ABSTRACT

A major link or node failure in a network can severely affect services offered by the network. In this paper, we address network performance, especially, in the event of a link failure in the presence of both unicast and multicast routing in wide-area datagram networks, such as the Internet. Since various services can exert different workloads (per connection/session), we have developed workloads to reflect both one-to-many and many-to-many multicast communications on top of multicast routing. Through network simulation, we quantify effects on how unicast services and multicast services are affected due to such a failure.

1. INTRODUCTION

In recent years, the Internet has seen surge in newer services besides the popular reliable unicast services such as telnet, ftp, and email. On one hand, there is increase in popularity of web browsing among reliable unicast services and the ability to provide Internet “voice call” for real-time oriented unreliable unicast services; on the other hand, multicasting services such as audio/video conferencing is now available [9]. Unlike unicast (point-to-point) services, multicasting services require “point-to-multipoint” or “multipoint-to-multipoint” transmission of packets. In order to support these services the network layer should be able to support multicasting, i.e., the network layer should support the delivery of a packet, having a single unique destination address, to multiple hosts with minimal duplication. There have been several works that have addressed multicasting (see, for example, [6, 7] and references therein).

As any network gains wide acceptability, an important issue to address is network survivability to address for a major link or a node failure. While typical congestion is attributed to increase in the traffic load, a node or a link failure adds another dimension to congestion since a large number of packet dropping can occur leading to significant retransmission at the transport layer (for reliable services) and rerouting in the network layer to move the traffic around the failure; in addition, certain connections can drop if application layer timer expires due to non-response within the acceptable time threshold. Thus, the transient behavior in the network and the adaptability of the network to a failure is a critical issue. While there has been considerable work on understanding the impact of a network failure on routing, design and performance for other types of networks such as circuit-switched networks, virtual circuit-based packet networks including ATM-based networks [2, 3, 8, 11, 12, 19, 21], there has been very limited work in this area for IP-datagram networks.

In this work, we consider the issue of network performance of a wide-area datagram network, such as the Internet, supporting multicast traffic along with unicast traffic, especially when subjected to a link failure. Failure in the network changes the network topology and hence the paths over which packets are routed. Routing in the network layer takes care of routing updates and actual routing of packets. Traditional unicast applications such as telnet, ftp, email, etc. operate over reliable transport protocol TCP; note that TCP is designed to compensate only for very minimal loss of packets (less than 1%, see [18]). Applications which use the multicast facilities of the network would typically run over unreliable transport protocol UDP as they are primarily delay sensitive than loss sensitive (although excessive loss is not desirable) and thus would not likely to have acknowledgements and retransmissions. In spite of this, we would like to minimize the loss of packets, due to a network failure, by rerouting the multicast packets after the failure to all destinations taking into account the change in the network topology.

In the past, work has been done towards studying and evaluating the working, functionality, interoperability and scalability of multicast routing protocols [7]. Some studies have also been carried out on performance of datagram networks with different unicast routing protocols and also when subjected to a failure [14, 15]. However, there has been little work on considering unicast and multicast routing together, and the interaction due to each other for the traffic workload generated, and specifically, when a network failure occurs. To conduct our study, we have developed MORN (Multicast Omnitraffic Routing Network) simulation tool that supports both unicast and multicast traffic routing and also considers workload behavior of the applications. In this paper we present multicasting routing MSPF (Multicast Shortest-Path First) and multicast workloads (telecast and conferencing) that we have implemented in the tool. We then present results on network performance. The rest of the paper is organized as follows: in section 2, we give an overview of routing and services that are related to our study environment. Our MORN tested development with multicast workloads, telecast and conference, as well as MSPF is described in section 3. Our simulation scenarios are discussed in section 4 while results and observations are presented in section 5.

* Research supported by National Science Foundation grant NCR-9506652, and by DARPA and Air Force Research Lab, Air Force Materiel Command, USAF, under agreement No. F30602-97-1-0257.

** Current affiliation: Advanced Network Technology Division, National Institute of Standards and Technology (NIST), Maryland.
2. ROUTING AND SERVICES OVERVIEW

We briefly overview pertinent routing and services in the Internet environment as they relate to the scope of this work. Since we are considering both multicasting and unicasting together, we visit each of the topics to discuss routing as well as workload for services.

2.1. Multicasting

There are two aspects of multicasting pertinent to our work: multicast routing and multicasting services. For multicast routing, there are two interrelated issues: the datagram forwarding mechanism and the routing protocol. The goal of the forwarding mechanism is to efficiently deliver packets to the multicast participants. To efficiently forward multicast datagrams across the network, a number of multicast forwarding techniques have been proposed in existing literature. These techniques can mainly be classified into the following major categories, namely, (1) "Simple-minded" Techniques, (2) source-based tree techniques, and (3) "shared-tree" techniques [7]. Flooding and (building a single) spanning tree for the entire network fall under the "simple minded" techniques. Since there can be several multicast sessions in different parts of the network and among different users, a specific group wanting to do a session for themselves can build a spanning tree for the group. An approach for this grouping idea is known as the source-based tree technique where the group-specific spanning tree is built from each potential source and done on a (source,group) pair basis. Reverse path forwarding, truncated reverse path broadcasting, and reverse path multicasting are the common source-based tree schemes. The third class of techniques is known as the "shared-tree" techniques. Unlike shortest-tree algorithms which build a source-based tree for each (source,group) pair, shared-tree algorithms construct a single delivery tree that is shared by all members of the group; thus, this is sort of a spanning tree for the group (instead of for the network), e.g. the core-base tree approach [7].

There are currently three well known multicast routing protocols for the Internet environment: Distance Vector Multicast Routing protocol (DVMRP) [20], Multicast Extensions to OSPF (MOSPF), and Protocol Independent Multicast (PIM) [6, 7]. The first two use source-based tree techniques for forwarding multicast traffic. In DVMRP, routing table updates are periodically exchanged among DVMRP neighbors using the distance-vector concept. The DVMRP routing table represents the shortest path spanning tree for each possible source and uses the reverse path broadcasting tree. DVMRP is being used on the MBone [7] and allows hosts to pass multicast traffic across routers not supporting multicasting using a technique called "tunneling". In MOSPF, the MOSPF-compatible routers maintain a current image of the network topology using the unicast OSPF link-state protocol. PIM, on the other hand, is independent of any particular unicasting routing protocol and uses both the source-based tree technique and the shared-tree technique for forwarding packets based on the size of the network and the geographical proximity of the group members.

Besides multicast routing, various multi-cast services or applications have been developed for Internet, and specifically, MBone that use multicast routing; e.g., multicast applications such as vic, vat, wb and sd developed at Lawrence Berkeley Laboratories [9]. In our study, we consider multicast services that use unreliable transport; however, these applications are assumed to have some "real-time" delivery requirement as well as group communications aspects such as one-to-many or many-to-many. Thus, workloads to generate multicast services are desirable.

A final issue which needs to be resolved for supporting multicast traffic is addressing. For IPv4, the Class D set of IP addresses are now used to uniquely address different multicast groups on the Internet. A host interested in receiving multicast traffic from particular groups uses the Internet Group Management Protocol (IGMP) [17] to inform the nearest router that it wants to join that particular group. The same protocol is used when the host wishes to leave the group. It may be noted that there is another issue important to multicasting, i.e. signalling for session setup/maintenance/teardown based on the soft-state approach, e.g. RSVP [22]; this is beyond the scope of the present paper.

2.2. Unicasting

In datagram networks, there are two main routing concepts[7, 16]: distance-vector with origination from the Bellman-Ford algorithm, and link-state with origination from the Dijkstra's shortest-path algorithm; these two concepts have been implemented, for example, in Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) protocol, respectively, used on the Internet. Unicast applications such as ftp, telnet, email use reliable transport protocol (TCP) which, in turn, uses network layer routing protocols for routing datagrams generated by applications. Newer applications that do not need reliable transport such as Internet "voice call" are also possible and are being implemented on the Internet and run on top of unreliable transport protocol UDP. In our work here, we restrict ourselves to unicasting applications that require reliable services such as ftp and telnet. Although one could generate traffic for network routing using Poisson packet arrival for ftp and telnet, it has recently been observed that Poisson modeling may not be appropriate [13]. Thus, using workload for applications such as ftp and telnet that has window-based flow control and acknowledgement-with-retransmission mechanism is more appropriate for generating workload (traffic) for connections for such applications. (For further details on TCP/IP stack, refer, for example, to [17].)

3. MORN SIMULATION TESTBED

Our goal here is to consider the routing aspects as well as to capture the traffic generated (workload) by various applications/services that need to use routing for transmission. Thus, we wanted the ability to generate traffic workload that mimics different applications. It has been suspected for sometime and have recently been documented that assuming services to have purely Poisson offered traffic is not realistic [13]. Thus, we wanted to consider workload for unicast applications such as ftp and telnet that capture their behavior (such as window-flow and acknowledgement-with-retransmission) to generate traffic. More importantly, our aim was to generate workload for emerging multicast services, yet address multicast routing also.

These requirements are addressed in our simulation tool MORN. We have built MORN on top of the publicly available...
unicast routing tool MaRS [1, 14, 15]. MaRS includes unicast workloads such as telnet and ftp, and a unicast routing component; for additional information about these components, see the references on MaRS [1, 14, 15]. MORN, in addition to the components of MaRS, includes a multicast routing component and two multicast workloads.

Multicasting services can be of two types: point-to-multipoint and multipoint-to-multipoint. Most existing multicast services are provided over UDP and has no notion of flow control or backoff in case of a congestion. On the other hand, most of them have the requirement to deliver at a given rate due to the "real-time" nature of these connections. Typically, the source is unaware of the recipients and just pumps data into the network and it is the task of the network (mainly the network layer) to deliver the data to the appropriate hosts. We have developed two "real-time" multicast workloads corresponding to the point-to-multipoint and multipoint-to-multipoint form of multicast traffic; we have also developed a multi-cast routing component, MSPF — these are described next and are implemented in the MORN simulation tool.

3.1. Telecast Workload

This workload component is used for simulating the point-to-multipoint or one-to-many kind of multicast traffic among a group. There is only one source node and the rest of the members of the group act as sinks. It simulates a steady stream of packets originating from the source based on a given inter-packet arrival time distribution. Both the source and the sinks are unaware of each other's identity. All the packets generated by the source have a connection number and the source address based on which they are routed to the appropriate group members. The packet transfer is unreliable, i.e., the receivers do not send any acknowledgements to the source upon receipt of packets nor does the source keep track of how many of the packets sent were successfully delivered to the sinks. This workload tries to mimic a "real-time" effect.

3.2. Conference Workload

The conference workload simulates the many-to-many or multipoint-to-multipoint type of multicast traffic. Each group consists of members who take turns acting as the source for the group while the rest of the members act as sinks. In this case too, the host which is acting as the source for the group generates a steady stream of packets per a specified inter-packet arrival time distribution and the network forwards these packets to all group members based on the connection number and source address. The transfer in this case too is unreliable. The main difference between the telecast and the conference traffic is that for the conference traffic all members are aware of the rest of the group members. We call every instance when a host takes a turn to transmit packets a session. The number of packets the host can send in that session is determined by an average value and a particular distribution. Once the host has completed transmitting all of its allocated number of packets, the conference group randomly chooses another member of the group to start acting as the source for the group. In this way various group members randomly act as sources for time periods determined by the average number of packets each wants to transmit.

This allows us to simulate various conference scenarios such as one particular host dominating the conference (e.g., group meeting where a manager talks the most with some inputs from group members) or the case where all hosts have equal probability of inputs and take turns (e.g., friends get together after a couple of months and share their experiences).

Both telecast traffic and conference traffic are independent of each other and the same node can be a part of different telecast and conference connections simultaneously. This can give rise to interesting multicast traffic flows in the network which are difficult to study analytically but can be studied using simulations.

3.3. Multicast Routing Component

We have defined and implemented a multicast routing component called MSPF (Multicast Shortest Path First); this is developed by modifying the SPF(Shortest-Path First) routing component of MaRS. SPF is link-state based and uses the Dijkstra's shortest-path algorithm where the nodes compute and broadcast the costs of its outgoing links in a periodic manner and also when a failure/repair of a link takes place [15]. The new MSPF component generates source-based trees for routing multicast traffic.

The multicast routing component has different routing tables for unicast and multicast traffic. Based on the global topology table, each component creates its own adjacency matrix which gives the cost to go from node i to node j for all node-pairs (i, j); the cost is infinite if node j is not directly reachable from node i. Based on this adjacency matrix, the MSPF component creates a unicast routing table which is implemented using the Floyd-Warshall algorithm [4]. The routing table gives the next hop for all node pairs as sources and destinations. The multicast routing component extracts the next hops for all destinations from the node it is connected to and stores them (the link towards the next hop) in the unicast routing table.

The multicast routing table is a three-dimensional table which stores the next hop for all sources for all multicast groups. Given the adjacency matrix and the global connection table, the MSPF component runs the Floyd-Warshall algorithm on a part of the network, i.e., on the nodes which are members of a particular multicast group, and generates the next-hop table for that group. If any destination is not directly reachable from a particular source then it uses the next-hop stored in the unicast routing table generated previously. The MSPF component then parses the next-hop table to determine which nodes (routers) are downstream to it for each source and stores the links connecting to those nodes (routers) in the multicast routing table. It repeats the above for all multicast groups and thus knows all the downstream nodes (routers) for all (source,group) pairs. The link costs are updated in a periodic fashion to recompute the routing table based on a given update cycle; the updates are also transmitted when a failure/repair occurs.

In order to support multicast traffic in addition to unicasting, we have implemented the ability to distinguish packet type in the node component in MORN. Whenever the node receives a multicast packet, it first checks to see if it is part of the group for that connection and pass the packet to the appropriate sink component, and it always looks up the multicast routing table and
forwards the packet to the downstream nodes (routers) if any for that (source,group) pair.

Recall that, in IP networks, each multicast connection is assigned a unique IPv4 Class D address. In order to emulate a separate IP multicast address (Class D Address) for each connection, we have given each multicast group a unique connection number (identifier). The connection profile is read by the simulator when it is started, and the simulator stores information about various connections, their types (telecast or conference) and the nodes which are members of the particular group. For telecast connections the first node in the list is the source, whereas for conference connections the first node starts the conference (refer to Table 1).

### 4. SIMULATION SCENARIOS

To study the impact of multicast traffic and physical failures on wide-area networks, we have considered an eleven node network spanning continental US interconnected by eighteen links. The topology of the test network is shown in Figure 1. There are two basic types of packets in the network: packets due to traffic workload and packets for routing updates. Workload packets are processed in a processing time of 1 msec. Routing update packets have priority over workload packets in the outgoing queue at each node. If the buffer space is available at the node, then workload packets are transmitted after a certain processing time while routing packets are immediately transmitted; on the other hand, if the buffer space is full, workload packets are dropped to make room for routing packets. The processing time for MSPF routing packets is 6 msec and the size of the link-state packet is $16 + 8e$ bytes, where $e$ is the number of neighbors of the node.

The network has a total of 27 connections consisting of both unicast and multicast services (Table 1) with workload configuration of 50% multicast load and 50% unicast load. Unicast workload consists of telnet and ftp connections and the multicast traffic consists of both telecast and conference types of connections. Both unicast workloads are reliably transmitted and have a window size of eight packets as described in [15]. The traffic is defined in terms of source-sink pairs between specific nodes. Multicast workloads define traffic between more than two nodes, e.g. consider the telecast connection (3,5,9,7,11) (Table 1) where node 3 is the source node and the rest are sink nodes for this group communication; thus, there are communications for the node pairs 3–5, 3–9, 3–7, 3–11 for this group. Due to multicast routing, however, a copy of a packet is originally created for this group at node 3 and is routed along the multicast tree.

Often, all links in a wide-area network are not of the same bandwidth and and, in fact, differ in their bandwidth and propagation delay times. We chose to have heterogeneous links in the test network to reflect the reality of a heterogeneous network through different link bandwidths.

To determine link bandwidths, we use a simple network sizing procedure. We first consider the traffic (average packet arrival rate) between all nodes based on the list of unicast and multicast connections defined with their workloads. Using this information, we estimate average flows on various links assuming that the traffic would take the least hop path (hop-based link cost). Once we have estimated the flows on different links we were able to calculate the maximum bandwidth required on each link based on an acceptable link utilization value and then, using modularization, we separated the links in the test network into three categories/types each having a different bandwidth. Referring to Figure 1, all links labeled ‘1’ have a bandwidth of 128Kbps (2 DSOS), all links labeled ‘2’ have a capacity of 320Kbps (5 DSOS) and all links labeled ‘3’ have a link capacity of 768Kbps (12 DSOS or half T1).

Most of the simulations have been carried out with all nodes having infinite buffer capacity. In practical networks, nodes do not have infinite buffer capacity and they drop packets if there is no buffer space left to accommodate any arriving packets. Unicast traffic running on top of a reliable transport (TCP) can compensate for this loss by retransmitting dropped packets but for multicast traffic considered here which runs over unreliable transport, this packet loss could have an adverse effect. In order to study the effect of packet loss of multicast traffic, we have also performed limited simulations by subjecting the network to the same conditions as described above but limiting the buffer size at specific nodes.

Our simulation runs were done for 600 seconds; of which, the first 60 seconds were used for the warm up time. To study a link failure, we have simulated to fail a link at 300 seconds with full recovery of the link at 400 seconds. This allows us to observe how the network responds to the failure and also how the network cools down after the repair. Multiple replications were done and the graphs discussed in the next section depicts the average; the 95% confidence-interval is found to be around 5% of the average.

### 5. RESULTS AND OBSERVATIONS

We summarize only some key results (due to the paper length limitation), and only for a specific set of scenarios. For the results reported here, we consider the effect due to failure of the link 9–7. This failure has the interesting property that at least a connection from each type of workload is affected: specifically, (9,5,7,4) for conference, (3,5,9,7,11) for telecast, (7,9) for ftp, and (3,7) for telnet. Obviously, the actual ones affected depend on the specific routing scheme and the link cost function used.

In any case, consider the conference workload (9,5,7,4); it is likely, that the transmission from source 9 to 7 will be on link 9–7 and from 9 to 5 will be on link 9–5 and from 9 to 4 could be on either path 9–7–4 or 9–5–4; after failure of link 9–7, this multicast connection will use the shortest spanning tree 9–5–4–7; this is if 9 is the source during the conversation. Since this is a conference traffic, other nodes also take turn to be the source, in which case the flow is different and the appropriate shortest path tree is in use. At the same time, the unicast ftp workload for connection (7,9) which would use link 9–7 before failure will use 9–5–4–7 after failure.

We first discuss our results for the infinite buffer case. The infinite buffer case allows us to see the impact on unicast services while there are also multicast services provided in the network. An interesting measure to look at is the delay observed for unicast traffic for acknowledged packets while the network also carries unreliable multicast traffic. We provide two measures for delay: instantaneous delay refers to delay computed in a 500 milisec time
period at a time while the average delay refers to computation of the moving-average delay over 60 seconds cycle.

Figure 2 shows the variation of the instantaneous network delay when the network is subjected to the failure of the link 9–7. In this case, the link cost used was hop-based. We see that as soon as the link fails, the instantaneous link delay jumps immediately and stays high throughout the duration of the failure; the increase in the delay is caused by the lack of bandwidth available in the network as well as the unreliable multicast traffic still maintaining its steady arrival rate. It is interesting to note that this is another manifestation of the “bully”ing effect of “real-time” unreliable transport based service over reliable transport service discussed, for example, in [10] in the context of congestion control; specifically, the bullying effect is much more pronounced in the event of a major failure compared to typical congestion. It also shows an immediate recovery following the restoration of the failed link in the network. It may be noted that the delay is not steady however; this can be attributed to the effect of window-based flow control. The average network delay shows a gradual rise and fall due to the failure since the moving-average over 60 seconds is shown in this case. The delay experienced by the unicast connection (9,7) displays behavior similar to the network delay in both the instantaneous and average cases. We can see from Figure 2 that the instantaneous delay experiences similar drastic jumps upon failure and recovery whereas the average delay has a gradual rise and fall. We observe that the delay experienced by the connection has a much higher rise compared to the network delay. The delay experienced by the traffic between nodes 7 and 9 is higher than the network delay partly due to the fact that the traffic is rerouted over link 4–5 which has a lower capacity and hence a higher delay to transmit packets. Figure 3 shows the queue size of packets at node 5 for link 4–5 and at node 4 for link 4–7 for 100 seconds before and after the failure. We see that since link 4–5 has lower capacity (which was dimensioned for the normal network load case discussed in the previous section), it cannot handle all the packets rerouted due to the failure and hence more packets are queued up at node 5.

Next we consider the effect due to different routing update period – this time for delay-based link costs. The routing component at each node periodically broadcasts its local topology table to all the nodes in the network to exchange information about changes in the network topology from the previous broadcast. If the frequency of these updates is very high, then the routing packets could add a significant amount of traffic on the network. On the other hand, if the frequency of these updates is very low, then the nodes do not get information about topology changes soon enough for them to change their routing tables according to the latest topology. Unlike networks that have only unicast traffic, we also have multicast traffic and, especially, conference workload; this type of workload adds a new dimension since for a given connection, different sources can take turn talking which then affect the direction of the flow of packets. Consequently, this aspect of taking turn to speak gives a new dimension to the link load at different instant of time, and since the routing update is affected by load, it is conceivable that more churning can happen if the updates are done too quickly. We varied the routing update period from about every ten seconds to about every second and the results are plotted in Figure 4. The rather high fluctuation for shorter update periods can also be attributed to the the churning we just described (among other factors such as window-flow affect, retransmission and so on), although this does not appear to be as noticeable in the case of hop-based link cost.

We have implemented MSPF in such a way that various links costs such as hop count, delay and link utilization can be independently considered. Figure 5 shows the change in the average network behavior when the link cost function is varied. We observe that the utilization-based routing has the worst transient response and maximum average delay, whereas delay-based and hop-based routing have comparable performance although delay-based routing triggers more fluctuation on average delay. We believe the reason for hop-based and delay-based routing delivering nearly similar steady-state and transient response is due to the artifact that the link bandwidths were dimensioned based on the network flow with hop-based routing! This could result in the packets following the similar paths for both hop-based and delay-based routing.

We now discuss the impact due to different distributions for inter-packet arrival time. Real-time traffic like digitally encoded voice and video tend to have constant sampling rates and hence the inter-packet arrival time would be deterministic rather than exponential. As a part of our study we varied the inter-packet distribution times for all multicast traffic sources from exponential to geometric and deterministic keeping all unicast traffic as before. Figure 6 shows the variation in the average network delay due to this change in the inter-packet time distribution. Note the uniform increase in delay for non-exponential distributions compared to the exponential case.

Finally, we discuss briefly results for the finite node buffer case. The finite buffer case allows us to observe the impact on multicast services since packet loss occurs due to finite buffer; recall that, while in the case of unicast services, reliable transfer is performed, i.e., if there is any packet loss, retransmission is done, this is not so for unreliable multicast services. Instead of arbitrarily setting finite buffer sizes at various nodes throughout the network, we decided to make use of what we have learned from studying the infinite buffer case discussed earlier. From runs done for the infinite buffer case, we were able to observe the maximum amount of buffer capacities used at each node; based on this information, we have configured the network so that the nodes directly (not all nodes) affected by the failure are set to 80% of the maximum buffer size observed for the infinite buffer case. For example, node 7 and node 9 were observed to require buffer sizes of 2.7 and 9.9 megabytes respectively when link 9–7 was subjected to failure in the infinite buffer run case; thus, for the finite buffer run case, the node buffers were set to 80% of the respective amounts for these two nodes while all other nodes have infinite capacity. Similarly, in another simulation, the nodes directly affected by the failure had infinite buffer capacity but the nodes over which the traffic was rerouted (for example, node 4 and node 5 are the nodes over which traffic is rerouted when link 9–7 is subjected to failure) had finite buffer sizes fixed to 80% of the maximum buffer size observed (for the infinite buffer case run). For both simulations we have also obtained the overall network delay experienced by a unicast connection, and more
importantly, the packet loss for multicast services. We observed that during the failure the unicast traffic experiences a higher delay due to finite buffer sizes but the increase is not very significant compared to the infinite buffer case. We observed that when the nodes directly affected by failure have finite buffers, of the total number of packets dropped about 5 to 10% were for multicast. On the other hand, when the nodes on the alternate path had finite buffers the amount of multicast traffic dropped was about 10 to 15%. Table 2 shows the percentage loss under normal case as well as under failure on average for both cases together. We measured the number of packets dropped for a specific multicast connection and found that to be as high as 40% in both the cases.

6. DISCUSSION

In this paper, we have considered both unicast and multicast traffic routing. Specifically, we have studied interaction of reliable unicast services with unreliable multicast services in the event of a network outage. Towards this end, we have developed the MORN simulation tool where multicast routing component (MSPF) as well as multicast workloads (telecast and conference) are implemented alongside unicast components. Our attempt was to quantify network performance for various scenarios and, especially, when there is a link failure. The impact on average network delay is found to be more stable for the hop-based link cost compared to delay-based or utilization-based link costs. An interesting observation is the heavy bullying effect of multicast services under a network failure, and also how a specific connection may be affected due to link sized for normal network condition and then faced with additional load due to a link failure. For the finite buffer case, we found the loss for multicast traffic to be fairly significant. Obviously, more studies are needed to understand a variety of other interactions. Nevertheless, this study quantifies results that have not been done/observed previously. It is clear that having infinite buffer is not the answer to reduce network congestion to address for a major link failure. Additionally, to reduce loss of packets for multicast services or to reduce delay for unicast services under a failure, having priority queueing at the nodes is desirable; but, more importantly, the network has to be dimensioned appropriately. Similar to the work done for other types of networks [5, 11, 12], proper network dimensioning methods need to be developed to address for multicast based services that have quality-of-service requirements, and, especially, to address for a network outage.

Acknowledgement

We like to thank the developers of MaRS at the University of Maryland for making MaRS source code publicly available. This has helped us considerably and has saved our development time significantly in creating the MORN simulator.

References


<table>
<thead>
<tr>
<th>Workload Type</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>conference</td>
<td>(9,5,7,4) (1,8,6,10) (4,2,10,6)</td>
</tr>
<tr>
<td>telecast</td>
<td>(2,(11,5,3,1,8)) (3,(5,9,7,11))</td>
</tr>
<tr>
<td>ftp</td>
<td>(2,9) (3,6) (3,8) (4,8) (5,8)</td>
</tr>
<tr>
<td>telnet</td>
<td>(2,7) (3,4) (3,7) (3,10) (5,6)</td>
</tr>
</tbody>
</table>

Table 1: Connections for all services used in the study

<table>
<thead>
<tr>
<th>Packet loss under NO failure</th>
<th>Packet loss under failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>telecast connections</td>
<td>15% 40%</td>
</tr>
<tr>
<td>conference connections</td>
<td>10% 35%</td>
</tr>
<tr>
<td>Multicast packet loss</td>
<td>2% 15%</td>
</tr>
</tbody>
</table>

Table 2: Multicast packet loss in the finite buffer case

Figure 1: Topology of the 11-node test network

Figure 2: Instantaneous and Average Delay for unicast traffic in the presence of multicast traffic

Figure 3: Queue Size Change at two different nodes

Figure 4: Network Delay for different routing update periods (delay-based link cost)

Figure 5: Network Delay for different link cost functions

Figure 6: Network Delay for different inter-arrival packet distributions