

Multimedia Teleconferencing Control Gateway

by

Leo Shang-hua Chang

Submitted to the

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

in partial fulfillment of the requirements for the degrees of

BACHELOR OF SCIENCE

and

MASTER OF SCIENCE


at the

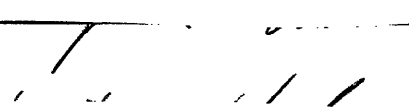
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

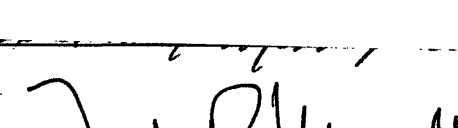
June 1995

(c) Leo Shang-hua Chang, 1995

The author hereby grants to MIT permission to reproduce and to
distribute copies of this thesis document in whole or in part.

Signature of Author 
Department of Electrical Engineering and Computer Science
May 26, 1995

Certified by 
John Wroclawski, Research Scientist
MIT Laboratory for Computer Science

Certified by 
William Mansfield, Director
Bell Communications Research

Accepted by 
F. R. Morgenthaler, Chair
Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUL 17 1995 Eng.

LIBRARIES

Multimedia Teleconferencing Control Gateway

by

Leo Shang-hua Chang

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 1995 in partial fulfillment of the requirements for
the Degrees of Bachelor of Science and Master of Science
in Computer Science and Engineering

ABSTRACT

Many current systems have computers supporting human-to-human interaction in real-time using multiple media. Such interaction can become complex, and multimedia teleconferencing control systems--or conferencing systems, for short--have been created to control the complexity and promote the development of applications. These systems provide media transport, transport resource management, and session management services to application developers.

Achieving interoperability between different conferencing environments is extremely difficult. One approach is to use a conferencing gateway, which is logically located between two systems and communicates with both. Gateways have the potential to provide interoperability between the systems without requiring modification to either and without loss of functionality to either user community.

A conferencing gateway was created to provide interoperability between two systems with very different environments--Bellcore's Touring Machine system and ISI's Multimedia Conferencing Control (MMCC) program. The conferencing gateway operates at the application level using a user proxy design. It processes control messages from both systems with cooperating sets of finite state machines.

The user proxy design was able to provide effective, though not complete, conferencing interoperability. No functionality or performance was lost to users of either system. The experiment revealed general conferencing system traits that benefit gateway solutions. Further research can uncover the potential role of gateways in a global conferencing solution.

Thesis Supervisor: John Wroclawski

Title: Research Scientist, Laboratory for Computer Science

Chapter 1

Introduction

Today's computing and telecommunications technologies have spawned research and development in the area of multimedia communications. Computers are used to support human-to-human interaction and collaboration in real-time using multiple media. Such interaction, which will be called teleconferencing, can become quite complex, and many systems have been developed to control the complexity and promote the development of useful applications. These systems must handle many tasks which can be separated into some general categories; media transport, resource management, session management, and application-level duties.

Media transport is the actual delivery of real-time media signals--e.g., video and audio signals--from one user's devices to another's. Resource management refers to management of media transport resources--e.g., switches, bridges, cameras, and monitors. Resource management includes such tasks as controlling transport resources, allocation of resources to particular activities, and making routing decisions. A session, sometimes called a "call," is the association of users and information involved in a conference. Although a session often refers to transport connections as well, session management deals only with the logical establishment, termination, and modification of sessions. That is, session management handles manipulation of the relationships and shared information among users in a session. Application-level duties include providing a user interface and implementing the user end of conferencing protocols.

1.1. Conferencing Systems

To most easily provide real-time, multimedia teleconferencing to users, transport, resource management and session management duties are handled by a multimedia teleconferencing control system--or conferencing system, for short. The conferencing system serves as a platform for application writers to use. The interface between the conferencing system and the applications allows application writers to easily use conferencing services. The conferencing system hides the complexity of implementing those services and lets different applications share the provided functionality.

Many conferencing systems have been created both for research and commercial purposes. Each of these systems has its own software control architecture. Each has its own set of control message protocols, session state management policies, and resource control mechanisms. For example, some systems use a centralized approach for session management[1,5,13]; others employ distributed methods.[3,10] Some provide other services such as directory servers for applications to access public information.

1.2. Interoperability

Increase in the popularity of teleconferencing is a phenomenon which industry and government efforts are anticipating. Providing functionality between all users--like today's telephony service--will be in the best interests of both providers and users. The existence of disparate conferencing systems leads to the question of how to provide conferencing between users of different systems. Conferencing systems will need to interoperate.

Because of the differences between control architectures, achieving interoperability between two conferencing systems appears difficult. Research-oriented conferencing systems were most often developed to examine the concepts and techniques necessary to provide useful communication. Each was designed with a specific semantic model of use and operating environment in mind--without consideration of how to accommodate interoperability with other systems¹. Commercial systems are developed, of course, to provide a marketable product. They usually have closed interfaces and proprietary protocols, which makes even the possibility of studying interoperability with such systems very difficult.

One approach for gaining interoperability is to develop general, flexible, and standard models and methods that all systems would have to be modified to use. This is a very difficult problem and research in this area is still in its preliminary stages.[6,9]

1.3. Conferencing Gateways

The interoperability solution explored in this thesis is the construction of a conferencing control gateway that is logically situated between two systems. The gateway needs

1. Perhaps due to the fact that "other systems" were often only in design or prototype stages, as well.

to reconcile the session and resource management styles and service provision mechanisms of the different control architectures. In addition, it needs to manage resources that connect the transport networks of the different systems.

The gateway is not a part of either system and neither system requires modification to use it. Gateway solutions can preserve functionality and efficiency enjoyed by conferencing system users and provide interoperability for important conferencing functions between different systems. Although gateway solutions cannot easily provide full interoperability for all conferencing functions, they do have the potential to part of an acceptable interim solution.

1.4. Thesis Plan

Chapter 2 presents conferencing systems in more detail, explores the solutions to the interoperability problem, and discusses the issues involved with developing design goals for a conferencing gateway. Chapter 3 describes the design of the gateway that was created for use between Bellcore's Touring Machine and Information Sciences Institute's Multimedia Conference Control (ISI's MMCC) program. Chapter 4 describes implementation details that were not covered in chapter 3. Chapter 5 evaluates the design and effectiveness of the resulting gateway, presents traits of conferencing systems in general that benefit gateway design, and discusses future work.

Chapter 2

Conferencing System Interoperability

2.1. The Conferencing System Model

The term “multimedia teleconferencing control system” is used to refer to many different kinds of systems. In this thesis, the abbreviated form--“conferencing system”--refers to an infrastructure that is provided to conferencing application developers. The interface to the conferencing system allows applications to easily provide real-time, multi-media, human-to-human interaction. Much of the complexity of establishing and manipulating such interaction is handled by the conferencing system. The conferencing system’s software implements the application interface. It also controls equipment that transports media among users. A piece of such equipment is called a resource, and examples include video capture cards, microphones, switches, and bridges. As shown in Figure 2.1, users interact with applications and can communicate with other users via the media transport resources. The environment of a conferencing system is often called its domain. That is,

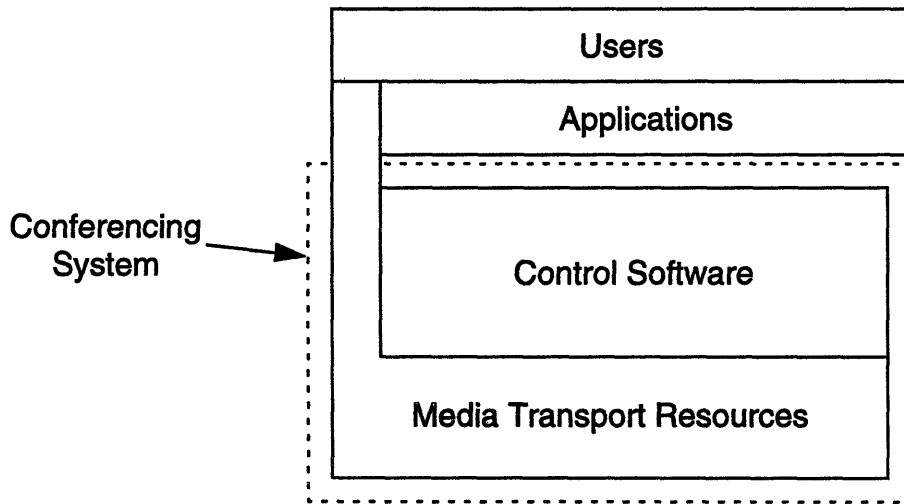


Figure 2.1. Logical location of a conferencing system

a system’s domain includes itself, its users, and the resources it controls.

Conferencing systems have strategies for session management and resource management. A session is analogous to a telephone call. It is a temporary association of users who can communicate with one another. Session management is the manipulation of

information associated with sessions--e.g., the participants and the media being used. The most complex parts of session management are negotiating among users about what kind of sessions to establish and ensuring that all participants are notified correctly about session information. Thus, protocols that initiate, modify, and terminate sessions are the heart of session management.

Resource management is the manipulation of transport resources. To realize the logical association in a session, resources must be chosen and controlled to physically transport media among users. Resource management translates logical media connections into allocation, configuration, and physical activation of the necessary resources.

Since the conferencing system handles session and resource management duties, applications can manipulate communication in the context of sessions. Applications can control transport resources with logical notions such as "a video session in which Bob is calling Chris." The conferencing system's session management handles negotiation and synchronization with the applications about session state--e.g., Chris' acceptance of the call. The system's resource management translates successful session establishment into media transport between Bob's and Chris' video equipment.

The idea of separating the conferencing system from applications is important. First, it allows for faster, easier development of different kinds of applications. The reason is that the conferencing system infrastructure isolates application developers from the complexities of handling all the details of multimedia communication.[1] Second, separating the control system from applications allows different kinds of applications to share the same set of hardware and control software. Application developers are given complete flexibility to use the control system. For instance, switches and mixers can be shared by a video conferencing application and a lecture application, or an application could provide both capabilities to the user. A video conferencing application that is integrated with the underlying control system would not allow for easy development of a lecture application that shares the same resources. The lecture application would have to have its own version of session and resource management. Third, even in a single application environment, separating the duties of the application and the control system provides easier management and maintenance of transport resources. Since the application need only deal with higher level notions of conferencing, transport resources can be updated or replaced

with modification only to the conferencing system's interface to those new resources.

The goal of conferencing gateways described herein is to provide interoperability between different conferencing systems. This chapter describes more precisely what this means. Section 2.2 gives some examples of conferencing system designs. Section 2.3 discusses tackling interoperability problems between systems--including the use of conferencing gateways. Section 2.4 examines the issues involved with more concretely defining design goals of gateways. Finally section 2.5 describes work that is related to conferencing gateway design.

2.2. Conferencing System Examples

There are some design principles that almost all conferencing systems follow. The first is the recognition of session and resource management duties. The details of session and resource management are always separated, but the implementation of session management must be linked to how resource information is gathered and reconciled.

The second characteristic that most systems share is the separation of "vertical" and "horizontal" protocols.[6] Horizontal protocols are used between logically remote entities--e.g., client-server, peer-peer. Session management depends heavily on the use of horizontal protocols. Vertical protocols are used within the conferencing system, allowing higher level parts of the system--e.g., session management--to communicate with lower level ones--e.g., transport resources.

The third common trait is the design assumption that transport resources can be replaced with more capable versions in the future. The resources themselves are most often described as separate from the heart of the conferencing system. The system's lower level interface to control the resources is separate from its session management and resource management, allowing the control architecture to be stable even as it incorporates improved transport technology.

Despite these shared design ideas, conferencing systems can differ in the structure of their software control architectures and their intended conferencing environments. Software architectures can range from being centralized and monolithic to distributed and replicated. Communication among session management and resource management entities can be peer-to-peer or client-server. The choice of architecture is influenced by the

system's intended operating environment. For example, using a very centralized architecture for conferencing among geographically separated users can lead to unreliable or slow performance.

Conferencing systems also differ in terms of the set of conferencing functions each offers. For example, systems have different kinds of servers that applications can query for information. Also, the interfaces presented to the application by different systems can have varying degrees of flexibility. For instance, some interfaces may allow description of only symmetric audio and video sessions, while others may allow asymmetric audio, video, and/or audiographics sessions.

The following sections present brief descriptions of several conferencing systems. The first two systems, Touring Machine and MMCC, are discussed in more detail here and in the next chapter since they are the systems involved in the control gateway that was created. The point of presenting these examples is to demonstrate that several different systems have been developed which fit the conferencing system definition used in this thesis. The fact they exist and have their own user communities introduces the problem of making them interoperate. Due to the differences among the systems' designs, developing conferencing gateways can provide insight into what system characteristics most affect the interoperability problem.

2.2.1. Touring Machine

The Touring Machine system consists of control software that is structured as a set of objects working together to provide the services supported by its application programming interface (API).[1] Touring Machine acts as a server, meant for a LAN environment, to application clients. It lets applications describe sessions in terms of connectors, each of which represent transport connections in a single medium between multiple endpoints. Endpoints are logical ports and are typed by medium, direction of flow, and owner. This scheme lets applications specify a variety of sessions. Sessions can have any number of participants because connectors can have any number of endpoints. Since endpoints are typed by direction of flow, asymmetric sessions of any configuration are allowed. For instance, an application can request an audio session with three participants who can speak and listen and two others who can only listen. Also, users can own an unlimited number of endpoints, so users can participant in simultaneous sessions using the

same media or even in one session with, for instance, multiple cameras.

Before using any of Touring Machine's services, an application must register at a station, which represents a client's workspace. Endpoints are also registered by the application. This registration information is made available to other applications by being stored in Touring Machine's name server database. The name server is actually an interface to a database that holds many types of user and session information. Applications and Touring Machine objects can all query the name server for information. In addition to audio and video media, Touring Machine also provides application-interpreted data streams and an inter-application text-message passing service.

Session management is handled in a centralized fashion by a session object. One session object is created for every active session. The single resource manager has information about the configuration of all transport resources in the environment. Resource objects--e.g., A/V switch objects and bridge objects--isolate the resource manager from the particular details of the hardware. The current implementation of Touring Machine has an analog transport network for audio and video. Vertical protocols are used completely within Touring Machine--e.g., between the session objects and the resource manager, between the resource manager and the resource objects, and between the resource objects and the actual resources. Station objects, which are the interface point for applications, use horizontal protocols to communicate with Touring Machine.

2.2.2. MMCC

MMCC actually includes an application-level user interface as well as session and resource management mechanisms.[11] Currently, both the application-level and control system duties are integrated into one program, but the separation between the two is documented and evident in the organization of the code. Applications can describe sessions primarily using participant lists and mechanisms for describing resource configurations in terms of media agents and their parameters. MMCC's media agents are tools that provide Internet media transport for a particular medium. MMCC's current version allows only symmetric audio and video conferencing and has no directory service analogous to Touring Machine's name server.¹

1. A symmetric "whiteboard" functionality with has not yet been added.

MMCC's session and resource management are completely distributed and rely on the Connection Control Protocol (CCP). CCP uses a peer-to-peer model for communication among distributed entities. MMCC instances send CCP messages (horizontal protocol) to one another to handle both session management and resource management issues. MMCC instances also send control messages locally to media agents (vertical protocol) which handle media transport. MMCC's distributed nature and its Internet-based media agents are tuned for WAN use.

2.2.3. Other systems

Hewlett-Packard's multimedia call system is a conferencing system with an API based on procedure calls.[5] It lets applications describe sessions in terms of Call objects, Party objects, MediaLines, and MediaPorts, which are almost semantically identical to Touring Machine's session objects, application instances, connectors, and endpoints, respectively. However, the call system's resource management is not as centralized as that of Touring Machine. The Call objects do communicate vertically with lower layers that control network resources; however, in addition, each Party has an associated Resource Manager--using vertical protocols to manipulate resources. Call objects communicate with Resource Managers and Parties with horizontal protocols. The call system has been implemented to control a local network of analog audio and video devices.

Xerox PARC's Etherphone conferencing system has application-level agents which reside at users' workstations communicating with horizontal protocols with a central session manager (called a connection manager).[13] As in Touring Machine, the connection manager communicates vertically with lower level agents which handle transport and resource management issues. Etherphone's conference model protocol has states that are semantically very similar to the session establishment protocols of Touring Machine and CCP--see section 4.3. Etherphone is also implemented on a LAN of workstations that control an analog audio and video network.

A conferencing system effort in the telephony environment is the work of International Telecommunications Union (ITU) Study Group 8. This group is in the process of making standards recommendations--the T.120 series of documents--for conference control.[8] T.120 is targeted at a specific network architecture--that of the telephone network.¹ Effectually, T.120 outlines a conferencing control system that uses the public

telephone network not only for media transport, but also for transport of control signals. Unfortunately, it does not accommodate the type of computer environments used by the other systems in this section--e.g., TCP/IP over LANs.

Vat and nv (which are MMCC's current media agents) are conferencing tools that manage audio and video communication, respectively. These popular tools can be used for conferencing by themselves; however, such conferencing does not incorporate a conferencing system as in Figure 2.1. Vat and nv do not provide an interface for other applications to use, and they do not work together for proper multimedia session management without higher-level coordination. They do, however, serve well as lower-level components of a conferencing system (such as MMCC) because they manage such duties as formatting media packets and controlling video capture cards and microphones.

Sd is an application that coordinates the use of conferencing tools such as vat and nv. It provides session management functionality by providing a dynamic list of advertised sessions to users who can join or advertise new sessions. Users can choose specific media for their sessions, and sd instantiates the appropriate conferencing tool(s). Sd is also does not fit the conferencing system model in Figure 2.1. Its session management scheme and user interface are closely coupled, and it does not provide an infrastructure for other applications to use.

2.3. The Role of Gateways

Because each conferencing system has its own protocols and models of operation, problems clearly exist for users of one system who want to conference with users of another. Even users of two systems that are very similar cannot interoperate. The most straight-forward solution to this problem is not to have interoperability among systems at all, but rather to have only one system used by everyone. The problem with this approach is designing such a system--or, at least, an appropriate set of rules that all systems must follow.

2.3.1. Standards Efforts

To address this problem, there are efforts underway to achieve interoperability by

1. For example, it can use the transport infrastructure outlined in the H.200 series of ITU documents.

defining standards to which all conferencing systems must conform. Unfortunately, it is difficult to design a model of conferencing--as well as a set of protocols and interfaces, etc.--that is both powerful and flexible enough for the variety of user needs and environments that exist today. Indeed, if a standard is too flexible in terms of implementation options, it loses its usefulness of being a standard. Consequently, current standards efforts either produce a system that is unaccommodating to many environments--see T.120 in section 2.2.3--or have trouble producing any guidelines at all. For example, the Internet Engineering Task Force (IETF) has chartered the Multiparty Multimedia Session Control Working Group (MMusic) to design and specify a protocol to perform the general session management for Internet multimedia teleconferencing. [6] The MMusic group has gathered much information about existing multimedia communication systems, but it has not been able to make significant progress toward its goal.

2.3.2. Gateway Solutions

If any of the current conferencing systems or a standard like T.120 becomes the dominant system for ubiquitous use, many existing systems would require major modification or replacement to be compatible. The users of those systems could lose many local conferencing services that they previously enjoyed.

A conferencing gateway sits logically between two systems and provides interoperability of many conferencing functions without requiring replacement or modification of existing hardware or control software. In addition, gateway solutions can provide interoperability while preserving local functionality and optimizations.

As will be discussed in section 2.4.2, gateway solutions can result in different levels of interoperability for different conferencing functions. In the long run, if multimedia teleconferencing becomes much more popular (as telephony is today), a solution involving gateways interconnecting many systems (much like electronic mail is today) may be satisfactory. Most likely, only basic functions will be interoperable between most systems--e.g., simple session establishment and termination. If full interoperability for all functions is important, gateway solutions will be an option for an interim solution. As such, they can help provide insight into conferencing among dissimilar environments which will help standards efforts.

2.4. Gateway Design

Gateway design goals focus on conferencing functions provided to applications. Inter-applications issues must also be considered. On a single system, users often run the same application to communicate with one another, and application interaction is avoided. However, in the inter-system case, each system already has its own set of applications, so interaction between different applications becomes more relevant. Also, when dealing with more than one system, the aforementioned inter-application issues are exacerbated by inter-system problems.

Interoperation models for gateway solutions are described in section 2.4.1. Section 2.4.2 presents a method for describing interoperability in terms of conferencing function behavior. Gateway design goals are specified in section 2.4.3, and section 2.4.4 describes how gateways must depend on applications.

2.4.1. Models of Interoperation

A gateway can be designed with two models of interoperation in mind. In the model used in this thesis, which will be called the existing-application model, the gateway can provide some interoperability between different applications residing on the two systems. This model has shortcomings. Applications can only use the conferencing functions common to both systems--i.e., only the intersection of the two systems' conferencing functions can be provided. Also, some services, such as application-to-application messaging, do not make sense, even if both systems provide them, because they involve application-level protocols.

A second interoperation model can address these weaknesses. The new-application model has application writers using the functionality provided by the gateway to create new applications that can be used across both systems. Services such as application-to-application messaging can be used. The problem of providing only common conferencing functions persists, but an adaptor that complements the gateway can be used on the deficient system, as in Figure 2.2.

Nonetheless, this second model has its own set of disadvantages. First, even if the scheme in Figure 2.2 were effective, having to create the adaptor eliminates a major advantage of building the gateway in the first place--the lack of modification to the con-

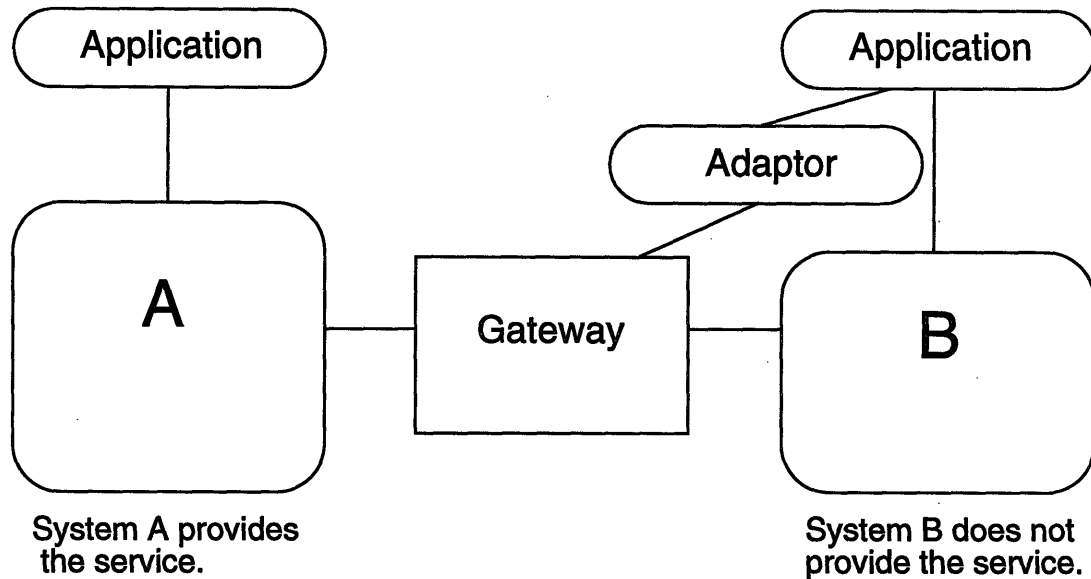


Figure 2.2. An adaptor in the new-application interoperation model.

ferencing systems. Further, it is not clear what kinds of services are feasible or even possible using adaptors. Second, this model does not apply to existing applications, only for applications that are written with the knowledge that the gateway has or will be implemented. Third, two versions of a new application must be written to work across both systems—one for each system’s application interface.¹ Fourth, the conferencing system-application interface is not always as open and well-defined as is required for the new-application model to be effective. Some conferencing systems do not have a well-documented interface, and some were built for one or more very specific applications. Some systems have a clean separation between infrastructure and applications in principle (and, sometimes, in documentation, as well), but their implementations do not exhibit a clear interface. All of these factors make an implementation of the new-application model more difficult to realize. Finally, interoperability issues can be more fully explored with the existing-applications model because application interaction plays an important role. When the implementations of applications and conferencing systems are not cleanly separated, inter-application issues become an unavoidable part of the interoperability problem.

1. Note that two versions of a new application can still be written to work with a gateway even though it was implemented with the existing-application model in mind.

2.4.2. Functional Levels of Interoperability

Conferencing function behavior can be described with levels of interoperability. More precisely, a certain functionality in a conferencing system is described by its behavior--as exhibited to applications--when interacting with another conferencing system. Such behavior can be classified into three categories. Interoperability Level 1 (IL1) refers to successful behavior that is transparent to the application in terms of correctness. Interoperability Level 2 (IL2) refers to behavior that invokes an error notification from the conferencing system to the application. Applications can gracefully handle such errors and notify users of the lack of functionality. Interoperability Level 3 (IL3) refers to unexpected behavior--i.e., behavior that differs from what is specified in the conferencing system's application interface. IL3 behavior, of course, is to be avoided as much as possible. It can include behavior as harmless as a lack of response to a request or as drastic as causing system or application software to crash.

Interoperability levels are defined only in terms of correctness of a conferencing function--e.g., not in terms of function speed--as seen by applications. The interoperability between two systems can be described with the interoperability levels of various functions. Also, interoperability levels apply only in the context of multiple systems interacting. It makes no sense to describe a function as having IL1 behavior in a single conferencing system. How well a service performs in the local domain is independent of its interactive behavior with another system.

2.4.3. Gateway Design Goals

With above issues in mind, the design goals of a conferencing gateway, using the existing-applications model, can be more precisely stated. The first goal, stated earlier, is not having to modify either conferencing system. Second, the gateway solution should not cause loss of any functionality--i.e., conferencing between users in the same domain should not be affected. Third, the gateway should provide IL1 interoperability for as many conferencing functions as possible, and IL2 behavior for all others. If there must be IL3 behavior, it can only be tolerated if it occurs very infrequently. For the gateway to be useful, IL1 behavior must be provided for at least those conferencing functions which are used frequently by users. Finally, users should be notified when to expect possible non-IL1 behavior--i.e., users should be made aware when they are requesting inter-domain

conferencing services.

2.4.4. Dependence on Applications

The last design goal is made possible by not only the gateway, but also by application writers. Direct communication with the user is the responsibility of the application. (Section 3.2.2 presents a simple way to reduce the gateway's reliance on applications for the Touring Machine/MMCC gateway.) IL2 behavior relies similarly on application writers. IL2 behavior is useless if a poorly written application does not handle errors in a way that is useful to users.

In addition, gateway solutions depend on applications following each conferencing system's application interface--in terms of both syntax and semantics. The interface description includes rules on how to use procedures or messages, what functions should be performed locally in association with each procedure or message, etc. Although interface specification documents may not be absolutely complete in this respect, the intended use of the interface is made clear.

Still, a little flexibility provided to the application can lead to misuse of the interface. For example, an application writer could choose to use an application-level messaging service, session names, or other text strings that can be passed between applications as a way to communicate session establishment information. For instance, an application could format its session names to convey some kind of participant information that signals other application instances to add other participants. In other words, although this application has not broken any syntactic rules of the interface, it is not using the conferencing system's standard method for session establishment.

A conferencing gateway's design goals and performance are based on the assumption that applications do not use such non-standard methods. That is, the gateway's design cannot be held responsible for interoperability problems that are caused by nonconforming applications.

2.5. Related Work

Conferencing systems like those described in section 2.2 form an important basis of work for conferencing gateways. The design similarities and differences among such systems form the basis of the conferencing interoperability problem and gateway design.

Protocol conversion is another important body of related work. A large part of conferencing gateway design involves reconciling disparate session management protocols. Protocol conversion work has included converter designs between specific protocols and attempts at developing general, automatable methods.[4,8,14] Network protocol converters often deal with lower level protocols relative to those handled by a conferencing control gateway. For example, many converters take approaches that involve attaching and detaching appropriate header material to convert between network formats.¹[4] Such approaches are not easily extended to conferencing gateway design because session management protocol messages have semantics that involve more than delivering data payloads.

Nevertheless, some ideas resulting from protocol conversion research are useful for the construction of the conferencing control gateway. Analysis from a service viewpoint leads naturally to the discussion of adaptors and provision of only common services in section 2.3.1.[2] In addition, such analysis resulted in the mapping of protocol messages between finite state machines (FSMs) much like a conferencing gateway's message processing described in section 3.3 and 4.3.

Approaches for automation of protocol conversion require protocols to be first described in terms of formally specified FSMs.[8,14] A method for describing the functionality of conferencing systems in terms of an appropriate formal specification language does not appear straight-forward. However, as both protocol conversion and conferencing gateways are better understood, gateway design may be able to benefit from general protocol conversion methods.

1. Electronic mail gateways use such an approach.

Chapter 3

Touring Machine/MMCC Gateway Design

This chapter describes a conferencing gateway design that attempts to meet the aforesaid design goals. The gateway was designed for Bellcore's Touring Machine and ISI's MMCC conferencing systems. The next section presents an overview of the gateway and gives a road map of the other sections, each of which describe a portion of the design. Implementation details of this design are presented in chapter 4.

3.1. Overview--the User Proxy Idea

The logical location of the conferencing gateway is pictured in Figure 3.1. It pro-

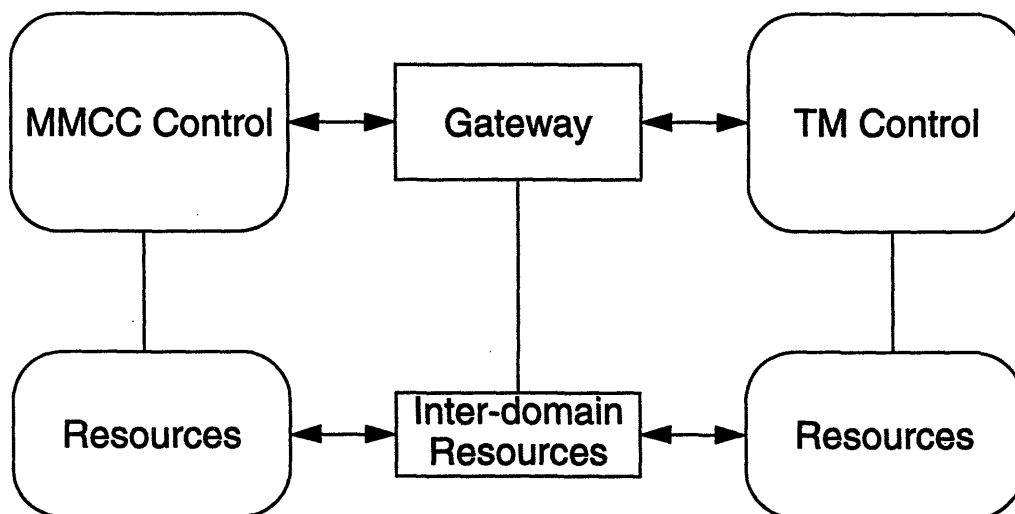
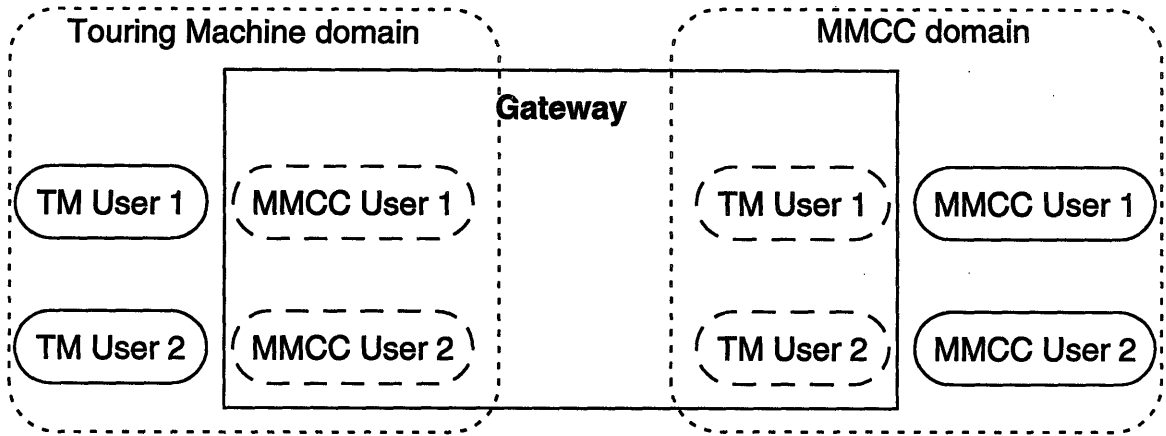


Figure 3.1. Gateway layout.

vides interoperability between the two systems without modification to either. The gateway communicates with each system at its application interface. Resources that provide media transport between the two conferencing domains are controlled by the gateway.

The key design idea for the gateway is the employment of a user proxy approach. From each system's viewpoint, the gateway looks like users of the other system. In other words, in one domain, the gateway is a proxy for those users that belong to the other domain. Consider Figure 3.2. Let X be the set of users in the original Touring Machine domain and Y be the set of users in the original MMCC domain. With the user proxy approach, each system is "fooled into believing" that it has the set $X \cup Y$ in its domain.



Original members of Touring Machine domain $X = \{ \text{TM User 1, TM User 2} \}$

Original members of MMCC domain $Y = \{ \text{MMCC User 1, MMCC User 2} \}$

Figure 3.2. User proxy approach.

Users of Touring Machine see the MMCC users as if they have actually been added to the Touring Machine domain, and vice versa.

To implement the user proxy design, the gateway is divided into five parts: a CCP message interface, a Touring Machine message interface, a CCP message processing unit (MPU), a Touring Machine MPU, and an inter-domain resource manager. The message interfaces intercept control messages that are normally meant for applications. Message interception is presented in section 3.2. Each message is parsed in order to discover the intended recipient and passed to the associated MPU. The MPUs contain finite-state-machines (FSMs) that play the principal roles as proxies. The MPUs are the heart of the gateway and are described in section 3.3. Finally, the gateway must play a role in realizing media transport between the domains. This is the responsibility of the gateway's inter-domain resource management, described in section 3.4.

3.2. Message Interception

In order to serve as a user proxy, the gateway needs a strategy to pose as a set of users in each domain. In the cases of both Touring Machine and MMCC, the gateway intercepts messages that are meant for represented users. This kind of interception involves manipulation of user information in each domain.

3.2.1. User Information

Both MMCC and Touring Machine provide mechanisms for an application to access information about users in the domain. Touring Machine has its name server database which applications can query. MMCC currently has an awkward scheme in which each instance has a copy of an initialization file. Future plans include incorporation of user directory services that would provide information about other users.

The gateway solution involves expanding the domain of each system to include the users from both domains. Therefore, information about MMCC users must be added to Touring Machine's user information, and vice versa. This information can include names, login names, electronic mail address, phone numbers, etc. Most important is the user's network address because each conferencing system uses this information to send and receive control messages. In order for the gateway to act as a user proxy, addresses for each newly added user must be modified to match that of the gateway. Thus, messages intended for users that are actually in the remote domain can be intercepted and processed by the gateway.

3.2.2. Naming

The names of users added from a different conferencing system can be modified in the local name space. The purpose of the modifications is to indicate that these users are not in actually in the domain. For instance, MMCC users could have "(MMCC)" appended to their names in Touring Machine's nameserver, and Touring Machine users could have "(TM)" appended to their names in the MMCC initialization files. Applications then have a transparent method for informing users when they are invoking inter-domain conferencing functions. This is important to users because they can experience inter-domain conferencing behavior that differs from that of local conferencing--see section 5.1. Furthermore, this naming scheme solves the problem of users in each domain with identical names.

3.3. Message Processing

When a message is parsed by the gateway, it is passed to the associated MPU. The message is actually passed to one of several finite state machines (FSMs) contained in the MPU. These FSMs can send messages to the corresponding conferencing system, and

internal messages can be passed among FSMs between the MPUs. Each MPU instantiates an FSM for a represented user for every active session in which he or she is involved.

The FSMs used in the MPUs of the gateway are based on those that applications use to implement their part of session management protocols. CCP's specification document describes its session establishment protocol with FSMs that applications should use. Touring Machine does not explicitly use FSM's in its API document, but FSMs can be designed to describe its session establishment protocol as well. These FSMs can be modified to communicate with one another within the gateway to allow the two protocols to interwork properly.

3.3.1. Session Establishment

The CCP and Touring Machine session establishment protocols have some similarities. Both are based on a two-phase commit scheme. Both have a stage in which an initiator constructs a session description which is communicated to all the callees. Both allow callees to accept or deny the request to join the session, and both have a second stage in which all parties are notified of the result.¹

Nonetheless, there are significant differences between the protocols. Because of Touring Machine's centralized control architecture, it has access to all resource information--e.g., network topology--and is able to perform resource management via internal vertical communication with media transport resources. In contrast, CCP uses horizontal control messages that deal with resource management issues. It employs messages that let callees communicate information about their media transport capabilities. Other messages are needed to deal with the distributed nature of MMCC's media agents.

Consequently, Touring Machine can process all data relevant to session establishment--callee consent, resource availability, etc.--and sends one positive or negative message to all pertinent users. CCP users, on the other hand, discover resource capabilities information, callee responses, and the success of media transport realization in separate stages of control message exchanges.

3.3.1.1. Message Semantics

Due to these protocol differences, CCP control messages do not always have a

1. See sections 4.3.1 and 4.3.2 for details about both protocols.

semantically equivalent Touring Machine counterpart. Table 1 shows a summary of important CCP and Touring Machine messages that have related semantics. The “signifi-

Table 1: Control Message Semantic Relationships

Touring Machine message	CCP message	Significant meaning
sessionCreate	Request	Caller sends this message to begin session establishment--check if transport resources exist for this session
	Connect	Caller sends this message to query callees for their approval
sessionActionRequest	Request	Callee receives this message as notification that the caller is initiating a session
	Connect	When a callee receives this message, the application should be queried for approval
sessionActionAccepted/ sessionActionDenied	Connectr (positive/ negative)	Callee sends this message to indicate approval/disapproval of the session
sessionActionCommit/ses- sionActionAbort	*Connectr(positive/ negative)	Caller receives this message to indicate that callees approve/disapprove with session
	*Statusr(positive/ negative)	Caller receives this message when media transport has begun/cannot begin
sessionActionCommit	Status (second one)	Callee receives this message when media transport has begun
sessionActionAbort	Disconnect	Callee receives this message when there are problems with media transport or callee acceptance

cant meanings” listed in the table are significant for designing inter-MPU behavior. Note that several Touring Machine messages are semantically related to more than one CCP message. For example, when a Touring Machine caller receives a sessionActionCommit message, he knows not only that all callees have accepted the session, but also that the associated media transport was successfully realized. A CCP caller receives two separate sets of messages that carry these two pieces of information.

3.3.1.2. Inter-MPU Design Choices

This kind of multiple semantic mapping between Touring Machine and CCP mes-

sages provides both flexibility and ambiguity when designing how FSMs will interact across MPUs. For instance, the designer of the gateway must choose to map Touring Machine's sessionActionCommit message to either CCP's Connectr or Statusr message. If it is mapped to the Connectr message, correctness can be sacrificed because a session may be established in the Touring Machine domain even if media agents are unsuccessful in the MMCC domain. Such IL3 behavior should be avoided. On the other hand, IL1 behavior can be achieved if sessionActionCommit is mapped to Statusr. Unfortunately, in this case, the amount of time for callee responses to reach the caller is lengthened. This added delay is especially costly because media agents are usually successful. The best solution--one that has faster IL1 behavior--is the mapping of sessionActionCommit to Connectr combined with session termination in the case of media agent failure. A detailed description of this scenario and its associated design choices are presented in section 4.3.3.

3.3.2. Session Termination

Both systems implement session termination by having each participant of the session remove itself until the number of participants falls below a meaningful value.¹ The gateway is involved only in inter-domain sessions--i.e., session with at least one participant from each domain. Therefore, if all of the session participants from either domain remove themselves, then the gateway deletes all of its FSM instances involved with that session.

The gateway FSM design for participant self-removal is straight-forward and described in section 4.4. However, IL3 behavior can occur in rare situations because of a particular policy in Touring Machine's participant self-removal protocol. Like Touring Machine's session establishment protocol, its participant self-removal protocol has a two-phase commit structure. If an error occurs while a client is trying to remove itself from a session--e.g., Touring Machine has an internal resource communication problem--Touring Machine leaves the client in the session. Because of this, if there is a problem with Touring Machine or the transport network, a client could conceivably be left in a session with no way of removing itself. This scenario occurs rarely. Still, this kind of policy for a con-

1. For Touring Machine, that value is one; for MMCC, it is two.

ferencing system not only introduces interoperability problems, but it also present clients in the single domain environment with a undesirable error-handling behavior. Certainly, if Touring Machine were deployed for public use, customers would not agree to be billed for calls that they could not leave.

The gateway depends on client self-removal for achieving simple, efficient, and proper session establishment and participant invitation (see section 3.3.3). It might be argued that it is poor gateway design to have these functions depend on a procedure that can “hang” clients. By that reasoning, though, it is poor design to ever establish a session in the first place because Touring Machine sessions may never be terminated. It is more proper to argue that the conferencing system’s design is the real reason for the lack of interoperability.

3.3.3. Invitation of new participants

Both systems allow session participants to invite new ones into the session. The process is very similar to session establishment in most respects. However, from the gateway’s viewpoint, participant invitation is fundamentally different from session establishment. Unlike session termination, which the gateway can perform in either domain to cancel the effects of session establishment, removing a recently invited participant can be problematic. It is a reasonable policy for a conferencing system (such as MMCC) to prohibit participants from removing others from a session. Even if a system provides a mechanism for removal of other participants (as Touring Machine does), the participant being removed is queried to accept such an action. Thus, it can be impossible for the gateway to unilaterally reverse participant invitation. Therefore, gateway strategy for participant invitation has the added requirement of being unable to cancel the effects of a successful invitation.

The Touring Machine/MMCC gateway handles this requirement except in rare situations of a particular scenario. The problem stems from Touring Machine’s policy of notifying uninvolved participants--i.e., participants of the original session except for the inviter--*after* an invitation has already successfully completed. If a Touring Machine client, who is already in session with at least one MMCC user, invites another Touring Machine client, the gateway represents at least one such uninvolved participant. Therefore, the gateway cannot take action in the MMCC domain until the invitation has already

occurred in the Touring Machine domain. If there is an error in the MMCC domain (which is extremely rare for reasons discussed in section 4.5.6), IL3 behavior results because the gateway cannot unilaterally remove the recently invited Touring Machine participant.

There is no possibility of IL3 behavior in the opposite scenario--when an MMCC user, who is already in session with at least one Touring Machine client, invites another MMCC user. The reason is CCP's distributed approach to resource management. The gateway must be notified of the invitation in order for the inviter to gather resource information before the invitation is finalized. Details of all participant invitation scenarios are presented in section 4.5.

3.4. Inter-domain Resources

Since the gateway represents users at the application level, it has access only to the limited amount of resource control that each system provides to applications. In the Touring Machine domain, the gateway can assign logical notions of media endpoints to particular physical ports. In the MMCC domain, the gateway controls software media agents which can be configured to use particular ports, as well. The gateway manipulates how endpoints and media agents share ports in order to realize inter-domain transport.

3.4.1. Touring Machine Endpoints

Touring Machine realizes media transport among clients by centrally controlling network resources--e.g., switches, bridges, and mixers.[1] When a session is established, Touring Machine connects endpoints owned by clients involved in the session. Applications can control which of its devices (Touring Machine sees each device as a physical port.) are assigned to each endpoint. For example, Figure 3.3 shows a client that has a station--Touring Machine's notion of a desk or workspace--with two different cameras. The client can control which of its cameras should be used for any particular video session. The endpoints in Figure 3.3 are logical notions used by Touring Machine's resource manager. The lines connecting each endpoint to a physical device represent assignments made by the application.

3.4.2. MMCC Media Agents

MMCC users invoke media agents which isolate the applications from the trans-

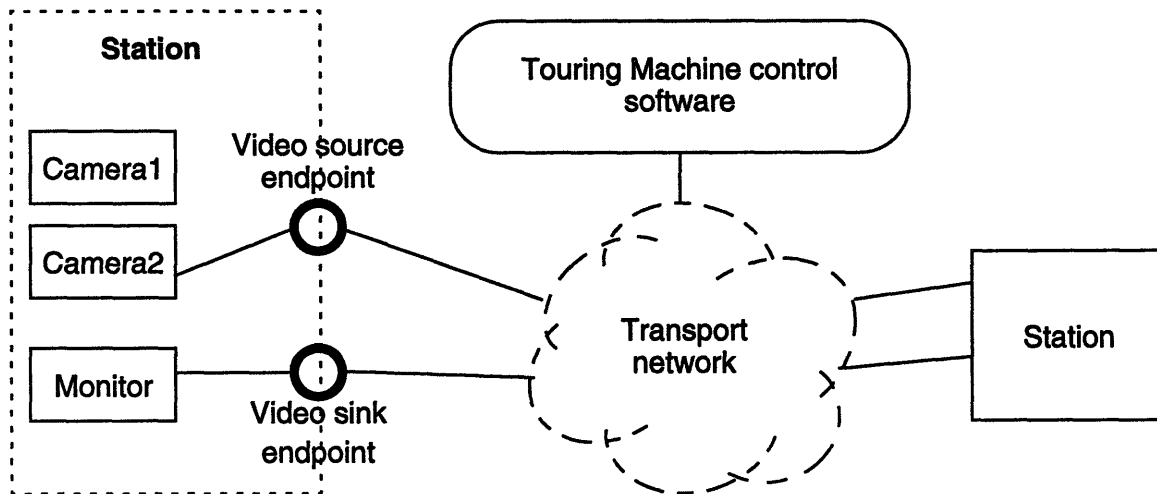


Figure 3.3. Touring Machine station video set up.

port details--see Figure 3.4.[11] To control multiple devices, a user can have several

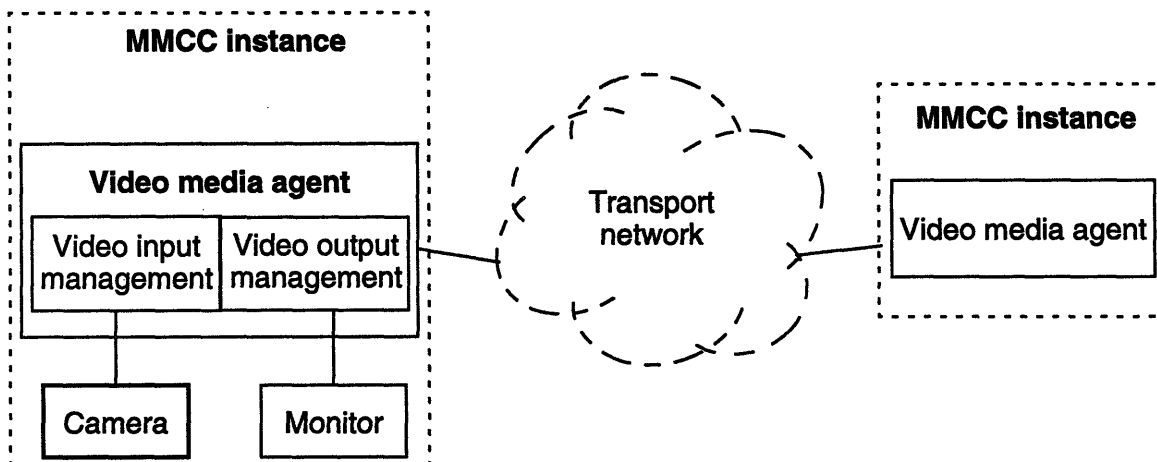


Figure 3.4. MMCC video media agent control set up.

media agents in use simultaneously. For example, to control two cameras and monitors, one user can invoke two video media agents to control each set of camera and monitor. When each media agent is invoked, it is assigned to a particular set of devices by the application.

3.4.3. Gateway Resource Duties

The gateway has an associated Touring Machine station which owns its set of endpoints, and it can invoke MMCC media agents. To realize media transport across the two domains, the gateway assigns Touring Machine endpoints to ports that can be controlled by MMCC media agents. Figure 3.5 shows how the gateway can connect the video trans-

port networks of the two conferencing systems.

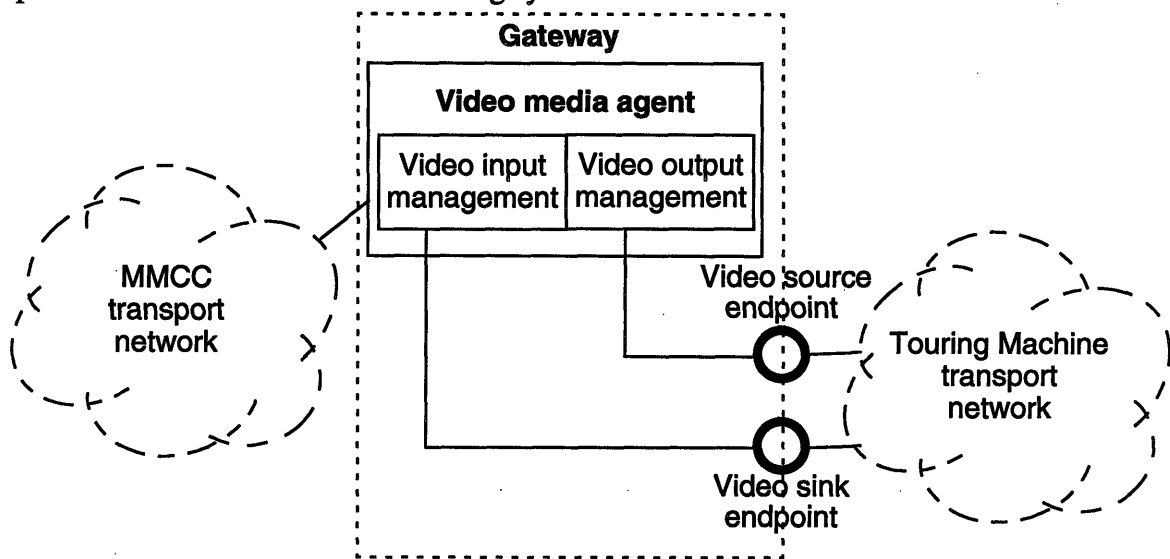


Figure 3.5. The gateway's inter-domain video resources.

The gateway does not perform any media bridging or mixing. Those tasks are handled by each conferencing system. The gateway simply transports already bridged or mixed signals between domains. For instance, suppose Figure 3.5 involves several MMCC users in a session with several Touring Machine clients. The signal at the gateway's video sink endpoint--the same signal that is transmitted to the MMCC transport network--is mixed or bridged by Touring Machine's resources. Similarly, the signal coming from the MMCC transport network into the gateway's video media agent is mixed or bridged by MMCC's media agents before being sent through the video source endpoint.

To simultaneously provide transport for more than one inter-domain session, the gateway must own multiple sets of Touring Machine endpoints and be capable of invoking multiple instances of media agents. Figure 3.6 shows how the gateway handles video transport for two simultaneous sessions. The gateway acts as a sort of simple software switch. The hardware associated with the gateway does not need configuring or any hardware switching because both Touring Machine and MMCC allow assignment of logical connections to physical devices during session establishment. The number of requested inter-domain sessions can exhaust the resource capabilities of the gateway--e.g., all audio/video ports are in use. Subsequent session establishment attempts can be simply aborted with negative `Requestr` and `sessionActionDenied` messages until resources are freed.

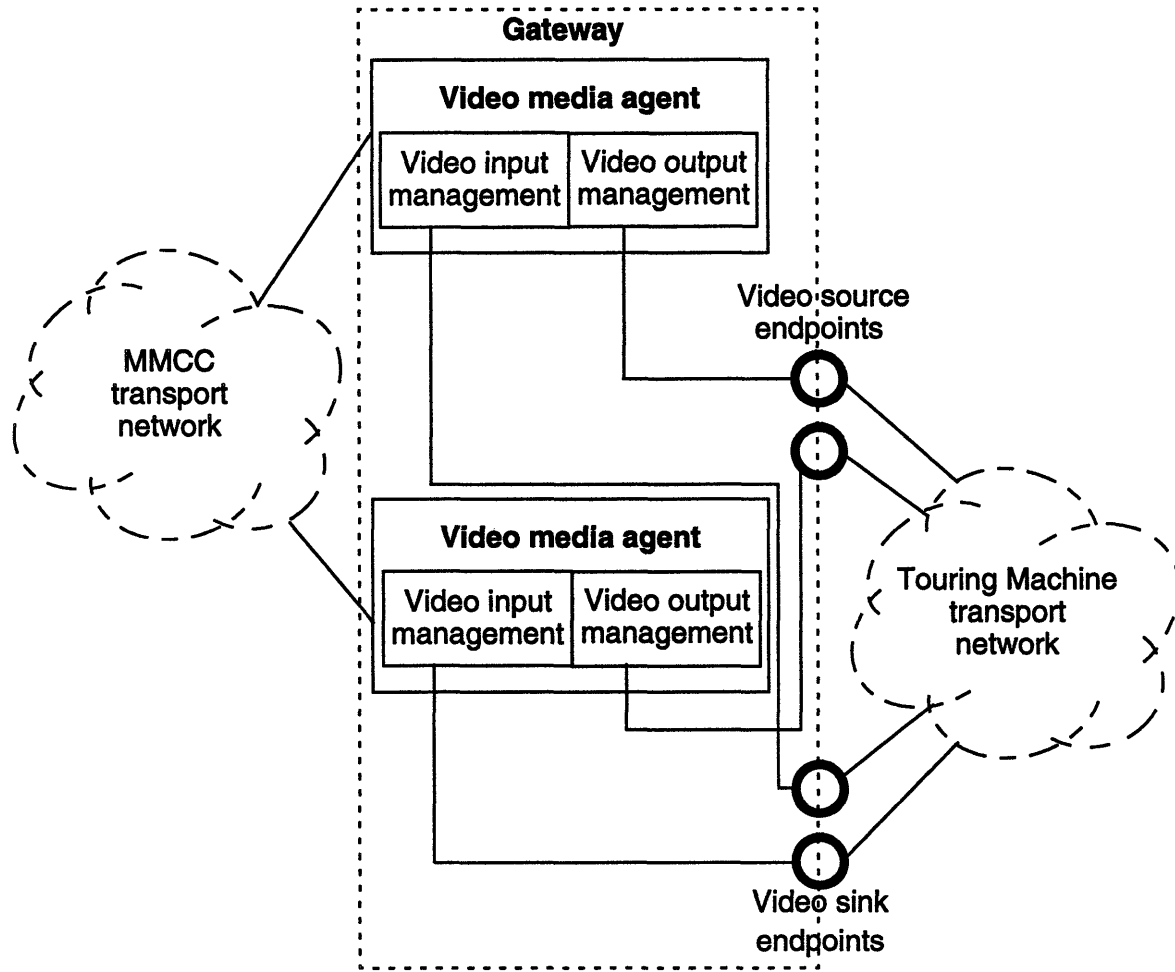


Figure 3.6. Simultaneous inter-domain video sessions.

Chapter 4

Touring Machine/MMCC Gateway Implementation

This chapter describes implementation details of the Touring Machine/MMCC gateway design that was described in chapter 3.

4.1 Message Interception

Touring Machine initializes its nameserver database with user information from a special file. In addition to names, login names, electronic mail addresses, phone numbers, etc., the file contains information about each user's station. The nameserver's station information includes a machine address and a list of media ports and types. When the gateway is deployed, each user in the MMCC domain is entered into the initialization file along with his or her office address, phone number, etc. Each MMCC user is assigned a special station--that of the gateway. The gateway's station information has the address of the machine on which the gateway will be running, and it will list all media ports that are controlled by the gateway. With this scheme, when Touring Machine sends a message to an MMCC user, it will use the information in the nameserver and actually send the message to the gateway's machine. The gateway's MPUs process messages based on the intended recipient. The gateway can identify whose message is being intercepted because each Touring Machine message lists the recipient's name in the message.

Currently, MMCC instances have an initialization file that lists each user's name, login name, machine address and communication port. Control messages are sent to the listed machine address and port associated with each user. All MMCC users must have identical initialization files, so insuring consistency when modifying these files is not an elegant process. On the other hand, the files are very simple, and initialization information does not change very often.

To have the gateway intercept MMCC messages meant for Touring Machine users, such users are simply listed in MMCC initialization files with the gateway's machine address. The gateway also has its own initialization file which is consistent with that of all MMCC users. Because MMCC's current version uses messages that do not contain the recipient's name, each Touring Machine user must be designated a unique port

on the gateway's machine. Otherwise, there is no way for the gateway to tell who is the intended recipient of an intercepted message. Having to assign these unique port numbers makes an inelegant process even less elegant. However, it is unavoidable due to the nature of MMCC's current initialization process.

After all of the above changes have been made, Touring Machine's nameserver, the gateway's initialization file, and each MMCC's initialization file contain data for all users in both conferencing domains.

4.2. Clients and Users in Touring Machine

As mentioned earlier, Touring Machine refers to application instances as clients. Active clients must register with Touring Machine before taking any other action.[6] A client's name consists of the concatenation of the user's name and the application's name. It is valid only if the user's name has already been entered into the nameserver's database. As a result of this distinction between a user and a client, at any time, there can be more users than clients--i.e., some users are not using Touring Machine--or more clients than users--i.e., some users have more than one application that has registered with Touring Machine. So, to be more precise about the user proxy approach in the Touring Machine/MMCC case, the gateway serves as client proxies for MMCC users in the Touring Machine domain and as user proxies for Touring Machine clients in the MMCC domain.

Since Touring Machine messages are addressed to clients, not users, the gateway instantiates FSMs representing clients in the Touring Machine domain. More precisely, in the Touring Machine MPU, one FSM exists per client per active session, and, in the CCP MPU, one FSM exists per user per active session.

Clients are not recognized until they register with Touring Machine. Applications can query the nameserver for information about which clients are active at any given time. In this fashion, Touring Machine provides an "active user" server to applications.

Since Touring Machine applications cannot communicate with users who are not registered as clients, the gateway must register clients that represent MMCC users. In order to register and subsequently intercept messages, the gateway must use not only the MMCC users' names, but also some application's name. The gateway handles this problem by querying the database, searching for all active clients, and storing every unique

application name. The gateway then registers every MMCC user under every unique application name that it found. For instance, if there are three MMCC users--Bob, Chris, and Dave--and two active Touring Machine applications--App1 and App2--then the gateway would register the following clients: "Bob:App1," "Bob:App2," "Chris:App1," "Chris:App2," "Dave:App1," and "Dave:App2." The gateway can use the nameserver to discover when clients register using previously unregistered applications. It can then register each MMCC user as a new client using the new application name.

The use of the nameserver as an "active user" server for MMCC users is lost. The gateway could deregister all clients associated with an MMCC user if that user fails to respond to CCP messages. In such cases, the gateway can assume that the user has terminated his MMCC instance. However, when the user restarts his instance, MMCC provides no way to notify the gateway. Consequently, the gateway does not deregister clients that are associated with MMCC users, and Touring Machine users see perpetually active MMCC users.

4.3. Session Establishment

4.3.1. Touring Machine Session Establishment

Figure 4.1 shows FSMs that correspond to Touring Machine's basic session establishment protocol for a calling and a called party.[6] The initiating application begins by sending a sessionCreate message, which contains a desired session name and description. Touring Machine confirms receipt of the message by sending a sessionRequestReceived message. It then sends a sessionActionRequest message describing the requested action to each client, including the initiator, involved in the initial sessionCreate message. Clients then respond positively with a sessionActionAccepted message or negatively with a sessionActionDenied message. Touring Machine processes these responses from the involved clients and sends them all either a sessionActionCommit message if the session is successfully established (including all resource management and physical transport operations) or a sessionActionAbort message if someone declined to enter the session or if something else goes wrong. Notice that Touring Machine employs the policy that the session is aborted if even one callee responds with a sessionActionDenied message.

Touring Machine handles all resource management and physical transport func-

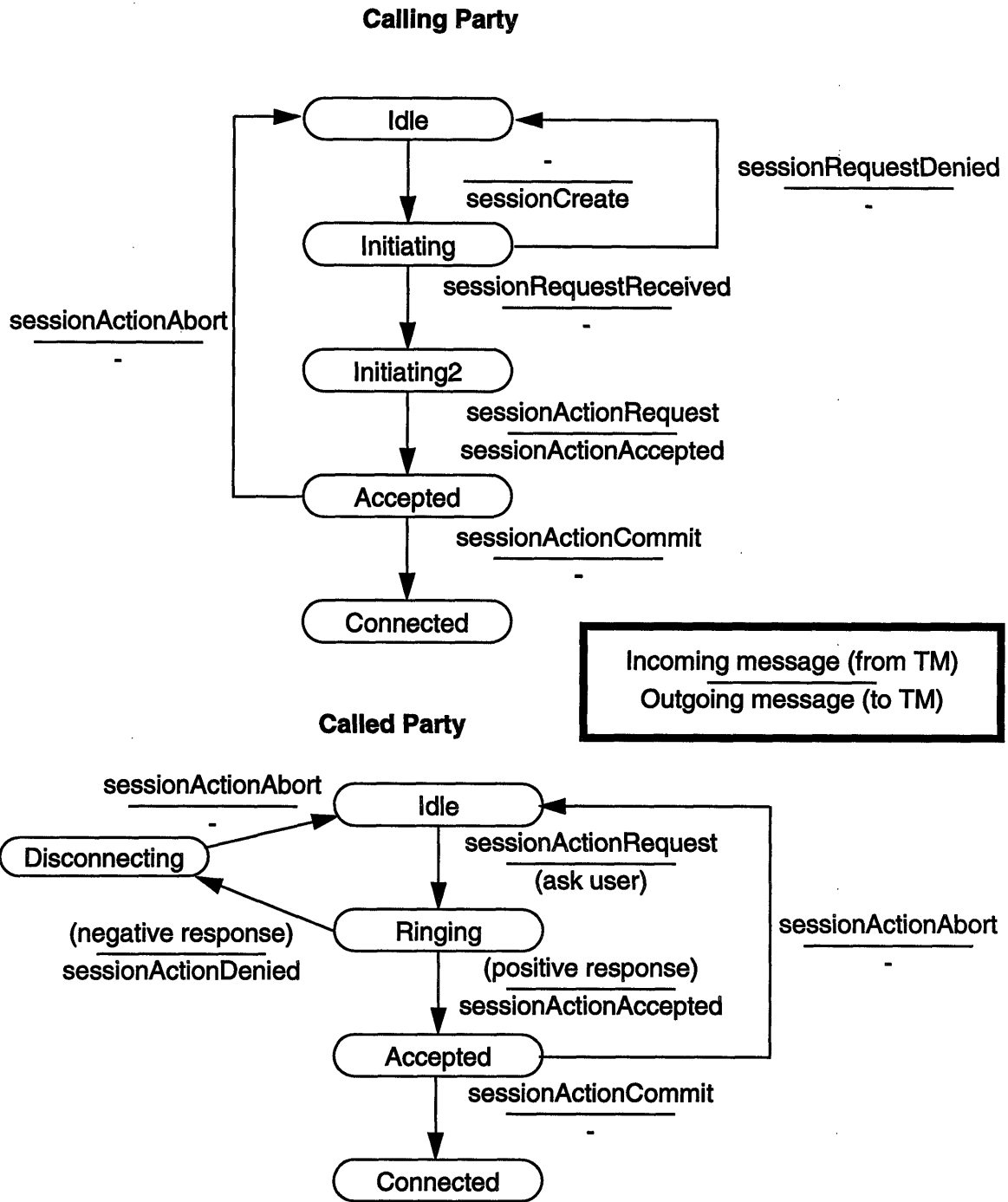


Figure 4.1. Touring Machine's session establishment protocol FSMs

tionality. It processes any feedback from resource objects during the session establishment procedure. Clients are isolated from everything except their end of the session protocol and any error messages that are sent to them in the form of `sessionActionAbort` or `sessionRequestDenied` messages.

Request message to all called users. Each callee checks the desired session configuration information included in the Request message with its own capabilities and responds with a Requestr message, which includes either a positive or negative reply. The caller collects all the Requestr messages and sends a Connect message to those users who have responded positively. When each callee receives the Connect message, a Connectr message is sent back with either a positive or negative reply depending on whether or not the user accepts the call. In contrast to Touring Machine, the caller implements its own policy if a negative Connectr message is received. It can either continue with the session by starting up its media agents and sending a Status message to all remaining callees or abort it by sending Disconnect messages. Assuming the caller chooses to continue, when each callee receives the Status message, it starts up its own media agents. Each callee sends a Statusr message to the caller with a response indicating the success of the media agents. When the caller receives all the Statusr messages, there is one final round of Status/Statusr messages to inform the callees that all media agents were started successfully before the session is established. If at any point in the sequence, the caller encounters an error or decides that too many callees have rejected the session--i.e., via negative Requestr, Connectr, or Statusr messages--the caller sends a Disconnect message to all remaining callees to abort the session.

Because CCP is used in a completely distributed environment, applications at the "ends" of the network must do more work without any help from a conferencing service in the "middle." In addition to processing protocol messages, applications must invoke and terminate local resource management functionality and media tools that handle physical transport--e.g., after receiving a Connect message.

4.3.3. FSM Modification Strategy--Touring Machine caller

The FSMs in Figures 3.3 and 3.4 are the basis of the FSMs used in the gateway's MPUs. They are the key for implementing the user proxy design. With the help of the semantic relationships in Table 1, a basic strategy can be developed for modifying the FSMs in Figures 3.3 and 3.4 for use in the gateway's MPUs. Figure 4.3 outlines such a strategy--employing several black-box procedures--for a Touring Machine client named Ann:App1 successfully calling a CCP user named Bob. The gateway intercepts a session-ActionRequest meant for Bob:App1 and instantiates a callee FSM representing him--

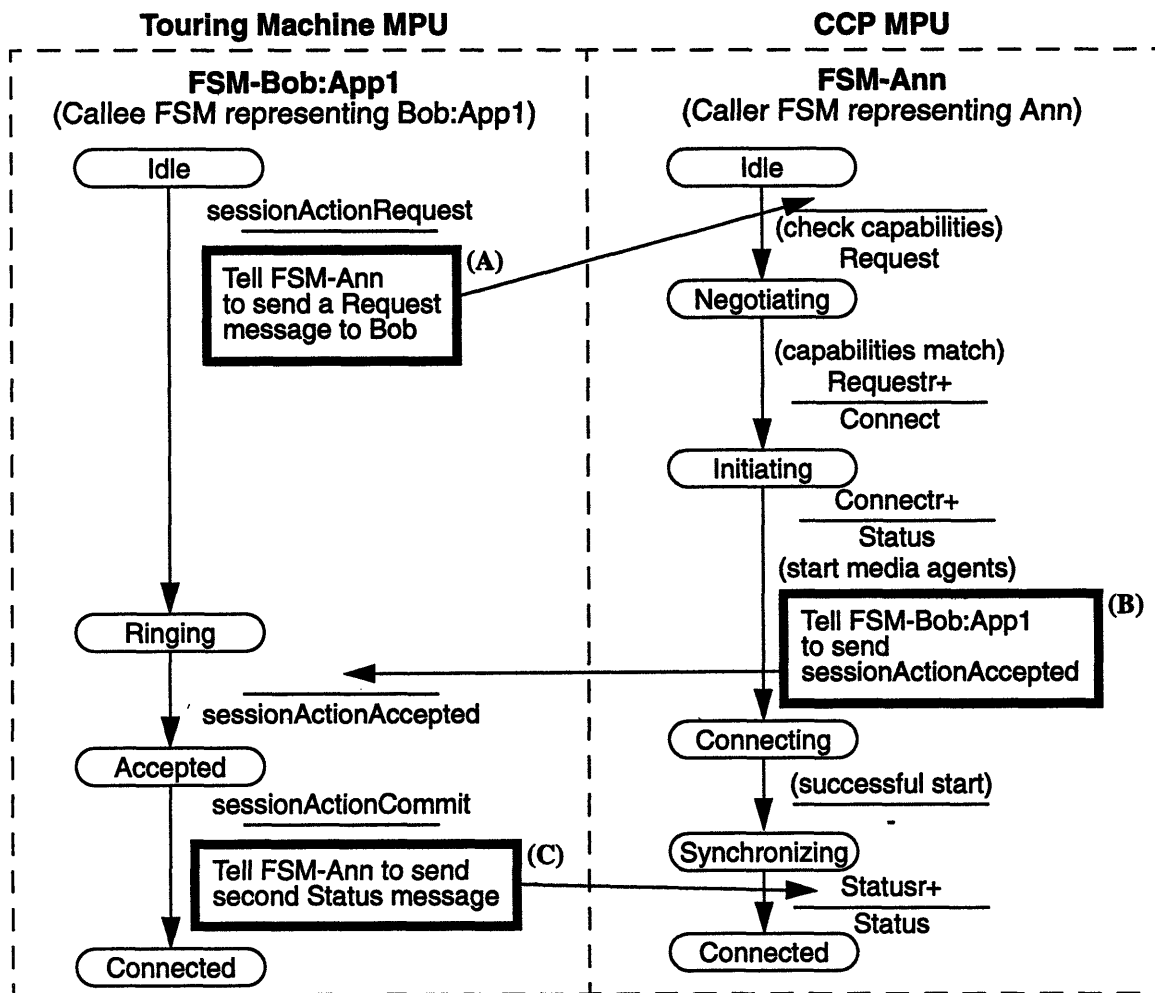


Figure 4.3. MPU FSMs for a Touring Machine client calling an MMCC user.

henceforth, called FSM-Bob:App1--in the Touring Machine MPU.¹ The normal Touring Machine FSM in Figure 4.1 would query the user for acceptance of the session, but the gateway cannot directly query Bob. Instead, the relationships in Table 1 are used: a Touring Machine callee receiving a sessionActionRequest message corresponds to a CCP callee receiving either a Request or a Connect message. CCP dictates that a caller start with a Request message, so the gateway instantiates an FSM representing Ann (FSM-Ann) in the CCP MPU and begins the CCP session establishment process by sending a Request message to Bob. FSM-Ann and Bob exchange CCP messages in the normal fash-

1. Since one FSM exists in the MPU per user (or client) per active session, an FSM is never actually in the Idle state. It either is instantiated and immediately transitions to the next state, or it is deleted as it transitions back to the Idle state.

ion until FSM-Ann receives a positive Connectr message. FSM-Ann sends a Status to Bob and starts its media agents as a normal CCP caller would. In addition, since a positive Connectr message was received, the relationships in Table 1 dictate that FSM-Bob:App1 send a sessionActionAccepted. When the sessionActionCommit message is received by FSM-Bob:App1, FSM-Ann should be told, since media transport has begun, to send the second Status message to Bob.

If FSM-Ann's capabilities match is unsuccessful or if it receives a negative Requestr or Connectr message, FSM-Bob:App1 is told to send a sessionActionDenied instead of a sessionActionAccepted message, causing session establishment to be aborted in both domains. If FSM-Bob receives a sessionActionAbort message, then FSM-Ann is told to send a Disconnect instead of a Status message, also causing the call attempt to be aborted.

The above strategy does not completely preserve the semantics of both protocols. When Touring Machine clients receive a sessionActionCommit message, they should be sure that there are no problems with transport resources at all. However, in Figure 4.3, it is possible that when Ann:App1 and FSM-Bob:App1 receive a sessionActionCommit message, trouble could have occurred either with the gateway's or Bob's media agents. One solution is for the gateway to immediately terminate the session (in both domains) when either its media agents fail or when a negative Statusr message is received from Bob. This solution is simple and, it happens to optimize the common case--i.e., MMCC's media agents usually start properly. In fact, since the common case on the Touring Machine side is also successful session establishment, the process could be further optimized by removing black-box procedure (C) in FSM-Bob:App1. With this approach, the gateway terminates the session if establishment fails in either domain. The result is Figure 4.4.

A second solution, illustrated in Figure 4.5, would have FSM-Ann wait until both the gateway's media agents have started and a positive Statusr message has been received before telling FSM-Bob to send the sessionActionAccepted message. Thus, Touring Machine will not send sessionActionCommit messages to its clients unless MMCC's media agents are successfully started. This solution does not quite preserve the semantics listed in Table 1, either--Touring Machine's sessionActionAccepted message is supposed

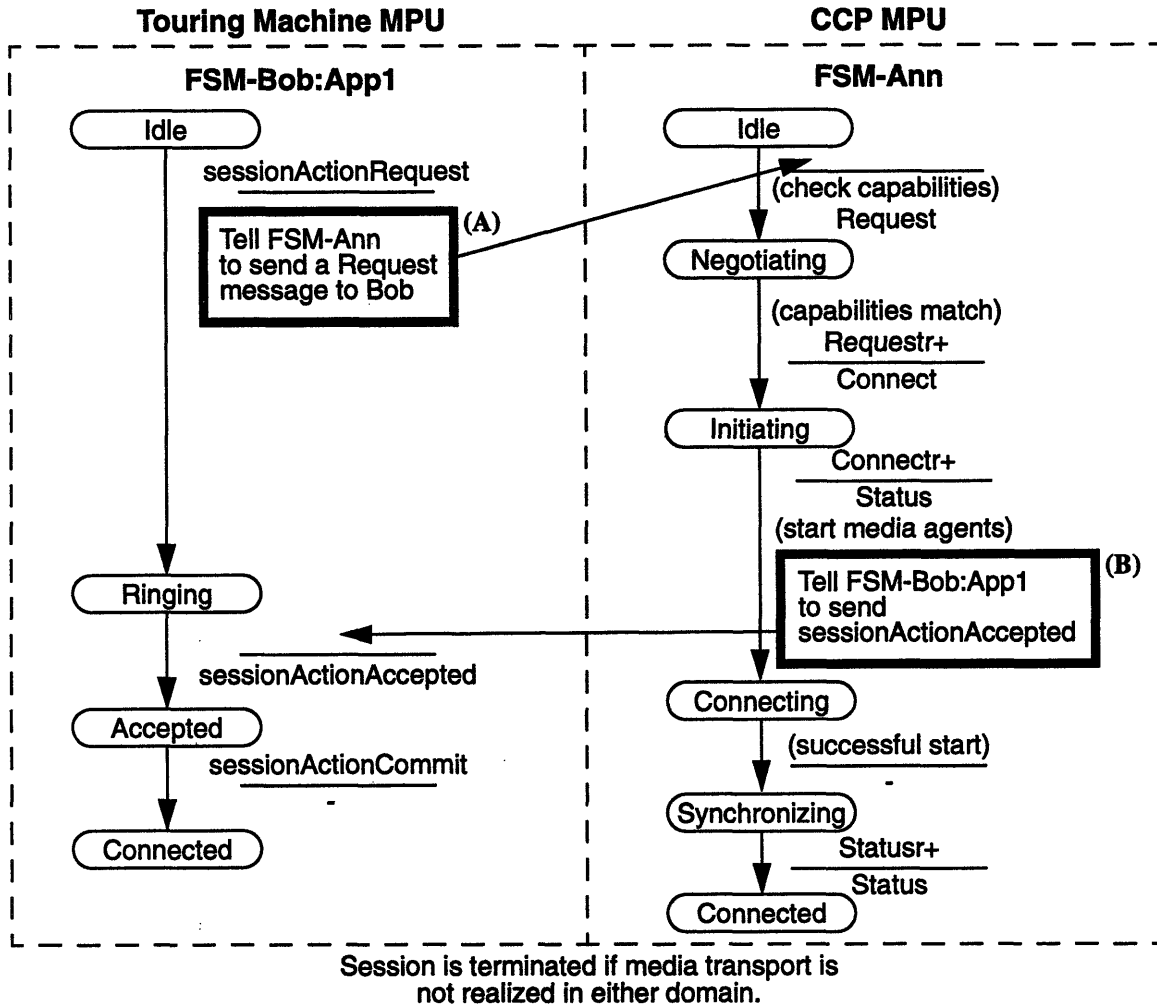


Figure 4.4. Better version of Figure 4.3.

to indicate only callee acceptance, not media transport success, as well. However, without modification to the conferencing systems, there is no way for CCP users to communicate the success of their media agents directly to Touring Machine. The main problem with Figure 4.5 is that it further lengthens the amount of time between FSM-Bob:App1's Ringing and Accepted states. This interval is already larger than what users and the conferencing system are accustomed to. Time-outs for both systems can be adjusted to accommodate the delays introduced by adding a gateway and another domain. However, exacerbating an already annoying problem for users in order to handle the *uncommon* case is not a wise choice. Therefore, Figure 4.4's simpler design was chosen for implementation.

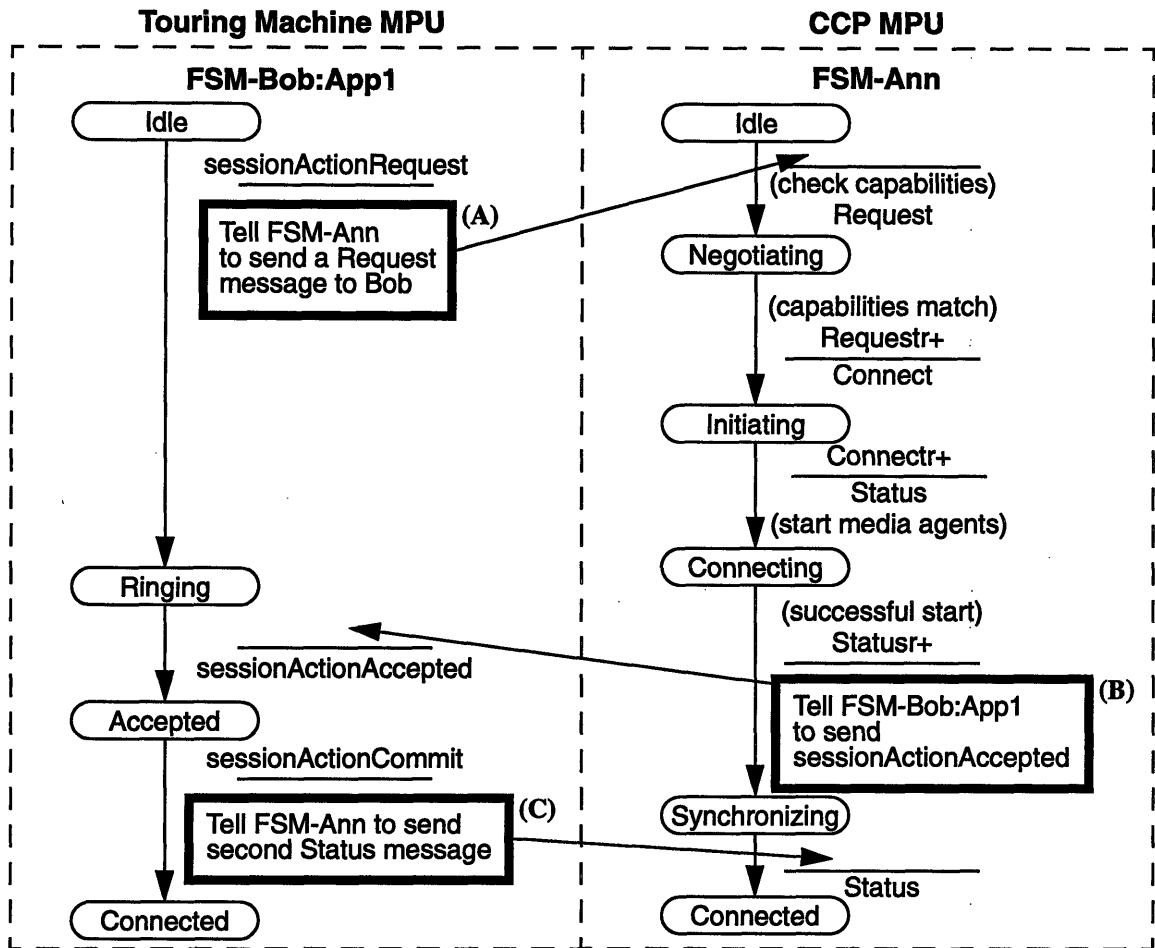


Figure 4.5. Alternative to Figure 4.4.

4.3.3.1. Multiple Callees

The strategy described in the previous section can be extended to include multiple callees. Suppose Ann:App1 is trying to call three other users: Bob, a CCP user named Chris, and a Touring Machine client named Dave:App1. The gateway does not deal directly with Dave:App1 at all. It must only pass on session description information, which, of course, includes Dave's involvement, to the CCP users. As Figure 4.6 shows, the gateway will intercept a sessionActionRequest message for Chris:App1, spawning the creation of FSM-Chris:App1. FSM-Chris:App1 tells FSM-Ann to send Request messages to both Chris and Bob. FSM-Chris:App1 can do this because the sessionActionRequest message includes information about all users who are in the proposed session. When the gateway receives a sessionActionRequest message meant for Bob:App1, FSM-Bob:App1 skips to the Ringing state because FSM-Chris:App1 has already communicated with

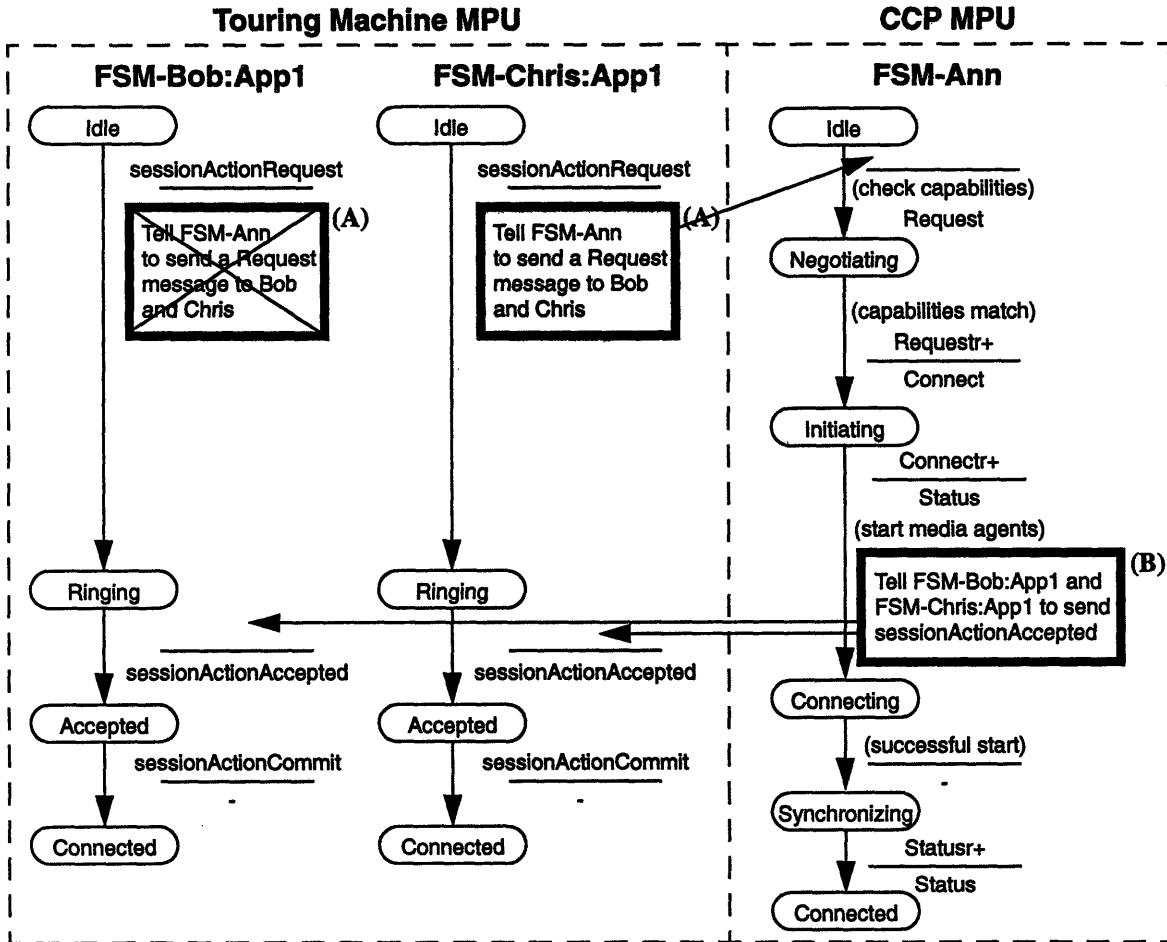


Figure 4.6. Touring Machine client calling multiple callees.

FSM-Ann. Note that the result would be identical if the sessionActionRequest message meant for Bob:App1 had arrived at the gateway first. The rest of the process is identical to Figure 4.4. When FSM-Ann receives a positive Connectr message from a callee, it tells the callee's associated FSM to send a sessionActionAccepted message.

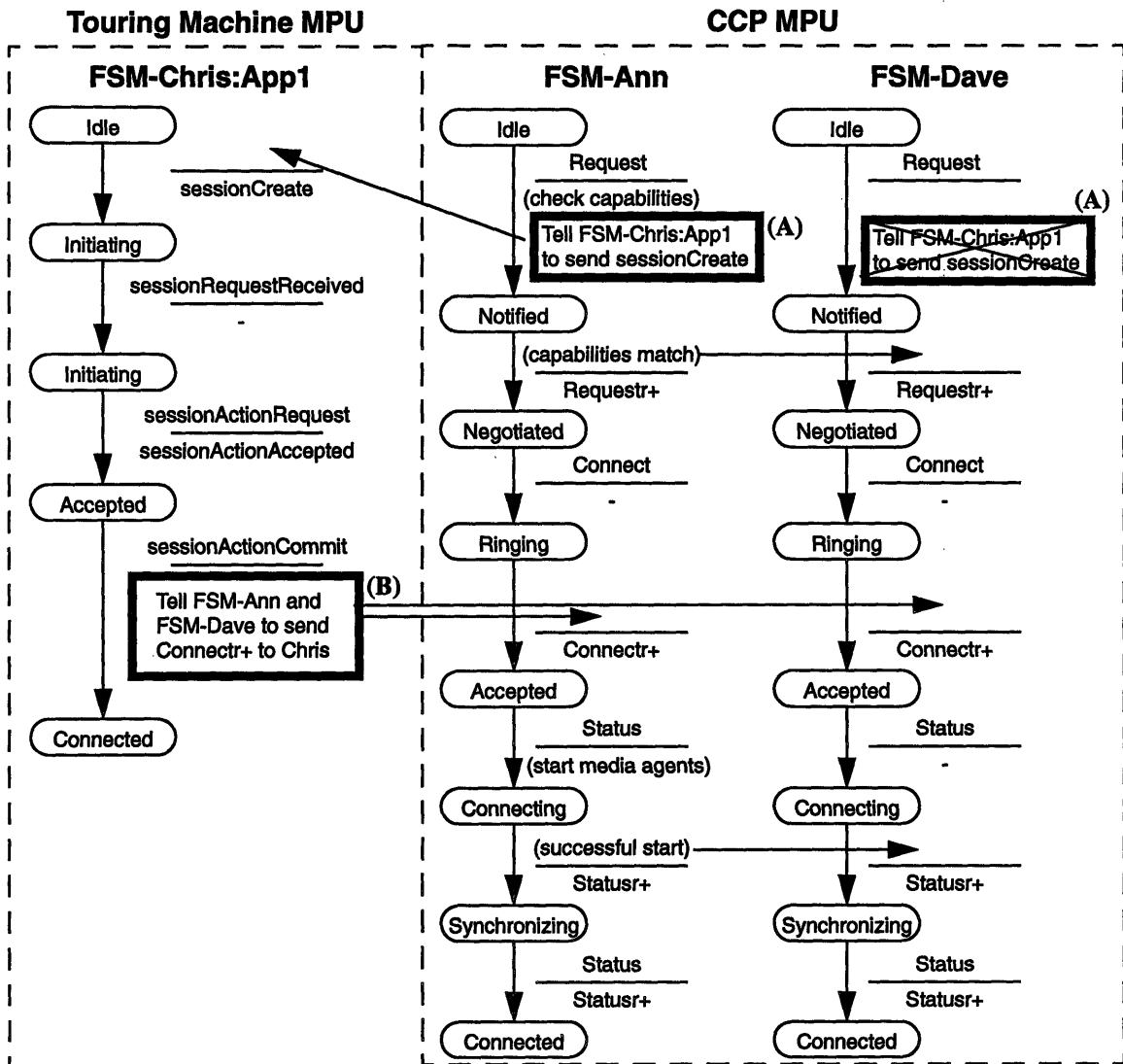
Notice that it is important that Touring Machine includes all the callees in the sessionActionRequest message. Otherwise, in order to execute black box (A), the gateway would need to wait until it received a possibly unknown number of sessionActionRequest messages before it could translate the request to the MMCC domain. The extra delay is troublesome because of the same response-time problem explained in section 4.3.3.

Recall that FSM-Bob:App1's black-box procedure (C) is absent in Figure 4.4 because the common case is successful transport realization by Touring Machine. In the

multi-party scenario, however, Touring Machine will send sessionActionCommit messages only if transport is trouble-free *and* if other Touring Machine callees--e.g., Dave--accept the session. It is difficult to argue generally that positive callee responses are either the common or the uncommon case. However, because of its relative simplicity, Figure 4.6 was still chosen for implementation.

4.3.4. FSM Modification Strategy--MMCC caller

By incorporating some of the lessons learned above, Figure 4.7 was developed for a multi-party call initiated by an MMCC caller. The figure shows gateway activity for



Session is terminated if media transport is not realized in either domain.

Figure 4.7. MMCC user calling multiple Touring Machine clients.

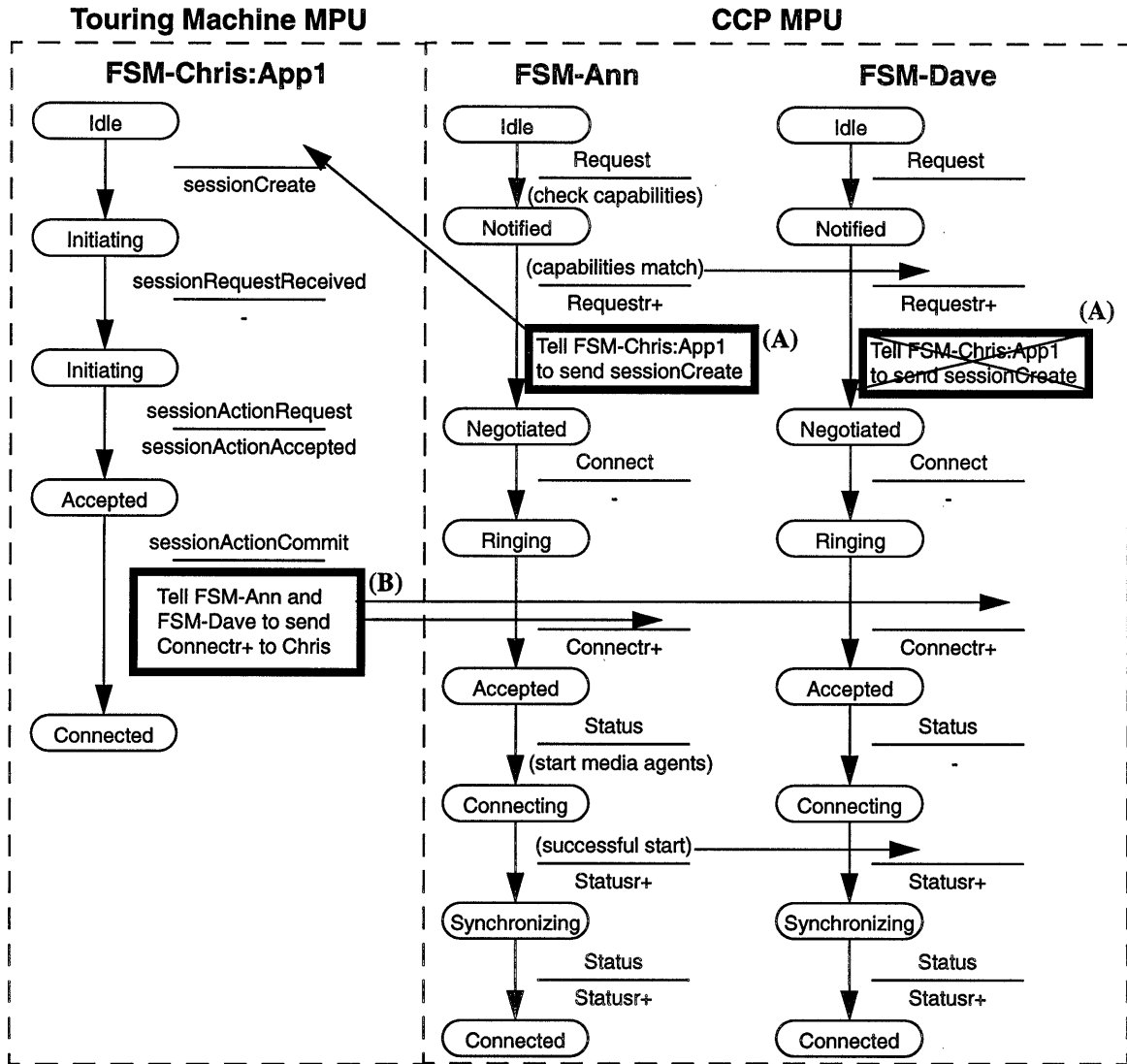
Chris successfully calling Ann:App1, Bob, and Dave:App1. The gateway intercepts a

Request message meant for Ann and instantiates FSM-Ann in the CCP MPU. Since Table 1 shows that sending a Request message and sending a sessionCreate message both correspond to how a caller initiates session establishment, FSM-Ann tells a newly-created FSM-Chris:App1 to send a sessionCreate message. The message includes a session description involving Ann:App1, Bob:App1, Chris:App1, and Dave:App1. As in Figure 4.6, it is important that the Request message contains a complete session description so that black box (A) can be executed in a timely fashion. When FSM-Dave receives the Request message from Chris, it jumps immediately to the Notified state. FSM-Dave and FSM-Ann continue with the CCP protocol until they reach the Ringing state. They must wait until the gateway is informed of Dave's and Ann's acceptance of the session, respectively--i.e., until FSM-Chris:App1 receives a sessionActionCommit message from Touring Machine. Notice that the gateway does not deal with Bob at all.

If FSM-Chris:App1 receives a sessionRequestDenied or a sessionActionAbort message, session involvement for Ann and Dave is aborted by FSM-Ann and FSM-Dave sending negative Connectr messages. If these callee FSMs ever receive a Disconnect message from Chris, the session must be terminated in the Touring Machine domain. This is the same approach used in Figure 4.4--if MMCC media transport fails, the session is terminated after it is started in the Touring Machine domain. Note that there is no analogous solution to Figure 4.5. The actual callees must be queried before FSM-Dave or FSM-Ann can transition to the Accepted state. Therefore, it is impossible for either to wait for media agents to start before telling FSM-Chris:App1 to send the sessionCreate message.

Since Table 1 shows the sessionCreate message corresponding to both the Request and the Connect messages, there is an alternative to Figure 4.7. Figure 4.8 shows callee FSMs in the CCP MPU waiting until they receive a Connect message before telling FSM-Chris:App1 to send the sessionCreate message. This choice would make sense if the gateway's capabilities are often mismatched with CCP callers. In such situations, a negative Requestr message would be sent by FSM-Dave, other FSMs would never be created, and Touring Machine would avoid unnecessary processing. However, in CCP's call model, the Request message also serves as notification to callees that Chris attempted to contact them. In Figure 4.8, this significant functionality of Request is lost.

Note that neither strategy can give Chris the policy choice that he normally enjoys



Session is terminated if media transport is not realized in either domain.

Figure 4.8. Alternative to Figure 4.7.

with MMCC callees. In cases where an CCP user calls more than one Touring Machine user, Touring Machine’s all-or-nothing policy when dealing with callee acceptance precludes implementing CCP’s normal flexible policy.

4.3.4.1. Client Name Choice

There is one issue that was ignored in the previous section. When FSM-Ann sends the sessionCreate message in Figure 4.7, the gateway must choose an application name to use in order to form client names. Although inter-application operation is not ruled out specifically by Touring Machine’s API document, the ubiquity of application names in the

control messages, the examples given in the document, and the existing Touring Machine applications seem to show a lack of consideration for inter-application use. Because of this, for practical use, the gateway must choose an application that is being used by all Touring Machine callees. So, referring again to Figure 4.7, if clients Ann:App1, Ann:App2, Ann:App3, Dave:App1, and Dave:App3 were registered, Chris-FSM would have sent a sessionCreate message involving either Ann:App1, Bob:App1, Chris:App1, and Dave:App1, or Ann:App3, Bob:App3, Chris:App3, and Dave:App3.

4.4. Session Termination

4.4.1. Touring Machine Participant Removal

Touring Machine's protocol for a client's self-removal from a session, pictured in Figure 4.9, is very similar to its caller session establishment protocol.[6] The session-

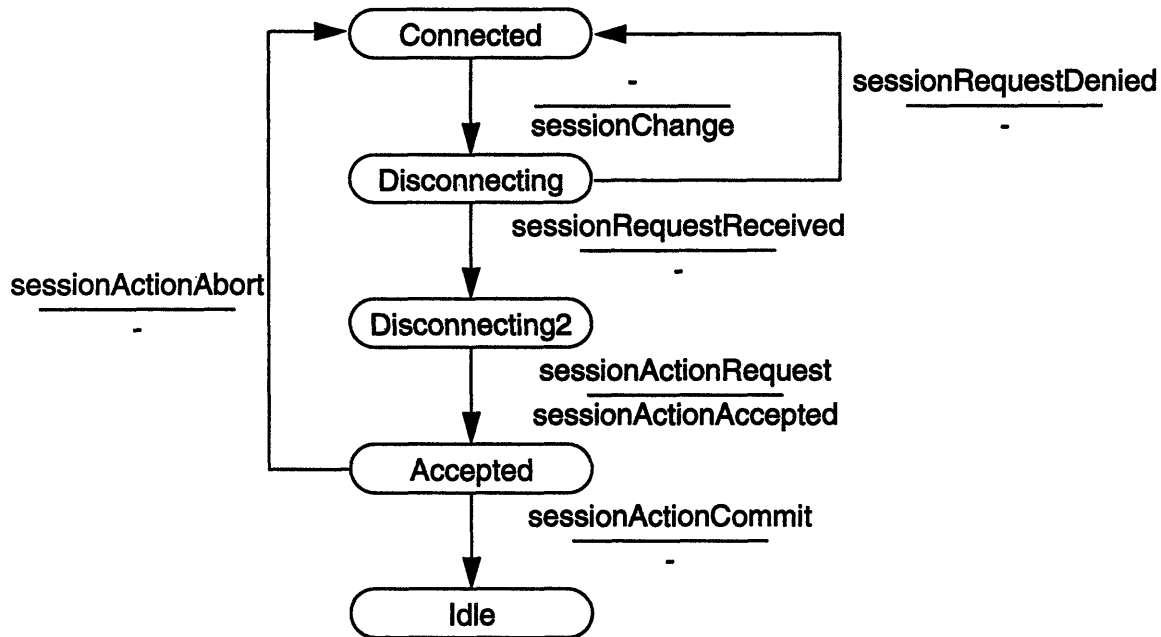


Figure 4.9. Touring Machine's participant removal

Change message has several uses--see section 4.5.1--only one of which is removing clients from a session. No matter what the proposed action is inside the sessionChange message, Figure 4.9 is the process that occurs. Hence, a client must go through the two-stage commit process just to remove itself from a session. The final sessionActionCommit message is sent to all relevant clients--i.e., the sessionActionCommit notifies all other clients of this client's removal from the session.

Notice that in Figure 4.9, if an error occurs while a client is trying to remove itself from a session--i.e., the client receives a `sessionRequestDenied` or `sessionActionAbort` message--Touring Machine leaves the client in the session. This is the problem described in section 3.3.2.

4.4.2. MMCC Participant Removal

In sharp contrast to Touring Machine's participant removal protocol is CCP's protocol, which is very simple.[12] A user removes himself from a session by sending a `Disconnect` message to all other users in the session. They respond with a `Disconnect` message. If any problems occur--e.g., some user is unreachable--the user transitions to the `Idle` state and terminates its media agents, anyway.

4.4.3. Touring Machine Client Leaving an Inter-domain Session

Figure 4.10 shows Ann:App1 leaving a session that included Ann:App1, Bob, and Chris. When the gateway receives the `sessionActionCommit` message announcing that

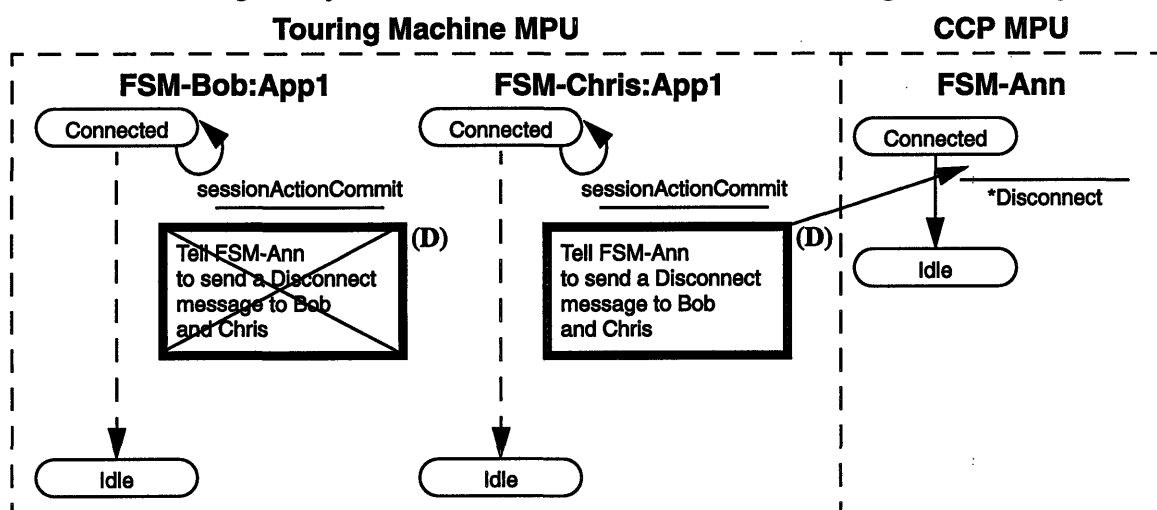


Figure 4.10. Ann:App1 removes itself from the session.

Ann:App1 has left the session, FSM-Ann is told to send a `Disconnect` message to Bob and Chris, and FSM-Ann is deleted. Bob and Chris remain in the session, but the gateway does not need to participate in any session that does not involve at least one user or client from each domain. Therefore, the gateway will terminate its media agents and delete FSM-Bob:App1 and FSM-Chris:App1 (represented by the transitions to the `Idle` state), as well.

4.4.4. MMCC User Leaving an Inter-domain Session

Figure 4.11 shows Bob leaving the session that included Ann:App1, Bob, and Chris. When FSM-Ann receives the Disconnect, it responds with a Disconnectr and tells Touring Machine MPU

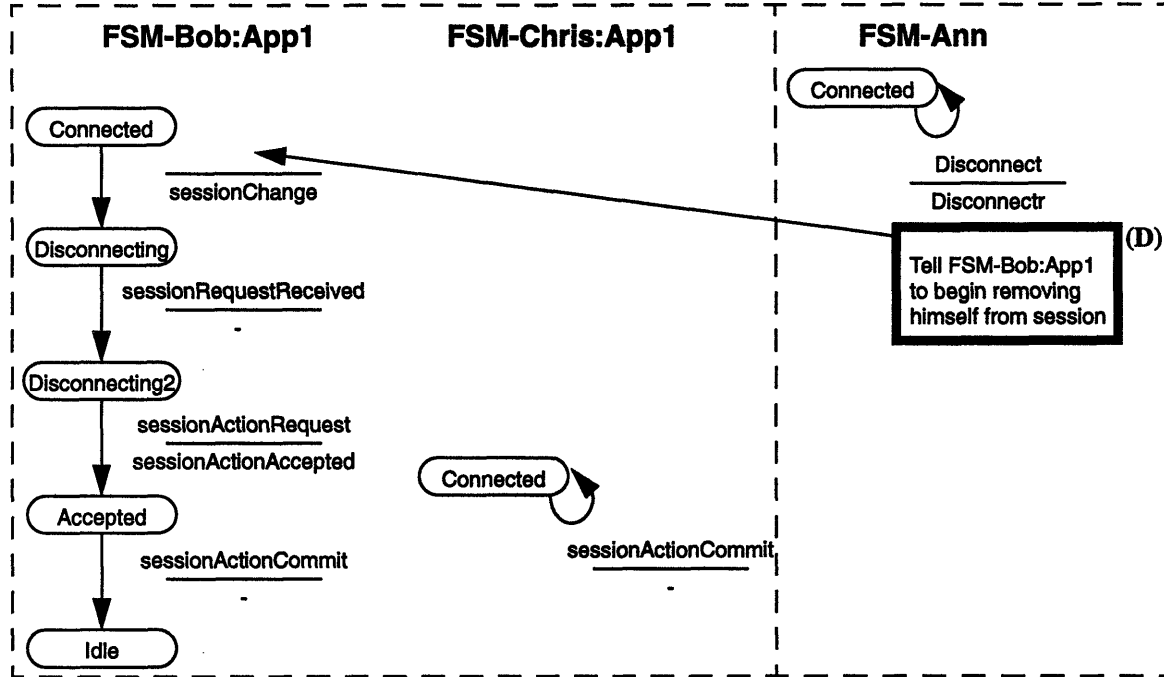


Figure 4.11. Bob removes himself from the session

FSM-Bob:App1 to begin the process of removing itself from the session. If this process is successful, the gateway deletes FSM-Bob:App1. If the process is unsuccessful, there is no way to reconcile the situation with the MMCC domain. This is the IL3 behavior mentioned in section 3.3.2.

4.5. Invitation of new participants

The gateway must deal with four scenarios to provide participant invitation: a Touring Machine client inviting a Touring Machine client, a Touring Machine client inviting an MMCC user, an MMCC user inviting a Touring Machine client, and an MMCC user inviting an MMCC user. Participation invitation protocols for each system and strategies for each of the four situation are presented below.

4.5.1. Touring Machine Participant Invitation Protocol

There are three possible types of clients involved with participant invitation. The client initiating the invitation (the inviter), the client being invited (the invitee), and other

clients who are in the original session. Touring Machine asks only the permission of the invitee, but all participants are notified when the invitation occurs.[6] The FSM for the invitee is identical to that of the called party in Figure 4.1. Figure 4.12 shows the FSM for the inviter. If the invitee accepts the invitation and Touring Machine can realize the trans-

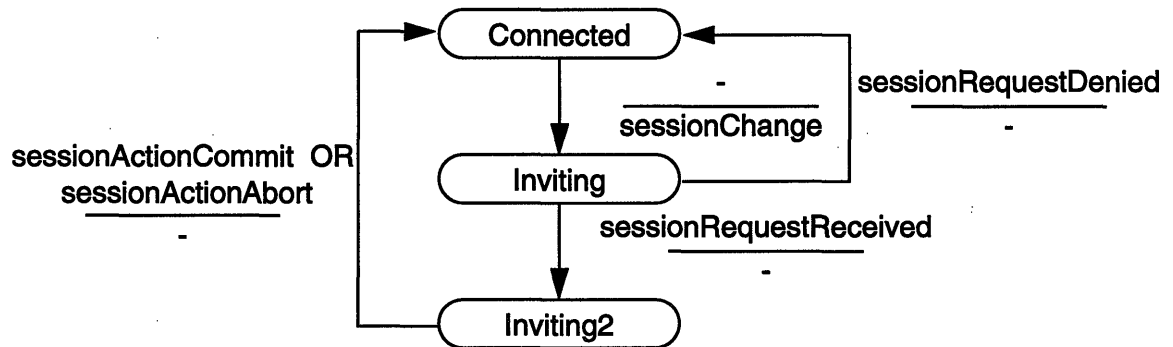


Figure 4.12. Touring Machine inviter FSM.

port, then the inviter, the invitee, and the other session participants all receive a sessionActionCommit and transition into the Connected state. If an error occurs or if the invitee denies the invitation, the inviter receives a sessionActionAbort and transitions back to the Connected state. The other session participants are unaffected.

4.5.2. CCP Participant Invitation Protocol

Like Touring Machine, only the invitee is asked for permission during an invitation, and the FSM for an invitee is identical the called party FSM in Figure 3.4.[12] Figure 4.13 shows FSMs for the inviter and other session participants. If the invitee accepts the invitation, all participants go through two rounds of Status and Statusr messages as during the session establishment protocol. If not, the inviter transitions back to the Connected state, and the other participants are unaffected.

4.5.3. Touring Machine Client Inviting an MMCC User

This scenario can occur when the original session is either inter-domain or purely intra-domain. The former is illustrated in Figure 4.14. The original session was between Ann:App1 and Bob, and Ann:App1 invites Chris to the session. Figure 4.14 is based on the strategy in Figure 4.5, not Figure 4.4. The reason is that participant invitation cannot always be reversed as easily as session establishment. If Touring Machine returns a sessionActionAbort message to FSM-Chris:App1, the gateway cannot remove Chris from the session after he has been added. Therefore, black box (C) is necessary to inform FSM-

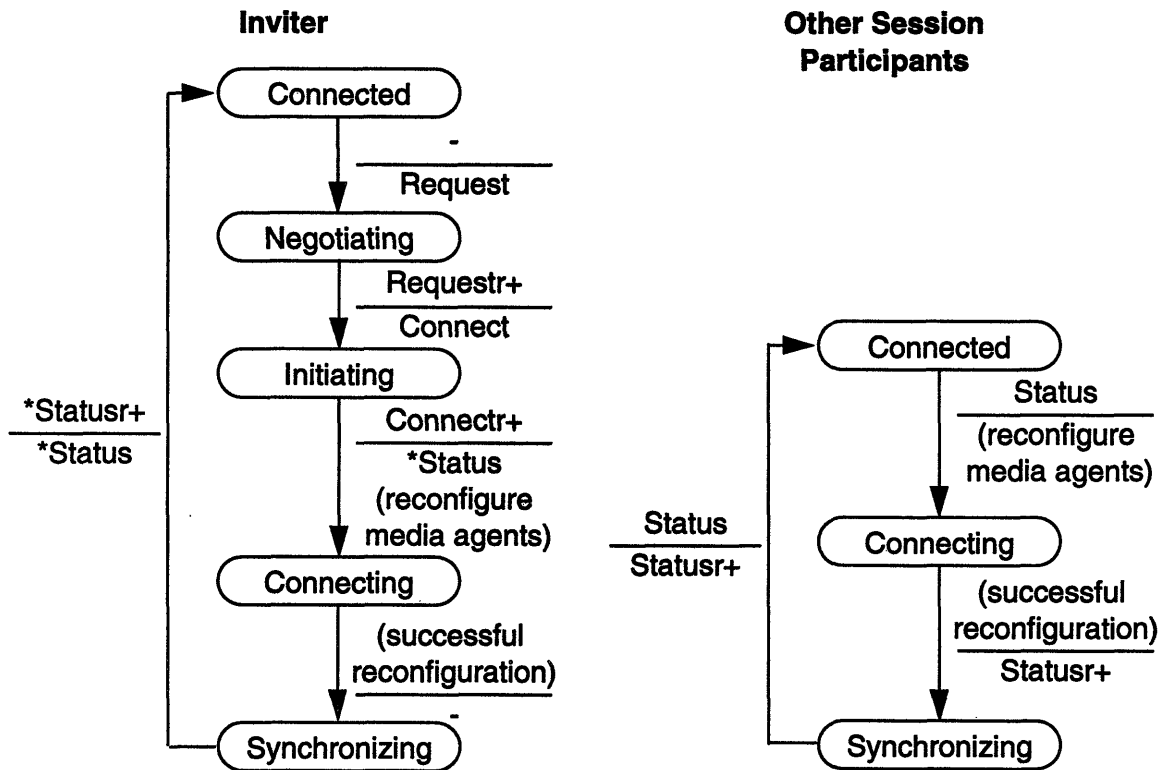


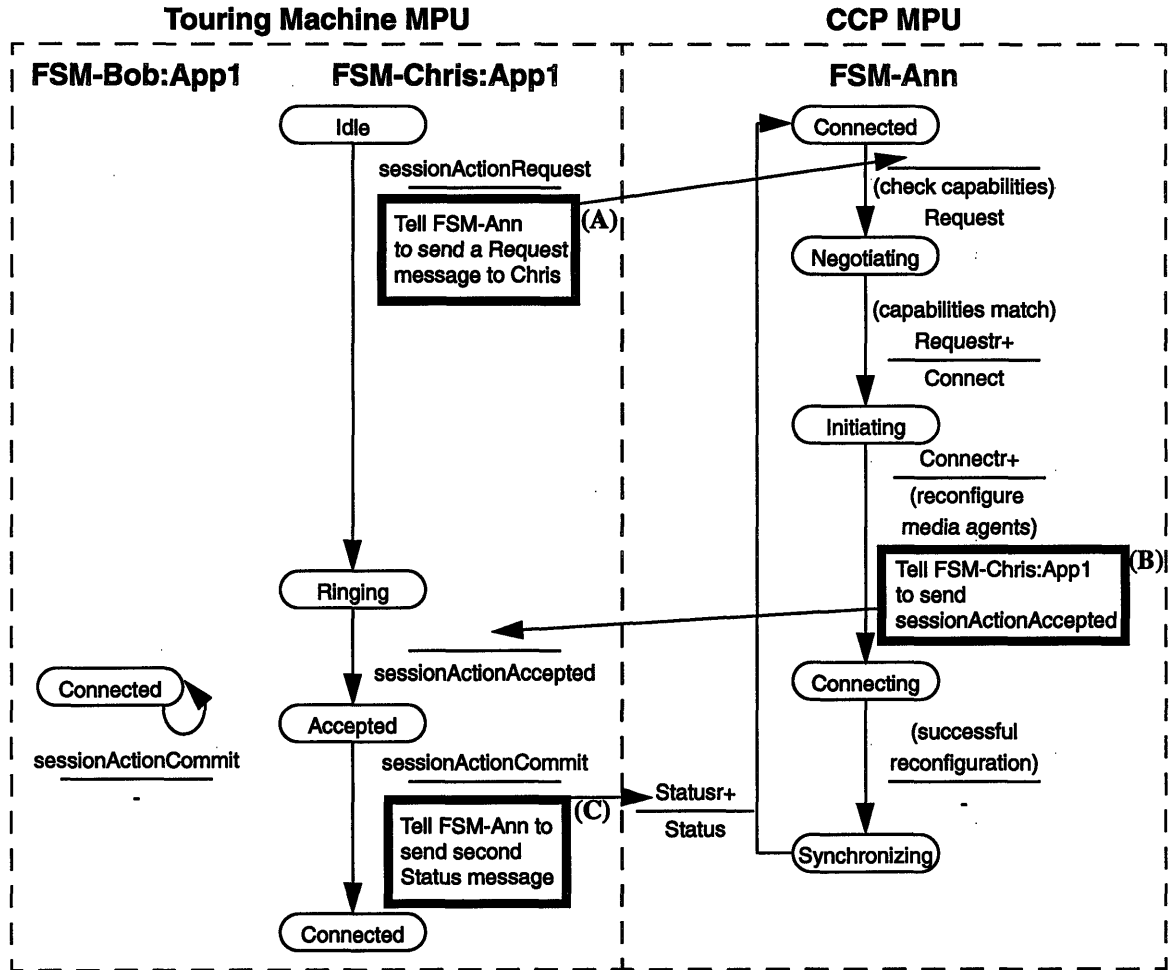
Figure 4.13. CCP FSMs for participant invitation.

Ann of the Touring Machine domain's success before the invitation is finalized. If either the gateway's or Chris' media agent reconfiguration fails, the gateway can tell FSM-Chris:App1 to remove itself from the session to keep the two domains consistent.

If the original session involved only Touring Machine clients--Ann:App1 and Dave:App1--and Ann:App1 invited Chris to the session, the resulting gateway activity would be analogous to Figure 4.14. The only difference occurs when FSM-Ann transitions to the Connected state after Chris is successfully invited: the Touring Machine MPU instantiates FSM-Dave in the Connected state.

4.5.4. MMCC User Inviting Another MMCC User

Since the gateway is not involved in purely intra-domain sessions, it need only deal with this situation if the original session involves a Touring Machine client. Consider an active session between Ann:App1 and Bob. Figure 4.15 shows gateway activity when Bob invites Chris to the session. The gateway is first notified of the invitation when FSM-Ann receives the Status message describing Chris's addition to the session. FSM-Ann responds by telling FSM-Bob:App1 to begin FSM-Chris:App1's invitation through Tour-



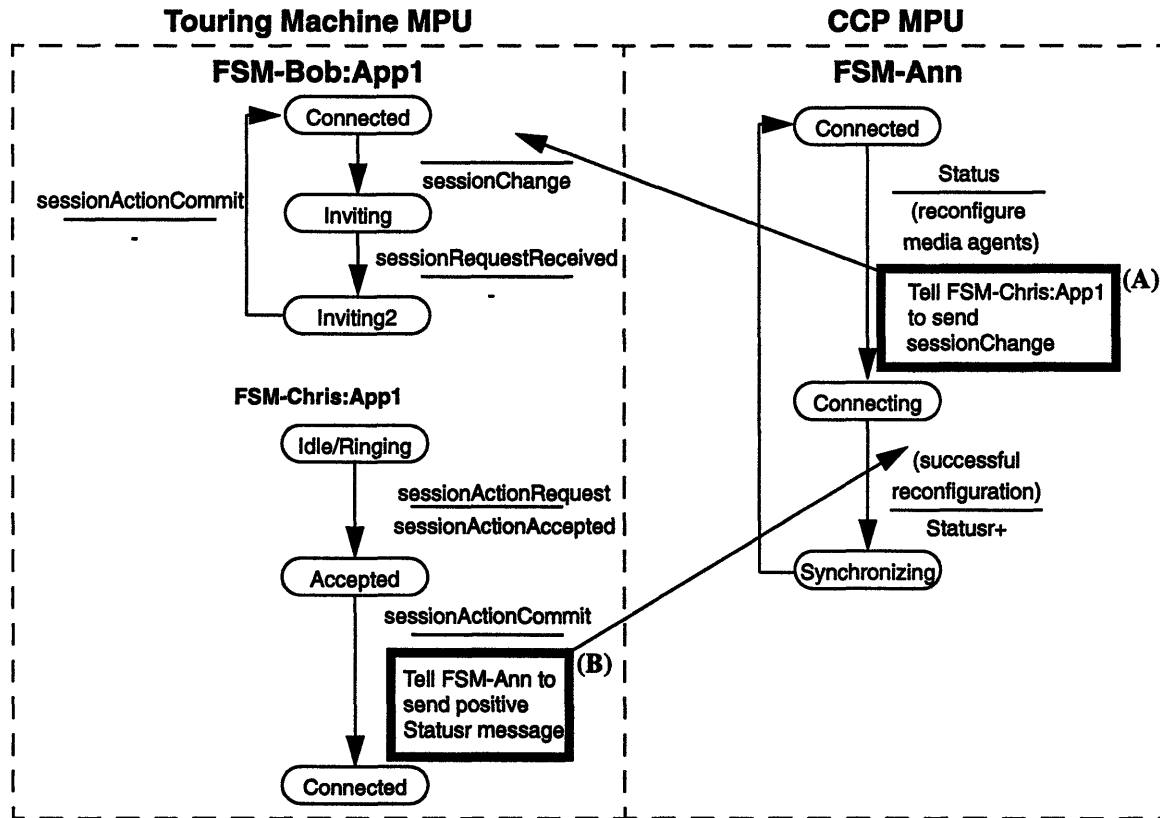
FSM-Chris:App1 removes itself from the session if MMCC's media agent reconfiguration fails.

Figure 4.14. Ann:App1 invites Chris to her session with Bob.

ing Machine. FSM-Chris:App1 is instantiated when the gateway receives the `sessionActionRequest` message, and it automatically responds with the `sessionActionAccepted` message because the real Chris has already agreed to the invitation. When the invitation is committed on the Touring Machine side, FSM-Ann replies to Bob with a positive `Status` message. At this point, an error can occur--i.e., Bob's media agents fail to reconfigure and Bob sends a `Disconnect` to FSM-Ann aborting the invitation. If so, the gateway removes FSM-Chris:App1 from the Touring Machine session.

4.5.5. MMCC User Inviting a Touring Machine Client

Figure 4.16 shows Bob inviting Dave:App1 to a session that originally had Bob and Ann:App1. This strategy is unlike any of the session establishment scenarios because



FSM-Chris:App1 removes itself from the session if MMCC's media agent reconfiguration fails.

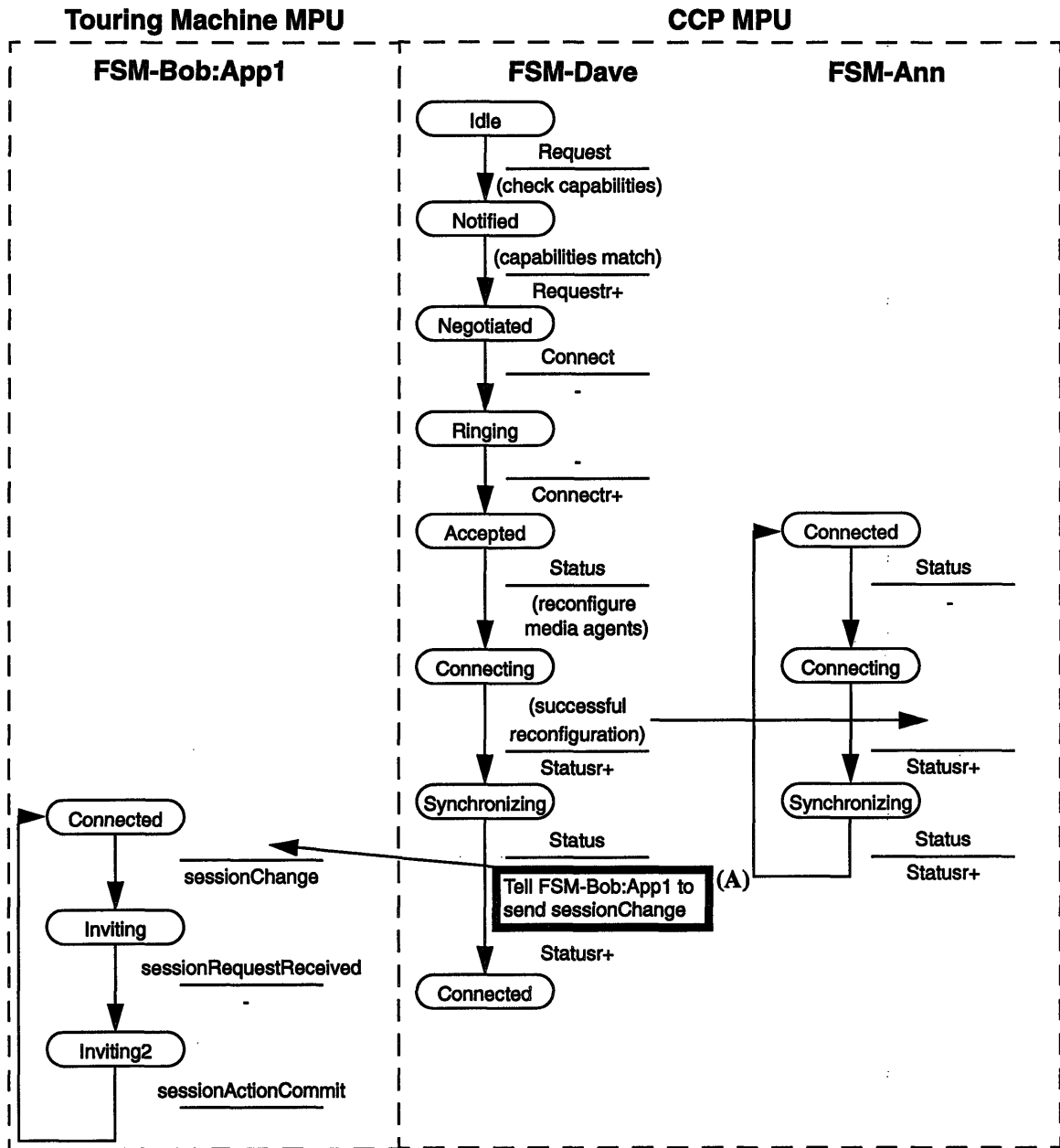
Figure 4.15. Bob invites Chris to his session with Ann.

of the irreversibility of adding Dave:App1 to the Touring Machine session. FSM-Dave automatically sends a positive Connectr message without any knowledge of Dave:App1's acceptance of the session. This is clearly breaking the semantics of the Connectr message. However, a failure to commit the invitation in the Touring Machine domain--either due to Dave:App1's disapproval or otherwise--can be reconciled by FSM-Dave removing itself from the session. On the other hand, if FSM-Bob:App1 were to query Dave:App1 by sending the sessionChange message earlier, Dave:App1 would be added to the session without any way to remove him if MMCC's media agent reconfiguration fails.¹

4.5.6. Touring Machine Client Inviting Another Touring Machine Client

As in section 4.5.4, since the gateway is not involved in purely intra-domain ses-

1. Actually, Touring Machine's sessionChange message does allow clients to propose removal of other clients from sessions--see section 5.1.1. However, removing Dave:App1 from the session would require his approval, which is hardly certain in this case.



FSM-Dave removes itself from the session if FSM-Bob:App1's efforts fail in the Touring Machine domain.

Figure 4.16. Bob invites Dave to join his session with Ann.

sions, it need only deal with this situation if the original session involves an MMCC user. Consider an active session between Ann:App1 and Bob. Figure 4.17 shows gateway activity when Ann:App1 invites Dave:App1 to the session. FSM-Bob:App1 receives the sessionActionCommit messaging describing Dave's addition to the session. FSM-Bob:App1 responds by telling FSM-Ann to notify Bob that Dave has been added to the session. In addition, when FSM-Ann transitions back to the Connected state, the CCP

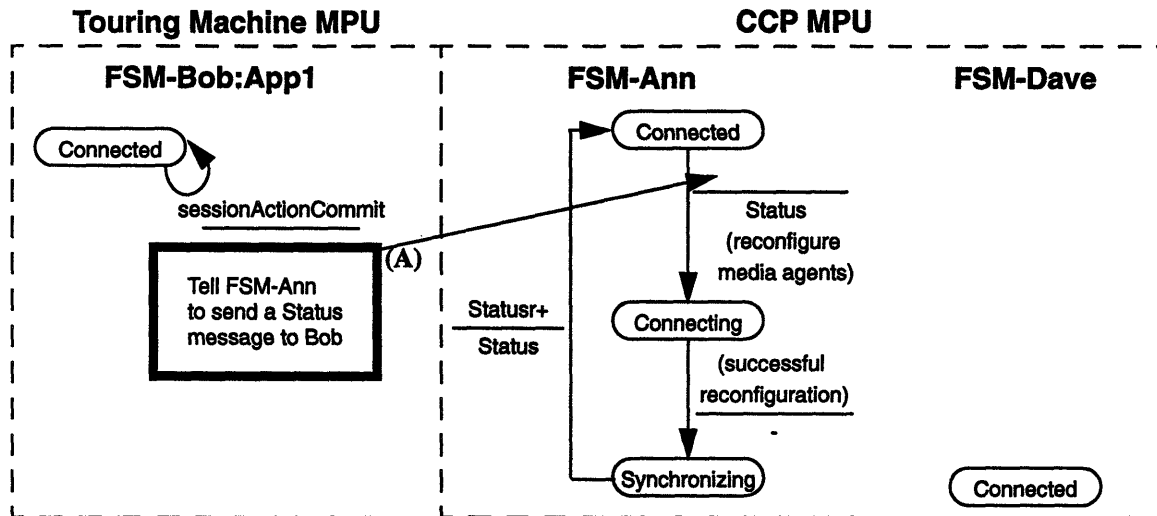


Figure 4.17. Ann inviting Dave to her session with Bob.

MPU instantiates FSM-Dave in the Connected state.

In this scenario, there is no way to handle an error with the gateway's media agent reconfiguration. When the gateway is first notified of the invitation, Dave:App1 is already a part of the session. The gateway can guarantee no better than IL3 behavior for such cases. However, MMCC media agent reconfiguration errors are rare. Furthermore, referring back to Figure 3.5, physical transport connections do not have to change in the MMCC domain because the new participant's media connections were added in the Touring Machine domain. Only the media agents' user interfaces need to be updated to reflect an extra participant.

4.6. Inter-MPU messages

To implement black box (A) in Figure 4.3, FSM-Bob:App1 has to relay the session description in the sessionActionRequest message so that FSM-Ann can send an analogous description in the Request message. Similarly, black box (A) in Figure 4.7 requires FSM-Ann to relay the session description from the Request message to FSM-Chris:App1 to use in the sessionCreate message. All of the black box procedures labeled (A) in Figures 3.5 - 3.7 and 4.1 - 4.17 also require communication of a session description from one MPU to the other. The CONNECT inter-MPU message is used to perform all of these tasks. In addition to the session description, the CONNECT message contains a flag indicating whether the session request is for a new session or for an invitation. It also contains a session identifier. All inter-domain sessions have a session identifier for each domain. The

gateway keeps track of which Touring Machine session name corresponds to which MMCC session identifier. Session identifiers in inter-MPU messages can be either the Touring Machine or MMCC version.

To relay either positive or negative responses between MPUs--i.e., all black boxes labeled **(B)** and **(C)**--the inter-MPU message **REPLY** is used. **REPLY** messages contain a session identifier, a boolean response, and a comment string. The response is true for all positive indications--e.g., `sessionActionAccepted`, `sessionActionCommit`, positive `Connectr`, and successful media agent reconfiguration--and false for all negative indications. Finally, the **DISCONNECT** message, which also contains a comment string, is passed between FSMs to implement black boxes labeled **(D)** in Figures 4.10 and 4.11. The comment string in the **REPLY** and **DISCONNECT** messages is most useful for communicating the nature of errors that occur.

Chapter 5

Evaluation and Conclusions

Section 5.1 presents an evaluation of the conferencing gateway design. Section 5.2 describes general principles of conferencing systems that benefit gateway solutions. Finally, section 5.3 discusses future work involving teleconferencing control and gateways.

5.1. Gateway Design Evaluation

The user proxy approach was used to create the conferencing gateway between the Touring Machine and MMCC systems. The representation of foreign users in each domain and the use of cooperating FSMs to process messages from both systems resulted in effective, though not complete, interoperability.

The user proxy approach dictates that the gateway interact with each conferencing system at the application interface level. Working at that level seems a straightforward decision considering the design requirement that neither systems' software be modified. It also lets the gateway take advantage of the main purpose of conferencing systems--to handle most of the complexities of session and resource management. This is most evident in the simplicity of the gateway's inter-domain resource management. The gateway has to handle only as much resource-related complexity as an application does. Consequently, it has only as much control over its media transport resources as an application does. Fortunately, the application's control--MMCC media agent control and Touring Machine endpoint assignment--are powerful enough for the gateway to perform its duties.

In many respects, developing the gateway resembles writing two simple applications--one for each system--that must work together well. Attempting to mesh the two systems' session and/or resource management schemes at a lower level would be a much more difficult--perhaps impossible--problem to solve.

5.1.1. Functionality Evaluation

The introduction of the gateway between the Touring Machine and MMCC domains caused no functionality to be lost for intra-domain conferencing. That is, MMCC users and Touring Machine clients have the exact same services for conferencing with

other users or clients in their domains as they did before the gateway was implemented. In addition, the gateway has provided interoperability between users of the different systems for most of the needed and popular conferencing functions.

Nonetheless, there were problems reconciling all of the policy and functionality mismatches between the systems. Table 2 summarizes the levels of interoperability achieved by the gateway for various conferencing functions. All of the IL3 behavior occurs only in rare situations or for non-crucial conferencing functions.

Table 2: Interoperability Levels

Conferencing Function	Interoperability Level for Touring Machine clients interacting with MMCC users	Interoperability Level for MMCC users interacting with Touring Machine clients
Symmetric audio/video sessions (multiple parties)		
Basic establishment	IL1	IL1
Callee-acceptance policy	IL1	IL3
Participant self-removal	IL1	IL3
Removal of other participants	IL2	N/A
Participant invitation	IL3	IL1
Asymmetric session establishment/ modification	IL2	N/A
Data session establishment	IL2	IL2
Application-level messaging	IL3	N/A
Nameserver use		
As directory database	IL1	N/A
As "active user" database	IL3	N/A

Section 4.3.4 showed that the gateway solution cannot satisfy CCP's flexible callee-acceptance policy because of Touring Machine's rigid policy. Table 2 lists this as IL3 behavior because the MMCC caller sees behavior that is unexpected--i.e., different than interaction with only other MMCC users. However, the severity of this IL3 behavior is minimal. The session request is denied by all Touring Machine callees uniformly, and the caller is free to try again.

IL3 behavior for MMCC participant self-removal was described in section 3.3.2. It is unfortunate and unavoidable; however, it is rare that IL3 behavior actually occurs, since the common case is Touring Machine successfully removing participants from sessions.

IL3 behavior for Touring Machine's participant invitation was described in sections 3.3.3 and 4.5.6. It can occur only when a Touring Machine client, who is in a session with at least one MMCC user, invites another Touring Machine client to the session. Furthermore, in such situations, IL3 behavior occurs only when MMCC's media agents fail to reconfigure properly, the infrequency of which has already been mentioned.

When session descriptions are relayed between domains, the gateway can easily translate simple conferencing concepts between the two systems' syntax. Both systems can specify participants and symmetric video and/or audio conferences. However, Touring Machine's `sessionCreate` and `sessionChange` messages give the application much flexibility. They allow the initiator to specify the details of asymmetric connections. MMCC does not support asymmetric conferencing. The gateway can recognize asymmetric session requests from the Touring Machine domain in the form of `sessionActionRequest` messages. The gateway denies them with a `sessionActionDenied` message which contains describing a capabilities mismatch. The result is IL2 functionality for Touring Machine clients attempting asymmetric session establishment or modification with MMCC users.

The `sessionChange` message also allows Touring Machine clients to initiate the removal of other clients from the session. The client(s) to be removed are queried with a `sessionActionRequest` for their approval before the action is committed. However, since MMCC has no such functionality, the gateway provides IL2 behavior by always denying such a request with a `sessionActionDenied` message.

Similarly, the gateway denies session requests from either domain that involve data media. The reason is that the semantics of the data transmission are defined by the application--actually, MMCC media agents control the types of data transmission, but applications choose the media agent. The gateway can satisfy video and audio session requests because it can assume that participants want to see video and hear audio. The format of such media is independent of how it is used. The gateway need only insure that the audio and video formats are properly converted between the two transport networks.

In the Touring Machine/MMCC case, The gateway is saved from controlling extra converter equipment because MMCC media agents and workstation hardware can convert to Touring Machine's analog transport. However, the gateway cannot make any analogous assumptions about how different applications want to interpret data in general. If both systems used more specific media formats of data whose interpretation is clear--audio and video are such formats--then the gateway could conceivably translate requests involving those media. For instance, if both systems had a "fax," "still-image," or "shared XWindow" data type, it could make sense for the gateway to establish the session and allow lower level mechanisms to reconcile format differences.

Touring Machine provides an inter-client, text message passing service.[6] The interpretation of messages is left entirely to the application. MMCC has no analogous service, and the existing-applications interoperation model does not support application-level signalling. Furthermore, the gateway cannot provide IL2 behavior. If a Touring Machine client sends an inter-client message to an MMCC user, the gateway would intercept the message and realize that no meaningful processing can happen with the message. The only way to notify the sender that the message delivery was unsuccessful is for Touring Machine to use a messageSendFailure message. Unfortunately, from Touring Machine's viewpoint, the message delivery was successful. To provide IL2 behavior, the gateway would have to simulate a messageSendFailure message from Touring Machine to the original sender. This kind of solution does not seem justified for a service that is not a part of the interoperation model.

Touring Machine's directory service lets its clients access information about MMCC users without trouble, since the information is entered into the database statically. However, as discussed in section 4.2, the "active user" service cannot be provided for MMCC users.

5.1.2. Performance Evaluation

Section 2.4.3 described gateway design goals only in terms of correct behavior of conferencing functions. The speed of conferencing functions also plays a role in the usability of gateway solutions. As might be expected, the amount of time it takes for inter-domain session establishment is approximately the sum of the times for normal session establishment for each domain. The same can be said for inter-domain session termi-

nation and participant invitation. Section 4.3.3 already mentioned that added delays for callee acceptance can be annoying for the caller.

Nonetheless, the added delays are tolerable because the users know when they are requesting inter-domain conferencing. Otherwise, users would experience what seemed like erratic behavior--conferencing with some users would be much slower than with others. In some sense, expectations of slow performance can ameliorate the annoyance of the delays. This is true because the severity of the delays is not extreme. Also, it is important that the speed of conferencing within each domain is unaffected by the presence of the gateway. Undoubtedly, if some or all intra-domain conferences had added delays, some users would prefer having faster local conferencing to having any inter-domain functionality.

5.2. Conferencing System Traits That Benefit Gateways

From the Touring Machine/MMCC experience, some general statements can be made about conferencing system characteristics that lead to more successful gateway solutions.

5.2.1. Common Conferencing Function Subset

The most important problem that gateway solutions present is the ability to provide only the common subset of conferencing functions. Thus, the first requirement for a useful gateway solution is the availability of the desired conferencing functions in both systems.

5.2.2. Reversibility of Session Procedures

A reliable participant self-removal protocol is key for reconciling session management. Touring Machine's policy of leaving participants in session when errors occur greatly handcuffs the capabilities of the gateway solution. When the gateway can initiate reliable session termination in either domain, session establishment is possible as long as there is some confirmation to the caller of the success of the session. Even if the mismatch of message semantics is much worse than that in Table 2, the gateway will be able to reverse session establishment in a domain if it is notified that session establishment was unsuccessful in the other domain. Inter-MPU communication will still have the same nature--i.e., CONNECT and REPLY from section 4.6 would still be used similarly.

In general, it is best for the gateway to have a mechanism for reversing any session procedure. The effect of having irreversible procedures is mentioned in section 3.3.3 for participant invitation. For such procedures, IL3 behavior can be a possibility in some cases. In those cases, a choice must be made to provide consistent IL2 behavior or IL1 behavior with some chance of IL3 behavior. Although IL2 behavior is clearly preferred in general, if IL3 behavior is extremely rare and the effects of the IL3 behavior are not severe, it may be better to choose IL1 behavior with some chance of IL3 behavior.

5.2.3. Distributed Control

A distributed control architecture tends to make gateway design more feasible. Distributed architectures leave more control at the “ends” of the network--i.e., where applications and the gateway reside. An example of the benefits of distributed control is the need for reversible session actions. Reversible session actions are not as necessary if more control of resource actions is given to applications. For instance, Touring Machine handles all resource-related processing centrally. As a result, IL3 behavior, as described in section 3.3.3, can occur from a Touring Machine client inviting another Touring Machine client. On the other hand, since MMCC needs to gather resource information from distributed instances, the gateway can act before resource configuration is finalized in the other participant invitation scenarios. For similar reasons, session establishment with MMCC callees can be accomplished without session termination--see Figure 4.5. However, there is no alternative with Touring Machine callees in Figure 4.7.

Another example of how distributed control helps the effectiveness of gateway solutions is MMCC's multiple callee acceptance policy. Sections 4.3.3 and 4.3.4 illustrate how MMCC's flexible policy of leaving control to the application facilitates the gateway design, while Touring Machine's centrally controlled policy inhibits it.

5.2.4. Specification of Data Media

Data media should be specified to convey how the media will be used. Gateways have the potential to convert between any two formats if the interpretation of the media type can be assumed. Audio and video are popular in conferencing systems and the gateway can always employ some converter mechanisms between the transport networks. Similarly, the gateway can handle different formats for still-image transfer, text, etc. For

instance, if Touring Machine used PICT for a still-image medium and MMCC used JPEG for its still-image medium, the gateway's inter-domain resource management could convert between the two. More session types would be feasible without affecting most of the gateway's processing.

5.2.5. Control Message Parameters

There are characteristics of control messages that make gateway design easier. As mentioned in section 4.1, message interception benefits from the inclusion of the recipient's identity in a control message. Also, multiple party session establishment, as described in section 4.3.3 and 4.3.4, benefits from control messages including all the participants' identities.

5.3. Future Work

The extensibility of the user proxy gateway design can be further explored with experiments between different conferencing systems. Observations could be made about the similarity of the design processes and of the effectiveness of the resulting gateway solutions to the Touring Machine/MMCC case. Efforts--akin to those in protocol conversion--could begin toward seeking general methods for gateway design.

The design of adaptors to provide conferencing functions outside the common function subset also requires further study. In addition, strategies for handling complex inter-domain transport scenarios will be important for general gateway solutions.

As each of these issues is examined, the potential of conferencing gateways as part of a global strategy can be studied. As teleconferencing approaches ubiquitousness, it is unclear if the needs of user communities can be met with gateway-based solutions, standards-based solutions, or a combination of each. Much will depend on the diversity of computing and network environments as telecommunications infrastructure evolves.

References

- [1] M. Arango, et al., "Touring Machine: A Software Platform for Distributed Multimedia Applications," *Proceedings 1992 IFIP International Conference on Upper Layer Protocols*, Vancouver, Canada, May 1992.
- [2] G. Bochmann and P. Mondain-Monval, "Design Principles for Communication Gateways," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 1, January 1990.
- [3] M. Chen, et al., "Software Architecture of DiCE: A Distributed Collaboration Environment," *Proceedings 4th IEEE ComSoc International Workshop on Multimedia Communications*, Monterey, CA, Apr. 1992.
- [4] P. Green, Jr., "Protocol Conversion," *IEEE Transactions on Communications*, vol. 34, no. 3, March 1986.
- [5] W. L. Hill and A. K. Ishizaki, "A Call Model for Distributed Multimedia Communications," HP Laboratories Technical Report, January 1993.
- [6] V. Mak, et al., "The Application Programming Interface to the Touring Machine," Bell Communications Research, Morristown, NJ, February 1993.
- [7] Minutes of the Multiparty Multimedia Session Control Working Group (MMusic), *Proceedings 31st Internet Engineering Task Force*, San Jose, CA, December 1994.
- [8] K. Okumura, "Generation of Proper Adapters and Converters from a Formal Service Specification," *Proceedings IEEE INFOCOM*, Austin, TX, 1990, pp. 564-571.
- [9] Proposed Draft of T.120, International Telecommunications Union/ Telecommunications Standards Sector, Question 10/Study Group 8, March 1995.
- [10] E. M. Schooler, "Case Study: Multimedia Conference Control in a Packet-Switched Teleconferencing System," *Internetworking: Research and Experience*, vol. 4, no. 3, June 1993, pp. 99-120.
- [11] E. M. Schooler, "The Connection Control Protocol: Architecture Overview," Version 1.0, USC/Information Sciences Institute, Marina del Rey, CA, January 1992.
- [12] E. M. Schooler, "The Connection Control Protocol: Specification," Version 1.0, USC/Information Sciences Institute, Marina del Rey, CA, January 1992.
- [13] H. M. Vin, et al., "Multimedia Conferencing in the EtherPhone Environment," *IEEE Computer*, vol. 24, no. 10, Oct. 1991, pp. 69-79.

- [14] Yow-Wei Yao, et al., "A Modular Approach to Constructing Protocol Converters," *Proceedings IEEE INFOCOM*, Austin, TX, 1990, pp. 572-579.