CONNECTION-ORIENTED REAL-TIME COMMUNICATION FOR MISSION CRITICAL APPLICATIONS -AN INTRODUCTION TO NETEX: A PORTABLE AND EFFICIENT TOOL KIT

B. Devalla, C. Li, A. Sahoo and Wei Zhao[1]
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
Email: zhao@cs.tamu.edu

Keywords: deadline guarantees, connection oriented communication, high speed networks, real-time, connection admission control.

Abstract

We report our project aimed at the design and implementation of a mid-ware tool kit, called *NetEx*, that provides deadline-guaranteed and connection-oriented communication services at the application layer. NetEx comprises of a *Host Traffic Manager* (HTM) and a *Network Traffic Manager* (NTM). An HTM resides in each participating host while the NTM can be implemented either at a central host or in a distributed manner. The HTM performs traffic scheduling and enforces traffic regulation at individual hosts. The NTM is the decision maker on connection admission control. We argue that portability and efficiency are two fundamental issues one has to address in the design of such a middle-ware tool kit. We present strategies adopted by NetEx in order to support portability and efficiency. We report system performance based on several benchmarks.

1. Introduction

1.1 Background

Deadline-guaranteed communication service is crucial for mission critical real-time applications. Generally speaking, communication services over packet-switching digital networks can be classified as *best effort* service, and *connection-oriented* service. Many of the digital communication networks deployed currently provide best effort service. That is, the underlying network (and its management) offer no prior guarantee regarding the quality of service (QoS) an application may receive. Best effort service is thus well-suited to public data networks that carry web, file transfer, and electronic mail traffic.

On the other hand, mission critical applications in embedded systems (such as those in submarine, aircraft or industrial process controllers) demand QoS guarantees. These applications comprise of distributed processes that execute on different hosts and cooperate by exchanging messages to achieve a common objective. A requirement of these applications is that they must accomplish these tasks by specific deadlines. The success of the system in supporting these applications therefore depends crucially on the ability of the communicating hosts and network to guarantee a pre-specified QoS, such as the transfer of all critical messages by their deadlines. In other words, for every message sent by an application, the worst case delay in transferring it must be guaranteed to be no more than its deadline.

Since 1980s, connection-oriented communication service has been proposed and developed to address the problems such as delay guarantees. A connection is an abstract communication service provided by the network to a particular application. It can be considered as a contract between a communication application and the underlying network: the application specifies the characteristics of its traffic and the network agrees to provide the requested quality of service to the application. The network will not admit a connection if the requested quality of service cannot be guaranteed. Thus, one may view a connection as a virtual link that has a certain traffic-carrying capacity and is dedicated for use by an application. Mission critical applications therefore preferentially use connection-oriented services.

[1]This work was partially sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number F49620-96-1-1076 and by Texas Higher Education Coordinating Board under its Advanced Technology Program with grant number 999903-204.

Much of existing work on deadline guarantees has concentrated on the network level, i.e., bounding the message delay in the network [1-6]. When using a network to support distributed mission critical systems, we also need to consider the application-to-application delay because it is this delay that determines whether application deadlines will be satisfied. The *application-to-application delay* is the time delay experienced by a message that is sent between application tasks.

The generic solution for guaranteeing end-to-end delay bounds in distributed systems consists of connection-oriented communications with some form of admission control and traffic regulation (typically based on packet scheduling at the network interface) [7-14, 18]. Supplementary research in protocols for guaranteed services on networks is reported in [15-17]. The Resource Reservation Protocol (RSVP) [15,16] is a receiver-oriented set-up protocol for connectionless networks for providing Quality of Service (QoS) guarantees. ATM Forum's UNI Signaling protocol (derived from ITU-T Q.2931) describes QoS and traffic parameters specification procedures [17].

The US DoD has recognized that information systems play a critical role in current and future military operations. Extensive studies have been carried out to examine potential problems and outline strategies to deal with them [28]. Attention has been particularly focused on improving the responsiveness, security, and reliability of communication services in a flexible architecture that readily embraces newer technologies in an efficient manner [29]. The NetEx project is envisioned to be an effort toward this objective.

1.2 NetEx Tool kit

NetEx (short for *Network Express)* is a software suite designed and implemented to deliver connection-oriented deadline guaranteed communication services at application layer for mission critical applications. User applications request connection set-up by specifying their traffic (using some standard traffic descriptor) and quality of service (e.g., delay bound). NetEx runs a connection admission control (CAC) procedure to check if the new connection can be accepted. This admissibility ensures that the deadline requirements of the new and existing connections can all be guaranteed.

NetEx comprises of a Host Traffic Manager (HTM) and a Network Traffic Manager (NTM) (Figure 1). An HTM resides in each participating host while the NTM can be implemented either at a central host or in a distributed manner. The HTM performs traffic scheduling and enforces traffic regulation at individual hosts. The design of the HTM enables its functions to be realized complementing those already provided by the underlying system. The NTM is the decision maker on connection admission control. The NTM is also responsible for resource allocation both in the network and at the host. It does so by relatively balancing the load on hosts and network.

In this paper, due to the space limitations, we will not provide a complete description of NetEx functionality, the design rationale, and performance evaluation. Instead, we emphasize on two most important design issues, namely portability and efficiency, and discuss the NetEx solution in addressing the same.

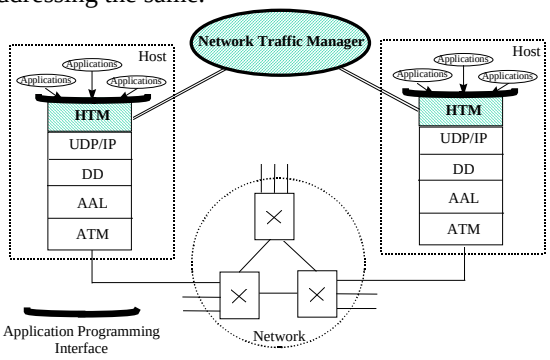

Figure 1 NetEx Architecture

1.3 Design Challenges

Computation time and resources are always at a premium in embedded systems. They therefore demand efficient implementations of any service built for them. In the case of delay guarantees, the overall system efficiency is manifest in the number of connections that can be guaranteed at any given time. In addition to *efficiency*, we also address the issue of *portability* in the design of NetEx. Portability is that aspect of software architecture that allows it to work seamlessly not only over many existing operational platforms, but also over those emerging in the foreseeable future.

The need for portability is immediately evident from the rigorous system requirements of mission critical applications. There has always been a constant demand for high performance network and host systems. The underlying platforms on which an embedded system is built may be frequently upgraded in terms of increase in size and/or of deployment of new devices. The software for such systems needs to accommodate this evolution.

NetEx adopts a two-fold strategy integrating host and network management to provide an efficient and portable solution for delay guarantees. First, it uses a *decomposition* approach for delay analysis, in

which a connection is broken into a set of servers, and the end-to-end delay of a connection is obtained by analyzing individual servers. There are several advantages of this approach: 1) The relatively simple nature of delay computation reduces the time taken during connection admission; 2) The delay bounds are tighter, thus increasing the number of connections that can be supported; and 3) Proper decomposition reduces the number of server types to be analyzed to a smaller subset even when the system is composed of a variety of networks and host operating systems. This in effect decouples the complexity of network topology from delay analysis. Our decomposition-based delay analysis routine can thus be easily configured to support different homogeneous and heterogeneous networks of the present and future. Thus, our decomposition approach of delay analysis increases system efficiency while simultaneously supporting portability.

The second strategy targets host traffic control. Needless to say, traffic in both sender and receiver hosts has to be managed in order to achieve delay guarantees at the application layer. Host traffic control depends significantly on the user application structure and operating environment. It can thus be enforced in a number of ways - by the applications themselves (with self-traffic regulation), by the underlying operating system (real-time scheduling of traffic), by middle-ware (like NetEx), or even by a combination of these. A second look quickly shows the hopeless complication in including traffic control in complex distributed applications; neither is it reasonable to expect the different flavors of operating systems to uniformly provide traffic control. A middle-ware solution to host traffic control in the form of NetEx thus looks promising in terms of applicability. However, a middle-ware in an effort to provide the needed control functions may choose to ignore any similar provisions inherent in the applications and operating system. While solving the portability problem, this approach will not be efficient. NetEx adopts a *complementation* approach. NetEx contains a spectrum of host traffic control functions. The user can configure NetEx to render functions that are necessary but complementary to those already provided by applications and operating systems. This paves way for supporting portability without sacrificing efficiency.

2. The Network Traffic Manager (NTM)

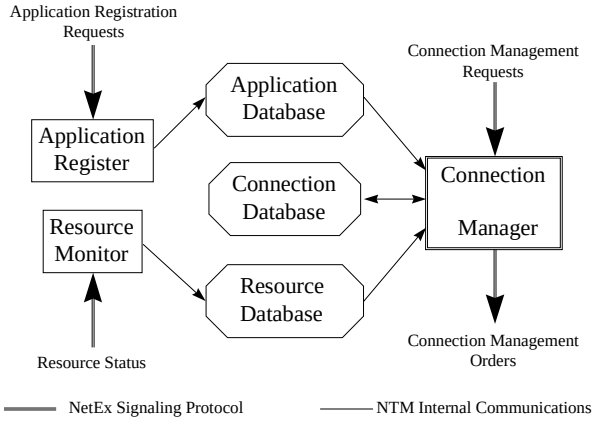

Figure 2 Architecture of the Network Traffic Manager (NTM)

2.1 NTM Architecture

In NetEx, the Network Traffic Manager (NTM) is the primary decision maker for connection management. Decisions such as whether a new connection can be admitted, and if so, how resources (e.g., buffer space, time slot, etc.) should be allocated are made by the NTM. To facilitate a complete management of connections, the NTM also has application registration and resource monitoring functions. Figure 2 shows the architecture of the NTM. Basically, the NTM consists of three databases and three execution modules. We now briefly describe their main functions:

- *Application Database and Application Register*. Application here refers to a user program that uses communication service in the system. To protect the system from misbehaving applications, we require all the applications to be registered. The API in the HTM is responsible to receive the registration and transfer it to the NTM. The *application register* is responsible to process the registration and store it in the *application database.*
- *Resource Database and Resource Monitor.* The admissibility of a connection depends on the availability of resources such as links and switches. The *resource monitor* utilizes the built-in mechanism in the network to periodically collect status information of resources and store it in the *resource database* which will be utilized for connection admission.
- *Connection Database and Connection Manager.* The function provided by the *connection manager* includes the connection admission control (CAC) and connection tear-down. The *connection database* maintains a list of records for the connections that are currently active.

2.2 Connection Admission Control Procedure

Of all the functions that NTM provides, connection admission control (CAC) is the most critical and challenging. The primary function of the CAC procedure is to determine whether or not, upon a request, a new connection can be admitted. A connection will be admitted only if the QoS (e.g., deadline) requirements of this connection and other existing connections can be satisfied. Prediction of worst-case delays is thus a big part of the CAC procedure. The CAC procedure implemented in the current version of NetEx has three major steps:

- *Validation of the CAC request*. This is to check if the initiator, the sender, and the receiver have proper rights for establishing the requested connection.
- *Route generation*. For the purpose of efficiency, in the current version of NetEx, a route with the shortest path is used. Our performance study has found that this method performs reasonably well in comparison with others while having a minimum run-time overhead [19].
- *Delay derivation and testing*. This is the most critical step. We first derive the delay bound of the new connection and then test if it is no more than the deadline requested. Since the introduction of the new connection may impact the delays of some existing connections, their delays will also have to be re-derived and tested.

2.3 Delay Analysis

Effective calculation of application-to-application delay is very important for the performance of NetEx. In this section, we will demonstrate how delay analysis contributes to NetEx efficiency and portability.

2.3.1 Decomposition Approach

To provide deadline guarantees, the system has to ensure that delay suffered by a connection is no more than the corresponding deadline. NetEx uses a *decomposition* approach to derive the delays. The decomposition approach consists of the following 3 steps: (1) Partition the (route of) connection into a sequence of servers; (2) Calculate delays at individual servers; and (3) Sum up the delays at all the individual servers to obtain the end-to-end delay of the connection. While it sounds straightforward, we need to address three issues in order to successfully apply this decomposition approach: (1) Proper network decomposition, (2) Comprehensive but concise traffic description and (3) Efficient and effective server analysis. We describe these three in the subsequent subsections.

2.3.2 Network Decomposition

We will illustrate network decomposition with the aid of an example. A network with an ATM backbone inter-connecting hosts on FDDI rings is shown in Figure 3 [30]. Network decomposition makes it simple to analyze delays in a structure that appears formidable for delay analysis.

Figure 3. ATM-Based Heterogenous Network (ABHN) Architecture

Consider the path of a connection between hosts X and Y (See Figure 3) on two LAN segments interconnected to two different interface devices [2]. We then hierarchically decompose the connection path into a sequence of servers. The traffic of the connection first goes through the sender host (X), the FDDI ring on the sender site (denoted as FDDI_S), an interface device on the sender site (ID_S), through the ATM backbone to the receiver host Y via the second interface device (ID_R) and the receiver FDDI ring (FDDI_R). The (worst-case) delays can thus be decomposed as,

$$d_{end\text{-}to\text{-}end} = d_X + d_{FDDI\_S} + d_{ID\_S} + d_{ATM} + d_{ID\_R} + d_{FDDI\_R} + d_Y \quad (1)$$

Each term on the right hand side of (1) represents the worst case delay at the corresponding component. Each of these components may be further decomposed. For example, we have

$$d_{FDDI\_S} = d_{FDDI\_MAC} + d_{Delay\_Line} \quad (2)$$

where FDDI_MAC represents the host FDDI network interface card and the Delay_line server accounts for the physical link. Similarly,

$$d_{ID\_S} = d_{IPS} + d_{FSS} + d_{FCCS} + d_{AOPS} \quad (3)$$

which breaks the ID_S server into Input Port Server (IPS), Frame Switch Server (FSS), Frame-cell Conversion Server (FCCS) and ATM Output Port Server (AOPS). Again,

$$d_{ATM} = d_{IPC} + d_{SF} + d_{OPC} \quad (4)$$

which represents the decomposition of the backbone ATM switch into Input Port Controller (IPC), Switching Fabric (SF) and Output Port Controller

[2] The interface devices are important components in ABHN architecture as they allow legacy LAN segments to be interconnected through the ATM backbone.

(OPC). We can write similar delay equations for the servers ID_R and FDDI_R on the receiver side. This decomposition virtually lists every single source of delay suffered by a connection path.

Via this decomposition process, we see that there are two types of servers in a network: 1) *constant* delay servers (i.e., the connection suffers a constant delay in them) and 2) *variable* delay servers (where the delay of a connection may vary depending on the traffic). Among the servers listed, the delay line in FDDI ring, the Input Port Server, the Frame Switch Server and Frame-cell Conversion Server for the Interface Device, and Input Port Controller and the Switching Fabric for the ATM switch are all constant delay servers. The delays in each of these elements can be measured.[3] This now leaves us with the problem of calculating delays in a much smaller subset of variable delay servers, namely FDDI_MAC server, ATM Output Server in the Interface devices and Output Port Controller of the ATM switch. Various methods for analyzing delays in these variable delay servers have been developed by us and others [1,2,13,22,30,31].

Through this process, we see that the end-to-end delay of the monstrous ABHN (Figure 3) can thus be obtained by merely summing up the delays in the simple servers with well-established delay analysis results, which is easy to be implemented and to be understood. Thus, the decomposition approach inherently endows simplicity to the delay analysis scheme.

We also observe that with the proper decomposition, even for such a complex heterogeneous network, only a few *types* of servers need to be analyzed. This observation is generally true for almost any digital networks. This illustrates another (more important) advantage of the decomposition approach. A decomposition-based delay analysis routine is able to evolve naturally with the development of newer networks. Introduction of a newer network only adds on a fewer number of servers to be analyzed.

As a result, a decomposition-based delay analysis routine can thus be easily configured to support different homogeneous and heterogeneous networks of the present and future. To exploit this advantage, the delay analysis routine in NetEx is initialized by a configuration file that specifies the decomposed network (i.e., the basic servers and their linkage). Any change of the network only necessitates a revision of this configuration file. For the details of this specification file, see [19].

2.3.3 Traffic Description

[3] These values are indeed offered by device manufacturer as part of device specifications.

The overall efficiency of the system depends heavily on the tightness of the worst case delays that are derived by the delay analysis routine. Tighter delay bounds allow NetEx to support more connections, thus making the CAC procedure efficient.

Traffic description presents itself as the sole parameter that can be manipulated in the entire delay analysis scheme. It is therefore important to choose the right form of traffic description. Stochastic models used in traditional networks to describe traffic only provide insights into the average performance of the system and cannot capture the worst case behavior of traffic as we needed. A traffic description based on peak rate, on the other hand, can frequently lead to over-estimating worst-case delay. While reasonably accurate in processor scheduling [26], the *periodic* (C, P) model (where C packets are transmitted periodically every P seconds) cannot correctly reflect the traffic status in the network, often resulting in loose delay bounds [32].

To solve this problem, several research groups have conducted comprehensive theoretical and experimental investigation [3,11,14,32]. The maximum rate function is identified to be suitable to describe traffic in embedded systems. The maximum rate function is defined as the maximum data arrival rate in a time interval of length I units at a particular point of the network for a connection. That is, for connection j, its traffic at the entrance of server i is given by

$$\Gamma_{i,j}(I) = \max_{\forall t>0}\left(\frac{\textit{total number of bits arrived in the interval } (t, t+I]}{I}\right) \quad (5)$$

The definition of $\Gamma(I)$ allows for an easy transformation of any other traffic representation in an user application. It provides sufficient information about the traffic so that delay bounds derived will be tighter than using other traffic models (say, peak rate model and periodic model). In practice, the use of $\Gamma(I)$ may be cumbersome due to: 1) It takes a large amount of memory space to store $\Gamma(I)$ if a closed form for $\Gamma(I)$ cannot be found; 2) It may not be feasible to extract the enormous amount of information needed to construct $\Gamma(I)$ for real world traffic. Approximation methods have been proposed to avoid these shortcomings [21]. In particular, the *point approximation method* is used to construct an approximate $\Gamma(I)$ when traffic information at limited number of points are given. Our experiments have shown that a six-point approximation can characterize real traffic with good precision [32]. NetEx uses the six-point approximation method to represent $\Gamma(I)$ functions.

2.3.4 Server Analysis

Given a network decomposition and a traffic description by the maximum rate function, server analysis is responsible to analyze individual servers in order to obtain (1) the worst case delay suffered by a connection at a server; (2) the buffer space needed at a server; and (3) the traffic description of a connection at the output of a server. While the first two are directly related to delay guarantees, the third one, the traffic description at the output of a server is necessary in order to analyze the subsequent servers.

For different type of servers, the analysis techniques are different. Due to the space limitation, we cannot present the methods to analyze all kinds of servers in this paper [See 30]. Nevertheless, we show how to analyze a server, namely an FIFO multiplexer here. This server serves connections in the order of First-In-First-Out. The output port controller of ATM switches and Interface devices are examples of this type of server.

Let server *i* be a multiplexer with FIFO scheduling policy. Let Ω_i be the set of indices of connections that pass through server *i*. Let $\Gamma_{i,j}$ and $\Gamma'_{i,j}$ be the traffic description functions of connection *j* at the input and output of server *i*, respectively, where $j \in \Omega_i$. Using these notations, the maximum buffer requirement at server *i*, Q_i, is given by

$$Q_i = \max_I \left(\sum_{j \in \Omega_i} I \cdot \Gamma_{i,j}(I) - I \cdot p_i \right) \qquad (6)$$

where p_i is the transmission speed (bits/second) of server *i*. (6) can be easily explained. By the definition of $\Gamma_{i,j}$, $\Sigma\, I\, \Gamma_{i,j}(I)$ represents the maximum number of bits that may arrive during any interval of length I from all the connections passing through server *i*. (Ip_i) is then the number of bits that may be transmitted by server *i* during an interval of length I. Hence, their difference (i.e., $\Sigma\, I\, \Gamma_{i,j}(I) - I\, p_i$) is the maximum buffer required during any interval of length I. Maximizing over I, we have (6). Let the real size of buffer at server i be S_i. Buffer overflow may occur and messages may be lost if $Q_i > S_i$. Hence, $Q_i \leq S_i$ is a constraint that must be satisfied during connection admission control.

Furthermore, for an FIFO server, the packet whose arrival results in the maximum queue length, should suffer the worst case delay. Hence,

$$d_{i,j} = Q_i\, p_i \qquad (7)$$

For the description of output traffic at server *i*, we have the following results:

$$\Gamma'_{i,j}(I) = (I + d_{i,j})\, \Gamma_{i,j}(I + d_{i,j})/I \qquad (8)$$

That is, we can express the output traffic, $\Gamma'_{i,j}(I)$, in terms of the input traffic, $\Gamma_{i,j}(I)$ and the delay, $d_{i,j}$. (8) follows directly from the delay property of maximum rate function [1]. Using (8), subsequent servers can be analyzed. Once all the delays at individual servers are obtained, we can substitute (7) into (1) to compute the delay for connection *j*.

This example of server analysis shows that in NetEx the delay computation for individual servers is relatively simple, given that the network is properly decomposed and the traffic is accurately described. Thus, the connection admission control in NetEx is efficient in the sense that admission decision can be made quickly and correctly. Recall that at the same time we are able to support portability as well.

3. Host Traffic Manager

3.1 HTM Architecture

The HTM is a host resident module present in every host. The HTM architecture is shown in Figure 3. It consists of several modules that work cooperatively in order to realize the HTM functionality. These modules are:

- *Run Time Traffic Control Module (RTTC).* Recall that, application-to-application delay guarantees necessitate a host-based entity to perform traffic management. Host traffic control functions are built into the RTTC. The RTTC is the execution engine in the HTM that schedules and regulates the packets belonging to different connections in a host.
- *Local Application and Connection Management Module* (LACM). While the decisions on connection admission are made by the NTM, the HTM has to provide necessary local support for the NTM to function efficiently. The needed support includes (i) information exchange between applications and the NTM, (ii) estimation of host delays (i.e., d_{source} and $d_{destination}$), (iii) local parameter setting (e.g., buffer location and size, traffic transmission rates, etc.), and (iv) local bookkeeping.
- *User Interface Module.* A user interface is a desired feature for monitoring and control. The User Interface Module serves as a graphical interface with point-and-click mechanisms to manage connections, monitor host and network status, collect traffic statistics, record connection history, serve as a debug console, etc.
- *Signaling Protocols and Application Programming Interface*. Signaling protocols are defined for reliable communications among the

NetEx modules. An application programming interface provides a library of routines an application may call to use NetEx services.

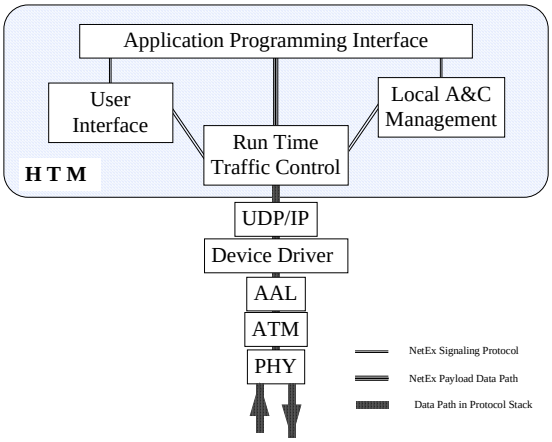

Figure 3 HTM Architecture and its Position in the Protocol Stack

Clearly, all these modules must be properly designed and implemented in order to achieve the objective of the HTM. Due to the space limitation, in the rest of this paper, we will concentrate on the Run-Time Traffic Control Module because it performs critical operations of host traffic management. A discussion of other modules can be found in [23].

3.2 Run-Time Traffic Control Module

3.2.1 Traffic Control and NetEx Approaches

Host traffic management is necessary for two reasons: 1) Packet transmission delay in the host has to be controlled. This is the latency between an application sending a packet and the packet actually appearing on the network, and 2) The amount of traffic sent into the network by an application must be regulated. Recall that for delay guaranteed service, applications have to adhere to their a priori traffic specifications. A policing function is thus necessary to deal with errant applications so that the misbehavior does not affect guarantees made to other connections. The main design challenge is to provide host traffic management that is portable across hosts with different flavors of operating systems and efficient in terms of the mechanisms for traffic control not adding too much overhead.

Management of host delay alone does not necessitate a component like HTM. Operating systems differ in their ability to schedule tasks with timing constraints. Many real-time operating systems[4] use sophisticated scheduling mechanisms that ensure task completion within deadlines. Similarly, applications with sufficient language support can manage host delays too. For example, Ada has real-time support. In either of these cases, the homogeneity of operating environment keeps the middle-ware layer very thin (i.e., less work for this layer) even for distributed applications. HTM becomes necessary when we need to provide delay-guarantees across a variety of platforms. RTTC in NetEx contains host traffic management options that are designed to work harmoniously with an assortment of operating systems and application structure. This is the crux of support for NetEx portability.

NetEx is a middle-ware solution that fulfills the traffic management needs for real-time applications that run over a variety of operational platforms. Depending on the level of control exercised, host traffic management tacks on a communication overhead. For the sake of efficiency, it is essential to exploit but not to duplicate any control functions already provided by the underlying operating system and applications. NetEx adopts a *complementation* approach that precludes any duplication of control. NetEx supports a number of different host traffic control functions each of which provides functionality complementary to those already provided by applications and operating systems.

Thus, NetEx is designed to offer a broad spectrum of options for traffic management in the host. One extreme is called *voluntary management (VM)* in which the applications and/or operating systems incorporate necessary techniques to ensure that messages are processed and transmitted at the host in a timely manner. On the other extreme, we have *enforced management (EM)*, where traffic generated by applications are explicitly controlled by NetEx. Note the tradeoff between the two extremes: In VM, the lack of any supervisory control can reduce the processing overhead in the host. It, however, places all the trust on the applications and operating systems to adhere to the traffic description specified at connection set-up time. Misbehavior of even one application can potentially affect the guarantees made to others. EM tightly controls the amount of traffic placed into the network by an application. If any application misbehaves, the system can promptly take corrective actions (including tearing down of the errant connection).

3.2.2 Two Implementations

NetEx is designed to work over the entire spectrum of traffic management allowing VM to EM and anything in-between. It allows the user to have her/his choice in the interplay of operating system support, application semantics, and network support to best suit her/his needs. The current version of NetEx implements two schemes that exploit the

[4] RT-Mach for example

flexibility of the design: *Table Driven* method, where packet transmissions are based on an execution table, and *Rate Function Controlled* method where a simple policing approach is used. The former is a form of EM while the latter is close to VM.

• *Table Driven (TD) traffic management.* The RTTC schedules packet transmissions based on an execution table. An execution table on a host consists of a number of execution slots where one or more packets can be transmitted per slot. The number of slots assigned to an individual connection is determined by the NTM at the connection admission time, based on the delay requirements. The RTTC continuously executes the table in a cyclic manner. NetEx thus provides strict traffic control via this table-driven mechanism. The traffic shaping is achieved by properly spacing the slots assigned to a connection. This requires virtually no support from the applications and operating system. Hence, it is a form of EM

• *Rate Function Controlled (RFC) traffic management*. Here for any packet generated by an application, the RTTC processes and transmits it as long as it conforms to the traffic description specified at the connection admission time. No traffic regulation is provided by the RTTC. The application has to police itself, if necessary. The queuing delay of a packet depends on how the application is scheduled by the underlying operating system. This then necessitates strong OS and application support to ensure the schedulability.

3.2.3 Performance Evaluation

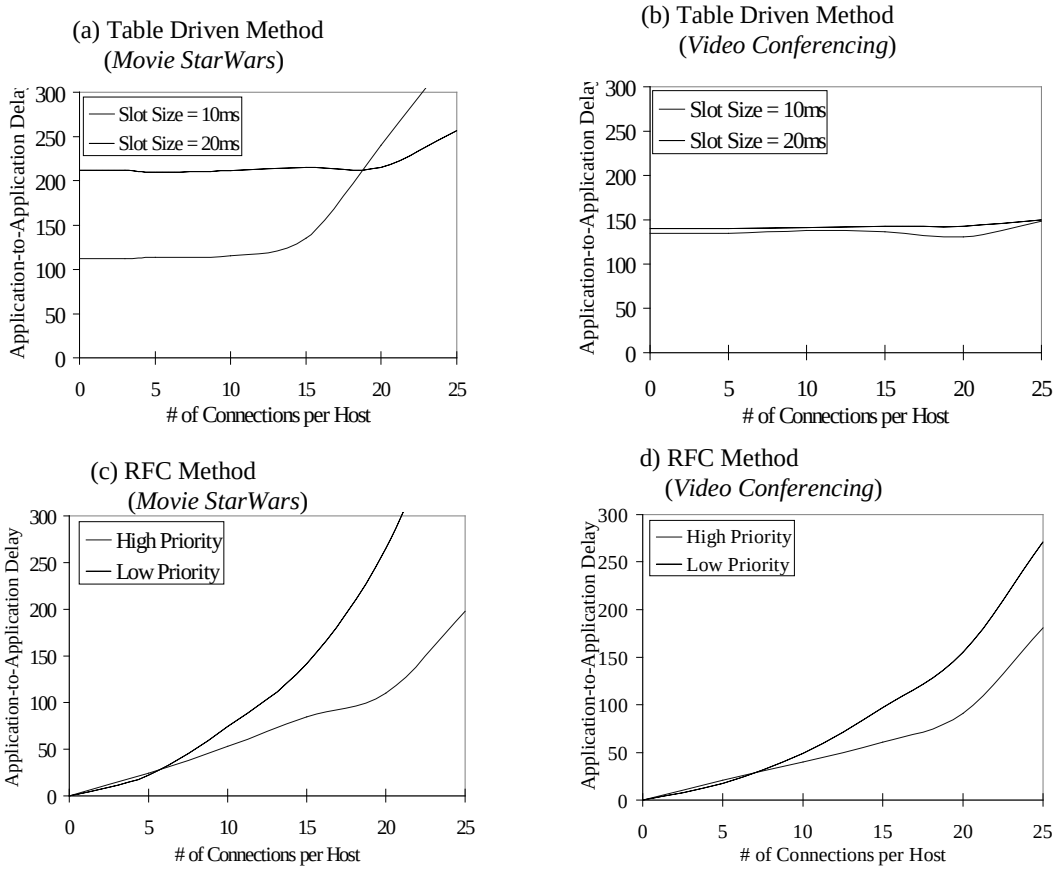

Figure 4 Delay vs. Host Load

In the Distributed System Laboratory of Department of Computer Science at Texas A&M University, we have implemented and tested a version of NetEx on an ATM LAN with five ATM switches (Fore Systems' ForeRunner and BayNetworks' LattisCell) and a cluster of Sun Sparc 4 workstations running Solaris 2.5. A comprehensive performance evaluation is not given here due to space limitation. Nevertheless, we report some findings on the relationship between host load and application-to-application delay.

The data furnished here are based on two popular benchmarks which have been widely used in

studies of this nature. The data sets are MPEG-1 encoded VBR video traces. The first benchmark is from the movie *Star Wars* [24] and the second is from a video conferencing trace [25]. The user application presents one MPEG frame every 40ms in each connection. The host load is measured by the number of such connections supported. For the case where the RTTC uses the table-driven method, the slot size is chosen to be 10ms and 20 ms, and all the applications are given the same priority. For the case where the RTTC uses the rate function control method, applications on a host are divided into two groups with one group given a higher priority than the other. Figure 4 shows the performance data. The ordinate is the delay in milliseconds[5] and the abscissa is the host load in terms of the number of connections supported.

From Figure 4, we make the following observations:

- As the number of connections per host increases, the application-to-application delay increases. At some point (say, 20 connections for the case of the Star Wars traffic with the TD method and a slot size of 20ms), the throughput is *saturated* in the sense that if the number of connections is further increased, NetEx can no longer admit any more connections regardless of their deadlines. This is because at that point buffers may overflow, causing packet loss. Saturation may also occur when the load on the hosts is too high to handle the traffic.

- For the table driven method, the saturation comes up earlier and more abruptly when the slot size is smaller (i.e., 10 ms) than when the slot size is bigger (20ms). This is because NetEx, as a traffic manager and server, has to consume CPU time. A server frequently awoken consumes more CPU capacity.

- With the table driven method, the delay is sensitive to the slot size. When the load is light (i.e., no more than 15 connections per host), the delay with the small slot size (i.e., 10ms) is better than with the bigger slot size (i.e., 20ms). However, when the load is heavy, the bigger slot size does better. Note that a small slot size implies relatively more overhead due to timer control. When the load is light, this overhead is not a problem. However, when the load is heavy, this overhead plays a significant role, causing drastic increases in the delay.

- For the RFC method, delay-load relationship is more sensitive than that of the table driven method. This is because the delay with the RFC method is directly dependent on the prevailing host environment (i.e., host scheduling and load). With the table driven method, the delay mainly depends on the table structure (i.e., the slot assignment).

- The high priority applications using the RFC method have a much smaller delay than the low priority applications. This justifies the observations commonly made about priority driven scheduling being a feasible method to manage delays. However, we notice that the high priority applications achieve their small delay at the cost of other (low priority) connections.

[5] The actual value of delay depends on the processing capacity of the CPU. On faster machines, this delay will be much lower than the numbers reported in this paper.

4. Summary

- Newly emerging applications in mission critical systems need communications service with deadline guarantees. Neither best effort service nor delay guarantees in the network layer provide a satisfactory solution. We have described the design and implementation of a middle-ware tool kit, NetEx, that aims at providing deadline-guaranteed connection-oriented communication service at the application layer. The key contributions which make our work innovative and unique are as follows: Based on the decomposition-based delay analysis method and a spectrum of host traffic control, we have developed techniques that are able to guarantee message deadlines at application layer for mission critical systems. To the best of our knowledge, no previous work in this area has been reported.

- In the design of NetEx, we emphasized on portability and efficiency of the tool kit. As a result, NetEx can be easily configured to work with almost any network and host systems of present and future. The alternative run-time traffic control schemes allow NetEx to meet the disparate needs of different applications under different operating system support.

- Our approach is compatible with the existing network standards and industrial practices. NetEx is realized with network products which are currently commercially available and does not require any change to them. Hence, this middle-ware tool kit is immediately applicable to mission critical systems which require connection-

oriented real-time communication at the application layer.

Currently, we are enhancing NetEx with additional core capabilities including support for heterogeneous networks, fault tolerance, and security.

5. References

[1] R. L. Cruz. A Calculus for Network Delay, Parts I&II.*IEEE Trans. on Information Theory*, Jan. 1991.

[2] D. Ferrari. Real-Time Communication in an Internetwork. *Journal of High Speed Networks*, Jan. 1992.

[3] J. Kurose. Open Issues and Challenges in Providing QoS Guarantees in High Speed Networks. *Proc. of ACM SIGCOMM '93*, Jan. 1993.

[4] C. M. Aras, J.F. Kurose, D.S. Reeves, and H. Schulzrinne. Real-Time Communication in Packet-Switched Networks. *Proc. of the IEEE*, Jan. 1994.

[5] N. Malcolm and W. Zhao. Hard Real-Time Communication in Multiple-Access Networks. *Journal of Real-Time Systems*, Jan. 1995.

[6] B. Devalla, A. Raha, and W. Zhao. Guaranteeing End-to-End Delays in Computer Networks, 1997. *(In preparation).*

[7] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. Ph.D. Thesis, MIT, Cambridge, MA, 1992.

[8] R. Bettati. *End-to-End Scheduling to Meet Deadlines in Distributed Systems*. Ph.D. Thesis, University of Illinois, Urbana, IL, 1994.

[9] M. D. Natale and J.A. Stankovic. Dynamic End-to-End Guarantees in Distributed Real-Time Systems. *Proc. of the IEEE RTAS*, San Juan, Puerto Rico, Dec. 1994.

[10] R. Rajkumar, M. Gagliardi, and L. Sha. The Real-Time Publisher/Subscriber Inter-Process Communication Model for Distributed Real-Time Systems: Design and Implementation. *Proc. of the IEEE RTAS*, Chicago, IL, May 1995.

[11] A. Raha, S. Kamat, and W. Zhao. Guaranteeing End-to-End Deadlines in ATM Networks. *Proc. of the 15th IEEE ICDCS*, Vancouver, Canada, June 1995.

[12] F. Feng, S. Kamat, and W. Zhao. Guaranteeing Application-to-Application Deadlines in Distributed Real-Time Systems. *Proc. of the 20th Conference on LCN*, Minneapolis, MN, Oct. 1995.

[13] A. Mehra, A. Indiresan, and K.G. Shin. Resource Management for Real-Time Communication: Making theory meet Practice. *Proc. of the RTAS*, Brookline, MA, June 1996.

[14] H. Zhang. Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines. *Computer Communications*, 1996.

[15] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource ReSerVation Protocol. *IEEE Network Magazine*, September 1993.

[16] D. J. Mitzel, D. Estrin, S. Shenker, and L. Zhang. An Architectural Comparison of ST-II and RSVP. *Proc. of IEEE Infocom'94*, Toronto, Canada, June 1994.

[17] ATM Forum. *ATM User-Network Interface Specification 3.1*. ATM Forum Manual, 1994.

[18] C. Lee, K. Koshida, C. Mercer, and R. Rajkumar, Predictable Communication Protocol Processing in Real-Time Mach, *Proc. of the Real-time Applications and Technology Symposium*, Brookline, MA, June 1996

[19] C. Li. *Design and Implementation of Network traffic management*. M.S. Thesis, Texas A&M University, College Station, TX, 1997.

[20] A. Raha, S. Kamat, and W. Zhao. Admission Control for Hard Real-Time Connections in ATM LANs. *Proc. of IEEE Infocomm*, San Francisco, CA, March 1996.

[21] F. Feng. *Integrated Traffic Control and Management for Hard Real-Time Applications in High Speed Networks*. Ph.D. Thesis. Texas A&M University, 1996.

[22] A. Raha. *Real-Time Communciation in ATM Networks*. Ph.D. Thesis. Texas A&M University, 1996.

[23] NetEx Group. *NetEx User Guide and Technical Reference*. Texas A&M University, College Station, TX, 1997.

[24] M. W. Garrett, and W. Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. *Proc. of ACM SIGCOMM*, London, Sept. 1994.

[25] O. Rose. *Statistical Properties of MPEG Video Traffic and their Impact on Traffic Modeling in ATM Systems.* University of Wuerzburg. Research Report Series. Report No. 101. Feb. 1995.

[26] C. L. Liu and J.W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, Jan. 1973.

[27] L. Sha, R. Rajkumar, S. Sathaye, Generalized Rate Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems. *Proc. of the IEEE*, Vol 82, No. 1, Jan. 1994.

[28] ABIS Task Force, *DoD Advanced Battlespace Information Systems Task Force Report*, 1996.

[29] Defense Information Systems Angency, *Baseline Common Operating Environment*, 1994.

[30] B. Chen, A. Sahoo, W. Zhao and A. Raha. Connection-Oriented Communication for Real-time Applications in FDDI-ATM-FDDI Heterogeneous Networks. *Proc. of the 17th ICDCS*, Baltimore MD, May 1997.

[31] L. Sha, S.S. Sathaye and J.K. Strosnider. Scheduling Real-time Communication on Dual-link Networks. Proc. of IEEE RTSS, December 1992.
[32] F. Feng, C. Li, A. Raha, S. Yu, and W. Zhao. Modeling and Regulation of Host Traffic in ATM Networks for Hard Real-Time Applications. *Proc. of 21st IEEE LCN*, 1996.

6. Author Biographies

Badari Devalla received B.E. in Electronics and Communication Engineering from Anna University, Madras, India in 1991 and M.S. in Electrical Engineering from Texas A&M University, College Station, TX in 1994. He is currently a PhD student in the Computer Science department at Texas A&M. His primary research interests are Quality of Service guarantees in high speed networks.

Cen Li received B.S. in Physics from Tsinghua University, Beijing, P.R.China in 1991 and M.S. in Physics from Texas A&M University, College Station, TX in 1996. He is currently a M.S. student in Computer Science department at Texas A&M. He has been working on the design and implementation of NetEx.

Anirudha Sahoo holds a B.S. in Electrical Engineering from Regional Engineering College, Rourkela, India, and an M.S. in Computer Science from Univ. of SW Louisiana, Lafayette, LA. He is currently a Ph.D. student in Computer Science at Texas A&M. His research interests are ATM-based heterogeneous networks, Virtual LANs and LAN switching.

Dr. Wei Zhao received his B.Sc. in physics from Shaanxi Normal University, Xian, China, M.Sc. degree and Ph.D in computer and information science from the University of Massachussetts, Amherst, MA, in 1983 and 1986, respectively. In 1990 he joined the Department of Computer Science at Texas A&M University where he is currently a full professor. Dr. Zhao leads the Real-Time Systems Research Group and Intelligent Networks for Mission Critical Applications. His current research interests includes real-time computing and communication, distributed operating systems, database systems and fault-tolerant systems. He has published over 100 papers in journal, conference and book chapters.