Chapter 2: The Tenet Realtime Protocol Suite 1

2.1. Introduction

This chapter describes the Tenet Realtime Protocol Suite 1. In this process, it presents many of the ideas of the realtime paradigm by example. Our research into fault management is based upon this paradigm. Therefore, this background information is essential, in order to place the issues involving fault management for realtime networks in the correct context. The simulation evaluation presented in Chapter 4 and the high level protocol design present in Chapter 5 are also based on the Tenet suite.

We first present in Section 2.2 the assumptions about the underlying packet-switched network, on which the Tenet model is based. We also present the service model in Section 2.3, which is based on the idea of a contract between the network and the client. The basic approach to providing realtime guarantees, given the assumptions made, is described Section 2.4. Section 2.5 discusses channel management in further detail, and describes two specific control protocols, designed for the Tenet Suite 1, which provide different levels of network management support. We also describe the support for routing needed for realtime networks. Finally, Section 2.6 surveys some other realtime schemes, either paper designs or implementations, to see if the mechanisms to solve the problems of realtime communication differ vastly. These observations will form the basis of our arguments, presented in the next chapter, that the basic principles of realtime communications are fairly uniform across implementations, and fault recovery mechanisms based on these fundamental principles should be applicable to all such networks.

2.2. Network model and assumptions

Realtime communication can be provided on a general packet/cell switching store-and-forward network if some assumptions are satisfied. The primary assumption is some form of scheduling which supports differential grades of service on the switches or routers (henceforth called nodes) of the network. The first implementation of the Tenet

Suite 1 is based on EDD (Earliest Due Date) scheduling. Guarantees may also be provided based on other scheduling schemes such as RCSP (Rate Controlled Static Priority) (72) and multiple priority versions of FCFS (29); even networks with nodes running different scheduling disciplines can be supported (29).

The nodes of the network must be connected by links that have boundable delays. They might be physical links on which the delay is bounded by the laws of physics. They might be sub-networks on which the delay is bounded by some other mechanism, such as SONET ATM networks. They might also be sub-networks on which the delay is bounded using a hierarchic application of the realtime network protocols (7).

The nodes of the network must also have sufficient computing power and programmability to run an establishment protocol that includes admission control tests of moderate complexity. For switches that do not have such programmability, we assume the presence of a controller which can perform these tests and run the establishment protocol.

For the purposes of this thesis, we impose the additional assumption that the network is simple, i.e., each link in the network is a physical link and does not represent a sub-network. This simplifies the explanation of the protocols significantly. For details of how to extend the realtime paradigm to internetworks, the reader is referred to (29).

2.3. Service model and assumptions

We assume a service model based on a server-client relationship between the network and the application. We use the word *client* to refer loosely to the application or the human user running the application. We assume that the clients of a realtime service will require guarantees about the service provided by the network. The guarantees will cover all performance parameters of interest to the client. The specific parameters that will be included in a contract between the network and the clients depend on the nature of the application, and on what the network is prepared to support. For research

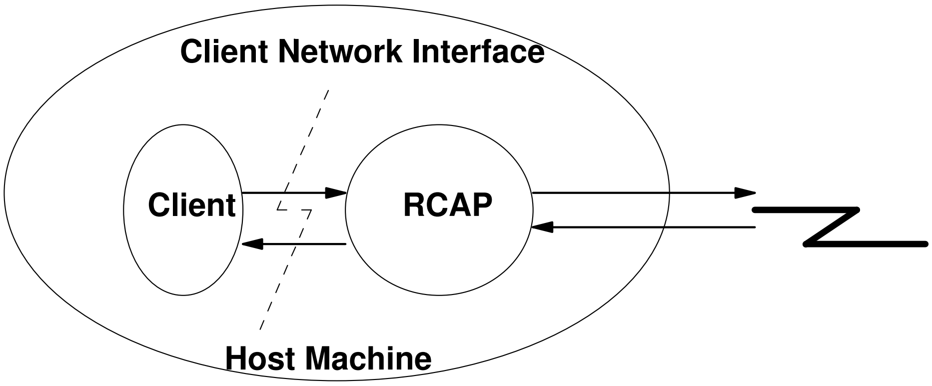

Figure 2.1: Service model for realtime network

purposes, we should try to support as many parameters as possible, and then evaluate the usefulness of each to the client and the cost to the network. The Tenet service model supports guarantees on delay, jitter and packet loss due to buffer overflow (see Table 2.1).

To support any realtime guarantees, the network has to limit the amount of load that the client may place on the network. This takes the form of traffic specifications, which the client declares when asking for performance requirements specified as described above. The exact set of parameters in the traffic description are also under investigation. More complex descriptions allow the network to allocate the resources more efficiently, leading to higher network utilization. However, simpler descriptions are less of a burden to the client (to choose the correct specification) and to the network (to process in order to allocate resources). The traffic specification used in Tenet Suite 1 uses a peak-average rate specification based on four parameters: X_{min}, X_{ave}, I, and S_{max} (26) (see Table 2.2).

The service model assumed involves a transaction between the network and the client. The client presents to the network a request for performance guarantees, and its traffic specification at the same time. The network processes this information based on its current load and resource state, and either accepts or rejects. If the request is accepted,

Parameter	Semantics
D_{max}	Upper bound on end-to-end delay
Z_{min}	Lower bound on probability of timely delivery
J_{max}	Upper bound on end-to-end jitter (optional)
W_{min}	Lower bound on probability of no loss due to buffer overflow

Table 2.1: Performance parameters of the Tenet Suite 1 interface

Parameter	Semantics
X_{min}	Minimum inter-message time
X_{ave}	Minimum average intermessage time
I	Averaging interval
S_{max}	Maximum messsage size

Table 2.2: Traffic parameters of the Tenet Suite 1 interface

the network is committed to delivering within the performance bounds each packet injected into the network, provided the client does not violate the traffic specifications. In the present service model, the performance is guaranteed except in the case of a network fault.

In the absence of sufficient experience with applications of such networks, we assume that any combination of performance requirements and traffic specifications is possible. The parameterized interface makes it possible to handle any combination, so that applications may find their exact requirements. This is especially important for research purposes, since we would like to find out what combinations are used by clients if no restriction is imposed on them.

Finally, we assume that the frequency of requests may be fairly high. We visualize such a realtime network being used to make video-phone calls, set up conference calls, access video servers, listen to digital recordings, and so on. As such, the lifetime of a connection will vary from minutes to hours. If a medium-sized network can support a hundred connections and the average lifetime is five minutes, this implies an average of twenty connections being established and torn down every minute. This frequency will increase with network capacity and size.

2.4. The Tenet Suite 1

The Tenet Suite 1 is a suite of protocols which provide realtime service on a network meeting the assumptions of Section 2.2. The protocols in the Suite are shown in Figure 2.2. The network-layer protocol, the Real-Time Internet Protocol (RTIP), provides a connection-oriented support for realtime communication in an internetworking environment. On top of this protocol, the Realtime Message Transport Protocol (RMTP) provides a message-based transport-layer interface to applications, while the Continuous Media Transport Protocol (CMTP) provides a time-driven interface for periodic traffic. These protocols coexist with the Internet protocol stack as

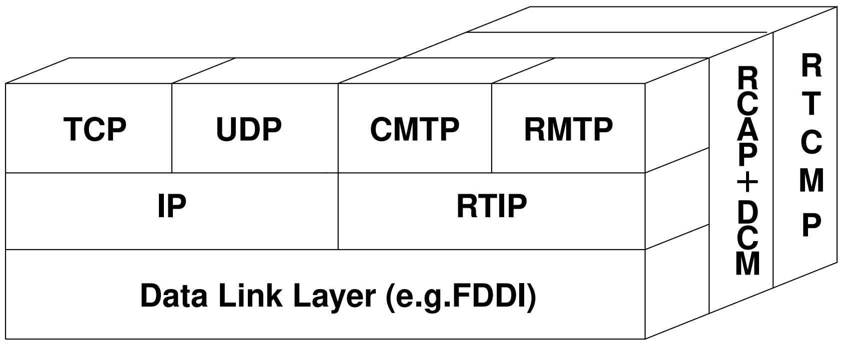

Figure 2.2: Protocols in the Tenet Suite 1

shown in the figure. The Realtime Channel Administration Protocol (RCAP) provides channel management services: channel establishment, teardown, and status enquiry. An extension to RCAP, the Dynamic Channel Management protocol (DCM), adds the ability to modify the parameters of the channel during its lifetime (51-53). To this suite, we plan to add a protocol for fault handling to be named the Real-Time Control Message Protocol (RTCMP).

The Tenet approach is based on the concept of *realtime channel*, which supports the service model described in the previous section by means of resource reservations on the network elements on the route of the channel. The resource reservation mechanism works in conjunction with the data forwarding protocols to ensure that sufficient transmission resources and buffers are available for the packets on the realtime channel as long as the client obeys the traffic specifications associated with the channel. This is made possible by admission control tests performed before each channel is accepted, which ensure that, if the channel is accepted, sufficient resources to meet the performance guarantees of all pre-existing realtime channels, as well as those of the new channel, are available at all nodes on the path of the channel.

The local tests (performed at a single node) ensure that the local performance requirements of the channel can be met without violating the performance guarantees of any pre-existing channel. The inputs to the tests are the local performance requirements and the traffic characteristics of the channel as seen at this node. The nature of the tests depends on the scheduling mechanism used in the data forwarding protocols. The tests for EDD scheduling are described in (27,28). This scheduling algorithm is implemented in the first version of RTIP.

These local performance bounds correspond to some resources reserved at the node on which the bound is given, since they cannot be used to establish any other realtime channel until the resources are released by the teardown of the channel. Since there is a limit on the number of realtime channels that can be supported, accepting a realtime channel implies that the number of subsequent channels that can be supported is reduced. For resources like buffers or bandwidth this reservation is easy to quantify, but the delay bound offered to the channel also constitutes a resource, which is harder to quantify. A lower delay bound corresponds to more resources being "blocked", since fewer low-delay channels can be supported, but the properties of this blocking effect depend on the scheduling algorithm implemented on the node. The control mechanisms must keep track of this resource state at each node, and quantify this "blocking" algorithmically, to ensure that, when resources are unavailable, new channels are not allowed to be set up.

To allow local guarantees to be extended to end-to-end performance guarantees, the traffic characteristics seen at each node must be computable. RTIP makes this easy by ensuring that the traffic specifications given by the client are obeyed at all intermediate nodes. We also need a mechanism to translate an end-to-end requirement to a set of local requirements, as well as a protocol to run all the local tests and make the resource reservations. The connection-oriented paradigm provides an efficient solution to this problem. The tests are run as part of the connection establishment procedure. This procedure is performed by the channel management protocol as described below.

2.5. Channel management

Channel management consists of the tasks of establishment, teardown, and status enquiry. In the Tenet Suite 1, the process of channel establishment is initiated at the source node, which computes a route for the channel based on its knowledge of the network topology and the resource state on each node in the network. In the first implementation, route selection relies on the underlying routing protocol available in the network.

Next, the source sends an establishment request message on this route. This message contains the performance requirements and traffic description specified by the client. Each node runs an algorithm to test the available resources against the requirements of the channel, and to choose the best performance bounds it can support. For example, the lowest delay bounds and the smallest jitter bounds are picked. The resources corresponding to these are tentatively reserved, information about the local reservations are added to the message, and the message is forwarded. When the message reaches the destination, the end-to-end performance achieved can be calculated (for example by adding the local delay bounds or multiplying the loss probability bounds). If the end-to-end performance is unacceptable, or if the destination chooses not to accept the connection, a denial message is returned. Otherwise, the excess performance values can be distributed among the nodes, so that they can relax the resource reservations accordingly. This is done on the reverse pass, when the destination sends an accept message on the path in the reverse direction. When this acceptance message reaches the source, the source can proceed to transmit data, and the network guarantees delivery within the performance bounds specified in the establishment request.

To see why we need to reserve the best amounts of resources on the forward pass and relax these amounts on the reverse pass, imagine a connection that needs to be established across a three-hop path through a network where the last hop is close to saturation. Let us say that the required end-to-end delay is 100 ms, and the links have transmission and propagation delays of 10 ms per link. This gives us a budget of 70 ms to distribute among the hops for queueing delay bounds. If the scheme naively allocates 20 ms per hop and reserves the appropriate resources at the first two hops to match that, the last node might be unable to meet the delay requirement if it is already highly loaded. On the other hand, if we allocate the lowest possible delays on the forward pass, we might end up with allocations of 10 ms, 10 ms, and 40 ms at the three nodes at the end of the forward pass, leaving us with a relaxation budget of 10 ms to distribute among the nodes.

2.5.1. Realtime Channel Administration Protocol

The Realtime Channel Administration Protocol (RCAP) implements channel establishment based on the approach described above. RCAP also performs the tasks of channel teardown and status reporting. During teardown the resources are released so that the blocking probability of future requests is reduced. Status reporting is used to find the resource reservation state of the channel at the intermediate nodes in the network and is useful for management and debugging.

RCAP is implemented as a daemon process that runs on each node in the realtime network. Each daemon is responsible for maintaining the resource state for the links outgoing from the node on which it runs, as well as for local CPU resources and buffers on the node. It maintains TCP/IP connections to neighboring daemons to exchange the

Message Name	Direction	Description
`establish_request`	Downstream	Message sent by source on forward pass to establish a channel; causes admission control tests to be run; best performance resources are reserved.
`establish_accept`	Upstream	Sent from destination to source on reverse path; indicates channel can be accepted with given requirements; causes relaxation of resources and confirmed reservations.
`establish_denied`	Upstream	Indicates that channel was rejected; causes resources to be removed;
`status_request`	Downstream	Requests information about resource reservation and channel state for given channel; proceeds along path of channel to destination.
`status_report`	Upstream	Carries collected information about status back to source.
`close_request_forward`	Downstream	Sent from source to destination to tear down channel; causes resources to be removed.
`close_request_reverse`	Upstream	Sent from destination to source to tear down channel; causes resources to be removed.

Table 2.3: RCAP messages

control messages described in Table 2.3.

The RCAP daemon also accepts messages from local processes on the same machine. The application programs on the source or destination machines communicate with RCAP by making procedure calls to an RCAP library, which constitutes the network control API. Procedures exported by the library enable the source application to request a connection or close it. The destination application is provided with registration, request reception, and request acceptance procedures. The procedure calls are converted by the library to control messages that the RCAP daemon recognizes.

The Tenet Suite 1, using the control protocol described above, has been implemented and deployed in several networking testbeds. Several studies of the performance of the suite as well as experiments of the use of the suite for video conferencing applications have been conducted (8,9,69).

2.5.2. Dynamic Channel Management

An extension to the basic services provided by RCAP is to allow the client or the network to change the parameters of the channel, such as the traffic specifications, the performance constraints, or the route, during the lifetime of the channel, if possible without affecting the flow of data. In (52) the Dynamic Channel Management (DCM) protocol is presented as such an extention. DCM is based on a round-trip approach similar to that used in RCAP; however, it needs to keep track of two sets of parameters, the old channel parameters and the new channel parameters. The forward pass consists of reserving the higher amounts of the resources required to support the two sets of requirements on nodes that are common to the old and new path. This ensures that, during the transition period, no packet misses its deadlines. On nodes that lie only on the new path, the new channel's requirements are used for the reservations. The reverse pass then performs the relaxation and informs the data transmission protocols of the new deadlines. On the common nodes, the higher amounts of the relaxed resources and the old resources are retained. A third pass is performed in the forward direction, along the path of the old channel. During this pass the resources for the old channel are removed from the nodes not on the new path. On the common nodes, the resources corresponding to the new requirements (after relaxation) are retained. The effect of such an alternate establishment is to change the parameters of the channel from the old set of parameters to the new (including possibly changing the route) without affecting the stream of data in transit.

2.5.3. Route update protocol

The route computation performed by the realtime scheme determines the path on which the channel establishment is attempted. For Suite 1, the implementors chose not to explore the routing function, by assuming that the routing protocol of the underlying network would be used. However, for DCM, a more complex routing function was worked out, where the route selection process is based on routing tables that summarize the resource state at all links in the network.

As a new channel is established or torn down in the network, the network state at the nodes in the path of the channel changes. The routing tables on all nodes must be updated to reflect the changed network state. Since the success of a following establishment attempt depends on the accuracy of the routing tables on which the route computation is based, the mechanism to maintain the routing information is important. For this study we assume a simple route update algorithm based on a reliable broadcast of any change to the resource state of each node to all nodes in the network, when a reservation is made permanent during the reverse pass, or when a channel traversing the node is torn down. Thus, each node has a picture of the resource state at all other nodes in the network, except that this picture may be slightly out of date. The further one node is from another, the more out of date its picture of the state of the other node is, since the route update message has further to travel.

During channel establishment, the admission control tests are performed on the actual resource state at each node on the path of the channel. During route computation, the same calculations are performed on the information about the resource state present in the local routing table. If the information were always completely up-to-date, the actual establishment would never fail. The only reasons why an establishment request might fail during the forward pass of the round trip are that either the information in the routing table at the node performing the route selection was out of date or another establishment proceeding concurrently caused the resource state at an intermediate node to change before the request reached the node.

2.5.4. Route selection

The selection of the route is based on the goal of reducing the resource usage in the network, subject to the primary constraint of meeting the delay bounds of the client if at all possible. In the Tenet Suite 1, this process is performed for each channel in isolation, without attempting to optimize the resource usage across the entire network, for reasons of scalability of the solution, and to prevent the requirement that other channels be moved when a new channel is set up. Since our network assumptions include a very dynamic channel state, where the establishment and teardown requests are frequent, the service provider cannot afford to optimize resource usage across the entire network for each channel request, especially since moving existing channels would involve additional channel management activity and may even affect the performance of the channel being moved.

In this context, the goal of minimizing the resource requirements of a channel is approximately equivalent to minimizing the number of links traversed by it, subject, of course, to the delay bound constraint.

A number of graph-theoretic procedures for selecting a route in a graph with various goals and constraints exist. The best known are the Dijkstra algorithm (24), which minimizes the cost of the route, and the Bellman-Ford algorithm (16), which minimizes the number of links in the path, subject to a cost constraint. If we set the cost of a link to the minimum delay bound which can be offered on that link to the channel under consideration, according to the resource state information available in the routing tables, the Bellman-Ford algorithm satisfies the requirement mentioned in the previous paragraph. This algorithm was also used to find the paths for realtime channels in (53), and is summarized below.

This smallest delay bound that can be offered to a channel on a given link can be computed knowing the traffic characteristics of the channel and the resource state of the link, using the same calculations that are performed during the forward pass of channel establishment. The cost of each edge on the graph representing the topology of the network is set to this value. Each node has a label, which is initialized to zero for the source and to infinity for all other nodes. This label represents the cumulative cost of reaching the node from the source given certain constraints. The algorithm works in phases. At the end of phase i, each node is labeled with the smallest cost of all i-hop or shorter paths from the source to the node. The algorithm stops when the destination is reached by a path with a cost less then the desired delay bound, or when i exceeds a given constant value. In the latter case, the algorithm fails to find a path.

2.5.5. Summary of Suite 1

This concludes our brief description of the Tenet Suite 1. We have, of course, focussed on the aspects of the suite that are important to an understanding of the rest of this thesis, and omitted many important and interesting details of the data forwarding protocols. Readers interested in the details of Suite 1 are referred to (9), which also includes a fairly comprehensive bibliography of the Tenet work. In this section, we have focussed on the channel management services, provided in the first implementation of Suite 1 by RCAP. We have also described RCAP extensions, such as DCM, the routing algorithm and the route update protocol, which were not part of the first implementation of Suite 1. Together, they form a networking environment that provides realtime communication with network management functionalities that allow the client or the network operator to make changes to the performance parameters, traffic specifications, and routes of existing channels. This environment is rich enough to support fault recovery and tolerance mechanisms. However, since we would like to design our mechanisms to be applicable to a wide variety of realtime networks, we will now provide a brief survey of other approaches to realtime communication in packet/cell switching networks.

2.6. Other realtime schemes

Variations on the realtime scheme described in this chapter have been suggested. For example, the establishment process may be speeded up by forwarding the establishment request before running the tests at each node; this method works well under the optimistic assumption that the tests will succeed. Data may also be transmitted before the acceptance message returns, with the understanding that, if the connection fails, the data will be lost. In this case, the first packet of data may also carry the establishment request (19).

Alternative approaches to the provision of realtime services on packet/cell switched networks have been proposed. In the predictive service proposed in (20), the network attempts to keep the performance given to the client from changing rapidly, but no guarantees are made. This approach does not meet the definition of realtime as used in this thesis. However, the guaranteed service proposed in the same paper does meet the definition, and the basic model, based on resource reservation during connection establishment, is similar to the model underlying the Tenet scheme. The Session Reservation Protocol (3) also fits the model described above closely. The differences lie in the underlying scheduling mechanisms and the details of the admission control tests.

The Heidelberg Resource Administration Technique (HeiRAT) (63) uses the ST-II network-layer protocol to provide unicast and multicast realtime connections. The ST-II implementation used in this investigation is enhanced with resource reservation algorithms very similar to the Tenet scheme (37). The Magnet-II Realtime Scheduling approach (38) supports a fixed menu of QoS classes. This simplifies the implementation of the admission control tests and the process of mapping end-to-end guarantees to local bounds. However, it limits the flexibility of resource allocation and the services provided.

Restrictive forms of realtime communication are proposed in (35,68), where the throughput is the main performance objective under consideration, and other parameters such as delay and loss rates can be computed secondarily. Resource reservation in the PlaNET network (19) uses the reservation model of (35). All of the above schemes are based on resource reservation during connection establishment.

Finally, there exist approaches to realtime communication which are fundamentally dissimilar from the Tenet model. The predictive service of (20) mentioned before falls into this category, since the performance provided is not guaranteed. The network merely *tries* to prevent the network performance from fluctuating rapidly, but performance is expected to deteriorate as load in the network increases. (70) uses the concept of soft reservations to avoid a completely-connection oriented model. It has certain advantages in the extremely heterogeneous environment of the Internet, but it does not offer true guarantees, since the reservations can be dropped without notice, or the path of data transmission may change during the lifetime of the conversation. (33) also proposes a connectionless approach in which the bandwidth partitioning model can be extended to approach realtime service, but the main goal of the system is not realtime service, but bandwidth partitioning.

2.7. Conclusions

While our experience with the Tenet Suite (8,9) demonstrated the benefits of realtime control in the network, it also highlighted some issues for further research. One key observation from our experiments with the realtime protocols is the need for fault handling mechanisms. This was especially keenly felt because the hardware and software that we were working with were experimental and crashed often in the earlier days of our experimentation. Providing fault recovery in those networks was not feasible, because the topologies did not offer alternate paths between end points. However, such redundancy would be needed in a commercial service network, in order to handle occasional failures without loss of service to clients. This aspect has been very carefully treated to date by network providers, and will have to be so treated as the new ATM based B-ISDN networks develop.

If realtime services are provided in an ATM or packet-switched network, they must follow some of the same principles of resource reservation and admission control that have been dealt with in this chapter. The next chapter isolates the common principles on which realtime service must be based, in order that we may study how to provide fault management services on networks that offer such services. Based on these fundamental principles, we designed fault management techniques that should be applicable to all networks that follow them. These techniques, and their evaluation through simulation, are described in the forthcoming chapters.