A Segmented Backup Scheme for Dependable Real Time Communication in Multihop Networks

Gummadi P. Krishna M. Jnana Pradeep and C. Siva Ram Murthy

Department of Computer Science and Engineering
Indian Institute of Technology, Madras - 600 036, INDIA
gphanikrishna@hotmail.com, mjpradeep@yahoo.com, murthy@iitm.ernet.in

Abstract. Several distributed real time applications require fault tolerance apart from guaranteed timeliness. It is essential to provide hard guarantees on recovery delays, due to component failures, which cannot be ensured in traditional datagram services. Several schemes exist which attempt to guarantee recovery in a timely and resource efficient manner. These methods center around a priori reservation of network resources called spare resources along a backup route. In this paper we propose a method of segmented backups which improves upon the existing methods in terms of resource utilisation, call acceptance rate and bounded failure recovery time. We demonstrate the efficiency of our method using simulation studies.

1 Introduction

Any communication network is prone to faults due to hardware failure or software bugs. It is essential to incorporate fault tolerance into QoS requirements for distributed real time multimedia communications such as video conferencing, scientific visualisation, virtual reality and distributed real time control. Conventional applications which use multihop packet switching easily overcome a local fault but experience varying delays in the process. However, real time applications with QoS guaranteed bounded message delays require a priori reservation of resources (link bandwidth, buffer space) along some path from source to destination. All the messages of a *real time session* are routed through over this static path. In this way the QoS guarantee on timeliness is realised but it brings in the problem of fault tolerance for failure of components along its predetermined path. Two *proactive* approaches are in vogue to overcome this problem. The first approach is forward recovery method [1,2], in which multiple copies of the same message are sent along disjoint paths. The second approach is to reserve resources along a path, called *backup path* [3,4], which is disjoint with the primary, in anticipation of a fault in the primary path. The second approach is far more inexpensive than the first if infrequent transient packet losses are tolerable. We focus on the second proactive scheme. Establishment of backup channels saves the time required for reestablishing the channel in reactive methods.

Two different schemes have been widely analysed for the establishment of backup channels. In the first, the spare resources in the vicinity of failed component are used to reroute the channel. This method of *local detouring* [3,4] leads to inefficient resource utilisation as after recovery, the channel path lengths usually get extended significantly. The second method *end to end detouring* was proposed to solve the problem in a resource efficient manner. But end to end detouring has the additional requirement that the primary and backup paths be totally disjoint except the source and destination. This might lead to rejection of a call even when there is considerable bandwidth available in the network. Further, this method of establishing backups might be very inefficient for delay critical applications if the delay of the backup is not within the required limits. In this paper we address these problems by proposing

to have *segmented backups* rather than a single continuous backup path from source to destination and show that the proposed method not only solves these problems but also is more resource efficient than the end to end detouring methods with *resource aggregation* through *backup multiplexing* [5-7].

We now explain our concept of segmented backups. Earlier schemes have used end to end backups, i.e., backups which run from source to destination of a dependable connection, with the restriction that the primary and the backup channels do not share any components other than the source and destination. In our approach of *segmented backups*, we find backups for only parts of the primary path. The primary path is viewed as made up of smaller contiguous paths, which we call *primary segments* as shown in Figure 1. We find a backup path, which we call *backup segment*, for *each* segment independently. Note that successive primary segments of a primary path overlap on at least one link and that any two non consecutive segments are disjoint. The primary channel with 9 links shown, has 3 primary segments: the 1st segment spanning the first 3 links, the 2nd spanning link 3 to link 6 and the 3rd the last 4 links, segments overlapping on the 3rd and 6th links. The backup segments established are also shown. In case of a failure in a component along a primary segment the message packets are routed through the corresponding backup segment rather than through the original path, *only* for the length of this primary segment as illustrated. In case of a fault in any component of a primary path, we give the following method of backup segment activation. If only one primary seg-

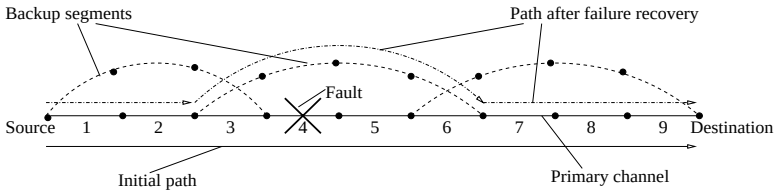

Fig. 1. Illustration of Segmented Backups

ment contains the failed component activate the backup segment corresponding to that primary segment as shown for the failure of link 4. If two successive primary segments contain the failed component activate any one of the two backup segments corresponding to the primary segments. Now we illustrate one of the advantages of

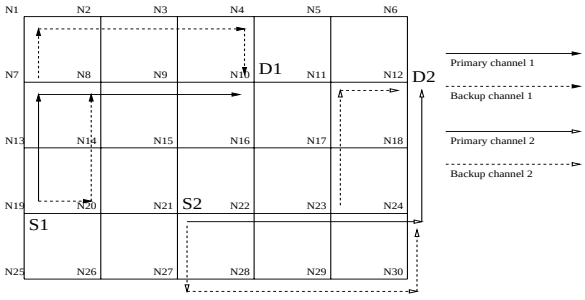

Fig. 2. Establishment of Segmented Backup Channels

the segmented approach over end to end backup approach with a simple example of a 5 X 6 mesh in Figure 2. Suppose the capacity of each link on the mesh is only 1 unit. There are 2 dependable connections to be established : S1 to D1 and S2

to D2. The primary paths (shortest paths) of these connections are shown in the figure. It is not possible to establish end to end backups for both the connections as both the backups contend for the unit resource along the link between N15 to N16. However, segmented backups can be established as shown in the figure.

2 Spare Resource Allocation

It is very important to address the issue of minimizing the amount of spare resources reserved. The idea is to reduce the amount of backup resources reserved by *multiplexing* the backups passing through the same link. We explain the method very briefly below. Refer to [5-7] for more detailed discussion. We note that the resources reserved for backup channels are used only during component failures in their primary channels. We consider *single link* failure model for our analysis, under the assumption that channel failure recovery time i.e., time taken for the fault to be rectified, is much smaller than the network's mean time to failure (MTTF). If primary channels of two connections share no common components and their backup channels with bandwidths b1 and b2 pass through link L, it is sufficient to reserve max(b1, b2) for both the backup channels on the link L in this failure model, as we know that both the backup channels can never be activated simultaneously. This is the idea of multiplexing. We discuss how *deterministic multiplexing* [5,6] applies to our scheme in comparison to earlier schemes.

We use deterministic failure model and calculate the minimum amount of extra resources that are necessary to be reserved to handle all possible cases of failure. We give below the algorithm we use to calculate the spare resources $S_L at link L$ under single link failure model. Let Φ_L denote the set of all primary channels whose backups traverse L. Let R_{P_S} denote the resource required at each link by the primary segment P_S.

Initialise $S_{I,L} = 0 \; \forall \; I, L$
loop for each link $I, I \neq L$
  loop for each primary channel segment $P_S \in \Phi$
    if P_S contains link I then
      $S_{I,L} = S_{I,L} + R_{P_S}$
    endif
  endloop
endloop
$S_L = max\{ \; S_{I,L} \; \} \; \forall \; I \neq L$

It is worth noting the complexity of this multiplexing algorithm. Its execution time increases steeply with increase in the number of links and connections in the network. At first sight it appears as if backup segments taken together, require to reserve more resources than a single end to end backup because segments overlap over the primary channel. But the backup segments tend to multiplex more as their primary segments' lengths are much shorter. Larger the number of backup segments, shorter the primary segments i.e., smaller the number of components in each primary segment and hence, greater the multiplexing among their backup segments. Our method tends to be more resource efficient since there is a considerable improvement in backup segments' *multiplexing capability* over end to end backup's capability. Therefore, our scheme is expected to be more efficient for large networks when a large number of calls are long distance calls.

3 Backup Route Selection

Several elaborate routing methods have been developed which search for routes using various QoS metrics. Optimal routing problem of minimizing the amount of

spare resources while providing the guaranteed fault tolerance level is known to be NP-hard. So we resort to heuristics. Several greedy heuristics for selecting end to end backup paths are discussed in [5]. A shortest path search algorithm like Dijkstra's is enough to find the minimum cost path where the cost value for a link can be made a function of delay, spare resource reservation needed etc. The complexity of our problem of selecting segmented backups is far greater as we have to address additional constraints due to our following design goals.

Improving Call Acceptance Rate: Our scheme tends to improve the call acceptance rate over end to end backups due to two main reasons. Firstly, it tends to improve the call acceptance in situations where there exists a primary path but the call gets rejected due to the lack of an end to end disjoint backup path. We have already shown this through a simple example in Figure 2. Secondly, by reserving lesser amount of resources it allows for more calls to be accepted. This method however, has the problem of choosing the appropriate intermediate nodes (the nodes chosen should not only allow backup segments but should also economize on the resource reservation).

Improving Resource Reservation: This sets up two opposing constraints. First, longer the primary segment of a backup segment, lesser will be the number of backup segments required. Too short primary segments can lead to a requirement of large amounts of resources for the large number of backup segments (Note that each of the backup segments requires more resource than the primary segment which it spans). On the contrary shorter primary segments lead to more multiplexing among their backup segments as described before. So we have to choose primary segments which are neither too short nor too long.

Increase in the Delay Due to Backup: We are interested only in backup segments which do not lead to an unreasonable increment in delay in case of a failure in their primary segment, which constrains the choice of intermediary nodes.

Even in case of end to end detouring we face these constraints but we have a very simple way out. The shortest path algorithm run on the network with the nodes of the primary path removed should give a very good solution and if it fails there does not exist any solution. In contrast, for our scheme we do not have the intermediate destinations fixed and we have to choose among the many possible solutions. In our heuristic we run Dijkstra's shortest path algorithm from source to destination removing all links in the primary path. If in the process, Dijkstra's search algorithm comes to any node in the primary path, we mark it as an intermediate node. Then, we take the node previous to it in the primary path (in the order of increasing distance from the source) and using it as the new source try to find a shortest path to the destination recursively. In order to ensure that the primary segment is not too small we use a parameter MINLEAPLEN which indicates the minimum number of nodes in any primary segment. Thus, we remove the first MINLEAPLEN nodes starting from the new source along the primary path every time before beginning the search for the shortest path to the destination. It is also important that the delay increment for any backup segment is below a threshold Δ for the backup to be of use. This tends to prevent lengthy backups for very small primary segments. In case the destination cannot be reached or the Δ condition is violated, we start Dijkstra's algorithm again from the first segment, this time avoiding the nodes which were chosen as the end of first segment, in previous attempts. The number of times we go back and try again (number of retries) is constant and can be set as a parameter.It is to be noted that our scheme tends to perform better in comparison to the scheme in [6] for large networks, with moderate congestion and for long distance calls. Further, it is important to note that for small networks with short distance calls this scheme mimics the end to end backup scheme in [6] as we do allow a backup to be made of just one segment. In case of connections with very

short primary path our heuristic chooses the backup with a single segment.

4 Failure Recovery

When a fault occurs in a component in the network, all dependable connections passing through it have to be rerouted through their backup paths. This process is called failure recovery. This has three phases: *fault detection*, *failure reporting* and *backup activation*. The restoration time, called failure recovery delay, is crucial to many real time applications, and has to be minimized.

In our model, we assume that when a link fails, its end nodes can detect the failure. For failure detection techniques and their evaluation refer to [8]. After fault detection, the nodes which have detected the fault, report it to the concerned nodes for recovering from the failure. This is called failure reporting. After the failure report reaches certain nodes, the backup is activated by those nodes. Failure reporting and backup activation need to use control messages. For this purpose, we assume a real time control channel (RCC) [6] for sending control messages. In RCC, separate channels are established for sending control messages, and it guarantees a minimum rate of sending messages.

Failure Reporting and Backup Activation: The nodes adjacent to a failed component in the primary path of a dependable connection will detect the failure and send failure reports both towards the source and the destination. In the end to end backup scheme, these messages have to reach the source and destination before they can activate the backup path. In our scheme, this is not necessary. Failures can be handled more locally. The end nodes of the primary segment containing the faulty component on receiving the failure reports initiate the recovery process. These two nodes send the activation message along the backup segment, and the dependable connection service is resumed. This process is illustrated in Figure 3. If there are k segments in the backup, then this gives about $O(k)$ improvement in the time for failure reporting. When a fault occurs, not only do we experience a

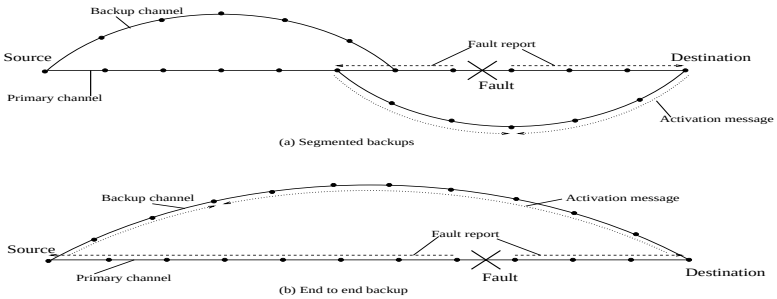

Fig. 3. Illustration of Failure Recovery

disruption of service for some time, but also packets transmitted during the failure reporting time are lost. Most real time applications cannot tolerate much message loss. In our scheme the message loss is reduced to a considerable extent. When a fault occurs in one segment of the primary, only the packets which have entered that segment from the time of the occurrence of the fault till the backup segment activation are lost. Other packets in the segments before and after the failed segment are not affected and will be delivered normally. This is in contrast to the end to end backup case, where *all* packets in transit in the primary path before the failed component, between occurrence of failure and backup activation, are lost.

5 Delay and Scalability

Delay: In Real Time Communication, it is essential to have the delays along both the primary and the backup channels to be as low as possible. Hence, we might have a restriction on the amount by which the delay along the backup exceeds that along the primary path. Let the total delay along the backup path not exceed the delay along the primary by Δ, a specified QoS parameter.
Thus, the constraint for choosing an end to end backup is,

delay(backup path) - delay(primary path) $\leq \Delta$.

In the case of segmented backups, this constraint is,

(delay(backup segment i) - delay(primary segment i)) $\leq \Delta$, $\forall$ i.

We see that in our case we have to minimize the delay increase for each segment independently. Hence call acceptance rate will be better since it is easier to find small segments than to find big end to end backups satisfying the Δ constraint.

Scalability: The segmented backup scheme scales well since it does not demand global knowledge and does not involve any kind of broadcast. There is no necessity for a network manager and this scheme works well in a distributed network. For Backup Multiplexing each node needs to know the primary paths of the channels whose backups pass through it. This is easily accomplished if the information is sent along with the packet requesting the establishment of backup channel. Upon encountering faults, control messages are not broadcast, but sent only to a limited part of the network affected by the fault.

In large networks, the effectiveness of the segmentation increases as the mean path length of connections increases. Since the calculation of spare resources using multiplexing has to be done per segment independently, this scheme scales better than the earlier end to end methods.

6 Performance Evaluation

We evaluated the proposed scheme by carrying out simulation experiments similar to those in [6], on a 12 X 12 mesh. We also implemented the end to end backup scheme [6] for comparative study.

In the simulated network, neighbour nodes are connected by two simplex links, one in each direction, and all links have identical bandwidth. For simplicity, the bandwidth requirement of all connections was put equal to 1 unit. The delay of each link was set to 1, thereby making delay along any path equal to its path length. Primary channels were routed using a sequential shortest-path search algorithm. The end to end backups were also routed using the same shortest-path search algorithm, with the condition that it does not contain any component of the primary other than source and destination. The amount by which backup path delay can exceed primary path delay was used as a parameter, Δ.

We find the backup segments as described in Section 3. The number of retries was set to 9 in our simulation experiments. The MINLEAPLEN parameter was set to 4. Connections were requested incrementally, between a source and destination chosen randomly, with the condition that no (source, destination) pair is repeated, and the length of the shortest path between them is at least MINPATHLEN. In our simulation studies, connections were only established but not torn down since (i) the computational time required for release of connections is considerably high, and (ii) earlier studies with end to end backups [5,6] also do the same.

The results are shown in Table 1. In this table, we show the statistics at different instants of time as in the simulation. The number of connections requested is proportional to the time. The network load at the time is also shown. Table 1 shows the average amount of spare bandwidth reserved per connection, both for segmented

(seg) and end to end (end) backups, for different values of Δ. We show the results for MINPATHLEN=6, and for MINPATHLEN=8. The average path lengths in the two cases was 10.8 and 12.3. The bandwidth of the links was chosen as 100 units for MINPATHLEN=6 and 90 units for MINPATHLEN=8. As expected, the spare bandwidth reserved was much lower for segmented backups. Also, the improvement is seen to be more in the second case. This illustrates that as the average length of connections increases, the effectiveness of segmented backups increases.

The cumulative number of requests rejected till an instant of time was also noted. The number rejected by the segmented backup scheme was seen to be much lesser than that of the end to end scheme.

Table 1. Average amount of spare bandwidth reserved per connection

MINPATHLEN = 6		$\Delta = 2$		$\Delta = 4$		MINPATHLEN = 8		$\Delta = 2$		$\Delta = 4$	
Time	n/w load	end	seg	end	seg	Time	n/w load	end	seg	end	seg
1245	42%	7.55	7.06	7.50	7.07	1284	53%	8.72	8.16	8.71	8.16
1559	51%	7.40	6.88	7.42	6.91	1488	59%	8.76	8.12	8.74	8.13
1846	57%	7.49	6.93	7.52	6.99	1708	63%	8.74	8.09	8.74	8.11
2148	63%	7.51	6.98	7.52	7.11	1911	65%	8.69	8.08	8.68	8.18
2442	67%	7.48	6.99	7.49	7.15	2131	66%	8.64	8.10	8.63	8.20

7 Conclusions

In this paper, we have proposed segmented backups: a failure recovery scheme for dependable real-time communication in distributed networks. This mechanism not only improves resource utilisation and call acceptance rate but also provides for faster failure recovery. We evaluated the proposed scheme through simulations and demonstrated the superior performance of the scheme compared to earlier end to end backup schemes [5-7]. In order to realise the full potential of the method of segmented backups, better routing strategies have to be developed for choosing intermediate nodes optimally. We also need faster algorithms for backup multiplexing.

References

1. P. Ramanathan and K. G. Shin, "Delivery of time-critical messages using a multiple copy approach," *ACM Trans. Computer Systems,* vol. 10, no. 2, pp. 144-166, May 1992.
2. B. Kao, H. Garcia-Molina, and D. Barbara, "Aggressive transmissions of short messages over redundant paths," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 1, pp. 102-109, January 1994.
3. Q. Zheng and K. G. Shin, "Fault-tolerant real-time communication in distributed computing systems," in *Proc. IEEE FTCS,* pp. 86-93, 1992.
4. W. Grover. "The self-healing network: A fast distributed restoration technique for networks using digital crossconnect machines," in *Proc. IEEE GLOBECOM,* pp. 1090-1095, 1987.
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 RTSS,* pp. 99-108, 1997.
6. S. Han and K. G. Shin, "A primary-backup channel approach to dependable real-time communication in multihop networks," *IEEE Trans. on Computers,* vol. 47, no. 1, pp. 46-61, January 1998
7. C. Dovrolis and P. Ramanathan, "Resource aggregation for fault tolerance in integrated services networks," *ACM SIGCOMM Computer Communication Review,* 1999.
8. S. Han and K. G. Shin, "Experimental evaluation of failure detection schemes in real-time communication networks," in *Proc. IEEE FTCS,* pp. 122-131, 1997.

This article was processed using the LaTeX macro package with LLNCS style