Nfaf - a Flexible, Modular Session Description Framework

Sarah Bell

Distributed Systems Group, ART
BT Research Laboratories
Martlesham Heath
Ipswich IP5 3RE
England

Contents

- Abstract
- Introduction
- Requirements Not Met by SDP
- Nfaf Outline
- Examples Using Nfaf
- Nfaf Syntax in More Detail
- Conclusions
- Future Work
- Glossary
- References
- Appendix

Abstract

We present a mechanism for describing the many types of real-time service expected to become widely available over the Internet as available bandwidth increases and multicast becomes prevalent. This mechanism, Nfaf, is designed to be as flexible and extensible as possible, allowing these varying services to be described. Its modular structure separates the various attributes required to describe a service into self-contained modules, also providing a separation between the user-oriented attributes of the service and the technical details which should be hidden from the user. It is this modularity that allows flexibility and extensibility, permitting services of simple or complex structure to be described, containing media or data streams of many formats, transported with many mechanisms. Key also is the ability to include policies within the modules to describe Quality of Service, security and charging. The framework is suitably generic to allow for the addition of new functionalities at a later date, such as the inclusion of complex metadata or emerging data formats and protocols. The self-contained modules allow for much flexibilty in the distribution of the session description, such that a very large description need not be transported complete.

Nfaf allows a complete description of the technical requirements for a real-time service. As such, it could be used be an advanced middleware to construct a bespoke application for handling the service from a set of components [M3 ref].

Nfaf is to be implemented using XML and forms part of a larger body of work looking at all aspects of real-time service discovery for the Internet, including announcement distribution and advanced session directory tools.

Introduction

Real-time audio and video enhanced services are becoming more common on the Internet. There are 2 commonly used methods of delivering these services, via the World-Wide Web or the MBone, a multicast overlay to the Internet. Many content providers such as record companies, film studios and radio stations offer live or on-demand multimedia content. The MBone tends to have more "academic" content, such as college lectures and technical conferences, as most of the institutions connected to it are of an academic or research nature.

There are several problems associated with the area of Service Discovery:

- service/content provider must provide a description of the service that adequately describes the features of the service and provides information on how to participate.
- this description must be distributed in an efficient, reliable and timely manner to potential users.
- the user must be able to view the descriptions of the available services in order to choose one to take part in.
- the joining of the service must be as seamless as possible so that that all users, regardless of technical know-how, are able to participate in the most complex of services.

This paper focuses on the first problem - describing the whole raft of real-time multimedia services, ranging in complexity

from a simple video conference to a multi-roomed virtual world, which we expect to become widespread on the Internet.

There are 2 methods for session description used on the Internet - Session Description Protocol is a standard used by the MBone community and will be discussed in detail below. The other method is no method, commonly used by WWW-based guides of real-time content such as RealNetworks' Timecast and Yahoo's Net Events. The format of the guides and the descriptions is arbitrary and the services listed are often limited to those using proprietary technology. It is not possible to perform the equivalent of a search engine query for real-time services.

There are several technologies available for describing the presentation of various types of multimedia content, including Microsoft's CDF and the W3C's SMIL. CDF describes 'channels' or sets of documents that can be pushed, pulled or streamed to the user. SMIL allows the description of the layout, synchronisation and presentation of media streams to be played within a SMIL-player. These techniques do not fulfil the requirements we have for session description (see below).

Session Description Protocol, SDP, is a simple text-based protocol for describing real-time services. It was developed as part of a suite of tools for multimedia conferencing over the MBone, including Session Announcement Protocol, Session Invitiation Protocol, the Session Directory tool SDR and the media tools Vic, Vat and Wb (figure 1). Although it is well suited to describing the very simple services that are common on the MBone today, perhaps containing 2 or 3 media streams, it is woefully inadequate for describing the complex and varied services that we anticipate being created as multicast becomes more widespread.

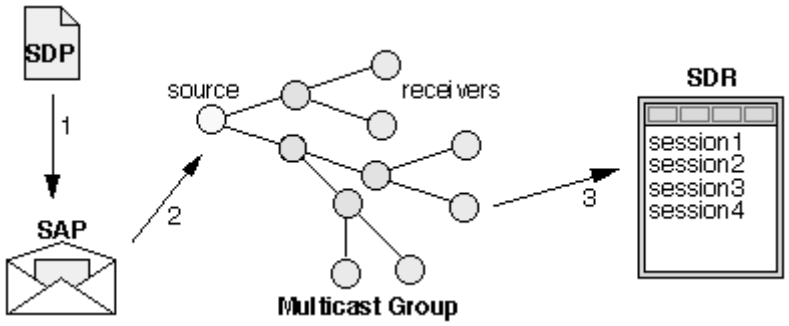

figure 1 - current multicast session announcement mechanism

Requirements Not Met by SDP

SDP was designed to describe fairly simple conferencing-type services, and it works well for this purpose. However its suitability for describing such multicast RTP streams and simple services leads to limitations in describing any other real-time services and new and emerging technologies in the areas of clients applications, transport mechanisms and protocols. Most of the limitations can be roughly divided into 2 areas: limited options for the service attributes and inability to provide complex structure.

Limited Options

For many of the attributes in an SDP descriptions, the available options are limited and 'hard-wired', meaning that extensibility is difficult.

- The media payload formats are restricted to RTP Audio/Video Profiles or 2 UDP application formats. Any stream using a different format cannot be described using SDP. A simple example is that of a media stream embedded in a web-page using a viewer plug-in - access to the page is via HTTP so the URL of the page cannot be described as the connection address using SDP.
- The transport mechanisms of the media streams are restricted to RTP and UDP - which means that any stream not transported using such a mechanism cannot be described using SDP.
- Simple layering of streams is supported but the only attributes that can be specified are the number of layers and the base multicast address. More complex addressing for layered multicast cannot be supported although such addressing will be a common requirement.

Lack of Structural Complexity

- An encryption key can be described for the entire session or individual streams, but there is no means of describing more complex security policies or mechanisms.
- The support for Quality of Service, at both the session and stream level, is limited to a bandwidth attribute ie. the amount of bandwidth required to receive the session or stream. Again, this does not allow for the description of more complex policies or mechanisms, such as prioritisation of streams or the use of reservations.

- Timing information is only applicable to entire sessions, not to individual component streams, so it would be impossible to describe a session with one stream that starts after another, for example.
- There is no support at all for any type of charging policies or mechanisms, either for entire services or the individual streams.

Other Problems

- The flat structure of an SDP decription means that it cannot be used to describe complex hierarchical sessions - sessions containing multiple subsessions. As sessions become larger, with more and more streams within, groups of the streams can emerge where the streams within a group share some common attributes, such as content. These groups are subsessions within the main session and can reflect the real-life structuring of the event. An example of this can be see in figure 3 (below). Subsessions allow the user to see this structure more clearly and to easily join only the subsessions that interest them. SDP cannot be used for this type of grouping - a very large session would have to be expressed merely as a long flat list of media streams.
- Information that is aimed at the user, such as a description of the session and contact details for the owner, is mixed up with technical information like data formats and IP addresses.
- Metadata about the session can be included using the pre-defined attributes a=cat:<*category name*> and a=keywds:<*keywords*>, but their use is not widespread and current tools ignore them. Without the use of metadata, it becomes much harder to search and categorise session descriptions - useful techniques for coping with large volumes of data.
- This flat structure also means that if any changes to the description need to be made, either before or during the lifetime of the session, then the entire description has to be replaced.
- A complete session description must be distributed - there is no ability to send minimal information which is extended before the session starts.

Nfaf Outline

In order to overcome these problems, a new type of session description mechanism has been designed, with the primary motives of flexibility and extensibility - New-fangled Announcement Framework, Nfaf. The key to Nfaf is the use of self-contained modules to describe the attributes of the session. There are 3 types of module, each containing related attributes for the session: base, media and option.

Base Module

The attributes contained in this module are aimed at the user - this is the information that the user needs in order to decide if they are interested in the session. Attributes include session name, short description, owner information, timing for the entire session, links to web-sites and some information about requirements for the entire session, such as 'is it encrypted' or 'does it have separate or bundled audio and video'. These requirements are not fully described in this module but provide enough information to help the user make a decision.

Media Module

This module contains the technical details about an individual stream, most of the information required by the user's machine to actually connect to that stream. This obviously includes such information as connection address and port and media format or client. It also contains timing information for the individual stream, allowing it to have separate timing within that of the session as a whole. The stream can also have its own description and owner information.

Option Module

The Option module is usually used to contain policy or mechanism information relating to security, charging, QoS or metadata IF required by the session or stream. An option module can be attached to a base module or a media module, depending on the desired scope and type of the option. A module for stream-level QoS data could describe the attributes associated with a layered encoding, whereas one for session-level QoS could contain rules for prioritisation of the various streams within the session if resources are limited.

Modular Architecture of Nfaf

The use of self-contained modules to contain various parts of a complete session description allows Nfaf to have a great deal of flexibility in many areas. Complex session structures can be reflected in the session description as the Nfaf modules can be linked into a hierarchical tree-like structure. This gives the end-user an insight into the session structure, allowing them to easily join subsessions of interest, and also allows the session source (the owner or creator of the session event) to build complex sessions from simpler subsessions.

Modularity also allows Nfaf to be extensible, so that emerging and future technolgies and functionality can be included in services and adequately described. The framework is suitably generic, and no data types are actually 'hardwired' into it, that

the addition of new attributes or new values for existing attributes will not be difficult (SDP has many of its data types hard-wired, so is not extensible at all). The use of the option module to add extra functionality to the session means that it is easy to describe QoS policies, security policies etc.

The distribution of session description is a problem that is not within the scope of this paper. However, the modularity of Nfaf should ensure that many types of distribution mechanism can be used for Nfaf descriptions. The self-contained modules within an Nfaf session description can be distributed together or separately, depending on the requirements of the chosen distribution mechanism. For extremely large descriptions, it may be more efficient to distribute only the top base module of each description, so that the user only has to retrieve the rest of the description's modules if they are interested in participating in that particular service. Issues such as the reliability and security of session description distribution are also not within the scope of this paper. Nfaf is entirely independent of the session description distribution mechanism.

Finally, the modular architecture of Nfaf allows the separation of user-oriented and technical information about the session. The information that the user requires in order to make a decision about joining the service is contained within the base module for the service (or the base modules for any of the subsessions). This information is non-technical in nature and provides the user with details about the service such as it's title and owner. The timing of the service and some information about the streams within are also provided, but not in great detail. The technical information required by the user's machine when joining the session, such as connection addresses, data formats and precise timing, is kept entirely separate in the media and option modules. This information is not required by the user and should be hidden as much as possible.

This separation of information, and the ability to distribute modules of the session description independently, means that precise details of the technical requirements of the session do not have to be specified when the session is initially announced. The session owner can distribute the base modules for the session, providing the initial advertising of the session's existence at some point in the future. The technical details can then be distributed at a later time, possibly close to the time the session is due to start. At this point, interested users can then retrieve the information they need to actually join the session. There are several advantages with this method of increasing the detail of session information. Firstly, it is a more efficient means of distributing the information - why keep sending the information to a multicast group or store it in a database until you absolutely have to. Also, the technical requirements of the session may vary from the time it is initially announced until it actually commences. This method means that the decisions about actual transport mechanisms, QoS policies, security etc. can be delayed so that fewer changes must be made. As an example, at the time of the initial session announcement, the session owner has little idea of the size of the potential audience. If users express their interest to the owner, based on their seeing that initial announcement, the owner can then use appropriate mechanisms for distribution of the session's streams.

Examples Using Nfaf

The modules described above can be used to build session descriptions for many types of real-time session, ranging from the most simple to extremely complex.

Simple Video On-Demand

The most simple example is that of a simple video on-demand service, which contains just a base module and single media module, as video and audio are bundled together (figure 2).

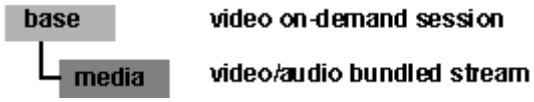

figure 2 - simple video-on demand

The base module contains everything the user needs to know about the video-on demand session, whereas the media module contains the technical details. The user-level information can be displayed in a session directory tool, allowing the user to search and browse through it, while the technical details remain hidden and are used if the user decides to join the service.

Multi-Track Conference Event

This scenario is an example of the use of subsessions to group streams together that share common attributes (figure 3). The conference is an event that consists of 3 subsessions, each of which shares some common attributes. The track subsessions each contain a video and an audio stream, with a session-level QoS policy determining the prioritisation of the streams if resources are scarce. The panel discussion subsession contains a bundled audio and video stream, which is layered. The details of the layering can be found in the media-level QoS policy attached to the media stream. The entire session has 2 option modules, one describing the charging policy and the other containing some metadata about the session content. The scope of these modules is the entire session and all its subsessions (unless overwritten by an option module of the same type

at a lower level).

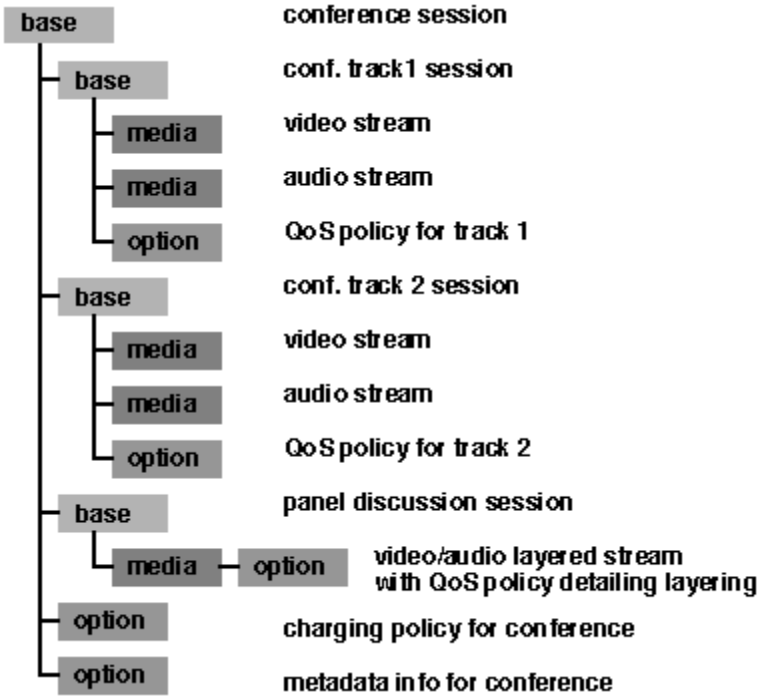

figure 3 - multi-track conference event

In this example, there is a charging policy that covers the entire conference event. This reflects the real-world practice of charging a registration fee for the conference, which once paid allows the attendee to participate freely in the various activities at the conference. All of this behaviour can be encapsulated in this virtual conference event using the Nfaf for session description.

The use of subsessions to divide the content of the event allows the structure of the session description to reflect that of the event or service itself. From the user's point of view, they can be made aware (via an appropriate session directory tool) that the session has such a structure and it is made clear that the user can participate in parts of the sessions (individual subsessions or even streams) without joining the entire session.

Nfaf Syntax in More Detail

Although the exact syntax of Nfaf is still evolving, the requirements for particular attributes and the range of values of those attributes have been determined. It is expected that Nfaf will be implemented in XML (see future work) but is currently uses a fairly simple text-based syntax as described in the example below.

The following is the Nfaf syntax that corresponds to the session described in figure 2, a very simple video on-demand session with a bundled media stream. The line numbers have been included purely for clarity.

```
1   (
      version=(nfaf1)
      type=(base)
      id=(00000001)
      info=(title="Demo Session")
5     source=(name="Sarah Bell" email=sbell@blah.com)
      media=(video=(client=RealPayerG2))
      time=(length=10m repeat=continuous)
      category=("Entertainment" "Comedy" "Alternative")
      options=(none)
10    modules=(m=00000002)
    )
    (
      version=(nfaf1)
      type=(media)
15    id=(00000002 00000001)
      media=(video=(client=RealPlayerG2)
      connection=(http://www.blah.com/demo.html)
```

```
    time=(length=10m)
)
```

Lines 1-11 are the base module for the description, lines 12-19 the media module. The modules can be distributed together or separately, depending on the requirements of the distribution mechanism employed.

Base Module

The attributes and values required in the base module are as follows.

- the version of Nfaf used in the module
- the type of module
- a unique identifier for that module. If the module is part of a subsession, then an identifier for the parent session's base module will also be specified.
- session title and descriptive text
- name and email/phone for the session source or owner
- list of media types in component streams, specifying client applications or codecs required
- start/stop time or length of entire session
- service/application specific attributes, such as categorisation
- list of options for the session, such as charging and security
- pointers to further base, media or option modules

Media Module

Similarly for the media module, the following attributes may be specified. It is worth noting that there may be repition of some of the data contained in the base module - this is to allow as much independence between the modules as possible.

- the version of Nfaf used in the module
- stream title and some descriptive text
- name and email/phone for the stream source
- connection address, port number and scope
- timing, wither start and stop times or length
- service/application specific attributes
- pointers to optional modules

Option Module

The syntax of the option module of course varies depending on the type of information or policy that must be described. That for a session and media-level QoS policy has been defined, those for security, charging, metadata etc. will be defined in due course. The following is for a session-level QoS policy.

- identifier of session to which policy applies
- list of QoS rules eg. a priority stating that video quality is more important for this session than audio
- service/application specific attributes

A media-level QoS policy would contain the following attributes.

- identifier of media stream to which policy applies
- list of mechanisms that will be required to receive the stream eg. layering
- timings of the policy, when it is required by and when it expires
- service/application specific attributes

The Nfaf for the multi-track conference event appears in an appendix to this paper.

Conclusions

Nfaf is a flexible and extensible mechanism for describing many types of real-time multimedia session, both interactive and passive. Its modular nature allows related session attributes to be grouped together easily and will allow announcements to be distributed in a piecemeal manner if required. The use of subsessions to split up complex sessions can also be described easily using this modular approach. It is hoped also that the modular nature will also make Nfaf easy to extend when new requirements for attributes, policies, application types, mechanisms etc. are discovered.

Nfaf is currently a text based protocol - the actual syntax is less important than the semantics. Work has started into looking at using XML to write Nfaf announcements.

Future Work

As previously mentioned, the next aim for the Nfaf work is to implement Nfaf using XML. XML is a standard for producing structured documents - it is actually a meta-markup language that allows markup language to be easily built for different purposes, hence we can define Nfaf as an XML-based markup language for session description. The structure of the markup language can be described either at the beginning of the document itself, or in an external file referenced by the document. This provides a great deal of flexibility and should allow Nfaf session descriptions to be extended easily where the need arises. Also there already exist a number of XML parsers, validators and editors, which will make it easier to handle Nfaf session descriptions.

Nfaf is currently being used to describe fairly simple loosely-coupled sessions. It could be extended to provide support for tightly-coupled sessions, incorporating access control and invitation mechanisms. We are also investigating the problems of changing the session attributes during the lifetime of the session, which raises a number of questions. What sort of attributes might be changed and who has the authority to change them? How would the changes be communicated to existing and future participants in the session? The requirements for these mutable sessions have already be raised by services involving virtual worlds and communication between the participants in the world.

Nfaf is one part of a larger body of work looking at several of the problems involved with real-time service discovery on the Internet. The features that Nfaf offers over and above SDP must be incorporated into an advanced session directory tool, so that the user may take advantage of them. Also there is work being carried out in the area of announcement distribution, developing techniques for flexible and efficient distribution over both intranets and the Internet.

Glossary

- announcement - a session description that has been distributed to potential users.
- client - the user's machine or computer, where the user is the receiver of the session.
- media application - any application on the user's machine that is capable of receiving media data (from the network). Examples of media applications are RealNetworks' G2 Player, Vic and Vat.
- multicast - an efficient mechanism for transporting data to multiple receivers, where the data is duplicated within the network rather than at the source.
- QoS - a measure of how well the entire system achieves the user's subjective requirements
- service - see session
- session - a collection of media streams that share common attributes for a specified period of time.
- session description - the attributes that describe a session fully enough for a user to join that session, so must contain technical information as well as information for the user.
- session directory - a tool that retrieves and displays session announcements, such that the user can look at some of the attributes of each session in order to make a decision whether to join the session.
- subsession - a session within a session, a collection of streams that share some attributes within the session.

References

- S. Ing, "Simplifying Real-Time Multimedia Application Development Using Session Description", submission to IS&N 99, April 99
- Session Description Protocol - SDP draft (draft status implies work in progress) - ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-07.txt
- Session Announcement Protocol - SAP draft (draft status implies work in progress) - ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-sap-00.txt
- Session Directory Rendezvous - SDR - http://mice.ed.ac.uk/mice/archive/sdr.html
- Session Invitation Protocol - SIP draft (draft status implies work in progress) - ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-sip-10.txt
- Channel Definition Format - CDF
- Synchronised Multimedia Integration Language - SMIL - http://www.w3.org/AudioVideo/
- Extensible Markup Language - XML - http://www.w3.org/XML/

Appendix

The following is the Nfaf syntax corresponding to the session described in figure 3.

```
( # conference session
  version=(nfaf1)
  type=(base)
  id=(0001)
  info=(title="Example Conference Event")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
```

```
  media=(video=(client=RealPlayerG2) audio=(client=RealPlayerG2))
  time(start="09:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
  options=(oc=0013 om=0014)
  modules=(b=0002 b=0006 b=0010 oc=0013 om=0014)
)
( # conference track 1 subsession
  version=(nfaf1)
  type=(base)
  id=(0002 0001)
  info=(title="Conference Track 1")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2) audio=(client=RealPlayerG2))
  time(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
  options=(osq=0005)
  modules=(m=0003 m=0004 osq=0005)
)
( # video for track 1
  version=(nfaf1)
  type=(media)
  id=(0003 0002)
  info=(title="Conference Track 1 video")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(type=live client=RealPlayerG2))
  connection=(226.0.0.100/1000)
  time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)
( # audio for track 1
  version=(nfaf1)
  type=(media)
  id=(0004 0002)
  info=(title="Conference Track 1 audio")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(audio=(type=live client=RealPlayerG2))
  connection=(226.0.0.101/1001)
  time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)
( # session qos for track 1
  version=(nfaf1)
  type=(option-sqos)
  id=(0005 0002)
  mandatory=(0004)
  optional=(0003)
)
( # conference track 2
  version=(nfaf1)
  type=(base)
  id=(0006 0001)
  info=(title="Conference Track 2")
  source=(name="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2) audio=(client=RealPlayerG2))
  time(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
  options=(osq=0009)
  modules=(m=0007 m=0008 osq=0009)
)
( # video for track 2
  version=(nfaf1)
  type=(media)
  id=(0007 0006)
  info=(title="Conference Track 2 video")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(type=live client=RealPlayerG2))
  connection=(226.0.0.103/1003)
  time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)
( # audio for track 2
  version=(nfaf1)
  type=(media)
  id=(0008 0006)
  info=(title="Conference Track 2 audio")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
```

```
  media=(audio=(type=live client=RealPlayerG2))
  connection=(226.0.0.104/1004)
  time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)
( # session qos for track 2
  version=(nfaf1)
  type=(option-sqos)
  id=(0009 0006)
  mandatory=(0008)
  optional=(0007)
)
( # conference panel discussion
  version=(nfaf1)
  type=(base)
  id=(0010 0001)
  info=(title="Conference Panel Discussion")
  source=(name="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2))
  time(start="11:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
  options=(none)
  modules=(m=0011)
)
( # bundled audio and video for panel discussion
  version=(nfaf1)
  type=(media)
  id=(0011 0010)
  info=(title="Panel Discussion Audio and Video")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(type=live client=RealPlayerG2))
  connection=(226.0.0.106/1010 policy=0012)
  time=(start="11:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
)
( # media qos policy for panel discussion video
  version=(nfaf1)
  type=(option-mqos)
  id=(0012 0011)
  mechanism=(layer=(base=226.0.0.106/1010 number=3))
)
( # charging policy for entire conference
  version=(nfaf1)
  type=(option-chg)
  id=(0013 0001)
  mechanism=(type=AAA)
  price=(fee=1000GBP)
  info=(location=htpp://www.aaa.net/)
)
( # metadata for entire conference
  version=(nfaf1)
  type=(option-mdata)
  id=(0014 0001)
  category=("Computing:Conference:Misc")
  keywords=("conference" "computing" "computer science")
)
```

Sarah Bell