INTERNET-DRAFT
Expires: December 1999

J. Ott/C. Perkins/D. Kutscher
Universitaet Bremen/UCL/Universitaet Bremen

June 1999

Requirements for Local Conference Control
draft-ott-mmusic-mbus-req-00.txt

Status of this memo

Abstract

In a variety of conferencing scenarios, a local communication channel is desirable for conference-related information exchange between co-located but otherwise independent application entities, for example those taking part in application sessions that belong to the same conference. In loosely coupled conferences such a mechanism allows for coordination of applications entities to e.g. implement synchronization between media streams or to configure entities without user interaction. It can also be used to implement tightly coupled conferences enabling a conference controller to enforce conference wide control within a end system.

This document defines application scenarios and requirements for a coordination infrastructure that provides local coordination within a conferencing end system.

This document is intended for discussion in the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at confctrl@isi.edu and/or the authors.

1. Introduction

1.1. Background

In the Mbone community a model has arisen whereby a set of loosely coupled tools are used to participate in a conference. A typical scenario is that audio, video and shared workspace functionality is provided by three separate tools (although some combined tools exist). This maps well onto the underlying RTP [5] (as well as other) media streams, which are also transmitted separately. Given such an architecture, it is useful to be able to perform some coordination of the separate media tools. For example, it may be desirable to communicate playout-point information between audio and video tools, in order to implement lip-synchronisation, to arbitrate the use of shared resources (such as input devices), etc.

A refinement of this architecture relies on the presence of a number of media engines which perform protocol functions as well as capturing and playout of media. In addition, one (or more) (separate) user interface agents exist that interact with and control their media engine(s). Such an approach allows flexibility in the user-interface design and implementation, but obviously requires some means by which the various involved agents may communicate with one another. This is particularly desirable to enable a coherent response to a user's conference-related actions (such as joining or leaving a conference).

Although current practice in the Mbone community is to work with a loosely coupled conference control model, situations arise where this is not appropriate and a more tightly coupled wide-area conference control protocol must be employed (e.g. for IP telephony). In such cases, it is highly desirable to be able to re-use the existing tools (media engines) available for loosely coupled conferences and integrate them with a system component implementing the tight conference control model. One appropriate means to achieve this integration is a communication channel that allows a dedicated conference control entity to ``remotely'' control the media engines in addition to or instead of their respective user interfaces.

Different approaches have been developed in conferencing systems in the past ([2] [3]) and some general purpose system exist ().

1.2. Purpose

The purpose of this document is to present some common scenarios for conference end system coordination and to derive a set of useful requirements that help to clarify the purpose, the architecture and

the scope of a coordination infractructure with respect to the goals of this IETF working group.

The main issues in this discussion are:

- Which are the architectural and protocol definition relevant requirements?
- Is it useful to consider such a coordination facility to be of general use for arbitrary applications that require coordination of distributed entities or would a solution that concentrates on coordination of conferencing systems only be more appropriate?
- What will be the role of the MMUSIC working group concerning the development of an infrastructure and protocol definition for the mentioned applications?

1.3. Terminology for requirement specifications

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant Mbus implementations.

1.4. Definition of terms

- Conference

  The relationship between a set of human beings that are communicating with each other. In this document, the term is used for both tightly and loosely coupled [FIXME] computer based conferences.

- Participant

  A (typically human) being in a conference.

- Member

  The system, including all software and hardware components, that is used in a teleconference to represent a single participant.

- End system

  A host or a set of locally interconnected hosts[1] that is used as an interface to a teleconference by a single participant.

---

[1] In this document, we use the term ``end system'' as a synonym for ``host'' in the simplest case. We do not want to ex-

clude, however, that the local system that serves one participant may be composed of several ``hosts'' in the Internet sense.

The end system runs all the required conferencing software (e.g. media agents, session directory, and a confernce controller). End system and software together constitute a member in each of the conferences a user participates in.

- Conference controller

  A dedicated application running on an end system that implements a horizontal conference control protocol through which it interacts with conference controllers on other end systems to implement conference control mechanisms and conference policies. The conference controller constitutes the electronic representation of its (human) user and her actions with respect to conference(s) as a whole (rather than with respect to individual media sessions).

- UCI

  A universal communication identifier of a person. This may be the e-mail address of an individual (or some other globally unique identifier) that is part of the information to identify her within a conference but can also be used to invite her via the Session Initiation Protocol (SIP) [6] protocol.

- Presence

  A presence corresponds to a person (identified by a UCI) being ``logged in'' at an end system and available for conferencing, i.e. a presence may be identified by the pair of a user's UCI and the respective end system's identification (such as a host name). A presence of a user may appear in many conferences (see below).

- Appearance

  An instantiation of a user's presence actually participating (i.e. appearing) in a conference is referred to as an appearance. There is a one-to-one correspondence between appearances and members.

- Application session (AS), Session

  The set of media agents/applications that act as peers to each other within a conference. For real-time data, this generally will be an RTP session [5]; for other application protocols, other horizontal protocols may define their own type of session concept. Possible synonyms are ``application group'' or ``media agent group''.

- Application instance, application entity, media agent

A program instance taking part in an application session for a conference participant. There can be more than one instance of

the same program in one session, there can also be more than one instance in different sessions.

2. Scenarios

In this section we present a few sample scenarios for the application of an coordination infrastructure.

2.1. Conferencing End System

We consider two different kinds of conferencing models:

- Loosely coupled conferencing [4], realized with light-weight sessions without explicit membership and conference control mechanisms.
- Tightly coupled conferencing where explicit membership and conference control mechanisms are applied. Enforcement of conference control can be provided by a ``conference controller'' -- a dedicated control component of an end systen.

2.1.1. Loosely Coupled Conferencing

A typical end system for loosely coupled conferencing consists of a set of media tools and (usually) a session directory for conference discovery and/or invitations. When joining a conference the session directory launches the media tools with certain parameters for multicast/host address, port number and codec details as announced in the session decription (SDP [5]). The tools can be controlled individually by the user, e.g. they can be terminated, suspended and their configuration can be changed.

After the media tools have been lauched there is no way to control their behaviour. If a common communication infrastructure existed within an end system several additional services would be possible:

- Coordinated termination/suspension

  Media tools could be terminated in an orderly fashion at the end of a conference upon receiving a ``quit''-message.

- Automatic source selection

A video tool could automatically display the video picture of the currently talking participant if an audio tool had the

possibility to share this information to other media tools. Lip synchronization would be another example in this category.

- Remote control of media tools

  Certain functions like muting/unmuting sources or configuring communication parameters do not necessarily have to be initiated directly via each tool's graphical user interface. Instead a controlling tool could integrate common functions and distribute control messages to the media tools.

- Failure detection

  Abnormal termination of media tools could be detected and dealt with appropriately. By sending regular ``alive'' messages entities can notice the existence of other entities and by not receiving further messages continously they can detect failure situations.

2.1.2. Tightly Coupled Conferencing

In a tightly coupled conferencing setting all of the aforementioned features will also be useful but there will be additional (conference wide) control aspects that will be highlighted in the following.

Explicit membership and conference control are usually implemented by conference controllers that are components of end systems. Conference controllers can maintain a conference state by applying a horizontal conference control protocol, e.g. SCCP [6].

Integrating a conference controller into an end system requires a local communication infrastructure that needs to provide some additional services compared to the loosely coupled scenario:

- Capability exchange

  One task of a conference controller can be to gather properties of the local end system concerncing supported media types and codecs and to enter a capability negotiation phase before setting up a conference. This would require the controller to obtain a capability description from each media tool and to later inform the end system of the result of the capability negotiation.

- Floor control

  One aspect of conference control in tightly coupled conferences is to formally control who is allowed talk (or send other media data) at a given time. A conference controller has to delegate

floor request from the local end system to the conference control session and to enforce the floor control at its local end system by sending appropriate control commands via the local

coordination infrastructure.

2.1.3. Summary of Requirements for local Coordination in a Conferencing End System

The scenarios described here assume end system composed of different entities using a local coordination infrastructure: There are open groups of entities, flexible configuration of entity sets, and user interaction. In detail this means:

Local coordination involves a widely varying number of entities: some messages may need to be destined for all local application entities, such as membership information, floor control notifications, dissemination of conference state changes, etc. Messages may also be targeted at a certain application class (e.g. all whiteboards or all audio tools) or agent type (e.g. all user interfaces rather than all media engines). Or there may be any (application- or message-specific) subgrouping defining the intended recipients, e.g. messages related to media synchronization. Finally there will be messages that are directed to a single entity, for example, specific configuration settings that a conference controller sends to a application entity or query-response exchanges between any local server and its clients.

Furthermore, messages exchanged between application entities may have different reliability requirements (which are typically derived from their semantics). Some messages will have a rather informational character conveying ephemeral state information (which is refreshed/updated periodically), such as the volume meter level of an audio receiver entity to be displayed by its user interface agent. Certain messages (such as queries for parameters or queries to local servers) may require a response from the peer(s) thereby providing an explicit acknowledgment to the application. Other messages will modify the application or conference state and hence it is crucial that they do not get lost. The latter type of message has to be delivered reliably to the recipient, whereas message of the first type do not require reliability mechanisms at the Mbus transport layer. For messages confirmed at the application layer it is up to the discretion of the application whether or not to use a reliable transport underneath.

The communication model varies for different applications: In some circumstances, e.g. when a controller requests capability descriptions from an entity, a request/response scheme is applied whereas in other situations, e.g. when an audio tool signals the beginning of a new talkspurt, frequently sent indications are used.

In some cases, application entities will want to tailor the degree of

reliability to their needs, others will want to rely on the underlying transport to ensure delivery of the messages -- and this may be different for each message.

Finally, accidental or malicious disturbance of communications through messages originated by applications from other users needs to be prevented.

The requirements that can be obtained from this are:

- reliablity (sometimes): failures (e.g. of entities) must be detectable
- scalability (concerning number of entities)
- security (authentication and encryption)
- well-defined PDU set (to allow for interoperability)
- extensibility (concerning PDUs and semantics)
- efficiency (small overhead, low latency)
- simplicity (easy to implement)

2.2. Media Gateway System

The following scenario describes a media gateway constituted of different entitities using a local coordination infrastructure. It is simliar to the conference controller scenario. The main difference is that usually direct user interaction is not required.

If we assume a media gateway to be composed of several distinct processed that require coordination, there is also a need for a coordination infrastructure that allows for reliable and unreliable communication within a group of entities. Since a media gateway is usually supposed to run without user interaction this example can impose even stronger requirements for reliability and security.

2.3. Wide Area Coordination

Wide area, i.e. non local, coordination is a totally different scope and is not considered in detail here.

3. Summary of Requirements

Taking the previous scenario desriptions into account the following requirements for local coordination infrastructure can be summarized:

Requirement	Loosely Coupled Conf.	Tightly Coupled Conf.	Gateway
Reliability	somtimes	sometimes	yes
Scalability	yes	yes	yes
Security	sometimes	sometimes	sometimes
Well-defined PDU set	yes	yes	yes
Extensibility	yes	yes	yes
Efficiency	yes	yes	yes
Simplicity	yes	yes	yes

Table 1: Matrix of requirements

Most of these requirements are important for all applications. Therefore a common coordination infrastructure for those scenarios is suitable as long as application specific extensions and specializations are possible.

4. Conclusions

Different applications of multimedia conferencing benefit from a suitable coordination infrastructure. To allow for use in different application scenarios with different configurations it is convenient to seperate mechanisms (transport issues, protocol syntax etc.) from semantics (concrete control command sets, parameters, policies).

Such an infrastructure will nevertheless not be of general use because of the specific protocol requirements mentioned above.

5. Authors' Addresses

Joerg Ott <jo@tzi.org>
Universitaet Bremen, TZI, MZH 5180
Bibliothekstr. 1
D-28359 Bremen
Germany

voice +49 421 201-7028
fax +49 421 218-7000

Colin Perkins <c.perkins@cs.ucl.ac.uk>
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom

Dirk Kutscher <dku@tzi.org>

Universitaet Bremen, TZI, MZH 5160
Bibliothekstr. 1
D-28359 Bremen
Germany
voice +49 421 218-7595
fax +49 421 218-7000

6. References

[1] S. Bradner, ``Key words for use in RFCs to Indicate Requirement Levels'' RFC 2119, March 1997

[2] M. Handley, I. Wakeman, J. Crowcroft, ``The Conference Control Channel Protocol (CCCP): A scalable Base for Building Conference Control Applications''

[3] H. Schulzrinne, ``Dynamic Configuration of Conferencing Applications using Pattern-Matching Multicast''

[4] Mark Handley, Jon Crowcroft, Carsten Bormann, ``The Internet Multimedia Conferencing Architecture,'' Internet Draft draft-ietf-mmusic-confarch-00.txt, Work in Progress, February 1996.

[5] M. Handley, V. Jacobson, ``SDP: Session Description Protocol'', RFC 2327, April 1998

[6] C. Bormann, J. Ott, C. Reichert, ``Simple Conference Control Protocol'', Internet-Draft draft-ietf-mmusic-sccp-00.txt, work in progress, June 1996