Galileo: a Tool for Simulation and Analysis of Real-Time Networks

Edward W. Knightly and Giorgio Ventre *

International Computer Science Institute, Berkeley, CA 94704
and EECS Department, University of California, Berkeley, CA 94720

Abstract

Galileo is a flexible tool for simulation of heterogeneous real-time communication networks and for development and validation of network protocols. Galileo provides several unique features that make it particularly suitable for the simulation and analysis of networks that provide quality-of-service guarantees. First, its object-oriented programming environment provides the means for a modular, hierarchical, heterogeneous description of networks. Second, its multimedia device interface provides the tools for a qualitative analysis of network protocols. Finally, Galileo's network interface provides interaction with actual networks to access real data and simulate realistic multimedia scenarios.

1 Introduction

In this paper we present Galileo, a tool designed and implemented to address several problems unique to the simulation and analysis of real-time communication networks. The tool is based on Ptolemy [2], an object-oriented software environment that serves as a foundation for simulation of heterogeneous, hierarchical systems. Galileo has several unique features that distinguish it from the vast number of existing network simulators such as Netsim, REAL, and NEST.

First, Galileo is object-oriented. This feature, inherited from Ptolemy, effectively exploits the modular and hierarchical nature of communication network protocols and facilitates their rapid prototyping and testing. The simulation environment, together with Ptolemy's graphical interface, provides a block diagram representation of the network where modules may be encapsulated into higher level modules. This representation allows easy configuration of networks and network protocols. The object-oriented paradigm extends to the protocol code (written in C++) and facilitates an easy exchange of a protocol's algorithms. The modular description of networks also provides the capability of simulating the heterogeneity of internetworks. Because future networks will offer a wide range of communication architectures and services in a rapidly changing scenario, Galileo provides easy integration of new network models into the existing ones, and supports modularity in their development.

The second feature of Galileo is quality-of-service simulation with an emphasis on multimedia applications. The primary goal of the real-time network simulator is to show the quantitative effects of network management decisions on the traffic, and to verify both the correctness and efficiency of the resource allocation schemes. In addition, Galileo facilitates the *qualitative* evaluation of protocols by interacting directly with multimedia devices and applications. For example, Galileo supports the sending of video packets across the simulated network showing the effects of the network on transmitted video streams. This feature is especially useful to the network designer because it provides the capability of analyzing the network's impact on the quality of the communication provided to end-users.

Finally, Galileo provides the tools for interacting with the network itself. For example, if a user wishes to simulate a video conference with another node on the network, Galileo reads the actual information on the current state of the network by interacting with the network protocols. With this information, Galileo is able to simulate the proposed video connection and to show the impact of the network on the communication. Thus, this feature provides the means for the study of realistic scenarios. Indeed, a tool for simulation and analysis of networks should not be isolated from the external world. The availability of actual data from the network components can dramatically improve the quality of the simulation results, particu-

*On leave from Dipartimento di Informatica e Sistemistica, Universitá degli Studi di Napoli "Federico II", Napoli, Italy

larly for multimedia applications, where precise models for traffic sources are still under development.

This paper is organized as follows. First, some issues unique to real-time networking and simulation are examined along with the Tenet protocol suite (under development by the Tenet Group at the University of California at Berkeley and the International Computer Science Institute). Second, Galileo's architecture is described in terms of its functional modules. Next, Galileo's main modules are described: the Simulation Module, the Multimedia Interface Module, and the Network Analysis Module. Finally, we present a brief example simulation and conclude.

2 Real-time communication

Real-time communication protocols must present mechanisms for the reservation and control of the network's resources. In this section, the general principles of real-time communication are presented along with the Tenet protocols, a real-time protocol suite.

2.1 General principles

The architecture of real-time (i.e., guaranteed performance) communication network protocols has two primary components. First, real-time protocols must partition the network's resources among various clients. That is, the protocols need an algorithm to allocate the network's bandwidth and buffer space to its clients according to their corresponding quality-of-service (QoS) requirements (as well as some pricing scheme). A series of tests based on analysis, measurements, or both are performed to check if sufficient resources are available to meet the client's QoS requirements. The calculations involve transforming the client's requirements for bandwidth, end-to-end delay, delay-jitter, and loss probabilities into local node parameters so that each node of the network guarantees local performance, thus ensuring global performance. Providing statistical bounds on local performance requires calculating the effects of heterogeneous traffic sources (sources with different statistical properties and different priorities) on each other. The client's traffic characteristics (such as minimum and average packet interarrival times and an averaging interval) are also used in the calculations. The real-time channel may then be viewed as a contract between the client and the network. If the channel is accepted, the network guarantees the requested performance bounds provided that the client obeys its specified traffic characteristics.

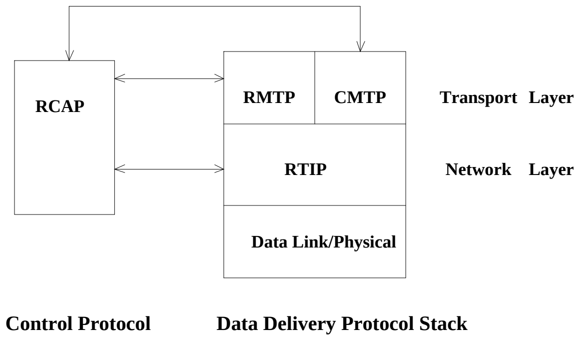

Figure 1: Tenet protocol suite

The second component of real-time communication is scheduling. A scheduling algorithm is needed that can provide the performance guarantees promised by the network. There are several options available, with the underlying requirement that packets are scheduled with some manner of fairness or priority. This is necessary to ensure that a non-real-time channel does not degrade the performance of a real-time channel and that a malicious real-time channel (one exceeding its promised traffic specification) does not degrade the performance of other real-time channels. The scheduling algorithm and admission control algorithm are interdependent with (for example) an HRR (Hierarchical Round Robin) scheduler requiring different tests than an EDD (Earliest Due Date) scheduler. Though the mechanisms and implementation details of various real-time protocols differ, these two components are inherent to guaranteed performance communication (see [4] for a survey).

2.2 The Tenet protocols

The Tenet real-time protocol suite consists of RMTP (Real-Time Message Transport Protocol), CMTP (Continuous-Media Transport Protocol), RTIP (Real-time Internet Protocol), and RCAP (Real-time Channel Administration Protocol) [1]. The logical structure of the protocols is shown in figure 1. Currently, of the four Tenet protocols, Galileo simulates RTIP and RCAP. The primary functions of RTIP and RCAP are as follows:

- RTIP is a simplex, sequenced, unreliable, guaranteed-performance packet service providing rate control, jitter control, packet scheduling, and data transfer. It also supports best-effort traffic in a separate, lower priority, FCFS queue.
- RCAP performs the admission control tests to provide mathematically provable guarantees for

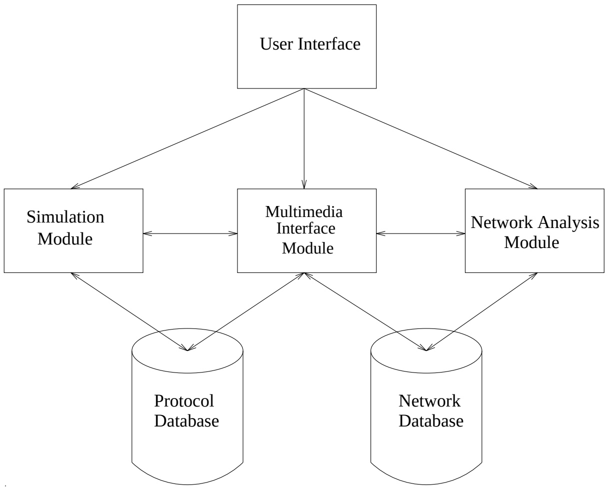

Figure 2: Galileo architecture

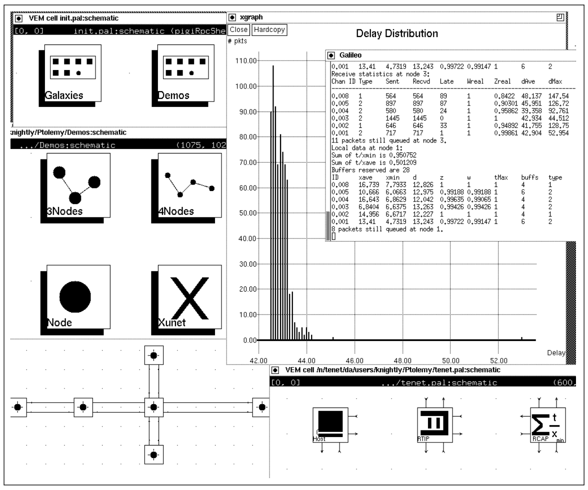

Figure 3: Graphical user interface

throughput, end-to-end delay, delay-jitter, and loss rates.

Galileo is flexible enough to support a variety of communication protocols (both real-time and non-real-time) in addition to the Tenet real-time protocols. Irrespective of the particular protocols involved, the unique requirements of simulating real-time traffic are the same. The simulator must provide quality-of-service validation in addition to performing admission control calculations. For real-time applications, this QoS validation must be qualitative as well as quantitative. Furthermore, the need for an object-oriented programming environment is also evident. With the current variety of algorithms for network functions such as admission control and scheduling, an object-oriented environment allows a comparison of these algorithms with minimal programming difficulty. In addition, these algorithms must coexist in order to simulate heterogeneous networks. The sections below describe how these simulation needs are met.

3 Galileo's architecture

Galileo's architecture is shown in figure 2. At the highest level is the graphical user interface shown in figure 3. The interface, provided by Ptolemy, allows networks to be created graphically by placing and connecting icons. The User Interface interacts with three modules: the Simulation Module, the Multimedia Interface Module, and the Network Analysis Module.

The Simulation Module consists of a library of programming modules such as protocols and network architectures. These programming modules are encapsulated and interconnected to simulate an entire network. The software environment is object-oriented — a feature inherited from Ptolemy and C++.

The Multimedia Interface Module provides the means for qualitative analysis of network protocols. For example, Galileo shows the effects of the network on a video stream in terms of delay, delay-jitter, and packet loss. These effects may be qualitatively analyzed for different network topologies, loads, and protocols to provide a more thorough study of how proposed algorithms affect multimedia traffic.

The Network Analysis Module provides the means for better simulating real networks by providing an interface to the actual network testbed via its control protocols. The Network Analysis Module gets the most recent information on the current state of the network from the local host (the node of the testbed on which Galileo is running). The local host will be required to have some information about the state of the network in order to make routing decisions. If the information provided by the host is insufficient for the required simulation, the Network Analysis Module will interact with the control protocol to probe the network for further state information.

The final components of Galileo's architecture are the Protocol Database and the Network Database. The Protocol Database provides the foundation for the simulations by defining the protocol algorithms. The Network Database stores the information obtained by the Network Analysis Module.

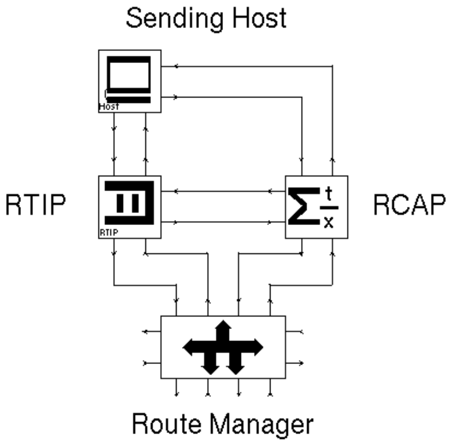

Figure 4: Tenet node

4 Object-oriented Simulation Module

Galileo's software foundation, Ptolemy, provides an object-oriented programming environment. Ptolemy performs most of the underlying functions needed in any discrete event simulation such as a graphical interface, discrete event scheduling, and efficient handling of packets. On top of these, Galileo provides functions specific to the simulation of communication networks. An example of the difference in the functions that are performed by Galileo and Ptolemy is the following. To send a packet, Galileo defines the packet's structure and when and where the packet is sent. Ptolemy schedules the packet (as an event in the simulation), provides reference counting for the packet, provides the means for sending the packet among protocols and nodes, and ensures efficient use of memory as packets are created and destroyed. This environment allows the Galileo programmer to be concerned with network issues such as channel establishment routines rather than with simulation issues such as event scheduling. C++, the programming language of Galileo and Ptolemy, provides the underlying support for object-oriented programming.

4.1 Coarse-grain modularity

Galileo's modularity allows a block diagram description of the nodes within the network as well as the protocols within the nodes. Figure 4 shows the block diagram configuration of a Tenet node. The interaction among modules in the node when creating a real-time channel can be described as follows. First, the sending host sends an establish request message to the node's RCAP. This message is sent to the RCAP in each node along a round trip from the source to destination and back. If the request is accepted, the RCAP at each node sends its RTIP the local guarantees. The sending host then sends data packets to the network via RTIP. The function of the Route Manager is to send a received packet to the node's appropriate protocol (RCAP or RTIP) or to the Route Manager of the next node in the packet's path. This model of the node provides a network layer (as in the OSI model) simulation along with added control by RCAP, and a higher level host to generate the traffic. Galileo is currently being extended to do transport layer simulations by designing the appropriate transport protocol modules, RMTP and CMTP, which will be placed between the Host and RTIP.

The modular nature of Galileo provides several advantages. First, at the highest level, a network is easy to create. The lower-left part of the graphical user interface shown in figure 3 shows the Galileo schematic of a six node network. This network was created by selecting (via the graphical interface) the appropriate node modules and interconnecting them. Creating networks with new topologies is as simple as connecting a block diagram and giving each block its appropriate parameters. Second, it is relatively easy to integrate new protocols. A new network layer protocol may be tested by removing the RTIP module in figure 4 and connecting the new protocol in RTIP's place. In a similar manner, nodes may be created with very different characteristics than the Tenet node. For example, another node in the network may support only IP and may not be able to support real-time protocols. The effects of real-time traffic traversing non-real-time nodes may then be studied.

4.2 Fine-grain modularity

Galileo's modularity also extends to the programming environment. For example, the default scheduling algorithm in RTIP is EDD. A second algorithm, Rate-Controlled Static-Priority Queueing [3], will also be used in the future. In Galileo's object-oriented programming environment, changing the scheduling algorithm only requires changing the scheduling object. This environment, coupled with Ptolemy's highly parameterized block-diagram representation, provides the means of choosing the scheduler for each node at run-time.

In sum, the block-diagram representation of the network and of the protocols within the network provides a flexible framework for rapid development and validation of network protocols. The modular nature of the programming environment extends this flexibility to the algorithms within the protocols. Such an environment allows simulation of heterogeneous networks such as complex internetworks.

5 Multimedia Interface Module

As communication networks begin to provide real-time guarantees, multimedia applications such as interactive video conferencing will utilize this service. The conference participants will have requirements for bandwidth, delay, delay-jitter, and loss rates. The application program may make efforts to improve the services offered by the network. For example, it may use buffering to absorb some delay-jitter or use (necessarily fast) signal processing to make up for dropped packets. The primary purpose of Galileo is not to simulate the final application but rather to show the raw effects of the network on multimedia traffic. Such a qualitative analysis will provide the network designer with a measure of how well the protocols are supporting a practical real-time load. The application designer will also benefit by using the qualitative analysis to better analyze the underlying service the network is providing.

Galileo's multimedia interface module provides the tools for qualitative analysis of network protocols. For example, it can show the effects of the network on a video stream's delay, delay-jitter, and packet loss. Currently, an interface is provided to display stored video. Work in progress includes displaying video from a camera and playing received audio.

5.1 Video

Two methods of displaying a simulated received video stream are the following. In the first method, the simulator displays the received video in real *simulation* time, displaying received video frames as the simulator processes them. We defer this method to the arrival of faster hardware. In a second method, the simulator stores the necessary information on disk, and recreates the received video stream when the simulation is complete. In either case, the most important result is to show the *raw* effects of the network on the video stream. To investigate the improvements of the application on the video stream, application modules may be added to Galileo. For example, an application module could be added that interpolates among missing missing pixels and attempts to smooth some of the network's delay-jitter with buffering. In this respect, Galileo can show the received video stream at several levels from the raw network stream to the final application stream. Such features better facilitate a qualitative analysis of the effects of protocols, traffic, and network configuration on the video stream.

5.2 Audio

Although the bandwidth requirement of an audio stream is not as stringent as that for video, a qualitative analysis is still useful. For example, the audio stream's maximum and average end-to-end delay will determine if a conversation is feasible. Also, the stream's delay-jitter will provide the destination application with information about buffer requirements and synchronization requirements. Audio streams will also be a valuable platform for analysis of probabilistic QoS guarantees. In this type of simulation, the final analysis on the acceptability of the received QoS is most easily judged when the received audio stream is played on the workstation speaker.

6 Network Analysis Module

This section describes the final component of Galileo's architecture, the Network Analysis Module. This module provides the means for a detailed analysis of a network testbed. With the testbed running the protocols of interest and the simulator running on one of the nodes of the testbed, an interaction between the simulator and the real network is useful to more quickly and easily explore the large parameter space of traffic models and protocol algorithms and mechanisms.

As described earlier, one of the most important features of Galileo is that it allows a complete understanding of the effects of transmission of continuous media over a real-time communication network. This effect can be obtained through simulation in two different ways. The first is via integration of models of network components, such as gateways and hosts, with models of sources of multimedia traffic. In Galileo this is accomplished by means of the Simulation Module and the information stored in the the Protocol Database.

The second solution is to simulate networks using real, updated data about the current state of the *physical* [1] network testbed. This data may be collected in the physical network through the Network

[1] we use the term *physical* to denote real as opposed to simulated components unless the distinction is clear from the context.

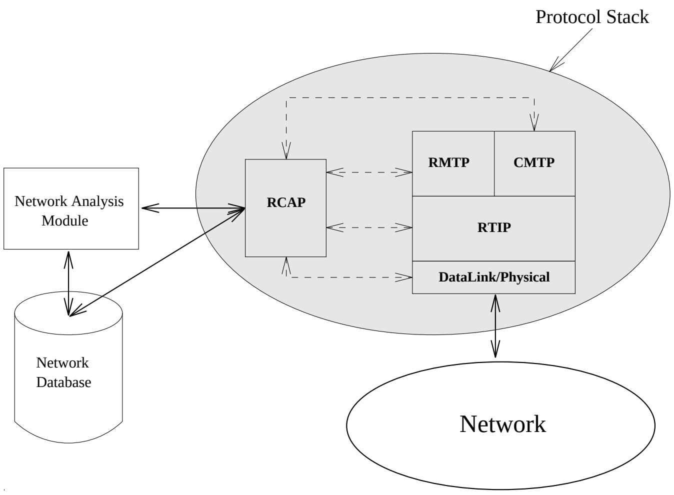

Figure 5: Network analysis module interactions

Analysis Module. Figure 5 shows how this feature is implemented for the Tenet real-time protocol suite. The Network Analysis Module was designed explicitly to allow Galileo access to the information regarding the physical network's traffic. This information is obtained directly through interactions with the local modules of the real-time protocols available on the node where Galileo is running.

There are two major issues related to interfacing the simulator with the physical network. First, which mechanisms are needed to provide the Network Analysis Module with updated information from the network? Second, how can the communication overhead be limited so that the gathering of information does not have a measurable effect on the network traffic? To solve these problems we propose a solution composed of two steps: *Network Database analysis* and *direct data collection*.

The Network Analysis Module accesses information concerning the state of the physical network through RCAP, the protocol responsible for the establishment and management of real-time connections. RCAP collects information both during the establishment and entire lifetime of a connection. The analysis and collection of this information and the use of this information by Galileo is described in section 6.1. If the information collected by Galileo from the Network Database is insufficient for the required simulation, Galileo will collect further information from the network by sending a status request message to the node of the network that has the required information. The method of further probing the network is described in section 6.2.

6.1 Network Database analysis

In the Tenet protocols, the establishment of a new real-time connection requires a number of tests in each node along the path from the source of the connection to the destination. The tests are required to verify the availability of the resources needed to guarantee the specified QoS. In order to exploit the information produced during these tests and to limit the amount of state-related information to be exchanged periodically in the network, Galileo uses the information produced by the existing channel establishment and management algorithms. That is, when channels are established from a node in the network, an RCAP control message makes a round-trip from the source to destination and back. During this round-trip, the message collects data (e.g., current utilization and reservation information) regarding the state of each node along the round trip. This information is then stored in the network's nodes so that other network mechanisms (e.g., routing schemes) may utilize it. RCAP collects data not only during the channel establishment tests but also during the entire lifetime of the connection. A summary of such information is also communicated to other interested network nodes (e.g., network gateways).

The information already available from the RCAP control messages represents the initial data to be included in Galileo's Network Database. The data is updated automatically whenever it is required, i.e., each time a new connection is established or modified and requires a different resource allocation in a node.

Thus, the first step in data collection involves Galileo utilizing information that is already available. The advantage of this limited distribution approach is that it reduces the communication overhead for maintaining updated information. However, this approach does influence the availability and the reliability of the data stored in Galileo's Network Database. When a simulation with actual data is required, Galileo's Network Analysis Module accesses the Network Database to get the information needed for the simulation. If the data concerning all the nodes in the simulation is available, Galileo can begin the simulation immediately. Otherwise, the second step in the network data collection has to be performed. This step involves the *direct* collection of data by Galileo.

6.2 Direct data collection

In this mechanism, Galileo tries to access the information produced by the protocols stored in nodes (of the physical network) other than the node on which

Galileo is running. When the Network Analysis Module requires information regarding the status of certain nodes, a message is sent to the local instance of the channel administration protocol (RCAP) to request the transmission of such data. The local RCAP forwards this message to the remote nodes and stores the additional data received in the Network Database.

This mechanism appears to be very efficient, since the transmission of information related to the state of a node is required only if a request is made. Furthermore, the request is limited to the nodes whose status is of interest to the Network Analysis Module. However, in this approach, the amount of time needed to gather the information is greater than the time required to access only the data in the Network Database, and it explicitly requires additional communication in the network for the collection of the data.

Even though the implementation alternatives presented here are dependent on the algorithms and the architectural solutions adopted in the Tenet real-time scheme, it should be noted that the same ideas can be applied to other real-time network architectures. The only condition is that the protocols available in such architecture are able to produce and gather reliable information about the current load in the network components. In order to improve the capability of Galileo to cope with networks adopting different approaches to guaranteed QoS in real-time communication, we plan to extend the Network Analysis Module to interact successfully with networks adopting different schemes for controlling and managing the resources allocated to real-time traffic.

7 Example

A simple six node network is used to demonstrate Galileo's capability to perform QoS simulation. The topology of the network is shown in the lower-left part of figure 3. In this simulation, channels (both real-time and non-real-time) are established between various nodes in the network. When a sending host receives an establish accept message from the network, the host begins to send packets according to a bursty two-state Markov process that obeys the client's agreed peak and average bandwidth constraints.

7.1 Quantitative analysis

The graphical user interface shown in figure 3 provides an example of the quantitative results provided by Galileo. The figure shows tables which give the

Original

Received

Figure 6: "Before" and "after" frame of a best-effort connection

statistics of each real-time channel as well as an automatically generated delay histogram of one of the real-time channels. The histogram shows quantitatively the effects of the jitter control scheme. That is, it shows the small delay-jitter received by a real-time channel even in the presence of heavy, bursty, cross-traffic.

With Galileo's ability to choose functional algorithms at run-time (when the object is created), experimenting with the large parameter space is greatly simplified. For example, if all incoming requests for real-time service are denied with a reported reason of insufficient buffer space even though the simulations show that the buffers are under-utilized, we would want to know which assumptions went wrong. Entities such as the admission control algorithm, the scheduling algorithm, traffic models, and many other mechanisms and parameters may be changed at run-time with parameters in the input file or via the graphical interface, providing a quick and insightful analysis.

7.2 Qualitative analysis

Multimedia services such as video and voice will be an important application for real-time protocols. Although a quantitative analysis is essential for validation of the protocols, a qualitative analysis is important for understanding how the real-time service will affect multimedia communication.

Figure 6 shows the effect of sending a best-effort video frame through the network in the presence of heavy cross traffic. In this simulation, a packet of unit size contains 48 bytes of data as in an ATM cell. Each pixel is 1 byte of data, so that a dropped packet causes 48 pixels to be lost. In this simulation, the real-time channels channels received all of their packets on time so that the received frame is identical to the sent frame. The degraded frame of the best-effort channel represents a non-real-time connection such as IP. Of course, an IP connection may resend the lost packets so that eventually a full image is received. However, if the application has performance requirements (e.g., it is interactive video), this excessively delayed information must be considered lost.

Current work by other members of the Tenet group includes extending Galileo to support multicast. With this addition, Galileo will be further equipped to simulate real applications such as multi-party video conferencing.

8 Conclusions

Galileo is an object-oriented tool designed and implemented to simulate real-time communication networks and protocols. We believe that Galileo provides several unique features necessary for analysis of real-time networks. First, the simulator is object-oriented. This provides a modular description of networks and network protocols and allows new protocols and algorithms to be easily redesigned and tested. Second, Galileo provides the multimedia interface required for a qualitative study of the network. For example, to analyze the quality-of-service of a video channel, it is essential to show the received video stream and observe it's delay, delay-jitter, and loss rates. Finally, Galileo provides an interface to real network testbeds. This feature facilitates a realistic analysis of the network by using the network's current state as simulation parameters.

Acknowledgments

The authors are especially grateful to Domenico Ferrari, Hui Zhang, and to all members of the Tenet Group for their ideas, comments, and criticisms.

This research is 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, Digital Equipment Corporation, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute.

References

[1] 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. Also to appear in *Computer networks and ISDN systems*.

[2] E. Lee and D. Messerschmitt. *The Almagest: Manual for Ptolemy*. University of California at Berkeley EECS Department, 1992.

[3] H. Zhang and D. Ferrari. Rate-controlled static priority queueing. In *Proceedings of INFOCOM'93*, San Francisco, California, March 1992.

[4] H. Zhang and K. Srinivasan. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–122, Zurich, Switzerland, September 1991.