Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.1074 – 1081

RESEARCH ARTICLE

FAULT TOLERANT REAL TIME COMMUNICATION IN MULTIHOP NETWORKS USING SEGMENTED BACKUP

K.Shirisha[1], Ch. Radha Krishna[2], Gousiya Begum[3], S.Vijaya Lakshmi[4]

[1,3,4] Computer Science and Engineering Department, MGIT, Hyderabad
[2]Technical Manager, Verizon India, Hyderabad

shirisha.krishna@gmail.com
radhakrishnach@gmail.com
gousiyabegum@gmail.com
vijayasattar@gmail.com

***Abstract* —Many distributed real-time applications demand guarantees on the message delivery latency and the recovery delay from component failures in multi hop networks. So special schemes have been proposed to provide timely recovery for real-time communications in multihop networks. These schemes reserve additional network resources (spare resources) a priori along a backup channel that is disjoint with the primary. Upon a failure in the primary channel, its backup is activated, making the real-time connection dependable.**

In this paper we propose a new method of providing backups called segmented backups, in which backup paths are provided for partial segments of the primary path rather than for its entire length as is done in the earlier methods.

In order to provide backups we use three protocols: 1) routing protocol, to determine the primary as well as the backup paths; 2) reservation protocol, to reserve resources efficiently along these paths; and 3) recovery protocol, to recover from any failure. We present the recovery protocol as a simple extension to the Internet Engineering Task Force (IETF)-recommended resource reservation protocol, RSVP.

Using above protocols we can offer: 1) improved network resource utilization; 2) higher average call acceptance rate; 3) better quality-of-service guarantees on propagation delays and failure-recovery times; and 4) increased flexibility to control the level of fault tolerance of each connection separately.

Keywords: Segmented backups, message delivery latency, recovery delay, backup channel, dependable connection, multihop network, primary channel, quality-of-service (QoS), real-time communication, resource reservation protocol (RSVP), routing protocol, recovery protocol

I. INTRODUCTION

The Advent of high-speed networking has introduced opportunities for new applications such as real-time distributed computation, remote control systems, digital continuous media (audio and motion video), video conferencing, medical imaging, and scientific visualization. Such distributed real-time applications demand quality-of-service (QoS) guarantees on timeliness of message delivery and failure-recovery delay [1]. These guarantees are agreed upon before setting up the communication channel and must be met even in the case of bursty network traffic, hardware failure (router and switch crashes, physical cable cuts, etc.), or software bugs. Most real-time communication schemes for multihop networks share three common properties: *QoS-contracted, connection-oriented,* and *reservation-based*. A contract between a client and the network is established before messages are actually transferred. To this end, the client must first specify its input traffic behavior and required QoS. Then, the network computes the resource needs (e.g., link and CPU bandwidths, and buffer space) from this information, selects a path, and reserves necessary resources along the path. (If there are not enough resources to meet the QoS requirement, the client's request is rejected.) The client's messages are transported only via the selected path with the resources reserved, and this virtual circuit is often called a *real-time channel*. While this reservation-based approach has been successful in providing "hard" guarantees on timeliness QoS, it causes a serious difficulty in achieving fault tolerance(because the approach relies on static routing) [3]. Applications using traditional best effort datagram services like IP experience varying delays due to varying queue sizes and packet drops at the routers. To ensure bounded message delays for real-time applications, special schemes such as resource reservation protocol (RSVP)[2] have been proposed.

In this paper, we propose and evaluate a new scheme to construct backup paths, which we call *segmented backups*. A segmented backup comprises multiple backup paths, each spanning a contiguous portion of the primary path. This is unlike an end-to-end backup where the backup spans the entire length of the primary path..

The spare resources reserved lower the maximum throughput of the system as the resources reserved for backup channels are used only during component failures. So, minimizing spare resources is an important metric while evaluating different schemes.

In RSVP, resources (such as link bandwidths and router buffers) are reserved a priori along the message transmission path from the source to the destination for the duration of a session. While RSVP can provide QoS guarantees on the packet transmission latency, it lacks quick failure-recovery mechanisms. In RSVP, when a channel fails, a new one is established. A successful recovery cannot be guaranteed as sufficient resources might be lacking at recovery time. Further, the channel re-establishment time could take a long time, especially when there is contention for resources among disrupted channels. Given that some of these applications (such as commercial video on demand) last for a long time, ranging from several minutes to hours, such failures might not be uncommon during a single session.

II. LITERATURE SURVEY

Segmented backups have numerous advantages over end-to-end backups.

• *Higher call acceptance rate*. This is due to primary paths that have a segmented backup but no end-to-end backup.

• *Improved network resource utilization*. This is because segmented backups are typically shorter than end-to-end backups and need less spare resources. Moreover, shorter backup paths lead to more efficient resource aggregation through *backup multiplexing* [3], [4], [5].

• *Better QoS guarantees*. A segmented backup can comprise multiple backups, each of which spans a part of the primary path rather than its full length. This allows for faster failure recovery and finer control of fault tolerance for long primary paths over components with varying reliability.

Resource Reservation Protocol

The *Resource Reservation Protocol (RSVP)* is a network-control protocol that enables Internet applications to obtain differing qualities of service (QoS) for their data flows [6]. Such a capability recognizes that different applications have different network performance requirements. RSVP was intended to provide IP networks with the capability to support the divergent performance requirements of differing application types. It is important to note that RSVP is not a routing protocol. RSVP works in conjunction with routing protocols and installs the equivalent of dynamic access lists along the routes that routing protocols calculate. Thus, implementing RSVP in an existing network does not require migration to a new routing protocol.

RSVP is used to specify the QoS by both hosts and routers. Hosts use RSVP to request a QoS level from the network on behalf of an application data stream. Routers use RSVP to deliver QoS requests to other routers along the path(s) of the data stream. In doing so, RSVP maintains the router and host state to provide the requested service.

RSVP Reservation Style

Reservation style refers to a set of control options that specify a number of supported parameters. RSVP supports two major classes of reservation: *distinct reservations* and *shared reservations*. Distinct reservations install a flow for each relevant sender in each session. A shared reservation is used by a set of senders that are known not to interfere with each other.

General RSVP Protocol Operation

The RSVP resource-reservation process initiation begins when an RSVP daemon consults the local routing protocol(s) to obtain routes. A host sends IGMP messages to join a multicast group and RSVP messages to reserve resources along the delivery path(s) from that group. Each router that is capable of participating in resource reservation passes incoming data packets to a packet classifier and then queues them as necessary in a packet scheduler. The RSVP packet classifier determines the route and QoS class for each packet. The RSVP scheduler allocates resources for transmission on the particular data link layer medium used by each interface. If the data link layer medium has its own QoS management capability, the packet scheduler is responsible for negotiation with the data link layer to obtain the QoS requested by RSVP.

The scheduler itself allocates packet-transmission capacity on a QoS-passive medium, such as a leased line, and also can allocate other system resources, such as CPU time or buffers. A QoS request, typically originating in a receiver host application, is passed to the local RSVP implementation as an RSVP daemon [6].

The RSVP protocol then is used to pass the request to all the nodes (routers and hosts) along the reverse data path(s) to the data source(s). At each node, the RSVP program applies a local decision procedure called admission control to determine whether it can supply the requested QoS. If admission control succeeds, the RSVP program sets the parameters of the packet classifier and scheduler to obtain the desired QoS. If admission control fails at any node, the RSVP program returns an error indication to the application that originated the request.

RSVP Messages

RSVP supports four basic message types: reservation-request messages, path messages, error and confirmation messages, and teardown messages. Each of these is described briefly in the sections that follow.

Primary backup segmented scheme

To establish dependable connections, earlier schemes have used end-to-end backups, i.e., backups that run from the source to the destination without sharing any components with the primary path (other than the source and the destination nodes themselves). In our new approach of *segmented backups*, we find backups for the primary path, taken in parts. The primary path is viewed as made up of smaller contiguous paths, which we call *primary segments*. We find a backup path for *each* segment, which we call *backup segment,* independently. By *segmented backup* we refer to these backup segments taken together.

We illustrate these terms in Fig. 1 where a primary channel with intermediate nodes N1 – N8 and links 1–9 is shown. The backup links are *A–K*. The primary channel has three primary segments, each with its own backup segment. The primary segments span links 1–3, 3–6, and 6–9, while their corresponding backup segments span links *A–C*, *D–G*, and *H–K*, respectively. These three backup segments together constitute the segmented backup for this primary path. Note that successive primary segments of a primary path overlap on at least one link.

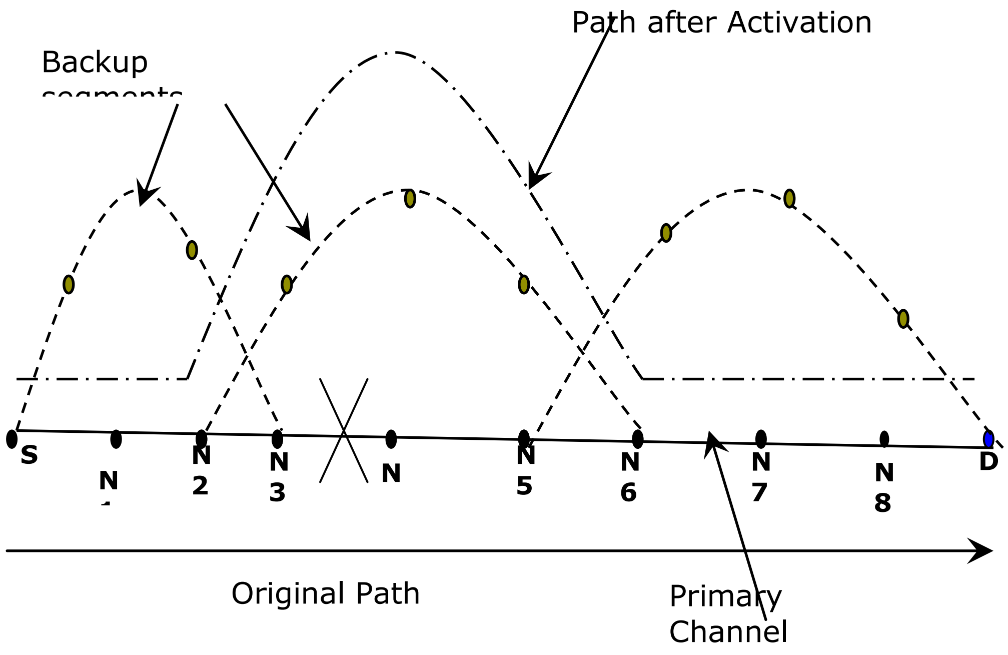


Fig1. Illustration of a primary channel with a segmented backup

III. SYSTEM DESIGN

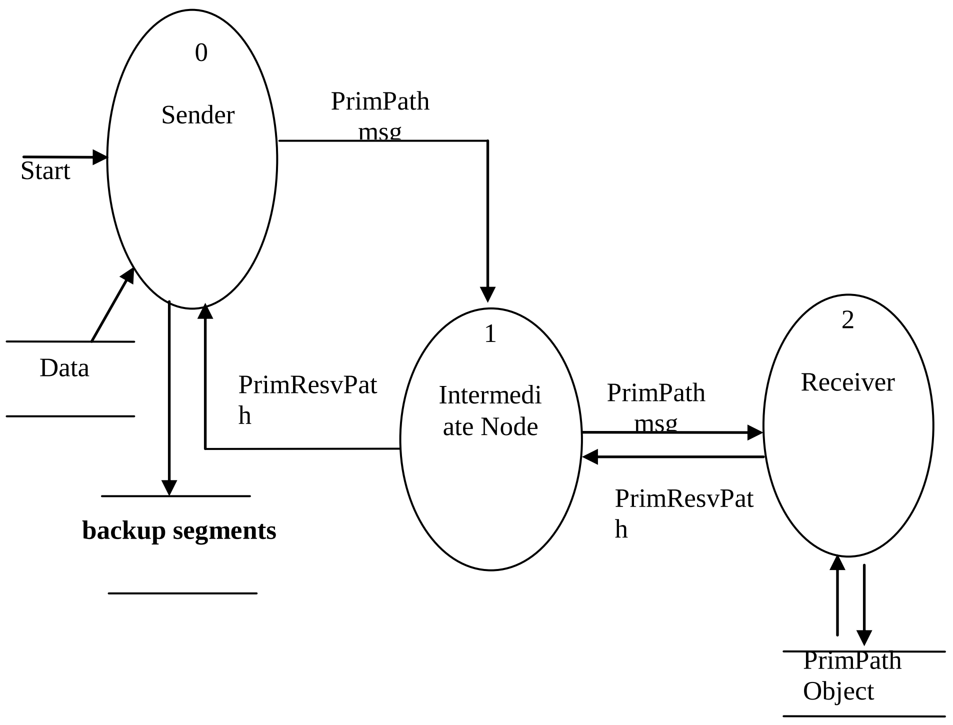


Fig2. Data Flow Diagram for Setup Message

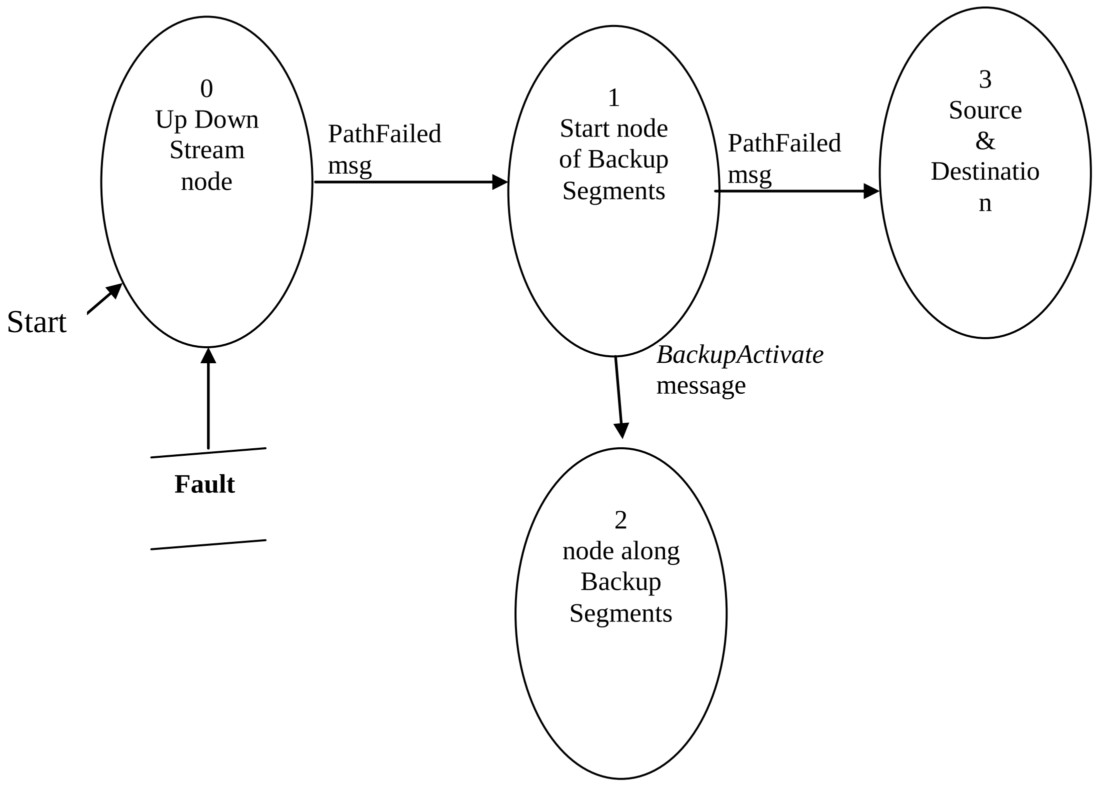


Fig3. Data Flow Diagram for Failure Detection and Backup activation

Design of Primary Segmented Backup Scheme

We discuss three protocols: 1) *routing protocol*, to determine the primary as well as the backup paths; 2) *reservation protocol*, to reserve resources efficiently along these paths; and 3) *recovery protocol*, to recover from any failure. We present the recovery protocol as a simple extension to the Internet Engineering Task Force (IETF)-recommended resource reservation protocol, RSVP, and show that its complexity is comparable to that of RSVP.

***Design of the Routing Protocol**:* The routing protocol determines the primary and segmented backup paths using the *Min_SegBak* algorithm. To compute the backup path, our algorithm assumes that every router has the global knowledge of the network topology. To obtain information about all routers and links in the network, our routing protocol can use any variant of link state protocol such as the Open Shortest Path First (OSPF) protocol [7] (a popular intra-domain routing protocol)[. With this global knowledge of the network, the primary path can be computed as the least weight path using Dijkstra's shortest path algorithm, while the backup path could be decided later by running the *Min_SegBak* algorithm at the source node.

Design of the Reservation Protocol: We propose simple extensions to RSVP to design the reservation protocol. The protocol primarily consists of the following three types of messages:

1) *Setup messages*. The source router initiates the reservation by sending a *Path* message to the destination along a route selected by the routing protocol. On receiving the message, the destination router sends a *Resv* message back to source in the reverse direction using the path state set up by the *Path* message. Resources are reserved at the routers along the path by the *Resv* message. We illustrate the connection setup process in Fig.2. The reservation is initiated by the source with a *PrimPath* message to the destination as shown in Fig. 2. The destination responds with a *PrimResv* message back to the source. The *PrimResv* message contains an object called *PrimaryPath*, which is filled up with the routers along the primary path as the message traverses it, as shown in Fig. This *PrimaryPath* object is used by the source to compute the primary and backup segments using *Min_SegBak* algorithm

2) *Maintenance messages*. Upon failure during the set up of either primary path or any backup segment, error messages like *PrimPathErr, PrimResvErr, BackSegPathErr*, and *BackSegResvErr* are generated. These messages also initiate the corresponding reservation teardown messages to free any resource reservations made prior to the routing failure.

3) *Teardown messages*. The messages *PrimPathTeari, Prim- ResvTear, BackSegPathTear*, and BackSegResvTear are used to free the resources reserved. They are propagated along the primary and backup paths.

Design of the Recovery Protocol: Failure recovery comprises three phases: detecting the fault, reporting the failure, and activating the backup. In our model, we assume that when a link fails, its end nodes can detect the failure, and that when a node fails, all its neighbors can detect the failure. The nodes that detected the fault report it to the start and end nodes of the corresponding backup segment, which then activate the backup to recover from the failure. We now introduce a new type of message, called the *failure-recovery message*, to report failures and to activate the backup.

IV. IMPLEMENTATION

Backup Route Selection

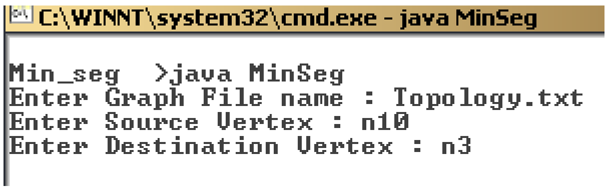
It is usually desirable to select a segmented backup with minimum *backup delay increment* (i.e., the difference between delays along the primary and backup paths) or minimum cost. Here we deal with construction of segmented backups that offer better QoS guarantees on reliability. However, were strict our discussion here to construction of segmented backups with minimum cost, where cost can be a function of path delay or path length or path resource reservation requirement. Though our goal is to design algorithms to select minimum cost segmented backups that yield a higher call acceptance rate. The problem of optimal routing of backups is NP-hard as it subsumes the following problem which is known to be NP-hard: *Is there a feasible set of channel paths such that the sum of traffic flows at each link is smaller than the link capability, when traffic demands are given?* The complexity of the problem increases greatly if one considers backup multiplexing. So, we resort to heuristics to select least cost backups to each individual primary path, ignoring the additional savings offered by multiplexing. A simple way to find a minimum- cost end-to-end backup for a primary path in a network graph G is to use some shortest path search algorithm such as Dijkstra's over a graph G' obtained from G by removing the components along the primary path. The problem of selecting minimum-cost segmented backups is, however, more difficult as typically there are a larger number of segmented backups than end-to-end backups and we have to find intermediate nodes where the backup segments meet the primary. We provide an algorithm called *Min_SegBak* that solves this problem.

Our algorithm to find the shortest segmented backup for *P* in *G* consists of three steps.

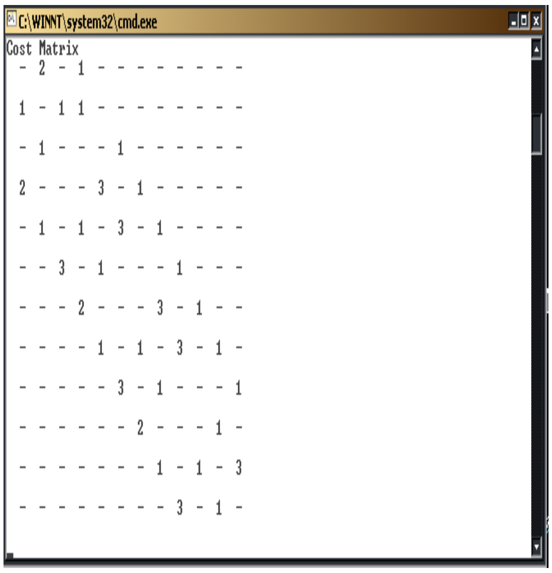
In step 1, we generate a modified graph *G'* (V,E) from *G(V,E')* on the same set of vertices.

In the step 2, we find the shortest path between *S* and *D* in graph and use it in step3 to obtain the segments of the minimum-cost segmented backup in *G*.

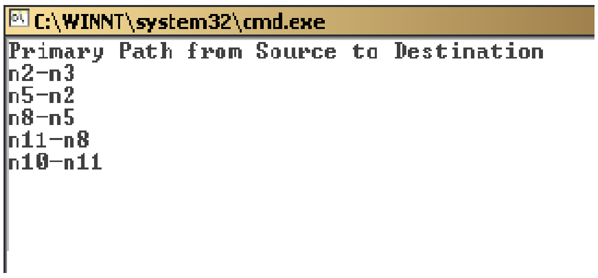
V. EXPERMENTAL RESULTS AND ANALYSIS



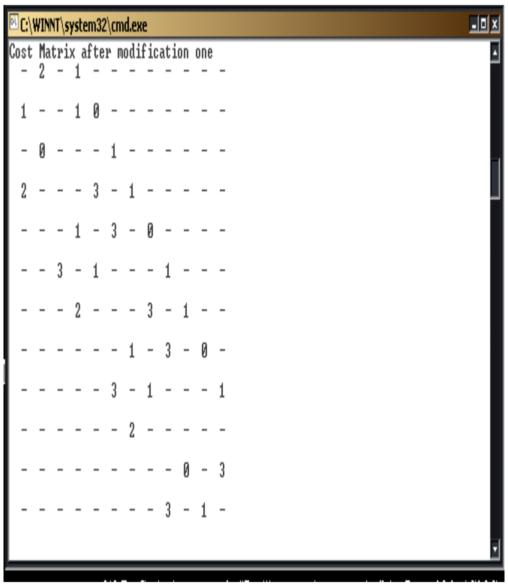
```
Min_seg  >java MinSeg
Enter Graph File name : Topology.txt
Enter Source Vertex : n10
Enter Destination Vertex : n3
```



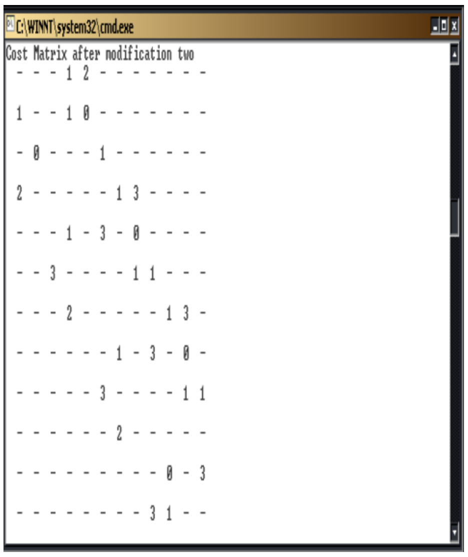
```
Cost Matrix
 - 2 - 1 - - - - - - - -

 1 - 1 1 - - - - - - - -

 - 1 - - - 1 - - - - - -

 2 - - - 3 - 1 - - - - -

 - 1 - 1 - 3 - 1 - - - -

 - - 3 - 1 - - - 1 - - -

 - - - 2 - - - 3 - 1 - -

 - - - - 1 - 1 - 3 - 1 -

 - - - - - 3 - 1 - - - 1

 - - - - - - 2 - - - 1 -

 - - - - - - - 1 - 1 - 3

 - - - - - - - - 3 - 1 -
```



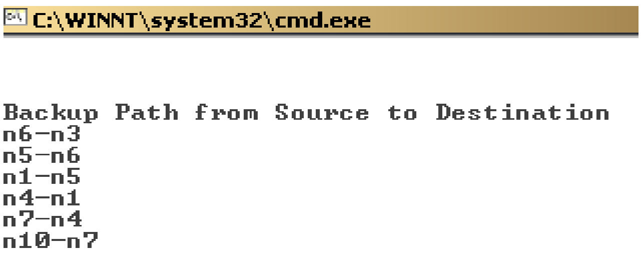
```
Primary Path from Source to Destination
n2-n3
n5-n2
n8-n5
n11-n8
n10-n11
```



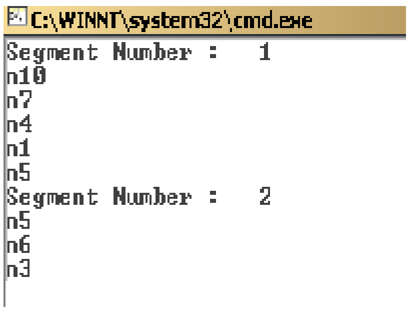
```
C:\WINNT\system32\cmd.exe
Cost Matrix after modification one
 - 2 - 1 - - - - - - - -

 1 - - 1 0 - - - - - - -

 - 0 - - - 1 - - - - - -

 2 - - - 3 - 1 - - - - -

 - - - 1 - 3 - 0 - - - -

 - - 3 - 1 - - - 1 - - -

 - - - 2 - - - 3 - 1 - -

 - - - - - - 1 - 3 - 0 -

 - - - - - 3 - 1 - - - 1

 - - - - - - 2 - - - - -

 - - - - - - - - - 0 - 3

 - - - - - - - - 3 - 1 -
```



```
C:\WINNT\system32\cmd.exe
Cost Matrix after modification two
 - - - 1 2 - - - - - - -

 1 - - 1 0 - - - - - - -

 - 0 - - - 1 - - - - - -

 2 - - - - - 1 3 - - - -

 - - - 1 - 3 - 0 - - - -

 - - 3 - - - - 1 1 - - -

 - - - 2 - - - - - 1 3 -

 - - - - - - 1 - 3 - 0 -

 - - - - - 3 - - - - 1 1

 - - - - - - 2 - - - - -

 - - - - - - - - - 0 - 3

 - - - - - - - - - 3 1 - -
```



```
C:\WINNT\system32\cmd.exe


Backup Path from Source to Destination
n6–n3
n5–n6
n1–n5
n4–n1
n7–n4
n10–n7
```



```
C:\WINNT\system32\cmd.exe
Segment Number :   1
n10
n7
n4
n1
n5
Segment Number :   2
n5
n6
n3
```

VI. CONCLUSION

In this paper a segmented backups and failure recovery scheme for dependable real-time communication in multihop networks is been proposed. This mechanism not only improves resource utilization and call acceptance rate but also can provide faster failure recovery and better QoS guarantees on end-to-end delays without compromising the level of fault tolerance provided. We have also used an efficient backup route selection algorithm. In order to realize the full potential of the method of segmented backups, routing strategies must be developed for backup segments to achieve better QoS guarantees on delay and bounded time failure recovery.

REFERENCES

[1] Krishna Phani Gummadi, Madhavarapu Jnana Pradeep, and C. Siva Ram Murthy, "An Efficient Primary-Segmented Backup Scheme for Dependable Real-Time Communication in Multihop Networks", *IEEE/ACM TRANSACTIONS ON NETWORKING*, VOL. 11, NO. 1, FEBRUARY 2003.

[2] A. Mankin *et al.*, "Resource reservation protocol (RSVP)," *IETF, RFC* 2208, 1997.

[3]. Seungjae Han and Kang G. Shin, A Primary-Backup Channel Approach to Dependable Real-Time Communication in Multihop Networks, *IEEE TRANSACTIONS ON COMPUTERS*, VOL. 47,pp.46-61. NO. 1, JANUARY 1998

[4] C. Dovrolis and P. Ramanathan, "Resource aggregation for fault-tolerance in integrated services networks," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Apr. 1998, pp. 39–53.

[5] S. Han and K. G. Shin, "Efficient spare-resource allocation for fast restoration of real-time channels from network component failures," in *Proc. IEEE Real-Time Systems Symp.*, 1997, pp. 99–108.

[6] Resource Reservation Protocol, Cisco

[7] "OSPF Version 2," *IETF, RFC* 2328, 1998.

[8] http://www.cisco.com/univercd/cc/td/doc/cisintwk/