Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels

Anindo Banerjea
The Tenet Group
University of California,
Berkeley CA 94720.

Abstract

The paper presents a simulation study of the use of dispersity routing to provide fault tolerance on top of a connection oriented realtime service such as that provided by the Tenet scheme. A framework to study the dispersity schemes is presented. The simulations show that the dispersity schemes, by dividing the connection's traffic among multiple paths in the network, have a beneficent effect on the capacity of the network. Thus, for certain classes of dispersity schemes, we obtain a small improvement in fault tolerance as well as an improvement in the number of connections that the network can support. For other classes of dispersity schemes, greater improvement in service may be purchased at the cost of decrease in capacity. The paper explores the tradeoffs available through exhaustive simulations. We conclude that dispersity routing is a flexible approach to increasing the fault tolerance of realtime connections, which can provide a range of improvements in service with a corresponding range of costs.

1. Introduction

This paper presents a simulation study of dispersity schemes to provide fault tolerance on top of an underlying connection oriented realtime service. The simulation experiments explore the tradeoffs involving the improvements in service provided and the associated cost in terms of decreased capacity of the network to support realtime connections.

In [1], we presented the idea of providing dispersity routing at the application layer, on top of a connection oriented realtime communication service. While the idea of dispersity routing has been around for many years [2], the traditional uses of dispersity routing have been at the physical layer of network communication, using hardware to perform the necessary encoding and decoding. However, advances in workstation speeds and encoding techniques [3] have made it possible to perform the necessary computation in software at high enough speeds to move this function up to the application layer. [1] discussed the benefits of this approach, such as a cheaper and more flexible service, as well as some of the difficulties introduced by this approach, such as the need to perform the route computation for the dispersity systems on a per session basis.

The basic idea behind the dispersity schemes is to make reservations on multiple paths through the network so that in the event of failure the available bandwidth of the system is not reduced to zero. A range of fault tolerance services can be offered based on this idea. For example, Forward Error Correction (FEC) techniques can be used in conjunction with multi-path reservation to provide a transparently fault tolerant service. While this approach would not be appropriate for a bandwidth-poor environment such as the current Internet, it may be a feasible solution in the Gigabit-per-second networks envisioned in the future. Moreover, the improvement in the service provided may allow some applications to use a shared network, which would otherwise require a private network.

The target applications for a transparently fault tolerant realtime communication service include remote surgery, space missions, industrial control, and so on. However, other interactive multimedia applications may also benefit from fault tolerance, if the associated costs, in terms of increased resource requirement, are low enough. The ability to put many different classes of applications, including those that require some degree of fault tolerance, on the same network makes it possible to reduce the cost of service by exploiting economies of scale.

This research was performed while the author was a Ph.D. student at the University of California, Berkeley, and supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Digital Equipment Corporation, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under several MICRO grants, and the International Computer Science Institute. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations. The author is now at Philips Research Laboratories, 1070 Arastradero Road, Palo Alto, CA 94304. Email: banerjea@prpa.philips.com

The dispersity schemes work on top of a connection oriented realtime communication service, such as that defined by the Tenet scheme [4]. The underlying network provides a mechanism for per-session resource reservation, based on the traffic specifications and performance requirements of the network clients, in order to establish connections with end-to-end performance guarantees through the network. The network models of [5-8] and ATM networks with QoS support also meet the assumptions of this paper. Since it seems likely that a large fraction of the future communication infrastructure will be ATM based, this work is relevant to a consideration of how fault tolerance may be economically provided in such an environment.

The resource reservation provided by the underlying realtime service control loss due to congestion and buffer overflow. That still leaves two sources of packet loss in the network: transmission errors and network failures, which are not covered by the guarantees provided by the Tenet scheme. The primary objective of the dispersity mechanisms are to handle network failures. However, tolerance to errors is also provided as a side-benefit of using FEC to control loss due to faults.

This paper extends the work of [1] by presenting a simulation study, which explores the service provided by the dispersity schemes, and the cost to the network in terms of resources used. In particular, we are interested in the reduction in the network's capacity to support realtime channels. Section 2 presents a framework to classify the dispersity schemes. Section 3 then describes the simulator design and the setup of the experiments. Section 4 presents the simulation results. Section 5 describes some related work. Section 6 summarizes the results and concludes the paper.

2. Framework

In this section, we will present a framework for our experiments by classifying fault tolerance schemes along four dimensions. This framework was developed in [1] but is summarized here for completeness. More details, such as the equations to compute the performance parameters of the sub-channels or the buffer requirement at the destination, can be found in [1]. The schemes for fault tolerance dealt with in this paper can be characterized by three variables: dispersity, redundancy, and disjointness. Finally, the schemes with redundancy may be hot or cold.

Dispersity

Dispersity is the idea, proposed by Maxemchuk in [2], of sending the information across a number of paths (N) in the network. In a realtime network, we also make reservations on each of the paths to guarantee performance on the dispersity system. The information to be sent is divided uniformly across the paths, either by fragmenting each message, or by sending them round-robin on the paths. The advantages of spreading the information out on N paths are:

- The load on any one specific path is smaller and the effect of bursts is spread out over the network.
- In the event of a network failure, the transmission capacity of the aggregate system is only partially affected.
- If the message is fragmented, transmission time is reduced to approximately $1/N^{th}$ of its single-path value. This is less important in high-speed networks.

Note that this system is not transparently fault-tolerant. It merely reduces the effect of the failure on the client. [1] describes some video coding schemes that allow the application to continue without interruption, with a reduced quality of service, in the presence of network failures. For example, a JPEG video stream, partitioned into multiple streams such that all the data for a frame follows the same path, but separate frames are sent on different paths, would have this property. If one of the paths fails, every N^{th} frame would be lost, but the resulting video stream would be usable.

The number of paths chosen, N, is one of the variables that characterize a dispersity system.

Redundancy

Redundancy is the idea of sending more information than the message, in order to be able to reconstruct the message in the event of loss in the network. In a dispersity scheme, we can send the redundant information along a separate path from the rest of the data. For example, of the N paths, only K may carry the message stream. Thus, for a given message, the system could break it into K equal sub-messages and transmit them on the K paths. The rest of the paths may be used to transmit redundant information, which would be used to reconstruct the original message in the event of loss of some of the K original pieces of the message. A simple redundant system can be designed with $K = N - 1$. In this case the single redundancy sub-channel carries a bitwise parity calculated over the $N - 1$ pieces of the message. If any one of the sub-messages is lost, the destination can compute its value from the remaining sub-messages and the parity sub-message. Error correction codes which work for arbitrary N and K exist [9]. In the case of maximum distance separable codes, if any $N - K$ of the sub-messages are received, the message can be recovered.

The variable K in relation to N defines the degree of redundancy. $K = 1$ is corresponds to duplicating the same information on N channels, uses N times the bandwidth of a non-fault-tolerant realtime channel, and has the largest fault tolerance. $K = N$ corresponds to a dispersity system without any redundancy.

A dispersity system with redundancy requires N/K times the bandwidth of a non-fault-tolerant realtime channel with the same traffic and performance requirements. When redundancy is used in combination with dispersity, we get the following additional advantages as compared to a dispersity system without redundancy:

- The system is tolerant to transmission errors. A certain number of the pieces of the message can be corrupted or lost without affecting the decoding of the message. The number depends on N, K, and the error-correcting code. It can be no larger than $N - K$.
- The system is transparently fault-tolerant. A certain fraction of the paths can be affected by failure, without interrupting the flow of the information. Again, the specifics of the error-correcting code determine the number of failures that can be tolerated.

Moreover, since the service of the underlying realtime channels is realtime, we obtain performance bounds on the service provided by the dispersity system. The application sees fault tolerant realtime service, with guarantees on packet-delivery that continue to hold in the presence of a restricted number of faults. The restriction on the number of faults covered depend on the level of redundancy of the system, and the nature of the FEC code used.

Disjointness

In general, the routing algorithm in the network must be able to recognize channels belonging to a dispersity system, and place them on disjoint paths. If the paths are not disjoint, the failures of the paths are no longer independent, since, if a shared link fails, two or more paths can simultaneously stop transmitting data. However, disjointness is a very stringent restriction, especially as N approaches the degree of edge connectivity of the network topology. By allowing some links to be shared, we might be able to set up many more connections in the network. We would like to answer the question: can any of the advantages of dispersity routing still be provided after relaxing the disjointness criterion?

If the constraint is completely relaxed, then it is possible for a link to be shared by all the paths in a dispersity system, leading to the undesirable characteristic that if that link fails, the capacity of the system is reduced to zero. Therefore, we must put in some constraint, which, while looser than the strict disjointness constraint, should still allow the dispersity system to have provably good tolerance characteristics. If we allow a link to be used by at most two channel routes, then we know that a single failure will not affect more than two of the paths. Then, by imposing the restriction $N - K \geq 2$, the system can continue to be tolerant to single faults, assuming maximum distance separable codes. Of course, increasing the degree of redundancy increases the overhead, and therefore decreases the capacity of the network to support traffic. We look at the costs and benefits of such methods below. In terms of our characterization, we can define a variable S, which places a limit on how many paths can share a link.

The routing algorithm in the network must take the variable S into account while routing channels which belong to a dispersity system. Of course, even if the system allows links to be shared (i.e., $S > 1$), the routing algorithm should try to find paths which do not share links, and only use paths with shared links if no disjoint paths meeting the delay constraint exist.

Thus, a dispersity system can be characterized by the triple (N, K, S). This triple also tells us how fault-tolerant the system can possibly be. If we use a maximum distance separable code, the system can tolerate up to $\left\lfloor \frac{N-K}{S} \right\rfloor$ faults in the network transparently. For example, the $(5, 3, 2)$ system can tolerate one fault with a maximum distance separable error-correcting code.

Hot vs. cold standby

In addition to the above three variables, a dispersity system with redundancy $(N > K)$ can be run in a hot or cold standby mode. In case of hot standby mode, the extra sub-channels carry FEC information during normal operation. In the event of packet loss or failure, the destination can recover without any exchange of messages with the source. In the case of cold standby, the extra sub-channels are not used, except in the event of a failure. This extra capacity can be used to transmit non-realtime traffic during normal usage, with the understanding that, in the event of failure, the capacity will be appropriated for use by the fault-tolerant realtime connection. When the failure occurs, we incur an extra delay before recovery due to the need to inform the source of the failure, so that it can shift the transmission to the back-up sub-channels. However, the capacity is guaranteed to be there, unlike what happens in the reactive fault recovery mechanisms described in [10], where the recovery may fail due to existing reservations in the network.

3. Simulator design

The simulator used for the experiments is based on a simulator for realtime channels written at the University of California, Berkeley. The network performs resource reservation and packet scheduling based on parameters provided by the client to meet the requirements of all accepted channels. The simulator models the connection management as well as the data transfer phase of realtime communication.

This simulator was modified to support the simulation of failure recovery and fault tolerance. When a failure is simulated, the failure recovery module simulates the connection management necessary to reroute the affected channels [10].

The simulator also supports fault tolerant channels, based on the ideas presented in the preceding section. When a fault tolerant connection is requested by the client, the network computes the traffic and performance parameters for the sub-channels, based on the parameters of the requested dispersity system and using the equations presented in [1]. It then routes the sub-channels belonging to a dispersity system on paths subject to the disjointness constraints specified in the request.

Routing for the sub-channels of a dispersity system is performed one by one. The first sub-channel is routed as a normal realtime channel. The additional constraints of disjointness are added for each subsequent route calculation. The changes to the routing algorithm were implemented as a set of preprocessing steps on the graph representing the network before the routing algorithm (modified Bellman-Ford[1]) was run. To enforce disjoint paths, the links corresponding to the paths which should not be used are removed from the graph by setting the delay on the path to infinity. To allow links to be shared S times, we maintain a variable U_e associated with each link e, which is the number of times the link e has been used by sub-channels in the dispersity system. When $U_e \geq S$, the link e is removed from the graph by setting the delay to infinity as before. If $0 < U_e < S$, we want the routing algorithm to find paths including e only if other paths do not exist. We replace each such link by n links, each with a delay of $1/n$ times the delay on the original link. Since the algorithm returns the shortest path which meets the delay constraints, it finds paths which do not use the shared links before it find paths including them, up to a difference in path lengths which can be tuned by changing n. We do not change the end-to-end delay on any path by this replacement process; thus, we are guaranteed to find all paths eventually.

[1] We use this algorithm to minimize the resource usage while meeting the delay constraints. The details are given in [11].

The simulator models the behavior of the dispersity system during data transfer at the level of detail of individual packets. Thus, the encoding and decoding operations, which occur at the bit-level, are not simulated. The transmission of a message on the dispersity system is modeled as the transmission of packets on each sub-channel of the dispersity system, but the bit level contents of the packets are not simulated. The arrival of the appropriate number of packets from the same message at the destination is interpreted as a successfully decoded message, and the statistics are updated accordingly. Encoding and decoding times are not modeled, since they are dependent on the coding algorithms and the available hardware.

4. Simulation results

The experiments performed using the simulator fall into two categories. The first set of experiments are intended to verify the level of fault tolerance provided by the dispersity systems. Since the reservations are proactive, and the guarantees provided on the sub-channels are mathematical and deterministic in the absence of losses in the network, we expect the service guarantees to hold, given that the assumptions about the fault(s) and other losses are satisfied. The simulations serve to confirm our expectation, as well as to demonstrate the workability of the systems to a fair level of detail. In the presence of network losses due to transmission errors and faults, which are not covered by the Tenet guarantees, the dispersity systems can provide lower packet loss than the basic realtime channel; this performance is experimentally evaluated in the first part of this section.

The second category of experiments performed deal with the impact of the dispersity systems on network capacity. The bandwidth used an (N, K, S) system is approximately N/K times the bandwidth required by an equivalent non-fault-tolerant realtime channel. In simulation, we measured the number of instances of each type of system that can be established in the network under varying load conditions. This provides us with a better understanding of the more subtle issues, such as the effect on the network capacity of splitting the resource requirements among N sub-channels on different paths and the probability of finding resources on N disjoint paths.

The topology being simulated, which we call the "core" topology, is shown in Figure 1, with an example $(3, 2, 1)$ dispersity system shown on it. The choice of topology is driven by the requirement that it must have multiple disjoint paths (up to five in our experiments) between nodes representing end stations, in order to allow the dispersity systems to be set up. In addition, the size of the topology that can be simulated is limited by its effect on the time and

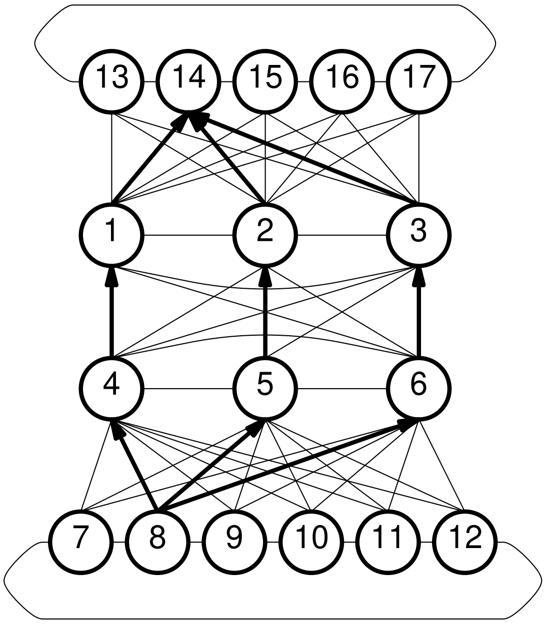

Figure 1: $(3, 2, 1)$ **dispersity system simulated**

memory requirements of the simulation. This specific topology was created based on the following rationale, the central 6-clique composed of nodes 1 through 6 represents a ‘‘core’’ wide area network with a high degree of connectivity, such as that provided by a telecommunications service provider. There is a link with 10 Mbps capacity and 10 ms latency between each pair of these nodes. The other nodes represent end systems on customer premises, which are connected to multiple switches in the core network (with 5 Mbps, 10 ms links) to provide redundancy. They are also connected among themselves into geographic groups (perhaps by a MAN) for additional redundancy. Each of these link (such as the one between nodes 7 and 8) have a capacity of 5 Mbps and a latency of 10 Mbps. The specific numeric results from the simulation are only applicable to this particular topology, and general results should be extrapolated to other networks with great care, and only if the other networks also have multiple similar length paths between endpoints.

The current Internet topology was not used for these experiments because it does not have sufficient degree of connectivity. However, we believe that the integrated services data network of the future must have a degree of redundancy comparable to the phone networks of today (rather than comparable to the Internet backbone of today) in order to continue to provide service in the presence of faults. Networks offering commercial service should be able to reroute most of the affected connections in the event of a fault, in order not to lose the revenue that these connections are earning, as well as not lose customers to competitors. Of course, rerouting can only be performed if the network has sufficient spare capacity on alternate paths. Therefore, some applications may need to provide a higher degree of fault tolerance using pro-active mechanisms such as dispersity routing. Since redundancy must exist in the network to support rerouting, we should expose it to the application layer[2], in order to allow the application to provide higher degrees of fault tolerance, when it is needed. [1] presents some of these arguments, as well as issues regarding the appropriate application-network interface, in more detail.

4.1. Performance results

The first set of experiments consisted of setting up one connection using the dispersity system under consideration, in a network carrying different levels of background load, and simulating the service provided in the presence and absence of faults and losses due to transmission errors in the network. Losses due to transmission errors are simulated by dropping each packet with a given probability on each link.

All the experiments described in this section were performed for all of the dispersity systems. We will not present all the results for each separate dispersity system. Rather, we will use a running example, to present some specific results. The general conclusions presented below from the specific experiment are applicable to the entire set of dispersity systems, unless otherwise noted.

The example system is a $(3, 2, 1)$ dispersity system, established from node 8 to node 14 in the network depicted in Figure 1. The traffic parameters for the fault-tolerant dispersity system were a mean packet interarrival time of 5 ms, a peak arrival rate equal to the mean rate, and a packet size limit of 10000 bits. This corresponds approximately to a 2 Mbps JPEG compressed stream. The end-to-end delay requirement is seventy milliseconds. The minimum propagation and transmission delay is thirty-five milliseconds. The buffer requirement at the destination for this set up was 15,000 bytes[3].

The simulation was performed for zero and one fault in the network, at various probabilities of packet loss on the links, and various extraneous loads in the network. The net-

[2] We are not suggesting that the network topology be exposed to the application layer. Rather an appropriate interface should exist between the application and the network to specify the disjointness and dispersity requirements, and the network should be responsible for the routing.

[3] See [1] for an analysis of the buffer requirement and the factors it depends on.

work load was created by establishing varying numbers of simple realtime channels in the network, before starting the observation of the fault-tolerant system. The measurement was performed by allowing the experiment to run for 39 seconds of simulated time. This took an hour or more of actual time, depending on the speed of the machine running the simulation and the other load on the machine, because of the level of detail being simulated. The simulator kept track of the number of packets dropped, the number of messages decoded, the buffers used, and so on. When a fault was simulated, all packets on the failed link were lost. The fault recovery process described [10] was activated, and the affected channels rerouted. Unlike the experiments in [10], data transmission across a link could also fail randomly with a probability determined by a parameter of the experiment, simulating losses due to transmission errors. At simulated time equal to 39 seconds, the measurements were written out to a file.

The general results that apply to all the experiments performed, but for which no graphs need to be shown, are described first. The service provided met the realtime guarantees in all cases. No packets were delivered after the delay bound. In the absence of losses due to transmission errors, no messages were lost for zero and one link failures. These observations were not affected by the level of the extra realtime load on the network, since the guarantees on the sub-channels hold irrespective of the other traffic.

In the experiments in which failure was simulated, one of the links in the central core was removed from service. This caused the fault recovery process to be triggered. The fault message reached the source in under 20 ms, because of the topology and proximity of the failed link to all sources. This is a measure of the time for notifying the source to perform switching for the cold standby sources, and is dependent on the topology. The fault recovery process also rerouted the failed sub-channel to an alternate path. In all our experiments, the reroute was successfully completed.

Figure 2 shows the effect of the loss rate due to transmission errors in the network on the loss of messages as seen by the application using the dispersity system. The two digits shown over each bar represent the tuple (N, K) for the dispersity system represented by the bar. The variable S is 1 for all the systems shown here. Only the systems with $N - K = 1$ and the $(1, 1)$ system (the simple realtime channel without any fault tolerance) are shown. On the x-axis, we have the probability of losing a packet on any link. Note that the graph shows loss rates in the network up to a 10^{-2} probability of losing a packet during each transmission on every link in the network. These are not meant to be representative of transmission error rates in real networks. The actual error rates seen in fiber-optic transmission systems are much lower. We simulated high transmission error rates so that we could observe a significant number of lost packets and its effect on the dispersity system. The height of each bar (along the y-axis) denotes the number of messages lost by the dispersity system due to lost packets. There are five bars for each loss rate, but up to a loss rate of 10^{-3}, most of them are of close to zero height. The number of packets lost on the sub-channels is larger, since a message is lost only if sufficient packets are lost to defeat the redundancy in the system.

The graph for packet loss when a network failure occurs is not shown because it is very similar to Figure 2. The losses for the $(1, 1)$ system increase, but all the others remain the same. The loss is controlled by two mechanisms. Firstly, the duration of the outage of a sub-channel is limited by the recovery process that reroutes the faulty sub-channel onto a new path. Secondly, the redundant coding provided (for systems with $N > K$) controls the number of lost messages during the outage.

Overall, in this set of experiments we see that, at the level of detail that our simulator models a realtime network, and subject to the limitations in the faults being simulated, the service provided by the dispersity systems meets our expectations, even in the presence of single link faults and reasonable transmission error rates. The realtime guarantees are never violated, and the error and fault tolerance is enhanced. With the error rates seen in current transmission systems, any of these dispersity systems will provide adequate protection against single failures. Thus, the choice of which one to use is dictated in part by the question of the degree of tolerance required, that is, whether tolerance against a single fault, two faults, or n-faults is required. The degree of tolerance is determined by $N - K$, assuming maximum distance separable codes. Between systems with the same value of $N - K$, the simple analysis based on the amount of extra bandwidth used ($\frac{N-K}{K}$) seems to suggest that the higher the values of N (and K), the lower the impact on the rest of the network. This issue of network capacity was further explored in the second set of experiments with the simulator.

4.2. Network capacity

The effect of the different dispersity systems on network capacity was tested by seeing how many connections of each type could be established starting with a given initial load. The initial load is determined by the set of simple realtime channels already established in the network before the experiment starts. The experiment is conducted for each type of dispersity system, for four initial load conditions.

For each value of load, we repeat the experiment for three sets of channels generated from different random seeds, which have the same load index[4] in the "core" network. The results are averaged from these three experiments. The number of connections that can be established is determined by attempting to set up a larger number than can be possibly successful, and then counting the number of successful establishments. The source and destination are chosen from opposite sides of the "core" topology randomly. The traffic and performance parameters of the dispersity systems are the same as in the last section.

The effect of the dispersity schemes without redundancy ($N = K$ systems) is explored first. As explained is Section 2, the service provided by such systems degrades partially in the event of a fault, until the failed sub-channel is rerouted. During this time, a lower bandwidth connection is maintained. The total bandwidth requirement of the dispersity system is the same as that of the non-fault-tolerant realtime channel; thus, one might expect that the same number of total connections can be established for each dispersity system as for the non-fault-tolerant realtime channel. However, the simple model based on bandwidth does not take into account the external fragmentation due to the size of the bandwidth requirement of the channel relative to the capacity of the link, and the probabilities of finding multiple paths meeting the delay constraints and having adequate resources.

Figure 3 shows the number of connections of each type which could be set up, for the dispersity systems without redundancy ($N = K$ systems), at different realtime load levels in the network. The non-fault-tolerant realtime channel ($(1, 1)$ system) is shown for reference. We note that, at all load levels, the number of connections that can be set up first increases and then decreases with N. The initial increase is due to the fact that each sub-channel in a dispersity system has a smaller-bandwidth requirement than the original request, and the requirement gets smaller with increasing N. Thus, the level of external fragmentation on the links decreases, since smaller bandwidth channels can be packed more efficiently on the links. This effect increases with the load on the network, since the remaining capacity of the network is smaller compared to the size of a single request, leading to more external fragmentation. This effect would be more pronounced in a network with small capacity, and less pronounced in networks where the capacity is much larger than the requirement of a single realtime channel.

The fact that the bandwidth requirement of each individual sub-channel is smaller than the total requirement of the dispersity system also has an important effect on the capacity of the network to accept subsequent realtime requests until the dispersity system is torn down. Since the amount of resources used on each path is lower, the effect of the request is spread out over the entire network, so more capacity remains on each individual path as compared to a realtime channel that places all its resource requirement on a single path. The amount of actual traffic from the dispersity system on each path is also lower than the traffic on the single path of an equivalent realtime channel. Thus, the dispersity system is friendly to other users, both realtime and non-realtime, of the network.

The subsequent decrease in number of connections established with increasing N is due to the limited number of disjoint routes in the network. The maximum number of link-disjoint paths between nodes on different sides of the "core" topology is five. Thus, as N approaches five, the number of alternative paths for the set of routes decreases. At $N = 5$, we need to find resources on all the available paths; if any path is too highly loaded, the request fails. The ability to establish a large number of fault-tolerant connections, and the probability of being able to establish one in a highly loaded network, decreases as N approaches the number of alternate paths available. For $N > 5$, no connections can be established in this network. Thus, the practicality of any dispersity scheme with large N is limited by the network topology.

The number of connections that can be established starting from a lightly loaded network is a measure of the capacity of the network with respect to the specific dispersity system. Thus, in this topology, the number of $(5, 5)$ connections that can be established is a little more than half the number of non-fault-tolerant realtime channels that can be established. As explained before, this is because N is at the limit imposed by the number of disjoint paths available in the network. On the other hand, we can actually establish more $(3, 3)$ connections than non-fault-tolerant realtime channels, because at this stage the effect of reduced external fragmentation is more important.

The number of connections that can be established in a *heavily loaded network* is indicative of the relative probability of being successful in establishing a connection using a specific dispersity system in a similarly loaded network. In other words, we can interpret the sizes of the bars at load=905 to mean that we can expect to establish a $(4, 4)$ connection with greater probability of success than even a non-fault-tolerant realtime channel if the network is heavily

[4] The load index is measured using the Queuing Delay Index, which allows us to represent bandwidth, schedulability, number of channels and number of links traversed by each channel with a single index in the context of the Rate Controlled Static Priority scheduler. This index was first presented in [10].

loaded. This result is surprising, since it counters our initial intuition that finding resources on four paths successfully, especially when there are only five available, should be harder than finding resources on just one. However, we should interpret the result to mean that at the high load level, because of the small amount of resources left on each link, the influence of external fragmentation is more important than the effect of the disjoint path restriction.

Figure 4 shows the change in behavior when we lift the strict disjoint-paths constraint and set $S = 2$. This allows the routing algorithm to return paths that share links between at most two sub-channels. In the event of a single fault, depending on which link failed, up to two of the sub-channels may fail. However, while the degradation of service will be more severe than for an $S = 1$ system, some communication will still continue if $N > 2$. The degraded service will be better for larger values of N. The graph shows that the negative effect of increasing N on the capacity of the network to support dispersity systems is greatly reduced, so that even the $N = 5$ system can establish as many connections as the non-fault-tolerant realtime channel in a lightly loaded network. At higher loads, the dispersity systems establish far more connections than the non-fault-tolerant realtime channel.

Figure 5 shows various dispersity systems with one redundant channel ($N - K = 1$ systems); the non-fault-tolerant realtime channel ($(1, 1)$ system) is shown for reference. We see that adding redundancy reduces the capacity of the network to support these systems, since they add some overhead to the network. This effect interacts with the improvement due to reduced external fragmentation and the effect of the disjoint-paths requirement. The dominant effect at low load and with small N is that of the bandwidth overhead. Thus, the $(2, 1)$ system can establish half the number of connections as the non-fault-tolerant realtime scheme, and the $(3, 2)$ system roughly two thirds of the number. As N approaches five, the effect of the connectivity of the topology starts being felt, so that the number of connections which can be established for the $(4, 3)$ system is less than three fourths. For $N = 5$, the effect of the network connectivity becomes large enough to drive the number of connections for the $(5, 4)$ systems below that of the $(4, 3)$ system. However, as we increase the load, the effect of reduced external fragmentation becomes more important, since the remaining capacity on some links becomes comparable to the bandwidth requirements of the channels. At extremely high loads, we can actually establish as many or slightly more $(5, 4)$ connections as a $(1, 1)$ system. At this stage, the effect of reduced fragmentation, caused by splitting the requests into smaller sub-channels, dominates. Note that we do not get the benefit of reduced fragmentation for the $(2, 1)$ system at all, since the sub-channels have the same bandwidth requirements as the original. Thus, for this system the total capacity requirement determines the number of channels that can be established at all load levels.

The conclusions we can draw from the above observations are: the effect of the increased bandwidth is partially offset by splitting a request into smaller sub-channels; this process also decreases the effect the scheme has on other traffic, and on future realtime requests; the advantage of splitting the request increases with the realtime load on the network, so that, in a highly loaded network, the dispersity systems can be set up as easily as a non-fault-tolerant realtime channel; and the disjoint-path constraint becomes significant as N approaches the number of available disjoint paths, reducing the number of dispersity systems with large N that can be established in a network, as well as making it more difficult to set up a dispersity system with large N in a highly loaded network.

Figure 6 shows the impact of allowing $S = 2$ on the ability of the network to accept dispersity systems with $N - K = 1$. We notice that the effect of the network connectivity is reduced, so that, for low loads, the number of connections that can be successfully established is determined primarily by bandwidth considerations. At higher loads, the reduced fragmentation caused by the smaller bandwidth requirements of the sub-channels allows many more connections to be set up for the dispersity systems. In fact, the $(4, 3)$ and the $(5, 4)$ systems are comparable to the non-fault-tolerant realtime channel at low loads, and better at high loads, in terms of the number of connections established. Unfortunately, if a shared link fails, the service provided by such systems is interrupted, so the fault tolerance in these systems only applies to failures of the unshared links in the system.

Figure 7 shows the effect of increasing the level of redundancy. The dispersity systems shown have $N - K = 2$; the $(1, 1)$ system is again shown for reference. We note that now the effect of the increased capacity dominates, so that, even at very high loads, fewer dispersity connections can be set up than non-fault-tolerant realtime channels. The main reason is that the sub-channels are now as large as the original channel, so we get no decrease in fragmentation. The effect of the disjoint-paths constraint continues to be visible. When we modify the constraint, by setting $S = 2$ (Fig. 8), we get no improvement for the $(4, 2)$ system. The explanation for this behavior is that the reduced fragmentation effect is not applicable, since each sub-channel has the same bandwidth requirements as the original request. Thus, the only factor is the bandwidth requirement, which restricts the number of $(4, 2)$ connections to roughly half the number of basic realtime connections. For the $(5,3)$ system, the fragmentation effect does play a small role, so that, at higher loads, the number of $(5,3)$ connections is comparable to the number of non-fault-tolerant realtime channels. *If* the network had a higher degree of edge connectivity, higher values of N would yield better capacity characteristics for $(N, N-2, 2)$ systems.

The fault tolerance characteristics of the $(N, N-2, 2)$ systems are comparable to or better than those of the $(N, N-1, 1)$ systems. Thus, if one were attempting to establish a $(4,3,1)$ connection, and the network failed to find a fourth disjoint path, one of the options available to the end-system would be to set $S = 2$, and attempt to establish a $(4,2)$ or a $(5,3)$ system. Comparing the number of connections that can be established in Figure 8 and Figure 9, we see that the $(5,3,2)$ system may indeed be a viable alternative to the $(5,4,1)$ or the $(4,3,1)$ systems, since the number of connections that could be established under similar load conditions are roughly equal.

5. Related work

The idea of dispersity routing was first presented in [2]. This work used statistical analysis, under Poisson arrival and steady state assumptions, to compute the reduction in transmission times and the load balancing effect of dispersity routing. [12] studied the use of dispersity routing for a medical image retrieval application on a circuit switched network, assuming on demand call placement with retries and a statistical limit on the tolerable delay, using similar analytical techniques. [13] studied queue buildup analytically assuming bulk arrival processes and the loss behaviour through simulation with on-off sources. Most of the assumptions if these works, such as statistical arrival processes, are not applicable in our network model.

In [1], we developed a framework for the study of fault tolerance schemes for realtime channels based on dispersity routing, and analyzed the benefits and costs which could be expected from such systems. We also compared our approach to the work in progress in the Integrated Services and ReSerVation Protocol (RSVP) Working Groups of the Internet Engineering Task Force (IETF) to extend the Internet service model to include realtime services [14, 15]. While this work is in progress, and as such no study of the failure recovery characteristics of this approach exists, the long time intervals associated with the route update protocols and the reservation refresh process (due to the requirements of stability and scalability) lead us to the conclusion that the recovery times of this approach would be significantly slower than the recovery times of explicit connection rerouting described in [16]. In addition, the connectionless routing paradigm[5] of the Internet makes it difficult to apply dispersity routing to provide fault tolerant communication [17]. On the other hand, the Internet approach leads to a very scalable design for a large and heterogeneous network. It seems likely that future networks will have both paradigms coexisting, such as ATM backbones connecting together non-ATM as well as ATM LANs, and the Internet protocols providing end-to-end connectivity. In such a network, dispersity routing could be used to improve the fault tolerance and capacity of the ATM backbone, but the fault tolerance provided would stop at the switches to the non-ATM LANs. The "core" topology of Figure 1 fits this model; nodes seven through seventeen represent switches connected to LANs, and we only simulated the ATM portion of the network.

Another approach to connection oriented fault tolerant realtime networks is the Single Failure Immune (SFI) [18] channel. SFI channels are also extensions to the basic realtime channels, but they only protect against single failures, and use more resources and require more support from the underlying realtime network than our approach. Isolated Failure Immune (IFI) [19] channels protect against multiple failures, but are only usable in special network topologies (e.g., hexagonal mesh network). In addition, they still use more resources and require more network support than our approach. Moreover, these approaches are not flexible, since they each only offer a single level of fault tolerance at a fixed cost.

Much research has been done in the area of of erasure and error correcting codes to combat packet loss or bit errors in computer networks [3, 9, 20, 21]. [1] discusses some of the relevant ones in more detail. Most works in the area of fault tolerance for computer networks focus on the issue of maintaining connectivity, not of maintaining performance bounds, in the face of network failures. This work is the first to present a simulation study of the tradeoffs involved in applying dispersity routing to improve the fault tolerance of realtime communication networks.

[5] One possible way to use dispersity routing in the Internet is to use source routing, where the source specifies the entire path through the network. This is not a general solution, because it requires the application to know the entire network topology in order to find the routes. In contrast, the dispersity routing schemes require the network to perform the route computation, with an appropriate interface to allow the application to specify its fault-tolerance requirements to the network [1]. An additional argument against the use of source routing to provide dispersity on the Internet is that current routers do not handle source routed packets efficiently, because source routing is an Internet Protocol (IP) option, which is not implemented on the fast path of commercial routers. This causes the forwarding of source routed packets to be significantly more expensive that the forwarding of regular IP packets.

Property	$(N, N, 1)$	$(N, N-1, 1)$	$(N, K, 1) \mid K < N-1$
B/W Overhead	None	$1/(N-1)$	$\frac{N-K}{K}$
Impact on network capacity	Increased capacity	Slight decrease	More decrease
Level of fault tolerance	None	1 fault	$N-K$ faults
Level of error tolerance	None	1	$N-K$
Duration of disruption	Recovery time	No disruption	No disruption
Service during disruption	Lower B/W realtime	No disruption	No disruption
Routing constraints	Easy for small N, hard for large N	Easy for small N, hard for large N	Easy for small N, hard for large N
Encoding/decoding complexity	None	Low	High

Table 1: Properties of various dispersity systems

Property	$(N, N-1, 2)$	$(N, K, 2) \mid K < N-1$	$(2, 1)$ **hot standby**	$(2, 1)$ **cold standby**
B/W Overhead	$1/(N-1)$	$\frac{N-K}{K}$	1	0
Impact on network capacity	Small decrease	Small decrease	Large decrease	No impact
Level of fault tolerance	Partial	$N-K-1$ faults	1 fault	1 fault
Level of error tolerance	1	$N-K$	1	0
Duration of disruption	Recovery time	No disruption	No disruption	Notification time
Service during disruption	No service	No disruption	No disruption	No service
Routing constraints	Easy	Easy	Easy	Easy
Encoding/decoding complexity	Low	High	None	None

Table 2: Properties of some more dispersity systems

6. Conclusions

In this paper, we have proposed dispersity routing as a mechanism to provide fault tolerance to realtime communication networks. We presented a framework to classify the various schemes. The schemes discussed include dispersity systems with various levels of dispersity (N), various levels of redundancy (K), and various levels of strictness of the disjoint-path constraint (S), and hot/cold standby systems. We presented a simulation model, which we used to study the performance of the dispersity systems and the capacity of the realtime network needed to support them. The properties of some of the schemes are summarized in Tables 1 and 2.

In conclusion, we have shown that dispersity routing is a very general method for improving the failure and loss tolerance of realtime channels. It can provide a very large variety of services: from transparent tolerance to graceful degradation of service; from instantaneous recovery to recovery within one round trip for message exchange; and from tolerance to single restricted failures[6] to complete tolerance to $N-K$ faults. The cost of the system, in terms of the network bandwidth needed, depends on the level of redundancy provided, but is ameliorated by the effect of spreading the resource usage among multiple paths, and by the reduced external fragmentation of the link capacity. Most of the network support required is already provided by realtime protocols such as those of the Tenet Suite.

7. Bibliography

[1] A. Banerjea, ‘‘A taxonomy of dispersity routing schemes for fault tolerant realtime channels’’, *European Conference on Multimedia Applications, Services and Techniques (ECMAST’96)*, Louvain-la-Neuve, Belgium, May 1996.

[6] Restricted to failure of the unshared links in an $(N, N-1, 2)$ system.

[2] N. F. Maxemchuk, ‘‘Dispersity Routing’’, *Proceedings of ICC’75*, San Francisco, California, June 1975, 41.10-41.13.

[3] A. Albanese, J. Bloemer, J. Edmonds and M. Luby, ‘‘Priority Encoding Transmission’’, *35th Annual Symposium on Foundations of Computer Science*, 1994.

[4] A. Banerjea, D. Ferrari, B. Mah, M. Moran, D. Verma and H. Zhang, ‘‘The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences’’, *IEEE/ACM Transactions on Networking 4*, 1 (February, 1996), 1-10. Also available as Tech. Rep.-94-059, International Computer Science Institute, Berkeley, California November 1994..

[5] I. Cidon, I. Gopal and R. Guerin, ‘‘Bandwidth Management and Congestion Control in PlaNET’’, *IEEE Communications Magazine*, October 1991, 54-64.

[6] D. P. Anderson, R. G. Herrtwich and C. Schaefer, ‘‘SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in Internet’’, Tech. Rep.-90-006, International Computer Science Institute, Berkeley, California, February 1990.

[7] C. Vogt, R. Herrtwich and R. Nagaragan, ‘‘HeiRAT - The Heidelberg Resource and Administration Technique: Design Philosophy and Goals’’, IBM Tech. Rep. 43.9243 , IBM ENC, Heidelberg, Germany, 1992.

[8] J. Hyman and A. Lazar, ‘‘MARS: The Magnet II Real-Time Scheduling algorithm’’, *Proceedings of ACM SIGCOMM’91 Conference*, Zurich, Switzerland, September 1991, 285-293.

[9] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*, North-Holland, New York, 1977.

[10] A. Banerjea, C. Parris and D. Ferrari, ‘‘Recovering Guaranteed Performance Service Connections from Single and Multiple Faults’’, *Proceedings of Globecom’94*, San Francisco, November 1994, 162-168. Also available as Tech. Rep.-93-066, International Computer Science Institute, Berkeley, CA.

[11] C. Parris, *Dynamic Channel Management*, University of California at Berkeley. PhD Thesis.

[12] N. F. Maxemchuk, ‘‘Dispersity Routing in High Speed Networks’’, *Computer Networks and ISDN Systems 25*, 6 (January 1993), 645-661.

[13] Q. Ding and S. C. Liew, ‘‘A Performance Analysis of a Parallel Communications Scheme for ATM Networks’’, *Proceedings of the IEEE GLOBECOMM’95*, Singapore, November, 1995, 898-902.

[14] L. Zhang, R. Braden, D. Estrin, S. Herzog and S. Jamin, *Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*, Internet Draft, May, 1996. available from ftp://ds.internic.net/internet-drafts/draft-ietf-rsvp-spec-12.txt.

[15] C. Partridge and S. Shenker, *Specification of Guaranteed Quality of Service*, Internet Draft, December, 1995. available from ftp://ds.internic.net/internet-drafts/draft-ietf-intserv-guaranteed-svc-03.txt.

[16] A. Banerjea, *Fault Management for Realtime Networks*, University of California at Berkeley, December 1994. Ph.D. Thesis.

[17] S. Chiou and V. O. K. Li, ‘‘Diversity Transmissions in a Communication network with Unreliable Components’’, *Proceedings of ICC’87*, Seattle, Washington, June 1987, 968-973.

[18] Q. Zheng and K. G. Shin, ‘‘Fault-tolerant real-time communication in distributed computing systems’’, *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*, Boston, MA, July 1992.

[19] Q. Zheng and K. G. Shin, ‘‘Establishment of Isolated Failure Immune Real-Time Channels in HARTS’’, *IEEE Transactions on Parallel and Distributed Systems 6*, 2 (Febrary 1995), 113-119.

[20] A. J. McAuley, ‘‘Reliable Broadband Communication Using a Burst Erasure Correcting Code’’, *Proceedings of ACM SIGCOMM ’90*, Philadelhia, PA, September, 1990, 297-306.

[21] E. Ayanoglu, C. I, R. D. Gitlin and J. E. Mazo, ‘‘Diversity coding: Using error control for self healing in communication networks’’, *Proceedings of INFOCOM’90*, San Francisco, California, June 1990, 95-104.

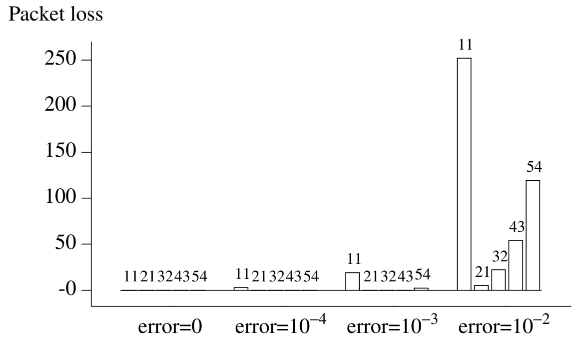

Figure 2: Packet loss vs. loss rate for dispersity systems with N - K = 1, no network failure

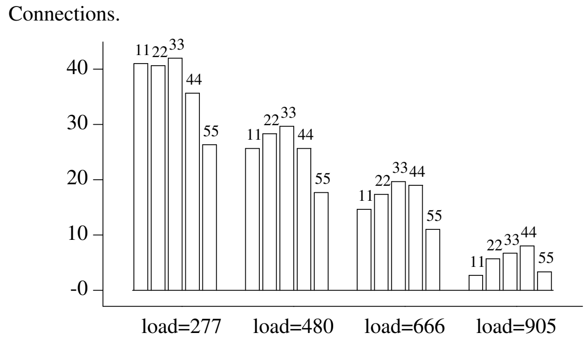

Figure 3: Number of connections established for dispersity systems without redundancy

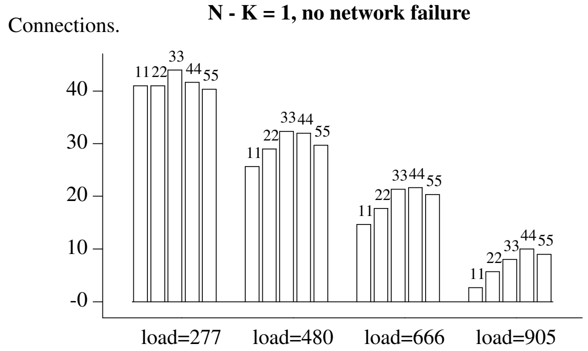

Figure 4: Effect of allowing S = 2 on dispersity systems without redundancy

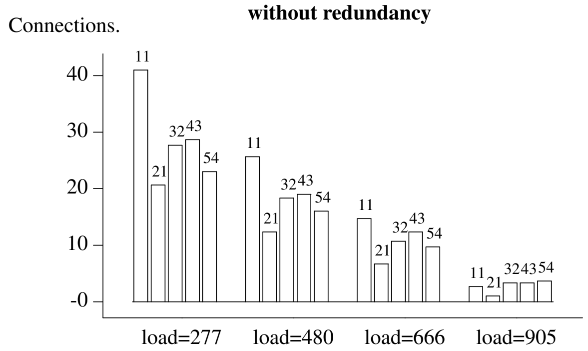

Figure 5: Number of connections established for dispersity systems with N - K = 1

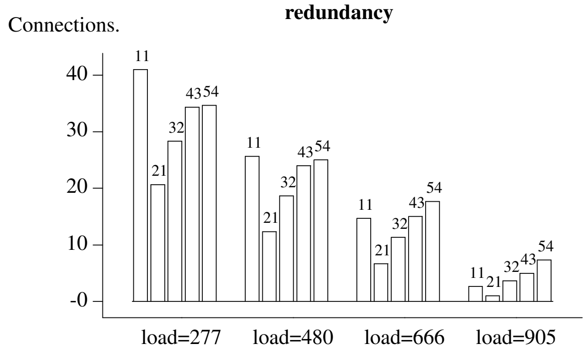

Figure 6: Effect of allowing S = 2 on dispersity systems with N - K = 1

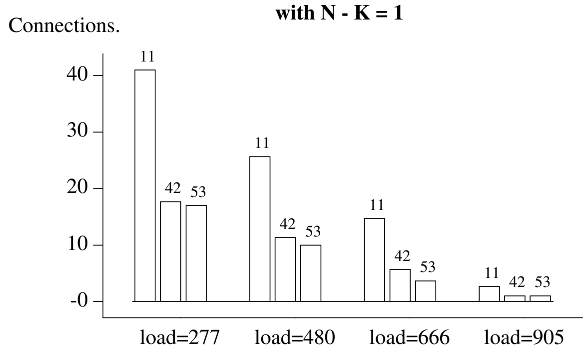

Figure 7: Number of connections established for dispersity systems with N - K = 2

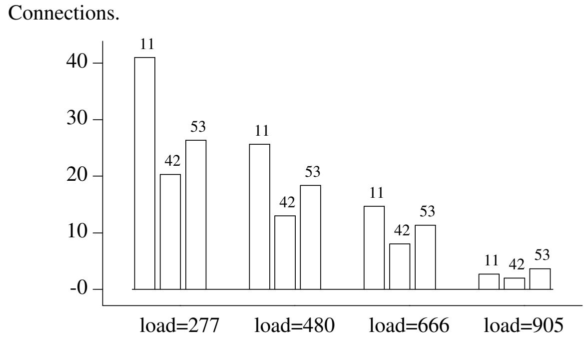

Figure 8: Effect of allowing S = 2 on dispersity systems with N - K = 2

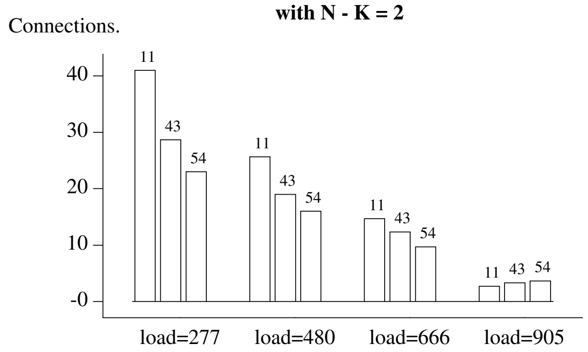

Figure 9: Equivalent dispersity systems with no shared links and N - K = 1