TOURING MACHINE:
A Software Infrastructure to Support Multimedia Communications

Mauricio Arango, Peter Bates, Gita Gopal, Nancy Griffeth, Gary Herman, Takako Hickey, Will Leland, Victor Mak, Lillian Ruston, Mark Segal, Mario Vecchi, Abel Weinrib, Sze-Ying Wuu
Bellcore, 445 South St., Morristown, NJ 07962-1910

The Touring Machine project comprises a series of systems experiments that address key technical questions important to realizing a public communications infrastructure in support of multimedia applications. One of the major design objectives is to create a software infrastructure that investigates potential uses of modern distributed-computing paradigms to cope with the increasing complexities of large-scale telecommunications software. Another goal is to develop and study a flexible Application Programming Interface (API) that supports rapid deployment of new multimedia services for a variety of wide area networks. The current version of Touring Machine, representing the second iteration of system design, includes desk-top video and audio devices connected through a network of multiple switches and other specialized hardware resources. The control software for this system can support a variety of applications to run on users' workstations. The system is used daily as a working prototype for facilitating research collaboration among several dozen users.

While hardware and network support for multimedia communications is an important and lively field of research, less work is being done on the software structure required to enable widespread development and use of multimedia communications applications in a public network environment. Traditional telecommunications call processing software, which implements communications services in the public telephone network today, has been found to be extraordinarily difficult to design, implement, maintain, and modify. In addition to anticipating the needs of emerging multimedia applications, the Touring Machine project is confronting the historical sources of complexity in communications control software. We are investigating new ways to structure software that provides flexible "connection management", "session control", and other functionality that facilitate better design and operation of communications applications. Our design considers many of the issues that are crucial if the system is to be used ubiquitously, such as fault tolerance, scalability, multiple administrative domains, and heterogeneity.

In contrast to other research projects in the area of multimedia communications systems, our primary interest is in providing the infrastructure, not in developing the applications themselves. Touring Machine provides a general Applications Programming Interface to the applications developer. This API supports a rich set of abstract capabilities of the system. The API has been designed in close collaboration with applications developers from the CRUISER™ and RENDEZVOUS™ projects within Bellcore, and has incorporated many concepts from work on BISDN signaling from the EXPANSE project. An important design objective in defining the API has been to impose a strict separation of *mechanism* from *policy*. Touring Machine does not impose *a priori* policies on applications; rather, the API provides mechanisms that can be used to realize the various policies required by different applications.

The Touring Machine API provides an abstract model of the capabilities of the system. The abstract model allows the application developer to concentrate on the logical specification of the application itself, instead of worrying about the physical realization of the communication service, such as routing across multiple switches and bridging for multi-party calls. The model also allows the application to have flexible and separate control of different transport media: audio, video, and data.

The abstractions supported by the API include client, session, connector, endpoint, port, and mapping. A *client* is application software that acts as an agent for a user of the system. A *session*

™ CRUISER is a TRADEMARK and SERVICE MARK of Bellcore. RENDEZVOUS is a TRADEMARK of Bellcore.

manages the communications interaction among multiple clients, representing the control relationship among them. The session provides mechanisms to enforce different policies (such as privacy) that are specified and agreed to by the clients on a per-session basis. The transport topology of a session is specified logically as a set of typed *connectors*: abstract multiway transport connections between logical ports (*endpoints*). A session may have one or more connectors per medium, with source and sink endpoints from participating clients. Each client specifies the physical *port* (the network-access resource) to which each of its logical endpoints is assigned, and may then *map* and *unmap* the endpoint to the port to share the port among multiple concurrent active sessions.

The API defines a set of messages that are passed between an application and Touring Machine. The functionality of the API can be divided into five categories, with their related messages. A more complete discussion and definition of the current Touring Machine API will be available in another paper. The five categories of API messages are: client registration, session establishment and modification, network-access control, name server query, and inter-client messaging.

The Touring Machine software is a distributed platform that realizes the API while attempting to incorporate the design objectives, such as scalability, mentioned above. The software consists of seven classes of objects which communicate by exchanging messages. The objects encapsulate various capabilities and system state. In our current implementation, the objects are realized as separate (heavyweight) UNIX™ processes.

Each of the objects is defined by the interface it supports--the set of messages that it exchanges with other objects to cooperatively realize the requests that clients submit using the API. The objects are: the station manager, which implements various *policies*, such as those for sharing of local resources among multiple clients registered at the same station, and policies for screening session origination and acceptance; the station object, which provides the interface that clients use to communicate with Touring Machine, using the API, and which manages the ports associated with a station; the resource manager, which coordinates the allocation and deallocation the resources of the system; various resource objects, which control the actual physical devices that realize the clients requests; the session object, which is dynamically created to coordinate negotiation among clients and maintains the logical state of the session; the transport object, which maintains the logical-to-physical mapping for the session; and the name server, which is the repository for static and dynamic system information.

The system is currently operational, supporting several applications that provide a basic audio/video communication service and a variety of simple data services to users. Other projects within Bellcore are creating more sophisticated applications which will soon be available. There are currently about 40 users of the system, with an expected growth to 120 users across multiple connected sites within Bellcore by early 1992. A number of other institutions are also in the process of installing Touring Machine as part of research collaborations with our organization.

Now that some of the basic software infrastructure has been created, we are turning to the many open research problems that remain. These problems include, among others: adequacy of the API as we and others gain experience writing application for Touring Machine; systems issues such as scalability, fault tolerance, and support for multiple administrative domains; methods to ensure security for the system and privacy for its users; and mechanisms for controlling interaction between different applications.

Rich Clayton, Rob Fish, Carlyn Lowery, Steve Minzer, and John Patterson collaborated with us in defining the Touring Machine API, and continue to provide valuable insight on applications-related and other issues. We thank Jane Cameron, Brian Coan, Dave Cohrs, Alex Gelman, Bob Kraut, Yow-Jian Lin, John Unger, and Doris Woods for numerous valuable discussions and other assistance.

™ UNIX is a registered trademark of UNIX Systems Lab. Inc.