n$ are removed, the graph G will be split into two parts G_1 and G_2.

As per step 1 of the algorithm RGMST, the spanning tree $T1$ of the graph G is found, as shown in Figure 2.5. As per definition of a spanning tree, there must be at least one edge, e_{uv}, connecting G_1 and G_2.

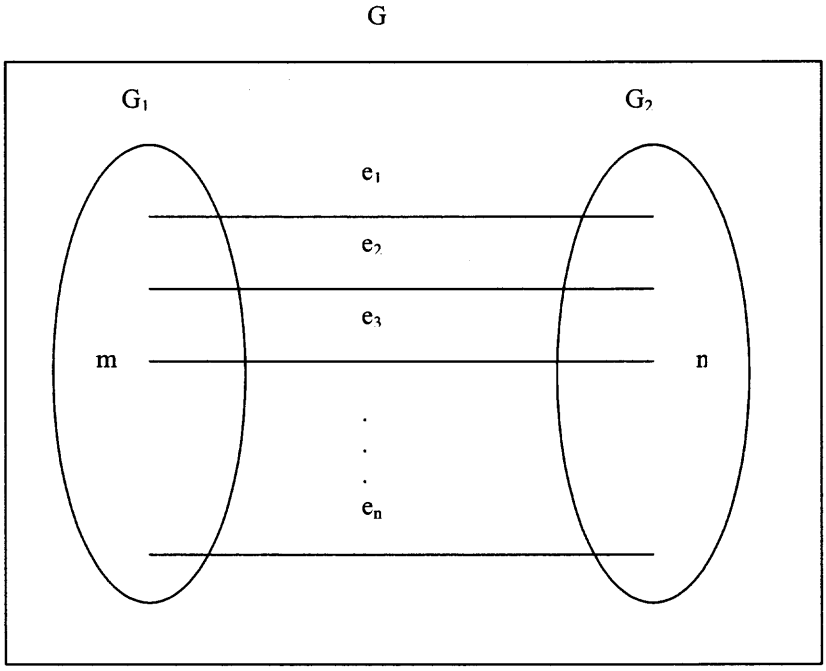

Figure 2.4 Given Network Topology: Edge-Cuts Explicitly Shown.

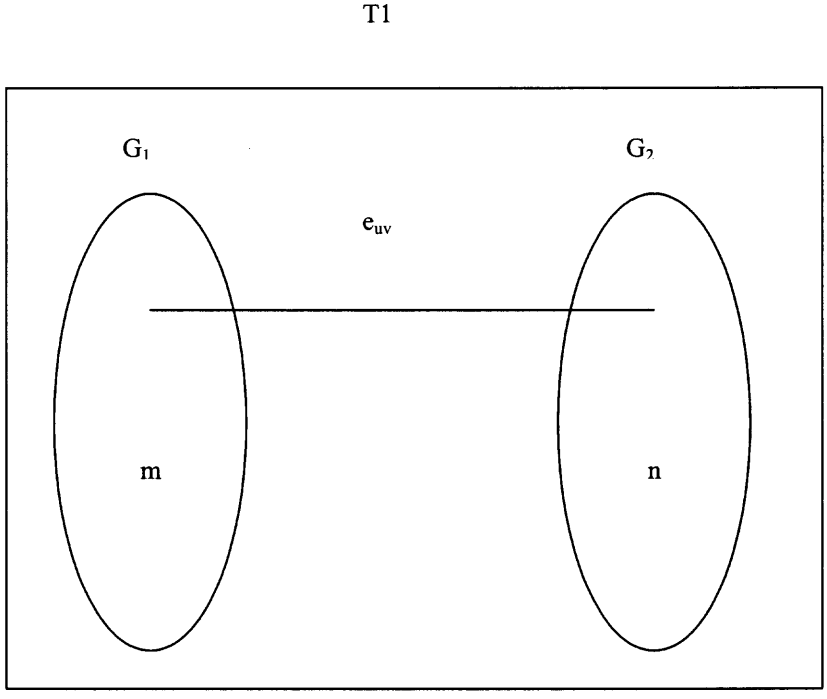

Figure 2.5 Spanning Tree of Given Topology.

Now, the vertices m and n are not directly connected as shown in Figure 2.5. As per step 2 of the algorithm, edge e_{uv} of $T1$ would surely be removed from G to form a set of intermediate graphs G'. This should leave all the edges other than e_{uv} from E_g in the set of intermediate graphs G' as shown in Figure 2.6. Note that G' should contain the edge e_3 connecting the nodes m and n.

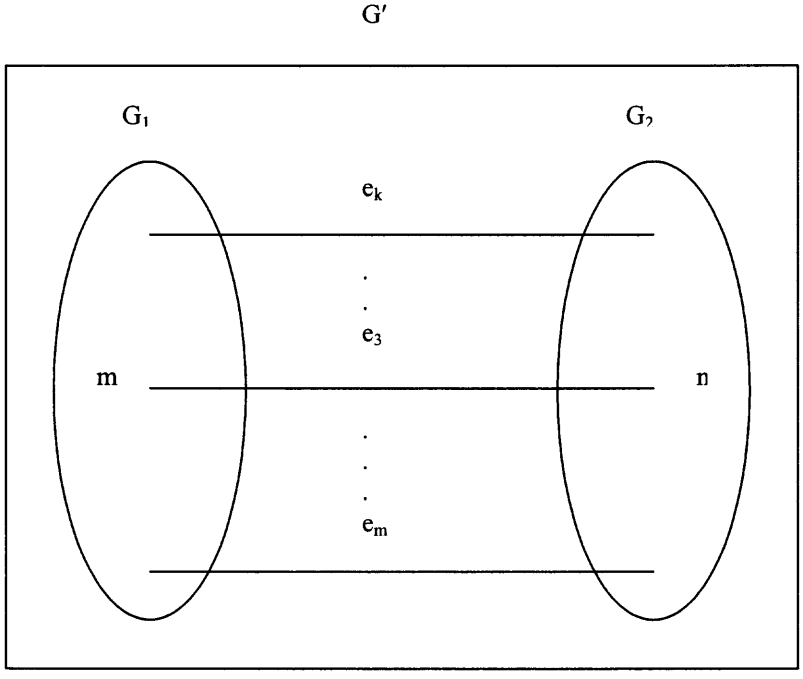

Figure 2.6 Intermediate Network Topology.

Now one of the spanning trees of the sub-graphs in G' should have a path from m to n, and hence the path should necessarily include an edge-cut, say e_p. Thus, in the final step of the algorithm when all the spanning trees are combined, the nodes m and n are connected by two edges (not necessarily a direct connection); e_{uv} from the spanning tree $T1$ and e_p. Note that e_{uv} and e_p are members of E_m, and E_m is a subset of E_g. This contradicts the basic assumption that $c(E_m)=1$. Thus, the reliable graph G_r that is found by the algorithm RGMST should have a minimum edge cut of two.

CHAPTER 3

CONCLUSION

A simple yet robust and efficient link state dissemination algorithm has been proposed. The proposed algorithm provides many properties that are required for QoS-based routing, which involves frequent dissemination of several dynamic parameters. In contrast to earlier works, algorithm RGMST involves less computation and more importantly makes the network robust. It makes sure that the number of edges in the minimum edge cut set of the given network topology is at least two. This provides extra reliability for the network and gives a lot of room for load balancing. A future work would be to demonstrate the algorithm's efficiency along with OSPF.

REFERENCES

[1] Shaikh, A. (1999). Efficient Dynamic Routing in Wide-Area Networks. Ph.D. Thesis, University of Michigan.

[2] Chen, S. & Nahrstedt, K. (1998). An overview of quality of service routing for next-generation high-speed network: problems and solutions (pp. 64-79). IEEE Network Magazine.

[3] Cheng, G. & Ansari, N. (2002). On Multiple Additively Constrained Path Selection (pp. 237-241). IEE Proceedings on Communications, 12, (5) vol. 149.

[4] Guerin, R. & Orda, A. (1997). QoS based routing in networks with inaccurate information: theory and algorithms (pp. 75-83). Proceedings of the INFOCOM'97.

[5] Korkma, T., Krunz, M. & Tragoudas, S. (2000). An efficient algorithm for finding a path subject to two additive constraints (pp. 318-327). Proceedings of the ACM SIGMETRICS'2000.

[6] Gang, L. & Ramakrishnan, K.G. (2001). A prune: an algorithm for finding k shortest paths subject to multiple constraints (pp. 743-749). Proceedings of IEEE INFOCOM'2001, vol. 2.

[7] Chen, S. & Nahrstedt, K. (1998). Distributed QoS routing with imprecise state information (pp. 614-621). Proceedings of 7th International Conference on Computer Communications and Networks.

[8] Wang, Z. & Crowcroft, J. (1996). Quality of Service routing for supporting multimedia applications (pp. 1228-1234). IEEE Journal on Selected Areas on Communications, 14, (7), vol. 14.

[9] Humblet, P.A. & Soloway, S.R. (1988/89). Topology broadcast algorithms (pp. 179-186). Computer Networks and ISDN Systems, vol. 16.

[10] Bellur, B. & Ogier, R.G. (1999). A reliable, efficient topology broadcast protocol for dynamic networks (pp. 178-186). In Proceedings of INFOCOM'99, vol.1.

[11] Zinin, A. & Shand, M. (2001). Flooding optimizations in link-state routing protocols. Networking Group, Internet Draft IETF, draft-ietf-ospf-isis-flood-opt-01.txt.

[12] Moy, J. Flooding over a subset topology (2001). IETF, draft-ietf-ospf-subset-flood-00.txt.

[13] Kleinrock, L. & Kamoun, F. (1997). Hierarchical routing for large networks; Performance evaluation and optimization (pp. 155-174). Computer Networks, vol. 1.

[14] Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (1990). Introduction to algorithms. Cambridge, Mass. MIT Press; NY: McGraw-Hill.

[15] Bellur, B. & Ogier, R.G. A Reliable. (1999). Efficient Topology Broadcast Protocol for Dynamic Networks. SRI International, Menlo Park, CA.

[16] Cain, B. (2000). Fast Link State Flooding. Nortel Networks, Billerica, MA.

[17] Choudhury, G., Maunder A.S, & Sapozhnikova V.D (2001). Faster Link-State IGP Convergence and Improved Network Scalability and Stability. AT&T Labs, Sanera Systems.

[18] Miyamura, T., Kurimoto, T. & Aoki, M. (2003). Enhancing the Network Scalability of Link-state Routing Protocols by Reducing their Flooding Overhead. NTT Network Service Systems Laboratories, NTT Corporation Tokyo, 180-8585 Japan.

[19] Ansari, N., Cheng, G. & Krishnan, R. N. (2003). Efficient and Reliable Link State Information Dissemination (in preparation).

CHAPTER 1

INTRODUCTION

1.1 Objective

Link State Protocol is the standard and conventional intra-domain routing protocol. Each router in the network advertises its link state information to its adjacent routers by what is known as Link State Advertisement (LSA in OSPF). LSAs are normally cost parameters, which might be a measure of anything like bandwidth, delay or for instance any traffic engineering property. These costs are manually assigned to each link in the network. Any change in the cost will necessitate flooding the entire network again, thus causing a ripple effect. While some cost parameters like delay and bandwidth for one link might appear to remain constant, it drastically changes when the network is perceived globally. For example, in QoS (Quality of Service) routing, when bandwidth is allocated and de-allocated on a particular path, the cost parameters keep changing dynamically forcing to start flooding time and again. This problem magnifies when the network has to be scaled. The network remains under-utilized since much time is spent on network stabilization and convergence.

Assuming Router A changes its cost (Figure 1.1), it is observed that Router D receives LSAs from Routers A, B and C, although D would discard the LSA that was received later. Thus, on a large network, most of the LSAs are discarded as duplicates. This is a serious impediment having a direct impact on scalability as the amount of overhead involved might increase exponentially in a densely connected network.

The topology in Figure 1.2 (numerical values over the links indicate the costs) is used for the purpose of all the forthcoming discussion.

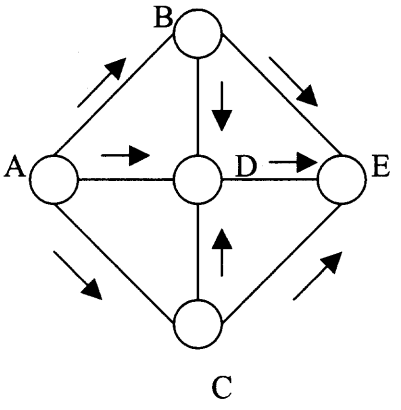

Figure 1.1 Sample Network Topology.

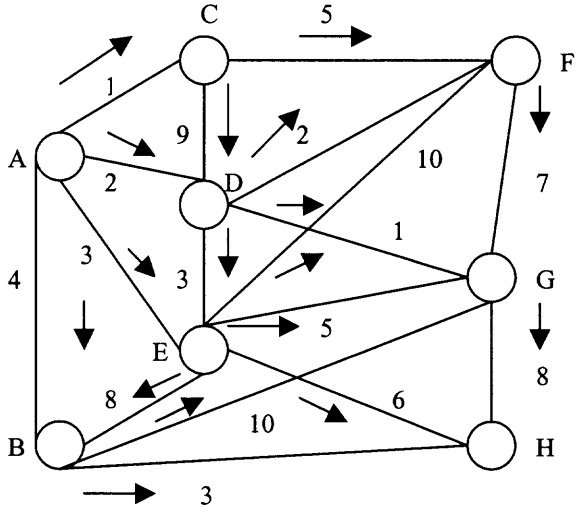

Figure 1.2 Network Topology Showing Link Weights Used for Analysis.

1.2 Background Information

Multiple and diverse applications with various quality of service (QoS) requirements are expected to be supported by the broadband integrated services network. Accordingly, a key issue is to select feasible paths that satisfy these (QoS) requirements. This problem is known as QoS routing [1].

1.2.1 QoS Routing

Consumer applications such as streaming live videos, packetized voice, multiplayer games, and Worldwide Web-based shopping, are now commonplace. Businesses also depend on the network for providing electronic storefronts, support and service to their customers, and a means to conduct day-to-day operations. The applications that deliver these new services introduce new traffic characteristics and impose new requirements on network performance, reliability, and availability. Yet, the Internet's fundamental service consists only of a packet delivery system that makes no promise regarding reliability, timeliness, or in-order delivery. The network follows a "best-effort" paradigm in which all packets are treated identically, regardless of the user application. This 'one model for all' architecture cannot carry on for long due to the proliferation of the above mentioned applications. Supporting these new types of applications requires more sophisticated mechanisms for link scheduling, buffer management, and route selection, all of which play an important role in meeting the new demands on the network. Examples of QoS include guarantees on network delay, throughput, or loss, for either individual application flows, or groups of flows.

1.2.2 Conventional Routing

Link-state protocol (e.g., OSPF or IS-IS) is the dominant type of IGP used in the current Internet. Each link in the network is manually assigned a preconfigured cost that may reflect the capacity or delay, for example. Each router in the network distributes information about the cost of its incident links to all other routers in the domain. Since the costs are relatively static quantities, this link-state information need not be distributed frequently, thus limiting the control and computational overheads. Using the received information, each router computes the shortest path to every other node in which the distance is in terms of the link costs. Routers recompute paths relatively infrequently, for example, only when new link-state information is received. These routes are stored in a next-hop forwarding table so that when a packet arrives, the router simply looks up the destination in the table and forwards the packet on the corresponding interface.

1.2.3 Providing QoS (Dynamic Routing)

While the conventional intra-domain routing described above is relatively simple, and exhibits low overhead, it offers little flexibility in managing network traffic. At best, an ISP may set link costs according to some notion of an expected traffic pattern such that traffic is distributed evenly throughout the network. However, when the volume of traffic between particular points shifts unexpectedly, the network load may become significantly imbalanced, leading to poor performance and utilization. These fluctuations may arise, for example, due to variations in user demand and changes in the network configuration, including failures or reconfigurations in the networks of other service providers. Network providers rely on coarse measurement tools to discover performance problems in the

network, and when the problem requires adjustment of the network configuration, the ISP typically has to manually reroute traffic.

These challenges have spurred increased interest in dynamic routing as a tool for managing network traffic and providing QoS guarantees. By selecting paths based on link utilization, dynamic routing responds to long and short timescale fluctuations in the traffic pattern, and automates the process of redirecting traffic. In doing so, it balances the network load and improves the overall network utilization. Furthermore, choosing routes based on resource availability rather than static link weights provides the ability to satisfy per-flow QoS requirements and improve application performance.

Despite these potential advantages, however, most backbone networks still employ static link state routing (e.g., based on routing protocols such as OSPF). Unlike static routing, load-sensitive routing algorithms require accurate and frequently distributed link-state information to make good routing decisions. Dynamic routing is particularly sensitive to link-state staleness. When excessive staleness occurs, out-of-date information leads routers to direct most traffic to a seemingly attractive path while an alternative path lies under-utilized. A new update arriving to correct the view causes the router to redirect all traffic to the underutilized path, reversing the roles of the routes. Frequent distribution of link-state information prevents oscillation, but runs the risk of flooding the network with control traffic. Similar issues apply to route computation. Accurate route selection requires that routers compute paths using the latest link-state information frequently (usually with more sophisticated and complex algorithms), which incurs higher computational overhead.

Recent research has focused primarily on three areas of dynamic and QoS routing. Theoretical work proposes new algorithms for QoS routing that optimize multiple QoS metrics (e.g., delay and throughput), or compute multicast routing trees subject to QoS requirements. Performance evaluation work compares the performance of several route selection algorithms, most often under specific network and traffic configurations. Finally, protocol development efforts consider issues such as link-state distribution policies, path set-up mechanisms, and integration into existing intra-domain routing protocols.

In order to guarantee convergence of a link state routing protocol, it is vital to ensure that link state Process Data Units or PDU (Link State Advertisements or LSAs in the case of OSPF) are delivered to all routers within the flooding scope limits. The scope can be an area or the whole AS depending on the protocol and the type of the link state PDU. The method used by link state protocols to achieve this implies that a) PDUs are transmitted reliably between any pair of routers, and b) whenever a new PDU is received, it is sent across all interfaces other than the one it was received on.

To fulfill the first requirement, link state routing protocols keep retransmitting new PDUs to the neighbors that have not acknowledged reception. As an example, in OSPF, a link state retransmission list is maintained for every neighbor of each interface. When an LSA is sent through an interface, it is put on the retransmission list of every neighbor associated with this interface and is removed from it only after the neighbor has acknowledged reception of the LSA.

Thus, in general, two issues are critical to QoS routing: state distribution and routing strategy [2]. As already discussed, routing strategy is used to find a feasible path,

which meets the QoS requirements; this has been extensively studied in the literature [3]-[8]. State distribution addresses the issue of exchanging the state information throughout the network and can be further decomposed into two sub-problems: when to update and how to disseminate the link state information. In this thesis research, the state distribution, especially on the latter sub-problem, is focused. A number of research works have also been reported on when to disseminate the link state information [9]-[12], which is, however, beyond the scope of this research.

1.2.4 Link State Flooding

Many existing link-state routing protocols recommend that link state information should be disseminated by simply flooding or flooding-like approaches. As a result, they possess the advantage of robustness, i.e., on the cases of link failures and node failures, link state information is still reachable to all nodes as long as the network is connected. On the other hand, because of the poor scalability of flooding, a large update interval has to be adopted in order to reduce the protocol overhead on network resources. For instance, a link disseminates its state information every 30 minutes in OSPF. Consequently, because of the highly dynamic nature of link state parameters, the link state information known to a node is often outdated. Hence, the effectiveness of the QoS routing algorithms may be degraded significantly. Moreover, distributing link state information by flooding also unnecessarily wastes network resources.

1.2.5 Driving Force

If multiple links connect two routers, flooding of new information will cause considerable overhead of link bandwidth and CPU time spent by the protocol. Consider an example shown in Figure 1.3 [11].

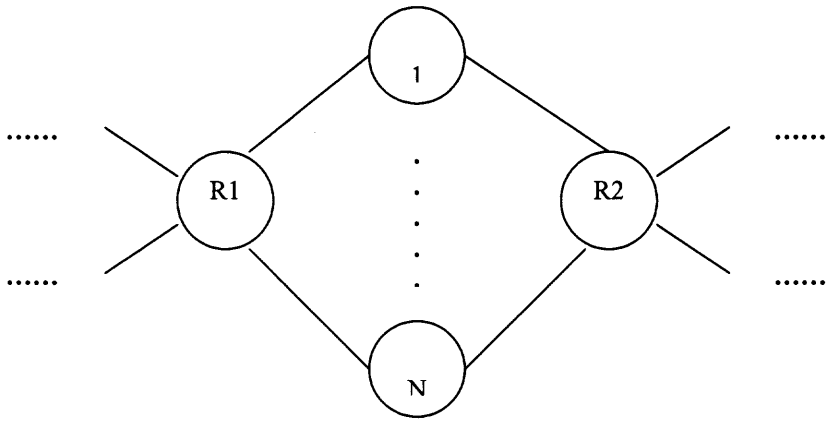

Figure 1.3 Illustration of Flooding Overhead.

When a new PDU is received at R1 from its LAN segment, it is stored in the database of R1 and flooded through all of R1's interfaces. Since flooding presumes sending the new PDU over all interfaces except from the one it was received, routers end up doing the following:

1) R1 sends not one, but N copies of the new PDU to R2.
2) Only the first copy of the PDU is actually installed in R2's data structure, but link bandwidth and CPU cycles are spent to transmit and process all N copies.
3) Furthermore, when R2 receives the first copy of the LSA and installs it, it floods back to R1 N-1 copies of it, again spending extra bandwidth and CPU time.

4) If R1 receives an acknowledgment from R2 on some links, but not from others, it will keep retransmitting unacknowledged LSAs though they are already in R2's LSDB.

The solution described in this document provides a technique to minimize the overhead that link state routing protocols cause in the described situation and use link bandwidth more efficiently.

1.3 Brief Survey of Existing Solutions

Zinin and Shand's [11] proposed idea is to move the flooding algorithm from the per-interface to per-neighbor basis. The technique is generic for all protocols utilizing reliable flooding and is based on the observation that the ultimate goal of the flooding algorithm is not to send link state PDUs over all interfaces, but to deliver them to all routers in the network. To implement this optimization, it is necessary to maintain a list of neighbors within an area. Whenever a new neighbor is discovered on an interface belonging to the area, the corresponding interface neighbor data structure is linked to the corresponding element in the list of neighbors. Based on the information in the list of neighbors, as well as on the type of interfaces they use, interfaces within the area are marked either flooding-active or flooding-passive. The process of election of flooding-active interfaces takes into consideration the costs of interfaces, giving preference to faster interfaces. Multi-access interfaces need special treatment since they may be (usually are) associated with more than one neighbor. However, if such an interface connects only two routers, it still may be marked as flooding-passive. Whenever the number of entries in the list or state of the adjacency in the list changes, the interface election algorithm is rerun. Note

that since the flooding paradigm is changed from the per-interface to per-neighbor basis, PDU retransmission is not performed for a specific neighbor on a specific interface, but is instead done for a specific neighbor in general, and it is enough to receive a single acknowledgment on any interface for sending router to stop retransmitting.

Kleinrock and Kamoun's [13] solution to this problem is to use the spanning tree of the given topology. Figure 1.3 is the minimum spanning tree of the topology in Figure 1.2 computed using Prim's algorithm [14].

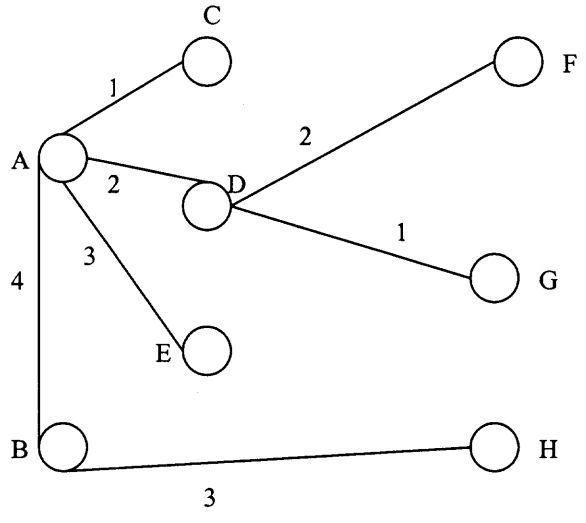

Figure 1.4 Minimum Spanning Tree for Distributing Link State Information.

Since a spanning tree is used, there are no redundant LSAs. Although it reduces the overhead by eliminating redundant LSAs, it makes the network less reliable. When any link goes down, the whole network might get divided into many forests. For instance, if the link between routers A and D goes down, the network becomes two separate trees. Any changes originating at routers A, B, C or E cannot be communicated to routers D, F

and G and vice-versa. Thus, the objective is to propose a more reliable and robust solution to this problem while guaranteeing minimum overhead.

Bellur and Ogier [15] proposed Topology Broadcast based on Reverse Path Flooding (TBRPF) algorithm to broadcast topology information to all nodes of a communication network. The basis for their algorithm is Reverse Path Flooding, in which messages generated by a given source are broadcast in the reverse direction along the directed spanning tree formed by the shortest paths from all nodes to the source. The major difference is that it uses minimum-hop trees (based on number of hops) instead of shortest-path trees (based on link costs) which results in less frequent changes in the broadcast trees since the topology of the network does not change that rapidly. Apparently, this disregard to the link costs is totally unfavorable for cost-effective routing.

Cain [16] used Reverse Path Flooding again in Fast Link State Flooding Algorithm. The convergence time of the network using conventional flooding is affected due to various aspects like time for failure detection, the flooding time itself, and the hold-down time (wait time to receive multiple LSAs before starting to calculate the shortest paths). This algorithm uses the data forwarding paths of the routers to send the Link State Updates (LSU). If a router detects a failure in one of its links, it sends a fast LSU to the flooding address. When the router whose direct link was cut from the source router receives the packet, it detects that it did not receive the packet on the interface that is closest to the source and hence floods the LSA using conventional flooding. This approach uses a combination of their algorithm with the existing conventional flooding and concentrates more on faster convergence of the network.

Chousdhury, Maunder and Sapozhnikova [17] discussed fast detection of failures. The proposed idea shows that faster hello exchanges (for detection of existence of each router), fast flooding and more frequent shortest path calculations reduces the scalability and stability of network as the hello exchange packets cannot be distinguished from other less critical packets by the receiving router. It proposes that marking such critical packets like the hello exchange would enable the router to queue them separately and prioritize them, thus improving network convergence time and stability.

Miyamura, Kurimoto and Aoki [18] proposed an improved solution to this problem. The algorithm first finds the spanning tree of the given topology. It then adds a new link to every node, which has a degree of one in the spanning tree but had a degree greater than one in the original topology (the new link to be added is determined based on its cost).

Most of the approaches for providing reliability do not make sure that the cardinality of the set of minimum edge cut of the entire topology is at least two. This is an essential condition to provide an alternate path for an LSA transfer through an alternate router if the other router is busy processing some other request or the router itself is temporarily unavailable for some reason. Thus, what is required is that at least two edges need to be cut to split the network into two or more islands. This is an essential condition to be satisfied for the topology to be robust.

CHAPTER 2

PROPOSED SCHEME

2.1 Objective

Intuitively, one would find a minimum Hamilton Circuit [14] from the given graph and use the path to send the LSAs. Hamilton Circuit is a cycle that starts at a particular source and passes through every vertex of a graph exactly once and reaches the source at the end. The major disadvantage of this approach is that finding a minimum Hamiltonian Circuit in a given graph is NP-complete, and hence would not be practical to use for fast convergence in real time networks. Also, the topology would not be robust.

In this section, a simple scheme that makes the network robust while reducing LSA related overhead as much as possible is proposed to enhance scalability [19]. The spanning tree (Figure 1.4) of the given topology in Figure 1.2 has already been derived. Now, a new graph is formed by removing the links present in the tree shown in Figure 1.4 from the original topology as shown in Figure 2.1. Note that some nodes (routers) may be left out due to this operation and they are discarded, as "don't cares" (node A in this case).

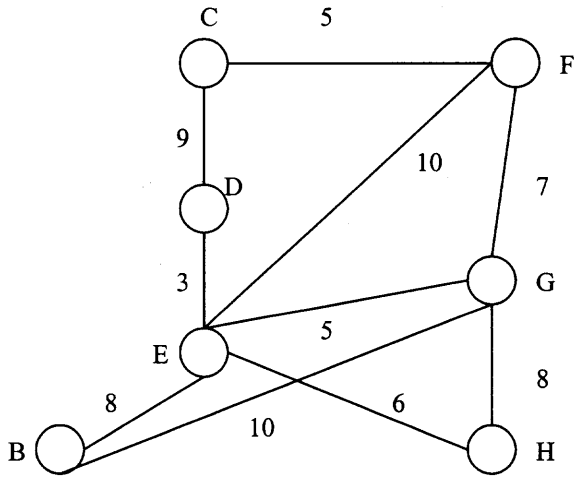

Figure 2.1 Intermediate Graph.

A spanning tree of this graph in Figure 2.1 is found again, which is shown in Figure 2.2. The final graph shown in Figure 2.3 is obtained by combining the two spanning trees obtained in Figure 1.4 and 2.2. Now, Figure 2.3 is the topology that is used to establish adjacencies for each node. Note that the degree of every node in this graph is greater than one (for every node that had a degree greater than one in the original topology in Figure 1.2) and the minimum edge cut of the graph is greater than or equal to 2. Another important thing to be noted is that when the Intermediate Graph need not be a single graph. It could be multiple graphs. So, the spanning tree of each of those Intermediate Graphs should be found and combined with the first spanning tree in Figure 1.4. This approach is referred to as Reliable Graph from Multiple Spanning Trees (RGMST).

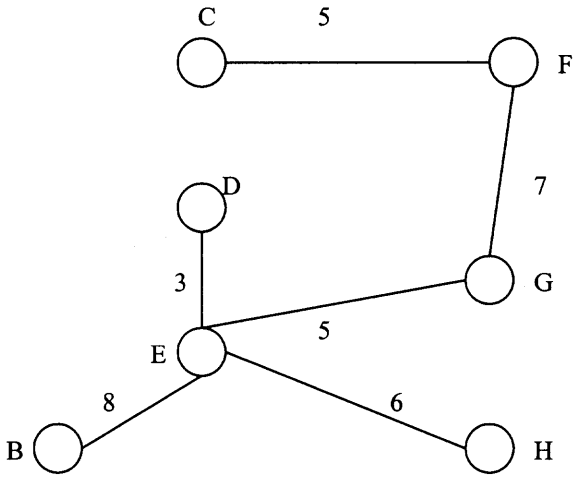

Figure 2.2 Spanning Tree of the Intermediate Graph.

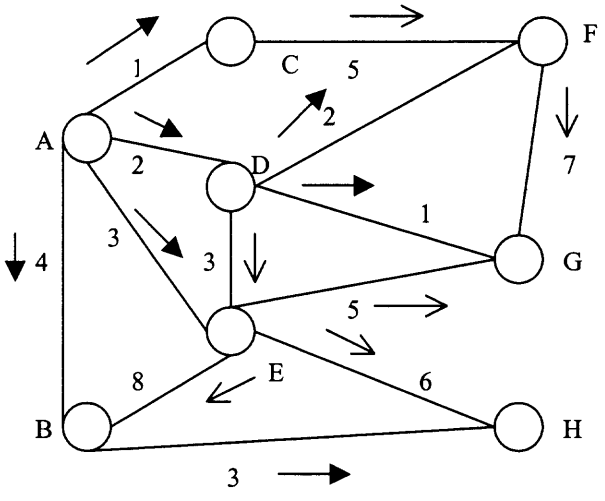

Figure 2.3 Final Graph to Establish Node Neighbors.

2.2 Algorithm RGMST

1. Find the minimum spanning tree (T1) of the original topology, G1
2. Remove each edge present in T1 from G1. This gives a new graph G2
3. Find the minimum spanning tree (T2) of the graph G2

4. Combine the spanning trees T1 and T2 which gives G. (A simple procedure is to add each link present in T2 to T1)
5. Graph G is the target graph which is used to establish adjacencies for each node

Now, the way flooding occurs in this graph is discussed. Each node sends out LSAs to its adjacent nodes, say first as per the spanning tree in Figure 1.4 and then again as per the spanning tree in Figure 2.2. Assume that the origination of an LSA is node A. Consider node D. At some time t, node D first sends LSA to nodes F and G according to the adjacency established in Figure 1.4. Next, it sends LSA to node E according to the adjacency established in Figure 2.2. If no adjacency exists for a node in one of the spanning trees, obviously no LSAs are sent to that node in its spanning tree. For example, node A does not have adjacent nodes in the second spanning tree (Figure 2.2) and hence no LSAs are sent.

2.3 Complexity Analysis

The graph can be represented by two ways [14]:

1. Adjacency Lists- Each node has a linked list associated with it. The list contains the set of adjacent nodes of this node.

2. Adjacency Matrix- If a link is present between two nodes, the corresponding value in the row-column is set to 1 else to 0.

The complexity of the algorithm varies as per the way the graph is represented. Assuming Prim's algorithm is used to construct the minimum spanning tree. Adjacency Matrix representation is used, and hence the complexity would be of O ($e\log(n)$) (e is the number of edges, n is the number of nodes).

Suppose E and N are the sets of edges and nodes in the given graph G, and E_i and N_i (i>=1) the sets of edges and nodes in one or more of the intermediate graphs. The complexity of step 1 of our algorithm is $E\log N$, and that of step 3 would be $E_i\log N_i$ (i varies from 1 to n, where n is the number of intermediate graphs). Hence, the complexity of our algorithm RGMST would be O ($E\log N$) where E is the max (E_i) (i varies from 1 to n) and N is max(N_i) (from the basic algorithm theory that if $T_1(n)=f(n)$ and $T_2(n)=g(n)$, then $T_1(n)+T_2(n)=O(\max(f(n),g(n)))$. Thus, the complexity of RGMST remains pretty much comparable to that of a regular spanning tree algorithm for dense graphs (where E is very large).

2.4 Proof of the RGMST

In this section, the proposed solution is defined and proved by mathematical analysis.

2.4.1 Definitions

Minimum Edge Cut: Given a graph (G, E), the set minimum edge cut is defined as a set with the least number of edges, E_m (E_m is a subset of E), such that only when we remove all the edges in E_m from the graph G, G can be split into two or more sub-graphs.

Reliable Graph: A reliable graph is defined as a Graph G_r with $c(E_m) \geq 2$ where $c(E_m)$ is the cardinality of its minimum edge cut.

2.4.2 Problem Formulation

Given a network topology G with a cardinality of the set of minimum edge-cut greater than or equal to two, Algorithm RGMST produces a topology with the minimum edge cut of cardinality of at least two.

2.4.3 Proof

Let Eg be the minimum edge cut set of the given graph G. Let $Eg=\{e_1, e_2, ..., e_n\}$. Thus, $c(Eg)=n$, $n \geq 2$ (this is the basic assumption). The validity of the proposed solution is established by proof by contradiction. Hence, assume that the solution does not hold the formulated problem. Thus, $c(E_m)=1$ (E_m is the minimum edge cut of the final topology in the proposed solution); arbitrarily assume two nodes m and n that are not directly connected, and m is a member of G_1 and n is a member of G_2. Since $c(E_g)=n$, the given graph G can be drawn as shown in Figure 2.4. G can be split into two parts as G_1 and G_2 connected by all the edges in E_g. Thus, if the edges $e_1, e_2, ..., e_n$ are removed, the graph G will be split into two parts G_1 and G_2.

As per step 1 of the algorithm RGMST, the spanning tree $T1$ of the graph G is found, as shown in Figure 2.5. As per definition of a spanning tree, there must be at least one edge, e_{uv}, connecting G_1 and G_2.

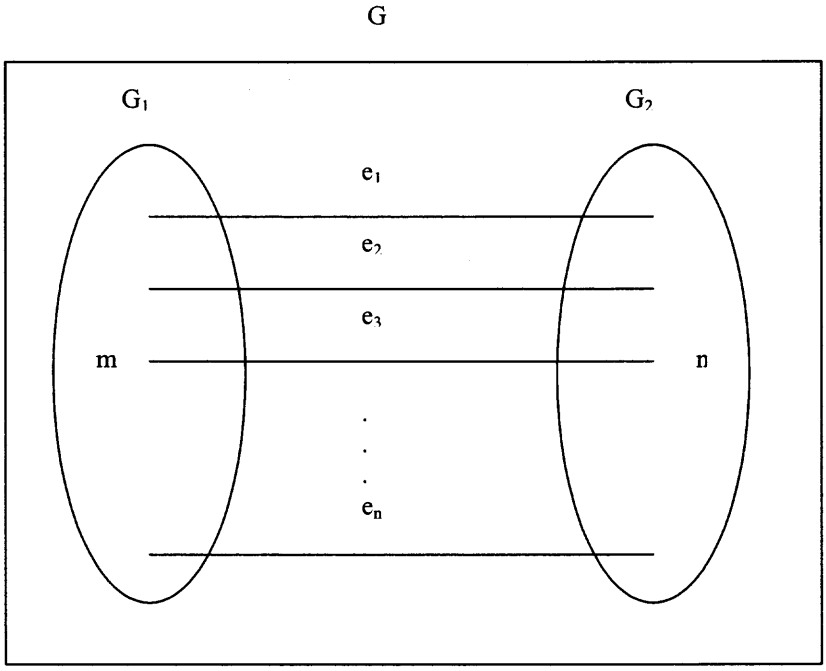

Figure 2.4 Given Network Topology: Edge-Cuts Explicitly Shown.

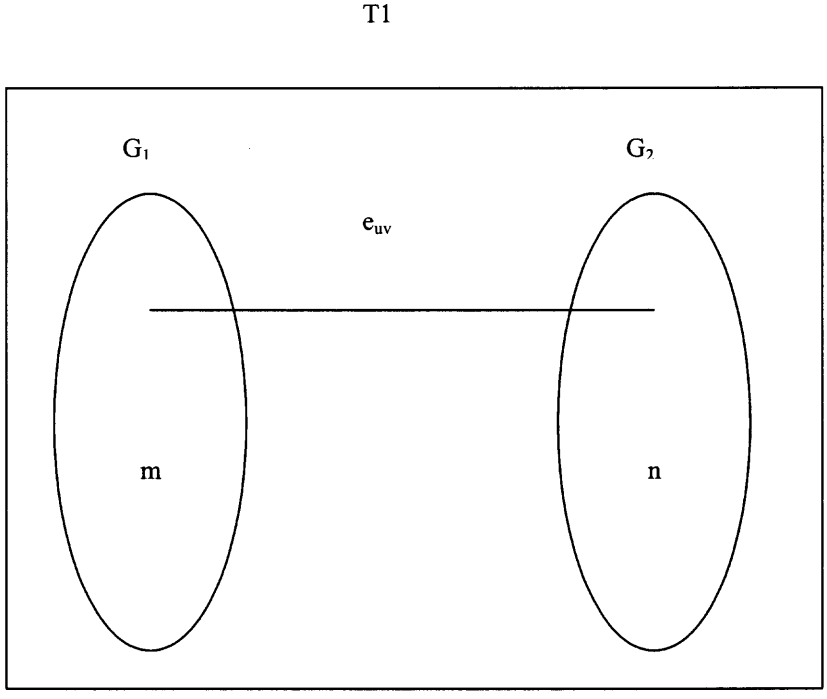

Figure 2.5 Spanning Tree of Given Topology.

Now, the vertices m and n are not directly connected as shown in Figure 2.5. As per step 2 of the algorithm, edge e_{uv} of $T1$ would surely be removed from G to form a set of intermediate graphs G'. This should leave all the edges other than e_{uv} from E_g in the set of intermediate graphs G' as shown in Figure 2.6. Note that G' should contain the edge e_3 connecting the nodes m and n.

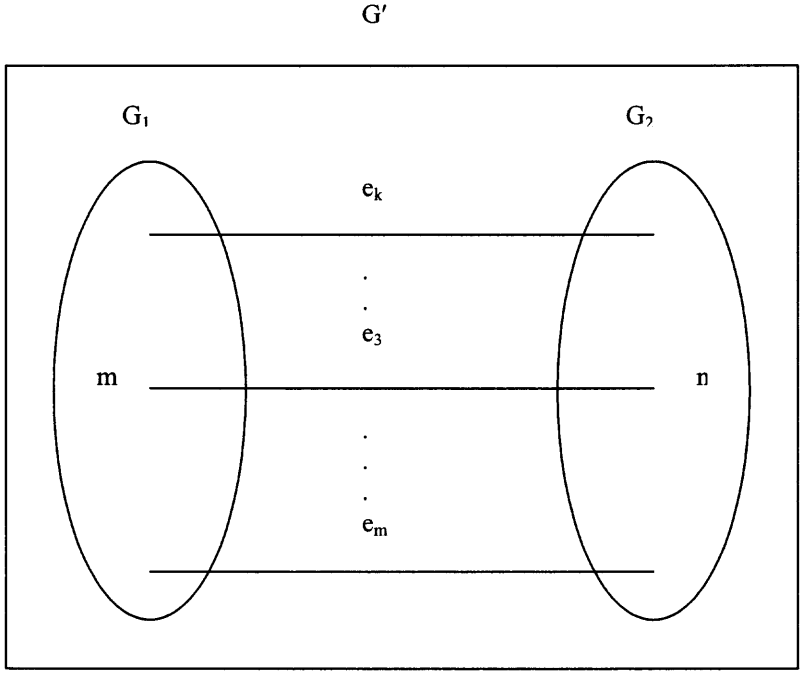

Figure 2.6 Intermediate Network Topology.

Now one of the spanning trees of the sub-graphs in G' should have a path from m to n, and hence the path should necessarily include an edge-cut, say e_p. Thus, in the final step of the algorithm when all the spanning trees are combined, the nodes m and n are connected by two edges (not necessarily a direct connection); e_{uv} from the spanning tree $T1$ and e_p. Note that e_{uv} and e_p are members of E_m, and E_m is a subset of E_g. This contradicts the basic assumption that $c(E_m)=1$. Thus, the reliable graph G_r that is found by the algorithm RGMST should have a minimum edge cut of two.

CHAPTER 3

CONCLUSION

A simple yet robust and efficient link state dissemination algorithm has been proposed. The proposed algorithm provides many properties that are required for QoS-based routing, which involves frequent dissemination of several dynamic parameters. In contrast to earlier works, algorithm RGMST involves less computation and more importantly makes the network robust. It makes sure that the number of edges in the minimum edge cut set of the given network topology is at least two. This provides extra reliability for the network and gives a lot of room for load balancing. A future work would be to demonstrate the algorithm's efficiency along with OSPF.

REFERENCES

[1] Shaikh, A. (1999). Efficient Dynamic Routing in Wide-Area Networks. Ph.D. Thesis, University of Michigan.

[2] Chen, S. & Nahrstedt, K. (1998). An overview of quality of service routing for next-generation high-speed network: problems and solutions (pp. 64-79). IEEE Network Magazine.

[3] Cheng, G. & Ansari, N. (2002). On Multiple Additively Constrained Path Selection (pp. 237-241). IEE Proceedings on Communications, 12, (5) vol. 149.

[4] Guerin, R. & Orda, A. (1997). QoS based routing in networks with inaccurate information: theory and algorithms (pp. 75-83). Proceedings of the INFOCOM'97.

[5] Korkma, T., Krunz, M. & Tragoudas, S. (2000). An efficient algorithm for finding a path subject to two additive constraints (pp. 318-327). Proceedings of the ACM SIGMETRICS'2000.

[6] Gang, L. & Ramakrishnan, K.G. (2001). A prune: an algorithm for finding k shortest paths subject to multiple constraints (pp. 743-749). Proceedings of IEEE INFOCOM'2001, vol. 2.

[7] Chen, S. & Nahrstedt, K. (1998). Distributed QoS routing with imprecise state information (pp. 614-621). Proceedings of 7th International Conference on Computer Communications and Networks.

[8] Wang, Z. & Crowcroft, J. (1996). Quality of Service routing for supporting multimedia applications (pp. 1228-1234). IEEE Journal on Selected Areas on Communications, 14, (7), vol. 14.

[9] Humblet, P.A. & Soloway, S.R. (1988/89). Topology broadcast algorithms (pp. 179-186). Computer Networks and ISDN Systems, vol. 16.

[10] Bellur, B. & Ogier, R.G. (1999). A reliable, efficient topology broadcast protocol for dynamic networks (pp. 178-186). In Proceedings of INFOCOM'99, vol.1.

[11] Zinin, A. & Shand, M. (2001). Flooding optimizations in link-state routing protocols. Networking Group, Internet Draft IETF, draft-ietf-ospf-isis-flood-opt-01.txt.

[12] Moy, J. Flooding over a subset topology (2001). IETF, draft-ietf-ospf-subset-flood-00.txt.

[13] Kleinrock, L. & Kamoun, F. (1997). Hierarchical routing for large networks; Performance evaluation and optimization (pp. 155-174). Computer Networks, vol. 1.

[14] Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (1990). Introduction to algorithms. Cambridge, Mass. MIT Press; NY: McGraw-Hill.

[15] Bellur, B. & Ogier, R.G. A Reliable. (1999). Efficient Topology Broadcast Protocol for Dynamic Networks. SRI International, Menlo Park, CA.

[16] Cain, B. (2000). Fast Link State Flooding. Nortel Networks, Billerica, MA.

[17] Choudhury, G., Maunder A.S, & Sapozhnikova V.D (2001). Faster Link-State IGP Convergence and Improved Network Scalability and Stability. AT&T Labs, Sanera Systems.

[18] Miyamura, T., Kurimoto, T. & Aoki, M. (2003). Enhancing the Network Scalability of Link-state Routing Protocols by Reducing their Flooding Overhead. NTT Network Service Systems Laboratories, NTT Corporation Tokyo, 180-8585 Japan.

[19] Ansari, N., Cheng, G. & Krishnan, R. N. (2003). Efficient and Reliable Link State Information Dissemination (in preparation).