Channel Groups

A Unifying Abstraction for Specifying Inter-stream Relationships

Amit Gupta and Mark Moran
{amit,moran}@cs.Berkeley.EDU
Tenet Group
Computer Science Division
University of California at Berkeley
&
International Computer Science Institute
Berkeley, CA

Abstract

A single distributed application typically requires setting up a number of real-time connections, or channels. Current schemes usually assume that different channels are independent, when in reality, important relationships often exist between them. We introduce a new abstraction called *channel groups* that allows network clients to describe these relationships explicitly to the network service provider. For example, by describing sharing relationships between channels, the network client enables the network to share resource allocations among related channels —lowering the cost and improving the scalability of communication. In addition, specification of other relationships, such as inter-stream synchronization, disjoint-path routing, relative dropping priorities, and simultaneous establishment provide a richer, more efficient service. Channel groups provide a unifying abstraction and an easily-extensible interface for specifying these and other relationships. This paper presents a general description of the channel group abstraction and demonstrates its usefulness in describing several types of inter-stream relationships.

1. Introduction

The increasing speed and connectivity of computer networks and the improvement of workstation capabilities are enabling a new class of distributed multimedia applications. (See e.g. [Adam93], [GruFec91]). The large amounts of data that can be stored in a high-speed, wide-area network and the real-time networking requirements of interactive distributed multimedia applications favor a proactive approach to network management, instead of the more traditional reactive approach used in current data networks. It is widely accepted that in order to provide real-time performance guarantees, some kind of *admission control* is necessary in order to control network load (see, e.g., [FeBaZh92], [ClShZh92], [GuAhNa91], [HyLaPa92]). Admission control schemes necessarily require characterization of future traffic. Current real-time systems usually treat traffic on different channels[1] as independent when determining their resource requirements, which may result in over-allocation of resources. This limitation may be overcome, in part, by using measurements over aggregations of channels, as in the *predictive service* [JaShZh92], to determine the resource requirements. However, this approach cannot be used when guarantees are required, when measured performance does not predict future behavior accurately (e.g. when a single channel can have a significant effect on the performance of other channels, such as near the edges of the network), or when the resource measurements are not available (e.g. when a channel is established).

Another approach is to allow clients to describe the application-specific relationships among channels. For example, workstation video-conferencing applications are usually written so that each participant only receives the video from the current speaker. In this case, the network client could inform the network that of the *N* multicast channels distributing video, only one is active at any time. The network then could use this information to overlap the *N* multicast trees as much as possible and share allocation of network resources among them.

The above example demonstrates one important type of relationship between channels. However, there are many others, some of which will be described briefly in this paper. The important characteristic about these relationships is that they are known by the application and cannot be easily discovered by the network. Therefore, we have developed a single, extensible abstraction, called *channel groups,* by which network clients can describe these relationships to the network. This paper will focus on the motivation and general description of the channel group abstraction. A detailed discussion of the design, the client-service interface, and various inter-channel relationships are beyond the scope of this paper. The next section discusses the motivation behind the channel group abstraction. Section 3 demonstrates the concept by describing several important relationships. Example channel groups for a few distributed multimedia applications are discussed in Section 4. Section 5 discusses some of the membership and implementation issues. Section 6 describes work in progress. Finally, Section 7 discusses some related work.

1. a *channel*, or *flow*, is a simplex data connection with traffic characterization for which the network provides real-time performance.

2. Motivation

In current real-time networking schemes, the traffic characterization, performance requirements, and establishment are handled separately for each channel or flow. However, many interactive distributed multimedia applications require a number of real-time channels carrying various types of media. For these applications, important relationships often exist between the channels. In the previous section we discussed an application-specific relationship that would allow a network resource allocation to be shared *among* channels, without sacrificing real-time performance guarantees. Other important relationships include synchronization between the data on related streams (e.g. between the video and audio of a speaker), routing directives concerning multiple channels (e.g. requesting that channels be routed over disjoint paths, if possible, in order to improve fault tolerance), establishment relationships (e.g. in a distributed seminar there is no point in establishing a channel from a listener to the speaker if the channel from the speaker to the listener could not be established), and the relative dropping priority for data packets across channels. Certainly other important relationships exist and will be discovered with the development of new interactive distributed multimedia applications.

Given that such relationships do exist between channels, there are two reasons it is important for clients to specify them to the network. The first, as discussed briefly in the previous section, is to allow allocation of network resources to be shared among related channels without compromising their real-time performance guarantees. This capability is important for lowering the cost and improving the scalability of communication. Scalability improvement will be greatest for communication among a very large number of participants, because the number of simultaneously active sources generally does not increase with the number of participants.

The second reason for allowing clients to specify inter-channel relationships is to improve the service provided by the network. For example, it is generally agreed that clients should have some method of indicating streams that should be synchronized on playout at the receiver (e.g. [CaSaCo92]). In general, such relationships are specified via ad hoc additions to the client-service interface. For example, in the case of inter-stream synchronization, channels are often grouped together into larger units (e.g. *bundles* in [CaSaCo92]). While this method is convenient for specifying synchronization, other methods would have to be introduced to specify other inter-channel relationships[2]. Clearly, a more general, extensible interface that enables clients to specify the inter-stream relationships in a uniform manner is desirable.[3]

We have designed the channel group abstraction to provide such an interface. The interface allows clients to specify inter-channel relationships by including them in a channel group of the appropriate type. Although we describe a few sample relationships in this paper, the abstraction makes no assumptions regarding the semantics of group participation. Instead, the network defines the types of groups supported, as well as the parameters and semantics of each group. Therefore, channel groups provide a unifying abstraction that allows clients to specify inter-channel

2. Some of the inter-channel relationships are discussed in Section 3 and in Section 6.

3. Channel groups provide a general abstraction at the client-service interface and different types of channel groups can be supported at different levels inside the network. For instance, relationships dealing with routing would be supported at the network layer, while those dealing with related parameters *may* be supported at higher levels. The choice depends on the specific network architecture. Similar arguments have been made by Escobar et al. in [EsDePa92].

relationships while imposing as few limitations and as little semantics on that specification as possible. The interface is extensible as new relationships can be supported without changing the existing primitives by adding new primitives or by using new parameter values for existing primitives. It should be noted that while channel groups provide a uniform interface for specifying inter-channel relationships, the abstraction does not imply any correspondence between the manner in which the network implements groups of different types. In addition, clients only need to know about the relationships that they use.

Thus, in defining the channel group abstraction we have distinguished the unit of data transmission (the channel) from the unit used for describing resource requirements, routing, establishment instructions, etc. (the channel group).

3. Relationships between channels

A channel group is described by a set of member channels along with the relationship that relates them. As mentioned in the previous section, a channel is defined as a simplex end-to-end connection with traffic characterization and performance guarantees. Relationships may be classified as mandatory or advisory. In specifying mandatory relationships, the clients indicate additional requirements that the network must meet for the channels in the group. In specifying an advisory relationships, the clients provide the network with additional information that may enable it to service a group of channels more efficiently by considering them together, rather than separately.

In this section, we discuss three representative relationships that may exist among different channels. We first discuss an advisory relationship, the *resource sharing* relationship. We then discuss a mandatory relationship, the *related parameters* relationship. We conclude this section with a description of the *multicast* relationship, which can be used to specify the multicasts in a simple and extensible manner. Some other relationships are discussed in Section 6. A detailed discussion of the different specifications and implementations of the various relationships is beyond the scope of this paper.

3.1 Resource sharing relationship

Consider an audio-conference of 80 participants. Due to the cooperative nature of the meeting, it may be assumed that no more than three persons speak at any given time (indeed, in an orderly meeting, only one person speaks at any time; two or more persons speak simultaneously only when they try to obtain the floor; clearly, this situation lasts for but a short period of time).

The communication is set up in the following manner: 80 multicast channels are established; one from each sender to all the participants. In a network that performs admission control, unless we have the additional knowledge that no more than three persons will be speaking simultaneously, the network will try to find enough resources to support the 80 multicasts independently. If it fails to do so, the conference will be rejected. This result is also valid for predictive service [ClShZh92] where the admission control [JaShZh92] is based on the client-specified traffic for the *new* channels.

This example illustrates what we refer to as the *resource sharing* relationship. The multicasts are related in that when some of the multicasts pass through the same node (or link), we need to reserve no more resources than are required to simultaneously support three multicasts. In other words, the 80 multicasts can share resources over common sub-paths.

A resource sharing relationship specification consists of a set of channels along with the resource 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. As the network is free to reserve more resources than absolutely necessary, this is an advisory relationship.

3.2 Related parameters relationship

Consider an application in which clients set up simultaneous audio and video connections from a sender to a destination. The application provides the necessary synchronization (lip-syncing) by time-stamping the data at the sender and appropriately buffering the two streams at the destination. However, because of buffer space limitations at the destinations, the application requires that *delay jitter* (the variation in the delay experienced by packets) be bounded for each channel and that the delay bounds for the two streams must be within 10 ms of each other.

This scenario is an example of a situation where the *related parameters* relationship is useful. In this particular case, a delay tuple, specifying the delay bound in milliseconds on the audio and the video channel respectively, of <50, 55> is acceptable, as is <100,110>, but the tuple <60,80> is not acceptable.

A related parameters relationship specification consists of a set of channels along with a predicate that links the particular parameters of the different channels. This is a mandatory relationship as the channel establishment is acceptable to the client only if the predicate is satisfied.

3.3 Multicast relationship

It is well recognized that efficient multicasting is crucial to the success of distributed conferencing as well as other related applications. For this discussion, we consider a *point-to-multipoint* model; the discussion is also applicable to *multipoint-to-multipoint* multicast models.

A multicast can be specified in a number of ways. We restrict this discussion to two alternative specifications. One possibility is that the multicast may be specified by naming the source, the destination set and the various properties of the multicast. We call this the monolithic multicast channel specification. The other possibility is to specify the multicast as a channel group, with the logical *point-to-point* (unicast) data streams as the members.

Clients may wish to specify a number of additional properties for the multicast. For example, a single multicast might have different QoS parameters for different destinations. Clients might also wish to specify the action to be taken if the multicast is only partially successful, either by specifying the more important participants or by specifying the quorum requirements for a successful multicast establishment. Clients also may wish to specify full-duplex communication mode.

Thus, there exist a number of important properties that the clients may wish to specify. The network may have mechanisms to support only a few of them. It is important that the multicast specification should be easily extensible, and that *different* networks, with *different* levels of support for multicasting, should be able to use the *same* multicast specification interface. In this case, the monolithic multicast specification would be unwieldy and inextensible. On the other hand, specifying multicasts by using channel groups allows the specification to be easily extended. In particular, we can support all the above constraints by composing multicast groups with channel establishment groups. A channel establishment group is described by specifying two sets of channels, the primary set and the secondary set, along with a *quorum* requirement on the primary set. The *quorum* specifies the number of members of the primary set which must be established for the client to accept the establishment of *any* member of the secondary set

4. Examples of channel groups

In this section, we discuss some example channel groups for certain multimedia applications. It may be observed that the channel groups are defined by relationships that may differ between two different invocations of the same application. Indeed, two different client application writers may choose different sets of relationships among different channels. There exist no unique ***best*** channel groups. The channel groups merely help specify the relationships that exist between different channels, according to the application software designers. Also, for simplicity, we use the term multicast channel to refer to the channels that comprise a channel group of type multicast.

4.1 Teleconference

Consider a simple teleconference. There are 12 participants. All of them are equipped with microphones. In addition, two of the participants have video input devices as well: Participant A has a video camera, and participant B has a document camera. Due to the nature of the conference, the following relationships hold:

- No more than 3 participants may speak at the same time.
- Only one video device is used to transmit data at a time.
- The maximum permissible difference between the delay experienced by a video packet, and the corresponding audio packet from A, as perceived by any participant, must be less than 10 ms.
- For the special participant A, the video channel should be established only if the corresponding audio is established.
- For the special participant B, the video channel should be established only if the corresponding audio is established.

We obtain the channel groups in Table 1 for this scenario. The channel groups have been given arbitrary names.

Channel Group	**Members**	**Relationship**	**Description**
Audio	All audio channels	Resource Sharing	Total requirement equals that of 3 audio channels
Video	The two video channels	Resource Sharing	Total requirement equals that of 1 video channel
Lip-Synch	The audio and the video channel from A	Related Parameters	Delay difference less than 10 ms.
Alloc A	The audio and the video channel from A	Channel Establishment	Allocate the video only if the audio can be established.
Alloc B	The audio and the video channel from B	Channel Establishment	Allocate the video only if the audio can be established.

TABLE 1. Channel groups for the teleconference

4.2 Seminar

Consider a simple seminar with one speaker and 60 listeners. All participants are equipped with microphones. In addition, the speaker has a video camera. Due to the nature of the seminar, the following relationships can be defined:

- No more than 3 persons may speak at the same time.
- The maximum permissible difference between the delay experienced by a video packet, and the corresponding audio packet from the speaker, as observed by any listener, is less than 20 ms. Also, the video data may not arrive *before* the corresponding audio data.
- There is no use in establishing communication from an audience member to the speaker unless the speaker can talk to the member.
- The speaker video channel should be established only if the speaker audio channel is established.
- It is preferable but not necessary that the audience members be able to listen to each other.

We obtain the channel groups in Table 2 for this scenario. The channel groups have been given arbitrary names. Note: there is one *Listener Audio* channel group for each listener, with membership as specified in this table.

Channel Group	Members	Relationship	Description
Audio	All audio channels	Resource Sharing	Total requirement equals that of 3 audio channels
Lip-Synch	The audio and the video channel from the speaker	Related Parameters	Delay for audio greater than the delay for video, but not by more than 20 ms.
Listener Audio	Audio to a listener, and the audio from that listener.	Channel Establishment	Allocate the audio from the listener only if the audio can be established to that listener.
Establish	The audio and the video channel from the speaker	Channel Establishment	Allocate the video only if the audio can be established.

TABLE 2. Channel groups for the seminar

4.3 Panel discussion

Consider a simple panel discussion. There are 4 panel members, including a moderator, with about 30 audience members. All panel members are equipped with microphones and video cameras. In this panel discussion, the audience can not interrupt the panel members, or ask questions. Due to the nature of the panel discussion, the following relationships hold:

- No more than 2 panel members may speak at the same time.
- Only one video device is used to transmit data at any time, i.e. the video from the panel member that has obtained the floor.
- The maximum permissible difference between the delay experienced by a video packet, and the corresponding audio packet from any panel member, as seen by any discussion participant, is less than 10ms.
- A video channel to an audience member should be established only if the corresponding audio channel is established.
- An audience member should get the audio from all 4 speakers; otherwise he is not interested in attending the discussion.
- The discussion should be called off if any panel member cannot speak to, or listen to, any other panel member.

We obtain the channel groups in Table 3 for this scenario. As above, the channel groups have been given arbitrary names.

Channel Group	Members	Relationship	Description
Audio	All audio channels	Resource Sharing	Total requirement equals that of 2 audio channels
Video	All video channels	Resource Sharing	Total requirement equals that of 1 video channel
Lip-Synch	The audio and the video channel from the same speaker to the same listener.	Related Parameters	Difference between the delay for audio and the delay for video not greater than 10 ms.
Audio Video Establish	The audio and the video channel from the same speaker to the same listener.	Channel Establishment	Allocate the video only if the audio can be established.
Audio Establish	The audio channels from different speakers to the same listener.	Channel Establishment	Allocate the channels only if all the audio channels can be established.
Panel Establish	All channels between panel members.	Channel Establishment	Allocate the channels only if all the audio channels between all of the panel members can be established.

TABLE 3. Channel groups for the panel discussion

5. Membership and Implementation Issues

In this section we discuss a few group membership issues and implementation considerations.

5.1 Membership Issues

It is tempting to insist that a channel may not be a member of two different groups at the same time. This restriction simplifies membership information management. However, the restriction also leads to a considerably poorer model (as can be seen from the examples in the previous section, channels are usually members of many groups at the same time); one that does not truly reflect the relationships that exist between different channels. We conclude that all implementations must provide support for channels as members of multiple groups at the same time.

An important question concerns the temporal dynamicity of channels as members of groups, i.e. the ability of channels to join and leave groups during their lifetimes. There are a number of good reasons for permitting channels to dynamically join (or leave) a channel group. The arguments against permitting dynamicity stem primarily from implementation considerations. Dynamicity in related parameters relationship, for example, may be inconsistent with the already allocated delay and jitter bounds. Dynamicity in the resource-sharing relationship may not be permitted because the network may not have enough resources to support a channel independently.

These considerations lead us to the following design. Channels may join or leave a channel group at any time. The network handles requests for dynamic changes in group membership by performing additional tests to determine whether the requests can be accommodated. The network should try to accommodate these requests; however, it reserves the right to reject them.

It is very useful to permit groups to be member of other groups. The specification of a group, say A, as a member of another group, say B, is a useful shorthand for saying that all the members of group A are also members of group B. Channels that dynamically join the group A automatically join the group B. We do not permits cycles in the group relationships. This means that if group A is a member of group B, then group B cannot be a member of group A, or of any group that is a member of group A (and so on).

5.2 Implementation Issues

Different types of channel groups have very different properties and in general, require very different implementation mechanisms. As was mentioned before, channel groups provide a unifying abstraction and a uniform interface to these vastly different implementation mechanisms. Some implementation issues include the client-service interface for specifying channel groups; implications for admission control, rate-control and scheduling disciplines; and the changes required in current networking protocols to support the additional functionality that can be specified with channel groups. The protocol issues are discussed in this section in the context of the Tenet real-time networking protocols described in [FeBaZh92].

A key requirement in the design of our client-service interface is that it include provisions for dynamic changes in group membership. A client informs the network about the channels and the groups it requires. The network then returns an identifier to the client for every channel and for every group. The client uses these identifiers to inform the service about the group memberships and relationships. The client then requests the network to establish the channel groups. A channel group establishment implies the attempt to establish all group members.

For supporting the resource sharing relationship, the network requires mechanisms for enforcing group traffic limits. A policy issue concerns the action to be taken when these bounds are exceeded. The network should preferably use the same policy as the policy used when single channel traffic bounds are exceeded. Our current scheme for channel groups gives the network complete flexibility, i.e. it is not required to drop offending packets or to do its best to forward them (this flexibility is limited by the fact that the network has to protect the real-time channels from other real-time channels and from non-real-time traffic). Thus, the network is free to choose the scheduling algorithms independent of this consideration.

A minimal channel groups implementation does not provide any additional functionality to the clients; however, it maintains compatibility with the proper channel group implementations, and the networking code can be incrementally upgraded to provide more complete support for channel groups without modifying the client-service interface. A minimal channel group implementation would support the client-service interface for most relationships. During establishment, the network would first attempt to establish all channels without considering the channel group relationships. When the service receives all establishment results, it would check if any mandatory relationships are being violated. If so, it would send a request to teardown all established channels and inform the client that the channel group could not be established. Else, it would inform the client that the channel group was successfully established. All dynamic membership change requests would be rejected.

A more complete implementation is planned for the Tenet real-time protocol suite. The suite is a set of communication protocols designed to provide real-time communication (i.e. with performance guarantees) in a packet-switching internetwork. We discuss the changes planned in this suite for supporting the additional functionality that can be specified by using channel groups.

The Real-time Channel Administration Protocol (RCAP) [BanMah91] handles connection-level channel management for the protocol suite in response to requests from application programs. During channel establishment, RCAP performs the admission control tests. Supporting channel groups requires RCAP to manage group membership information. We are modifying RCAP to be a layered control protocol stack for the second generation protocol suite. The new RCAP will provide support for channel groups.

The Real-time Internet Protocol (RTIP) [ZhVeFe92] provides a simplex, connection-oriented, guaranteed performance, unreliable, sequenced packet delivery service. It occupies a place analogous to IP in the Internet protocol suite. RTIP does not require any change for supporting channel groups per se. However, RTIP needs changes for providing the additional functionality that can be specified by using channel groups. For example, RTIP needs extensions to enforce the traffic bounds on a per-group basis (for enforcing *resource sharing* relationships) and for the network to support client-specified dropping priorities (Section 6). Implementing channel groups does not require any changes in the transport-layer protocols, namely, the Real-time Message Transport Protocol (RMTP) and the Continuous Media Transport Protocol (CMTP).

6. Work in Progress

In this section, we briefly describe some of the work that is currently in progress. An interesting idea concerns client specification of the packets to be dropped if the total traffic for a resource sharing group exceeds the limit specified in the group relationship. This priority scheme can be merged with hierarchical coding to obtain unified dropping priorities for data packets across different channels[4]. We are looking into parameterized relationships, i.e. relationships in which some of the relationship parameters depend on the properties of the group (e.g. group size).

A key issue involves the dropping priority of packets on *different* channels, where the packets from lower priority channels are dropped before the packets from higher priority channels are dropped. For example, consider a distributed conference where the resource allocations are shared among the participants. The conference organizers will desire that the network provide a higher priority to the conference moderators, thus ensuring that the moderator(s) will be heard if too many people try speaking at the same time. Simple changes are required in the rate-control mechanisms of the data transmission protocols (network layer) to support these priorities.

Hierarchical coding provides a mechanism for network clients to indicate the critical packets whose timely delivery is most important. This mechanism can be viewed as specifying the relative dropping priorities for packets *within* a channel. The channel group priority mechanisms can be combined with hierarchical coding to provide an integrated priority scheme by which the applications can specify priorities, on a per packet basis, across different channels. This integration may be useful; however, it adds to the delay in time-sensitive streams and to the complexity of the client-service interface and networking code. Also, hierarchical coding is not very useful if the channels have deterministic performance guarantees.

A simple example of a parameterized relationship is a video conference where the current speaker transmits high-resolution images, while all other participants transmit low-resolution images. In this case, the conference needs enough resources to support ***one*** high-resolution, and ***N-1*** low-resolution image streams, where ***N*** is the size of the conference. Here, some attributes of the resource sharing relationship are not specified as constants, but as functions of the *size* of the conference. This dependence has a significant impact on the dynamic changes in group membership. We are looking into the different aspects of the parameterized relationships.

We are currently extending Galileo [Knightly92], a simulator for real-time networks, to include support for channel groups. The experiments planned include comparing the usefulness of the different specifications of the resource sharing relationship. Experiments have also been planned to evaluate the usefulness of the related parameters relationship.

7. Related Work

Research concerning relationships between channels has primarily focussed on the inter-stream synchronization. Da Costa Carmo et al.[CaSaCo92] introduce a multimedia flow grouping called *bundles*. A *bundle* is defined as a “parallel grouping of several flows within a given temporal range”. They also specify a *bundle transfer function* to specify the inter-flow synchronization requested. The *bundle* abstraction is very similar to the channel group specification with the related parameters relationship, where the predicate restricts the difference between delays for the concerned jitter-controlled channels. The *bundles* abstraction is different, however, in that it is

4. Consider a distributed video CSCW. The clients may request that if possible, all the video should be sent at high-resolution. However, if too many participants send a lot of video data, the video of all the participants be reduced to lower resolution, till the traffic conforms to the group specification. However, the clients may wish to specify some ordering in this degradation.

based on a time-stamping model while we plan to build related parameters channel groups on the basis of jitter-controlled channels. The *strands* of Rangan and Vin [RanVin91] and the *ropes* of the Berkeley DASH project [Anderson90] are also similar to bundles.

Campbell et al. describe an ***orchestration*** service for *coordinating* multiple related transport virtual circuits in [CaCoGaHu92]. They say that the reason why they employ the term orchestration rather than synchronization is because cross-stream relationships may encompass more than just temporal coordination. However, they limit themselves to the cross-stream relationships at the transport layer. Indeed, apart from synchronization, the only other possibilities that they consider is linking QoS degradations together on one virtual circuit to corresponding compensations on another. It is not clear whether they have any mechanisms for linking QoS degradations.

A close parallel can be drawn between the resource sharing relationship and the Resource ReSerVation Protocol (RSVP) described in [HPNRR92], and the resource sharing in [MorGus92]. RSVP supports M x N multicasts and allows each receiver to specify the number of sources to be simultaneously transmitting to it. However, the discussion in [HPNRR92] as well as [MorGus92] is restricted to the resource sharing aspect of the channel group abstraction, though the resource sharing relationship is not clearly defined. Another difference is that in RSVP, the number of sources sending to a destination is limited by active filtering within the network. In contrast, channel groups specify relationships that does not require actions to be taken by the network.

In the predicted service [ClShZh92], [JaShZh92], [Zhang93], classes are used for QoS specification. Also, considerable work has been done by Van Jacobson and Sally Floyd in partitioning the network resources between paying organizational entities and/or application classes [Floyd93]. At first glance, the class mechanism looks like a special form of group relationship and the network partitioning scheme looks like resource sharing channel groups. However, the similarities are very superficial and the work is very different.

Suggestions for Future Work

There are a number of interesting ideas and extensions to the current work on channel groups. The current work is part of ongoing research to develop a second generation scheme for real-time communication. Of particular interest are interactions between channel groups and other new ideas and mechanisms for real-time communication, including the interactions with advance reservations and resource partitioning. The second generation scheme will include guaranteed advance reservations for real-time communication. To support this capability, techniques have been devised to partition the resources of a real-time network into several private *virtual* subnetworks with independent admission control. Another interesting idea concerns using the channel group information as directives for channel routing.

RFC 1363 proposes a flow specification, whose goal “is to be able to describe any flow requirement” [Partridge92]. Unfortunately, this flow specification does not include any support for describing the relationships that may exist between different flows. The flow specification designers view the flow specification as simply specifying the traffic and some QoS requirements; the other features of channel setup are handled using other mechanisms which lie outside the scope of the flow specification [EsDePa92], [Partridge93]. However, we feel that the flow specification should be extended to address these important inter-flow properties. It is not clear as to how this may be best done, and as to how difficult it is likely to be.The key problem is that it is very hard to find a single compact format for such a wide variety of relationships.

Conclusions

We have introduced a new abstraction called channel groups. The abstraction allows network clients to specify explicitly the relationships that exist between related channels. This explicit specification enables the network to achieve higher efficiency, both in resource usage and in client-service interactions. Channel groups also provide a simple, powerful and extensible specification for multicasts. We have discussed some of the issues involving channel groups, including dynamicity in group membership, dropping priorities between packets of different channels, and various implementation issues. Our current work includes incorporating channel groups in the second generation of our real-time protocol suite.

Acknowledgments

We would like to thank Domenico Ferrari, who guided us in this research. Thanks are due to Srinivasan Keshav, Ed Knightly, Bruce Mah, Clemens Szyperski, Giorgio Ventre and Makiko Yoshida for their suggestions. We would like to thank Lixia Zhang, David Clark, Sally Floyd and Craig Partridge for valuable insights and information. This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Digital Equipment Corporation, Hitachi Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this paper are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

References

[Adam93] J. Adam, “Special Report: Interactive Multimedia”, IEEE Spectrum, (Mar. 1993), pp. 22-39.

[Anderson90] D. Anderson, “Meta-scheduling for Distributed Continuous Media”, Technical Report UCB/CSD 90/599, University of California, Berkeley, CA (Oct 1990)

[BanMah91] A. Banerjea, B. Mah, “The Real-Time Channel Administration Protocol”, Proc. 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video, (Nov. 1991).

[CaCoGaHu92] A. Campbell, G. Coulson, F. Garcia, D. Hutchison, “A Continuous Media Transport and Orchestration Service”, Proceedings of SigComm 92, Baltimore MD (Aug 1992)

[CaSaCo92] C. Carmo, P. Sannes, J. Courtiat, “Basic Synchronization Concepts in Multimedia Systems”, Proceedings of 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego CA (Nov 1992)

[ClShZh92] D. Clark, S. Shenker, L. Zhang, “Supporting Real-time Applications in an Integrated Services Packet Network: Architecture and Mechanism”, Proceedings of Sigcomm 92, Baltimore MD (Aug 1992)

[EsDePa92] J. Escobar, D. Deutsch, C. Partridge, “Flow Synchronization Protocol”, Proceedings of Globecom 92, Orlando FL (Dec 1992)

[FeBaZh92] D. Ferrari, A. Banerjea, H. Zhang, “Network support for multimedia - A Discussion of the Tenet approach”, Technical report TR-92-072, International Computer Science Institute, Berkeley CA (Nov 1992)

[Floyd93] S. Floyd, personal communication (Jan-Feb 1993).

[GruFec91] J. Grudin, L. Feczko, “Special Section on Computer-Supported Cooperative Work”, Communications of the ACM, December 1991, pp. 30-90.

[GuAhNa91] R. Guerin, H. Ahmadi and M. Naghshineh, “Equivalent Capacity and It’s Application to Bandwidth Allocation in High-Speed Networks”, IEEE J. Select. Areas Commun., September 1991.

[HPNRR92] “Integrated Services Packet Network Architecture Development,” Xerox PARC, High Performance Networking Research Report (Dec. 1992)

[HyLaPa92] J. Hyman, A. Lazar, G. Pacifici, “Joint Scheduling and Admission Control for ATS-Based Switching Nodes”, Proceedings of Sigcomm 92, Baltimore MD (Aug 1992)

[JaShZh92] S. Jamin, S. Shenker, L. Zhang, “An Admission Control Algorithm for Predictive Real-Time Service”, Proceedings of 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego CA (Nov 1992)

[Knightly92] E. Knightly, “Galileo: An Object-Oriented Simulator for Real-time Networks”, M.S. Project, University of California, Berkeley CA (Dec 92)

[MorGus92] M. Moran, R. Gusella, “Network Support for Interactive Multimedia”, Proceedings of 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego CA (Nov 1992)

[Partridge93] C. Partridge, personal communication (Jan 1993)

[Partridge92] C. Partridge, “A Proposed Flow Specification”, Request For Comments, Network Working Group, RFC 1363 (Sep 1992)

[RanVin91] V. Rangan, H. Vin, “Designing File Systems for Digital Video and Audio”, Proceedings of the 13th ACM Symposium on Operating Systems Principles, Monterey CA (Oct 1991)

[ZhVeFe92] H. Zhang, D. Verma and D. Ferrari, “Design and Implementation of the Real-Time Internet Protocol”, IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, Tuscon AZ (Feb 1992)

[Zhang93] L. Zhang, personal communication (Jan-Mar 1993)