Efficient Spare-Resource Allocation for Fast Restoration of Real-Time Channels from Network Component Failures *

Seungjae Han and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109–2122.
email: {sjhan, kgshin}@eecs.umich.edu

Abstract

Since real-time applications usually require not only timeliness but also fault-tolerance, it is essential to incorporate fault-tolerance into real-time communication services that are indispensable to distributed real-time applications. The techniques for failure recovery in datagram communication are not adequate for real-time communication, because they cannot provide recovery-delay guarantees. To ensure fast recovery of a real-time channel from network component failures, we need to reserve network resources (called "spare resources") along a backup route before failures actually occur. The focus of this paper is on minimizing the amount of spare resources while meeting the fault-tolerance requirement. Specifically, we present resource sharing mechanisms and backup-route selection algorithms, and evaluate their efficiency with extensive simulations.

1 Introduction

In an Internet-like multihop packet-switched network, messages experience random delays due to their contention for network resources, and message streams (i.e., sessions) suffer from uncertain throughput resulting from this contention. Special schemes are therefore needed to guarantee bounded message delay or predictable message throughput, i.e., for QoS-guaranteed service. Resource reservation in conjunction with admission control has been a popular approach. Basically, resources (e.g., CPU/link bandwidth, buffer space) are reserved for each session along a static path, and messages of the session are sent over this path. Such a virtual-circuit is often called a *real-time channel/session*. The amount of resources to be reserved is calculated from the session's traffic generation behavior and QoS requirement. The admission test checks the availability of necessary resources; a real-time channel request will be accepted (rejected), if (not) enough resources are available.

While this reservation-based approach has been successful in providing timeliness QoS guarantees, it causes a serious difficulty in dealing with network component failures. It is because messages of a real-time channel/session can traverse only the path on which the necessary resources are reserved *a priori* for the channel, so that they cannot be detoured around the faulty components on the fly, unlike in datagram communication. Two approaches have been conceived for fault-tolerant real-time communication in multi-hop networks. The first approach is the forward-recovery method [1–3], where multiple copies of a message are sent simultaneously via multiple disjoint paths to mask the effects of failures. This is an expensive technique but has an advantage of handling failures without any service disruption. However, if infrequent packet losses due to transient failures are tolerable (as in many real-time applications like multimedia networking), the approach to detect and recover, in real-time, from persistent failures is a cost-effective alternative, because it eliminates (or greatly reduces) the resource overhead in achieving fault-tolerance. In this paper we do not consider the forward-recovery approach, but focus on the alternative approach mentioned above.

The simplest way of recovering from a disabled real-time channel is to establish a new real-time channel which does not use the failed components. This *reactive* method is studied in [4]. Despite its low overhead in the absence of failures, it does not guarantee successful recovery, because the channel re-establishment attempt can fail due to resource shortage at that time. Even when there are sufficient resources, contention among simultaneous recovery attempts for different disabled channels may require several trials to succeed, thus delaying service resumption and increasing network traffic.

To avoid/minimize resource shortage during failure recovery, one can reserve *spare resources*. The scheme

*The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895 and the Office of Naval Research under Grant N00014-94-1-0229. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

proposed in [5] follows this concept; spare resources in the vicinity of the faulty components are utilized to reroute the disabled channels. Similar schemes [6–8] have been developed for reliable telecommunication service, which has resemblance to real-time communication in packet-switched networks such as dedicated resources and static routing. A drawback of this *local detouring* approach is that the resource usage becomes inefficient after failure recovery, because channel path-lengths are usually extended significantly by local detouring (e.g., in mesh networks).

End-to-end detouring solves this problem, which requires about 30% less spare resources than the local-detouring approach in mesh networks [9, 10]. In this approach, a *backup channel* is established between two end-points of each real-time channel (i.e., a *primary channel*), and the backup channel takes over the primary channel's role in case of failure to the primary. Each backup channel reserves its own spare resources, so that there will be no conflict/contention between recovery attempts. Furthermore, since a backup channel is established before failures actually do occur, one can use it immediately upon occurrence of a failure to the primary, without the time-consuming channel (re)establishment process.[1] Thus, the failure-recovery delay of this method is much smaller than that of the reactive method, and hence, we call it '*fast recovery*.' There is a hybrid approach of reactive and end-to-end methods. In this approach, spare resources are reserved but backup paths are not decided until failures actually occur. When failures occur, each faulty channel attempts to establish a new channel by "claiming" the reserved spare resources.

The end-to-end detouring approach has been studied mainly for ATM-based telecommunication networks. Some of recent efforts on this approach can be found in [9–12]. Essentially, they derive optimal routing of channels (i.e., VPs in an ATM network) to minimize spare resources while guaranteeing fast recovery under a deterministic failure model[2]. They assume that all channel demands are known at the time of network design and change very rarely; the addition of a new channel requires recalculation of all channel paths. Since the derivation of the optimal routing is computationally very expensive, these schemes cannot be applied to a 'dynamic' environment where short-lived channels are set up and torn down frequently. Our interest is in developing an efficient end-to-end detouring scheme, which is applicable to such a dynamic channel setup/teardown environment in multi-hop packet-switched networks. The next section gives an overview of the proposed scheme.

[1] The time required to establish a real-time channel is relatively large and unbounded even without contention, since channel-establishment messages are usually sent as datagrams and non-trivial calculation is necessary at each node on the channel path for the admission test.

[2] Typically, the single link failure model.

2 Primary-Backup Channel Scheme

A *dependable real-time connection* (a $\mathcal{D}$-connection for short) consists of a primary channel and one or more backup channels. Each backup channel remains as a cold standby until it is activated, i.e., it does not carry any data under a normal condition. When a channel is disabled by a network component failure, the failure is detected and reported to the channel's end nodes, which are responsible for further failure handling. If the disabled channel is a primary channel, one of its healthy backups is activated to become a new primary channel. If the disabled channel happens to be a backup channel, a new backup channel is established to preserve the fault-tolerance capability of the corresponding $\mathcal{D}$-connection. Unless multiple failures simultaneously disable all channels of a $\mathcal{D}$-connection or occur to its primary channel before its backups are properly set up, 'fast recovery' is always guaranteed.

The above failure-handling scenario requires to solve two major problems: (i) backup establishment before a failure and (ii) failure handling upon occurrence of a failure. In this paper, we will focus only on the first problem without addressing the issues related to the second problem such as failure detection, reporting, and backup activation. (See [13, 14] for solutions to these issues.) Two key issues in backup establishment are 'backup-route selection' and 'spare-resource calculation.' In contrast to the existing end-to-end schemes for non-dynamic environments, we separate these two issues from each other, so that backup channels can be routed incrementally without requiring recalculation of existing channel paths, and the amount of required spare resources is determined from the given routing results.

The rest of the paper is organized as follows. Section 3 describes our approach to the problem of spare resource calculation. Sections 4 and 5 deal with the issue of backup-route selection. Section 6 presents simulation results, demonstrating the superior performance of the proposed scheme. The paper concludes with Section 7.

3 Spare-Resource Calculation

The links/nodes used by a primary channel may preferably be avoided in routing its backups, in order to prevent a single failure from disabling all channels of the same $\mathcal{D}$-connection. Throughout this paper, we assume disjoint routing of the channels belonging to the same $\mathcal{D}$-connection. As a result of disjoint routing, equipping each $\mathcal{D}$-connection with a single backup reduces the network capacity of accommodating $\mathcal{D}$-connections by 50% or more, as a backup channel requires at least the same amount of resources to be reserved as its primary channel. The large spare resources can seriously degrade the attractiveness of our scheme.

To reduce the overhead by spare resources, we have developed a resource sharing technique, called *backup multiplexing*. Its basic idea is that on each link, we reserve only a very small fraction of link-resources needed for all backup channels going through the link.[3] In what follows, we present two methods for backup multiplexing: (i) 'deterministic' method with 'resource aggregation' and (ii) 'probabilistic' method with 'admission overbooking.'

3.1 Deterministic Multiplexing

This method adopts a deterministic failure model and calculates the exact amount of spare resources which are just enough to handle all possible cases under the assumed failure model. As an example, the algorithm to calculate the spare resources (s_ℓ) at link ℓ under the single-link failure model is given in Figure 1 (a). Whenever a backup channel is established, this algorithm has to be run on all links of the backup path. In Figure 1 (a), Φ_ℓ^1 denotes the set of all primary channels whose backups traverse ℓ, and r_m is the resource required at each link by the primary channel M_m. One backup for each connection is enough to tolerate any single link failure. The algorithm for the single node failure model can be easily devised by slightly modifying this algorithm.

Usually, spare-resource reservation based on single failure models provides a sufficient level of fault-tolerance, since the channel failure recovery time is much smaller than MTBF(Mean Time Between Failures) of the network components. Nevertheless, if a higher level of fault-tolerance is required, multiple backups should be set up. As an example, the algorithm for double-link failure tolerance is presented in Figure 1 (b). To tolerate all possible double-link failures, each primary channel needs two backups. In Figure 1 (b), Φ_ℓ^2 denotes the set of all primary channels whose second backups traverse ℓ, and $B1_m$ denotes the first backup of M_m.

---

loop for each link i, $i \neq \ell$
  loop for each primary channel $M_m \in \Phi_\ell^1$
    if M_m contains link i **then**
      $s_{i,\ell}^1 \longleftarrow s_{i,\ell}^1 + r_m$
    endif
  endloop
endloop
$s_\ell \longleftarrow max\{s_{i,\ell}^1\}, \forall i \neq \ell$

(a) An algorithm for single-link failure tolerance

loop for each link i, $i \neq \ell$
  loop for each primary channel $M_m \in \Phi_\ell^1$
    if M_m contains link i **then**
      $s_{i,\ell}^1 \longleftarrow s_{i,\ell}^1 + r_m$
    endif
  endloop
endloop
loop for each link pair (i,j), $i \neq j$, $i \neq \ell$, $j \neq \ell$
  loop for each primary channel $M_m \in \Phi_\ell^2$
    if M_m contains i and $B1_m$ contains j **then**
      $s_{i,j,\ell}^2 \longleftarrow s_{i,j,\ell}^2 + r_m$
    endif
  endloop
endloop
$s_\ell \longleftarrow max\{s_{i,\ell}^1 + s_{i,j,\ell}^2\}, \forall i \neq \ell, \forall j \neq \ell$

(b) An algorithm for double-link failure tolerance

Figure 1: Deterministic multiplexing algorithms

3.2 Probabilistic Multiplexing

The probabilistic multiplexing method is designed to cope with non-deterministic failures, assuming that each network component fails with a certain rate instead of assuming the maximum number, or a particular type, of failures. Backup channels are multiplexed indirectly via a modified admission test (or *meta-admission test*). Thus, some existing backup channels are not accounted for in the admission test of a new backup channel, which is, in essence, equivalent to resource sharing between the new backup and those backups unaccounted for. Deciding which backup channels will not be accounted for in the admission test of a backup channel is a crucial problem. Our strategy is to multiplex those backups which are less likely to be activated simultaneously.

The probability of simultaneous activation of two backups of two different $\mathcal{D}$-connections is bounded by the probability of simultaneous failure of their respective primary channels. This probability depends on the routing of the primary channels, and increases with the number of components shared among the primaries. Assuming that failures occur independently with the same probability λ, for each link, we can calculate the probability — denoted by $\mathcal{S}(B_i, B_j)$ — of simultaneous activation of two backups, B_i and B_j, whose primaries are M_i and M_j, respectively; $\mathcal{S}(B_i, B_j) \approx sc(M_i, M_j) \cdot \lambda$, where $sc(M_i, M_j)$ is the number of components shared between M_i and M_j. (See [14] for the derivation.) Based on this probability, the set of backups to be multiplexed together is determined for each backup on a link. B_i and B_j are multiplexed if $\mathcal{S}(B_i, B_j)$ is smaller than a certain threshold ν, called the *multiplexing degree*. So, the rule to decide resource sharing in the single backup configuration is:

$$M_i \bowtie M_j \quad \Rightarrow \quad B_i \parallel B_j,$$

where $M_i \bowtie M_j$ indicates that $sc(M_i, M_j) \cdot \lambda \geq \nu$, and $B_i \parallel B_j$ indicates that B_i and B_j are not multiplexible. When $\Pi_{B_i,\ell} = \{B_\alpha, B_\beta, \ldots\}$ denotes the set of backups which are not multiplexed with B_i on link ℓ, the spare resource requirement at link ℓ is equal to the highest resource demand among all sets of $\{\Pi_{B_i,\ell} + B_i\}$.

To tolerate double failures, the following set of rules

[3] In this paper, we consider only link-bandwidth for simplicity, but other resources like buffer and CPU can be treated similarly.

should be applied in the double backup configuration:

Rule-1:	$M_i \bowtie M_j \Rightarrow B1_i \parallel B1_j$,
Rule-2:	$B1_i \bowtie M_j \Rightarrow B2_i \parallel B1_j$,
Rule-3:	$B1_i \bowtie B1_j \Rightarrow B2_i \parallel B1_j$,
Rule-4:	$(B1_i \bowtie M_j)\&(M_i \bowtie B1_j) \Rightarrow B2_i \parallel B2_j$,
Rule-5:	$(M_i \bowtie M_j)\&(B1_i \bowtie B1_j) \Rightarrow B2_i \parallel B2_j$,

where $B1_i$ and $B2_i$ are the first and second backups of M_i, respectively. Rule-1 is for the relation between first backups, Rule-2 and 3 are for the relation between first and second backups (belonging to different connections), and Rule-4 and 5 are for the relation between second backups. For example, Rule-2 says that $B2_i$ and $B1_j$ should not be multiplexed to prepare for the case when M_i fails and a shared component between $B1_i$ and M_j fails, which causes the simultaneous activation of $B2_i$ and $B1_j$. Other rules can be reasoned similarly.

3.3 Comparative Discussion

Under deterministic multiplexing, spare resources at each link is determined as an aggregated entity, s_ℓ. This type of backup multiplexing is possible only when resource reservation is completely interchangeable among channels. However, this condition does not hold for all real-time channel schemes, but in general it is valid only for 'rate-based' schemes, not for 'scheduler-based' schemes (we borrowed this classification from [15]).

In the rate-based schemes [16, 17], QoS has a static relation with the traffic characteristics. For example, a higher message rate (hence, higher bandwidth) results in a smaller message delay. In these schemes, the admission test at a link simply examines whether the demanded resources exceed the available resources, since the amount of resources determines the QoS level. By contrast, in the scheduler-based schemes [18, 19], the delay requirement of a channel can be specified independently of its traffic characteristics. In such schemes, the admission test checks for the schedulability of a channel by deriving a feasible priority assignment to meet its delay requirement while considering the worst-case contention with existing channels. Because the priority of a channel is determined by considering not only its bandwidth requirement but also its delay requirement, resources needed to guarantee the QoS of a channel may not be sufficient for other channels with a different delay requirement, even if they have the same traffic characteristics. The inapplicability of deterministic multiplexing to scheduler-based schemes is detailed further in Appendix A.

Probabilistic multiplexing is more general in that resource overbooking through the meta-admission test is possible regardless of the underlying real-time channel schemes. Tolerance of deterministic failures is also possible with this method. For instance, if ν is set to λ, fast recovery from any single node/link failure is guaranteed, because no backups whose primaries overlap with each other will be multiplexed. Similarly, if ν is set to 3λ, any single link failure can be tolerated. However, the amount of spare resources resulting from probabilistic multiplexing may not be as small as that from deterministic multiplexing (if it is applicable). It is because the primary channels of the members of $\Pi_{B_i,\ell}$ may not overlap with M_i at the same component.[4] The resource overbooking technique overestimates the spare resource needs, so that it actually can tolerate most of single channel failures instead of single component failures.

4 Initial Backup-Route Selection

The shortest-path (i.e., minimum-hop path) algorithm is often used for channel route selection. A node which wants to set up a real-time channel broadcasts a route-search message to find a shortest path. If network topology information is maintained at each node, a path can be found without broadcasting route-search messages. Recently, more elaborate routing algorithms have been developed, such as those in [20–22]. For instance, in [21], a smallest-delay path, instead of a minimum-hop path, is selected. In [20], a smallest-delay path is selected among minimum-hop paths. Another popular metric of QoS routing is the residual bandwidth, so as to favor a path with larger available bandwidth. The algorithm presented in [22] uses a metric which aggregates multiple routing parameters such as throughput, delay, and error rate. Unlike the above-cited research, our interest is in backup-route selection.

4.1 Optimal Routing Problem

Routing of backup channels has a significant impact on the amount of required spare resources. Essentially, we want to minimize the amount of spare resources while providing the required fault-tolerance level. Unfortunately, there doesn't exist any efficient algorithm for 'optimally' routing backup channels; the problem of finding a path set with multiple constraints is known to be NP-complete. The NP-completeness proof of the following decision problem – which is subsumed by the backup routing problem — can be found in [23]: "*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 channel traffic demands are given?*" The backup route-selection problem is therefore NP-complete, even without considering backup multiplexing. As a result, we have to resort to heuristics for initial backup route selection, that reduce complexity at the expense of optimality. The optimization of initial routing is discussed in Section 5.

4.2 Greedy Heuristics

A backup path is selected among a set of "eligible" paths, where the eligibility of a backup path can be defined by maximum path length, end-to-end delay, or bandwidth. By 'greedy', we mean the selection of a path which appears best when a backup channel is set up. A shortest-path search algorithm like Dijkstra algorithm is used to find the minimum-cost path where a cost value is heuristically assigned to each network

[4] We only check if those primary channels overlap with M_i or not.

link. When multiple backups are set up for each $\mathcal{D}$-connection, one can use such algorithms as in [24, 25] which find multiple disjoint paths.

The link cost functions considered in this paper are:

$$\begin{aligned}
f_1(\ell, B_i) &= 1, \\
f_2(\ell, B_i) &= \Psi_\ell, \\
f_3(\ell, B_i) &= \Delta_{B_i,\ell}, \\
f_4(\ell, B_i) &= f_1(\ell, B_i) + \omega_1 \cdot f_2(\ell, B_i), \\
f_5(\ell, B_i) &= f_1(\ell, B_i) + \omega_2 \cdot f_3(\ell, B_i), \\
f_6(\ell, B_i) &= f_2(\ell, B_i) + \omega_3 \cdot f_1(\ell, B_i), \\
f_7(\ell, B_i) &= f_2(\ell, B_i) + \omega_4 \cdot f_3(\ell, B_i), \\
f_8(\ell, B_i) &= f_3(\ell, B_i) + \omega_5 \cdot f_1(\ell, B_i), \\
f_9(\ell, B_i) &= f_3(\ell, B_i) + \omega_6 \cdot f_2(\ell, B_i),
\end{aligned}$$

where ℓ is a link identifier and B_i is the backup channel to be routed. f_1 and f_2 are popular cost functions for general real-time channel routing, and will be used as references for performance comparison in Section 6. f_3 is a new cost function we devised by exploiting the property of backup multiplexing. The other cost functions are composed by combining these three basic cost functions.

When f_1 is used, all links have an identical cost, and hence, the minimum-hop path will also be the minimum-cost one. This simple cost function differs from the rest in that it does not utilize the knowledge on resource usage. f_2 is an attractive/popular cost function when increasing the network throughput is a main concern. The rationale behind this cost function is to disperse resource reservation for balancing traffic loads. Ψ_ℓ in f_2 is equal to the total resources reserved on link ℓ by both primary and backup channels. $\Delta_{B_i,\ell}$ in f_3 is the incremental spare resources at ℓ, if B_i is to be established on ℓ.

The weights in the composite cost functions are selected so that the first terms may become primary factors and the second terms may be used only to break ties among multiple candidate paths. For example, ω_1 is small enough to ensure $\omega_1 \cdot f_2(\ell, B_i) < f_1(\ell, B_i)$. Greedy-style routing heuristics are named according to the cost functions used, as GM_1, GM_2, …, GM_9, respectively.

4.3 Information Management

Different kinds of information are needed for different routing heuristics. GM_1 requires the information about network topology with the health information of each network component, GM_2 requires the information about resource reservation at each link, and GM_3 requires the path information of primary channels to predict the aftermath of backup multiplexing. There are two options for managing such information; each node can maintain (i) a database of global knowledge (about others) or (ii) a database of local knowledge (about itself). In the former, all information is broadcast in the network whenever any change (e.g., channel setup or teardown) occurs, so that the source node of a channel can decide its path using the information in the node's database. In the latter, there is no information broadcast, but instead, routing messages are broadcast to execute a distributed shortest-path algorithm. Information outage is inevitable in both options, because the network condition may change during the resource reservation stage when multiple channels are established simultaneously by different nodes.

However, even though the information used for routing is out-of-date during resource reservation, fault-tolerance QoS guarantee remains unaffected, because backup multiplexing has nothing to do with the stale information used by routing heuristics. It only results in routing a backup channel over a less efficient path.

5 Periodic Reconfiguration

In the greedy routing, those backup channels which have already been established are not disturbed for routing a new backup channel. Intuitively, if we reroute early backup channels which had been routed without considering later backup channels, we can improve the overall resource usage efficiency.[5] Repair of failed components is another case in which rerouting existing backups is beneficial. Network component failures will disable/activate backups, and new backups will be routed to substitute for the old (i.e., disabled or promoted) backups by avoiding the failed components, possibly on a longer path than the original backup. When the failed components are repaired, the efficiency of resource usage can be improved by rerouting some backups over the repaired components.

Recalculating all existing backup paths whenever a new backup is established is what we want to avoid in the dynamic channel setup/teardown environment. Instead, we adopt a 'two-step' approach. Channels are set up quickly using a greedy routing method, then reconfiguration (for optimizing resource usage) is performed periodically considering only the channels which exist at the moment the reconfiguration starts. Integer Programming (IP) can be used for the optimal reconfiguration of existing backups. However, its computational complexity is very high. If the search space of the IP model includes all possible paths, the dimension of search space is an exponential function of the product of link numbers and channel numbers. In Appendix B, we present, as an example, an IP formulation to achieve 100% fast recovery from any single link failure under deterministic multiplexing. In this model, the dimension of constraint matrix for the simulation condition used in Section 6 easily reaches several thousands. Long computational delay involves a risk that some reconfiguration decisions may become less beneficial or meaningless, since existing backups can be torn down or new channels can be added dur-

[5] Rerouting primary channels will make further improvement possible, since routing of primary channels plays a key role in backup multiplexing. However, moving primary channels is a complex process and can cause service disruptions. So, we do not consider the rerouting of primary channels.

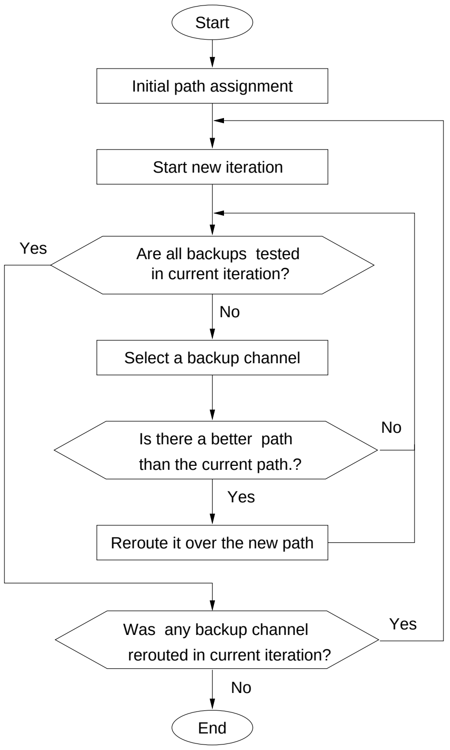


Figure 2: The iterative optimization method

ing the optimization. Therefore, IP is not a practical solution for a large-scale network in the dynamic environment. As a practical alternative, we developed an iterative optimization method.

The flowchart in Figure 2 depicts the iterative optimization method. Starting with an initial path assignment, a new minimum-cost path of each backup is searched, one at a time. When there is a new path which requires less spare resources than the original path, the backup is moved to the new path. The iteration is continued until no further improvement is made. This algorithm always converges and the number of iterations is bounded by the square of the total number of backups in the network, as the total cost (i.e., network-wide sum of spare resources) monotonically decreases as channels are rerouted. However, the final result is not necessarily optimal since we do not explore the benefit of rerouting multiple backups simultaneously. Another reason for sub-optimality is the lack of checking local optimality.[6]

6 Performance Evaluation

In this section, we evaluate the performance of routing heuristics in conjunction with backup multiplexing via extensive simulations. The metric for performance comparison is the average spare resource at a link to meet the fault-tolerance requirement.

[6] Rerouting a backup over a path with the same or larger cost can reduce the costs of other backups by a greater margin.

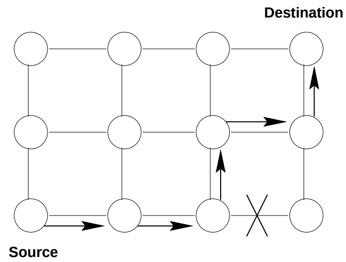


Figure 3: Primary channel routing

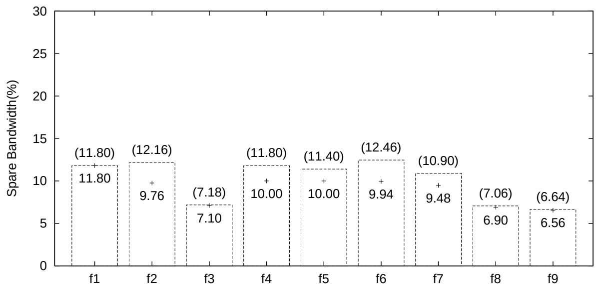
Some general simulation setups are as follows. The simulation network consists of nodes and links, where two adjacent nodes are connected by two simplex links of the same capacity, one for each direction. $\mathcal{D}$-connections are established incrementally, one at a time. The fault-tolerance requirement is 100% fast recovery from any single link failure. To meet this requirement, each $\mathcal{D}$-connection is equipped with a single backup in addition to a primary channel. Channels of each $\mathcal{D}$-connection were routed sequentially, such that the primary channel was routed first and its backup next. A primary channel is routed on a topologically shortest path which is closest to the boundary of possible routing area (see Figure 3, in which the cross sign means the unavailability of a link due to failures or resource shortage). This allows a wider search space for backup routing. A backup is routed on a path which is link-disjoint with its primary. The end-to-end delay requirement of each channel is assumed to be met if the channel path is not longer than the shortest-possible path by more than 2 hops.

6.1 Simulation Results

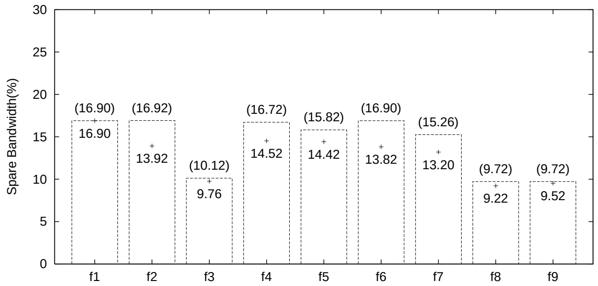
The first experiment (Case 1) was conducted in a 5×5 torus network (i.e., a wrapped square mesh), in which each node has four incoming links and four outgoing links. The bandwidth requirement of each $\mathcal{D}$-connection was identical, so each channel (both primary and backup) required 2% bandwidth on each link of its paths. The end points of each $\mathcal{D}$-connection were evenly distributed across the network. A total of 600 $\mathcal{D}$-connections were established, such that there existed a $\mathcal{D}$-connections between each node pair, i.e., $25 \cdot 24 = 600$.

As a result of establishing 600 $\mathcal{D}$-connections, 30% of total network capacity was reserved for primary channels. We call this value 'primary load' (pl), which indicates the ratio of total bandwidth consumed by all primary channels to the total bandwidth capacity of all links. Figure 4 shows the simulation results for Case 1. The average amount of spare resource required without backup multiplexing was 36% of total network capacity.[7] We call this value 'backup load' (bl). The average spare bandwidth by the greedy method is depicted as a bar, while the reduced spare bandwidth after the iterative optimization is labeled with '+'.

[7] In this case, the overhead of backup channels without mul-

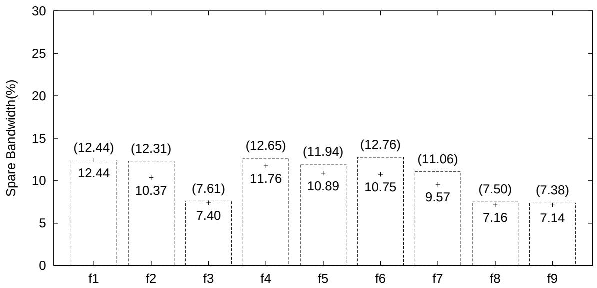


(a) Deterministic multiplexing

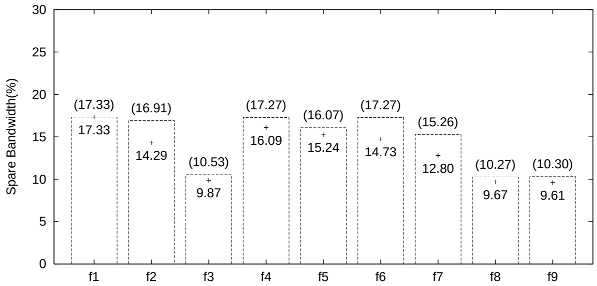


(b) Probabilistic multiplexing

Figure 4: Case 1 (pl = 30%, bl = 36%)



(a) Deterministic multiplexing



(b) Probabilistic multiplexing

Figure 5: Case 2 (pl = 30%, bl = 36%)

Then, we relaxed the homogeneity condition of Case 1 in next experiments. In Case 2, the bandwidth requirement of each $\mathcal{D}$-connection was not identical, but three types of connections were mixed: 1/3 connections of 1% bandwidth, 1/3 connections of 2% bandwidth, and the remaining 1/3 connections of 3% bandwidth among a total of 600 $\mathcal{D}$-connections. The selection of connection end-points and the network topology were the same as Case 1. Figure 5 shows the simulation results.

In Case 3, while keeping other conditions the same as Case 1, the selection of connection source nodes was restricted such that all connections were originated from only 10 randomly-selected nodes among 25 nodes. The destination nodes were evenly distributed as in Case 1. In Case 3, the pattern of resource reservation had hot-spots around the selected source nodes, and not all backup channels could be established without multiplexing. Thus, the backup load could not be measured. The simulation results are shown in Figure 6.

In Case 4, we established 600 $\mathcal{D}$-connections in a 5 $\times$ 5 unwrapped square mesh, in which 18 boundary nodes do not have wrap-links. The source/destination nodes were evenly selected, but the bandwidth requirement of each channel was set to 1% instead of 2%, considering the smaller network capacity of the unwrapped square mesh. The resultant primary and backup load were 25% and 30%, respectively. The simulation results are presented in Figure 7.

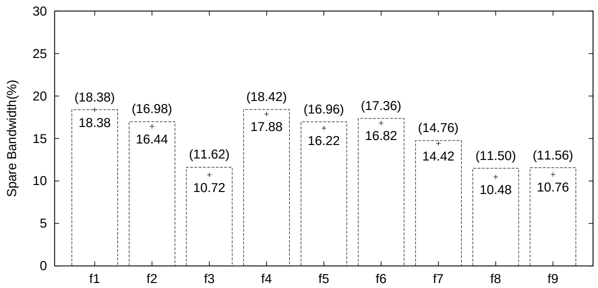
tiplexing is $36/30 \times 100 = 120\%$.

6.2 Qualitative Analysis

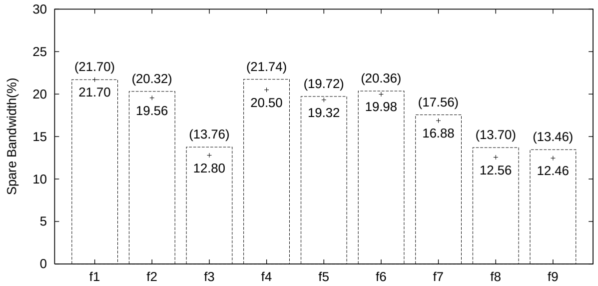
Some observations from the simulation results and their implications are summarized below.

(i) For the initial route selection, those heuristics which include f_3 (i.e., GM_3, GM_5, GM_7, GM_8, GM_9) outperformed the other heuristics by a large margin. Particularly, GM_3, GM_8, and GM_9 showed good performance, where the performance of GM_2 was not much better than that of GM_1. The performance was improved by properly combining basic cost functions. GM_5 and GM_7 outperformed GM_1 and GM_2, and both GM_8 and GM_9 did over GM_3. These results suggest that the general-purpose heuristics like f_1 and f_2 are less effective for backup routing than those heuristics which directly take advantage of backup multiplexing.

(ii) Inhomogeneous network conditions degrade the performance of backup multiplexing. With mixed bandwidth requirements (Case 2), the efficiency of backup multiplexing was degraded, though not significant. Large performance loss was observed in Case 3. It is because channel routes were concentrated on the hot spots, thus discouraging backup multiplexing. The same explanation is possible for Case 4, in which channel routing is more concentrated around the central area of the network than in the torus. In general, backup multiplexing is less effective in a sparsely-connected network like an unwrapped mesh, because there are a smaller number of possible backup paths, and hence, backups have to share the same links even if they are related, which hinders effective resource sharing among backups. As a consequence of the smaller

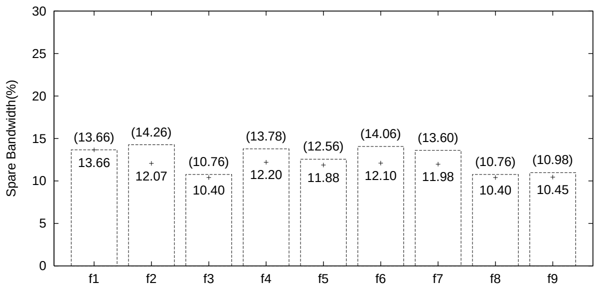


(a) Deterministic multiplexing

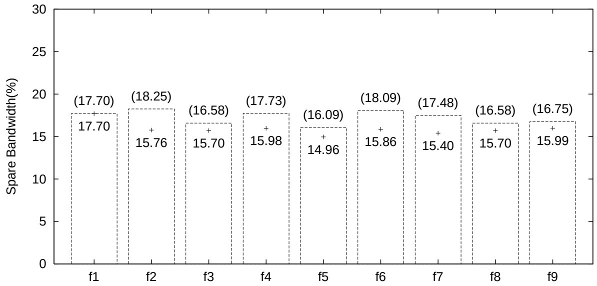


(b) Probabilistic multiplexing

Figure 6: Case 3 (pl = 30%, bl = N/A)



(a) Deterministic multiplexing



(b) Probabilistic multiplexing

Figure 7: Case 4 (pl = 25%, bl = 30%)

routing space of the unwrapped mesh, the overall performance of backup multiplexing is poor and the impact of routing heuristics was not substantial. However, when we injected other inhomogeneous conditions into Case 4, the discrepancy between routing heuristics emerged.

(iii) While the impact of iterative optimization was not substantial in the heuristics which use f_3 as a primary cost function, it was significant in some other heuristics. For instance, after the optimization, the performance of f_2, which yielded a similar level to f_1 before the optimization, was improved up to 20%. The large improvement by iterative optimization implies that the performance of the corresponding heuristic is sensitive to the order of channel establishments. The minor impact of optimization on f_3-contained heuristics hints that with those heuristics, we can set the interval of periodic optimization to a large value.[8]

(iv) Both deterministic and probabilistic multiplexing significantly reduce spare resources, regardless of the network conditions and routing heuristics. As explained in Section 3, the deterministic method requires smaller resources than the probabilistic method under the deterministic failure scenario, which was employed in our simulation. To illustrate the efficiency of these multiplexing methods, we compare their performance with that of a simple *brute-force* multiplexing method, in which the same amount of spare resources is reserved for all links without considering the network condition. In Figure 8, the smallest possible spare resources by brute-force multiplexing is compared with those by our multiplexing methods.

[8]Yet, we need to start the optimization algorithm immediately after the repair of faulty components.

7 Conclusion

We proposed a scheme for dependable real-time communication, by equipping each real-time channel with a pre-determined backup route along which spare resources are reserved in advance for fast failure recovery. To reduce the amount of such spare resources, we developed backup multiplexing methods for resource sharing among unrelated backup channels. We also

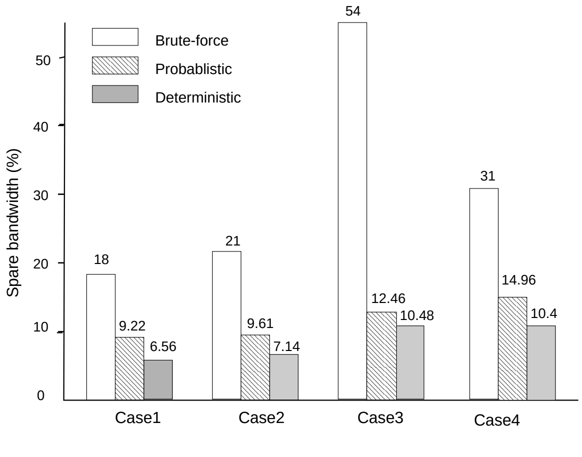


Figure 8: Comparison with brute-force multiplexing

developed heuristics for backup routing. We then evaluated via simulation the effectiveness of the proposed scheme under various network conditions. The simulation results have shown the routing heuristics which directly exploit the property of backup multiplexing to outperform other more general heuristics like minimum-hop routing or load-balancing routing by a significant margin. With a proper routing heuristic, backup multiplexing enables very efficient resource utilization. For instance, only as low as 20–30% resources for active channels are needed to be reserved as spare resources to guarantee fast recovery from any single link failures in a torus, while more than 120% spare resources are necessary without 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] A. Banerjea, "Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels," in *Proc. ACM SIGCOMM*, pp. 194–205, August 1996.

[4] A. Banerjea, C. Parris, and D. Ferrari, "Recovering guaranteed performance service connections from single and multiple faults," Technical Report TR-93-066, UC Berkeley, 1993.

[5] Q. Zheng and K. G. Shin, "Fault–tolerant real–time communication in distributed computing systems," in *Proc. IEEE FTCS*, pp. 86 – 93, 1992.

[6] W. Grover, "The selfhealing network: A fast distributed restoration technique for networks using digital crossconnect machines," in *Proc. IEEE GLOBECOM*, pp. 1090–1095, 1987.

[7] C. Yang and S. Hasegawa, "FITNESS: Failure immunization technology for network service survivability," in *Proc. IEEE GLOBECOM*, pp. 1549–1554, 1988.

[8] J. Baker, "A distributed link restoration algorithm with robust preplanning," in *Proc. IEEE GLOBECOM*, pp. 306–311, 1991.

[9] R. Kawamura, K. Sato, and I. Tokizawa, "Self-healing ATM networks based on virtual path concept," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, pp. 120–127, January 1994.

[10] J. Anderson, B. Doshi, S. Dravida, and P. Harshavadhana, "Fast restoration of ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, pp. 128–138, January 1994.

[11] K. Murakami and H. Kim, "Near-optimal virtual path routing for survivable ATM networks," in *Proc. IEEE INFOCOM*, pp. 208–215, 1994.

[12] R. Iraschko, M. MacGregor, and W. Grover, "Optimal capacity placement for path restoration in mesh survivable networks," in *Proc. IEEE ICC*, pp. 1568–1574, 1996.

[13] 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.

[14] S. Han and K. G. Shin, "Fast restoration of real-time communication service from component failures in multi-hop networks," in *Proc. ACM SIGCOMM*, pp. 77–88, 1997.

[15] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 122–139, January 1994.

[16] A. K. J. Parekh, *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*, PhD thesis, MIT, February 1992.

[17] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet-switched networks," *ACM Trans. Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.

[18] D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 10, pp. 1044–1056, October 1994.

[19] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-8, no. 3, pp. 368–379, April 1990.

[20] C. Parris and D. Ferrari, "A dynamic connection management scheme for guaranteed performance services in packet-switching integrated services networks," Technical Report TR-93-005, UC Berkeley, 1993.

[21] Z. Whang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[22] R. Vogel, R. Herrtwich, W. Kalfa, H. Wittig, and L. Wolf, "QoS-Based routing of multimedia streams in computer networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1235–1244, September 1996.

[23] D. Kandlur and K. G. Shin, "Traffic routing for multicomputer networks with virtual cut-through capability," *IEEE Trans. Computers*, vol. 41, no. 10, pp. 1257–1270, October 1992.

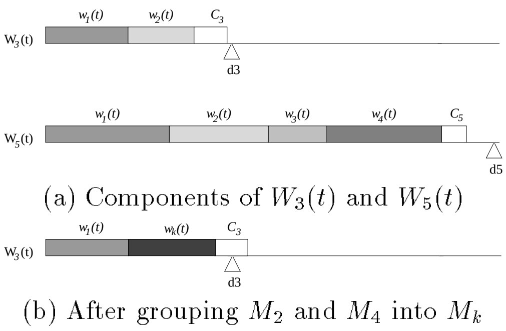
[24] J. Whalen and J. Kenney, "Finding maximal link disjoint paths in a multigraph," in *Proc. IEEE GLOBECOM*, 1990.

[25] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proc. ACM SIGCOMM*, pp. 43–51, 1991.

Appendix A: Inapplicability Proof

To apply deterministic multiplexing, resources should be exchangeable between real-time channels. More specifically, the underlying real-time channel scheme should satisfy the following condition: *On a link, multiple channels can be grouped into a single channel and they can be separated into independent channels again if needed.* We will prove that this condition is not satisfied in scheduler-based schemes, by showing a counter example in the RTC scheme [18], a scheduler-based scheme.

In the RTC scheme, a real-time channel M_i on a link ℓ is described by a set of parameters (C_i, d_i, p_i), where C_i is the maximum service time on ℓ for any message of channel M_i, d_i is the maximum permissible delay for messages belonging to M_i at this link, and



(a) Components of $W_3(t)$ and $W_5(t)$

(b) After grouping M_2 and M_4 into M_k

Figure 9: A counter example

p_i is the minimum message inter-arrival time. When a set of channels $\{M_i = (C_i, d_i, p_i), i = 1, \cdots, m\}$ run on a common link ℓ and the channel set is ordered according to channel priorities with M_1 being the highest-priority, the time required to process a message of M_i in the presence of higher-priority messages is:

$$W_i(t) \;=\; \sum_{j=1}^{i-1} C_j \cdot \lceil t/p_j \rceil + C_i.$$

Suppose at $t = 0$ the message arrival of a channel coincides with message arrivals of all other channels of equal or higher priority (i.e., the worst case for M_i). Then, the message deadline of M_i will always be met if $W_i(t) \leq d_i$, where t is the instant of M_i's message arrival.

Assume, for example, there are five channels $M_1, \cdots, M_5$ on ℓ and we want to group M_2 and M_4 into a new channel M_k. M_1's guarantee will not be affected since the new channel M_k need not come before M_1 in the channel-priority list. However, the grouping can be critical to M_3 and M_5. The relations between the system-time requirement and the delay requirement of M_3 and M_5 are illustrated in Figure 9(a), where $w_i(t) = C_i \cdot \lceil t/p_i \rceil$. The priority of M_k should be higher than M_3, since the guarantee for M_2 cannot be achieved regardless of the bandwidth reserved for the channel with lower priority than M_3. The bandwidth reserved for M_k should be greater than that for M_2 and that for M_4. If the bandwidth requirement for M_2 is greater than that for M_4, simply using the parameter set of M_2 as that of M_k will work fine. But in the opposite case, $w_k(t)$ which should be located at the position of M_2 becomes greater than $w_2(t)$ as shown in Figure 9(b). Hence, the QoS guarantee for M_3 will be violated as a result of grouping M_2 and M_4 into M_k.

Thus, there is no general way to find a parameter set which subsumes any of two parameter sets, while preserving the guarantees for other channels. A similar argument can be applied for other scheduler-based schemes.

Appendix B : IP Formulation

The notation used in the IP formulation for the optimal reconfiguration of a backup channel under deterministic multiplexing is as follows:

$\hat{L}$: the set of l links.

c_i: the bandwidth capacity of link i.

s_i: the spare bandwidth on link i.

a_i: the bandwidth for primary channels on link i.

r_m: the bandwidth requirement of primary channel m.

$\hat{M}$: the set of all primary channels.

$\hat{E_m}$: the set of E_m eligible paths for the backup channel of primary channel m.

$\hat{P_j}$: the set of P_j primary channels which run over link j.

$X_{k,m}$: 1 if k-th path in $\hat{E_m}$ is selected as the backup path of primary channel m, and 0 otherwise.

$Y^i_{k,m}$: 1 if k-th path in $\hat{E_m}$ contains link i, and 0 otherwise.

$Z_{m,j}$: 1 if primary channel m contains link j, and 0 otherwise.

The information about primary channels is given. Thus, $\hat{P_j}$, a_i, and $Z_{m,j}$ are given, and $\hat{E_m}$ and $Y^i_{k,m}$ can also be derived from the network topology. Since c_i and r_m are constants, the only variables in this IP formulation are s_i and $X_{k,m}$, essential for backup route selection. All values are non-negative integers.

Our goal is to minimize the sum of spare resources at all links, so the objective function is $Min\left\{\sum_{i=1}^{l} s_i\right\}$.

The constraint set that should be satisfied to tolerate all single link failures is:

$$s_i \geq 0, s_i + a_i \leq c_i, \quad \forall i \in \hat{L},$$

$$\sum_{k=1}^{E_m} Z_{m,j} \cdot X_{k,m} = 1, \quad \forall j \in \hat{L}, \forall m \in \hat{M},$$

$$s_i - \sum_{m=1}^{P_j} \sum_{k=1}^{E_m} X_{k,m} Y^i_{k,m} \cdot r_m \geq 0, \quad \forall i \in \hat{L}, \forall j \in \hat{L}, i \neq j.$$

The first constraint is straightforward. The second constraint indicates the property that only one backup path should be selected among $\hat{E_m}$ for each primary channel m, which is disconnected by the failure of link j. When all possible failure scenarios are considered, all primary channels will fail at least once, so the number of equalities needed to specify this constraint will be equal to the number of primary channels in the network. Hence, this constraint can be rewritten as $\sum_{k=1}^{E_m} X_{k,m} = 1, \quad \forall m \in \hat{M}$.

The third constraint represents the property that the spare resource on link i should be sufficient to meet the resource demands of backup activation caused by the failure of link j. The number of inequalities resulting from this constraint is proportional to $(l \cdot P_j \cdot E_m)$.