A Mechanism for Dynamic Re-routing of Real-time Channels[*]

Colin Parris Hui Zhang
Domenico Ferrari

Computer Science Division
University of California at Berkeley
Berkeley, CA 94720

TR-92-053
August 1992

Abstract

Various solutions have been proposed to provide real-time services (i.e., services with guaranteed performance requirements) in packet-switched networks. These solutions usually require fixed routing and resource reservation for each conversation. The routing and reservation decisions, combined with load fluctuations, introduce the problems of network unavailability and loss of network management flexibility. We believe that these problems can be alleviated by properly balancing the network load. In this paper, we present a mechanism that dynamically reroutes a real-time channel without disruption of service to the clients. This mechanism is one component in a framework to investigate load balancing in a real-time internetwork. We show that the mechanism can be incorporated into the Tenet real-time protocol suite with minimal changes and overhead.

[*]This research was 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, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

1 Introduction

With the advent of high-speed networking, there is an increasing demand for new applications such as the transmission of digital video and audio over packet-switched networks. These applications have stringent real-time performance requirements in terms of throughput, delay, delay jitter and loss rate [2]. New solutions have been proposed to provide real-time services in packet-switched networks. These solutions usually require fixed routing and resource reservation on a per-conversation basis [4, 6, 9]. Resource allocation and routing decisions are made at the beginning of the conversation based on the resource availability and network load at that time, and are kept for the duration of the conversation. However, the resource availability and the network load may change during the lifetime of the connection. The resource allocation and routing decisions made at connection setup time, and the changes in resource availability and network load occurring during the lifetime of the connection may introduce two problems: (1) new requests may be denied due to unbalanced resource allocation (see the example in Section 2), and (2) some network management functionality may become less flexible. The severity of these problems increases in an internetworking environment.

One solution to these problems is, in certain cases, to balance the load in the network by dynamically re-routing existing connections. The challenge lies in the fact that this re-routing should not appreciably impact the quality of service of the existing connections.

This paper presents a mechanism for dynamic re-routing of real-time channels. Section 2 provides the motivation for load balancing, and discusses other uses of this re-routing mechanism. Section 3 gives an overview of our real-time environment and a framework for the investigation of load balancing. A dynamic re-routing mechanism is described in detail in Section 4. A summary and a discussion of future work are given in Section 5.

2 Motivation

This section discusses the motivation behind the use of load balancing and the subsequent design of a dynamic re-routing mechanism.

Load balancing in real-time internetworks is desirable to increase network availability, thereby increasing total revenue and maximizing network utilization; to allow network administrators to reclaim resources; and to reduce the impact of unscheduled, run-time maintenance on clients with real-time services.

In networks offering real-time services, special emphasis must be placed on making these services available to the client. The type of network unavailability that is being referred to here is a by-product of the admission control scheme, which is needed to guarantee service performance. To provide real-time services, resources have to be allocated for use by clients. Efficient routing and resource allocation decisions made on previous real-time clients reduce the probability that a new client's request is refused by the admission control scheme. The more efficient the routing and the resource allocation the greater the number of real-time channel requests that the admissions control scheme can accept. As it is very difficult to predict new client requests, efficiency can be increased by re-routing current connections such that new requests are accommodated.

A simple example is shown in Figure 1. In this example, the guaranteed resource is bandwidth. The maximum bandwidth of each link is 10 Mbps. Along the upper route , Upper_0_3, there is a single client who has reserved a bandwidth of 5 Mbps. There is also a single client who has reserved 5 Mbps of bandwidth along the lower route, Lower_0_3. A third client requests a single channel with with a bandwidth of 10 Mbps. While the network has an aggregate unallocated bandwidth of 10 Mbps, this bandwidth is distributed across two disjoint routes, hence the network is forced to deny the client's request, and the network's peak utilization remains at 50%[1]. With load balancing, the

[1] We assume that multi-path routing is not supported.

single connection along Upper_0_3 can be re-routed along Lower_0_3, thus making the Upper_0_3 route available to the new client. The network peak utilization is now 100%.

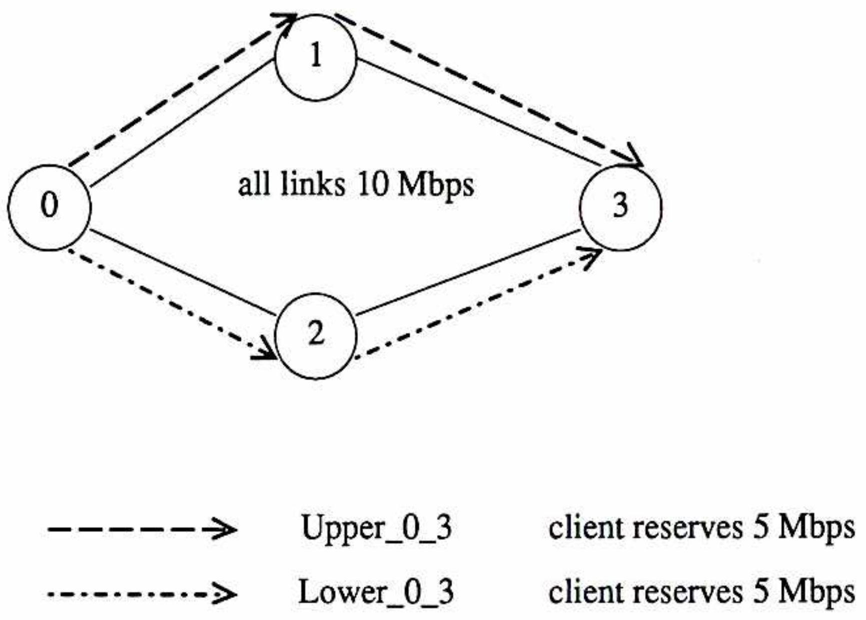

Figure 1: Load Balancing Example

In real-time networks consisting of sub-networks, there will be occasions when sub-network administrators wish to reclaim the resources in their sub-networks. This situation usually arises when the sub-network is a public network, and higher priorities are assigned to specific types of services. An example of this can be seen in long-distance telecommunication carriers, which permit access at minimal cost, usually to universities and other research organization, to specific high-speed trunks for the purpose of research. If there is a remote failure within the carrier network, the carrier reclaims these trunks, and carrier traffic is then re-routed to the trunks. Upon reclamation, real-time clients cannot be dropped (their service contracts are legally binding), and must be transparently re-routed by the load balancing mechanism. Sub-networks of government laboratories will have similar needs, as immediate demands by high-priority government users (within the sub-network or from other legitimate sub-networks) will necessitate reclamation of resources, and require load balancing to accommodate re-routing of real-time traffic.

The third reason for a load balancing scheme is to reduce the impact of unscheduled, run-time maintenance on real-time clients. Unscheduled, run-time maintenance is done when non-catastrophic errors are detected on a component, where the component is either a link, a switch, or an entire sub-network. Although this component has not failed, its error rate has exceeded some predetermined threshold set by network administrators. If this component is not repaired, it may cause a degradation of service quality and possibly a violation of the real-time service contracts. Upon detection of these components, it may be necessary immediately and transparently to re-route the real-time channels that are being impacted. These components can then be corrected, and if desired, the real-time channels can be re-routed along their original routes. In this scenario, the network repairs itself with minimal impact to its real-time clients.

These are the three major justifications for the implementation of a load balancing scheme. It should be obvious that the load balancing mechanism can be used in other contexts, such as the support of fault tolerant services, and the automation of failure recovery techniques. In the first context, some fault tolerant services may provide multiple disjoint channels carrying the same data, with one channel defined as the primary and the others as secondary. The degree of fault tolerance provided is dependent on the number of disjoint secondary channels. Upon a failure of the primary and/or of some secondary channels, the degree of fault tolerance can be automatically and transparently re-established by the use of the load balancing mechanism. In the context of failure recovery, the mechanism can be used to automatically recover by selecting an alternative route that bypasses the failure.

3 Background

In this section we present an overview of our scheme to provide real-time services and a framework for the investigation of load balancing.

3.1 Real-time Channels and the Tenet Real-Time Protocol Suite

A *real-time channel* is a communication abstraction that defines real-time communication services associated with traffic and performance parameters in a packet-switched network [4]. A channel needs to be *established* before data can be transferred. During the channel establishment phase, a communication client specifies its traffic characteristics and end-to-end performance requirements to the network; the network then translates the end-to-end parameters into local parameters at each node, and reserves resources accordingly. During the data transfer phase, by appropriate scheduling and rate control, the local performance requirements are met at each node, and the client specified end-to-end performance guarantees are satisfied.

The Tenet real-time protocol suite implements the real-time channel abstraction in an internetworking environment. The protocol suite decouples control and data delivery functions. There are two control protocols, two transport layer data delivery protocols and one network layer data delivery protocol. For the purposes of this paper, only the two control protocols: the Real-Time Channel Administration Protocol (RCAP) [1], the Real-Time Control Message Protocol (RTCMP) and the network layer data delivery protocol: the Real-Time Internet Protocol (RTIP) [8] are relevant. RCAP is responsible for establishment, tear-down and modification of the real-time channels, while RTCMP is responsible for control and management during data transfers. RTIP provides a host-to-host, simplex, unreliable, sequenced, guaranteed-performance packet service.

At each host and gateway traversed by a channel, there is a locally unique 16-bit integer called *local channel ID* associated with the channel. A real-time channel is uniquely identified by the pair (source host IP address, source local channel ID) throughout the network. This unique identifier is stamped into the header of each RCAP and RTCMP control message and is used to look up the state associated with the channel at each node. For a data RTIP packet, however, only the next node's local channel ID is stamped in the packet header to speed up the state lookup.

3.2 Framework

Our framework consists of the basic mechanisms that are needed to implement a load balancing policy. After identifying these basic mechanisms, we can investigate the effects of a variety of policies.

The mechanisms that comprise the framework are:

- the routing mechanism,
- the performance monitoring mechanism,
- the dynamic re-routing mechanism,
- the load balancing control mechanism.

The routing mechanism implements the routing algorithm which selects a route in adherence to certain routing constraints, e.g., that a primary and its alternative route must be completely disjoint. The performance monitoring mechanism monitors the appropriate network performance indices and reports them to the load balancing control mechanism. A dynamic re-routing mechanism is needed to establish the alternative route and to perform a transparent transition from the primary route to the alternative route. This mechanism is the focus of this paper, and will be described in Section 4. The load balancing control mechanism receives information from the performance monitoring mechanism, and determines whether load balancing can be attempted using the load balancing algorithm defined by the policy. If load balancing can be attempted, the routing mechanism provides an alternative route, and the transition from the primary route to the alternative route is accomplished using the dynamic re-routing mechanism.

With the support of this framework, load balancing policies can be investigated. It should be noted that some of the characteristics of the mechanisms are dependent on the policies under investigation. In investigating load balancing policies, it is useful to have criteria according to which different policies can be defined and compared. Some of these criteria are presented below (this list is by no means exhaustive):

- How many alternative routes are chosen, and with what characteristics ? Are the routes completely or partially disjoint?
- What network performance indices are monitored to determine the load balancing threshold? What should this threshold be? How often is load balancing done?
- What channels are re-routed during any load balancing period?
- What performance parameters and benchmarks are used to assess the effects of the load balancing scheme? What is the overhead due to load balancing?

4 Dynamic Re-routing Mechanism

As discussed in Section 3, we assume that there is a performance monitoring mechanism to detect load changes and a load balancing control mechanism to make the decision to re-route a channel. When channel i is to be re-routed, the source of the channel tries to establish a new channel that has the same traffic and performance parameters and share the same source and destination with channel i, but takes a different route that satisfies the re-routing constraints. We call the new channel a *shadow channel* of channel i. After the shadow channel has been established, the source can switch from channel i to the shadow channel, and start sending packets on it. After waiting for the maximum end-to-end delay time of channel i, the source initiates a teardown message for channel i.

There are four issues to be considered: a) how to distinguish a channel from its shadow channel, b) how to allocate resources if the two channels share some link(s), c) how to switch from the old channel to the shadow channel if they share the same node(s), d) how to ensure that packets are delivered in order, in spite of the re-routing of the channel.

The first issue is how to distinguish a channel from its shadow channel. Since we would like the re-routing to be transparent, we want the shadow channel to appear the same as the old channel to the client. However, we also need to distinguish between a channel and its shadow channel inside the network. As described in section 3, each real time channel has a unique identifier throughout the network. In the Tenet real-time protocol suite, the unique identifier is the pair (source host IP address, source local channel ID). For the re-routing to be transparent to the client, the shadow channel is assigned the same unique identifier as the old channel. To distinguish between the old channel and its shadow channel inside the network, an additional bit, the *shadow bit*, is used. This bit needs to be included in the header of each control message and stored in the channel state at each node. Having such an additional bit effectively expands the unique ID space of real-time channels from the network's point of view, but it does so in a way that is transparent to communication clients. Notice that having one shadow bit would limit the number of channels having the same channel identifier to be at most two at any one time. One implication is that, before the teardown of the old channel, the shadow channel should not be re-routed again. However, this does not prevent a re-routed channel from being re-routed again after the old channel has been torn down.

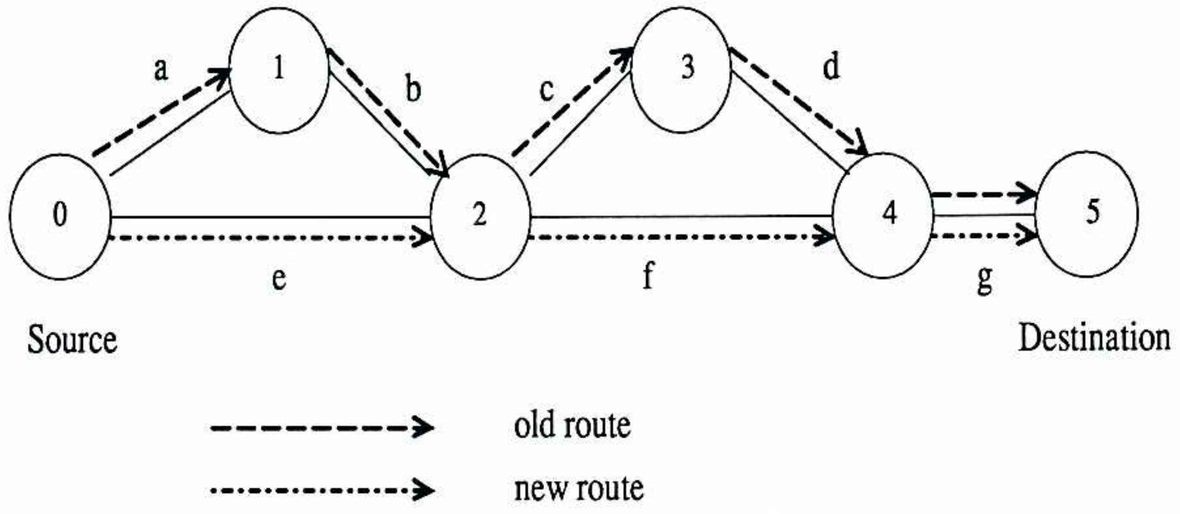

Figure 2: Network With Shadow Route

The second issue deals with resource sharing between a channel and its shadow channel. It is possible that the shadow channel shares part of the route with the old channel (e.g., link g in Figure 2), in which case it may be desirable to let the two channels share resources.[2] This requires the modification of both the establishment and teardown procedures of a new channel: when a channel establishment message arrives at a node, if the requested channel is the shadow channel of a channel traversing the node (i.e., there is already a channel passing through the node with the same unique ID) and sharing the same output link, additional resources are needed only when the shadow channel has more stringent local performance requirements than the existing one; when a channel teardown message arrives at a node, if the channel's shadow channel is also at this node and sharing the same output link, the channel's resources can only be partially released to the extent that the performance requirements for the shadow channel can still be met.

The third issue has to do with the transition from the old channel to the shadow channel when they share the same node but have different output links. Notice that a data packet has only the local channel ID in the packet header. To distinguish between a packet from the old channel and a packet from its shadow channel, we have three alternatives:

- use the same shadow bit technique as discussed above,
- change the forwarding information in the channel state at the appropriate time, or
- choose a different local channel ID at the node for the shadow channel.

To implement the shadow bit technique, we may use either one of the 16 bits in the current local channel ID, or an additional bit. Using one of the current 16 bits is undesirable for two reasons: a) it shrinks the local channel ID space, and b) the unique channel ID, which includes the source local

[2]In the case when network resources are abundant, reserving resources for both channels during the transition period may simplify the design and implementation [5].

channel ID, would not be the same for the old channel and the shadow channel if they have different output links at the source. Using an additional bit is also undesirable because a 17-bit quantity is more difficult to manipulate in a computer than a 16-bit quantity; having extra overhead on a per data packet basis is unacceptable.

It is possible to change the forwarding information in the channel state at each node. However, changes need to be done simultaneously at all the relevant nodes to avoid routing loops and ensure that performance guarantees are met. This is difficult to achieve in a distributed networking environment if the number of nodes shared by the old channel and the shadow channel is greater than 1.

Choosing a different local channel ID is a much cleaner solution: different forwarding information is associated with different states indexed by different local channel ID's; all the packets, either from a normal channel or a re-routed channel, are processed in exactly the same way, and no modification needs to be made to the data delivery protocol to account for the re-routing. There is one complication in adopting this approach: the source local channel ID needs to be the same for the old channel and the shadow channel because it is part of the channel's unique ID.

Our solution is to adopt a combination of the second and third alternatives. At each node except the source, the third alternative is used, i.e., a different local channel ID is chosen for the shadow channel. The source is treated differently to keep the unique ID the same for the shadow channel and the old channel: instead of choosing a different local channel ID for the shadow channel at the source, we adopt the second alternative, i.e., we change the forwarding information in the channel state when the source is switching from the old channel to the shadow channel. Notice that in this case, only the forwarding information at the source needs to be changed, and the problem of synchronizing the changes in all nodes no longer exists.

Although the procedures outlined above can make a smooth transition from the old channel to the shadow channel, there is still a possibility that packets may arrive at the destination out of order. Let the packet numbered `n+1` be the first sent by the source along the shadow channel. It is possible that packet `n+1` arrives at the destination earlier than packet `n`, since they took different routes. Adding a re-sequencing capability at the receiver would solve the problem. However, this may be too costly in a real-time environment to be a feasible solution. Another solution is to bound the minimum delay of a packet along the shadow channel. This can be achieved by controlling delay-jitter [7, 3] in the shadow channel, where delay-jitter is defined to be the maximum difference between the delays experienced by any two packets in a real-time channel. Controlling delay-jitter in a real-time channel essentially imposes a lower bound on the end-to-end delay of any packet. Assume that the end-to-end delay bounds[3] are the same for the old channel and the shadow channel. As long as the delay-jitter bound of the shadow channel is less than the minimum inter-packet spacing of the old and shadow channel, packet `n` will always arrive at the destination before packet `n+1`.

5 Summary and Future Work

In this paper, we have presented a mechanism that dynamically re-routes a real-time channel without disruption of service to the clients. This mechanism is one of the components of a framework needed to implement load balancing in a real-time internetwork. We believe that load balancing in a real-time network is important as it can potentially increase network availability, allow network administrators to reclaim their sub-networks, and immediately reduce the impact of faulty components on real-time clients. This dynamic re-routing mechanism can be incorporated into the Tenet real-time protocol suite with minimal overhead, as shown in the paper. There are still many aspects of the loading balancing problem to be investigated, some of which are listed below.

- The design of the load balancing control mechanism.

[3]In a real-time channel, all the packets delivered are guaranteed to have an end-to-end delay less than or equal to the end-to-end delay bound of the channel[4]

- The choice of the routing algorithm and the design of the routing mechanism.
- The determination of load balancing parameters and benchmark workloads.
- The design, implementation, and analysis of simulation experiments on the load balancing scheme.
- The extension of the re-routing mechanism for use in fault tolerance and error recovery.

References

[1] Anindo Banerjea and Bruce Mah. The real-time channel administration protocol. In *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, 1991.

[2] Domenico Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, 28(11), November 1990.

[3] Domenico Ferrari. Design and applications of a delay jitter control scheme for packet-switching internetworks. In *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, 1991.

[4] Domenico Ferrari and Dinesh Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[5] Amit Gupta, 1992. personal communication.

[6] Claudio Topolcic. Experimental internet stream protocol, version 2 (ST-II), October 1990. RFC 1190.

[7] Dinesh Verma, Hui Zhang, and Domenico Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm'91*, pages 35–46, Chapel Hill, North Carolina, April 1991.

[8] Hui Zhang, Dinesh Verma, and Domenico Ferrari. Design and implementation of the real-time internet protocol. In *IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, Tucson, Arizona, 1992.

[9] Lixia Zhang. *A New Architecture for Packet Switched Network Protocols.* PhD dissertation, Massachusetts Institute of Technology, July 1989.