Chapter 1

Introduction

This chapter provides a general introduction to the thesis and sets out the motivation for the research in multimedia group communication. It also outlines the structure of the thesis by giving an overview of other chapters.

1.1 Background

Multimedia refers to the text, video, audio, graphics, etc., which comprise different media types integrated in one system to convey information to human users. From a computing perspective, these are means of communication between humans and computers - and between humans using computers as communications tools [Herrtwich 92]. In a distributed computing environment, this requires network support to provide the multimedia functions.

Different media types have different characteristics. For example, some media, such as text and graphics, have time independent values. These media are called *discrete*. Other media, such as audio and video, have values that change over time and these changes contribute to media semantics. These media are called *continuous*. However, the terms "discrete" and "continuous" do not refer to the internal data representation in a computer system, but to the user's view of the data. When media types are related for simultaneous delivery, it is necessary that they are synchronised for the media playout. Hence synchronisation of different media types involved in communication is an important component of multimedia applications. When establishing a multimedia connection (i.e. a call) between the various end-system devices, it is required that the operations on applications (programs handling operations on multimedia playout) are made in synchronous manner without the concern for the behaviour of individual media involved, for effective presentation. This provides the functionality of perception in a more natural way like a TV application. But, unlike a TV application, the computer and multimedia enable interactive communication between the user and multimedia systems [Steinmetz 92].

Besides the synchronisation services, another vital requirement for multimedia applications is the set of data transmission services. The data transmission services are concerned with connection management between a group of multimedia users, providing appropriate Quality of Service (QoS) for each participant. On the other hand, individual use of the distributed multimedia resources is entailed in membership management services. The basic functionality of group management services is to maintain a membership list.

From the foregoing synopsis on multimedia communication it is not surprising that multimedia applications have become issues of great interest in research. This is because of their potential for conveying information and enhancing communication in distributed computing environments. For example, applications such as distance learning, multimedia conferencing, video on demand, computer supported collaborative work, and various applications in the world of entertainment e.g. virtual bands, are potential beneficiaries of multimedia communications.

Many issues have been raised on how best to support multimedia communication. This is because multimedia requires a new communications infrastructure among which are:

- the ability to integrate media types as well as being able to manipulate the individual media;
- supporting the spatial and temporal requirements of continuous media.
- the ability to support group based multimedia applications.

While new high speed networks provide the potential to support high bandwidth for multimedia communications, there is a need to provide mechanisms that would enable integration of different media types in multimedia applications, especially for group oriented applications.

1.2 Multimedia Multi-user Applications

There are currently numerous distributed group or multi-user applications such as those described by [Hagsand 91], [Dollimore 92], [Wilde 93], [Kovacs 94], [Pendergast 95], [Muller 96]. These applications often involve only one or two types of communication media, such as text, audio, video, combined audio/video, or game simulation. Combining invocations of several of these applications forms a distributed multimedia group session (or *call* in some literature [Vin 92], [Heijenk 94][Waters 95]), where several users (a distributed group) agree to have a meeting (a session) using an array of media types (multimedia) to support their co-operation. As the number of users and media types increases, there is clearly a need for some form of session-management facility. The scope of the thesis focuses on area, i.e. the multimedia group session management.. The motivation for this endeavour is discussed in the following section.

1.3 Motivation

Current distributed multi-user (or group) applications manage their own connections to achieve the multi-user effect [Garcia 93], [Wilde 94], [Trevor 95]. This often presents difficulties when several applications of different types (e.g. video conferencing, virtual whiteboards) are managed as a single multimedia application. Although several group applications may be combined into one (such as a combined video/ audio/ text/ whiteboard conferencing system), if the combined application cannot have new media added dynamically, the management problem still remains. The systems are therefore non-integrated. Session management needs to be dynamically configured to allow newly developed types of media to be incorporated.

A number of session management architectures have been defined. These session architectures rigidly define a session or call as being an object which contains a number of channels of different media types (called media channels). Media channels are representations of running applications: a video-conferencing channel is an instance of a video-conferencing application. A single application may manifest itself as several channels, such as when an audio-conferencing channel and a video-conferencing channel together form a complete audio/video conference.

Users that connect to a call may participate in each of its channels in various ways: as a producer, as a consumer, as both, or as neither. The precise rules for connection (protocols and restrictions) vary with each architecture. In some, users must participate in certain ways before any connections take place. The rules are usually configurable, and are fully supported by the architecture.

Problems arise with these architectures when external resources, including other sessions or calls, are involved. For example, by adding a video-on-demand (VoD) application to the call (i.e. by adding a new channel), there is now a new user, who is not one of the original users, who provides a video to be played, and could therefore be seen as a (unique) producer of the new channel. This leads to complications. How do original consumers become of the new producer.

There are also problems with formation of sub-calls and call-call interaction. How do you merge the calls? And how do you extricate them afterwards? What if the calls involve different sets of media-channel types? These issues are not addressed in the current architectures. Furthermore, the existing architectures often define the rules of participation, preventing more complex rules from being used.

From the current literature, it is stated that a multimedia group session involves a set of users registered against a set of applications, and all users participate in all applications [Heijenk 94]. However, this approach is too restrictive for some of the configurations such the individual participants requiring to participate in a session in certain ways. A participant may wish to join a session as a consumer, a producer, or both for only certain media types involved in the multimedia group session. There is also the need to form sub-group sessions with the same sets of media types and users. Therefore, a flexible approach to multimedia group session management is desired.

Therefore this aims at designing an open architecture for multimedia group management services whereby only the protocol for participating is defined - the permissions and constraints are left to the programmers of multimedia applications.

1.4 The Structure of the Thesis

The general group communication and group based activities are reviewed in Chapter 2. This is done with the aim of analysing the various group mechanisms and functions that have been used for group communication and to discern the extent to which the models derived from group communications support may be applied to multimedia group applications. The chapter also discusses past and ongoing activities in the area of group communication.

The review of group based activities gives pointers to the requirements for support of multimedia group based applications. These requirements are analysed in Chapter 3. The aim of the analysis is to identify the multimedia group management services, and their functions in a distributed computing environment.

In Chapter 4, abstract services for multimedia group management are defined. Subsequently, a design of a model for the services and the mechanisms used for by the services is obtained. The architecture of the model is contrasted against the existing architectures, which are obtained from a review of related work elsewhere.

Chapter 5 presents an implementation of the model. A protocol for communications among objects of the model is defined. This implementation provides the user with a set of library functions which can be configured for application-specific needs. In Chapter 6, performance of the model is evaluated. Also scenarios where the model could be applied are discussed.The conclusion in Chapter 7 summarises the issues discussed in the thesis, pointing out the novel contributions of the research, as well as directions of possible further work.

Chapter 2

A Review of Group Communications Activities

As explained in *Chapter 1*, this thesis is about management services for the support of group communications in distributed multimedia applications. However, it is important to give an overview of group concepts and group communication in order to discern the various approaches used to effect group communication. In addition, this chapter also examines some of the past and ongoing activities in the area of group communications with the purpose of verifying how these activities have been employed for group communications.

Group communication, especially in an environment employing *multimedia,* has become an active area of research. As will be described later, in *Chapter 3*, multimedia applications requires the support of both the real-time media transmission, so called *continuous media* type, of which category are audio and video sequences, as well as the support for *non-continuous media* types such as text and graphic images.

Group applications such as computer aided learning, video conferencing, Computer Supported Co-operative Work (CSCW), joint editing, and group decision making require the use of multimedia to augment communication and hence enhance the collaborative activity. A further characteristic of collaborating entities, be it machine processes or human beings, is that the entities will normally be geographically distributed, but require to exploit multimedia communications to bridge distance, time and space. But the use of multimedia imposes a variety of demands on communication systems, ranging from the mode of communication by group members (i.e. whether synchronous or asynchronous, which requires certain predictions about resource reservation), to presentation or media play-out, of which the latter relates to the quality of service, synchronisation of various media types, and group management.

While it is true that management services are well known in distributed computing systems, multimedia group communication poses new challenges since there are more players and various media types involved in the communication. In reviewing the general group communications concepts and seeing how they have been hitherto exploited in support of groups, it should be possible to verify why multimedia group communication requires different management strategies.

The chapter is structured as follows: *Section 2.1* introduces general concepts on group communications both literally and technically. In *Section 2.2,* aspects of group communications are discussed, while *Section 2.3* discusses the models and toolkits used to design group applications. *Section 2.4* presents some of the recent related work on multimedia group communications. The implications of this review work are discussed in *Section 2.5,* while the summary of the chapter is presented in *Section 2.6. Section 2.7* is the conclusion, which sets the *motivation* for the research in multimedia group management in distributed computing systems.

2.1 General Group Communications Concepts

The meaning of the word *group* is used to identify a set of individual objects which are related in some definite way differentiating them from others. In communication systems this cluster of objects (e.g. set of users, object entities, a set of peers, processors, etc.) shares common services and facilities accessible through high level *abstract* interfaces. The purpose of introducing groups into communication is to allow a collection of processes to be dealt with as a single abstraction [Tanenbaum 92]. This is desirable in a distributed computing environment where several co-operating processes are involved in communication and hence should be managed as one entity. This idea is often employed in distributed operating systems where we have client groups on the one hand and server groups on the other, for example, the Interface Groups described by [Oskiewicz 1990]. The Interface Groups are divided into server and client groups as shown in Figure 2.1 The server group enables a collection of interfaces to provide a single service at a single interface. Server groups mask replication and distribution by hiding the details of addressing, parallel execution, consistency and error recovery, from clients. On the other hand, client groups are means by which a group of objects appears to be a single entity to the server group, since a service is provided at a single interface. Therefore, the Interface Group application mechanisms provide group transparencies, a client treats a group's service as if it were provided by a single server, i.e. a single call is made and a single result may be obtained from the server group. Also a member of the server group may not know the existence of the other co-servers. The key objective of such an arrangement is to make the network transparent to the user or automated object, who both assume that they are continuously interacting with a single stand-alone system. The group transparency exhibited in distributed systems include communication transparency, reply-handling transparency, naming transparency, failure transparency and real-time systems transparency.

On the other hand, group members who interact, and who would be typically at different distinct locations, do so through abstract interfaces. The abstract interfaces encapsulate the group's internal state, provide location transparency, and hide how the group members interact. Figure 2.2 illustrates group member interactions at abstract interfaces.

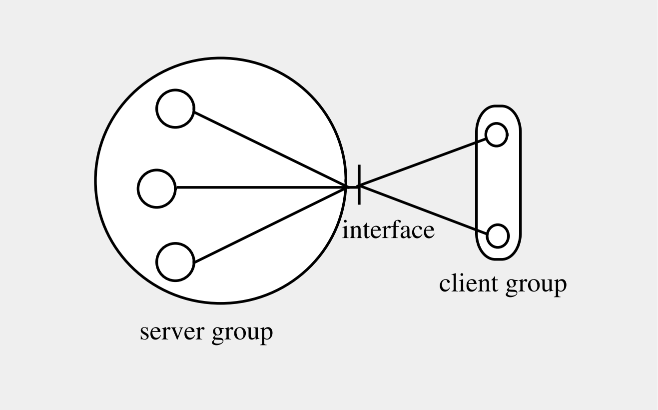


Figure 2.1: Interface Group abstraction

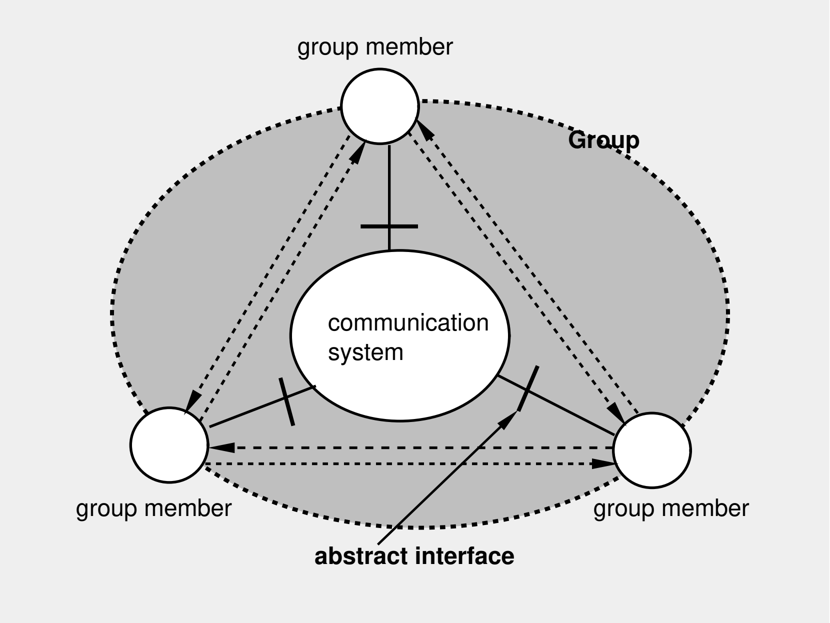


Figure 2.2: Abstract interactions of group members

The following sections describe group characteristics. As will be seen, characterising groups provides us with the means by which to model applications that apply group communications concepts and subsequently to develop appropriate technologies for such applications for different group communication domains. In fact, this is essential to the thesis: it is necessary to examine the general concepts of group characteristic features and see to what extent they can be applied to support multimedia group management.

2.1.1 Group Characteristics and Features

In order to develop group communication abstractions, there are certain basic group characteristics to consider as they influence both design and implementation decisions made to support specific applications. Groups have distinct features. The group features refer to the way groups are organised; this covers attributes characterising the group as an entity, how group members interact among themselves or with a central object. This interaction is referred to as a *group session*. A session is an instantiation of an *active group*. In [Cocquet 95a], an active group consists of group members that have accepted to participate in a group-association, and who should have initially enrolled with the group. While members of a given group interact with one another, it is also possible to have inter-group interaction or communication. This may be seen in a situation whereby two or more groups are engaged on the same task but on different aspects of the task. In such a case, the session consists of sub-sessions and usually there will be session managers responsible for each session. The session managers provide the interfaces for inter group communication. Figure 2.3 illustrates this situation.

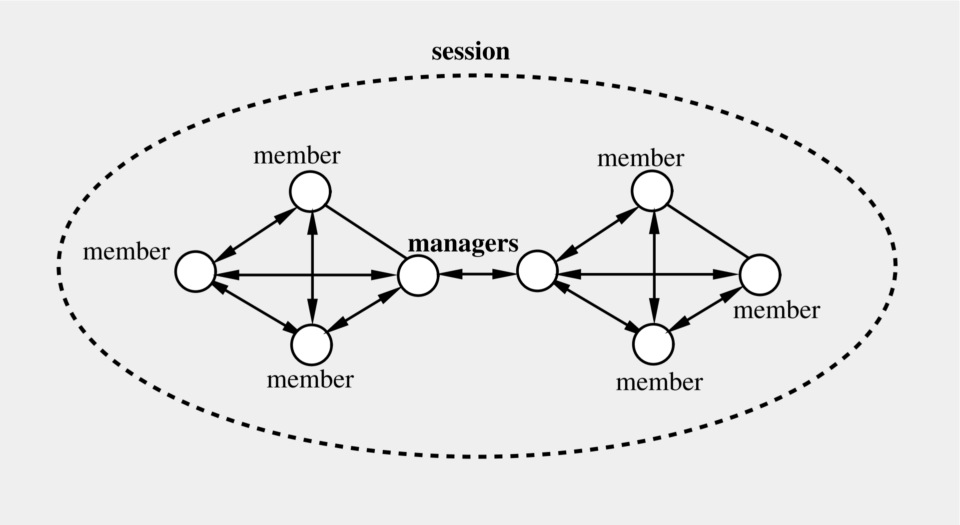


Figure 2.3: Composite group session

The characterisation of group communication has been recently a subject of much deliberation in the Multi-peer Taxonomy forum [Cocquet 95a] as a work item with the CIO project[1]. The objective of this activity is to be able to formulate a consistent group communication taxonomy that applications developers of group communication services can use at all levels of communication. An outline of the group features is as follows.

2.1.1.1 Group Membership

An entity becomes a member of a given group when it subscribes to the group or

---

[1] CIO: Coordination, Implementation and Operation of Multimedia Services (A RACE 2060 project)

qualifies for a given membership criteria. The criteria are defined a priori as membership rules or group (management) policy. Group membership may be defined as *static* or *dynamic.* Within a static group, the group membership remains permanent throughout the group's lifetime, while in a dynamic group, members are free to join and leave the group. Members of a group may be *active* or *inactive*. Group members may also belong to other groups. Some members may be involved in performing specific group activities. These members may be classified as *group masters, initiators, managers* , etc.

In general, a group member identifies with the group by means of a group name(*group id*) and all group members can be reached by a single *group address*.

2.1.1.2 Characteristic Features of Groups

All attributes concerned with group organisation and group membership make up a group's characteristic features. As noted before, group members that may potentially participate in a group communication form an *active group*. A group where currently no communication can take place is called a *passive* group. Groups that allow a change in membership during their lifetime are called *dynamic groups,* otherwise they are *static groups*. In *open groups* non-group members can take part in the communication whereas in *closed groups* only members are allowed to send or receive group messages. *Restricted* groups restrict communication to members of the group. *Unrestricted* groups allow the communication of their members with other individuals or groups. Groups can be *permanent* (e.g. directory services), *long-term* groups (e.g. application groups comprising a long-term co-operation) or *short-term* groups (i.e. a group is just setup for a session). In *determined groups* individual members are known to each other. *Partially-determined groups* refer to groups where a sub-group of members is known to one or more other members. In *anonymous groups* single group members are unknown to each other; an e-mail list is an example of such a group. Figure 2.4 shows the simple group structure, while Table 2.1 summarises the group features.

Participants in a group communication can be related through different roles and rights in the group, the kind of data exchanged, and through interaction and operations affecting the group or individual participants. Further, a distinction can be made among group members by their different roles, i.e. *initiator, master, voters* (members who have to give a vote for certain operations), *acting* and *non-acting* participants, etc. Participants can be *active*, i.e. currently participating in the group communication and able to receive the exchanged data, or *passive*. These features are regarded as group interrelationships.

A further dimension of interrelationship is the notification of operations on shared objects and of changes in the state of the group and its members. To be aware of other participants it is necessary to notify the whole group or a sub-set of participants of operations and actions of single participants (i.e. joining, leaving, etc.). Operations that have to be notified are called *noticeable* operations, participants that have to be notified are called *affected* participants. Table 2.2 shows the interrelationships among group participants.

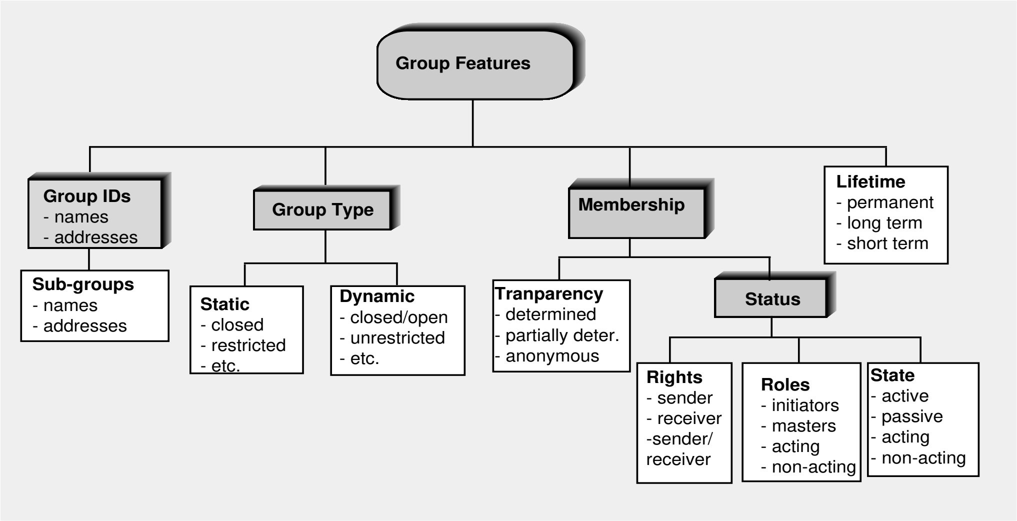


Figure 2.4: Group Structure

Features	Parameters
state	active, passive
constitutional properties	static, dynamic; open, close; restricted, unrestricted
lifetime	permanent, long-term, short-term
membership transparency	determined, partially-determined, anonymous

Table 2.1: Group Features

The kind of interaction which might occur between participants is the mode of communication. Interaction between entities can be *asynchronous* (participants do not have to be present at the same time) or *synchronous* (all participants have to be present at the same time). In the synchronous case interaction can be either unrestricted or restricted through certain mechanisms (e.g. floor control or token mechanism). This can apply to the whole group or to a sub-set of participants. *Uncontrolled* interaction allows unrestricted sending of data in that all participants are allowed to send data at any time. *Controlled* interaction refers to a restricted data exchange between participants. Rights towards data exchange can change during the lifetime of the application (*dynamic*) or be defined for the whole communication (*static*). Changes in group interactions are controlled and restricted by rules or conditions. If only a sub-set of participants has distinctive rights for data exchange we have *sub-controlled* interactions. The membership in a sub-set can be *variable* or *fixed.*

Interrelationship	Parameters
affiliation	group member, non-member
role	initiator, master, acting, non-acting; voter
state	active, passive
notification	affected, non-affected

Table 2.2: Interrelationship Characteristics

2.1.1.3 Group Services

In order to support group communication, many services are provided for specific applications. These services, as mentioned before, are accessible through high level abstract interfaces. Three main categories of services exist: these are: *group management services*, *group communication services*, and *synchronisation services* [Fortier 88], [Mauthe 94].

Group management services deal with general activities relating to the housekeeping of a group which supports a specific application domain. These services implement the various policies which provide the group with its own individual properties, hence, making one group distinct from others. Generally, group management is performed by a unique computational object or group member called the *group manager*. The minimum functionality performed by a system of group management services is that of controlling the group's population. The management entity maintains a record of the population in the group, controls changes to this population and may supply information about this population when requested. The record of the population is generally referred to as the *member list*. Around this list many other management policies specific to the group communication requirements may be established.

Furthermore, group management services also relate to group operations. Group operations provides the means by which a group is created, group members interact, and by which potential members may subscribe to the group. The operations are usually invoked on a management entity which may or may not be a member of the group. These operations are responsible for the creation of the groups as well as the administration of groups and their members.

The group operations may involve services that require to create a group, to destroy a group, to name or identify a group (group addressing).

Group membership operations comprise such services as joining a group, leaving a group, sending messages to group members, receiving messages from group members, and notifying group members.

In addition to the group and membership operations, there are also query operations on groups and group members. These operations involve querying information about groups and group members, e.g. groups in a given domain, and a group's membership list.

The query services are usually manifested as directory services. A directory service will normally store static data such as groups characteristic features which may be retrieved on a query. The directory service is one of the major components of a group management system.

On the other hand, group communication services provide abstractions over communication protocols which essentially support many-to-many communications between the group members, while synchronisation services provide real-time synchronisation between semantically related media streams, and ensure that the delivery of messages from different users is such that all users see the same ordering of events.

The above general principles and concepts on group communication have been variously exploited in building group communications applications. A review of some of the areas of applications is presented in the next section.

2.2 Group Communications

Group communication abstractions form essential building blocks within various open system domains spanning client-server network models, distributed processing systems, computer supported co-operative work (CSCW), and, generally, distributed multimedia group communications. These communication domains enable group applications for interactions among human users on one hand, and process groups for communication systems on the other.

Group interactions using computer mediated support include applications such as Video on Demand, Video Conferencing, and co-ordinated work such as joint editing, CSCW, and group decision making (voting). As these are high-level groups, they require only the setting up of communication facilities, assuming that the group membership is constrained by the group policy. The basic group member interactions and group applications follow the patterns described in the AMIGO project [Smith 89], and in [Benford 92], [Bannon 90]. The AMIGO project conducted research into messaging in group communications, focusing on messaging system of X.400 in particular. Although this study was limited to a specific aspect, the modelling of groups within AMIGO provides a source of basic requirements for group management services [Wilde 94]. The minimum facilities required to set up such group communications and applications are basic conference facilities, which have been employed in projects such as MICE [Handley 93], and MICA [Vonderweidt 91],

The process group communication is a well known phenomenon especially in the area of distributed operating systems such as Amoeba [Kaashoek 91], V-kernel [Cheriton 85], and Chorus [Fortier 88]. In these cases the objectives have been to provide mechanisms for supporting communicating entities as processes to enhance fault-tolerance, reliability and location transparency (as was mentioned in *section 2.1*). This largely has been the essence of process group and in the literature this was described by [Liang 92]. The process group communication abstractions have been applied in developing toolkits such as ISIS [Birman 89], [Davies 94] and the ANSAware [Oskiewicz 1990], [Herbert 94], [Coulson 92], [Bauer 94]. Such tools are commonly known as *groupware*. The term groupware has emerged to signify software systems which address the co-operative nature of the work [Hagsand 91], [Williams 92], making the user aware that he or she is a member of a group in a distributed system. This co-operative work or the CSCW as it is commonly known, enables people to share information (ideas, files, processes etc.).

In summary, this section has outlined the basic group communications concepts. In the next section, some of the models and toolkits that have been designed to aid in developing group communication support services are presented. The aim is to identify some of the mechanisms used by the models to provide the required services.

2.3 Models and Toolkits for Group Communication

A number of group communications models have been proposed in the literature. In [Simon 96], six main paradigms commonly used to structure the nature of co-operating processes have been identified. These are:

- the master-slave model
- the client/server model
- the peer-to-peer model
- the group model
- the distributed object model
- the multimedia streams model

An overview of these models is as follows:

2.3.1 The Master-Slave Model

In this model, the master process initiates and controls any dialogue with its slave processes. The slave processes show very little intelligence, only responding to commands from the single master process. The configuration of the model is shown in Figure 2.5. As can be inferred, the model has limited application in distributed systems as the slave processes do not make use of distributed resources. However, the model was popular for on-line centralised computers running time-sharing applications which required this sort of environment.

2.3.2 The Client/Server Model

The client/server model is a widely used paradigm for structuring distributed systems communications. Processes called *servers* are responsible for the provision of services to clients. This is the case illustrated by the interface group in Figure 2.2. The services are accessed via a well-defined interface which is made available to the client group. When a server or server group receives a request from a client, it executes the appropriate operation and sends a reply back to the client. The mode of communication or interaction in this model is request/reply. A client can potentially generate a request for a service at any time. A server may have multiple interfaces e.g. a management interface and a separate interface through which client requests are handled [Oskiewicz 1990].

The request and response actions of the client and server processes are complementary and with distinct roles, unlike in the master-slave model. Sometimes it is desired that a small manageable number of servers be configured as clients and servers (for centralised resources) which improves management, instead of having potentially each computer being configured as a client or a server. The model described in the former case is known as a **peer-to-peer** model, because every process has the same functionality as the peer process. The configuration of the model is shown in Figure 2.6.

2.3.3 The Group Model

In this model, a set of processes co-operates in a number of ways in sending and receiving messages. In one way, a process may send a message to all other members of the group and may receive responses from one or more members of the group; or in the case of a news group, when a subscriber posts a news item, all other subscribers receive it. In video conferencing, multiple participants are involved and may have facilities such as whiteboards whereby if someone writes on the whiteboard, every other participant should receive the new image. Therefore participants act as both senders and receivers of

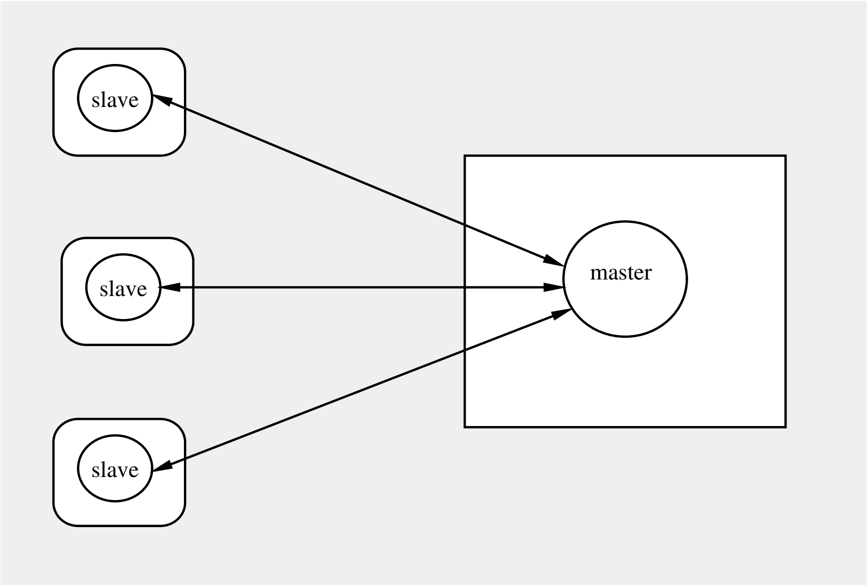


Figure 2.5: Master-slave model

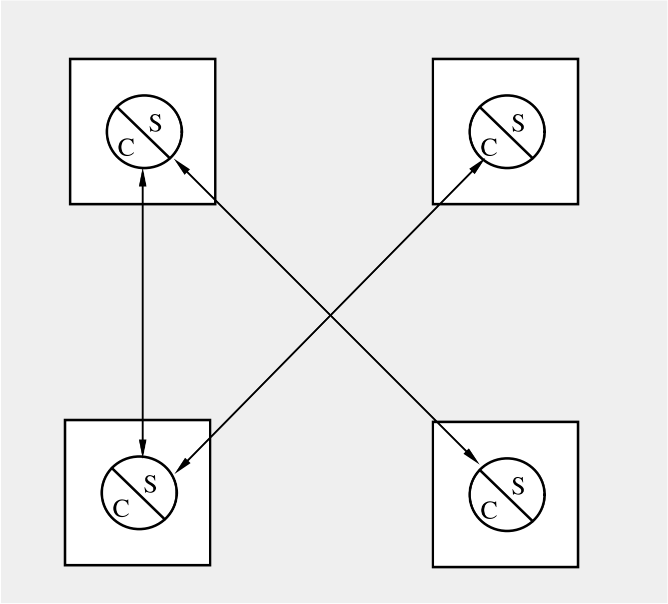


Figure 2.6: Peer-to-peer model: C = client function; S = server function

the group messages. Figure 2.7 [Simon 96] is an expanded group communication model based upon Figure 2.1, illustrating the process of a group member sending messages to other group members. This model is explained Chapter 5 with respect to how the underlying communication systems should provide the support for group communications. The underlying communication semantics to support group communications are:

- one-to-one (peer-to-peer)
- one-to-many (variously known as multipeer, multiparty, or multicast)

- one-to-all (broadcast)
- many-to-one (concentration).
- many-to-many

These issues are in the remit of data transmission services, and reference can be found in [Sloman 1987], [Waters 92], [Mauthe 95b][Rezende 95b] among others in the literature.

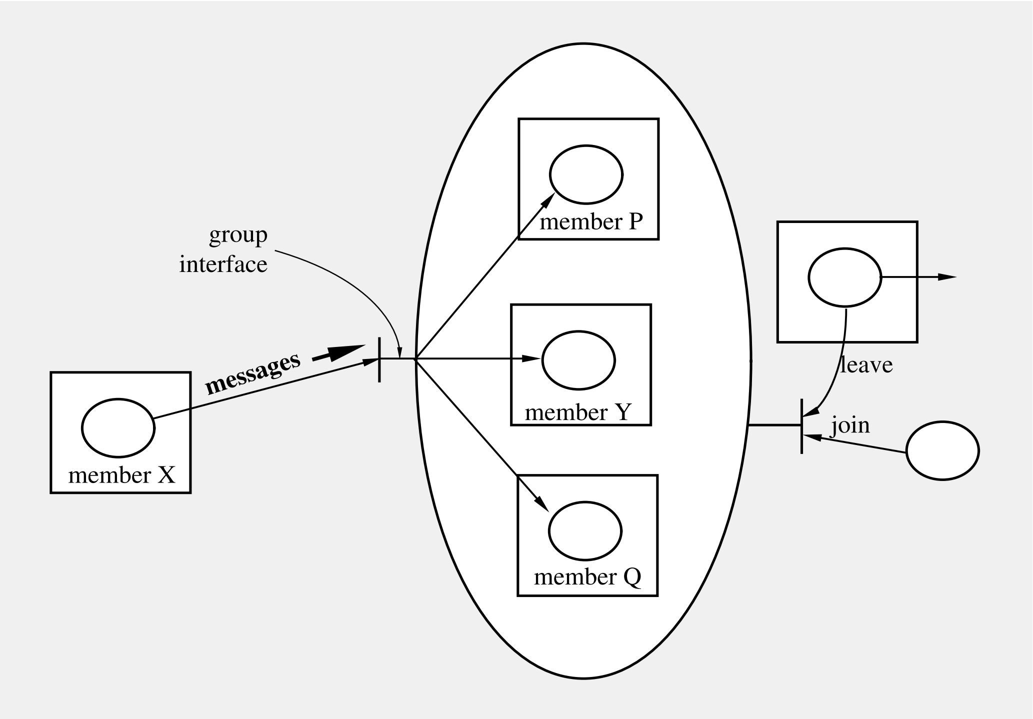


Figure 2.7: The Group Model

2.3.4 The Distributed Object Model

The object model offers a different philosophy of structuring and maintaining information in a distributed system. An object is described in terms of its *attributes* (internal state data) and methods (procedures which operate on the state data). In this way an object encapsulates a set of capabilities that defines its interface to the outside world, and through which an object's services are manifested. Since an object's internal state is hidden, the only way of examining its state is by sending a message through one of its capabilities (or methods).

Objects communicate in a manner similar to the client/server model, i.e. one object sends a message to another and waits for a response. However, unlike the client/server model objects exist independently of each other, interacting only through their service interface.

As noted in [Simon 96], an object model provides important advantages in the development and maintenance of distributed information systems. This is achieved through the perceived object's characteristics viz. *encapsulation, classes*, *polymorphism* and *inheritance* [Booch 1994], upon which object-oriented designs are based. For example a management service system might cluster members into *classes* rather than having several different lists. Managed objects at lower layers *inherit* the characteristics of their "parents". Thus the initial relationship of group members is maintained, and channels can be managed as high level links between users. Those links are represented by several transport connections, and represent connection management of distributed group applications. Therefore, although each layer performs different functions, the management of an individual layer would be essentially the same.

Objects can be written to handle a wide range of media types, using essentially the object interface (*polymorphism*) which provides a consistent approach to handling multimedia objects. This is an attractive approach to the modelling of multimedia applications and has been exploited in the model for multimedia group management services presented in the thesis. This is discussed in Chapter 4 and subsequent chapters.

2.3.5 The Multimedia Stream Model

Multimedia streams is an emerging paradigm for supporting group communications due to integration of different media types to augment group interaction. Modelling of interactions between multimedia objects uses object-oriented and distributed object paradigms [Simon 96]. As described in the above section, objects provide service interfaces and service semantics. In a distributed environment, the multimedia object communication must keep consistency in semantically related media streams, e.g. audio with video, as well as meeting real-time requirements of the media streams of given *media types*. A generalised architecture for modelling multimedia data streams is shown in Figure 2.8 [Simon 96].

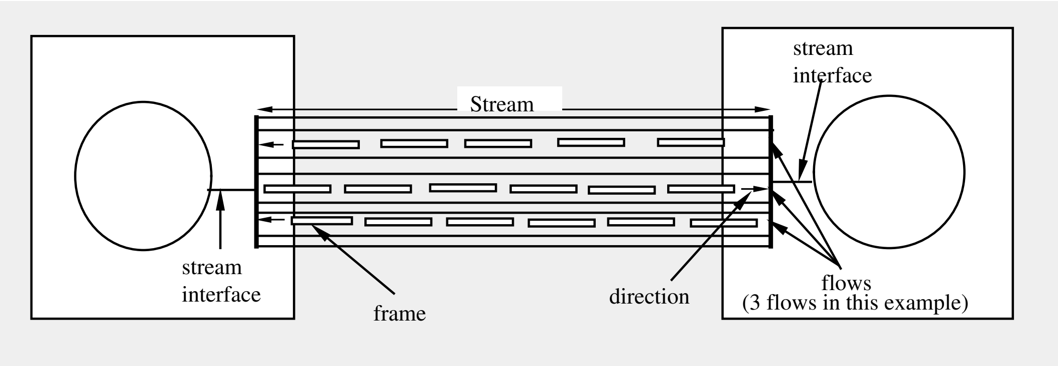


Figure 2.8: Multimedia streams model

A multimedia stream can be described as a continuous medium with a well-defined start and stop end, which takes place at a defined rate and exhibits [Linington 94]. The multimedia stream model encompasses the following concepts:

- A multimedia stream interface is one in which all interactions are **flows.** A is an abstract ion of a sequence of interactions, resulting in conveyance of information from a producer object to a consumer object. For example, a video conferencing stream interface may consist of three flows: audio, video and data.
- Each stream interface comprises a finite set of **action templates**, one for each flow type in the stream interface. Each action template for a flow contains:
    1. the name of the flow;
    2. the information type of the flow (e.g. audio, video or data);
    3. whether it is a consumer or a producer (but not both).
- A flow has a set of **frames** (or signals).
- A frame has a name and a set of typed arguments (**attributes**).
- The streams are themselves typed and can be conformance type checked.
- Frames are transmitted by producer via non-blocking writes and read by consumers via blocking reads. This is analogous to writing frames to a buffer; and application reads from the buffer.

Streams form basic units of multimedia communication. Definition of media streams and their applications to communication are discussed in Chapter 4.

2.4 Applications of Group Models

The group models described so far have been variously applied in communications systems. Auerbach *et al* [Auerbach 91] outlines some of the group models which hitherto have found some applications.

In Cheriton and Zwaenepoel ([Cheriton 85]) the management of process groups is employed in the V-kernel. In this system, the information about the host group (i.e. the set of hosts on which the group membership reside) is encoded in the group. Each member has knowledge of the membership in the group and can thus receive a packet and pass it to all local members. The notion of host groups is also exploited in IP-multicasting [Deering 89]. This approach to group communication exploits both peer-to-peer and group models.

In Navaratnam *et al* ([Nanaratnam 88]) different group communication mechanisms are built on top of V-kernel. The approach considers primary managers as well as secondary managers to manage group membership. However, the primary and secondary managers maintain only local membership information, similar to [Cheriton 85]. For membership interaction, it is not clear how a source, which is not a part of the group, can find out about the group membership or communicate with a group.

As for the ISIS group model [Birman 89] each group member maintains a complete group membership. This method is useful in fault-tolerance applications. However, it is argued [Auerbach 91] that many applications do not require this level of global agreement and can do so without the associated cost of providing this service.

ISIS [Birman 89] and ANSAware [Herbert 94] toolkits are designed to aid application developers to build distributed systems. ISIS provides reliable distributed computing based on process groups and implemented as a set of library calls on top of UNIX. In [Oskiewicz 1990] the ANSA document describes an Interface Group as a means by which programming structure of decentralised objects is realised. Groups are based on the exploitation of multi-end-points communications: namely the ability to invoke operations on a collection of objects without needing to know the exact membership of the collection or location of the members. This capability provides the basis for distributing implementation of a service over a set of objects and provides location transparency. ANSA groups are closed.

Thus ISIS and ANSA groups use both group and distributed models for process communications.

In a description of modelling groups for group communications, [Wilde 94] describes the work at Swiss Federal Institute of Technology which presents a model of Group Management Services (GMS) designed to support collaborative interactions among groups of distributed users using different applications, i.e. a service for the support of information sharing. GMS enables applications to share application-independent information (such as group membership and administrative information), and applications can also store application-dependent information which can only be interpreted by a closed set of applications (i.e. those who know the syntax and semantics of application-dependent information). A directory service mechanism is a major component of this model. However, there is no reference to explicit group management services for multimedia applications, although the model suggests a distributed functionality.

Other models concerning group communications management have mainly revolved around transport and network services e.g. Henckel [Henckel 93], [Simon 94], [Pasquale 92], [Pendergast 93], and [Twachtmann 93], [Mathy 94b]. And, just like the aforementioned models, [Auerbach 91]'s model is basically on data transfer mechanisms for the transport user that wishes to employ multicast services. However, it does not spell out how external users and agents that become involved in group communications in, say, a user co-operative task or a conferencing environment can be represented. Nevertheless, if we adhere to the philosophy behind OSI/RM, applications are too removed and insulated by the distributed system from the concepts and mechanisms of these transport issues. In this respect [Wilde 96] presents a model for transport -independent group communications platforms.

2.5 Work on Group Communication and Multimedia Applications

It is generally felt that the mechanisms and methods employed for present day group communication are not suitable for multimedia group applications. For example, the provision of location transparency is not suitable for co-operative activity which requires that members of the group be aware of each other. Also virtual synchrony (as employed in ISIS group) is undesirable for real time applications requirements. A virtual synchronous system "looks synchronous" to every process in the system, but executes asynchronously when observed from outside [Isis 92]. It is with such short comings, and many other issues concerning the traditional group applications that new ways are being sought to devise mechanisms for multimedia group communications.

The problem of multimedia group communications has been addressed from different viewpoints by different research groups and organisations. The context in which the research is carried out in this area differs from project to project, depending on the overall project goals and objectives of the organisation. For example, in the Internet community, multicast and group communication is addressed with a strong bias towards communication and protocol aspects. A number of protocols providing functions and mechanisms for group communications have been proposed e.g. MBone, RSVP, RTP, RTCP [Erickson 94], [Schulzrinne 95], [Zhang 94]. Other groups' research activities have been concerned with group communication taxonomy and terminology and are suggesting different schemes and models in this context [Mathy 94a]. At the moment, most of the projects currently concerned with multicast/group communication have their own special goals e.g. the current GCommS[1] project at Lancaster University. The way in which problems are addressed must therefore be seen in the context of their work and

---

[1] Group Communication Services for Distributed Multimedia Applications

research done by these groups.

In this section some of the research activities that have been or are concerned with multimedia group communications issues are presented. These projects show different approaches for multimedia and group communication. They are only a few examples of recent work in this area, therefore this discussion can only give an impression of the work currently being carried out in this area.

2.5.1 The EuroBridge Project

The EuroBridge Project was a meta project of the RACE II programme, initiated with the aim of compiling the communications requirements of various RACE projects [Heinrichs 93]. The project's activities have subsequently led to the definition of the EuroBridge Service Platform which is: "The service communication platform will be developed on the SUN and IBM - compatible Unix Workstations and to support the OSI development environment (OSIDE) implementation of the full OSI stack as well as the TCP/IP protocols". The project was established with three aims:

- to enhance existing application services for multimedia.
- to enhance implementation and functionality of the underlying layers to support adequately the application layer requirements.
- to provide the user with integrated, true multimedia services, comprising of data, video/audio and including conventional services.

In essence, the EuroBridge project was meant to investigate the required multimedia services to support the transmission of multimedia information as required by applications. It was also meant to formulate a uniform communication service platform which would be highly flexible and adaptive, and capable of dealing with a broad variety of quality of service requirements dictated by the applications. Hence a full OSI protocol stack would need to be exploited.

The EuroBridge project thus provided a pointer for the need to provide flexible and adaptive environment for group based multimedia applications. Thus any architecture developed would be required to provide generic services. Hence multimedia group management services would need to conform to this requirement to meet with the flexibility required.

2.5.2 Multimedia Integrated Conferencing for European Researchers (MICE)

MICE was a project between various partners from European countries. Its aim was to provide a service to enable internetworking between European researchers located in a heterogeneous environment via multimedia conferencing technology (including audio, video and a shared workspace) [Sasse 94]. During the project a multi-way integrated conferencing service between a number of European pilot sites and a link to appropriate communities in the US was developed [Handley 93]. The MICE technology allows conferencing between participants located in conference rooms which can accommodate up to ten people at workstations.

The proposed services are multi-way shared workspace, multi-way video, multi-party voice and multiplexing and conference management. Different shared workspace systems are supported by different partners. The goal is to support a wide range of applications suited for different communication domains. For compressed video the international H.261 standard was adopted [Kirstein 93]. Audio and video data are treated independently. The main reasons for this approach are the fact that audio is much more sensitive to delay and jitter than video, and that compressed audio (especially voice) requires much less bandwidth than video. It is therefore better suited to be multicast to a large number of participants.

An important element in the MICE architecture is the Conference Multiplexing and Management Centre (CMMC). Its main function is to bridge the heterogeneity gap, i.e. to allow communication between users at different networks with different facilities. Further, it provides a relay between different coding standards, multiplexes audio and video; reserves, controls and manages resources, and ensures multicast of shared workspace data [Handley 93], [Kirstein 93]. Video originating at a remote site codec is decompressed by hardware or software after eventual pre-processing at the CMMC. The CMMC provides support for receivers connected over ISDN or to low bandwidth connections. A conference control system based on the CAR conference control system (CAD/CAM for the Automotive industry under RACE) [Handley 92], offers functions to create, delete, join and leave a conference, floor control and control of conference video channels.

Conference participants are currently connected through three international networks, viz. EBONE (Pan European Multi Protocol Backbone), EuropaNet and Internet. Communication over ISDN is also supported. Links are upgraded but no special communication facilities have been specifically installed for MICE.

Practical experience with MICE has been gathered during various conferences and a series of seminars for researchers and students. In particular, [Almeroth 95] *et al* have developed a measurement tool for characterising the dynamic behaviour of MBone sessions. It is hoped that the measurement obtained will be of great benefit in understanding how future networking infrastructure with multicast and real-time capabilities will handle multiple MBone sessions effectively. Seminars were multicast using MBone. The greatest problem during the seminars has been poor audio quality caused by packet loss in the Internet. Video transmission was also affected but it proved that low video quality is less disturbing in a seminar situation. A large overhead for the set-up of seminars was reported [Sasse 94].

Mbone is gradually emerging as the IP multicasting backbone. However, the drawback in its utilisation is that not all routers implement IP multicasting, and it also requires that multicast addresses be reserved. Furthermore, the flooding mechanism used by the IP multicasting are not suitable for less densely populated networks.

2.5.3 The MERCI Project

This is a continuation of MICE, funded in the European Union's Telematics programme [EEC 95]. It is aimed at research and development activities for multimedia data traffic in support of multimedia conferencing systems. The proposal builds on the earlier work, especially on the tools that have been developed in EuroBridge and MICE, with regard to Multimedia Conference Control, and multicast facilities. Various standards are involved regarding the coding techniques for audio and video data streams and it is hoped that the users of the Multimedia Conferencing Systems do not have to concern themselves with the heterogeneity of the technologies. Design of the user interface is of some considerable concern in this project. On the whole, the purpose of the MERCI project is to pursue the development of the tools so that they are available to other multimedia applications.

The research community is the first target group for experimenting with this project. This is because the research community can readily access network facilities over work-stations and PCs and therefore only require upgrading of the software/hardware already available at different locations. Synchronisation of various media types, particularly audio and video media streams, and the recording of sections/whole conference for later playout is another concern of the research activity.

2.5.4 The GCommS Project

The ***GCommS project*** at Lancaster University addresses mainly end-to-end aspects of multipeer communication [Hutchison 93]. The research is divided into three closely related areas, viz. end-to-end services and protocols, multipeer QoS, and scalability and reliability in multicast communication. The major components of the GCommS architecture are a multicast service (*M-Connection Service*), a multipeer service (*N2N-Connection Service*), and a communication manager (*GDC-Manager*) to provide virtual multipeer connection [Rezende 95b]. The M-Connection Service is fully specified and has been mapped onto XTP 4.0 protocol mechanism to prove its feasibility. A number of tests and simulations have been carried out to gain experience with the service and to find the most suitable protocol architecture to base it on. Protocols currently being evaluated for this and the two other services are XTP 4.0 and an IP based protocol architecture using UDP, IPv6, RSVP and possibly RTP/ RTCP.

The main issue of the GCommS work on multipeer QoS is the support of QoS in a heterogeneous group environment. A multipeer QoS model to support receivers with different quality requirements and capabilities has been specified. In addition, filtering mechanisms have been implemented to provide different data qualities for different receivers on a multicast stream. This allows to provide each receiver the quality it can handle while still utilising the advantages of multicast. Algorithms to provide data transfer reliability in multicast communication have also been developed. Different degrees of reliability ranging from full reliability to probabilistic reliability can be supported by these algorithms. The research on communication and reliability issues is carried out in close co-operation with researchers at the Laboratoire MASI, Université Pierre et Marie Curie, Paris. The project has also links to the European COST 237 project concerned with multimedia telecommunication services and other European research institutions.

The GcommS project formed a basis for this research in group management services as described in [Mauthe 94]. The objective of the initial study analyse the various group mechanisms and functions that had been used for group communication.

2.5.5 Standards Issues In Group Communication

The ITU (International Telecommunications Union) is one of the standardisation bodies presently looking at Group Communication Standardisation, another being ISO (International Organisation for Standardisation). The activities of these bodies as far as group communication is concerned are summarised in [Mauthe 95b]. It is noted in [Benford 92] that "Group Communication was adopted as a new work item for messaging standardisation (within ISO/IEC JTC1 /SC18/WG4) in 1989." At the time, two approaches towards developing services and protocols to support group communication were under consideration:

(1) To make extensions to existing services such as MHS and then produce a specific standard for a particular application, which would be quicker; however, this was seen as restrictive and would overload new applications with backward compatibility problems.

(2) To build generic Group Communication services on top of which many specific applications could be standardised. This was seen as a more flexible approach, but would take too long to produce a standard.

Against this background, it was imperative to harmonise the different views on efforts towards standardising the Group Communication Services. The notion of a generic Group Communication Services (GCS) was largely adopted and seen as " a new OSI service which would provide common facilities to a range of Group Communication applications". These would include:

- a shared information base
- a global name space for the group communication objects
- co-ordination of underlying OSI services such as MHS and Directory Service.

The relationship between the GCS, applications and the underlying services is shown in Figure 2.8 [Benford 92].

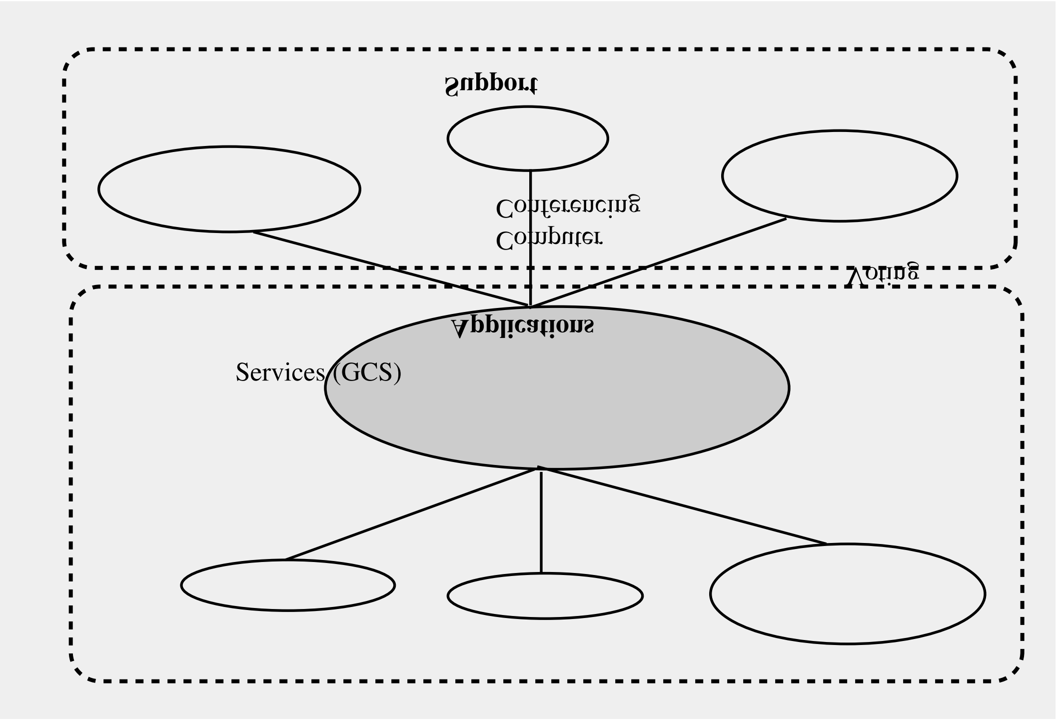


Figure 2.9: Relationships between GCS, applications & underlying support

Subsequent consideration on how GCS should be configured resulted in splitting the work into two task forces:

(1) The Group Communication Service - This would specify the Group Communication Service information Model and Abstract Service Definition and show how the GCS may be refined onto a combination of MHS, Directory Service, and other OSI Services

(2) Asynchronous Computer Conferencing -This would specify the

Abstract Service Definition of Computer Conferencing and would provide a mapping onto GCS.

The work of a latter task force is described in [Benford 92]. However, it is not yet clear that the standardisation of Asynchronous Computer Conferencing has been brought into effect; nor have there been attempts to standardise Synchronous Computer Conferencing with facilities for Interactive Real-time Applications. The activities that reflect on the former case have been taken up by the working committees on multipeer communications in [Cocquet 95a] i.e. Multipeer Data Transmission Services (MPDT), as follows:

2.5.5.1 Multipeer Data Transmission (MPDT)

The Multipeer Data Transmission communication was one of the Components of Enhanced Communications Functions and Facilities (ECFF) that sought to enhance the lower OSI layers (layers 1 - 4) for data communications [Campbell 93]. The underlying philosophy about the Multipeer Data Transmission (MPDT) is that it should be possible to send data from many senders to many receivers in a single action. This has the advantage of reducing the network load when messages are destined for more than one recipient.

It was observed that in order to progress MPDT services and protocols, a consistent architecture needed to be defined. In this respect, the work initiated by [ECFF 92] was later revisited by Mathy at University of Liege [Mathy 94b] who aimed to extend some of the concepts of Transport Group Associations which entailed Group Conversations and which had been cited in the document on MPDT [ISO/IST/6 92]. The current status is that activities on MPDT are now being considered under ISO ECTS and ITU-T X-TMS projects [Cocquet 95b], with the objective of formulating a Multipeer Taxonomy for standardisation.

2.5.5.2 Summary on Standardisation

From the foregoing discussion on standardisation activities, it is evident that a standard or set of standards remains to be produced for group communication. Views were solicited from the Multicast Taxonomy forum (osimcast electronic mail forum) debating issues raised in the documents such as the "Draft Taxonomy of Multicast operation" [ISO/IEC N8693 93], the "Group communication Framework" [Mathy 94a], and the "Comments on Document WG4/N 833 (4K64R1) and proposed new Draft Taxonomy" [AFNOR 94]. It is from such discussions that the current [Cocquet 95a] document on Draft Multi-peer Taxonomy was generated. Given all these standardisation activities, the bottom line is that there are still no standards for group communication services, and any new experiences that come out of research in group communication continue to be invaluable pieces of input to the standards bodies.

2.6 Discussion

So far a number of issues relating to group communications and applications have been presented. While we agree that there have been a number of contributions towards the support of group communications, there is still a lack of support for multimedia group communications including the provision of group management in distributed computing environments. It was reiterated in parts of this chapter that multimedia group communication involves many participants, both users and external agents, each of which category of entities plays a given role in a group. The problem that arises in this case is the difficulty of transmitting the same information to a number of participants who may have very different individual requirements, and the availability of information processing agents.

When referring to multimedia group communication, we take into account the large amount of multimedia information (a combination of text, audio, video, graphic or still images) that is to be transmitted to multiple recipients. A typical scenario might involve a number of *clients* that wish to collaborate on some given task and thus constitute a group. Associated with this group are application processes which may handle video, audio or text/graphics media types. If a user (a client) now wishes to send a piece of information to other members within the group, he may do so by using one or more of the media types. For example he may use the word processor application to send text/graphics data, or multiple media for a multimedia document, as well as for real-time multimedia information. The group may be made up of a number of clients with differing capabilities and different roles. In order for the group to operate as a cohesive unit, group activities need to be co-ordinated by group management services. For example, when a group of people use a multi-user application with the same rights, each member should have the same set of capabilities of accessing its data [Dollimore 92]. The group management facility should ensure that this is achieved.

So far the conventional models hitherto applied to group communications are not suitable for describing group communication in distributed multimedia applications for the following reasons.

1. The interaction patterns of group communications in the conventional models were meant to provide fault-tolerance, reliability, and location transparency. This requires that groups be *closed*. On the other hand, applications engaging in co-operative tasks require that groups be open. Moreover, transmission of real-time data does not require the reliable mechanisms provided by these models as this leads to overheads in synchronisation process.

2. There are no explicit mechanisms for dynamically creating groups with characteristic features, or the ability to destroy groups.
3. There are no mechanisms for adding potential members of varying capabilities, different roles, different rights etc. to groups, i.e. the ability to create groups with differing policies and membership rules.
4. Some of the group models presented provide local membership lists. There is a need to provide global information about groups and provide a service for querying this information.
5. The dynamic management of groups and group members is desired in co-operative applications.

The group communication models which have so far been developed have been targeted for specific applications. When these models are used for applications such as CSCW, they force the latter to manage all the factors relating to group communications in an attempt to realise a co-operative system. For example, [Garcia 93] observed that in the current multimedia environments, the developers of group applications are forced explicitly to manage group membership, group characteristics, connection establishment, dynamic conference control and synchronisation. The short-coming of this approach is that in the development of *groupware*, each application develops its own method of supporting collaboration, thus incurring heavy development costs and reducing compatibilities among different systems. As stated in [Trevor 95], this is painfully slow and a problematic task: ideally, the CSCW applications developers should be free to concentrate on the semantics of the application and what should be provided, rather than being concerned with group communications services. If a class of application requires a certain functionality or quality of service, the lower layers should provide it [Verissimo 92]. Apart from freeing the user from the burden of providing these services, such lower layers would have been optimised and thoroughly tested. But in the absence of this support, the collaborative applications become unable to collaborate with themselves [Wilde 94].

Therefore, to eliminate the problems raised by CSCW, it becomes necessary to provide some form of *generic group* communication platform, which is flexible enough to allow full collaboration among groups and applications [Daiz 93]. Thus instead of each CSCW application employing its own semantics and policy, there should be a *suite* of general and tailorable mechanisms which support commonly *used* concepts, and therefore helps in the re-use of these general mechanisms instead of *re-inventing* the wheel. This would allow for the sharing of collaborative activities among different applications because they would share a common group support platform [Hartley 95]. But current developments in CSCW indicate that the typical support provided by the operating system is inadequate for successful operation of collaborative applications [Rodden 92]. The operating system approach for communications does not model groups well enough to provide an efficient service.

Some of the areas of support that CSCW applications have identified are:

- the provision of general group mechanisms to support commonly used concepts of group features and operations at lower layers;
- flexibility in applying group mechanisms.

As noted before, a few of today's distributed system platforms and operating systems support group communication. However, since most of these systems have been designed to support specific tasks such as distributed file systems and databases, fault tolerant systems, etc., the group framework and the services they provide often do not serve all the needs of multipeer applications. For example, in [Herrtwich 92], [Muller 92], it is stated that, from a computing perspective, multimedia is seen as a means of communication between humans and computers - and between humans using computers as communications tools. Therefore, the functions and group structure of distributed system toolkits and operating systems are in general too simple, inflexible and/or specific to support high level multimedia group communication.

2.7 Summary

This chapter has outlined some of the activities involved in group communication in order to bring into focus the state-of-the-art equating to group communications in distributed multimedia applications. Group formation and communication, their constituent properties and characteristics have been reviewed. The purpose of introducing groups into communication is to be able to manage a collection of entities as a single set of communicating objects. These objects (human users, processes, possessors, etc.) are designed to perform specific tasks to achieve a desired goal using services provided by group support systems.

Because different objects or entities are involved in group communication, it is important to observe the specific *roles* of group members. The *nature* of the group (i.e. static or dynamic) and the *mode* of communication (asynchronous or synchronous) are further attributes of group communication. These features are common constituent properties of any group communication environment, which have been reviewed in the chapter.

It has been also noted that group concepts have been employed variously in computing environments especially in distributed operating system and network management, for which group mechanisms have been used to provide services for high level group applications. However, while the modelling of group communications using these mechanisms (e.g. operating system-, ISIS-, ANSA-groups) have provided invaluable utilities for communication, the mechanisms fall short of supporting multimedia applications in distributed computing environments

Since multimedia communications essentially involves heterogeneous environments and different end-systems, standards are being formulated to avoid incompatibilities. The standardisation work on multimedia group communication is currently an active area of research, as evidenced by production of the T120 standard and the work on multipeer taxonomy.

2.8 Conclusion

Not all issues related to multimedia group communication in distributed applications have been investigated or discussed in this chapter, but some of the proposals in this area look promising. However, while the infrastructures to support multimedia group communications have been investigated in major projects such as MICE [Handley 93], and MBone [Erickson 94], [Almeroth 95], among others, further work has to be done to define sufficient functions and services for multimedia group communication.

It should be noted that the general group management services are not yet available for multimedia applications. Group management is the area of interest for this research. The concerns for this area are the location of different management services, their functions and mechanisms and how they can be applied to support multimedia group communications in a distributed computing environment. Why do multimedia applications make the management of groups different, and hence require different support strategies? Subsequent chapters examine this issue by addressing the question of *how to provide flexible multimedia group management services*, starting with the requirements for group multimedia applications in the next chapter.

Chapter 3

Requirements for Group Management for Distributed Multimedia Applications

In today's computer systems, information can be represented through audio and/or video, in addition to text, image, graphics and animation. The integration of these multimedia types enhances the use of computational power to support group interactive applications. Because the data for these media types can be transmitted through computer and telecommunication networks, there is a growing need to support applications for multimedia distribution and co-operative work [Steinmetz 95].

Multimedia applications are being used to support users performing tasks ranging from computer-aided learning, remote conferencing, to co-operative design. The storage, transmission and presentation of audio and video sequences derived from recordings, on-line video cameras, computer-synthesised animation and simulations play a prominent role in multimedia applications. In a computerised environment, many multimedia applications use the underlying distributed systems support to enable several users to interact with each other or to share common resources [Coulouris 90]. This mode of communication contrasts with a multimedia environment for a distribution service such as TV. As stated by [Steinmetz 92], [Pasquale 92], in the distribution services, the server station sends audio and video information to a group of remote listening clients. Each viewer can decide whether to receive the distributed information or not, but may not influence the sender. Other forms of dissemination services include radio broadcasts, newspapers and bulletin boards.

On the other hand, computer-mediated multimedia applications are meant to allow clients to interact with the sources of information. Services to effect this mode of communication are being sought in a variety of application domains including research groups, business and industry, education, medicine, and entertainment. This provides an opportunity for multimedia group communication, i.e. the use of various media types to support users of multimedia applications to perform co-operative and/or distributed tasks.

However, this process requires management services.

So far the arguments for management services are well known in the area of computer networks and distributed systems applications as described in [Coffield 87], [Wang 89], [Sloman 94], [Langsford 94]. In fact these arguments have been well exploited in communications services to manage communication systems such as nodes or host groups in the networks [Deering 89], [Ngoh 91], and to effect connection establishment with associated QoS i.e. the resource allocation methods that can be tailored to meet the specific requirements of the applications [Coulouris 90]. For example [Garcia 96] presents a QoS model where *flow management* is used to establish, maintain and release continuous data flow along the various network paths (for point-to-point as well as point-to-multi-point services). When a client joins a point-to-multipoint service by registering with some group management entity, the latter invokes flow establishment by contacting the appropriate resource management module at a particular source end-system. Flow establishment takes place between the source and the client through the aid of a handshake scheme which employs a reservation protocol similar to RSVP [Zhang 93], and ST-II [Topolic 90].

In a distributed systems environment the management services are meant to provide policies and mechanisms to achieve intra- and inter-site communications among co-operating processes [Fortier 88]. In essence this requires a communications manager whose varied functions would include the registration of processes within the communications systems, opening and closing of logical paths from processes to other processes (e.g. one-to-one, one-to-many, one-to-all) and manipulation of these paths in a dynamic fashion. Paths must be allocated and de-allocated to share the resource effectively. Additionally the communications manager may be required, in a real-time network, to provide guarantees for messages requiring real-time service.

In principle, management is seen as a means by which to plan and organise resources, control the use of resources, to ensure that the resources remain accessible, and to report the unavailability of resources [Wang 89]. For a distributed computing system this is achieved by *processing* and *communication management* [Coffield 87], [Smith 89]. Tools such as ISIS have been used to support communication management in distributed systems by providing facilities for fault-tolerance, reliability, and other functionalities. However, multimedia group communication places different requirements on communication systems that need different management strategies, which cannot be met by the conventional approach to management in a distributed computing environment. In fact in the literature [Mauthe 94] and [Cocquet 95a] have noted that there are no general group management services for distributed multimedia applications; most of the work such as that described by [Gupta 93], [Gupta 94], [Szyperski 93], [Szyperski 93 b], have focused on the design and development of real-time communication services and network support for continuous media applications. However, the extent to which network and communication services utilisation will impact on multimedia group applications will depend on the development of suitable management services.

Therefore this chapter aims to investigate the requirements of multimedia group management services, their functions and mechanisms in distributed computing environments, and to derive services that will isolate communication management from applications specific details. It is considered that this approach simplifies the programming abstractions, allowing easy access to communications facilities such as multicast support, and the location of multimedia resources. The chapter is structured as follows. *Section 3.1* is an outline of multimedia group communications and their characteristics. In *Section 3.2* multimedia group applications are presented with the view to deriving multimedia group management requirements, which are discussed in *Section 3.3*. The multimedia group management issues to be investigated are set out in *Section 3.4,* which is the conclusion of the chapter: the important aspects for management services of multimedia group communication are outlined.

3.1 Multimedia Group Communications

Multimedia group applications are some of the key applications that will exploit audio, video and text/graphics and other media types. However, this use of variety of media types for communications, especially audio and video, requires a large amount of bandwidth; in these circumstances multimedia communications will naturally exploit the emerging high-speed network technologies. The Integrated Services Digital Networks (ISDN) and Broadband-ISDN have been eagerly studied and technical decisions to use Asynchronous Transfer Mode (ATM) switching technology over Broadband-ISDN have been made. Since there is no doubt that high-speed network technologies will provide the necessary bandwidth for multimedia communications, the current group communication issues are focusing on group members' interactions and the number of participants that the communication system can control simultaneously before the group's size becomes unmanageable. In this section a review of the characteristics of multimedia group communications is presented.

3.1.1 Characteristics of Multimedia group communications

Computer oriented communication applications should be able to support two or more geographically separated persons or groups of people who interact using various media types, i.e. video, audio, whiteboard (for shared applications), inter alia. Such communication is classified as Multiparty-Interactive Multimedia applications (MIM) [Szyperski 93 b], [Ventre 93], [Moran 94], and is characterised by the functional aspects of the mode of information exchanged at the transport-level subsystem, i.e. point-to-multipoint (including point-to-point). This support is required for multimedia group applications in the form of one-to-many and many-to-one interactions enabling communications among multiple users to exchange information over computer networks supported by distributed media types. The features of multiparty (multipeer) communication are described in the following sections.

3.1.1.1 Mode of Interaction

This refers to the way participants interact by exchanging data depending on the ability to produce and consume multimedia data. Traditionally, three different types of interactions may be identified as follows.

Dynamic:

This allows any participant to send and/or receive data without any form of restriction. However, it is assumed that at a higher-level of group communication such as meetings, social protocols will prevail. The assumptions in this case are that:

1) Out of many participants, usually only one participant will talk at a time in a conferencing application.

2) In computer mediated co-operative applications, a lot of the time participants will concentrate on image or data applications, and so no one will talk will for lengthy periods.

Static:

This relates to the case where participants that are allowed to send/or receive data are a statically a determined set that will not change through the application's lifetime. For example in an opera, the orchestra produces the music while the audience only receives it.

Controlled:

In this case, the subset of participants allowed to send information can change during the course of an application, but the change must be governed by conditions or rules of the application. For example in a seminar, or a conference, the role of a speaker may be assumed, in principle, by any of the participants while in a

lecture course the instructor may occasionally allow students to
ask questions.

3.1.1.2 Data flow

This relates to the way information flows among the participants. It may take the form of one-to-many data flow, or many-to-many types of flow. An example of the first case is the conduct of a seminar where the speaker and the attendees are expected to intervene, one at a time, while the second case is illustrated by meetings in which multiple communications can simultaneously coexist. It is observed that in other models of interpersonal communications, such as panels and debates, an $M \rightarrow N$ data flow can exist, where usually $M < N$.

3.1.1.3 Accessibility

Sometimes the access to data flow is restricted, and this requires that participants undergo some form of admission control or monitoring process. Access control may take two forms: *controlled* and *uncontrolled*.

In the controlled access model, a participant must be enrolled to take part in group communication. For example, in a lecture course, a student must be enrolled in that class. In uncontrolled access, no kind of control is enforced on the participants, for example in public hearings since everyone is usually encouraged to attend.

There are various methods of enforcing access control. A naive approach is to use lists which, for instance, may restrict participants at a conferences. A list may be prepared ahead of time (static) or it may be dynamic, i.e. modified during the event. In the latter case, for instance a conference may be announced ahead of time and participants register, while others may join only at the last minute. There is a difference in the way the two groups may be treated in that for the first group, participants are known and are therefore guaranteed to be accepted, while the acceptance of the latter group will depend on the availability of resources.

3.2 Multimedia Applications

3.2.1 Media Characteristics

So far "multimedia applications" has been a notional treatment as a "means" of conveying information in a computing environment without considering the constituent properties of the multimedia. If the support of multimedia type group communication is to be sought, it is imperative that the nature of multimedia applications is understood.

The development of multimedia applications has to take into account functional characteristics of a given media type. As noted in [Muller 92], [Herrtwich 92], although multimedia has become a buzzword, and the recognition that future systems should provide multimedia functionality, there *is* some uncertainty about what a multimedia system actually is and what its particular function should be. In [Shepherd 95] multimedia is defined as "many media" where a medium is a means of transporting information or a particular way to communicate. From a computing point of view, it is a means of the interaction between humans and computers, or between humans via information technology.

One major characteristic that determines a true multimedia system is *media flexibility.* i.e. a multimedia system should be able to combine both *discrete* and *continuous* media types. In this respect, equipment such as a TV set, VCR, desktop publishing systems handling text and graphics, are not multimedia systems. On the other hand, an editor with voice annotation is a multimedia system. A further requirement on a true *multimedia system* is that there should be a provision for handling each medium independently, hence providing the opportunity to combine them in an arbitrary way. For example, a system such as a VCR stores video and audio together in which case each media stream cannot be accessed separately, whereas if both audio and video were stored separately say on different disk files, manipulation of individual media types would be possible. Computer systems are the ideal tools for realising media flexibility. Therefore a computer-mediated multimedia applications system is characterised by the computer-controlled generation, manipulation, presentation, storage, and communication of independent discrete and continuous media [Herrtwich 92].

When different media types are used to enhance human-computer communication, and to effect communication between humans, a multimedia group communication is realised. A given group session may use a variety of media types, depending on the *application domain* (discussed in section *3.3.2*), and will therefore employ different multimedia services. Two main categories of service are normally perceived for multimedia group communication applications, viz *dissemination* and *interactive* services. In the following sections, the characteristics of these services are described, with the intention of deriving the management requirements that may be applied for distributed multimedia group communication.

3.2.2 Dissemination Services

Multimedia dissemination services are akin to TV/radio broadcast services although the activity takes place at a computer terminal or a workstation. Such services include Video on Demand (VoD), tele-shopping and archive databases. In this mode of communication, the source provides the information to bona fide members registered to the service, but the recipient cannot edit the information received. In [Pasquale 92], [Kompella 92] these are referred to as large scale dissemination services. Communication in this manner requires very little group management as the members of the group are not aware of each other. It is a case of an "unknown group" in the terminology of group communication taxonomy; only the service provider (the group manager or owner) knows the group membership list to which the information is distributed and interacts with individual members, and vice versa. Group members may join in or opt out of a "group session" e.g. in a given VoD session. The group manager should effect the sending of the data when a member joins the group session or to stop the sending of the data when the member opts out of the session.

3.2.3 Interactive Services

The second set of multimedia group communication services, i.e. interactive services, is where group applications interact with the system or with each other supported by computers and communication systems [Mauthe 94]. Although in some cases there are no major differences between dissemination and interactive services (e.g. in receiver controlled VoD and tele-shopping), the main distinctive feature between the two is the degree of interaction which takes place within a group (or sub-group). The main purpose of the interactive applications is the communication among group members, whilst in dissemination services, the main purpose is the distribution of information.

The largest category among interactive applications are the CSCW applications. The term CSCW characterises the co-operative work of a group of two or more users supported by a computer based system [Hagsand 91]. CSCW applications support group work using the developments of computer hardware, networking and systems support. These applications cover most of the characteristics and requirements of other multimedia group applications. Their highly interactive nature and their high quality of service requirements are the reason why CSCW applications are regarded as those multimedia group applications with the highest demands. A detailed discussion of CSCW systems in general is out of the scope of this thesis, but more information on CSCW systems can be found in [Grudin 92], [Turner 92] and [Trevor 95].

Apart from the dissemination and interactive services, there are other examples of multimedia applications. Some of these are described in [Williams 94] include office automation, service industry applications (education - distance learning, finance, and health), retail applications (publishing, travel, and property), domestic applications, science and engineering, and cultural applications.

3.2.4 Applications Scenarios

It is not a straightforward task to identify the requirements of complex applications such as CSCW because the existing systems are based on today's platforms, and are adapted to their specific environment. Investigations of their requirements reflects to a high degree the capabilities of the systems they are built on. To look only at such applications is not sufficient. Therefore, in order to provide the required functionality of group management services, there is a need to consider and analyse group application scenarios to get a better idea of the requirements of future multimedia group applications. Two typical applications scenarios are presented: *Group Applications in Manufacturing and Business,* and *Collaborative Computing*

3.2.4.1 A Group Application in Manufacturing and Business

Imagine the act of co-operating car manufacturers. The construction of the engine is done in Germany, the body is designed in Japan, the car is manufactured in Wales and the marketing office is located in London. During the development of a new model, engineers, experts and managers from different sides have to "meet" frequently. The meetings might be planned ahead or be spontaneously convened when problems are encountered. The meetings are held as audio-video conferences with tools for shared CAD, 3-D models, shared document editing and video from storage servers. Different participants have different rights towards the shared objects. Engineers, for instance, would be allowed to change drawings during a meeting whereas it would not be very sensible to allow a marketing manager to make any technical changes. Since the participants are experts in different areas their view on the shared data might be different. An engineer is probably not very interested in the colour and design of the steering wheel. His main interest is its technical functionality, whereas for a marketing expert the colour might be essential to attract the right target group. The quality requirements of the exchanged data varies. At the viewing of CAD drawings a complete and consistent version has to be edited at the screen of every participant whereas the right pixel colour for an advertisement might not be considered so important. However, when it comes to a certain point in the discussion, for example the shape of the mudguard, an expert might want to have a more detailed view of the 3-D model and therefore would like to upgrade the quality of the output. Audio and video have to be of conference quality, and audio especially is very sensitive to delay. Group management has to be provided to set up a group, initiate a meeting, invite group members and other participants. Access control and security mechanisms are of vital importance in this area. During meetings multimedia documents may be exchanged between different members of the group.

This scenario has the following characteristics:

1) participants are distributed over a wide area network and probably connected through various heterogeneous networks;

2) there is a wide variety of media types employed, data views and quality requirements of different users;

3) participants have different access rights;

4) the quality of the continuous media data should be adjustable to the QoS required by different participants;

5) the audio conference mode is used; the conference is unregulated, i.e. everybody is free to speak at any time;

6) dynamic media scaling is desirable for an application where many different participants with different expertise are involved;

7) plans, models, documents and videos can be stored at different places. Concurrent operations on them have to be ordered in a way that keeps multiple copies consistent;

9) if meetings are planned ahead it should be possible to reserve all necessary resources beforehand;

10) membership of the group is known or determined;

11) the group is closed but dynamic since some participants may opt out of the discussion if not interested in certain aspects of the conference or rejoin the conference;

12) this is a short-term group, i.e. it is set up only for a session.

3.2.4.2 Collaborative Computing

Collaborative Computing is characterised by a group of users working together on some given tasks through computer mediated applications. In a typical collaborating computing application, otherwise known as Computer Supported Co-operative Work (CSCW), many tools are employed. These include electronic mail, bulletin boards, screen sharing tools, text-based conferencing systems, telephone conferencing systems, conference rooms, and video conferencing.

There are currently a number of systems supporting multimedia applications for group communication. However, most of the design activities have been concentrated in developing tools for shared applications, and services for connection management. In a distributed system, applications are normally removed from the issues of the transport services. For Group Communication, applications should be concerned with generation, manipulation and control of multimedia information, without the concerns for communications of independent discrete and continuous media types. In [Williams 94] it is stated that lack of system infrastructure has further inhibited the development of distributed multimedia applications. The majority of applications have been developed directly on top of a hardware base with little help being provided in the management of resources across the network. This implies that most of the applications are tightly bound to proprietary hardware environments, making it difficult to port to other configurations. Hence standards need to be developed to cope with the heterogeneity in the infrastructures.

A collaborative computing application was realised in a scientific application in a research project between Lancaster University and ICI Chemicals & Polymers. In this case, the potential of collaborative computing using distributed multimedia applications in a scientific environment was investigated in [Williams 93], [Williams 94]. It was observed that scientific applications have much tighter restrictions on information quality compared to commercial management or educational applications. A decrease of media quality during the transmission or playout often makes the information useless for any further examination. During the study a prototype of a microscope application was developed . The following scenario is based on this example.

A group of biologists examines a cell sample under a microscope. Like conventional cameras, microscopes in a fully automated system are remotely controllable. Only participants who are declared as “sample owners” are allowed to control the microscope. High quality video, X-ray spectra and high resolution pictures are transmitted from the microscope to remotely sited experts examining the sample. Experts involved in a sample examination may have differing expertise on specific parts of the sample. Therefore different views on the sample by these experts are required. Whereas one expert needs to see all movements of the sample (e.g. vesicular breathing), another is interested in the cell structure and therefore needs a high resolution detail of the picture[1]. The experts discuss the viewed sample over an audio connection. Only the sample owners and experts who are consulted are allowed to take part in the discussion: passive participants such as students are just allowed to listen to the discussion and to see the video output (or parts of it). Depending on hardware and software capabilities and personal needs, participants may have individual quality requirements. The set of participants is dynamic; participants can join and leave the group session. The video sequences, X-ray spectra and every other output of the microscope session may be recorded for documentation and later examination. The microscope output can be incorporated in multimedia documents created by multiple authors using shared editing tools. Figure 3.1 shows a typical layout of this scenario.

---

[1] Note, they are still viewing the same sample but each of them has a different view of it and hence has different requirements towards the output.

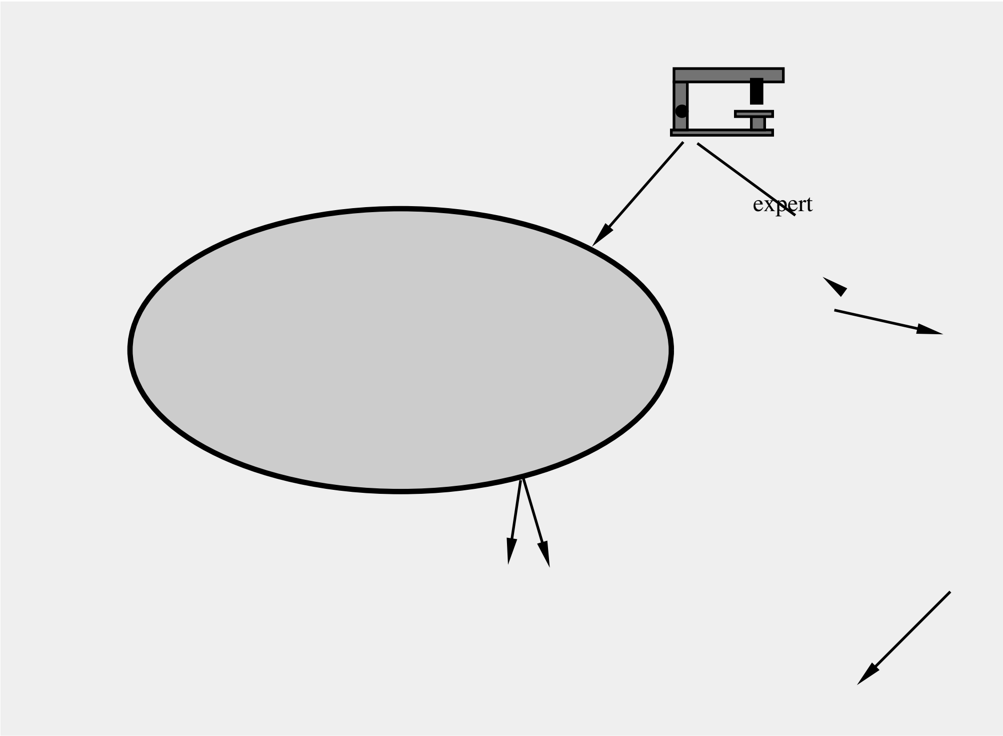

expert

Figure 3.1: Example Collaborative Configuration

Major characteristics of this scenario are:

1) high quality and reliability requirements, especially for video data;
2) for non-continuous media (e. g. X-ray spectra) a fast, reliable and error free data transmission is also required but this service is not time-critical; any transmission errors can be corrected through re-transmission;
3) audio quality can be adjusted to the available QoS;
4) the conference mode is not controlled and experts can be located at the same site or at different sites;
5) participants have different rights; only sample owners are allowed to manipulate the sample and to operate the microscope, therefore control commands in real time have to be transmitted; concurrent commands by different sample owners have to be ordered at the device;
6) various different connections with different topologies are required; video has to be transmitted from the microscope to all participants (1 : N); audio has to be transmitted between some

experts and all other participants (M : N); control commands flow between sample owners and microscope, etc.; all these connections have different requirements towards bandwidth, reliability and timing;

7) a group manager to administer the group and its participants is required.

The above issues show how group communication using distributed multimedia applications can be characterised, and require support from application to communications sub-systems. This thesis considers the management services requirements, which are discussed in the following section.

3.3 Multimedia Group Management Requirements

Having considered the general group communication issues in *Chapter 2*, the structuring of computer-based group applications and the nature of interaction among group participants in the previous sections, the requirements for multimedia group management can now be stated.

It was asserted in *Section 2.1.1.3* that for any group communication or computer aided collaborative activity, a number of categories of service are required, e.g. the data transfer services, the synchronisation services and management services. For example, in the scenarios discussed, to provide services for multiple participants, using a variety of media types and with varied modes of communications, it may be required that the underlying communications system has addressing mechanisms to support 1:1, 1:N, N:1 and M:N for data transfer and control information. It is also clear from the scenarios that synchronisation services for the different media types employed are required. The basic functionality of management services is to control access to the media types, control of the number of participants using the media types, and setting policies for using the media types. Other management functionalities are described in *Section 3.3.2*. On management services, we refer to media types involved in communication, and the methods the participants use to interact with the media types.

From the above application scenarios it can be inferred that the requirements of multimedia group applications are manifold. The consideration of media types involved in communication aid in structuring multimedia group communication, resulting in the provision of group management services for multimedia applications. In [Williams 94] two separate aspects of multimedia communication are identified, namely the *forms of media* available for group interaction, and the ability to communicate efficiently using the media types, i.e. the extent to which multimedia applications for group communications can be supported is dependent, to a greater degree, on the ability of the underlying system to support the communication, namely the data transfer sub-systems and the synchronisation services. These issues are discussed in [Mauthe 95b], [Namuye 95b].

Since distributed multimedia group applications involve use of a variety of media types and their integration on a number of workstations interconnected by one or more networks, the majority of future co-operative applications will be located in this global environment and will require sophisticated tools to support collaboration and communication between users scattered over a wide geographical area. Multimedia group management will be an important component in supporting group applications. For example, in the scenarios discussed, applications involved in conference communications will need the support for conference initiation and termination, the dynamic behaviour of conference participants, and the conference control.

The conference communication facilities have been investigated by designing and using tools based on the support provided by such systems as Multicast Backbone (MBone) [Almeroth 95], ISDN[Horn 1990], and Rapport [Ahuja 92], [Conti 92]. These multimedia conferencing systems enable people to hold real-time discussions in which they share voice, video and whiteboard applications [Almeroth 95]. At the time of writing, MBone currently is in use as an experimental virtual network that provides the means for IP multicasting for data to any number of connected hosts. One of the primary aims of MBone is to investigate the real-time one-to-many delivery of audio and video conferencing data across the Internet. Hence a lot of tools have been developed and are being used extensively to support multicast conferences, talks, radio broadcasts, concerts, and group working meetings.

ISDN and Broadband-ISDN have also been studied extensively [Muller 92] and have shown promise in supporting multimedia group interactive applications. From the user point of view these would provide guaranteed service contracts with certain quality of service; from the technology point of view, the use of Asynchronous Transfer Mode (ATM) as switching fabric promises to guarantee these services.

Assuming the underlying communications systems provide one-to-many and many-to-one support to effect many-to-many group interactions, multimedia group management will be required to map these interactions in a distributed multimedia resource environment. This mapping is important from the applications' point of view since at the level of data transfer, the multimedia resources have no notion of groups; they are merely resources that group members can query and/or use. In the following sections, an outline of the requirements for managing the use of a variety of media types to support group communication is presented.

3.3.1 General Requirements

In general, group communication is concerned with allowing participants of the group to interact and establish rules for that interaction. On the other hand, group management is concerned with defining and controlling the group objectives [Verissimo 92]. For multimedia group communication, the situation is essentially intensive in that there are many more players involved using different media types and requiring that media types be located, shared, and their use controlled. The group-management entity will be required to co-ordinate all the operations on media, providing the group user agents with the information about the media types, such as their location, and the policies imposed on their use. When a user agent requests the use of a media type, the management entity sees to it that the transport service system arranges for the appropriate QoS, the list of recipient or sender processes, as required by the application. For example, while a media of type *text* may not place stringent conditions on transport services (apart from reliability), other media such as *audio* may indicate a quality of voice (CD, stereo, mono, etc.) or mode. Others such as *image/graphic* media may require presentation ranging from mono, greyscale to colour, while media such as *video* may require playout modes ranging from mono slow scan, greyscale slow scan to full colour scan. These are parameters that need to be passed to the communications sub-systems for appropriate media communication, apart from the timely delivery and minimal jitter delay for continuous media types. The client agent should indicate to the multimedia group applications management entity the required level of performance for these parameters to the underlying communication subsystem.

3.3.2 Multimedia Management Requirements

Apart from the above requirements, the management entity should provide services to carry out the following functions on given media:

Membership Administration

> There should be support for group members to join or leave given media, as well as an indication of their nature of participation i.e. *only senders*, *only receivers*, or both *senders and receivers*. During a group session, mechanisms should be provided dynamically to exclude or include participation in a media for group members who either leave or wish to join a group session. This ensures efficient use of both media and network resources.

Security

> Both authentication and access rights should also be performed when a user agent requests use given media. Therefore the

management entity will need to have the access control list which specify the users for which it is allowed to initiate the service.

Connection Transparency

Multimedia group communication will depend on the underlying service to support multiparty data streams in which given a medium with one source should be able to send to multiple destinations. However, it is important that the source is able to transmit such a stream without being concerned about a possibly dynamic set of participants currently receiving the stream. This responsibility should be the concern of the multimedia group management system. This further simplifies the programming abstractions, allowing easy access to communications facilities such as multicast support. It is the responsibility of the multimedia group management system to arrange for the locations of connections to which the source should transmit to the destinations.

- *Location of Media*

  Since multimedia resources will exist in a distributed system, information about their location and the type is required. A form of directory service for querying this information will be required in this case.

- *Control*

  In the case of shared media such as VoD a large scale dissemination service, there should be a provision for applying control functions such as fast-forward, fast-reverse, pause, etc. without disrupting other viewers. In the case of a collaborative environment, the shared control of the VoD should be reflected on all other group members' views.

Inclusion/exclusion of media

There should be provision for facilities to include media when required, and to exclude the media when not required during a group session. This dynamic and flexible environment makes efficient use of network resources.

Scalability

As a consequence of dynamic support for join/leave actions during a group session by group members, it should be possible to scale the medium of communication to the current group members, i.e. a source should be able to send to multiple receivers.

Use of multiple media

Since some group applications such as video conferencing will make use of multiple media types, the group session manager should ensure that each group participant receives the mandatory number of media types.

In summary, the multimedia group management services need to provide the functionalities required to support the above requirements, i.e. the means of enabling many procucers of multimedia to send to many consumers, control participation in a given media type during group communication, and the fact that participants will be having different rights towards media types.

3.4 Conclusion and the way forward

In this chapter an investigation of group management service requirements for distributed multimedia applications has been made. While a number of proposals have been advanced for group management services, most have concentrated on services supporting information sharing or connection management i.e. applications and transport services.

It has also been observed that there are problems in collaborative services arising out of connection management, and group membership management for a given multimedia application. These two factors have resulted in forcing each application to develop its own management strategies for collaboration. This thesis suggests that for multimedia group management services, there should be mechanisms that map group multimedia interactions in a distributed environment in a transparent way. To meet these requirements, there should be a suite of services (or service elements) that will:

1) instantiate, manage and control various media types involved in group communication;
2) isolate communication management from applications specific details, allowing easy access to the underlying communications sub-system;
3) be generic to media types; this will provide the flexibility in programming multimedia group applications in a media independent way.

Thus the overall objective of multimedia group management services is to provide *facilities* for dynamically instantiating and managing the media types involved in group communication. In order to provide such functionalities, the management system should be modelled as a suite of services that:

- effects the instantiation of media types to multiple users;
- allows for independent manipulation of a media type in a multimedia applications;
- uses low-level media type support to realise high-level group sessions in a collaborative environment and allow the formation of sub-groups or sub-sessions

The service placement is shown in Figure 3.2 for a distributed environment.

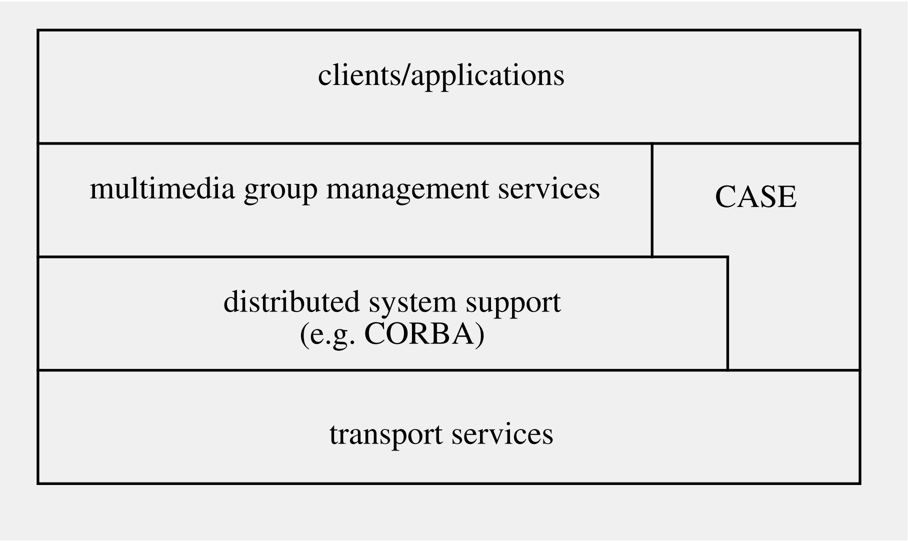


Figure 3.2. Layered architecture for multimedia group management services

For applications, it is important to be able to pick any service without being forced to use all other service as well. Some users might just need a basic data transmission service, and therefore use the Common Applications Service Elements (CASE), whereas others require a whole range of services. Figure 3.2 depicts a possible service configuration in the context of the application platform.

The next chapter discusses the modelling and design of the envisaged services for multimedia group communication management.

Chapter 4

Design of Multimedia Group Management Services

There have been a number of activities on modelling and designing Group Support Systems e.g. [Chen 1991], [Vogels 92], [Pendergast 95], in the realm of CSCW work, while proposals for the requirements of High Speed Networks to support multimedia applications have been advanced e.g. [Waters 92], [Kompella 92], [Pasquale 92], [Mauthe 95b], emphasising the need for multicast support for sending data from a source to a group of receivers, so that applications such as video conferencing, distance learning, distributed panel discussions, and Video on Demand, can be supported.

It was mentioned in *Chapter 2* that while group management services have played an important part in distributed computing environments, the management strategies employed are not suitable for managing distributed multimedia group applications. This situation has naturally led to the search for new methods of managing multimedia applications for group-oriented activities.

In the previous chapter, it was observed that most of the tools developed to support collaborative activities tend to be very application specific and restrictive. This is because each application develops its own methods for management and communications services. The argument against this approach was that group collaborative activities cannot be suitably supported in a distributed multimedia environment. Hence a suggested proposition in the thesis, arising from the requirements specified to alleviate the problem of applications' specific management and communications issues, advocates for the provision of support that separates applications-oriented activities from communication issues. While the transport services would be responsible for multimedia data transfer and its integrity, management services would be concerned with configuring the transport services for group communication.

In order to fulfil these requirements, it was proposed that both users (and programmers) need to be provided with a suite of services. It should also be possible for applications to define new parameters and appropriate functions. In this respect a generic service architecture would be most suitable.

The use of multimedia in a given application by different users (actual people through their user agents) necessitates that a "media channel" be established for the medium (the concept of a media channel is defined later in the chapter). In multimedia group communications, participants should be able to receive from different media sources, which, however, should be kept consistent with the whole group state, and synchronised if they are related media types. However, each media channel should be managed separately. In other words a group session would be simply a realisation of a set consisting of users/members and the media channel involved in communication. Thus the management of use of media types and the mapping of the applications requirements to the use of them are fundamental to multimedia group communications.

In order to provide a suite of generic services, the multimedia group management service architecture should take into consideration a number of *service elements*[1] as well as the standard CASE tools that are conventionally available to the applications.

In essence, the motivating factors for the new approach to Multimedia Group Management Services (MGMS) is the management and control of the individual media types by providing mechanisms that facilitate the higher-level applications to realise group-based activities in a distributed multimedia computing environment. Therefore this thesis explores the development of a service architecture for MGMS that is intended to offer generic system level support for this class of applications. Furthermore, the use of a given service requires certain policies. As stated in [Davies 94], [Langsford 94], policies are means of management to guide the actions of the "manager" about the explicit behaviour of the system. Although policies may be different in forms, a suitable MGMS model should be one that provides supporting mechanisms that will effect the same management services for different media types, but leaving the applications free to tailor their own policies for the desired services to meet their requirements. This is a further pointer to the need for providing generic services.

As was shown in Chapter 3 it is appropriate to place the MGMS between the application layer and distributed systems support so as to insulate multimedia group applications from transport issues. Therefore one consideration to take into account is the configuration of the mapping functions by the group management entity of the application's requirements to underlying transport services. (For instance there should be some indication stating a certain quality of service, i.e. the resource allocation methods

---

[1] Services made available by the service-provider are a collection of several individual services called *service elements*; and associated with each service element there are procedures (or functions) implemented by the service provider [Jain 90].

that can be stated to meet specific application(s) requirements using a given media channel).

With the above overview of the requirements for multimedia group management services, the aim of this chapter is to present a design for MGMS that provide an architecture for flexible multimedia group management services. This chapter investigates methods that are flexible in providing generic (base) services for both users and application developers. The strategy employed is one that examines the concept of a *media channel* as the management unit for an instance of a single media type, and hence multimedia group communication management at this level is a matter of managing the media channel units.

The chapter is structured as follows: *Section 4.1* analyses the conceived multimedia group management services, their characteristics and operations and subsequently states the needed services to fulfil these requirements identified in *Chapter 3*. In *Section 4.2* a design for the MGMS presented, describing the service elements, their primitives and operations. *Section 4.3* describes some of the work that has been done on multimedia group communications. The aim of this section is to compare the design approaches and the functionalities of the systems derived therein, with the view presented in the thesis. A discussion of the suitability of the MGMS model is presented in *Section 4.4*. The conclusion in *Section 4.5* summarises the important aspects of the MGMS architecture and points to the direction of implementation of MGMS as presented in the model.

4.1 Analysis of Requirements

This section describes the nature of the requirements and subsequently states the services needed to fulfil the requirements. As outlined in the previous chapter, the MGMS should fulfil the following:

- membership administration,
- support for multimedia sessions,
- scalability of a given medium of communication to multiple users and groups,
- inclusion/exclusion of instances of media types,
- provision of control of instances of media types,
- security,
- provision of selected information on connection requirements,
- admission control.

The objective of this analysis is to identify relationships among these requirements that will aid in modelling the service elements for MGMS. As Booch states, [Booch 1994], the conventional approach to analysis is to focus on the behaviour of objects of interest when subjected to certain operations. The objects considered in this case are the media channels (for example, audio, video, text media types) while the operations eliciting their behaviour are entailed in the use of the media channels.

4.1.1 Multimedia Group management Service Components

From the list of requirements three broad categories of service components may be conceived i.e.:

- client/application services
- media types administration services
- media types information services

The three components are shown in Figure 4.1; they represent the building blocks for MGMS; their characteristic features are listed in Table 4.1.

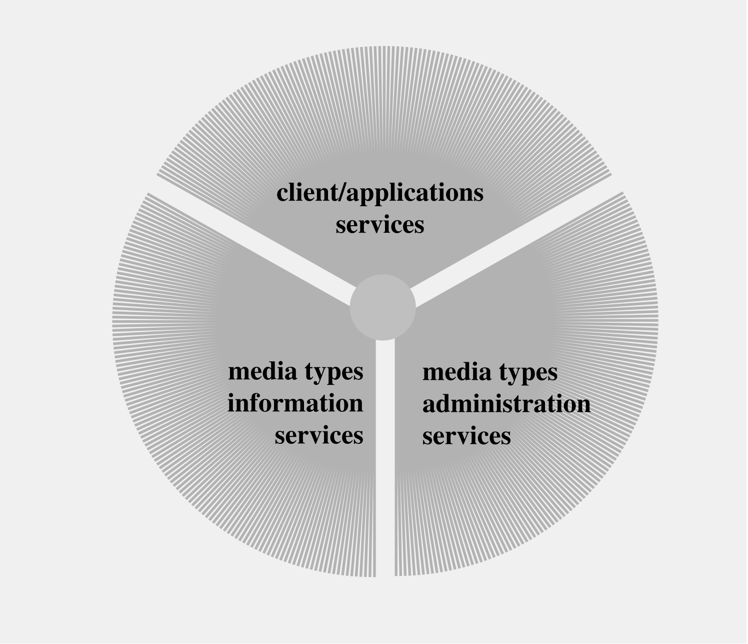


Figure 4.1: MGMS Components

MGMS COMPONENT	SERVICE SUPPORT	SERVICE CHARACTERISTICS	SERVICE OBJECT
client/applications	GUI	- instantiation of media sources and sinks - join/leave media channel - multimedia group support - control of use of media channels - directory service queries	client
media types administration services	communication manager policies	- control of media channel - security/authentication - media channel policies	media channel manager
media types information services	- list of media channels - location of media channels	- directory services	client

Table 4.1: MGMS Characteristic Features

The interactions of the above components can be modelled as shown in *Figure 4.2*. These interactions form the service components of MGMS; they are discussed in the following sections.

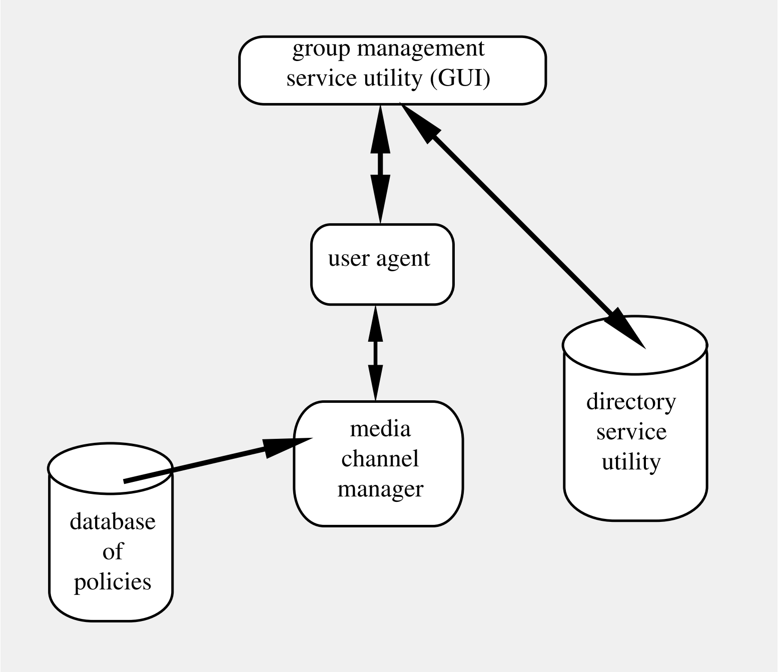


Figure 4.2: MGMS Component Architecture

4.1.2 Client/Applications Component

Clients or application processes run on the end systems and are responsible for maintaining the use of media channels by invoking the management operations on the MGMS User Agent (UA), which may be supported by a Graphical/Textual User

Interface (GUI). The UA functionality involves the following operations.

- querying information on given media channels; this may require a directory service facility for global information;
- joining a media channel or a list of media channels that the client/user is interested in (assuming the user subscribes to the channels or may be allowed by some as a guest);
- leaving media channels that the user is currently participating in;
- creating a new media channel and inviting others to join in;
- having a privileged user to allow or disallow clients the participation in the media channel; such a user may be a service provider or some higher level management entity;
- contact participants by phone call or e-mail;
- add media channels to individuals' list of media channel (similar to Netscape Bookmarks).

This component may also be configured to list several media channels which users may potentially participate in and thus form a group session. In other words:

> *a multimedia group session consists of potentially several media types such as audio/video conferencing, shared document editing, shared viewing of Video-on-Demand applications, which a group of people may participate in.*

The multimedia group management services (MGMS) considered in this thesis are those that manage individual media channels whose sources transmit media streams in multimedia applications as illustrated in Figure 4.3.

The overall design philosophy of MGMS at the media channel level is to provide a management strategy which is simpler for the application, but which hides the "real" architecture. All multimedia applications can then have the same control (e.g. connection management) interface. In this way each media type can be controlled/managed independent of the application.

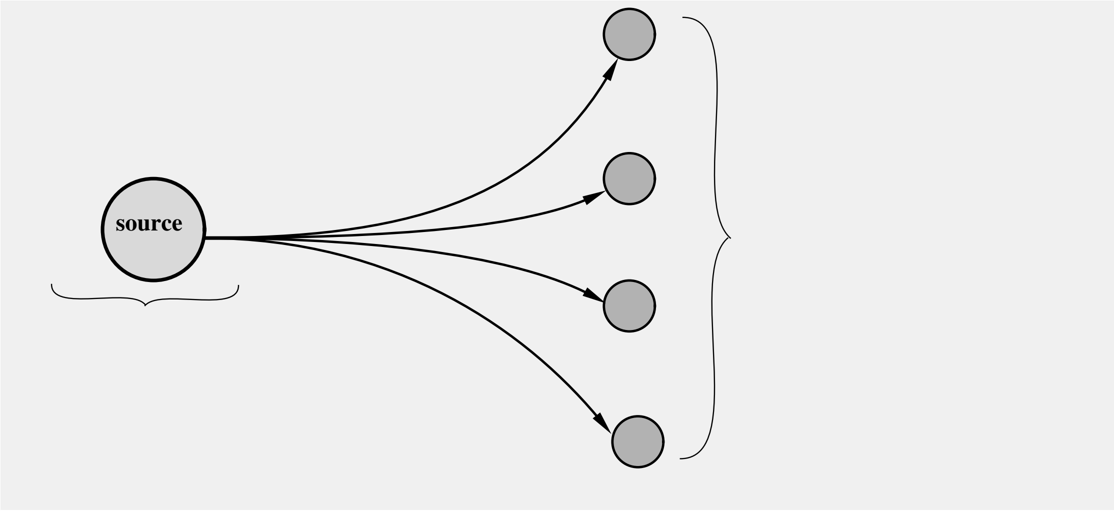

source

Figure 4.3: Agents of human group members participating in a media channelwith one source

4.2 Design of Multimedia Group Management Services

The multimedia group management service model is designed around the concept of a *media channel*, as the basic unit of management in a group *user-agent*, which therefore manages multiple (multimedia) applications. A media channel in turn, is built using *streams*, which will be described first.

4.2.1 Streams

A stream may be defined as:

> *"an object of communication between a producer and multiple consumers, distributed across a network."*

The producer produces a sequence of data (the stream) in a defined format, and this is multicast to each of the consumers, who are expecting the data to be of that format. Each consumer may, independently of the others, ask to receive only a subset of the data by issuing Quality of Service (QoS) requests in the opposite direction to the stream, so that intermediate nodes in the network may filter out the unwanted data. As noted by [Fluckiger 95]:

> *Quality of service is a concept based on the statement that not all the applications need the same performance from the network over which they run. Thus applications may indicate their specific requirements to the network, before they actually start transmitting information data.*

The stream may also stop and start, but always according to rules that the consumers are aware of. This is illustrated in Figure 4.4.

4.2.1.1 Media type

A media type defines the data representation between the producer and the consumers of the data stream, i.e.

> *a* ***media type*** *represents the format of the data and the state changes* (**Δ-state**), *travelling between producer and consumer, and the QoS requests from consumer to(wards) producer of the stream.*

A producer produces one stream of a particular media type, while a consumer consumes one stream of a particular media type. The media type of a producer or a consumer is the media type of the stream which it produces or consumes.

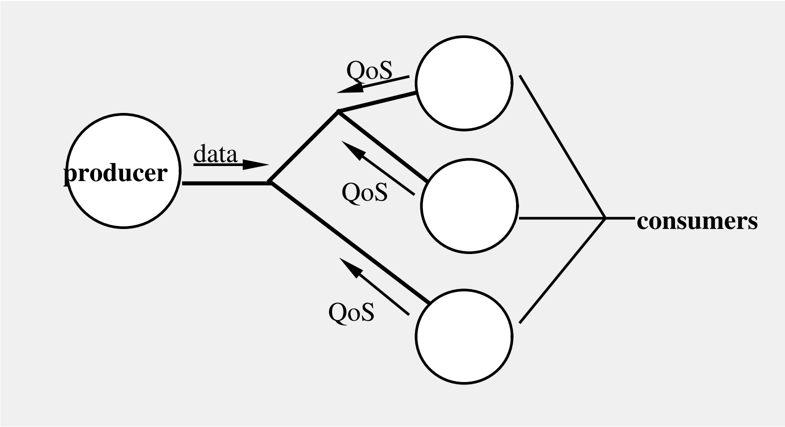


Figure 4.4: Stream: producer/consumer configuration

4.2.1.2 Example media types

4.2.1.2.1 MPEG

MPEG, i.e. Motion Picture Expert Group, is a coding data standard developed by ISO/IEC JTC1/SC 29/WG11 to cover motion video as well as audio coding according to the ISO/IEC standardisation process [Steinmetz 95]. Besides the specification of video and audio coding, the MPEG standard provides a system definition, which specifies the combination of several data streams. For example, the Δ-state messages can be **MPEG** frames of various quantization (resolution) levels or enhancement layers. QoS requests comprise indications of the required layers (4 values for **MPEG-2**). Producers of this type generate MPEG frames on-the-fly, either by encoding them from a live video source, or by parsing an MPEG file from a backing store.

4.2.1.2.2 SunAudio

The Δ-state messages are 8-bit μ-law-encoded packets with a real-time header. There are no QoS requests (or rather, there is only one QoS level defined). Producers sample audio from a microphone, or obtain samples from a file, while consumers play out the payload of the Δ-state messages to a speaker, or append to a file.

In general, a stream S consists of a media type T, a producer P, and a set of consumers C, such that:

$$\mathbf{S = < T, P, C>}.$$

4.2.2 Media Channels, Sources and Sinks

A media channel may defined as:

> "*a set of producers (which we call sources), a set of consumers (sinks) and a media type. Each source produces one stream of the channel's media type, while each sink consumes multiple streams - one per source. Therefore, each source is multicasting its stream to each sink.*"

The media channel's state is the aggregate of its sources' states, plus the number of sources. Each sink maintains a state which is composed of the (partial) copies of each of the sources, plus the number of sources. All sources registered with the channel must produce state and change-in-state messages that conform to the syntax of the media type of the channel. Similarly, the sinks must be able properly to interpret those messages.

The main requirement of a media channel is that all the sources must send to all the sinks, and conversely, all the sinks must receive from all the sources. We use a *media channel manager* (MCM) to maintain a register of sources and sinks, in a single channel, and this is responsible for ensuring that all (and only) the appropriate streams are established. Sources and sinks connect to the manager to indicate their participation in a channel.

The main aim of the media channel is for each sink to have a mirror of all the sources' portions of the channel's state. For each individual sink, the mirror of any of the sources' state does not have to be perfect: the sink can instruct the source to give it only enough information for a lower-quality mirror, independently of the other sources and sinks - it is able to have the messages filtered.

4.2.2.1 Examples of Classes of media channels

4.2.2.1.1 *Video Conferencing (VC)*

In a video conferencing application, the state of the media channel is shared amongst all contributing participants, which will usually be a subset of all the registered participants. The observing participants run sink programs for, say media type H.261, and register them with the video conference media channel, whose media type is also H.261. The contributing participants each run a source program and register it.

A contributor's portion of the state is the image that his desktop camera is producing. This image is (potentially) constantly changing, so his source program will be continuously streaming out data which is multicast to all sinks.

A sink will receive multiple video streams, one from each source. They are distinguishable, and so the sink can display them in separate windows. If the user's machine is overloaded by all the input, he may instruct the sink to lower the quality of all its windows, and will not affect other participants' displays.

As well as an image's resolution, frame rate and colour, the media type may allow some frames to be marked as important. Sources which have contributors who are currently talking can mark their frames as important, a sink which is being overloaded can filter out less important frames. To the user, only the participants who are talking will appear - others will disappear or freeze.

4.2.2.1.2 *Video-on-Demand (VoD)*

This scenario involves one source which produces video streams, while users configure their sinks to select the qualities that their machines are capable of, and that they are happy with.

A single user can not control, i.e. pause, rewind, playback, etc., as this is part of the state, and changes to it must affect everyone. *If someone wants to do this independently of other viewers, the media channel model expects that person to instantiate a new media channel for the same video.*

If users wish to share control of the playback speed (or which video is being used, or which frame is to be displayed, which are also part of the state) a second media channel (the control channel) should be setup as illustrated in Figure 4.5. It should have only one sink, which is able to control the source program of the other channel. Users who wish to have control should activate programs registered as sources to the new channel, through which they communicate with the source of the video channel. The state that is represented by the new channel is "*how we are* ***all*** *going to watch this video*". The control sink knows which people want the video and at which speeds.

4.2.2.2 Management Operations

Media channels are used by the group session members for the management of sources and sinks included in the media channels in order to participate in them (media channels). However, the use of media channels in a group session imposes certain restrictions on the channel that determine what can be a source or a sink within the media channel. For example a group session defines "VoD data" as being "focused producer " type; only one source is allowed and not (necessarily) associated with any group member. Group members may participate in this channel as consumers, so they may only provide sinks for this channel.

Some media channels may be "focused consumer" (e.g. "VoD control"), i.e. members participate only as producers, and an external object (not a member of the group) as the only consumer. The focused producer and consumer MCs are shown in Figure 4.6.

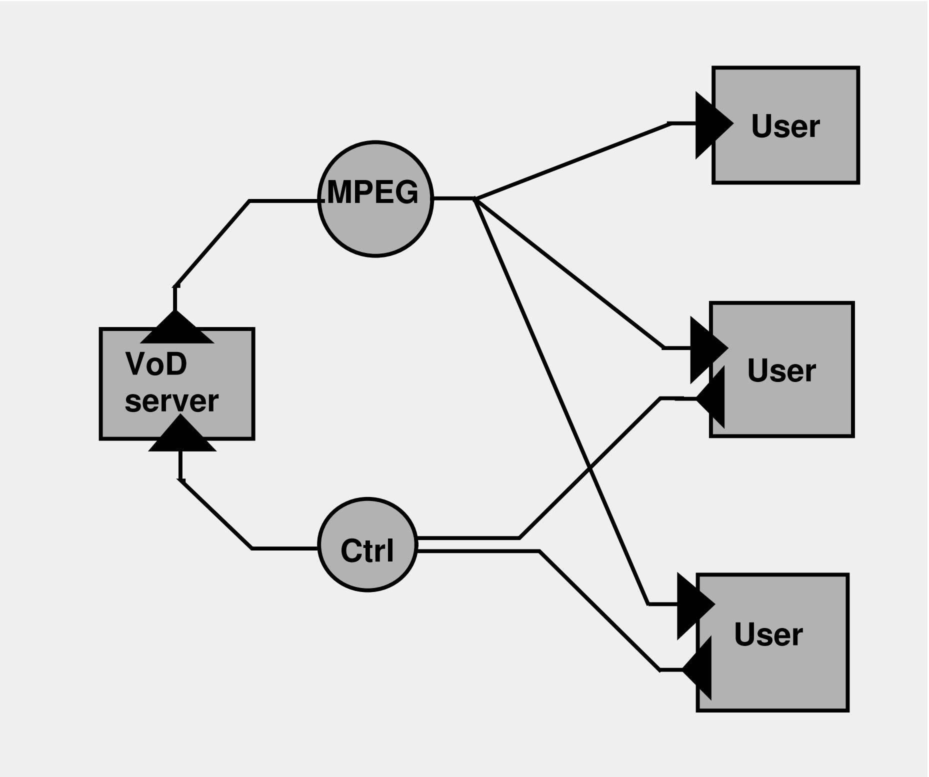


Figure 4.5: VoD Configuration

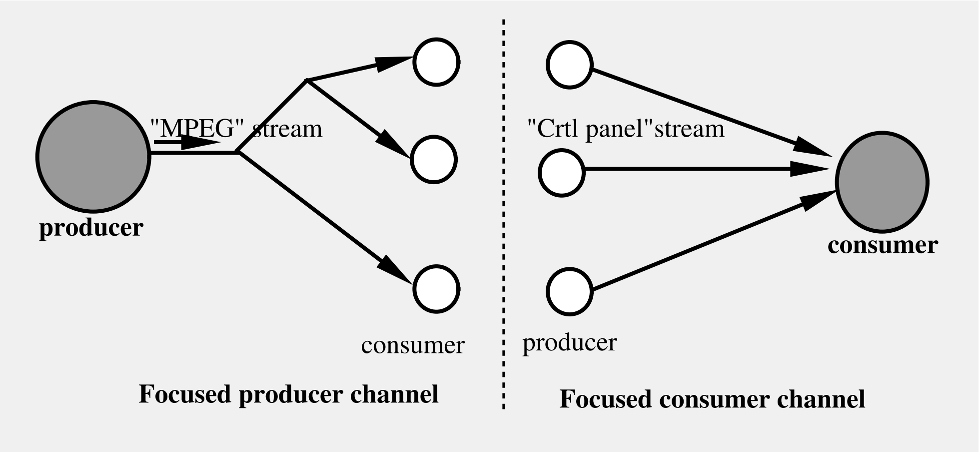


Figure 4.6: Producer/Consumer configuration

During a group session each member participates in each channel to a certain degree. For example, most of the members will be producers and consumers of "H.261" streams within a VC channel, while they will be just consumers of "MPEG" streams within the "VoD data" channel. However, some may control the "VoD data" channel, being producers of "control panel" streams ("VoD Ctrl" channel), i.e. start/stop/rewind commands.

4.2.2.2.1 *User Agent*

Each member has an agent (the User Agent) to maintain the state of a member i.e. manage participation in media channels. It includes a database of media channels that the user knows about, and may wish to participate in. It is also a representation of the user, including his preferences, the capabilities of his terminal, and his involvement in channels.

The user agent is aware of multiple channels such as Media Channel 1 (MC-1), Media Channel 2 (MC-2) and Media Channel 3 (MC-3) in Figure 4.7), and how to run a source or a sink for each channel. The list of media channels is user-configurable. The user may indicate to the agent that he wishes to participate in a channel as a source or sink, and the agent will identify the type of the channel, and run the appropriate program automatically (Figure 4.7). When the user wishes to terminate his participation, the agent simply destroys the relevant program. In effect, the user agent presents a service that allows the user to:

- retain a reference to a channel that the user may later wish to participate in; discard such a reference;
- participate in a channel as a source; cancel such participation;
- participate in a channel as a sink; cancel such participation.

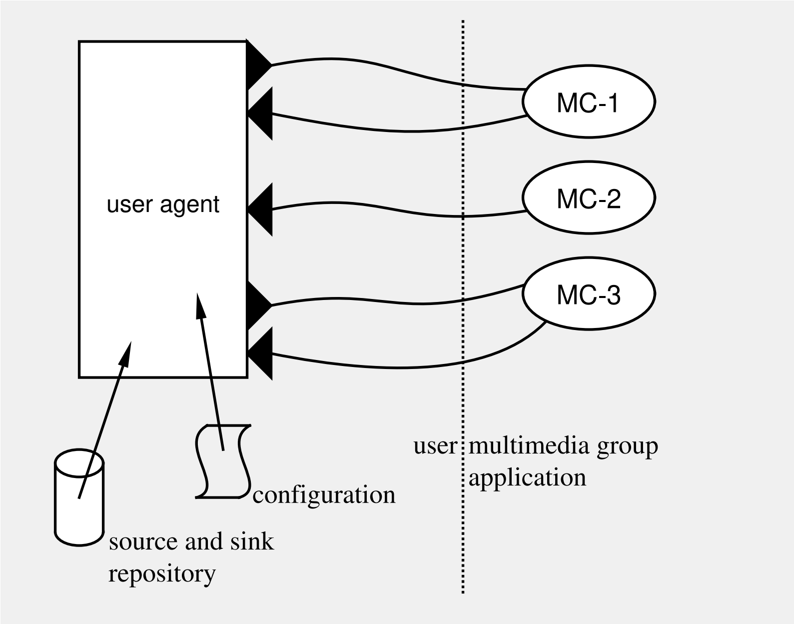
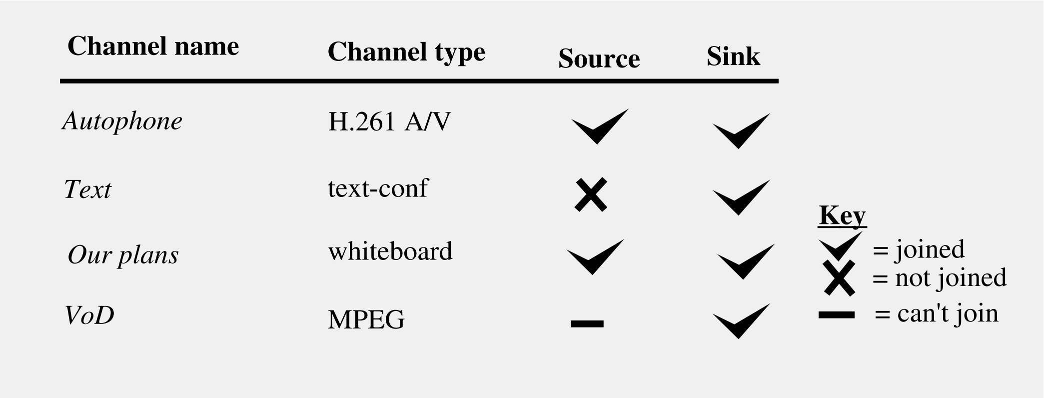


Figure 4.7: User agent configuration

A typical display of a user agent is shown in Figure 4.8 which lists the channels, giving their names and media types, and indicating the user's participation in each one. It may also include other per-channel information such as how many are participating, and who they are. Or it may provide this information on request.



Channel name	Channel type	Source	Sink
Autophone	H.261 A/V	✓	✓
Text	text-conf	✗	✓
Our plans	whiteboard	✓	✓
VoD	MPEG	—	✓

Key
✓ = joined
✗ = not joined
— = can't join

Figure 4.8: User agent display

Finally, sources and sinks can refer identification requests to the agent that spawned them, enabling other agents to establish who is behind a source or sink.

4.2.2.3 Creating and managing media channels

Creating a media channel involves running or activating a media channel manager (MCM), and specifying its media type. Sources and sinks can then be registered with it at will. Figure 4.9 shows a design of a VoD service with VoD *data* and *control* channels and their associated elements. The interactions between the MCM consists of Remote

Procedure Call (RPC) messages at *signal interfaces*, while the interaction between sources and sinks consists of streams at stream interfaces.

Figure 4.10 shows the interaction between the MCM and the client of the VoD service.

The state of a media channel is represented by a management object or the channel manager. In Figure 4.10 a VoD application for a *join* request by m1 (group member) is shown. m1 issues a *join* request and also submits its sink address to the MCM. MCM requests the sink for a sink address (MPEG player), which is returned to the MCM that the MPEG source will use to transmit the VoD data from VoD server. If m1 also wants to join the VoD *control*, a similar process is followed, except that m1 now joins the VoD *control* as a source.

In practice, a multimedia group communication would have topologies such as the ones depicted in Figure 4.11. In this case two media channels are involved: a shared document channel (a focused application) and a video conferencing channel (unfocused application).

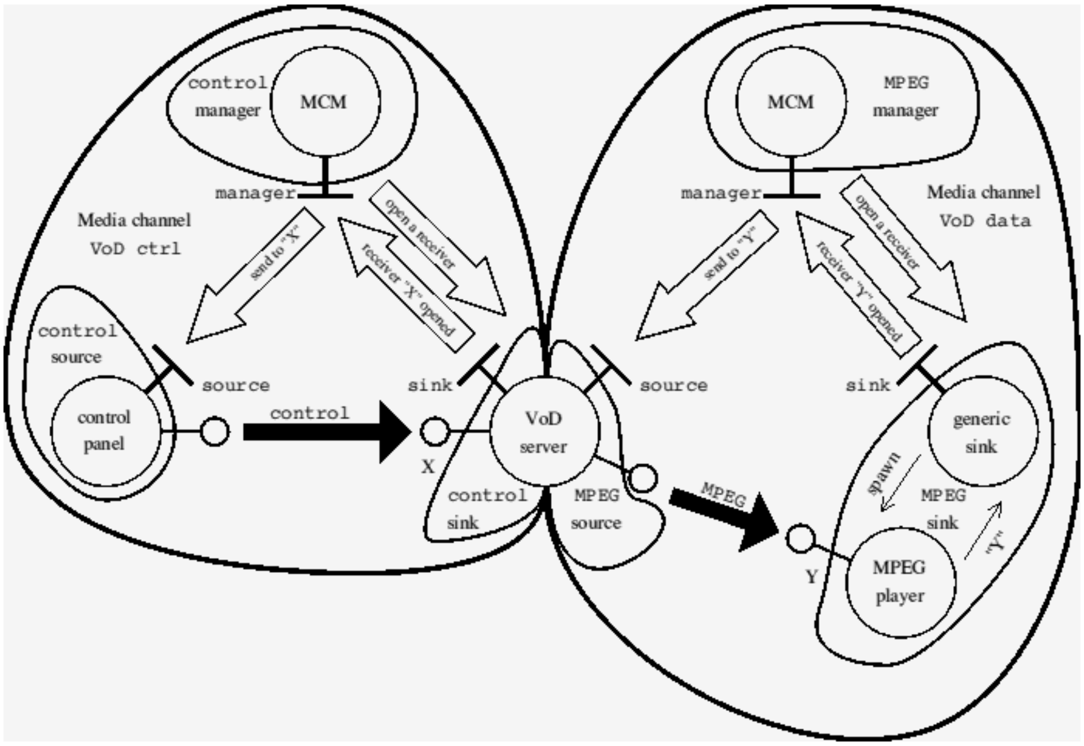


Figure 4.10: A VoD server architecture

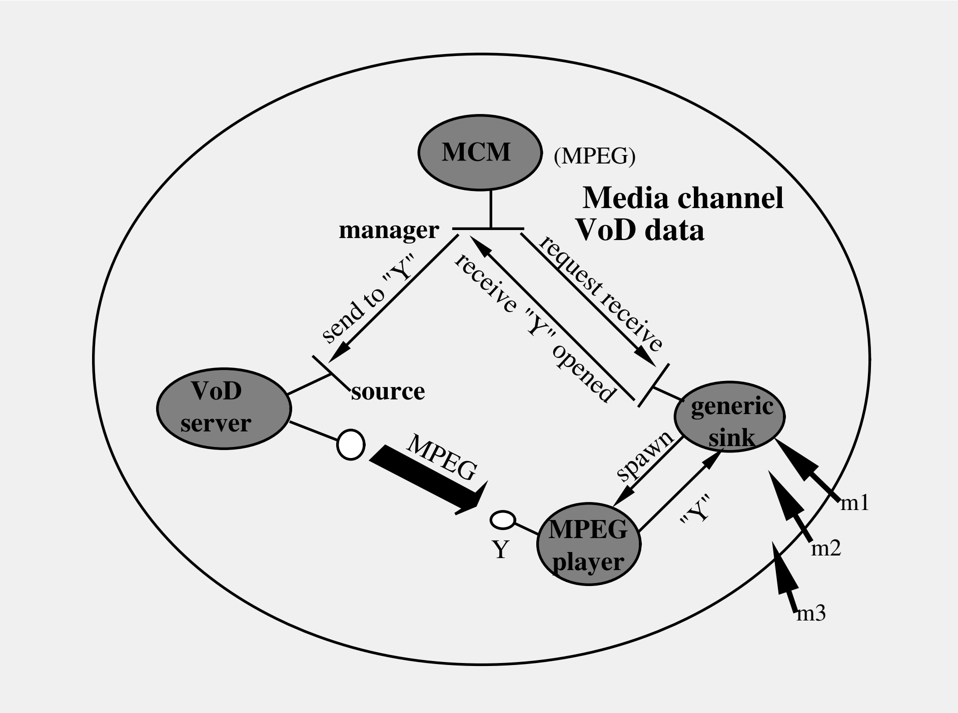

MCM
(MPEG)
Media channel
VoD data
manager
send to "Y"
receive "Y" opened
request receive
VoD server
source
generic sink
MPEG
spawn
"Y"
Y
MPEG player
m1
m2
m3

Figure 4.11: Application topologies

In Figure 4.12, a configuration of sources and sinks is shown as a two dimensional table in sending and receiving data from two media channels (in VC each source sends to every sink; each sink receives from every source; in VoD the roles of sending and receiving are accomplished by the use of the two media channels i.e. VoD Data and VoD Ctrl).

Each element of the table for a source, for example A, is consumer address provided, and to which A must ensure it is transmitting (e.g. e1). Thus, A is transmitting implicitly to E. This table shrinks and grows in both dimensions as sources and sinks are added or removed from the channel. The media-channel manager is responsible for ensuring that the table remains consistent. Each sink (accessed via its management address) is responsible for providing the media-channel manager with consumer addresses to fill the elements of the sink's row, and for decommissioning consumer addresses that are no longer required when the table shrinks. Each source is informed of which consumer addresses it should or should not send to as a sink is added to or removed from the table. Figure 4.13 shows the configuration for the source and sink management addresses for dynamic creation of sources and sinks.

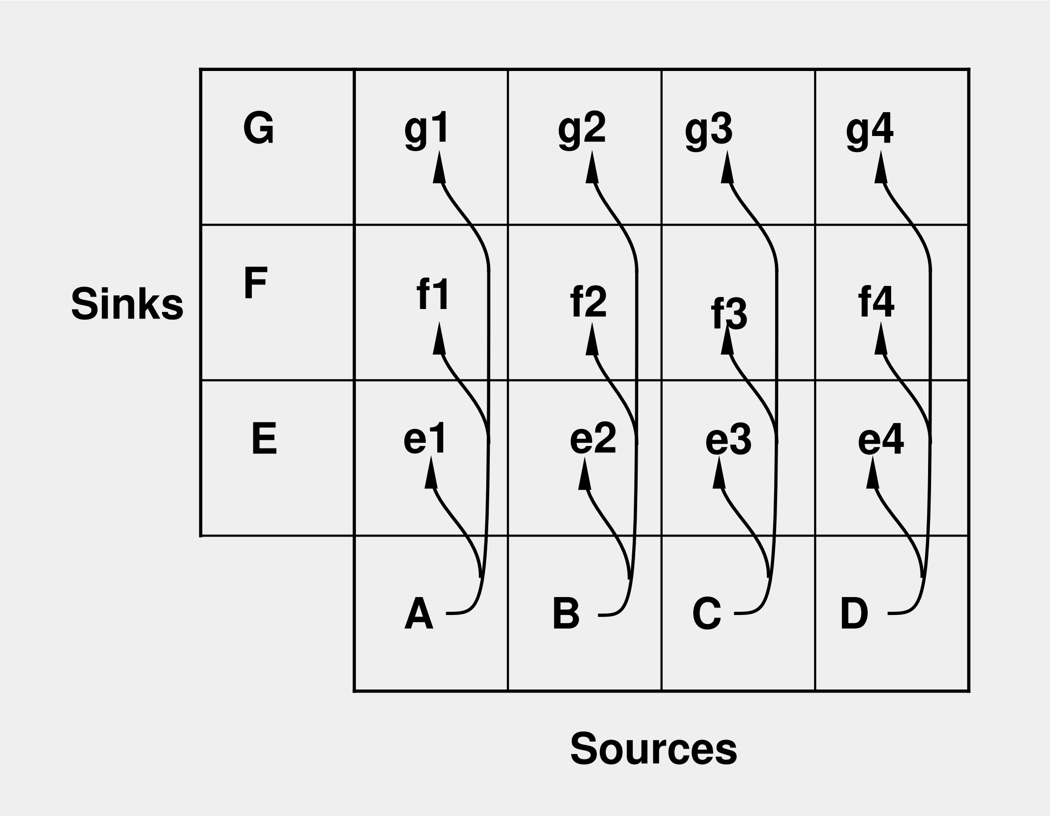


Figure 4.12: Media channel state

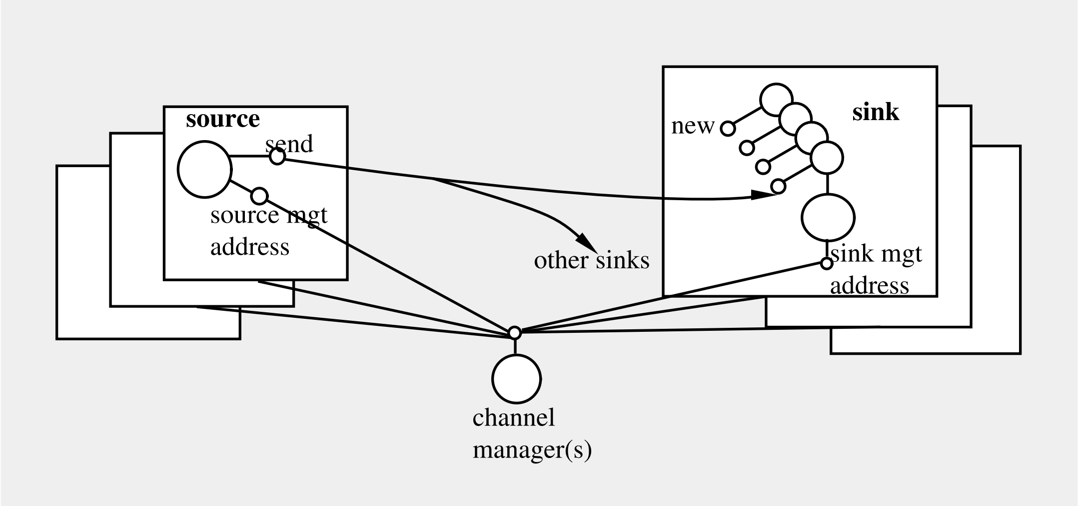


Figure 4.13: configuration of source and sink management addresses

4.2.2.4 Media Operations

4.2.2.4.1 *Who will run an MCM?*

This will depend on what it is for. In a simple telephone call, which has the potential for having many participants, the person who initiates the call will likely *run*, *own,* and be responsible for the *manager*. Alternatively, he or she may purchase a manager from a service; perhaps he wishes to run it centrally with respect to other users rather than on his doorstep. A VoD service may run an MCM to enable policies to be enforced on the channel.

4.2.2.4.2 *Who will run sources and sinks?*

Again this depends on the application: a VoD service will run a VoD source; VC sources and sinks will be run by the users they are serving.

4.2.2.5 Putting it all together- an application scenario of media channels

In a distributed multimedia application, media channels should be considered as objects around the network which have been created by people for their particular needs, and are destroyed at will by their owners. In this model, the media channels do not have any particular restrictions as to their context, such as belonging to exactly one group session. A *group session* is just a way of interpreting a set of MCs as being associated, that is a set of related media channels whose sources and sinks are all owned by a group of people such that each person in that group has a source and/or a sink for each of the media channels. The related channels are interpreted as multimedia group session, i.e. it is a session only because of the way it is used, not the way it is defined.

A user agent contains a set of tuples, each tuple containing a media channel, and an indication of whether the agent has a source and/or a sink connected to that media channel. When a user wishes to participate in group communication, he ensures his agent is running and then adds media channels that he is interested in to the set of media channels in the agent. (He can also remove those he is not interested in, that may have been left over from an earlier usage.) He can then run media channel clients for each of the media channels that he wishes to participate in (the agent should be configured to run them for him, like a web browser).

A user may get to know about various media channels that he may be interested in as follows:

- there could be a well known, permanent resource, such as BBC1,
  that he can get from some directory service;
- he can create one himself, such as a video that he wants to show

to other people, that he is responsible for managing;

- he could ask a regional service, such as a campus manager factory, to create one for him, so that he doesn't have to remain responsible for it if he finishes with it while other people are still using it;
- someone else could tell him about one ("Hey, they've now got channel 5 on the web - it's at..."), so it's becoming well known, at least to this person;
- someone may tell his agent about one, because they want to talk to the other person who wishes to have a video conversation with him, so he creates a channel, and passes it over to our user's agent; the user is informed that a new channel has been given to him, along with a message about who's calling, and the user can choose to answer (by joining the channel) or not (by removing the channel).

The advantage of using the media channel model is that each user maintains a list of channels that the user is interested in, and when this list is the same as another user's lists, one might interpret them as being in a single group session, but any of those members is free to branch out to other things. There's no need to have a central object as a group session manager, with a table of channels against users, which is too restrictive when it comes to merging groups which may be using the same set of media channels.

4.2.2.6 Example:

A user, say **B**, might be watching a football match with the sound turned off, whilst participating in a serious UFO discussion with other users, say **A** and **C**, but nobody else in the discussion needs to know about the user's participation in the match.

A: Anyway, I want to bring up this next batch of sightings we've had over Scotland.

B: Yeees!

C: Have Carlisle scored?

B: Yeah. First time in two years!

A: I wouldn't mind some attention here, please.

C: (To B) Can you give me the address?

B: Sure.

B:textchannel:

group://sports.cumbria.org.uk/matches/live/carvbirm

C: Thanks.

C now watches the match with **B.**

A: You two can leave if you're not listening.

B: No, no! We are, really.

C: What he said.

A: No, you're not. Go away.

User **B** creates a new VC channel, and sends the address to **C**, who joins it. **B** and **C** then leave the other VC channel, and **A** goes on talking to the rest of his flock. However, **B** and **C** retain the text channel, so if anything important happens in **A**'s conference (important enough to be typed), they can rejoin

Thus the media channels are resource objects, and their management services specify how the resources should be used and the operations that are applicable to them. The next section deals with these aspects, for designing MGMS.

4.3 The Media Channel Architecture

Given the definition of a media channel and its associated properties as outlined above, it is appropriate to consider a media channel as comprising several instances of given class of managed objects (sources and sinks). In [Tucker 94], it is noted that a managed object definition should strictly be described as a *managed object class definition*, and a class may be thought of as a set of managed objects satisfying a given type (i.e. satisfying a given definition) that exist. In this respect the "media channel" object provides instances of sources and sinks that are managed by the media channel manager. The creation and deletion of sources and sinks offers the flexibility of using a given media type, and also accommodates the dynamic changes in the use of the media. In this section, an architecture for a set of Multimedia Group Management Services (MGMS) intended to provide generic system level support for multimedia group based applications is described.

4.3.1 Media Channel Communications

The concept of a media type has been elucidated as a class definition for enabling media channels to be created for multimedia communication. The use of various media channels in the media channel management entails managing objects (resources) around the network. For example, the owners of video conferencing facilities may offer a video conferencing channel such as the ones used by MBone. This channel may be a composite channel, i.e. made up of different media types e.g. audio, video, text. etc. Or the video conferencing application may be constituted by different media channels. It really depends on the application, the services provided to the application, and the policies imposed on the use of a given media channel.

The following section discusses the envisaged services and the primitives that model the services.

4.3.2 Services and Service Primitives

It has been reiterated in parts of this chapter that MGMS should be flexible enough to meet the user's requirements specified according to applications' needs. To achieve the flexibility required, we need to specify classes of services that will ultimately give rise to type of services by the individual applications. Basically, the MGMS is required to provide facilities to:

- add a user to a media channel;
- remove a user from a media channel;
- allow the user to control the stream from a given source;
- track the media-channel-relevant information and access to the information from different locations;
- add a media channel to an application;
- remove reference to a media channel from an application.

For applications developers, it should be possible to configure each of the above operations as a service element. Each service element has a number of service primitives[1] associated with it, and which provide the user with an Application Programming Interface (API). These issues are discussed in the following sections.

Add user to media channel

The user wishes (his agent) to participate in a channel. The user can participate in any number of channels at once. The user can participate in each channel zero or one time as a source of that channel, and zero or one time as a sink of that channel. The agent identifies the type of the channel, and selects an appropriate source or sink object to invoke. Two primitives are identified with this service element:

---

[1] Primitives are notional operations that can be brought together to provide services. Primitives fall into one of two categories: they are either *status* or *control*. Status primitives *find out* something, control primitives *change* something [Coffield 87]

add_user_src_to_mc(channel);

add_user_snk_to_mc(channel).

The user agent invokes the operation such as *src_joining* on the channel, and then behaves according to the *include_output* and *exclude_output* responses returned by the service primitives as shown in Table 4.2.

Similarly the user agent invokes the *snk_joining* operation on the channel, and then behaves according to the *prepare_input, accept_input,* and *reject_input* responses returned by the service primitives shown in Table 4.2 (page 69).

Remove user from media channel

The user wishes to end his (agent's) participation in a channel as either a source or a sink. The agent identifies which of its "child processes" is acting as the source or sink for the specified channel, and removes it. Two primitives are associated with this service element, viz:

remove_user_src_from_mc(channel);

remove_user_snk_from_mc(channel).

Import media channel (add media channel)

The user wishes his agent to be aware of an existing channel. No channel is created; merely the address of an existing channel is given to the agent. The primitive associated with this service element is:

import_mc(channel)

Discard media channel (remove media channel)

The user wishes his agent to discard the reference to a media channel. The channel is not destroyed. The service primitive used is:

discard_mc(channel)

Create media channel

A media channel is to be created and to behave according to a set of policies. The result is a reference to a media channel. The policies are essentially application dependent; for example access rights to the media channel, maximum number of participants, mode of participation for each participant (i.e. as a source or a sink or both).

How the channel is created is beyond the scope of this specification. The primitive associated with the service element is:

create_mc(policies)->(channel);

Destroy media channel

Destroy an existing media channel. Its reference subsequently becomes invalid. How the channel is destroyed is beyond the scope of this specification. The service primitive used is:

destroy_mc(channel)

The various operations on the media channel are handled by the media channel manager which instructs the sources and sinks to include or exclude the sending to or receiving from addresses in their lists. The syntax of the operations in all cases is:

sender: receiver.message(arguments)

Table 4.2 shows the various operations that are invoked by the sources and sinks on the media channel manager and vice versa, in order to register with the media channel.

In summary, this section has shown the structuring of media channel management services based on service elements and their associated primitives and operations based on the media channel model. The implementation of these services is the subject of the next chapter.

4.4 Related Work

The concept of a *channel* has been exploited in various ways to provide a service that can be used by multiple participants. For example a TV channel such as BBC1 allows any one to participate and receive the broadcast programmes. It is an open channel. Other channels such as SKY MOVIES are closed channel since a potential user must explicitly indicate to the service provider that he wishes to participate in the channel, and may be allowed do so on subscription to the channel in accordance with the stated requirements.

Within the domain of computer applications channels have been discussed as means of sending data from one source to multiple receivers. They are usually referred to as multicast channels e.g. in [Pasquale 92], [Pendergast 93].

CLASS OPERATIONS	CHARACTERISTICS
snk:mc.snkjoining(media type, validate agent); snk:mc.snkleaving();	The invoking sink is registering (snk_joining) or deregistering (snk_leaving) with the invoked media channel. The channel should invoke other calls on the sources and sinks subsequently registered with it, according to the policies which the channel was associated.
src:mc.srcjoining(media type, validate agent); src:mc.srcleaving();	The invoking source is registering (src_joining) or deregistering (src_leaving) with the invoked media channel. The channel should invoke other calls on the sources and sinks subsequently registered with it, according to the policies which the channel was associated with when created.
mc: src.reach(output : address) mc: src.close(output : address)	The source should include or exclude `output' from the list of addresses that it multicasts its stream to.
mc: snk.prepare(mgrref, snkref : reference) snk: mc.prepared(mgrref, snkref : reference; input : address)	The manager wishes to connect a source to the sink, and is asking the sink to provide an address to receive through (an input). The sink should refer to this connection as mgrref when responding to the manager, and if snkref is not NULL, then the manager believes that the sink has already provided an address which the sink refers to as snkref. The sink should find the input identified by snkref, or create a new one if snkref is NULL, and should return prepared_input(mgrref, snkref, addr), where addr is the (new) address of that input.
mc: snk.listen(mgrref, snkref : reference; input : address)	The manager wishes to connect a source to a sink, but has already arranged an address (input) through which the sink will receive the stream. snkref is a reference to the sink's existing structure that represents the state of that input, or NULL if it has no existing structure.
mc: snk.snkclose(mgrref, snkref : reference)	The manager wishes to terminate a connection between a source and the sink. The manager identifies the connection by mgrref, while the sink identifies it by snkref.

Table 4.2: Operations on a media channel (mc)

The work described by [Gupta 93] presents the notion of *channel groups* in a single distributed application, with the view that a number of real-time connections do have certain relationships among them. The introduction of channel groups enables the clients to describe these relationships explicitly to the network service provider. For example a sharing relationship may enable the network to share resources among related channels, thus lowering the cost and improving scalability of communication. The resource sharing relationship specification consists of a set of channels along with the specification for the entire group. The relationship says that at any network entity, the resources required by all the members should not exceed the resource specification for the entire group. Other relationships such as inter-stream synchronisation, relative dropping priorities, may exist. Since the objective is to have an application define relationships among channels, the *channel groups* clearly address issues above the MGMS model. It is an example of how to provide facilities for setting a group session, which is application-specific. Media channel management services provide the flexibility to the individuals to use different media channels at will without the concern for the relationships among the channels. Within the framework of the media channel model group sessions are not formally defined. A multimedia group session is just a way of interpreting a set of media channels as being associated with a given application, i.e. it is a session only because of the way it is used, not defined. Therefore the media channel mechanisms are means that the channel group abstractions may use to create a multimedia session with groups and sub-groups, a consequence of the definition of the media channel. However, it is important to note that this work introduces an important element of requirement in the design of MGMS, i.e. the scalability of the media channels to multiple participants. This has been provided for in the design of the MGMS by allowing media channel manager to effect the instantiation of several sink addresses, and hence each sink is able to receive media data from multiple sources of the same media type.

In [Wilde 94], a framework for Group Management Services (GMS) is described. The model does not specifically describe group management services for multimedia applications but presents a framework for applications for sharing information in collaborative work environments. The GMS provides a set of object types which are used to construct a data base of information that group members can share during collaborative activity. Although the GMS is an application level process, it points to the requirement for the support of multiple media types required in collaborative environment, and would therefore profit from the services of the MGMS to support collaboration.

A parallel approach to the design of the MGMS is the work described in "Communication Systems Supporting Multimedia Multi-user Applications" by [Heijenk 94]. In this work, the idea of service elements and service primitives are presented. While the MGMS refers to sources and sinks as means of establishing streams of data between communicating end-systems, in [Heijenk 94] they are referred to as INs and OUTs respectively. A Call Model approach is used; it is defined as a dynamic association between users and the service provider. The service provider offers the functions of one or more (possibly related) media types to the users. Although the approach is, in many respects, similar to the design of MGMS, the Call Model is suitable for the transport service user. It would be unsuitable, as observed in the design criteria for MGMS, for application developers for interactive multimedia applications to build directly on the mechanisms the Call Model provides. Rather these should be hidden by the facilities such as those envisaged in MGMS. In the Call Model, the applications manage transport connections. Besides, the Call Model does not indicate how the envisaged services are to be implemented, while MGMS does consider an implementation strategy. Above all, the distinctive feature between the Call Model and the MGMS is that in the former, group sessions are formally defined by a 'call'. By abandoning the group session (or call) in MGMS, definitions of subgroups (and sub-sessions) are easier to define (because the channels aren't exclusively associated with a single session any more). The multimedia group session topology was illustrated in Figure 4.10, but in [Vin 92], [Heijenk 94], a multimedia group session involves a set of users orthogonally registered against a set of applications, and all the users participate in all the applications, under one Group Session Management entity (GSM). A user invokes the GSM by the Group Session Agent (GSA). The GSM then initiates connections onto the media channels to the user. This is illustrated in Figure 4.14. However, the argument against this configuration is that the media channels are exclusively associated with a given session, thereby being too restrictive. The media channel model proposes a more flexible arrangement as shown in Figure 4.15 for which the group session agent is now configured as the User Agent (UA) as described earlier in the chapter. The user is free to participate in any number of media channels, which are not associated with any given session, and which are user configurable.

The media channel approach, as illustrated in Figure 4.16, allows the implement of the group session manager (S) as just one more media channel (discussed in detail in Chapter 6). There is no sense of a privileged user who can control access to the 'call' as is the case in the Call Model. This is now just a user who can access the user-agent as a source. If there is no such manager, then there is also no such special user. By expressing the access rights as a media channel, we could have any number of special users, i.e. we do not assume that there will always be exactly one. This situation is illustrated in Figure 4.16.

Figure 4.4 showed the interaction the source-sink interaction: a source sends a single stream of a particular media type to multiple sinks. Each sink sends a QoS request of the media type to each source. This definition is essentially missing in the call model.

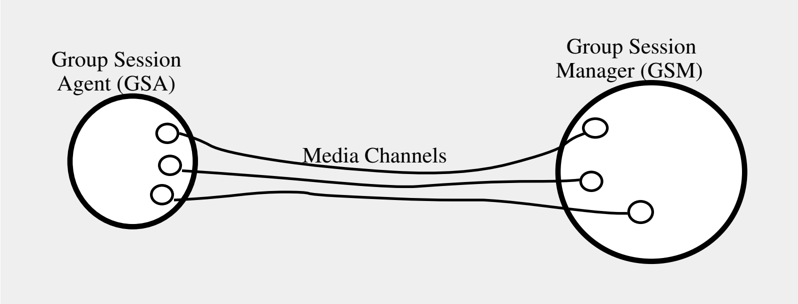


Figure 4.14: General Group Session Configuration

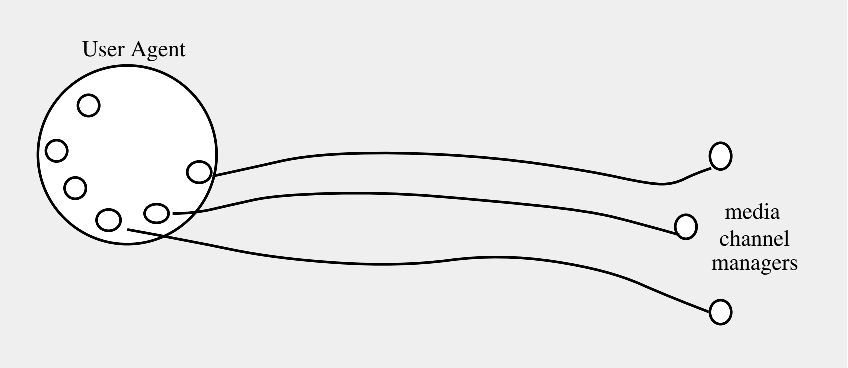


Figure 4.15: Configuring Media Channels with User Agent

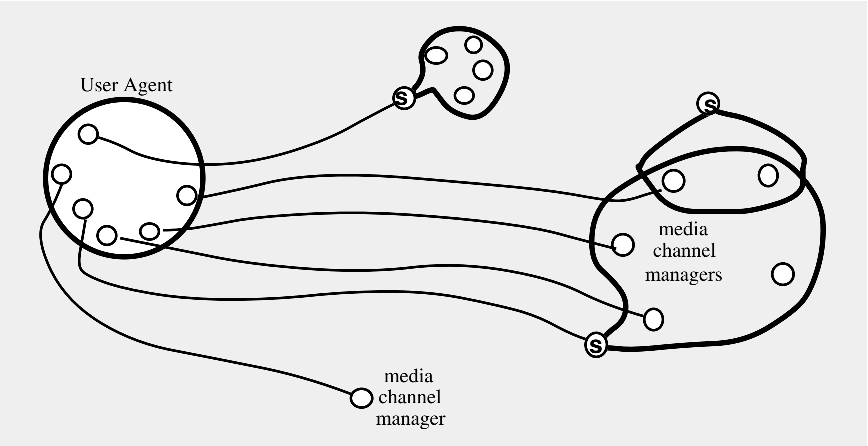


Figure 4.16: User Agent configuration for a group session

4.5 Discussion

The design has presented a suite of primitives and operations on the media channels to provide multimedia group management services. These services are generic i.e. they are not tied to any specific type of media employed in group communication, but are configurable to applications specific requirements. Hence the same services are used for different media types. In this way the model provides for generic interfaces for provide a unified management approach for dealing with different media channels, whether the channels have *focused* or *unfocused* sources and sinks.

The media channel manager configuration provides a generic interface via the sink management interface as shown in Figure 4.17. The sources' and sinks' interfaces do not depend on media types, thus allowing a management entity to communicate with them using similar services.

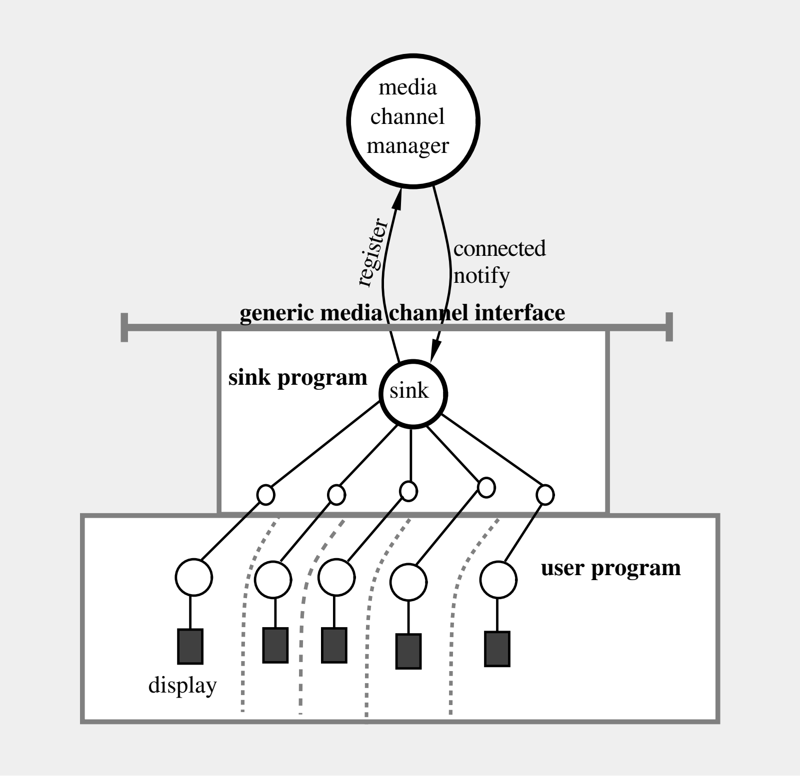


Figure 4.17: Generic Interface

The communication between the sink and manager objects is always the same for any media type; it essentially involves the passing of an address from the user process spawned off by the sink to the manager so that a source can be instructed to send to that address.

For a video application, the functions of the generic interface may be performed as a template to the interface between the sink and user programs since individual consumers can run independently, i.e. there is no interaction between them. The displays represented in Figure 4.17 may be images from different sources of the same media type.

For an audio application, the sink processes receiving from different audio sources need to be mixed to form a single output for the audio output device (e.g. */dev/audio*). However, more work is required for this conversion and therefore the sink/user programs interface is not generic to the audio application. (This is illustrated in *Section 6.4.2.2*). Hence the general case of providing a generic interface with our model is at the sink management interface and the media channel manager.

It suffices to say that the generic interface scheme employed for management hides a real architecture: all multimedia applications use the same control (i.e. connection interface) in that applications can be controlled or managed independent of application type (video, audio, etc.). In other words the design provides a class of services to applications developers.

The above mechanisms provide a common way of writing source and sink applications for any media channel. In the implementation (*Chapter 5*) the above functions are provided as a set of libraries. These mechanisms supporting multimedia communications are generic to all media types.

A further merit of the service specification in this model is that it isolates communication management from applications specific details. This is particularly important as it simplifies the programming abstractions, allowing easy access to communications facilities such as multicast support, and the location of multimedia resources without the concern as stated before.

The emphasis on developing the MGMS model has been on the management of individual media types. Since the individual media types have no knowledge of groups, sub-groups can be set up using the same media types; the scenarios depicting such an arrangement are discussed Chapter 6. The implication of this approach is that different multimedia applications can be freely and spontaneously set up and terminated during the course of collaboration. This is desired especially where participants involved in a collaborative activity engage in different tasks simultaneously.

4.6 Summary

The media channel model consists of three major components: the media channel manager object (**manager**), the media channel source object (**source**) and the media channel sink object (**sink**). Figures 4.9 and 4.10 show the configuration of the objects for an application that may use a media channel of type MPEG. The objects interact through interfaces which are typically of two types: signal interfaces and stream interfaces. The signal interfaces provide the communication between the media channel manager and the sources and the sinks, while the stream interfaces provide the data flow between the sources and the sinks. The bindings between the manager to the sources/sinks provide the end-to-end abstractions that allow an arbitrary m:n connection i.e. they allow m sources to send to n sinks. This situation is illustrated in Figures 4.12 and 4.13.

4.7 Conclusion

The objective of designing Multimedia Group Management Services is to provide generic services to multimedia applications for group management. The strategy employed is to separate the application-oriented issues from the communications concerns.

A number of other research activities concerning multimedia group communication have approached management issues with the view to sharing information, and because of the lack of support from underlying communication systems, the tools developed in these models have tended to be specific to group session applications. Furthermore, the models have been too restrictive in the use of media types, for example, not allowing the flexibility for users to form sub-groups within a session. The design presented here is an open architecture that surmounts such restrictions.

Each of the service elements defined and the specification of its primitives and associated operations provides the flexibility to use it as a generic interface to any given media type, simplifying the communication semantics between the application and communications sub-systems. Subsequently the task of managing group communication in a multimedia application environment is greatly simplified.

However, it must be stressed at this point that in the media channel model, only control messages are passed between the sources/sinks and the media channel manager, to effect establishing a channel for communication with a given media type. The model does not concern itself with the data formats, synchronisation and the monitoring and maintenance quality of service i.e. orchestration. Nevertheless the reliability of the control message is highly desirable.

In conclusion, the media channel model meets the basic requirements for multimedia group management services embodied in the service elements viz:

- providing means by which several sources send to one or more sinks of the same media type and thus catering for media scalability;
- controlling the population of participants using the same media type through use of the media channel manger with associated policies;
- co-ordination of operations on the media channel by the media channel manager;
- provision of different rights to access the media channel by participants;
- providing an API for application developers.

How an implementation of the conceived services may be realised, is the concern of the next chapter.

Chapter 5

An Implementation of Media Channel Services

The design of Multimedia Group Management services has been presented in the previous chapter. The design provides an architecture for generic services based on the service elements that are required for use with three classes of objects, viz. the sources, sinks and the media channel manager plus the user agent. In the media channel model, managers, sources and sinks are objects distributed across the network. The management services are meant to enable potential users to access given media channels by invoking these objects through their media channel user agents.

The media channel model provides a framework of abstract services which are described as generic services and which would enable a high-level management entity to create and manage a distributed multimedia group session. As has been reiterated in parts of the thesis, the media channel model provides for the realisation of a group session as a set of related media channels; group sessions are not formally defined, but interpreted as a set of related media channels in a multimedia application.

The abstract model defined services to add sources and/or sinks when a potential user joins a media channel, or to add or remove a media channel, or to control the behaviour of a media channel. Some of these capabilities are provided by the protocol defined for the communicating objects as shown in Figure 5.1. The fundamental design concept is to provide a generic interface to multimedia applications where several media types are involved. It then becomes easy to manage a multimedia group as well as sub-groups communication in a group session.

The aim of this chapter is to present an implementation of the media channel services for multimedia group management in a distributed environment. The implementation involves two categories of service: the low-level services that are used by the objects to communicate among themselves; and the high-level services that are invoked by the user agent.

The chapter is structured as follows: *Section 5.1* presents the service model, underlining the protocol issues involved in designing the MGMS model conceived in *Chapter 4.* Then the implementation model is presented in *Section 5.2. Section 5.3* discusses some communication infrastructure required to support the management services. A summary of the implementation is presented in *Section 5.4.*

5.1 Service model

By way of definition, services are high level functions provided to support the system [Coffield 87]. In a distributed system a service is an abstraction of a set of operations provided to users allowing them to perform a particular function. Services can be divided into *application* and *support* services. For example Netscape is an application service that allows a user to browse the world wide web. Others in this category include electronic mail, directory service, document data transfer, and Electronic Data Exchange. A communication service such as Mbone is a support service that allows users to establish connections and set up group sessions by providing conference communication facilities.

A service provider uses the resources of the distributed system to provide a service to users. Resources can be both software and hardware entities. In the thesis, media channels have been considered as resources distributed across networks. They are communication resources and therefore belong to the category of support services.

5.1.1 Media Channel Management Services

The MGMS process entails the provision of management services to enable:

- the querying of information on given media channels
- joining and leaving of media channels
- instantiation of new media channels for a multimedia group session
- establishing sessions and sub-sessions
- media channel membership administration

The requirements for these services are fulfilled in the following communication protocol and membership policies.

5.1.1.1 Protocol

This protocol defines the allowable messages between four possible types of media channel objects: source, sink, manager, agent. The configuration of message flow is illustrated in Figure 5.1.

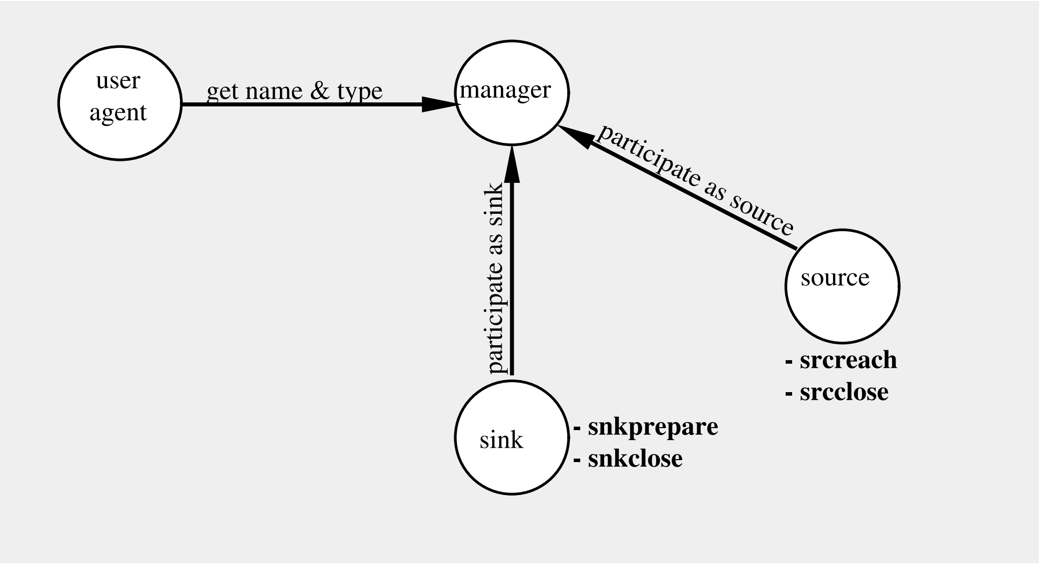


Figure 5.1: Communication between media channel objects

There is an interface between the source and the manager, the sink and the manager, and the agent and the manager.

The requirements of the agent-manager protocol are that the manager should be able to ask the agent to validate a sink or source address("is this sink one of yours/authorised by you?"), and that the agent should be able to obtain the name and type of the media channel from the manager.

For the source-manager protocol, the source should be able to indicate that it wishes to participate in the channel as a source, or not. In addition, the protocol should be flexible enough to allow the manager to instruct the source to set up connections using possible multicast mechanisms that underlying communications system can support.

For the sink-manager protocol, the sink should be able to indicate that it wishes to participate in the channel as a sink, or not. In addition, the protocol should be flexible enough to allow the manager to instruct the sink to set up connections using a number of possible multicast mechanisms as described below.

5.1.1.1.1 *Multicast Mechanisms*

The media channel management facilities are responsible for arranging for locations of the connections, i.e. providing individual/group addresses. However, it is assumed that the network layer is able to provide addressing mechanisms so that data is communicated to *media channel members*. Multicasting is one such a mechanism, as opposed to broadcasting; the latter does not discriminate *group members.*

Multicasting has been and still is an area of active research, with add-ons to the requirements to support multimedia applications. Notable literature has been presented by

[Ngoh 91], [Waters 92], [Szyperski 93], [Pendergast 93] on multicasting for distributed multimedia applications. The main aim of multicasting is for a source to transmit data to a set of receivers or group members [Hughes 87] during a *group session*. The ability to support group sessions is currently being studied by the Internet Engineering Task Force [Handley 96], [Holfender 96].

Multicast support mechanisms that may be employed for a single stream along side how multiple streams may be implemented are shown in Figure 5.2. These mechanisms are **multicast address** and **multicast set.**

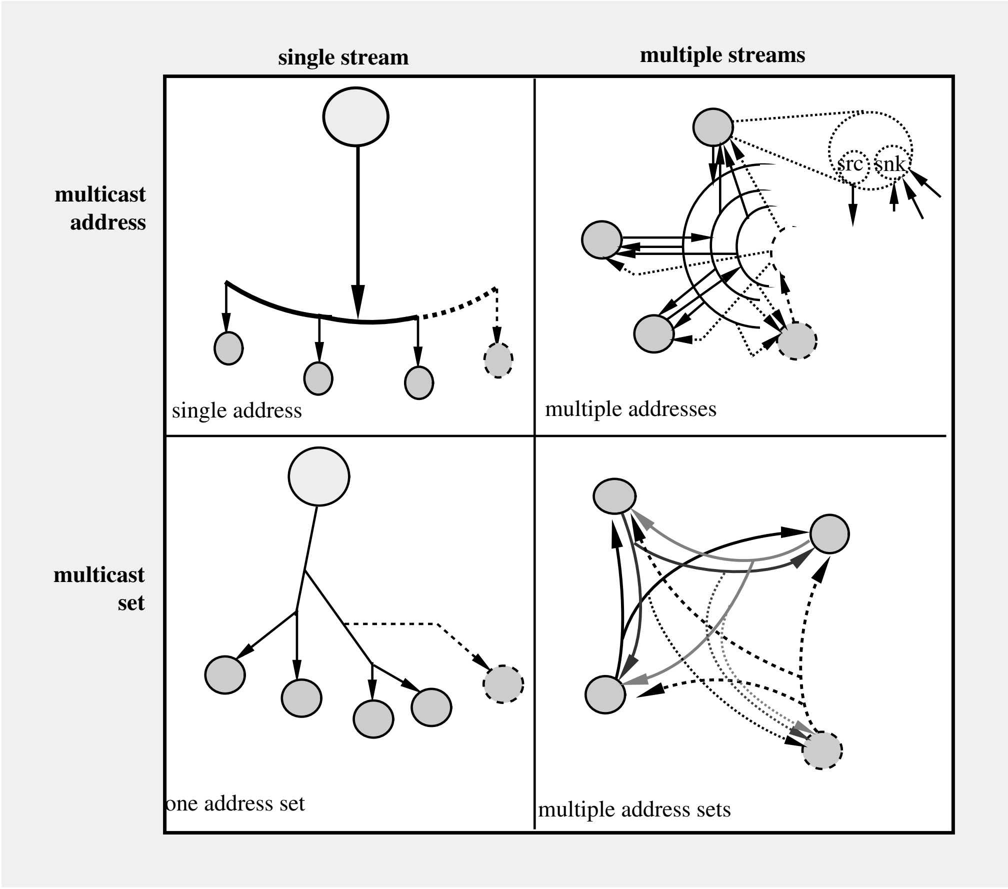


Figure 5.2: Multicast mechanisms

When multicast addressing is used, a multicast address is created for each stream, to which the source of that transmits, and to which all the sinks bind.

When a multicast set "addressing" is used, each sink *sets* a consumer address for each source, and each source transmits to a set of addresses consisting of one address from each sink.

The above mechanisms may be applied to multiple streams, constituting a media channel, where a user's set of consumers form a sink, and a user's producer is known as a source.

It should be noted that any of the above multicast mechanisms may be used for focused and unfocused group session types depending on what the transport system supports. On the other hand the application may choose to employ focused or unfocused media channels for the different group session channel types as seen in the VoD and VC examples respectively. A VoD channel involves a focused source, VoD-Ctrl channel is a focused sink, while a VC channel is unfocused.

5.1.1.1.2 *Protocol Syntax*

1) **agent-manager protocol**

To the manager:

`gettype` ();

`getname` ();

`isvalid` (address);

To the agent:

`typeis` (string);

`nameis` (string);

`validate` (address);

2) **source-manager protocol**

To the manager:

`srcjoining` (string, address);

`srcleaving` ();

`srcprepared` (connection ref, address);

To the source:

`srcprepare` (connection ref)

`srcreach` (connection ref, address);

`srcclose` (connection ref);

`channeldead` ();

3) **sink-manager protocol**

To the manager:

`snkjoining` (string, address);

`snkleaving` ();

`snkprepared` (connection ref, address);

`snkbroken` (connection ref);

To the sink:

`snkprepare` (connection ref);

`snkreach` (connection ref, address);

`snklisten` (connection ref, address);

`channeldead` ();

5.1.1.1.3 *Semantics*

1) **agent-manager protocol**

`gettype` () - the agent wants the channel's media type;

`getname` () - the agent wants the channel's human-readable name;

`isvalid` (*addr*) - the agent acknowledges that it owns or authorises the source/sink specified by the address *addr*, in response to '`validate`(*addr*)';

`typeis` (*mtype*) - the channel's media type *mtype* is as specified, in response to '`gettype` ()';

`nameis` (*name*) - the channel's human-readable name *name* is as specified, in response to '`getname` ()';

`validate` (*address*) - the manager wants the agent to acknowledge that it owns or authorises the source/sink specified by the address.

2) **source-manager protocol**

`srcjoining`(*mtypte*, *agt)* - the source wishes to participate in the channel as a source of the specified media type, and is authorised or owned by the specified agent *agt;*

`srcleaving` () - the source no longer wishes to participate in the channel as a source;

`srcprepared` (*ref*, *addr*) - the source is now sending on the specified address, in response to '`srcprepare`(*ref*)';

`srcprepare` (*ref*) - the manager wants the source to choose its own address to send to, and report back with '`srcprepared`(*ref*, *new_addr*)';

`srcreach` (*ref*, *addr*) - the manager wants the source to transmit to the specified address;

`srcclose` (*ref*) - the manager wants the source to stop transmitting to the address that is associated with the specified reference;

`channeldead`() - the channel is about to be destroyed.

3) **sink-manager protocol**

`snkjoining` (*mtype agt*) - the sink wishes to participate in the channel as a sink of the specified media type, and is authorised or owned by the specified agent;

`snkleaving` () - the sink no longer wishes to participate in the channel as a sink;

`snkprepared` (*ref*, *addr*) - the sink has prepared an address on which it is listening, in response to '`snkprepare` (*ref*)';

`snkbroken` (*ref*) - the resources that the sink has reserved for the specified connection have failed - the manager may request the sink to set up new resources;

`snkprepare` (*ref*) - the manager wants the sink to specify a consumer address that it can listen on, and return it using '`snkprepared` (*ref*, *addr*)';

`snkreach` (*ref*, *addr*) - the manager wants the sink to receive data from a producer with the specified address;

`snklisten` (*ref, addr*) - the manager wants the sink to receive data by listening to the specified address;

`channeldead` () - the channel is about to be destroyed.

Figure 5.3 illustrates a simple arrangement that involves two sinks and one source for a single media channel, showing the message flow from the manager and source/sinks at the source/sink management interfaces, for a VoD service (application).

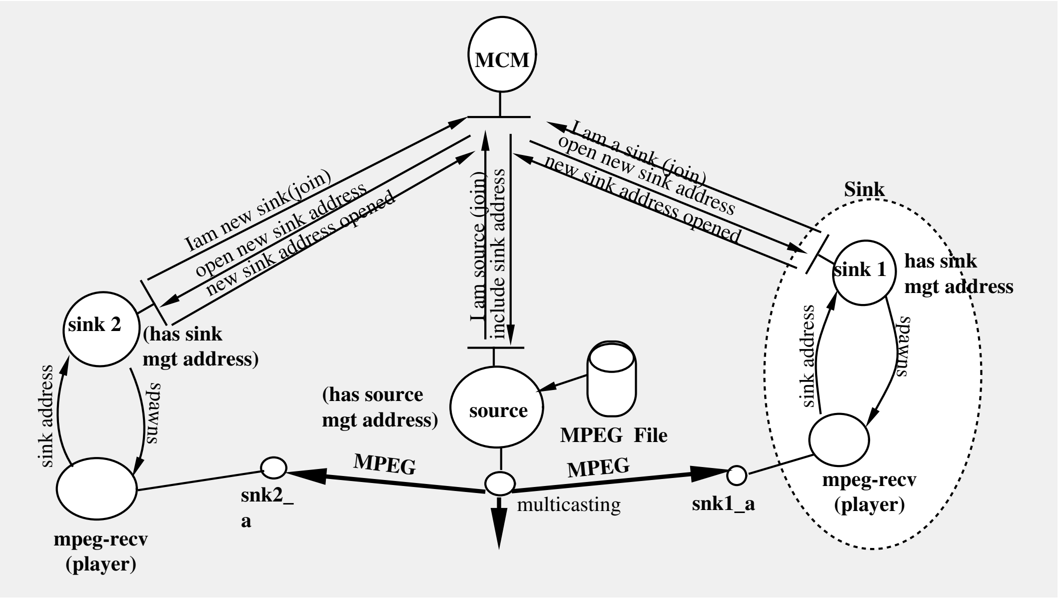


Figure 5.3: Service protocol involving a media channel manager (MCM), a source & 2 sinks for a VoD service

The sequence of events for the VoD service with a medium of type MPEG would be as follows:

1. Run the manager, *mgr*
2. Run a sink, `sink1`
3. Run a source, *src*
4. *sink1* is asked for a consumer address to receive from source
5. *src* is told to include *sink1*
6. Run another sink, *sink2*
7. *sink2* is asked for a consumer address to receive from *src*
8. *src* told to include *sink2*

The above steps could also be formalised as follows:

(manager starts; creates management address, *mgr*)

(sink `1` starts with argument *mgr*; creates management address *snk1*)

snk1:*mgr*.`snkjoining` ("MPEG", NULL)

(source starts with argument src; creates a management address *src*)

(source multicasts to { })

src:*mgr*.`srcjoining` ("MPEG", NULL)

(manager creates entry in table referred to by *srcsnk1*)

mgr:snk1.`prepare` (*srcsnk1*)

(*snk1* starts new viewer or whatever, with consumer address *snk1_a*)

snk:mgr.`prepared` (*srcsnk1*, *snk1_a*)

(manager now enters *snk1_a* into entry referred to by *srcsnk1*)

mgr:src.`reach` (*snk1_a*)

(source now multicasts to {*snk1_a*})

(sink 2 starts with argument *mgr*; creates management address *snk2*)

snk2:mgr.`sinkjoining` ("MPEG", NULL)

(manager creates entry in table referred to by *srcsnk2*)

(source now multicasts to {*snk1_a*, *snk2_a*})

mgr: snk2.`prepare`(*srcsnk2*)

(sink 2 starts new viewer or whatever, with consumer address *snk2_a*)

snk2:mgr.`prepared` *(srcsnk2*, *snk2_a)*

(manager now enters *snk2_a* into entry referred to by *srcsnk2*)

mgr:src.`reach` (*snk2_a*)

(Note that in this protocol the manager sends to objects declared as sources and sinks. In the contract of the services, if an object declares itself as a source, it should transmit to addresses that appeared in `include` messages and have not been subsequently `closed`.)

5.1.1.2 Instantiating New Media Channels

As stated before, media channels are resource objects and their management services specify how the resources should be used and the operations that are applicable to them. Instantiating a new media channel, say during a multimedia group session, involves running a media channel manager of specific media type. Sources and sinks can then register with it. Therefore, a media channel manager is a well known service or resource

5.1.1.3 Media Channel Membership Administration

Membership of a media channel is constrained by the media channel policy imposed by a higher level management entity and/or the media channel manager. When a media channel is created it is meant to behave according to a set of policies. These policies are effected by the media channel manager.

It should be emphasised that sources and sinks are the *mechanisms* by which a given media channel is utilised by an application (user). In the framework of management model [Davies 94], policies dictate how to utilise the available mechanisms and managers apply the policies to objects which provide the mechanisms. Thus "management", "policy" and "mechanisms" are canonical attributes of a service. According to [Davies 94], a formal definition of a policy is:

> *"a statement of principles or common purposes which serve as a continuing guide-line and establishes limits for discretionary action by managers .. policies are both restrictive and permissive at once".*

Thus policies limit the actions which a manager may take, but at the same time provide a design space within which a manager may operate. In this respect *mechanisms* may be defined as:

> *"the means by which a required result may be achieved",*

while the *management* is the process by which:

> *"the selection and use of the mechanisms according to the stated policies are made".*

Furthermore the media channel managers are the objects that carry out the management processes. They are relatively intelligent objects which construct a design space based upon the policies they are provided with and act within this design space to control an object or its behaviour. Thus the media channel manger object controls the current membership of sources and sinks to the media channel according to the media channel policies. The policies may be employed for:

- authentication
- security
- quorum (in case of a group communication session),

and they form media channel membership criteria. However, the policies are primarily application dependent. For example, a certain video conferencing (VC) channel may have a policy on membership of at least two sinks for VC transmission. The policy on a media channel quorum may apply similarly.

The above analysis of the service model forms the service architecture for media MGMS, i.e.:

- a set of operations on sources and sinks of media channels
- the mechanisms effecting the operations;
- management strategies applying desired policies;

How the services may be realised is implementation dependent. In the following section a description of an implementation for the media channel model is presented.

5.2 Implementation Model

The media channel manager effects the changes in the state of the media channel when a client becomes a source or sink of the channel. This section describes the implementation methodology of media channel management.

5.2.1 Environment used

Based on the above *abstract service* model for the media channel an implementation model is derived. The implementation has been done according to the hardware and software configuration available in Lancaster University Computing Research Laboratory, consisting of Sun workstations and PCs on the local area network. An outline of the environment is as follows.

5.2.1.1 Software

This consisted of ANSI C for programming and unix operating systems Solaris and SunOS running on various Sun Sparc workstations, as well as Linux on PCs. One of the Sparc workstations running Solaris was used as a producer of live video and audio data streams.

5.2.1.2 Configuring media transmission

A media channel object, called `mc_obj`, derived from generic distributed object (see Appendix: `mc_obj` **API**) was configured to effect formatted message transmission among media channel objects. In this environment objects may transmit unformatted messages to each other using TCP or UDP. TCP has been used for reliability for communications with the media channel manager. (UDP is used to implement streams between producers and consumers.)

The media channel objects are addressable through

`host:port.id`

The `id` allows objects to be multiplexed onto a single port.

5.2.2 Environment Developed

The implementation provides functionalities for:

- building media channel (MC) applications using the media channel library
- facilities to run MC applications.

A brief outline of these functionalities is as follows.

5.2.2.1 Media Channel Library

The media channel library (MCL) defines an `mc_obj`. The characteristics of the `mc_obj` are that:

- it can be network-distributed
- one `mc_obj` can send a formatted message to another `mc_obj`
- message formats correspond to abstract service model messages.
- it can be used to implement a source/sink/manager/agent. In this case the user is expected to configure `mc_obj` to meet the desired requirement.

The detailed structure of the `mc_obj` and the operations (functions) that are invoked upon it are presented in Appendix.

In addition the MCL defines `mc_src` and `mc_snk` (derived from `mc_obj`) from which a source or sink can be implemented respectively. The `mc_src` and `mc_snk` facilities reduce configuration requirements for applications programmers associated with `mc_obj`. They can also be used hide the network problems associated with:

- duplicate messages
- validation of references, which `mc_obj` does not.

The behaviour of MCL object classes is as follows:

`mc_snk` and `mc_src`

An `mc_snk` or `mc_src` object can be created. In order for these to process incoming messages in a user-defined way, the user must specify a set of functions (`mc_snkmethod` or `mc_srcmethod`) which will be called on various events. The description of these functions is outlined in Appendix.

A `mc_snk` or `mc_src` is opened by associating it with and object multiplexer (`obplex_t`), i.e.

- `obplex_t *om`: this gives the object s complete network address, and routes incoming messages to it.
- a manager (the full address)
- a set of user-defined functions, `*fs`,
- a user-defined value, `ctxt`.

The action of opening a `mc_snk` or `mc_src` involves initialising the local `mc_snk/mc_src` structure, and issuing a `srcjoining/snkjoining` message. To form the `srcjoining` or `snkjoining` messages, a media type and an agent address (currently unused) are also required.

Now, when other messages are received from the manager, the arguments are extracted, and some of the functions in `*fs` may be up-called.

mc_snkmethod

```
typedef void *(mc_snkopen_f)(void *ctxt,

            const obplex_objaddr_t *addr, mc_connection_t *ref);

typedef void (mc_snkclose_f)(void *ctxt, void *uref);

typedef struct {

 mc_snkopen_f *open, *openon;

 mc_snkclose_f *close;

} mc_snkmethod;
```

`(*mc_snkmethod::open)(ctxt, rcv, mcref)` is invoked when the manager has issued a `snkprepare` message to the sink. `ctxt` is a user-defined handle associated with the `mc_snk` object that represents the sink in question. The address that the function provides must be written into `*rcv`. The media channel library refers to this connection in future as `mcref`. The user-defined code refers to the connection as the `(void *)` value that the function returns.

`(*mc_snkmethod::openon)(ctxt, rcv, mcref)` is invoked when the manager has issued a `snklisten` message to the sink. `ctxt` is a user-defined handle associated with the `mc_snk` object that represents the sink in question. The address that the sink must listen or bind to will be found in `*rcv`. The media channel library refers to this connection in future as mcref. The user-defined code refers to the connection as the value that the function returns.

If the manager instructs the sink to close a connection, by issuing a `snkclose` message, the media channel library will invoke

```
(*mc_snkmethod::close)(ctxt, ref)
```

where `ctxt` is the user-defined handle associated with the relevant `mc_snk` object, and `ref` is the value returned by the earlier call to `open` or `openon` that established the connection.

mc_srcmethod

```
typedef void *(mc_srcinclude_f)(void *ctxt, const
   obplex_objaddr_t *addr,

                  mc_connection_t *ref);

typedef void (mc_srcexclude_f)(void *ctxt, void *uref);

typedef struct {

 mc_srcinclude_f *include;

 mc_srcexclude_f *exclude;

} mc_srcmethod;
```

`(*mc_srcmethod::include)(ctxt, rcv)` is invoked when the manager has issued a `srcreach` message to the source. `ctxt` is the user-defined handle associated with the `mc_src` object that represents the source in question. `*rcv` is an address to which the source must send. The returned value is the user defined reference for the address.

`(*mc_srcmethod::exclude)(ctxt, ref)` is invoked when the manager has issued a `srcclose` message to the source. `ctxt` is the user-defined handle associated with the `mc_src` object that represents the source in question. `ref` is the value returned by the earlier "include" call, and identifies to the user the address that must be excluded.

Closing down

```
int mc_snkclose(mc_snk *obj);
```

```
int mc_srcclose(mc_src *obj);
```

A `mc_snk/mc_src` object can be closed down with one of these two functions. This causes the relevant *srcleaving* /*snkleaving* message to be passed on to the manager.

5.2.2.2 Applications Provided

From the MCL, *mcmanager*, *mcsink*, MPEG-1, and SunAudio application programs were developed for "SunAudio" and "MPEG" media types. and used as follows:

"SunAudio"

- produce/consume 8 kHz and μ-law audio using /dev/audio device
- UDP sockets to transfer data; naive multicast used.

"MPEG"

- produce/consume MPEG-1 video
- `mpeg_recv` displays using X Window System
- `mpeg_live` captures and encodes using XIL
- UDP sockets to transfer data; naive multicast used

5.3 Summary

In this chapter, an implementation of MGMS has been described. The objective is to provide a flexible environment for multimedia group management services for applications engaged in group communications as well as programmers. This is achieved by managing individual media channels that may be employed in a given group session. Policies are employed to effect access rights to the media channel as well as providing security mechanisms. The overall merit of the media channel model is the provision of mechanisms to support a group session. This is accomplished by a session channel that maintains a list of media channels. As it has been reiterated in parts of the thesis, the media channel model does not have a notion of a group sessions, i.e. sessions do not formally exist. A group is simply a way of interpreting a set of related media channels by a session channel.

The media channel model has been used to structure the abstract services described and an implementation realised through a number of procedures that support high level and low level functions. The high level functions are initiated by the user (or an application), while the low level functions pass messages between the media channel objects. In the next chapter, we examine the application of the media channel model for multimedia group management services.

Chapter 6

Applications of The Media Channel Model

A media channel is used to model a way of communication and management of change in some shared resource involved in the media channel. The channel has a media type which defines the structure and the meaning of the state, what sort of changes can occur, and how they are expressed in communication. Sinks can register with a channel to observe the current state of the resource, and to receive updates to the state. The state is held by zero or more sources (the number of sources is also part of the state) which are registered with the channel. Each source sends details of its portion of the state to all sinks, and each sink can distinguish between data from one source and another.

The basic function of the media channel is to provide interfaces for high-level multimedia group management service abstractions to support multimedia group sessions. A group session is represented by a User Agent (UA). By abandoning the idea of media channels restricted to group sessions, definitions of subgroups are easier.

The media channel model allows the implementation of a group session manager as just another media channel "composed" of other media channel managers.

The aim of this chapter is to demonstrate application of the media channel model to (model) multimedia group sessions. The chapter is structured as follows. *Section 6.1* defines the characteristics of group session management. In *Section 6.2* the user agent is described. The user agent provides the interface between the user and the multimedia group management system. *Section 6.3* describes the experimental setup and briefly outlines the process involved in capturing and sending audio/video data streams. The issues to be investigated and the results of the investigations are presented in *Section 6.4. Section 6.5* deals with the performance analysis and evaluation of the results obtained in *Section 6.4.* A discussion of the performance of the media channel model is presented in *Section 6.6;* while *Section 6.7* forms the conclusion of the chapter.

6.1 Sessions

In order to participate in a multimedia group application using the user agent, the user enters references of the relevant channels into the user agent, and then indicating how he or she wishes to participate in each of them. This could be made easier by providing a database of the channels that form a complete application, and also indicating how a user should participate in each one (for VC, a source and a sink are recommended; for a VoD video channel, only the sink is useful). This database can itself be seen as a media channel, i.e. the database is the state, whose media type we define to be "session", and the state must be mirrored (copied) to potentially many users of the database. The multimedia group application is now referred to as a session. The database is initially configured and subsequently altered by connecting a source to the session channel, and a sink program can show the list of component channels to the user.

By combining the session sink and the user agent, participation in a session now only involves entering the address of the session channel, and activating the sink (which is part of the agent) of that channel. This sink then extends the list of channels held by the agent according to the database held in the state of the session channel, and then begins appropriate participation in each of them automatically. When the user wishes to terminate participation in the session, he or she destroys the sink, which causes all the other channels in the session to be removed along with their sources and sinks. (Note that the session database only offers a suggestion to the agent about how to participate. The user is still free to specify an alternative mode of participation.)

As a session is just another channel, it may form part of other sessions, allowing sub-sessions to be defined. The session media channel is characterised as follows:

In summary, the session media channel knows about the other media channels from the way it has been set up i.e. it has a list of media channel managers. The session media type characteristics are presented in the following sections.

"Session" Media Type

> *A "Session" media type is a set of media channels (media channel, participation) tuples that define what is available to the participants. Sinks are informed of new channels that become available to the "group" by the* ***session media channel manager****, and also which channels are removed, and the sinks (which are typically user agents) may choose to act on these as they wish. Sources (which also may be user agents, or any access control program) decide whether a Media Channel (MC) is part of the session or not.*

6.1.1 Session Management

6.1.1.1 User Agent Group Session Mechanisms

Suppose three people decide to engage in a co-operative activity. They have the capability of receiving audio, video, text (and other forms) of information through the source and sink mechanisms. The various elements of this scenario are represented as follows:

Let p_1, p_2, $p_3 \equiv$ the three people;

A, V, T ≡ Audio, Video and Text conferencing media channels respectively.

In addition to these channels, one of the three persons, say p_3, has already a VoD channel from which to view a film, say the "Lion King". As well as viewing the film, he can also apply control functions. Therefore two channels are available for the VoD service, i.e.:

$LK_D \equiv$ "Lion King" Data Channel

$LK_C \equiv$ "Lion King" Control Channel

The three persons now possess sets of sources and sinks as indicated in Table 6.1.

User	Sources	Sinks
p_1	(A,V,T)	(A,V,T)
p_2	(A,V,T)	(A,V,T)
p_3	(A,V,T,LK_c)	(A,V,T,LK_D)

Table 6.1: Participation in media channels

Now any of the three persons has the motivation to start a group session, that is by creating a group session manager. He may call this channel "Our_session" (S) of type "Session" (which has been defined.). The process for initiating a group session is as follows:

(1) p_1 : creates a new channel S, passes the address of S through some other convenient mechanism, e.g. T, to others;

(2) p_2 and p_3 (as sinks of T): see address of new channel, join S as sinks;

(3) p_1: also joins S as a sink;

(4) One of the users e.g. p_3, joins S as a source, adds A, V and T to list in S;

(5) p_1, p_2, and p_3: are informed by S of the channels A, V and T; (although they already have these, it is useful to define them as part of the session in case any one else joins on).

In effect, S is now responsible for ensuring that all its sinks (currently p_1, p_2, p_3) are aware of the channels in its database even as the new channels are added/removed. Additionally p_3 can add LK_D and LK_C to S. The session scenario is illustrated in Figure 6.1 and Table 6.2.

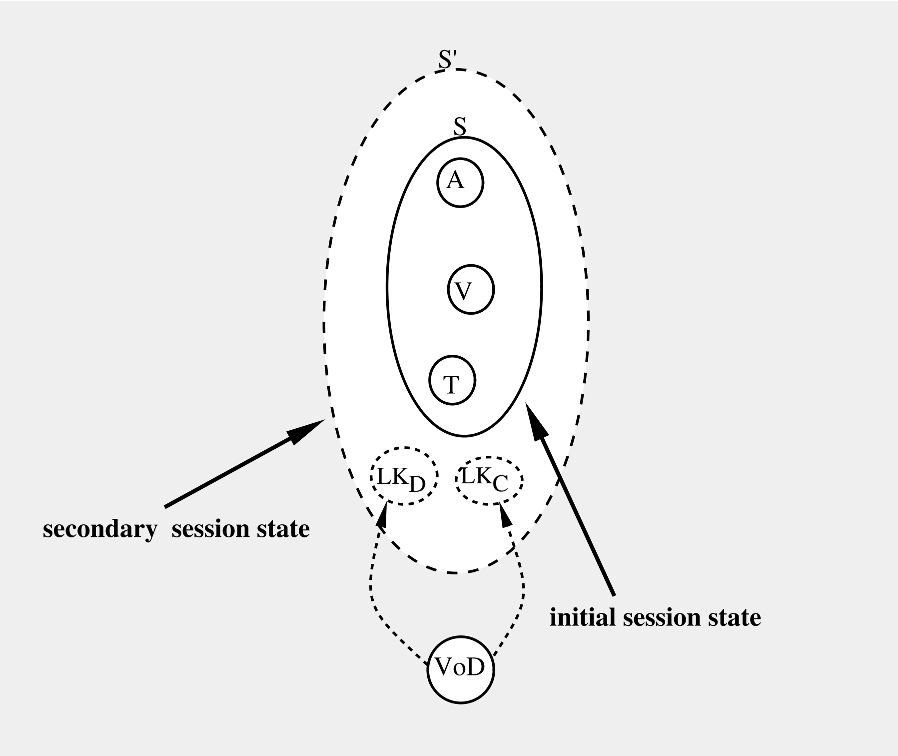


Figure 6.1: Media channel session configuration

Session (S)		
Media Type	**src**	**snk**
A (audio)	✓	✓
V (video)	✓	✓
T (text)	✓	✓
LK_D (Lion King)	▬	✓
LK_C (Lion King)	✓	▬

Key

✗ = not joined

✓ = joined

▬ = can't join

Table 6.2: Group session scenario.

If some other person, say p_4, wants to join the session (S), he joins as a sink of S so he receives the details of A, V, T, LK_D (including how to participate in each).

Consider the case of p_4 joining the session. The advantage of having the session media channel is that it enables p_4 to be informed of the state each of the other media channels in the session, instead of inquiring about each of one of them individually. Hence this is a more responsive management strategy.

In the spirit of the media channel model, the group session media channel manager has, among other tasks, to:

- monitor dynamic joining and leaving of media channels,
- make notifications of new media channels that become available to the session,

Authentication and security are done by another channel of, for example ":Membership". The membership media type is characterised as follows:

2. "Membership" media type

The state of a channel of this type consists of two sets of user agents. One set is for those that may join a particular session as sources, and the other set is for those as sinks. Sinks of "this" media type are media channel managers (or anything that needs to determine the validity of a source or sink, such as in cases where there is no centralised manager). When a User Agent (UA) sends a message to a Media Channel Manager (MCM) to indicate the validity of one of the UA's sources or sinks that has registered with the MCM, the MCM can check with this media type for the validity of the UA.

The functions of "Membership" media type would be:

- administer media channel membership, i.e. who can join onto each channel.
- security.

Session management involves the use of media channels that become associated with a session. We illustrate this further with the following example.

Consider a session setup for a discussion on Multimedia Group Communication (MGC). The session S is associated with a number of media channels for communication (including sub-sessions as media channels). Hence the session S for MGC is such that:

S = <media channel, participation> tuples associated with MGC. (1)

The implication of (1) is that S (actually the session manager), knows about the media channels - their addresses. A user joins S as a sink to find out how to participate in the session. Table 6.3 illustrates how S may present the session when invoked by a UA.

In Table 6.3, S_1 and S_2 are sessions for other media channels (although media channels themselves with respect to S), and hence may be construed as being sub-sessions of S (participants join onto session channels as such; they join onto other channels optionally as sources, sinks or both). In the example of setting up a MGC session, MGC begins by categorising the requirements of MGC and some broad categories of these requirements emerge as:

- Group Communication Services (GCS),
- Group Management Services (GMS).

The session then breaks into sub-sessions consisting of GCS and GMS. Some members of MGC join GCS while others join GMS. Just as S is associated with MGC, we may associate S_1 and S_2 with GCS and GMS respectively. We may present the association of the media channels with the relation "is known by" which is illustrated as a mapping in Figure 6.2.

Media Channel	src	snk
S	—	✓
AV	✓	✓
T	✓	✓
...	...	...
S_1	—	✓
S_2	—	✓

Table 6.3 Session Management

S_1 and S_2, as sessions (session managers) may be associated with their own media channels, represented by such sets as:

$S_1 = \{AV_1, T_1 ...c_1\}$,

$S_2 = \{AV_2, ... c_2\}$,

$S_i = \{c_i\}$,

where c_i is any other media channel in a given set such as a whiteboard, and i = 1, 2 .. for a given set.

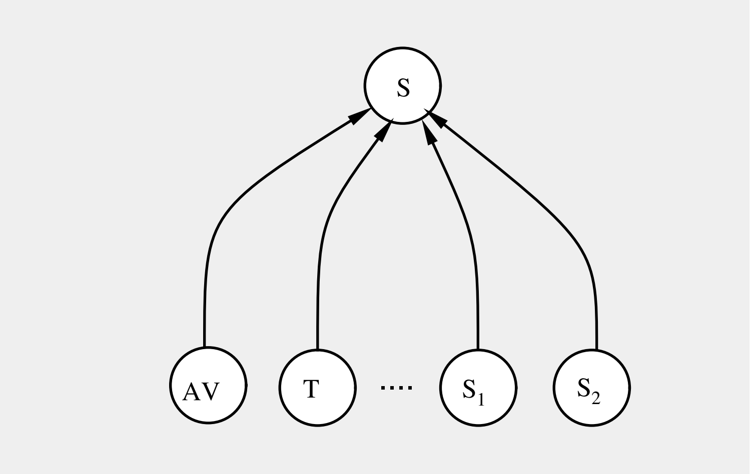


Figure 6.2: Media channel mapping onto session S

As has been noted before, media channels are distributed objects around networks. Their association with various sessions are realised by mapping relations such as that depicted in Figure 6.2. In the scenario presented, where S_1 and S_2 also constitute sessions, the overall mapping may be represented as shown in Figure 6.3.

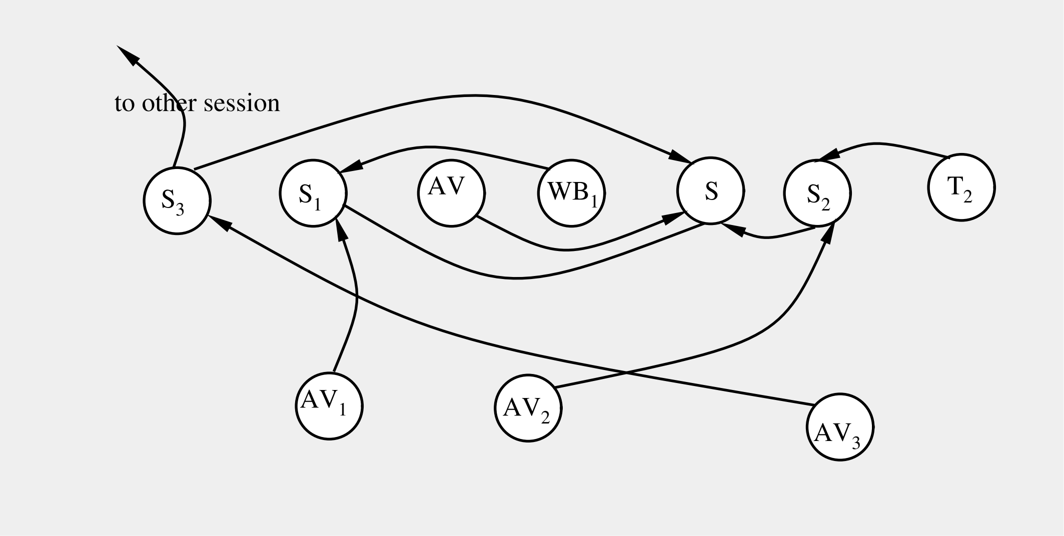


Figure 6.3: Distributed media session management

The advantage of forming sessions is that the session manager is responsible for the session address, which in turn points to various media channel addresses. When a session address ceases to exist, all the references to media channels associated with the session are automatically discarded at once, which is more efficient than having to leave the channels that the user in not interested in individually.

6.2 User Interface

Each participant runs a User Agent on his or her workstation. User Agents (UAs) are objects which run on each user's workstation and provide a user interface for multimedia management service facilities. The interface may be presented via a User Interface (UI) in terms of graphical/text menu selection. The UI allows access to functions such as :

- browse through available channels already obtained;
- start applications for new channels, and join or leave the channels.

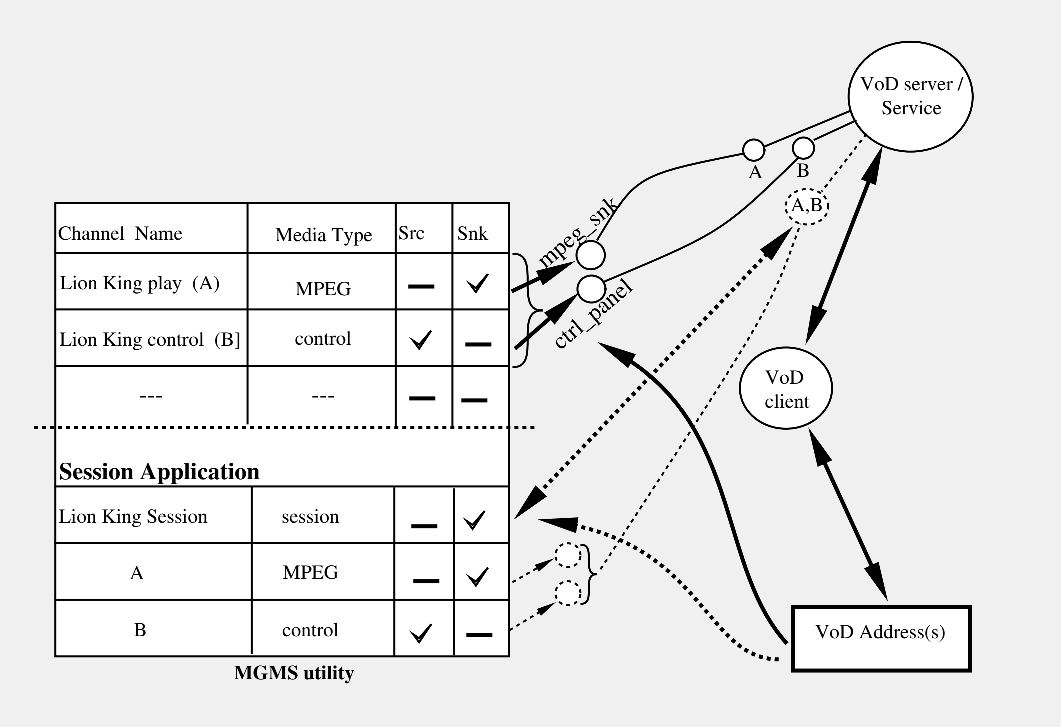
A sample display of UI for Multimedia Group Management Services is shown in Figure 6.5.

Figure 6.5(a) is the initial display when the GMS application is started. Figure 6.5(b) shows the channels that the user has obtained and may use them as per the requirement of the channel.

Suppose the user requires the service of Video on Demand server. The sequence of events is as shown in Figure 6.4. The user starts an application "VoD Service" which runs a VoD client. The VoD client has enough knowledge to enable it contact the VoD server for a service such as renting films. The client is then given a list of films (media channels), e.g. Lion King, that the VoD service maintains together with their addresses. The user then selects (copies) an address(s) and passes it to the multimedia group management service application (MGMS) utility. The MGMS uses the addresses invokes "mpeg_snk" application to contact the VoD server for the purpose of joining the desired media channels at service points "A" for a video data channel, while the "ctrl_panel" application is used for the video control channel, at service point B. The protocol for establishing communication is as described in Chapter 4 (Section 4.1.3.5).

For a Lion King "session" (shown in the lower part of Figure 6.4), the service access points A and B are implicit to the session channel, which comprise the "MPEG" video data channel, and the "control" channel. When the user joins onto the "session" channel (as a sink of the channel), the two channel are automatically shown to him or her, including how to participate in them, i.e. as a sink for the "MPEG" video data channel, and as a source for the "control" channel.

When a user decides to add the new media channels to the existing list, the display is as shown in Figure 6.5 (c). Otherwise temporary windows are opened for the new applications and are discarded at the end of the service.



Channel Name	Media Type	Src	Snk
Lion King play (A)	MPEG	—	✓
Lion King control (B]	control	✓	—
---	---	—	—

Session Application

Lion King Session	session	—	✓
A	MPEG	—	✓
B	control	✓	—

MGMS utility

Figure 6.4: A VoD service configuration

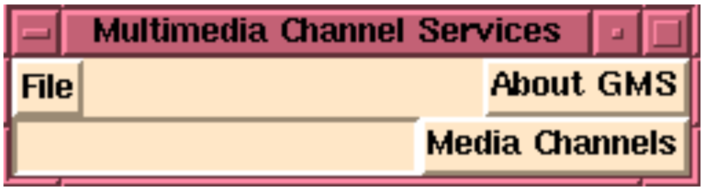
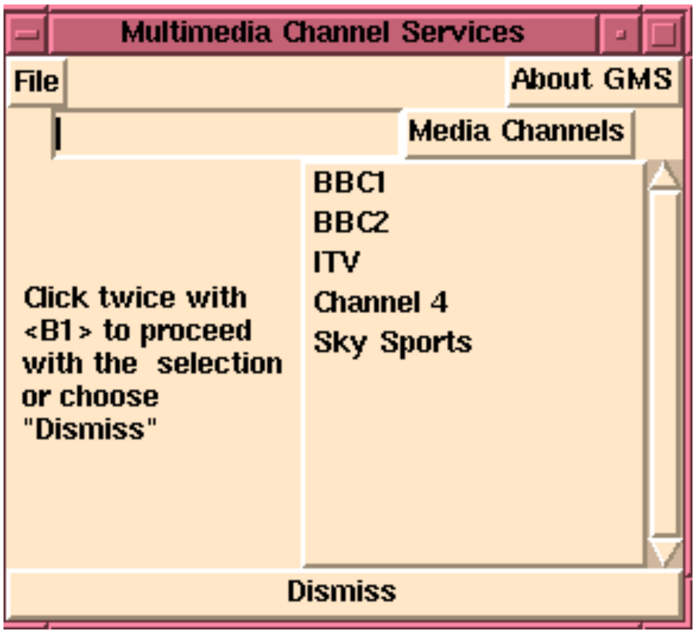
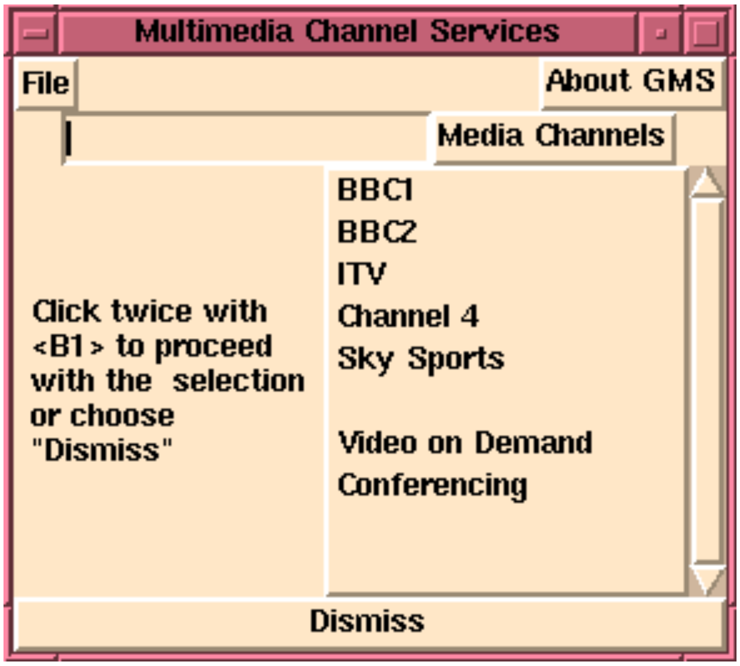


Figure 6.5(a): Main menu



Click twice with <B1> to proceed with the selection or choose "Dismiss"

Figure 6.5(b): Display for selection "Media Channels"

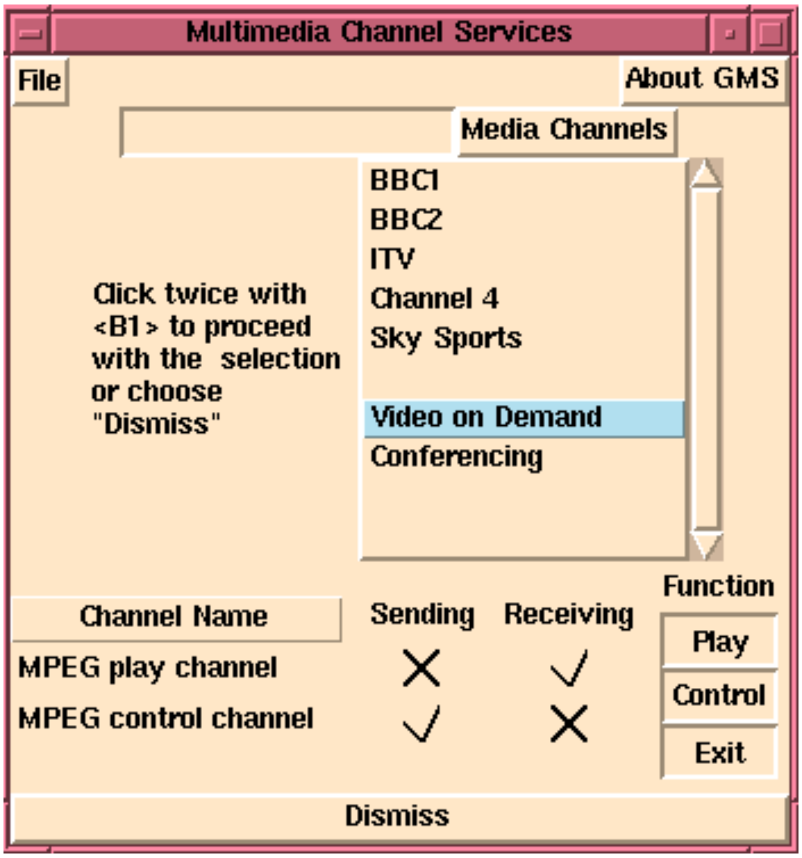


Click twice with <B1> to proceed with the selection or choose "Dismiss"

BBC1
BBC2
ITV
Channel 4
Sky Sports

Video on Demand
Conferencing

Figure 6.5(c): Display for selection VoD media channel



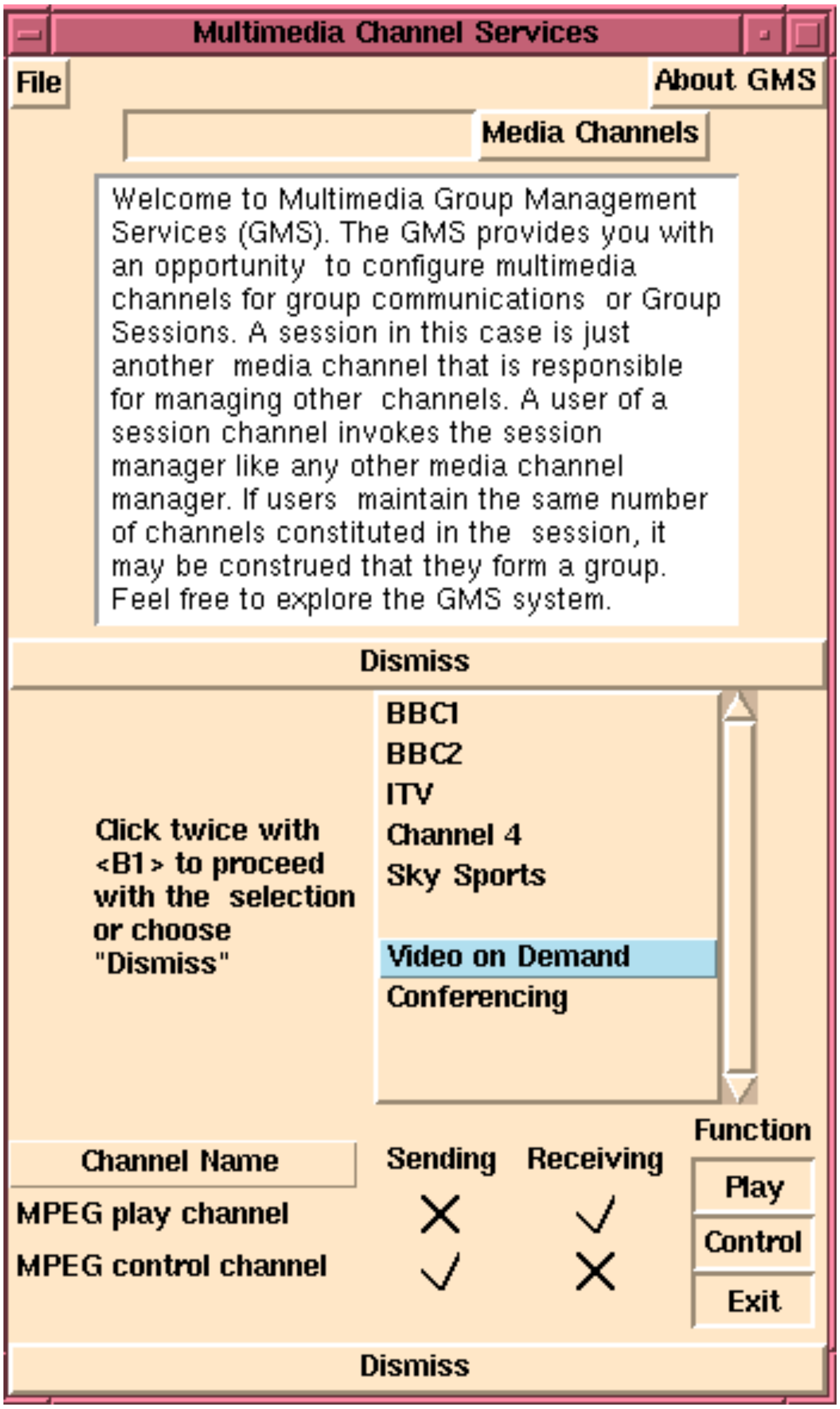
Click twice with <B1> to proceed with the selection or choose "Dismiss"

BBC1
BBC2
ITV
Channel 4
Sky Sports

Video on Demand
Conferencing

Channel Name	Sending	Receiving
MPEG play channel	✗	✓
MPEG control channel	✓	✗

Figure 6.5(d): User Agent configurations for media types



Welcome to Multimedia Group Management Services (GMS). The GMS provides you with an opportunity to configure multimedia channels for group communications or Group Sessions. A session in this case is just another media channel that is responsible for managing other channels. A user of a session channel invokes the session manager like any other media channel manager. If users maintain the same number of channels constituted in the session, it may be construed that they form a group. Feel free to explore the GMS system.

Click twice with <B1> to proceed with the selection or choose "Dismiss"

Channel Name	Sending	Receiving
MPEG play channel	✗	✓
MPEG control channel	✓	✗

Figure 6.5(e): Information window for "About GMS" system

The display in Figure 6.5 (d) shows the information that the user agent needs in order to participate in a channel, i.e. as a source or sink or both. However, this is only for demonstration as the information available to the user is the display of the media channel name (and not necessarily the media type and how to participate in the media channel), and the control panel (play, control, etc.). In real applications, users of the MGMS system would be provided with information about the system. A demonstration of this arrangement is shown in Figure 6.5(e) in the dialogue box.

In the case of forming sessions using the MGMS system, we can envisage some configuration as follows:

An application configures sessions with, say, media channels A, B, C, D. Suppose these sessions are designated S_1(A, B, C, D), S_2(A, B, C, D), S_3(✗).

S_1 may be regarded as a session where group members participate in the media channels as senders and receivers. Therefore anyone joining S_1 automatically joins on all the channels as a source and a sink of each channel.

S_2 would be a channel for passive participants in which case a user joins the session as only a sink for each of the channels.

S_3 is a channel that users selectively participate in each of the channels as they wish.

Table 6.3 shows the configuration for the above scenario. This approach makes it convenient to implement sub-groups or sub-sessions and to some extent constrains all the possible configurations of the sub-groups or sub-sessions.

Sessions:	S_1		S_2		S_3 **(active on select)**	
Channels	Src	Snk	Src	Snk	Src	Snk
A	✓	✓	✗	✓	✗	✗
B	✓	✓	✗	✓	✗	✗
C	✓	✓	✗	✓	✗	✗
D	✓	✓	✗	✓	✗	✗

Table 6.3: Media channel configurations for group sessions

6.3 Application Programmer

An application programmer writes a new distributed multimedia group application by identifying new and existing media types that are involved in the application. The programmer defines a new media type by specifying the formats of the state and the messages indicating how the state changes (**Δ-state**), and *QoS* levels of the new media type, and by *writing* appropriate source and sink programs that conform to these formats. For example consider a Video on Demand (VoD) application which the programmer requires to configure. The information entailed in this configuration would be as follows.

Application: Video on Demand
Types: "MPEG" Video, Sequence Control

"MPEG" media type

state: a single video frame
Δ-state: an MPEG-1 I, P or B frame
QoS levels: q-level(1-31), colour(boolean), rate(+ve real)

Sequence Control

state: frame position, frame rate (including direction)
delta-state: stop, reset, play, etc.
QoS levels: none

Programs for customer's use:

mpeg_sink (MPEG Video sink)

video_control (Sequence Control source)

vod (client program to access VoD server)

Programs for service-provider's use:

video_source (MPEG Video source, Sequence Control sink)

vod_server (server program)

6.4 Experiment: To Demonstrate The Media Channel Model Performance

6.4.1 MPEG Player

Figure 6.6 shows a display of an MPEG player for two sources of MPEG files. The arrows indicate the direction of media streams from sources to the sinks. Each sink receives a copy of the source stream. If now a third source is added, the media channel manager instructs each source to send to the three sinks. Each sink would then display three images from the three sources. On the other hand, if a sink joins the media channel, the media channel manager instructs it to prepare to receive media streams from the number of sources already in the session, while at the same time it instructs each source in the session to send to the new sink. Thus the media channel manager behaves in accordance with the state described in *Section 4.1.3.5.*

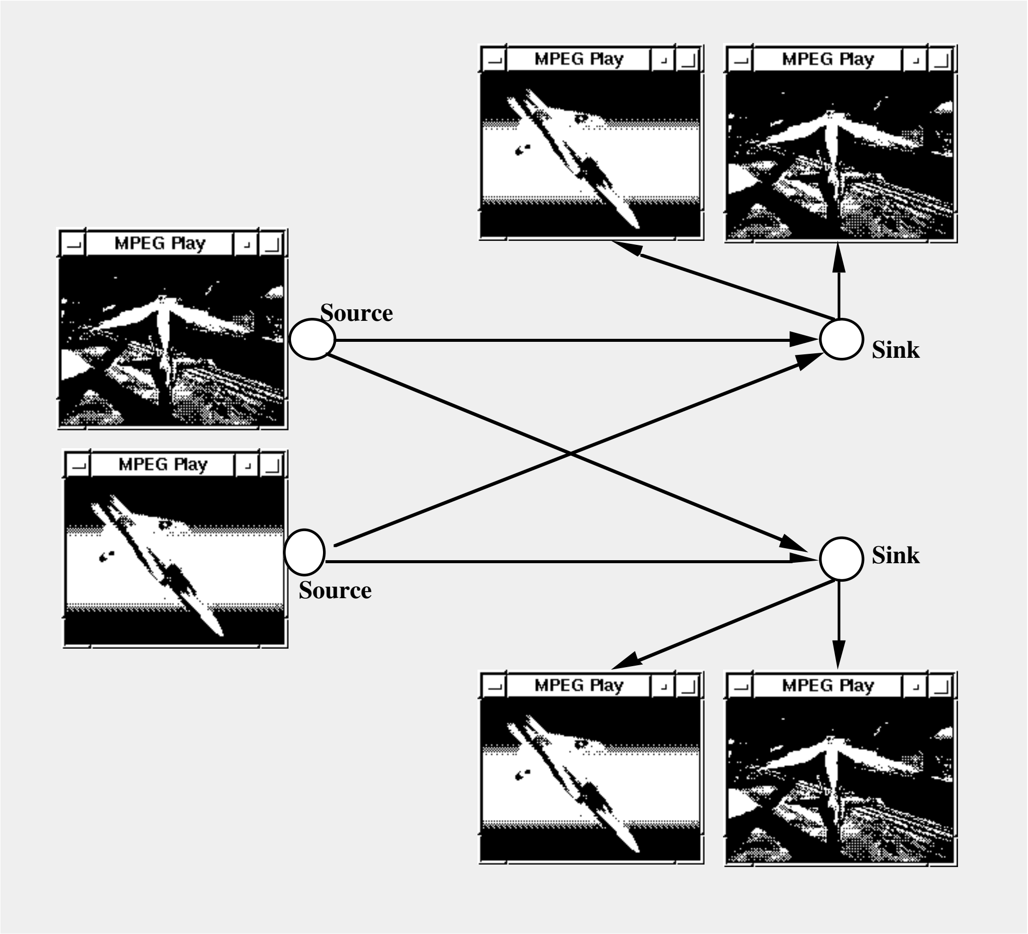


Figure 6.4: MPEG player

6.4.2 Audio/Video Conferencing

The main components of the system set up are shown in Figure 6.5. The microphone and the camera attached to a computer are processed as two separate media channels.

6.4.2.1 MPEG Stream

The video stream of MPEG-1 type is first captured by the camera, then compressed, encoded and sent as a series of frames to remote sink destinations. For simplicity, Figure *6.5* shows the direction of flow of the MPEG stream. The manager maintains group membership lists and is responsible for controlling and monitoring the various *joins* and *leaves* on the media channel.

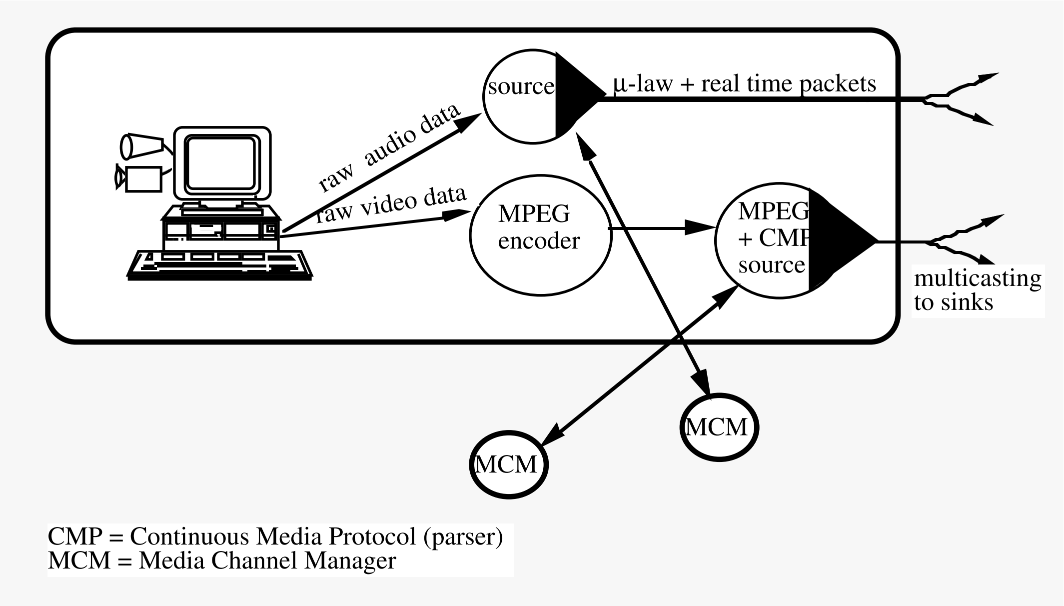


Figure 6.5: Media channel configuration for MPEG and audio sources

The basic mechanisms of encoding and transmitting the MPEG stream are shown in *Figure 6.6*. The process of encoding is done by using a SunVideo XIL library, i.e. Compression Image Sequence (CIS) which is then parsed at the source and transmitted as a (naive) multicast stream.

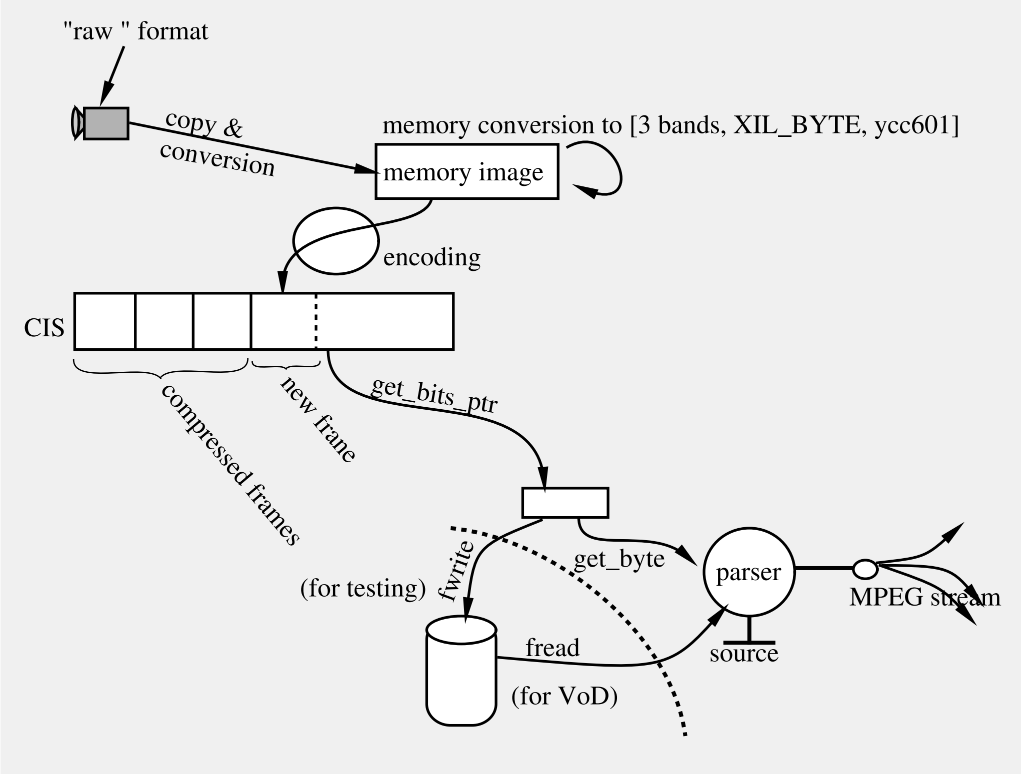


Figure 6.6: Encoding MPEG stream

To investigate the performance of the media channel model, a conference application consisting of three units of audio/video capabilities was setup. The logical connections of the three units is shown in Figure 6.8. Other participants who do not have the audio/video capability are free to join the *text* channel of the conference which can be accessed by their UAs.

6.4.2.2 Audio Stream

The audio stream is captured by */dev/audio* input of the microphone and sent directly as a sequence of bytes to the remote sink destinations. However, for interactive sessions, the difficulty lies in trying to mix audio streams at the output (speaker) through */dev/audio* since it (*/dev/audio*) only handles one channel. The solution is to buffer the input signals at the destination and mix them at a constant rate to the output (*/dev/audio*). This is shown in Figure 6.7.

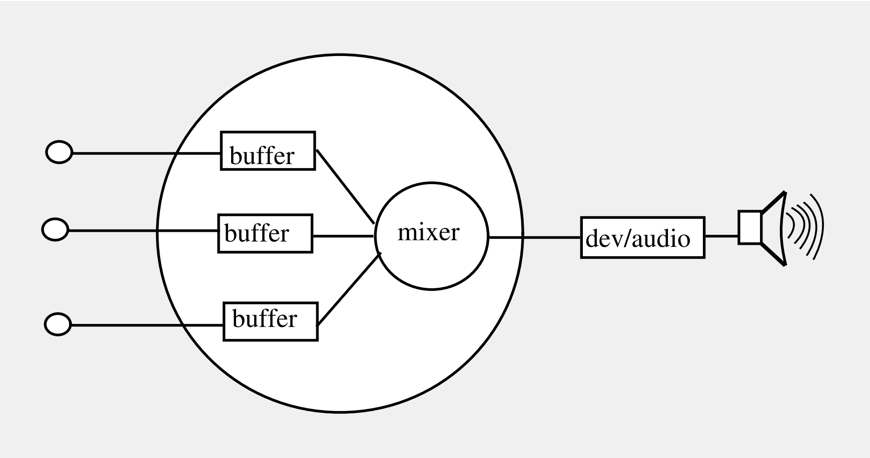


Figure 6.7: Data acquisition system for audio sink

The */dev/audio* system used encodes audio in a **μ-law** format which is algorithmic and expressed as:

$$y = \frac{\operatorname{sgn}(m)}{\ln(1+\mu)} \ln\left[1 + \mu\left|\frac{m}{m_p}\right|\right] \qquad (2)$$

where m is the instantaneous amplitude of the audio signal and sgn (m) is the sign of m, y is the instantaneous **μ-law** value.

m_p is the peak value of the signal,

μ is a constant that takes values of 100 and 255

Since mixing audio signals implies adding their amplitudes (in this case), equation (2) should be expressed in the variable m to be able to add the linear values of m. The transformation of (2) yields the following expression:

$$|m| = \frac{m_p}{u}\left[(u+1)^y + 1\right] \qquad (3)$$

where $|m|$ is the linear amplitude. When values of $|m|$ are mixed, they must be averaged over the total number of sources destined to a sink. Suppose a $|m|$ = x and n is at the start the number of audio streams to the sink. The average $\bar{x}$ is simply given as:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (4)$$

Now if a new source joins the conference, and n' and $\bar{x}'$ are the new states, then

$$\bar{x}' = \frac{\bar{x}n + x_{n'}}{n'} \text{ where } n' = n+1 \qquad (5)$$

However, $\bar{x}$ (or $\bar{x}'$) must be converted back to the **μ-law** format for */dev/audio* to recognise it for the output.

6.4.2.3 Starting a Conference

In Figure 6.*8* agents of participants on machines A, B and C interact using audio and video channels. Since the participants can talk to each other, they use a human protocol to control access audio channel.

Guests are welcome to join the conference session as passive participants provided they have the capability to receive the streams from the media channels. In this particular setup, only the participant on machine A is a producer of both video and audio streams. The other participants produce only audio but consume all the media streams from machine A. A screen dump of a display of a participant on machine A to another participants' screen is shown in Figure 6.9. In this figure, two researchers (the author on the right hand side) are engaged in a discussion in the Computing Department's Sky Lab, while being watched by their colleagues from their office terminals (Linux PC machines).

In a less sophisticated scenario, i.e. one that does not involve the session manager, an individual may set up (or obtains) a media channel and give his address to potential participants. Depending on the application, the potential participants join the channel as source or sink or both. However, if the group is large, the session manager configures the scenario for the application automatically. That is the functionality of the session media channel.

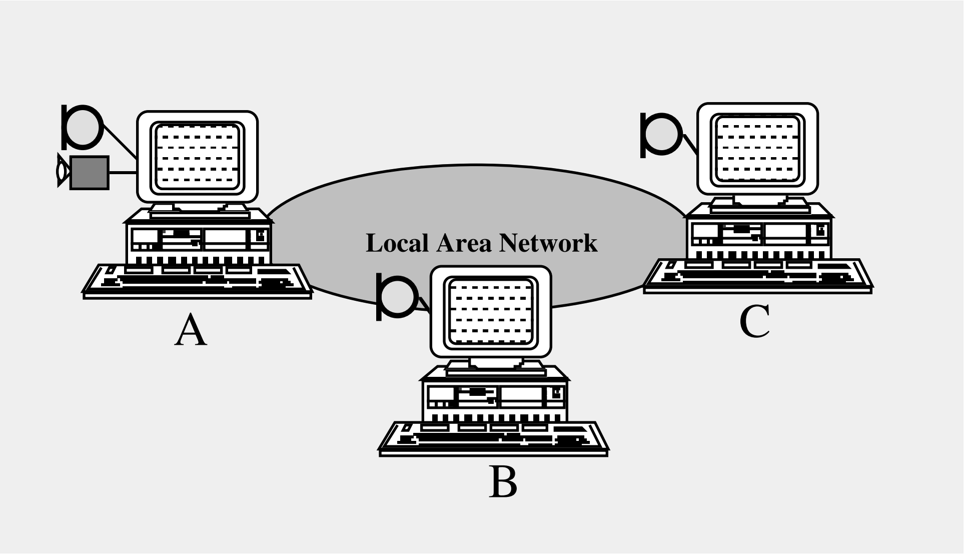


Figure 6.8: Conference Setup

Figure 6.9: Image from machine A
(Figure 6.8) to other participants

6.4.3 Performance

The performance of the source/sink interaction is examined with respect to scalability. As the number of sinks increases, the number of messages exchanged between the media channel manager and sources and sinks increases. For example, a source that joins a media channel is instructed by the manager to send to sinks currently present in the channel. Conversely, a sink that joins the media channel is instructed to prepare to receive from several sources that have already joined the media channel. An investigation of how the media channel manager performs its functions in effecting the source/sink interaction was done as follows.

6.4.3.1 Source Performance and Evaluation

A single source was made to transmit to a number of sinks running X-windows on a Linux machine. The number of sinks was progressively increased from one. With the current number of sinks joined onto the media channel a source was started, and the (start) time was recorded. When a response was received from the manager, the time was printed out by the sink program and the difference which corresponded to the time the media channel manager instructed the source to send to the current number of sinks. The source did not know how many sinks there were. As messages arrived one after another, corresponding times were recorded (printed to a file), at the end of which the source was stopped when no more print outs were being recorded It was observed that the Linux machine would handle the processing of up to 17 sinks effectively before going into error. The experiment was repeated 5 times. The average times for each set of sinks are plotted in Figure 6.10.

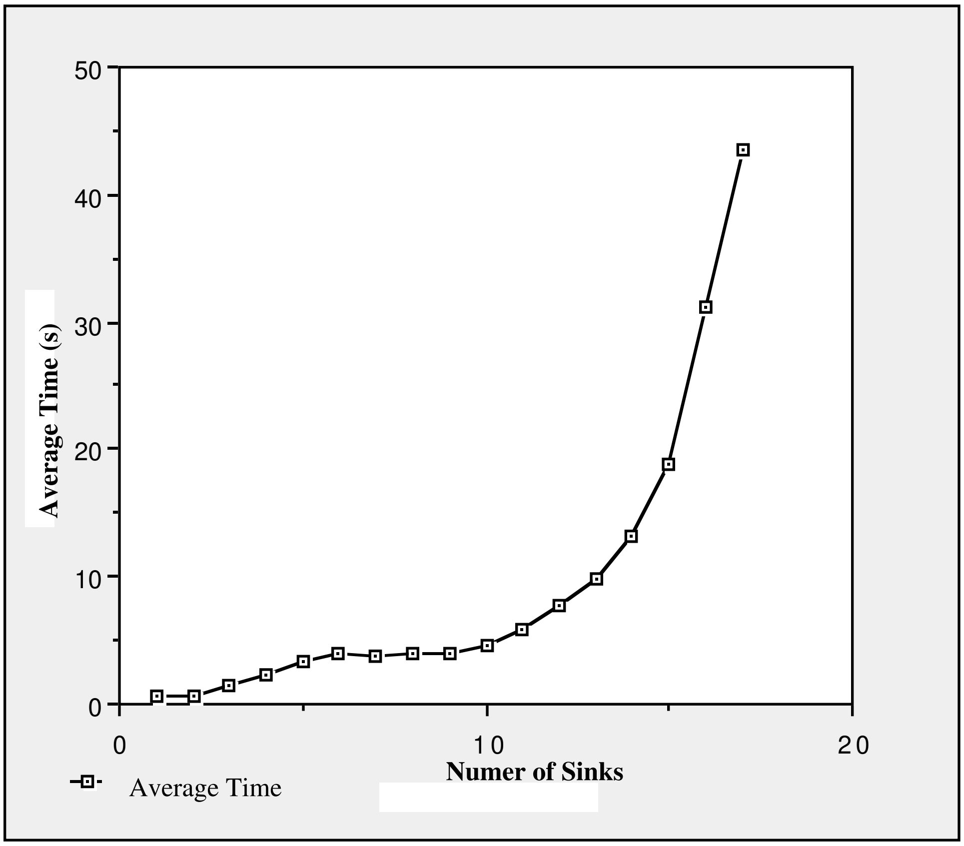


Figure 6.10: Response times for media channel manager with increasing number of sinks

From Figure 10, it can be seen that the response times for the media channel manager to a source to send to the current number of sinks that have joined the media channel increases rapidly after 10 sinks. In practice, different machines would have different processing capabilities; the fast the processing capability, the shorter the response times expected.

6.5 Discussion

It should be noted that whenever a media channel is created, the motivation is to provide the facility for potential users to join the channel; hence the need to have the media channel manager. In a simple protocol such as one-to-one communication, there is no requirement for a media channel manager. However, a one-to-one communication does not make efficient use of communication resources since each connection is managed individually.

Each media channel is managed by the MCM, and a session, being just one more media channel, requires a session manager. The Membership of the channels to the session provides the facility to announce to the members of the session channel that new sessions have become available.

6.5.1 Using the Media Channel Model for Media Conferencing

In the current implementation, there is a "source" host for audio/video streams. Only video sinks can be connected and audio is derived directly from the */dev/audio* device on the host machine. We do not have the synchronisation mechanisms for the audio/video but do provide the bind indication where such a need may arise to enable the underlying communication facility to do the low-level synchronisation. This is done by providing a media Id or a signature for the related media channels (types), for example:

audio/video

audio/text

video/audio/text

etc.

Synchronisation is very important in group sessions. As noted by [Lindsey 95]. a session medium which supports human interactions with a number of media channels should be grouped together and managed. The media channel model provides for this requirement. However, more work is required in the grouping process to define the flows and interfaces that constitute the session channel. This was indicated in *Section 6.3* for applications programmers. This information is used by the supporting communication system to allocate resources on instantiation of the media channels (session), and to effect the manipulation of separate media streams (possibly in real time).

6.5.2 Distributed Group Management Services

The purpose of distributed management services is to enable global interactions of distributed multimedia applications, and also to make services available to local domain. Thus scalability of group management services for distributed multimedia applications is enhanced by distribution of management services. However, this requires that a global name (address) is known for a given media channel with respect to each manager of a local domain. Each manager has a unique address, and is responsible for the local domain. For example we may have distributed management services that involve three managers as shown in Figure 6.11.

There is an increasing quest for developing infrastructures for distributed multimedia management services. For example, in the telecommunications industry, BT [Marshall 95] has proposed an Information Services Supermarket (ISS) scheme; depicted in Figure 6.12 [Hutchison 95]. The aim of ISS is to envision the integration of telecommunications services with those of computing and to enable a third party service provider to offer interactive multimedia services to diverse terminals over a Broadband network. In this respect multimedia group management services would ease access and use of the services from the retail domain, i.e. interactive and distributed services, as has been discussed earlier in the thesis.

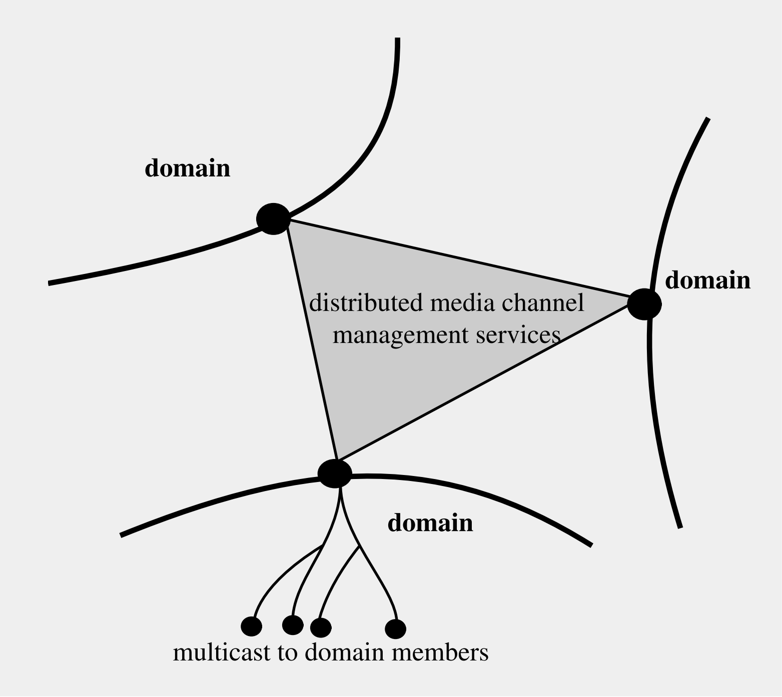


Figure 6.11: Distributed media channel services

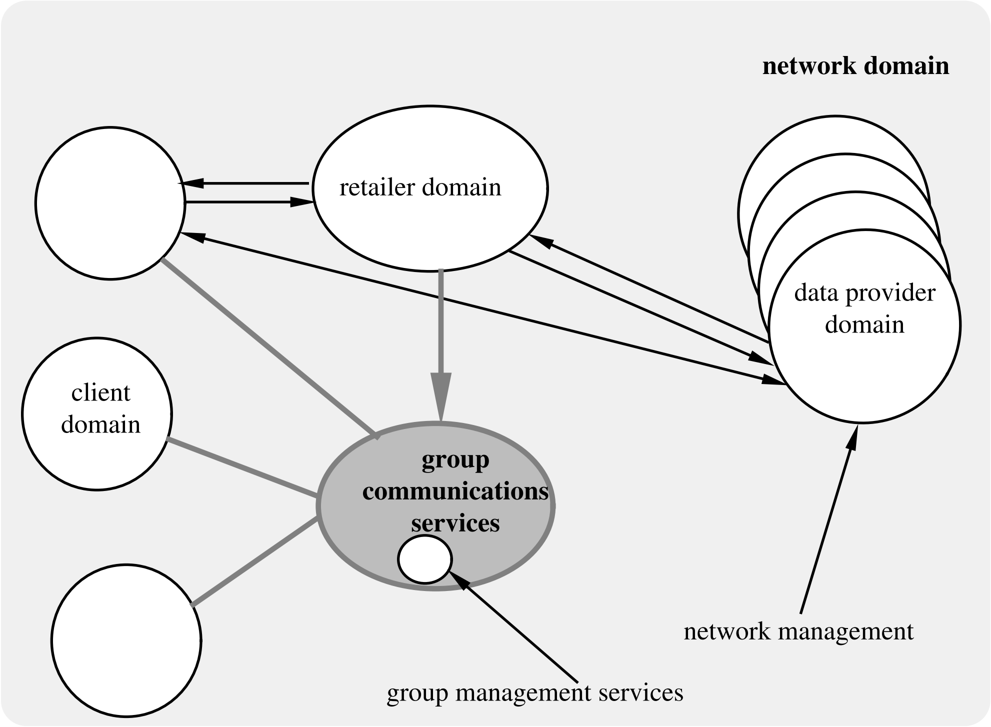


Figure 6.12: Integrating group communication services

6.5.3 Communications Support

Currently the media channel model is implemented using a naive multicast, i.e. point-to-point multicast. It is essential that the communication infrastructure provides multicast facility for group addressing. This was outlined in *Section 5.3.1.1* on the need for multicast support for group addressing. However, it is assumed that the person who configures a media channel manager will have knowledge about the communication infrastructure and hence will configure the media channel accordingly.

Chapter 7

Conclusion

This thesis has discussed issues related to multimedia group communication services as one of the major requirements of distributed multimedia group communications, in addition to transport and synchronisation services. The theme carried through the discussion has been on how to provide flexible multimedia group management services.

In this concluding chapter, a summary of contributions of the thesis is presented. Since some important additional issues relating to multimedia group management services need to be investigated, suggestions on further work are outlined.

7.1 Summary of the Thesis

In the introduction of this thesis, the motivation for research in group management services was outlined. However, in order bring into focus the state of the art equating to the need for multimedia group management services, a review of group communication and group based activities was first presented. It was observed that the models hitherto presented for group communications were not suitable for multimedia group applications. With this contention, the objective was to seek a suitable architecture to model multimedia group communications. The architecture would be based on the requirements of multimedia applications. These requirements were focused on multimedia group management services as the theme of the thesis, since, as it turned out, multimedia group communications require diverse sets of services among which are transport and synchronisation services.

The requirements for multimedia group management services, their functions and mechanisms, and how they could be applied to support multimedia communications in distributed computing, were investigated. Essentially, the underlying factors were that the current distributed multi-user group applications manage their own connections to effect group communication. This application dependent approach renders collaborative applications unable to collaborate amongst themselves. Besides, the management strategies employed were too restrictive with no flexibility for forming sub-sessions.

Furthermore, it was observed that if several applications of different types are managed as a single multimedia application, the addition of another media type presents a problem. In addition, the users of multimedia streams also need the flexibility to suspend and resume various media streams as well as manipulating individual media types.

The above requirements led to the formulation of a *media channel model* with an architecture for multimedia group management services, based on four functional objects: the user agent, the media channel manager, source and sink. When implemented, these objects provided the user with a set of library functions which can be configured for application-specific needs. This was one of the important contributions of the research and is outlined among the main contributions of the thesis in the following section.

7.2 Main Contributions of the Thesis

This thesis has detailed the potential for using media channels to model multimedia group management services Having observed that most of the models developed for group management services for multi-user multimedia are applications specific or are too restrictive, the media channel model shows a *novel approach* to multimedia group management services. The strategy used was to define the behavioural rules of media channels in general (i.e. non-media-type-specific), allowing the further development of call-management applications which handle multiple media types. Also, by allowing the media channel model to run as a stand-alone architecture (i.e. not relying on a call-management application), applications are free to form parts of multiple calls or to be used independently. Writing new distributed group applications now involves defining new media types.

The advantages of the media channel model are that:

1) It presents a user agent module that dynamically configures distributed media types for applications
2) It enables a session to be configured as just another media type.
3) No formal sessions are defined. Sessions are simply a convenient way of managing distributed media channels for multimedia based applications and they manifest themselves through the user agent as session media channels. By abandoning formal definitions of sessions, it is easier to define sub-sessions because the media channel are not exclusively associated with a single application any more.
4) The interface defined between the media channel manager and the

source, sink and user agent is flexible to include possible multicast mechanisms. In this way a uniform appearance to all group applications in their collective connection management is encouraged. In addition:

- while the definition of interaction between media channel manager and the source/sink management addresses is dependent on the media channel (e.g. MPEG manager), the mechanisms that support the communication are generic to media types, and
- the mechanisms supporting media sources (e.g. MPEG source) and media sinks (e.g. MPEG player) are media type specific while being generic to media channels.

5) The architecture presented is open and therefore very flexible: only the protocol for participating is defined - the permissions and constraints are left to the programmers of the channel-management entities.

In view of the above characteristics of the media channel model, it can be claimed that the media channel model allows both users and application programmers the flexibility to use the available multimedia resources by providing generic services i.e. the infrastructure for multimedia group management services.

7.3 General Comments

This thesis has provided a framework for multimedia group management services. However, it should be noted that the ways in which the services are implemented may vary, depending on the kind of support facilities available, especially low-level communication support such as multicast services.

The Group Management Services for the support of distributed multimedia applications were among the requirements identified in the GCommS project besides the transport and synchronisation services. As stated before, the GcommS work is about defining services for multipeer connections and the management of these connetions, for the support of QoS in heterogeneous environment. In the media channel model, each consumer may, independently of the others, ask to receive a subset of the data by issuing QoS requests in the opposite direction of the stream. The filtering mechanisms that have been implemented in the GcommS project may be used to support different QoS for different consumers on a multicast stream.

Reliability for communications by the consumers and producers with the media channel manager is highly desired. The issues on reliability are being addressed in the GCommS project.

7.4 Further Work

The issues discussed in the thesis have led to identification of further research needs. These issues extend to optimising the implementation of the media channel model, while others are big research issues. These are briefly outlined as follows:

7.4.1 Further Implementation Work

7.4.1.1 Integrated Source-Sink Program

At the moment, the user agent cannot cope with applications that join to the media channel as both source and sink. Currently the source and sink run as two separate programs. These need to be integrated into one program.

7.4.1.2 Reliability

Although messages that are transmitted between the media channel manager and other media channel objects are control messages, there is a need to ascertain the reliability of the messages. Currently the reliability feature does not exist in the implementation.

7.4.1.3 Incorporating QoS re-negotiation

When a sink fails to receive its requested QoS, it may indicate this to the manager by deregistering. If the policy is that there must be a minimum number of participants, then the manager could use the sink's deregistration to terminate all the other connections. However, this may be confused with the sink's user not wishing to participate. It may be helpful if we redefine the manager's view of the sink's state from "participating" / "not participating" to "participating - QoS okay" / "not participating" / "participating - QoS failed", to allow the manager to update the sources' and sinks' lists appropriately.

7.4.1.4 Operating systems support

In order to realise the full potential of the media channel model for multimedia group based applications, especially those that have real-time characteristics, there is a need to cope with the real-time support of these applications. Lancaster University has been involved in research to enhance the Chorus operating system to support real-time transmission and synchronisation of media types. Conclusive results of the research are yet to be reported but so far the work done is promising enough for the exploration of further work on MGMS.

7.4.2 Research Work

7.4.2.1 Distributed Multimedia Group Management Services

The current implementation of the media channel manager has been realised as a centralised object. In *Section 6.5.2* it was clearly shown that there is a need to implement a distributed system for media channel management. At the time of writing, an investigation of ODP is planned with the objective of designing distributed media channel management schemes. This will be based on the CORBA standard.

7.4.2.2 Inter-channel synchronisation

The media channel model does not address inter-channel synchronisation. Since the user agents deal with sets of related media channels during a session. It should be possible to define other operations via the media channel managers for the channels to be synchronised. This may be done by defining another media type appropriate to address inter-channel synchronisation at this level

Finally, it should be noted that the subject of Multimedia Group Management Services is at an early stage: this thesis has made a contribution towards an open architecture and services that are and may be appropriate to forthcoming group-based applications. It is anticipated that many other interesting research topics will emerge as the subject matures.

Cited References

[AFNOR 94] AFNOR, "Comments on Document WG4/N 833 (4K64R1) and proposed new Draft Taxonomy", *AFNOR - Association Francaise de Normalisation*, Comments, January 1994.

[Ahuja 92] S. R. Ahuja and J. R. Ensor, "Coordination and Control of Multimedia Conferencing", *IEEE Communications Magazine*, vol.30 (5), 1992, pp 38-43.

[Almeroth 95] K. C. Almeroth and M. H. Ammar, "Characterisation of Mbone Session Dynamics:", *Ceorgia Institute of Technology, Networking and Telecommunications Group*, Technical Report, GIT-CC-95-45, June 9, 1995.

[ECFF 92] ANSI(ECFF), "Information processing systems - Telecommunications and information exchange between systems - Guidelines for Enhanced Transport Mechanisms", *ANSI (USA)*,Project, ISO/IEC JTC1/SC 6 N, 01/12/1992.

[Auerbach 91] J. Auerbach, M. Gopal, M. Kaplan and S. Kutten, "Multicast group membership management in high-speed wide area networks", *In Proc. of 11th International conference on Distributed Computing Systems*, Arlington, TX, USA, 1991, pp 231-8.

[Bannon 90] T. Bannon and I. Page, "group: A Distributed Group Specification and Management Service", in *UNIX - The legend Evolves*", *Proc. of UKUUG conference*, Buntingford, UK, 1990, UKUUG, pp 61-76.

[Bauer 94] M. A. Bauer and e. al, "A distributed system architecture for a distributed application environment", *IBM Systems Journal*, 33 (3), 1994, pp 399-425.

[Benford 92] S. Benford, M. Turoff and J. Palme, "An ISO standard to support asynchronous group communication", *Computer Standards & Interfaces*, 14, 1992, pp 363-373.

[Birman 89] K. Birman and a. K. Marzullo, "ISIS and the META Project", *Sun Technology*, (Summer), 1989, pp 90 -104.

[Booch 1994] G. Booch; "Object-Oriented Analysis and Design with Applications", The Benjamin/Cummings Publishing Company Inc, Redwood City, California 94065, 1994 pages, Second Edition.

[Campbell 93] A. Campbell, G. Coulson and D. Hutchison, "A Multimedia Enhanced Discussion on ECFF", *ISO/ICE*, Standards, ISO/IEC JTC1/SC6/WG4 N832, May 1993.

[Chen 1991] M. Chen and Y. I. Liou, "The Integrated Group Support Environment", *In Proc. of International Conference on Systems Sciences*, Kauai, HI, USA, 1991, V. S. Milutinovic B.D, IEEE Computer Soc. Press, pp 333-342.

[Cheriton 85] D. R. Cheriton and a. W. Zwaenepoel, "Distributed Process group in V Kernel", *ACM Transactions on Computer Systems*, 3 (2), 1985, pp 77-107.

[Cocquet 95a] P. Cocquet, "Draft Multi-peer Taxonomy", *ISO* Recommendations, ISO/IEC JTC1/SC6N, 7-3-1995.

[Cocquet 95b] P. Cocquet, "Multi-peer Transport Services", *ITU-T X-TMS Projects - SC6/WG4*, Request For comments -(ECFF), 25-07-1995.

[Coffield 87] D. T. Coffield, "Network and Distributed Systems Management: *Issues, Architectures and Implementation*", *Ph.D Thesis*, University of Lancaster, 1987.

[Conti 92] G. Conti and J. R. Ensor, "Requirements for multiparty data communications protocol", *in Proc. of 1992 International Zurich Seminar on Digital Communications. Intelligent Networks and their Applications.*, Zurich, Switzerland, 1992, C. Petitpierre and Schlegel, IEEE, pp 367-373.

[Coulouris 90] G. Coulouris, J. Dollimore and T. Kindberg; "Distributed Systems: Concepts and Design", Addison-Wesley Publishers, Cornwall, 1994 644 pages, Second Edition.

[Coulson 92] G. Coulson, J. Smalley and G. S. Blair, "The Design and Implementation of a Group Invocation Facility in ANSA", *Lancaster University*, Technical Report, MPG-92-34.

[Davies 94] N. Davies, "An Object Management System to Support Distributed Multimedia Design Environments", *Ph.D Thesis*, Lancaster University, 1994.

[Deering 89] S. Deering, "Host Extensions for IP Multicasting", *Stanford University*, Request for Comments, RFC 1112, August 1989.

[Daiz 93] M. Diaz and T. Villemur, "Membership services and protocols for cooperative frameworks", *Computer Communication*, 16 (9), 1993, pp 548-556.

[Dollimore 92] J. Dollimore and G. Coulouris, "The Relevance of Object groups and Multicast in Shared Distributed Object Systems", *In Proc. of Fifth ACM SIGOPS European Workshop*, Le Mont Saint-Michel, France, 1992.

[EEC 95] C. EEC, "MERCI Proposal for B2", *EEC*, March 14th, 1995.

[Erickson 94] H. Erikson, "MBONE: The Multicast Backbone", *Communication of the ACM*, Vol. 37 (No. 8), 1994, pp 54-60.

[Fluckiger 95] F. Fluckiger; "Understanding Networked Multimedia: Applications and Technology", Prentice Hall, 1995, 620 pages.

[Fortier 88] P. J. Fortier; "Design of Distributed Operating Systems, Concepts and Technology", McGraw-Hill International Editions, Singapore, 1988 pages.

[Garcia 93] F. Garcia, "Continuous Media Transport and Orchestration Services", *Ph.D Thesis*, Lancaster University, 1993.

[Garcia 96] F. Garcia, D. Hutchison, A. Mauthe and N. Yeadon, "QoS support for distributed multimedia communications", *in Proc. of IFIP/IEEE International Conference on Distributed Platforms*, Dresden, Germany, 1996.

[Grudin 92] J. Grudin and S. Poltrock, "CSCW and Groupware: Overview of Behavioural Issues and Survey of Systems", *in Proc. of ACM 1992 Conference on Computer-Supported Cooperative Work*, Toronto, 1992.

[Gupta 94] A. Gupta and D. Ferrari, "Resource partitioning for multi-party real-time communication", *International Computer Science Institute*, Technical Report, TR-94-061, November, 1994.

[Gupta 93] A. Gupta and M. Moran, "Channel Groups", *International computer Science Institute*, Technical Report, TR-93-015, March, 1993.

[Hagsand 91] O. Hagsand, "A Multi User Draw Editor", *in Proc. of MutiG workshop*, Stockholm, Sweden, 1991.

[Handley 96] M. Handley, J. Crowcroft and C. Bormann, "The Internet Multimedia Conferencing Architecture", *University College London*, INTERNET-DRAFT -Internet Engineering Task Force, 22 Feb. 1996.

[Handley 92] M. Handley and S. Wilber, "Multimedia Conferencing: from prototype to national pilot", *in Proc. of Proc. INET'92, International Networking Conference*, Kobe, Japan, 1992, H. Ishida, Internet Society, Reston, Virginia, pp 483-490.

[Handley 93] M. J. Handley, P. T. Kirstein and M. A. Sasse, "Multimedia Integrated Conferencing For European researchers (MICE): piloting activities and the conference management and multiplexing centre", *Computer Networks and ISDN Systems*, 26 1993, pp 275-290.

[Hartley 95] S. Hartley, "A Database System Approach For the support Group Communications", *Lancaster University*, 3rd Year Project Report, May 1995.

[Heijenk 94] G. J. Heijenk, X. Hou and I. G. Niemegeers, "Communication Systems Supporting Multimedia Multi-user Applications", *IEEE Network*, (January/February), 1994, pp 34 -44.

[Heinrichs 93] B. Heinrichs, K. Jakobs, K. Lenssen, W. Reinhardt and A. Spinner, "EuroBridge: communications for multimedia applications", *Electronics & Communication Engineering Journal*, 5 (1), 1993, pp 45-50.

[Henckel 93] L. Henckel, "Multimedia Communication Platform: Specification of the Enhanced Broadband Transport Service", *GMD FOKUS*, Deliverable, R2060/GMD/CIO/DS/P/002/B2, September 10, 1993.

[Herbert 94] A. Herbert, "An ANSA Overview", *IEEE Network*, (January/February), 1994, pp 18 - 23.

[Herrtwich 92] R. G. Herrtwich and R. Steinmetz, "Towards Integrated Multimedia Systems: Why and How", *European Networking Center*, Technical Report, Number 43.9101.

[**Holfender 96]** Mieland Holfender "Mbone VCR on Demand Service", *Seminar*, March, 1996.

[Horn 1990] D. N. Horn and V. M. Show, "An ISDN Multimedia conference Bridge", *in Proc. of IEEE TENCO'90:1990 IEEE Region 10 Conference on Computer Communication Systems*, Hong Kong, 1990, IEEE, pp 853-6.

[Hughes 87] L. Hughes, "A Multicast Transmission Taxonomy", *in Proc. of Distributed Processing. Proceedings of the IFIP WWG 10.3 Working Conference*, Amsterdam, Netherlands, 1987, M. H. Barton, E. L. Dogless and G. L. Reijns, North-Holland, pp 87-102.

[Hutchison 95] D. Hutchison "Information Services *Supermarket*"; *Personal Communication*, Dec. 1995.

[Hutchison 93] D. Hutchison, D. Shepherd and G. Coulson, "Functions and Facilities for the Support of Group Applications in a Distributed Multimedia Environment", *Lancaster University*, Research Proposal, March, 1993.

[Isis 92] Isis; "Isis Distributed Tool kit User Reference Manual", Isis Distributed Systems, Inc., Ithaca, NY 1450, 1992 pp 6.1 - 6.6, Ver. 3.

[ISO/IEC N8693 93] ISO/IEC, "Draft Taxonomy of Multicast operation", *ISO,* ISO/IEC JTC1/SC6/N, 12 October, 1993.

[ISO/IST/6 92] ISO/IST/6, "MPDT Transport Service", *ISO*, Technical Report, Number 92/63706, 2nd June 1992.

[Jain 90] B. N. Jain and A. K. Agwawala; "Open Systems Interconnection: Its Architecture And Protocols", Elsevier Science Publishers B.V, Amsterdam, 1990 pages.

[Kaashoek 91] M. F. Kaashoek and A. S. Tanenbaum, "Group communication in AMOEBA Distributed Operating System", *in Proc. of IEEE 11th Inter. Conf. On Distributed Computer Syst.*, 1991, pp 222-230.

[Kirstein 93] P. T. Kirstein, M. J. Handley and M. A. Sasse, "Piloting of Multimedia Integrated Communication for European Researchers (MICE)", *in Proc. of INET '93*, 1993.

[Kompella 92] V. P. Kompella, J. C. Pasquele and G. C. Polyzos, "Multicasting for multimedia applications", *In Proc. of IEEE INFOCM '92: Conference on Computer Communications*, Florence, Italy, 1992, IEEE, pp 2078-85.

[Kovacs 94] L. Kovacs, "The GroupSpace Concept", *IEEE*, 1994, pp 320-325.

[Langsford 94] A. Langsford and J. D. Moffett; "Distributed Systems Management", Addison-Wesley, Cornwall, UK, 1993, 307 pages,

[Liang 92] L. Liang, S. T. Chanson and G. W. Neufeld, "Process Group Communications: Classifications and Requirements", *Computer*, 1990.

[Lindsey 95] D. Lindsey and P. F. Linnington, "RIVUS: A Stream Template Language for Capturing Multimedia Requirements", *in Proc. of Second International COST 237 Workshop on Teleservices and Multimedia Communications*, Copenhagen, Denmark, 1995, D. Hutchison, H. Christainsen, G. Coulson and A. Danthine, Springer, pp 259-277.

[Linington 94] P. F. Linington, "Multimedia and Streams", *In Proc. of ANSAworks*, 1994, APM Ltd, Poseidon House, Castle Park, Cambridge, UK.

[Marshall 95] I. W. Marshall and M. Bagley, "The information services supermarket -an information network prototype", BT Technology Journal, vol. 13 (2), 1995, pp 132-142.

[Mathy 94b] L. Mathy, O. Bonaventure, g. Leduc and A. Danthine, "The Multipeer Data Transport Service and Group Communication Framework", *University of Liege*, Technical Report, R2060/ULg/CIO/IN/P/008, January, 1994.

[Mathy 94a] L. Mathy, H. Leduc, O. Bonaventure and A. Danthine, "A Framework for Group Communications", *University of Liege*, Technical Report, R2060/ULg/CIO/IN/P/005.

[Mauthe 95b] A. Mauthe, G. Coulson, D. Hutchison and S. Namuye, "Group Support In Multimedia Communications Systems", *In Proc. of The second COST 237 Conference on Teleservices and Multimedia communications*, Copenhagen, Denmark, November 20 - 22, 1995b, pp 1 -18.

[Mauthe 94] A. Mauthe, D. Hutchison, G. Coulson and S. Namuye, "From Requirements to Services: Group Communication Support for Distributed Multimedia System", *In Proc. of Second International Workshop on Advanced Telservices and High-Speed Communication Architectures(IWACA)*, Heidelberg, 1994, R. Steinmetz, Springer-Verlag, 1994, pp 266-277.

[Moran 94] M. Moran and R. Gusella, "System Support for Efficient Dynamically Configurable Multi-party Interactive Multimedia Applications", *Tenet Group and Hewlett-Packard Laboratories*, Technical, 1994.

[Muller 92] G. Muller, "Strategic Directions towards Multimedia Systems.", *Institur fur Informatik und Gesellschaft, Abteilung TELEMATIK, Univesität*,

[Muller 96] N. J. Muller, "Multimedia Over The Network", BYTE, (March), 1996, pp 73 -78.

[Namuye 95b].Namuye, D. Hutchison, A. Mauthe and G. Coulson, "Group Management Services for Distributed Multimedia Applications", *In Proc. of Second Communication Symposium*, Manchester Metropolitan University, United Kingdom, 1995b, pp 40-43.

[Nanaratnam 88] S. Nanaratnam, S. Chanson and G. Neufeld, "Reliable group communication in distributed systems", *In Proc. of The 8th ICDCS*, 1988, pp

[Ngoh 91] L. H. Ngoh, "Multicast support for group communications", *Computer Networks and ISDN Systems*, 22 (3), 1991, pp 165-78.

[Oskiewicz 1990] E. Oskiewicz, J. Warne and M. Oslen, "A Model for Interface Groups", *Advanced Networked Systems Architecture (ANSA)*, ISA Project, 14 September 1990.

[Pasquale 92] J. C. Pasquale, G. C. Polyzos, E. W. Anderson and V. P. Kompella, "The Multimedia Multicast Channel", *In Proc. of 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, 1992, pp 0185-0196.

[Pendergast 95] M. Pendergast and C. S. Hayne, "Alleviating Convergence Problems in Group Support Systems", *Computer Supported Co-operative Work (CSCW)*, 3 (1), 1995, pp 1-27.

[Pendergast 93] M. O. Pendergast, "Multicast Channels for Collaborative Applications: Design and Performance Evaluation", *Computer Communication Review*, 23 (2), 1993.

[Rezende 95b] J. Rezende, A. Mauthe, D. Hitchison and S. Fdida, "M-Connection Service: A Multicast Service for Distributed Multimedia Application", *In Proc. of* 1995, XTP Forum.

[Rodden 92] T. Rodden, J. A. Mariani and G. Blair, "Supporting Cooperative Applications", *Computer Supported Cooperative Work*, 1 1992, pp 41-67.

[Sasse 94] M. A. Sasse, U. Bilting, C. Schulz and T. Turletti, "Remote Seminars through Multimedia Conferencing: Experiences from the MICE project", *In Proc. of Proc. INET '94*, 1994, pp 251-259.

[Schulzrinne 95] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *Internet Engineering Task Force, Audio-Video Transport WG*, INTERNET-DRAFT, November, 1995.

[Shepherd 95] W. D. Shepherd, "Introduction to Multimedia Systems", *In Proc. of Fundamentals of Multimedia Systems*, Lancaster University, 1995.

[Simon 96] E. Simon; "Distributed Information Systems: From Client/Server To Distributed Multimedia", McGraw-Hill Publishing Company, Berkshire, England, 1996 414 pages,

[Simon 94] R. Simon, T. Znati and R. Sclabassi, "Group Communication in Distributed Multimedia Systems", *In Proc. of IEEE: Proceedings : Distributed Multimedia Systems*, 1994, pp 294 - 301.

[Sloman 94] M. Sloman, "Network and Distributed Systems Management", Addison-Wesley, Suffolk, UK, 1994 666 pages,

[Sloman 1987] M. Sloman and J. Kramer; "Distributed Systems and Computer Networks", Prentice-Hall International (UK) Ltd, Exeter, UK, 1987 336 pages,

[Smith 89] H. Smith, J. Onions and S. Benford, "Distributed Group Communication: the AMIGO information model", Ellis Horwood Limited, West Sussex, 1989 pages,

[Steinmetz 95] R. Steinmetz and K. Mahrstedt; "Multimedia: Computing, communications and Applications", Prentice Hall P T R, Upper Saddle River, NJ, 1995 383-418 pages,

[Steinmetz 92] R. Steinmetz and M. Thomas, "Modelling Distributed Multimedia Applications", *In Proc. of International Workshop on Advanced Communication and Applications for High Speed Networks*, Munich, Germany, 1992, pp

[Szyperski 93] C. Szyperski and G. Ventre, "Efficient Multicasting for Interactive Multimedia Applications", *Tenet Group*, Technical Report, TR-93-017, March 1993.

[Szyperski 93 b] C. Szyperski and G. Ventre, "A Characterisation of Multi-Party Interactive Multimedia Applications", *The Tenet Group, International Computer Science Institute, Berkeley*, Technical Report, TR-93-006, 1993.

[Tanenbaum 92] A. S. Tanenbaum; "Modern Operating Systems", PHI, 1992 pages,

[Topolic 90] C. Topolic, "Experimental Internet STream Protocol, Version 2 (ST-2)", *Internet*, Work in Progress, RFC 1990,

[Trevor 95] J. Trevor, "Infrastructure Support for CSCW", *Ph.D Thesis*, Lancaster University, 1995.

[Tucker 94] J. Tucker; "OSI Structure of Management Information", M. Sloman , "Network and Distributed Systems Management", Addison-Wesley Publishing Company, 1994, 95 - 130,

[Turner 92] J. Turner and R. Kraut, "*Sharing Perspectiv*e", *In Proc. of CSCW '92*, Toronto, 1992.

[Twachtmann 93] B. Twachtmann and R. G. Herrtwich, "Multicast in the Heldelberg Transport System", *IBM European Networking Center Heidelberg*, Technical Report, 43.9306,

[Ventre 93] G. Ventre and C. Szyperski, "Network Support for Multi-Party Interactive Multimedia", *University of California at Berkeley*, Internal Report, 1, 1993.

[Verissimo 92] P. Verissimo and L. Rodrigues, "Group Orientation: A Paradigm for Distributed Systems of the Nineties", *In Proc. of Third Workshop on Future Trends of Distributed Computing Systems*, Taipei, Taiwan, 1992, IEEE Computer Society, pp 57-63.

[Vin 92] H. M. Vin and P. V. Rangan, "System Support for Computer Mediated Multimedia Collaborations", *In Proc. of* 1992, pp 203 - 209.

[Vogels 92] W. Vogels, P. Verissino and a. L. Rodrigues, "Requirement for High Performance Group Support in Distributed Systems", *INESC*, Report, 1 April, 1992.

[Vonderweidt 91] G. Vonderweidt and e. al, "A multipoint Communications Service for Interactive Applications", *IEEE Transactions on Communications*, 39 (12), 1991, pp 1875 - 1885.

[Wang 89] B. Wang, "An Architecture For Domain Based Distributed Systems Management", *M. Phil Thesis*, University of Lancaster, 1989.

[Waters 95] G. Waters, "Multi-Party Communication over Packet Network", *Ph.D. Thesis*, University of Kent at Canterbury, 1995.

[Waters 92] G. A. Waters, "Multicast Provision for High Speed Networks", *In Proc. of 4th IFIP Conference on High Performance Networking*, University O Liege, 1992, A. D. O. Spaniol, pp G1-1 to G1-16.

[Wilde 93] E. Wilde, "Multi-User Multimedia Editing with the MultimETH System", *In Proc. of 3rd Australian Multi-media Communications, Applications and Technology workshop*, Wollongog, Australia, 1993.

[Wilde 94] E. Wilde and C. Burkhardt, "Modelling Groups for Group Communications", *Swiss Federal Institute of Technology*, Technical, Draft, 1994.

[Wilde 96] E. Wilde, M. Nanduri and B. Plattner, "A Transport-Independent Component for a Group Session Management Service in Group communication Platforms", *In Proc. of The European conference on Multimedia Applications, Services and Techniques*, Louvain-la-Neuve, Belgium, 1996, pp

[Williams 93] N. Williams and G. S. Blair, "Distributed Multimedia Computing : An End User Perspective", *Computing Department, Lancaster University*, Internal Report.

[Williams 94] N. Williams and G. S. Blair, "Distributed Multimedia Applications: A review", *Computer Communications*, 17 (2), 1994, pp 119-132.

[Williams 92] N. Williams, G. S. Blair, G. Coulson, N. Davies and T. Rodden, "The Impact of Distributed Multimedia Systems on Computer Support For Co-Operative Work.", *Lancaster University*, Internal Report, MPG-92-43,

[Zhang 93] L. Zhang and e. al, "A New ReSerVation Protocol", *Internet Draft*, Work in Progress.

[Zhang 94] L. Zhang, R. Braden, D. Estrin, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", *Internet Engineering Task Force*, Internet Draft.

Appendix: The Media Channel Objects and Library

Media channel objects

```
#ifndef __mc_h

#define __mc_h

#include <sys/types.h>

#include <netinet/in.h>

#include <obplex.h>

typedef unsigned long mc_conid;

typedef struct {

    mc_conid con;

    obplex_objaddr_t pt;

} mc_conaddr;

typedef enum {

    mc_SRCREACH,

    mc_SRCCLOSE,

    mc_SRCPREPARE,

    mc_SRCPREPARED,

    mc_SNKLISTEN,

    mc_SNKCLOSE,

    mc_SNKREACH,

    mc_SNKPREPARE,

    mc_SNKPREPARED,

    mc_SNKJOINING,

    mc_SNKLEAVING,

    mc_SRCJOINING,

    mc_SRCLEAVING,

    mc_GETNAME,

    mc_GETTYPE,

    mc_NAMEIS,

    mc_TYPEIS,

    mc_ISVALID,

    mc_VALIDATE,

    mc_CHANNELDEAD,

    mc_SNKBROKEN

} mc_msg_t;
```

```
typedef struct {
    mc_msg_t type;
    union {
    mc_conaddr conaddr;
    mc_conid con;
    obplex_objaddr_t addr;
 } data;
} mc_msg;
typedef struct {
    void (*srcreach)(void *, const obplex_objaddr_t *, mc_conid,
                       const obplex_objaddr_t *);
    void (*srcclose)(void *, const obplex_objaddr_t *, mc_conid);
    void (*srcprepare)(void *, const obplex_objaddr_t *, mc_conid);
    void (*srcprepared)(void *, const obplex_objaddr_t *, mc_conid,
                       const obplex_objaddr_t *);
    void (*snklisten)(void *, const obplex_objaddr_t *, mc_conid,
                       const obplex_objaddr_t *);
    void (*snkclose)(void *, const obplex_objaddr_t *, mc_conid);
    void (*snkreach)(void *, const obplex_objaddr_t *, mc_conid,
                       const obplex_objaddr_t *);
    void (*snkprepare)(void *, const obplex_objaddr_t *, mc_conid);
    void (*snkprepared)(void *, const obplex_objaddr_t *, mc_conid,
                       const obplex_objaddr_t *);
    void (*snkjoining)(void *, const obplex_objaddr_t *, const char *,
                       const obplex_objaddr_t *);
    void (*snkleaving)(void *, const obplex_objaddr_t *);
    void (*srcjoining)(void *, const obplex_objaddr_t *, const char *,
                       const obplex_objaddr_t *);
    void (*srcleaving)(void *, const obplex_objaddr_t *);
    void (*getname)(void *, const obplex_objaddr_t *);
    void (*gettype)(void *, const obplex_objaddr_t *);
    void (*nameis)(void *, const obplex_objaddr_t *, const char *);
    void (*typeis)(void *, const obplex_objaddr_t *, const char *);
```

```
    void (*isvalid)(void *, const obplex_objaddr_t *,
                      const obplex_objaddr_t *);
    void (*validate)(void *, const obplex_objaddr_t *,
                      const obplex_objaddr_t *);
    void (*channeldead)(void *, const obplex_objaddr_t *);
    void (*snkbroken)(void *, const obplex_objaddr_t *, mc_conid);
} mc_dispatcher;

typedef struct {
    obplex_t *objs;
    obplex_objid id;
    mc_dispatcher *fs;
    void *handle;
} mc_obj;

int mc_init(mc_obj *ptr, obplex_t *om, mc_dispatcher *fs, void *h);
void mc_term(mc_obj *ptr);
int mc_getname(mc_obj *ptr, obplex_objaddr_t *addr);
int mc_i_srcreach(mc_obj *, const obplex_objaddr_t *, mc_conid,
       const obplex_objaddr_t *);
int mc_i_srcclose(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_srcprepare(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_srcprepared(mc_obj *, const obplex_objaddr_t *, mc_conid,
        const obplex_objaddr_t *);
int mc_i_snklisten(mc_obj *, const obplex_objaddr_t *, mc_conid,
        const obplex_objaddr_t *);
int mc_i_snkclose(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_snkreach(mc_obj *, const obplex_objaddr_t *, mc_conid,
       const obplex_objaddr_t *);
int mc_i_snkprepare(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_snkprepared(mc_obj *, const obplex_objaddr_t *, mc_conid,
         const obplex_objaddr_t *);
int mc_i_snkjoining(mc_obj *, const obplex_objaddr_t *, const char *,
        const obplex_objaddr_t *);
int mc_i_snkleaving(mc_obj *, const obplex_objaddr_t *);
```

```
	int mc_i_srcjoining(mc_obj *, const obplex_objaddr_t *, const char *,
		const obplex_objaddr_t *);
	int mc_i_srcleaving(mc_obj *, const obplex_objaddr_t *);
	int mc_i_getname(mc_obj *, const obplex_objaddr_t *);
	int mc_i_gettype(mc_obj *, const obplex_objaddr_t *);
	int mc_i_nameis(mc_obj *, const obplex_objaddr_t *, const char *);
	int mc_i_typeis(mc_obj *, const obplex_objaddr_t *, const char *);
	int mc_i_isvalid(mc_obj *, const obplex_objaddr_t *,
		const obplex_objaddr_t *);
	int mc_i_validate(mc_obj *, const obplex_objaddr_t *,
		const obplex_objaddr_t *);
	int mc_i_channeldead(mc_obj *, const obplex_objaddr_t *);
	int mc_i_snkbroken(mc_obj *, const obplex_objaddr_t *, mc_conid);

	struct mc_joining {
	   mc_msg_t id;
	   obplex_objaddr_t addr;
	};
#endif
```

mc_snk and mc_src objects

```
	#ifndef __mccli_h

	#define __mccli_h

	#ifdef __cplusplus

	extern "C" {

	#endif

	#include <obplex.h>

	#include <list.h>

	#include "mc.h"

	#ifndef NULL

	#define NULL 0

	#endif
```

```
typedef struct mc_connection_tag mc_connection_t;
typedef void *(mc_snkopen_f)(void *ctxt, const obplex_objaddr_t
   *addr, mc_connection_t *ref);
typedef void (mc_snkclose_f)(void *ctxt, void *uref);
typedef struct {
 mc_snkopen_f *open, *openon;
 mc_snkclose_f *close;
} mc_snkmethod;
typedef void *(mc_srcinclude_f)(void *ctxt, const
   obplex_objaddr_t *addr, mc_connection_t *ref);
typedef void (mc_srcexclude_f)(void *ctxt, void *uref);
typedef struct {
 mc_srcinclude_f *include;
 mc_srcexclude_f *exclude;
} mc_srcmethod;
typedef struct mc_snk_tag mc_snk;
typedef struct mc_src_tag mc_src;
int mc_snkopen(mc_snk *obj, obplex_t *om, const obplex_objaddr_t
   *mgraddr, const char *media_type, const obplex_objaddr_t
   *agtaddr, const mc_snkmethod *fs, void *ctxt);
int mc_snkclose(mc_snk *obj);
int mc_srcopen(mc_src *obj, obplex_t *om, const obplex_objaddr_t
   *mgraddr, const char *media_type, const obplex_objaddr_t
   *agtaddr, const mc_srcmethod *fs, void *ctxt);
int mc_srcclose(mc_src *obj);
/* private parts of snks and srcs */
struct mc_snk_tag {
   mc_obj base;
   void *ctxt;
   const mc_snkmethod *method;
   obplex_objaddr_t mgraddr;
   obplex_t hashtable;
   list_hdr(mc_connection_t) cons; /* list of connections */
};
```

```
struct mc_src_tag {

   mc_obj base;

   void *ctxt;

   const mc_srcmethod *method;

   obplex_objaddr_t mgraddr;

   obplex_t hashtable;

   list_hdr(mc_connection_t) cons; /* list of connections */

};

struct mc_connection_tag {

   list_elem(mc_connection_t) others;

   obplex_objaddr_t addr;

   mc_conid id;

   void *ref;

};

/* end of private parts */

#ifdef __cplusplus

};

#endif

#endif
```

Library

`mc_obj` API

```
int mc_init(mc_obj *ptr, obplex_t *om, mc_dispatcher *fs, void *h);

void mc_term(mc_obj *ptr);

int mc_getname(mc_obj *ptr, obplex_objaddr_t *addr);
```

`mc_init` initialises an `mc_obj` pointed to by ptr. `om` points to an object multiplexer which will define *ptr 's network address. fs points to a structure containing pointers to functions to be invoked when messages are passed to *ptr. `h` is a user-definable handle which is passed back when those functions are called.

`mc_term` deallocates resources for `*ptr`. It is no longer an addressable object, and cannot have any of its `fs` functions invoked.

`mc_getname` writes the canonical distributed address of `*ptr` in `*addr`.

```
int mc_i_srcreach(mc_obj *, const obplex_objaddr_t *, mc_conid,
                      const obplex_objaddr_t *);
int mc_i_srcclose(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_srcprepare(mc_obj *, const obplex_objaddr_t *,mc_conid);
int mc_i_srcprepared(mc_obj *, const obplex_objaddr_t *, mc_conid,
                      const obplex_objaddr_t *);
int mc_i_snklisten(mc_obj *, const obplex_objaddr_t *, mc_conid,
                      const obplex_objaddr_t *);
int mc_i_snkclose(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_snkreach(mc_obj *, const obplex_objaddr_t *, mc_conid,
              const obplex_objaddr_t *);
int mc_i_snkprepare(mc_obj *, const obplex_objaddr_t *, mc_conid);
int mc_i_snkprepared(mc_obj *, const obplex_objaddr_t *, mc_conid,
                      const obplex_objaddr_t *);
int mc_i_snkjoining(mc_obj *, const obplex_objaddr_t *, const char *,
                      const obplex_objaddr_t *);
int mc_i_snkleaving(mc_obj *, const obplex_objaddr_t *);
int mc_i_srcjoining(mc_obj *, const obplex_objaddr_t *, const char *,
                      const obplex_objaddr_t *);
int mc_i_srcleaving(mc_obj *, const obplex_objaddr_t *);
int mc_i_getname(mc_obj *, const obplex_objaddr_t *);
int mc_i_gettype(mc_obj *, const obplex_objaddr_t *);
int mc_i_nameis(mc_obj *, const obplex_objaddr_t *, const char *);
int mc_i_typeis(mc_obj *, const obplex_objaddr_t *, const char *);
int mc_i_isvalid(mc_obj *, const obplex_objaddr_t *,
                      const obplex_objaddr_t *);
int mc_i_validate(mc_obj *, const obplex_objaddr_t *,
                      const obplex_objaddr_t *);
```

```
int mc_i_channeldead(mc_obj *, const obplex_objaddr_t *);

int mc_i_snkbroken(mc_obj *, const obplex_objaddr_t *, mc_conid);
```

Each of these functions passes a formatted message (depending on which function is invoked) to the object whose address is pointed to by the second argument. The object refered to by the first argument is indicated to the receiving object as being the sender. All remaining arguments are used to form the message.

`mc_conid` is a manager-defined reference value used to identify the relationship between one of its sources and one of its sinks.

```
typedef struct {

    void (*srcreach)(void *, const obplex_objaddr_t *, mc_conid,

                      const obplex_objaddr_t *);

    void (*srcclose)(void *, const obplex_objaddr_t *, mc_conid);

    void (*srcprepare)(void *, const obplex_objaddr_t *, mc_conid);

    void (*srcprepared)(void *, const obplex_objaddr_t *, mc_conid,
                      const obplex_objaddr_t *);

    void (*snklisten)(void *, const obplex_objaddr_t *, mc_conid,

                      const obplex_objaddr_t *);

    void (*snkclose)(void *, const obplex_objaddr_t *, mc_conid);

    void (*snkreach)(void *, const obplex_objaddr_t *, mc_conid,

                      const obplex_objaddr_t *);

    void (*snkprepare)(void *, const obplex_objaddr_t *, mc_conid);

    void (*snkprepared)(void *, const obplex_objaddr_t *, mc_conid,

                      const obplex_objaddr_t *);

    void (*snkjoining)(void *, const obplex_objaddr_t *, const char *,

                      const obplex_objaddr_t *);

    void (*snkleaving)(void *, const obplex_objaddr_t *);

    void (*srcjoining)(void *, const obplex_objaddr_t *, const char *,
                      const obplex_objaddr_t *);

    void (*srcleaving)(void *, const obplex_objaddr_t *);

    void (*getname)(void *, const obplex_objaddr_t *);

    void (*gettype)(void *, const obplex_objaddr_t *);

    void (*nameis)(void *, const obplex_objaddr_t *, const char *);
```

```
    void (*typeis)(void *, const obplex_objaddr_t *, const char *);
    void (*isvalid)(void *, const obplex_objaddr_t *,
                    const obplex_objaddr_t *);
    void (*validate)(void *, const obplex_objaddr_t *,
                    const obplex_objaddr_t *);
    void (*channeldead)(void *, const obplex_objaddr_t *);
    void (*snkbroken)(void *, const obplex_objaddr_t *, mc_conid);
} mc_dispatcher;
```

An `mc_dispatcher` structure is passed to an `mc_obj` when it is initialised. The structure specifies which of the user's functions should be invoked when the `mc_obj` is passed a message. The first argument is the user-defined handle h passed to the `mc_obj` when it was initialised - the user function is expected to know how to interpret it. The second argument is the address of the sender. The format of the remaining arguments is specific to the message type, and their values are extracted from the message.