Chapter 7: Conclusions

7.1. Introduction

This thesis started out to answer the question of how to improve the fault-handling characteristics of realtime networks. In this chapter we will summarize the arguments and results presented. Then, we will return to the original requirements spelled out in Chapter 1, to see if the mechanisms proposed in the thesis satisfy them. We will briefly present the main contributions of our work, and then close with a discussion of the issues open for future research.

7.2. Summary of the dissertation

In Chapter 1, we looked at the realtime channel paradigm, noted its promise in the light of the requirements of future broadband integrated-services networks, and felt the need to investigate mechanisms to deal with fault recovery and fault tolerance for realtime networks. Such mechanisms are beneficial, both to the client, who gets improved service, and to the network, which does not lose revenue due to lost traffic. We identified *proactive* mechanisms to be used if a disruption of service is not tolerable to the application and the cost of the extra resources required is a less important issue. *Reactive* schemes are useful for dealing with failures when the cost of providing redundancy is too high in a particular network or when the application can tolerate rare disruptions due to failures. We surveyed some work in designing and evaluating survivable network topologies. We looked at existing fault-recovery techniques, and noted that the assumptions, on which they are based, are not applicable to realtime networks. We also looked at fault-tolerance techniques, and surveyed some forward error correction and dispersity routing strategies that might be useful for realtime channels.

In Chapter 2, we described the Tenet Realtime Protocol Suite 1. We outlined the basic approach, as well as the specific control protocols RCAP and DCM. We also surveyed some other realtime schemes. Chapter 3 extracted the common principles behind a number of these schemes. These basic principles, which are fundamental to realtime networks, may be taken as a starting point for designing the fault-handling mechanisms, so that these mechanisms may be applicable to as wide a set of realtime networks as possible. Thereafter, we described a very general framework for the fault-recovery process. We considered a number of basic design alternatives, and made some fundamental decisions. A further set of choices were made by default, when we chose not to further explore the issues of the fault-detection process, the route update protocol, the channel management protocol, and the algorithm to select the min-hop path from a graph. Finally, we identified a number of issues to explore by simulation.

Chapter 4 described our simulation experiments, by which we made decisions about the issues we had chosen to explore. We also showed that, with the correct choices, schemes within the design space we had chosen to explore could recover a significant portion of the affected traffic within reasonable amounts of time. The schemes were also shown to be tolerant to changes in network load, network topology, traffic mix, and simultaneous failures of two links. In Chapter 5, we put the ideas together into a description of a protocol to reroute realtime channels in the event of a fault, which would work in the realtime network environment provided by the Tenet Protocol Suite 1, including the DCM control protocol.

Finally, in Chapter 6, we examined the problem of fault tolerance, and explored a specific proactive mechanism, namely, dispersity routing. We worked out the details of a scheme to use dispersity routing to provide fault-tolerant realtime communication, simulated the service provided, and explored the effect on network capacity. We concluded that dispersity routing could provide a wide variety of fault-tolerant realtime services when implemented on top of a basic realtime service. Even when redundancy was used to improve the fault tolerance and loss tolerance of the dispersity systems, the network costs were found to be lower than anticipated for many of the schemes, because the effect was spread out over a number of paths in the network. Implementation at the higher levels of the protocol stack is possible if adequate network support is provided. The necessary network support was also described in Chapter 6.

7.3. Requirements re-examined

We started out in Chapter 1 with a set of requirements for the control mechanisms of a realtime network. The control mechanisms were required to:

- Enhance the ability to deal with failure conditions
- Use resources efficiently in bandwidth-poor networks
- Provide fast control in high-bandwidth wide-area (high delay-bandwidth product) networks
- Provide a range of services: transparent fault tolerance for applications which need it, low disruption times for others
- Work efficiently in the common cases, but deal gracefully with the unlikely occurrences
- Be applicable to a variety of networks, under a variety of conditions

Some of these requirements may conflict. For example, the goal of providing efficient use of resources in a bandwidth-poor environment might lead to increased latency of response, which might be unacceptable in high delay-bandwidth networks. Thus, a range of mechanisms are needed. We looked at a bipartite classification of network control mechanisms, proactive and reactive. The proactive schemes have resources ready to handle worst-case scenarios. Thus, they react faster, but are more wasteful of resources. Reactive schemes involve a latency for reaction, but only use resources when needed, so they are more efficient. Thus, depending on the application requirements, and the resource availability in the network, the correct set of mechanisms may be chosen to satisfy the needed requirements. In the next section, we will describe the mechanisms that were proposed in this thesis, and evaluate the extent to which they satisfy these requirements.

7.4. Contributions

We described two sets of mechanisms in this thesis. The fault-recovery mechanisms developed in the first half of the thesis are designed to satisfy the requirements of efficient operation in the common fault-free case, but restore as many realtime channels as possible in the event of a failure. The fault-tolerance schemes provide a higher level of reliability at increased cost. We will see to what extent they meet the requirements described in the previous section.

The network recovery mechanisms described in Chapters 3, 4, and 5 meet the requirement of efficient use of resources. In the common fault-free case, the network uses no extra resources, as compared to the control mechanisms without fault-handling capability. In the event of failure, the simulation studies performed show that the reroute process uses minimally more resources than would be required in the best possible case (i.e., if each channel were to be routed in an empty network). Furthermore, in the networks of the scale considered, the reaction is fairly rapid, since the average reroute is completed within 100-200 ms depending on the network topology. The mechanisms also deal well with two simultaneous failures, different network topologies, load conditions, and traffic mixes. Only one level of fault-recovery service is provided, since all channels are subjected to the same recovery process.

Thus, the first, second and fifth requirements are satisfied. The requirement of speed is also partially satisfied, since the recovery is fast enough for many applications in the topologies tested, but may not be adequate for wide-area networks, where the latency for reaction would be higher. Of course, the recovery process itself is not realtime, since there is no guarantee that it will succeed at all.

The average recovery time is dominated by the network transmission and propagation times. Thus, the average recovery time would grow approximately proportional to the round-trip time on the realtime channels. Since most of the applications requiring realtime services require reasonably *low* delay bounds, the fault-recovery mechanisms will also perform adequately on any network where a such a service can be offered.

The fault-tolerance mechanisms described in Chapter 6 span a wide range of efficiencies for network resource usage. The systems without redundancy actually increase the network capacity, compared to the basic realtime channels. They also provide graceful degradation of service in the face of faults in the network. They work with a number of loss-tolerant compression schemes, such as JPEG and block compression, as well as with hierarchical compression schemes in conjunction with PET. The systems with redundancy add bandwidth overhead in return for increased fault tolerance. The cold standby schemes allow the spare capacity to be used by best-effort traffic, but suffer some latency to switch to the backup channel when the fault occurs. Thus, the set of mechanisms described allow us to provide a range of services at varying levels of efficiency for resource usage. The service offered is insensitive to the load on the network, since the proactive reservation provides service guarantees. The schemes are valid in any network topology with a high degree of edge connectivity.

Thus, the dispersity schemes satisfy all the requirements. Even the efficient resource use requirement is met, to various extents, by the different systems; the systems without redundancy meet it fully. Together with the recovery protocol, they provide a powerful way of improving the fault tolerance of realtime networks, offering a variety of levels of improvement in service, at different costs in terms of the reduced capacity of the network.

In addition to the actual schemes presented, the thesis also provides a very useful framework to evaluate ideas for the fault recovery of realtime channels. The ideas presented are applicable to any network that satisfies the assumptions of Section 3.3. These are the minimum possible set of assumptions about the realtime network. Further, we attempted to provide convincing arguments to justify the portion of the solution space for recovery mechanisms that we explored in detail. We hope to have convinced the reader that all interesting solutions to the problem should lie within the framework we provided. Thus, our framework can be used to compare and evaluate any further ideas of fault recovery for realtime networks that might be developed in the future. In particular, the load index we developed is useful, since it allows comparisons of a multi-dimensional quantity, the amount of traffic, in one dimension. While this load index is restricted to RCSP scheduling networks, the basic ideas may be applied to other bandwidth reservation and priority schemes, and the load indices developed may also be used for diverse applications such as routing and charging.

Finally, the chapter on fault tolerance presented a comparison of the existing approaches to fault tolerance for realtime channels with the realtime dispersity systems. In this process, we spelled out a set of criteria, which may be used to evaluate and compare other fault tolerance mechanisms for realtime channels, when they will be developed.

7.5. Future work

One very important task left to be accomplished is the implementation of the ideas proposed in this thesis in a realtime networking environment. Arguing about the validity of our ideas from first principles is important, in that we can be more certain that the ideas are worth the time and effort of implementation. However, no amount of simulation or analysis can prove a concept as effectively as implementation.

Unfortunately, the time is not yet mature for these ideas to be tested in an actual implementation. The assumption of our schemes is that a realtime network with high degree of edge-connectivity exists. So far, the Tenet protocols have only been implemented in testbeds, where the topology is too simple to try these ideas out. The only existing DCM implementation does not yet support route modification. No other realtime network implementations with the richness and functionality of the Tenet Suite exist as yet. Thus, the all important last step, implementation of the RTCMP protocol, must wait until the realtime environment is ready to support it. Given the rate at which realtime networking research and deployment is being conducted, this time should not be very far in the future.

Many questions have been raised, elsewhere in this thesis, and left unanswered. Some of them are implementation-related, such as that about whether an implementation of the fault tolerance schemes in user space can be made sufficiently efficient. One may also ask how the recovery and tolerance model fits into the Tenet Suite 2 interface. Others are more basic, such as that about whether multiple priorities for failure recovery can be provided within the recovery model. Can network security be included into the model without major changes to the interfaces proposed? These questions, and many others, are left open for future research.