INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Efficiency Comparison of Real-Time Transport Protocols

Pasquale Di Genova[*] and Giorgio Ventre[†]

TR-95-004

March 1995

Abstract

In this paper we consider the problem of providing efficient network support to distributed real-time applications with different communication requirements. In the case of resource reservation protocols, the level of efficiency of a transport service connection provided by a communication system is influenced by the applications requirements, in terms of amount of network resources needed to provide guaranteed Quality of Service. We consider the Tenet protocol suite, a connection-oriented internetworking set of protocols based upon resource reservation. The suite provides a real-time network service (i.e., a service with guaranteed performance) to two types of applications: continuous media (CM) clients that generate data at regular time intervals (e.g., video and audio); message oriented clients that generate data at arbitrary times (e.g., urgent messages and remote control applications). We compare the performance of the transport protocol for CM clients (CMTP) to that of the transport protocol for message oriented clients (RMTP). In particular, we consider the buffer usage in the underlying real-time internetwork protocol (RTIP). The results of the simulations show that in the CMTP case, by taking advantage of the regular nature of CM clients, proper mechanisms can be adopted to further smooth traffic, so that buffers are used much more efficiently than in the RMTP case.

[*]The Tenet Group, Computer Science Division, Department of EECS, University of California, Berkeley and International Computer Science Institute, Berkeley. E-mail: {digenova,ventre}@icsi.berkeley.edu

[†]Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli "Federico II" Napoli, Italy.

1 Introduction

Under the term *Real-Time Communication* a wide variety of quite different communication requirements is included. They range from those charaterizing applications exchanging isochronous data such as digital video and audio, to those typical of hard real-time applications like control systems and robotics.

The characteristic common to all these applications is the need for a communication service capable of providing a well defined Quality of Service (QoS) in terms of, for example, available bandwidth, end-to-end communication delay and jitter, and data losses. We might define such a service *reliable*, provided that the term *reliability* be redefined, by assigning it a broader meaning that implies the capability of a system to *guarantee* a set of quality-of-service parameters.

However, the different characteristics shown by distributed real-time applications require the availability of a communication service capable of satisfy such diversity in the framework of a guaranteed QoS environment.

Several protocols have been proposed or implemented in the last years, presenting different approaches to the problem of providing guarantees in a communication service[1][13][4] [2].

Proposals adopting resource reservation schemes in addition to rate and admission control mechanisms, appear to be capable to offer a stronger QoS semantic, capable to satisfy even the communication requirements of the most demanding real-time applications.

The major drawback of the resource reservation approach is, of course, a reduced level of network utilization that these protocols present. This problem derives from the need of preventing network users from being jeopardized by other users even when these are transmitting at worst-case conditions. The costs related to factors such as bandwidth allocation, memory buffer and cpu utilization in the nodes of modern, high-bandwidth, network architectures, give enormous importance to the development of techniques for optimal allocation of such resources. We believe that this challenging goal can be achieved by studying and exploiting the charateristics of typical real-time applications, in order to provide them with a communication service tailored to their specific needs.

The use of QoS parameters can assist resource management and help in matching user requirements and traffic characteristics to the network's capabilities. Characteristic QoS parameter values for different traffic streams are shown in table 1 [8].

Traffic Type	MaxDelay(s)	Max Delay Jitter (ms)	Av. Throughput (Mbps)	Acceptable Packet Error Rate
voice	0.25	10	0.064	10e-1
video	0.25	10	100	10e-3
image	1	-	2-10	10e-9

Table 1: Characteristic QoS parameter values for different traffic types.

In particular, this table considers multimedia types. Voice, video and sequenced image display are called continuous media types because traffic is generated at regular time intervals: a voice bit stream of 64kbps is generated by 8 bit encoding of samples taken at intervals of 125 μs; video frames sent at 30 frames per second are distant 33 ms from each other.

Many multimedia applications use these media types[12]. An example is *computer supported collaborative work* (CSCW). The goal of this service is to create a virtual common working environment to favour interactions amongst geographically distributed professionals. Participants in a CSCW application, other than generating and receiving video and audio data, also share distributed editors, spreadsheets and drawing spaces. Another example is the *Virtual Café*. Remote users can interact through a common electronic space where interaction is informal, with users that can freely join or leave the meeting. A third example is a tele-lecture in which a lecturer sends audio, video and digitized slides to students located in remote sites. Students wishing to intervene send audio and video data to both the teacher and the remaining students of the virtual classroom.

In all these examples, to facilitate interaction, the receivers need be kept up to date with participant sending data. For this reason, data need be sent regularly. However, not all real-time applications have this requirement. There are real-time applications that generate data irregularly and in a limited amount. These kind of applications, called message-oriented or message driven applications, require network guarantees in all the cases where data need arrive at destination within a prefixed deadline. Typical message-oriented applications are those performing remote measurement or control, as well as high-performance computing on distributed memory systems [10].

As an example, we consider the case of *telepresence* applications, i.e. applications that enable a user to be virtually present in a remote site at the scope of, for example, measurment reading[9]. An example of this kind of application is teleprogramming where a user remotely controls or manipulates devices such as sensors or robot[7].

Since CM clients provide traffic with a certain regularity, they can characterize their future behavior better than real-time message oriented clients. Thus, CM transport services can use this a priori knowledge of future data transmission timing to smooth traffic and thus more efficiently utilize network and end-system resources such as buffers.

The goal of this study is to compare the performance of a real-time transport protocol suitable suitable for CM applications to that of a real-time transport protocol suitable for message-oriented applications. In particular, we consider as performance index the utilization of buffers in the network nodes.

The results of our study show how buffers can be used more efficiently. This is important since higher efficiency means both an increase in network resource utilization (e.g., buffers) and improved network performance (e.g., end to end delay). If the buffer requirement of a channel is small then many connections can flow through a node without suffering buffer overflow. Consequently, if buffer overflow is avoided then retransmission is unnecessary and a high throughput can be achieved. High throughput can be achieved also for a second reason: limited buffer requirement means smaller queues in the network nodes and thus smaller queueing times.

The paper is organized as follows: section 2 considers transport protocols for real-time applications; section 3 gives an overview of a specific real-time protocol stack (upon which we have developted our simulations) called Tenet; section 4 describes one of its transport portocol suitable for continuous media applications; section 5 explains a rate control scheme that we implemented at such layer to increase buffer efficiency; section 6 shows the simulation scenario we adopted, and highlights the results we obtained from our simulations; finally, we give the conclusion of our study.

2 Transport Protocols for Real-Time Applications.

Why is a specific transport layer protocol needed for continuous media traffic?

Current transport protocols are not well-suited towards serving real-time or CM clients in a heterogeneous network; the introduction of a specific transport protocol for CM clients is an attempt to better satisfy continuous media service requirements.

Transport layer service requirements can be supported through two different approaches. In the first approach, a general transport layer service is offered but options are provided on this service to adapt it to the client's service requirements. This is the approach taken by TCP, which implements a flow control mechanism based upon a sliding window. To make this protocol suitable to different flavours of applications, a dynamic window management scheme has been introduced such that the width of the sliding window is adapted to the actual communication needs [3].

In the second approach multiple transport protocols are designed, each tailored to meet the various transport service requirements of end users. As we have seen in the introduction, real-time applications can be considered as divided in two main classes: CM applications and message-oriented applications. Since these have different requirements, the Tenet approach has been to split the transport layer vertically into two distinct parts and implement a specific protocol for CM clients and a specific protocol for message-oriented clients. The transport protocol for message-oriented applications delivers/receives data to/from the network layer at irregular intervals. For these applications irregular data transfer is sufficient (i.e., transmission is "message driven") but network performance such as end to end delay and throughput need be guaranteed.

Typical applications which require a real-time message transport protocol are:

- Urgent messages, such as distributed or remote process control applications. In these cases, end to end delay need be limited to a few seconds.
- Electronic mail service with guaranteed delivery latency, where end to end delay

need not be necessarily small but must be upperbounded[1].

• Remote control or real-time data acquisition, where reliability and timely delivery of small/medium size messages is crucial for the success of the application.

On the other side, the transport protocol for CM applications delivers/receives data to/from the network layer isochronously conserving the timing characteristic of the sender. Typical continuous media application [12]are:

• Videoconferencing

• Virtual Classroom

• Voice communication

• CSCW applications

• Virtual Café

Furthermore, transport protocols for CM applications provide the following functionalities to implement rate control, that is to control the rate at which packets are sent to the network; enable self-timed data transmission since data is generated isochronously and thus transmisson is time-driven; provide a data stream abstraction, that is a simplex, end to end, continuous, sequenced, periodic transfer of data.

This results in an improvement of buffer usage in the network nodes and a reduction of interactions between sending and receiving client since no flow control is needed. The stream abstraction is more natural for CM clients. If a client requires a full motion video, a full motion stream is created. If a client requires slow motion, a slower stream is set up. If a client requires an audio connection, an audio stream is created. The client only sees these high-level characteristics (full motion video, slow video, audio) of its traffic. He need not to know anything about the network-oriented QoS parameters such as average throughput, maximum delay on which the resource allocation and admission control schemes are based upon at the real-time network protocol layer. It is the duty of the transport protocol translate the different stream types into the characteristic QoS parameter values. Without a stream, if the client wanted to freeze a video frame, it would stop sending packets, but the connection would still be there, perhaps blocking resources, even though it is not being utilized. With the stream abstraction, clients have a better control over use of resources be-

[1]TCP ensures reliable packet delivery but gives no guarantees on delivery latency.

cause in general, one stream is associated with each end-user service requirement (full motion video, slow motion video, freeze frame). By having the stream abstraction at an upper layer (i.e, the transport layer), we can easily translate the CM client requirements into QoS parameters (see fig 1).

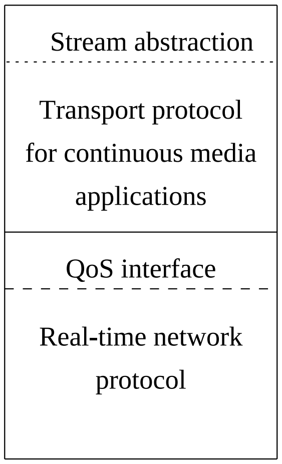
Stream abstraction
Transport protocol for continuous media applications
QoS interface
Real-time network protocol

Figure 1: Stream abstraction provided by the transport layer and QoS interface provided by the network layer.

From this viewpoint we can restate why a specific transport protocol is necessary for CM clients: if we can tailor the transport service to the client's service requirements (and by making CM streams visible at the transport service) then we can specify the user's requirements more naturally and can use more efficiently the network resources allocated to the transport service connection.

3 The Tenet Real-time Protocol Suite

We choose to study the efficiency of network resource usage in a real case scenario. We considered proposed and existing real-time protocol suites that have been studied[1] and implemented[13][4] [2]. Amongst these, we choose to consider the latter because of the availability of an implementation of the suite and because a simulation model was already availabile. Before entering in the details of the simulations, in this section we give an overview of the Tenet protocol suite.

The Tenet suite is a connection-oriented internetworking protocol suite based upon resource reservation. The suite incorporates the concept of *real-time channel*, a simplex unicast end-to-end connection with performance guarantees (e.g., guarantees on end-to-end delay and packet loss probability). The number of real-time channels the network can accept is limited by channel admission control tests which take place before establishing a new channel[5]. A new version of the protocol suite, based on a multicast channel abstraction, is currently under development[6]. The network/client interface model takes the form of a contract between the client and the network. The client promises to obey certain traffic restrictions and the network undertakes to provide a certain pre-agreed level of service.

This service model implies:

1. *Admission control.* The admission of new real-time channels must be controlled so to ensure that their resource usage does not violate guaranteed real-time services to existing established channel.

2. *Resource reservation.* Buffer space need be reserved to ensure that packets will not be dropped; likewise switching bandwidth and processing power to ensure an upper bound on packet delivery.

The client defines the traffic specifications through:

•x_{min}: minimum packet interarrival time

•x_{ave}: average packet interarrival time

•S_{max}: maximum packet size

• I: averaging interval for x_{ave}

The clients Quality of Service requirements are:

$$Q\vec{o}S_r = (D, Z, W, J, U)$$

where D is the end to end delay, Z is the probability that the delay bound is satisfied $Z = Prob\{delay \leq D\}$, W is the probability of packet loss due to buffer overflow, J is the delay jitter (max variation in packet delivery), and U is the probability that delay jitter is satisfied.

Real-time channels can be:

• deterministic: a real-time channel where Z=U=W=1; all packets arrive and arrive on time (guarantees are absolute).

• statistical: a real-time channel where $0 \leq Z * U * W \leq 1$; some packets are lost and/or are late and/or have high jitter (only some guarantees are made).

• best-effort: a real-time channel where Z=W=U=0. No guarantees are made.

The Tenet Suite 1 consists of four protocols as shown in figure 2.

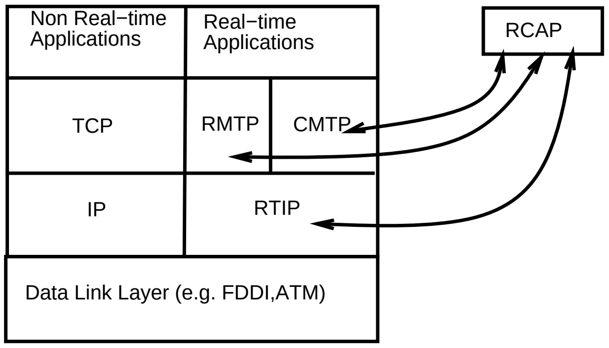

Figure 2: Tenet real-time protocol stack.

•Real-time Channel Administration Protocol (RCAP): A resource management protocol that performs channel setup based upon the clients QoS performance requirements, its traffic description, and the availability of network resources.

• Real-Time Internet Protocol (RTIP): An internetworking protocol which provides data transfer on a simplex, unreliable, *guaranteed performance* packet service. It schedules packets according to reservations made by RCAP.

• Real-time Message Transport Protocol (RMTP): an end-to-end real-time data transfer protocol which provides an unreliable guaranteed performance message service to the clients. RMTP maintains state at the end points of the channel. RMTP state is used to regulate the rate of packets entering the network, and the fragmentation and reassembly of messages into packets.

• Continuous Media Transport Protocol (CMTP): a transport protocol suited for clients which generate and/or consume data at regular time intervals (i.e., continuous media clients). CMTP provides unreliable, end-to-end, continuous, sequenced, periodic transfer of data.

The Tenet protocol suite can coexist with the TCP/IP protocols. and can be easily implemented in a wide spectrum of internetworking environments. In fact, to adapt Tenet real-time protocol suite to different networks, we need to change only:

1. The admission control test
2. The local bound computations

4 Description of the Continuous Media Transport Protocol

The basic components and interactions of the CMTP implementation are illustrated in figure 3

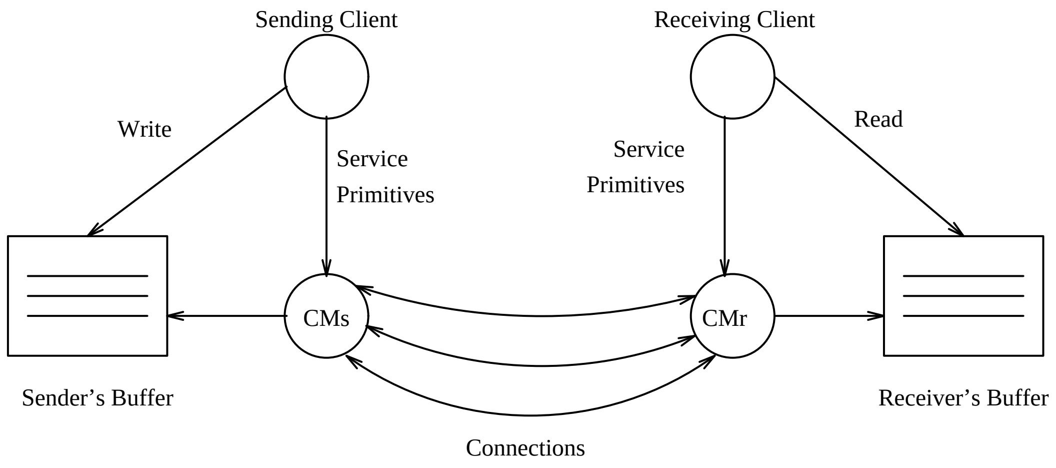

Figure 3: Basic interactions of CMTP.

On the left side we have the sending host. It consists of a sending client, a CMTP entity (CMs) and a buffer shared between the two.

Similarly, on the right side we have the receiving host. The service CMTP offers is divided in a set-up phase and a data transmission phase.
Set up requests are handled on behalf of CMTP by RCAP, which reserves resources at the sending and receiving hosts and establishes a network level real-time connection via RTIP.

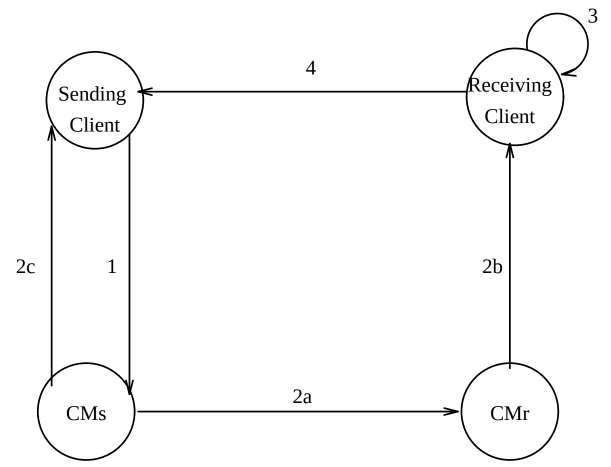

Figure 4: Set up Phase of CMTP.

The main steps of the set up phase are shown in figure 4:

1. The sending client (Cs) requests a continuous media (CM) connection from the CMTP entity on the sending side (CMs), specifying its traffic characteristics and the QoS requirements.

2. If the CMTP entity is willing to handle the request, it passes the request to the CMTP entity (CMr) on the receiving side (2a) and thus up to the receiving client (2b). If not, CMs informs Cs that the request was denied (2c).

3. If the receiving client accepts the request, then the connection is established; otherwise, the connection is refused.

4.The sending client is informed of the receiving client's decision.

The main steps of the data transmission phase are shown in figure 5:

At the sending side.

1. The sending client informs the transport service (CMs) that transmission begins by opening a stream.

2. The sending client writes data into the buffer it shares with CMs.

3. CMs informs CMr that data is being stored in the receiver's buffer. Data transfer occurs through RTIP.

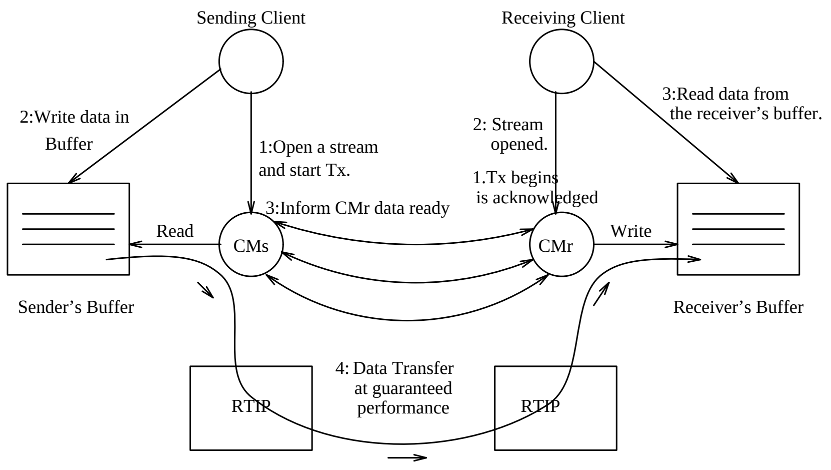

Figure 5: Data Transmission Phase of CMTP.

4. Once per period CMs copies the data from the sender's buffer to the receiver's buffer, satisfying the performance requirements specified during set up phase.

At the receiving side.

1. CMr receives an Open PDU packet from its peer entity on the sending side (CMs).

2. CMr informs Cr that the channel has been opened.

3. The receiving client reads data from the receiver's buffer.

Finally, figure 6 shows the main steps of the stream closure of CMTP.

1. The sending client informs CMs that it wants to close a stream.

2. CMs empties the sender's buffer (2a) and informs CMr that the stream is being closed by sending a Close PDU packet (2b).

3. CMr informs the receiving client that the stream has been closed.

5 A Rate Control Scheme

To implement a rate control scheme, we choose a sliding window algorithm. A sliding window is used to *smooth* the traffic sent by the CM client. We introduce the following relation:

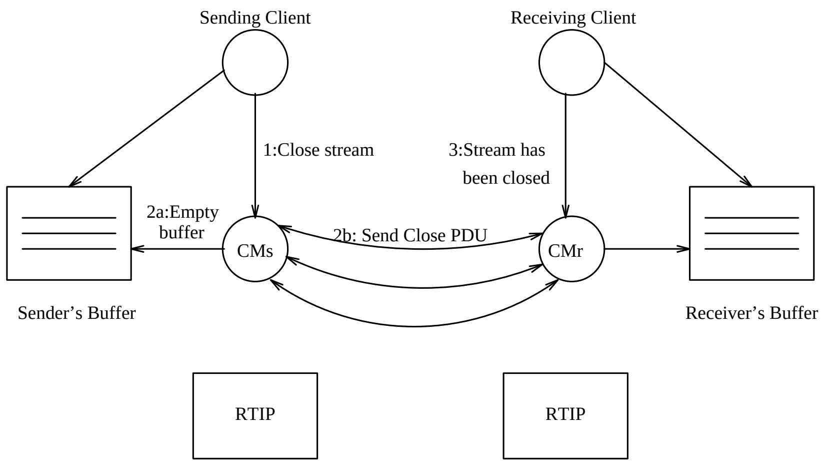

Figure 6: Stream Closure of CMTP

$$I = n * T \tag{1}$$

where I is the averaging interval defined previously and the width of the sliding window, T is the frame time, and n is the number of frames per window. If the window begins at t=0, the window interval can be represented as:

$$I = I_0 = [0, n * T] \tag{2}$$

At time t = T (i.e., the next window) can be represented as:

$$I = I_T = [T, (n+1)T] \tag{3}$$

that is, a translation of width T of the original window.

The condition on x_{ave} implies an upper bound on the number of packets that can be sent in a window of length I (i.e., $\frac{I}{x_{ave}}$). The condition on x_{min} implies an upper bound on the number of packets that can be sent in T (i.e., $\frac{T}{x_{min}}$). Considering CMTP's sliding window rate control scheme, the maximum number of packets that we can send in the last frame of an interval is determined by the conditions on x_{min} and x_{ave}:

$$MaxPackets = min\{\frac{T}{x_{min}}, \frac{I}{x_{ave}} - NP_{n-1}\} \tag{4}$$

where NP_{n-1} is the number of packets sent in the previous (n-1) frames.

This equation, for the numeric values of T=10ms, I=50ms, n=5, x_{min}=2.0ms, x_{ave}=2.5ms and a packet arrival distribution as shown in figure 7 becomes:

$$MaxPackets = min\{5, 3\} = 3 \tag{5}$$

Let us consider the next time frame of width T, that is the interval [6T, 7T]. Let us calculate how many packets the algorithm allows to send in this time frame.

Formula 4 becomes:

$$MaxPacketsin[6T, 7T] = min\{\frac{T}{x_{min}}, \frac{I}{x_{ave}} - NP_{n-1}\} = min\{\frac{10}{2}, \frac{50}{2.5} - 18\} = 2 \tag{6}$$

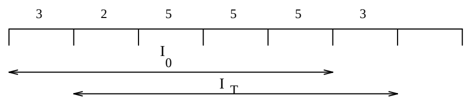

Figure 7: An example of how many packets arrive per time frame T to RTIP using the sliding window rate control scheme.

RMTP does not implement a sliding window rate control scheme. Therefore, it will send packets at intervals of x_{min} until the x_{ave} condition does not hold (i.e., when $\frac{I}{x_{ave}}$ packets have been sent) and then it will wait until the next interval (I) before sending further packets (see figure 8). This behaviour affects the number of buffers utilized in RTIP.

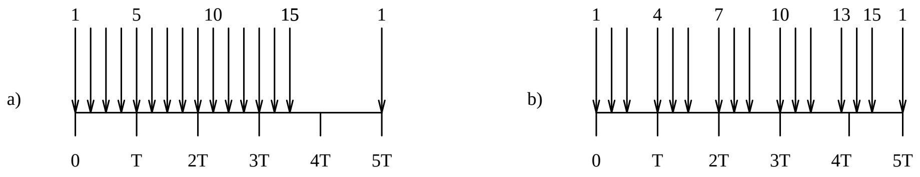

Figure 8: Packet interarrival distribution over I (I=5T) for RMTP (a) and CMTP (b).

6 Simulation Scenario

We have implemented a simplified model of CMTP on the real-time network simulator Galileo[11] for comparison with RMTP. In particular we were interested in evaluating buffer usage.

Assumptions for the model In developing our model of CMTP, we chose the following traffic characteristics for traffic generated by CMTP:

•$x_{min} = 2.0ms$

•$x_{ave} = 2.5ms$

•$I = 50ms$

• Type of realtime channel: deterministic

These parameters control the rate at which CMTP entities send data. As long as the traffic that CMTP sends to RTIP behaves according to these parameters, RTIP will guarantee CMTP's performance requirements.

Furthermore, we made the simplifying assumptions:

1. There are always buffered packets ready to be sent.
2. All packets are of the same size.
3. There is no delay jitter control.

Because we assume no delay jitter, and because CMTP is a deterministic time-driven protocol (because of the first assumption), we can consider RTIP to be a work-conserving queue (i.e., no waiting packets are left unserved).

CMTP is deterministic because it sends packets according to the sliding window rate control scheme described in the next section. The assumptions for x_{min}, x_{ave} and I are based on typical values. From these values, CMTP implements its rate control scheme.

Using our simplified model of CMTP for this study, we show that CMTP utilizes buffers more efficiently than RMTP because it smooths traffic. We ran many simulations comparing the number of buffers used when a particular number of CMTP or RMTP clients were running. In this study, we show the results produced by four CMTP clients versus four RMTP clients sending packets at the peak rate (worst case). The number of buffers used is shown in figure 9.

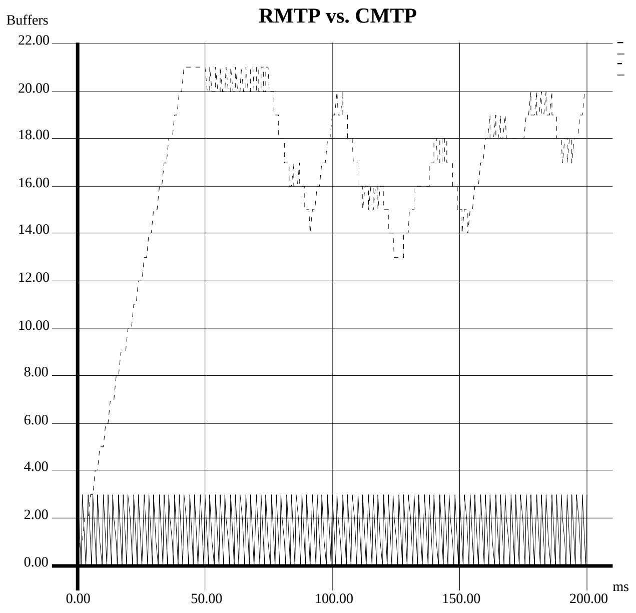

Figure 9: Buffers utilized for four RMTP clients vs. four CMTP clients.

As the figure shows, CMTP uses buffers more efficiently than RMTP because of its sliding window rate control scheme. When using CMTP, the maximum number of buffers used is three, while for RMTP the maximum number is over twenty. Since CMTP smooths traffic, the number of buffers used oscillates between zero and three. In contrast, RMTP does not smooth traffic, so the number of buffers used first increases and then oscillates. The initial upward sloping portion of the graph (for $t < 50$) occurs because the sources are sending packets at an interarrival rate of x_{min}.

This continues until the x_{ave} condition is no longer satisfied (around t=40, i.e., when the number of packets sent in the interval I is equal to $\frac{I}{x_{ave}}$). In fact, RMTP sends packets at 2 ms intervals. Thus in I (of width 50 ms) we can send $\frac{I}{x_{ave}} = 20$ packets. If the first packet is sent at t=0, and further packets at intervals of 2ms, the last packet (number 20) is sent at t=40 ms. The subsequent plateau means that RTIP is sending out packets at the same rate that they are arriving ($40 < t < 70$).

The successive oscillations are due to a source that either stops sending packets (i.e., the number of packets sent in the interval I is equal to $\frac{I}{x_{ave}}$) or starts sending packets again because a new interval I begins.

7 Conclusions

The goal of the simulation was to examine how CMTP can improve network resource utilization (buffers) when considering CM clients. Because of the regularity of data produced by CM clients, CMTP was able to *smooth* it using a priori knowledge about the timing of future data transmission.

We described a sliding window rate control scheme used by CMTP to smooth traffic. Since RMTP is unable to smooth traffic, we showed that it caused RTIP to use more buffers than did CMTP. These preliminary results provide justification for having a separate transport layer protocol for CM clients and to continue the implementation of CMTP in the internetwork. However, the need for a reliable transport service tailored for message-oriented real-time applications, requires the development of techniques to improve the efficiency of the implementation of protocols such as RMTP. We plan to pursue this goal by analysing the communication patterns of typical message-oriented applications, in order to devise an optimized resource allocation scheme.

References

[1] D. P. Anderson, R. G. Herrtwich, and C. Schaefer. Srp: A resource reservation protocol for guaranteed-performance communication in the internet. Technical Report TR-90-006, International Computer Science Institute, Berkeley, California, February 1990.

[2] A. Banerjea, D. Ferrari, B.A. Mah, M. Moran, D.C. Verma, and H. Zhang. Tenet real-time protocol suite: Design, implementation, and experiences. Technical Report TR-94-059, International Computer Science Institute, Berkeley, California, November 1994.

[3] D. Clark, V. JAcobson, J. Romkey, and H. Salwen. An analysis of tcp processing overhead. *IEEE Communications Magazine*, June 1989. pp 23–29.

[4] L. Delgrossi, R.G. Herrtwich, and F.O. Hoffmann. An implementation of st-ii for the heidelberg transport system. technical report no. 43.9303. Technical report, European Networking Center, Heidelberg Germany, IBM ENC, 1993.

[5] D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia - a discussion of the tenet approach. Technical Report TR-92-072, International Computer Science Institute, Berkeley, California, October 1992.

[6] D. Ferrari and A. Gupta. Resource partitioning for real-time communication. *First IEEE International Symposium on Global Data Networking, Cairo, Egypt*, December 1993.

[7] J. Funda. Towards delay-invariant remote manipulation. Technical report, University of Pennsylvania, Dept. of Computer and Information Science, May 1991.

[8] D.B. Hehmann, M.G. Salmong, and H.J. Strittgen. Transport services for multimedia applications on broadband networks. *Computer Communications*, 13, 1990.

[9] D.R. Hofstadter and D.C. Dennet. *The Mind's I: Fantasies and Reflections on Self and Soul.* Basic Books, New York, 1981.

[10] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multihop networks. *IEEE Transaction on Parallel and Distributed Systems*, 5(10):1044–1056, October 1994.

[11] E. Knightly and G. Ventre. Galileo: A tool for simulation and analysis of real-time networks. Technical Report TR-93-008, International Computer Science Institute, Berkeley, California, March 1993.

[12] C. Szyperski and G. Ventre. Efficient multicasting for interactive multimedia applications. Technical Report TR-93-017, International Computer Science Institute, Berkeley , California, March 1993.

[13] L. Zhang and S. Deering et. al. Rsvp: A new resource reservion protocol. *PRELIMINARY DRAFT*, 1993.