A Scheme for Component Based Service Deployment

Steve Rudkin and Alan Smith

Distributed Systems Group, BT Laboratories,
Adastral Park, Martlesham Heath,
Ipswich, IP5 3RE, England
{srudkin, asmith} @jungle.bt.co.uk

Abstract. A dynamic and open service market should be characterised by the frequent appearance of new and diverse services. But in the internet environment, where a significant proportion of the functionality of new services is supported by Customer Premises Equipment (CPE), deployment can be an issue. How can such a diverse range of new services be easily deployed - particularly in a heterogeneous environment, like the internet, where different users have different terminal capabilities, and access bandwidths. This paper describes how the delivery, to the CPE device, of a recipe for dynamically building an application on the CPE device may be used to simplify the deployment of new services. In particular it shows how IBM's Bean Mark Up Language, an XML-based scripting language for describing a configuration of JavaBeans, can be used.

Keywords: Component, service creation, service deployment, session description, XML, BML.

1 Introduction

The ever increasing growth of the Internet is rapidly realising the most basic requirement of a universal service market – namely universal connectivity. At the same time, the end to end design principle of the Internet [1] is driving service intelligence to the edge of the network, and often to the customer premises equipment (CPE). As the capabilities of CPE devices (e.g. PCs, mobiles, set-top boxes) grow, we can expect a significant proportion of the functionality of new services to be supported by the CPE device.

In this paper, we regard a *service* as a function delivered to users through the execution of software components on one or more CPE devices, on an IP network and on a number of servers. We refer to the configuration of components on a particular machine (especially the CPE device) as an *application*. As our focus is on real-time services, we use the term *session* to refer to a particular instance of some real-time service (such as a conference, game, or TV program).

Many services share a great deal of functionality in common – both on the CPE device and on remote servers. For a number of years there has been increasing interest in the application of Component Based Software Engineering techniques to support the reuse of common software components in the development of new applications [2]. We are interested in the application of these techniques in the creation of new services. The creation of services should be a matter of describing the relationships between pre-existing components and adding session specific information such as: Internet addresses and media content. The creation of service instances should not be in the hands of specialist programmers, but rather the domain of service providers.

In a dynamic and open service market, we would expect to see the frequent appearance of new and diverse services. But, each new service may bring with it a deployment problem – how to distribute the necessary functionality to each CPE device? Whilst the new services would typically share some functionality with previously deployed services, we can also expect many differences. The heterogeneity of the Internet further complicates deployment. With varying terminal capabilities, access bandwidths, and intermittent connectivity, different service capabilities may be needed to suit the circumstances of different users.

This paper describes how the delivery, to the CPE device, of a recipe for dynamically building an application on the CPE device may be used to simplify the deployment of new services.

Previous work [3] describes how the Session Description Protocol (SDP) [4] can be used as a simple recipe for building a real-time application. For example it shows how the media fields could be used to select appropriate media processing components for configuration within the overall application. In this paper we extend this idea in a number of ways.

Our first extension is conceptual. We now consider a session description to be a description of an instance of a real-time service used by a service provider to advertise the associated service. For example an on-line training company might advertise a particular training session using a session description. The result is that a service creation environment can be split into a service definition environment (at the service provider) and a service instantiation environment (on the CPE device). The session description generated by the service definition environment is passed to the service instantiation environment, where it is used as a recipe for building the application.

Our second extension concerns the introduction of remote component services. Component services may be server-based (e.g. payment gateways, key distribution servers and media servers) or network-based (e.g. multicast or network QoS). We assume that a service may be composed from a number of component services as well as some functionality that resides on the CPE device (which partly acts to pull the various component services together into the overall service). Each component service may be offered by any service provider. In this way we seek to go beyond the retailer model [5] developed for TINA in order to produce a more open market place.

Our third extension, and main topic of this paper, concerns implementation. We show how we can use IBM's Bean Mark Up Language [6,7] as a way of defining the configuration of components that make up the application on the CPE device.

The paper is structured as follows. Section 2 discusses session descriptions. Section 3 sets the context for the work described in this paper by outlining the process we expect to see for creating and deploying new real-time services in an open market. Section 4 outlines the approach we have taken for dynamically building applications on the CPE device. Section 5 describes an example. Section 6 describes future work and section 7 concludes the paper.

2 Session Descriptions

On the Internet Mbone[1], a session directory tool (e.g. *Session Directory Rendezvous* (SDR) [8]) is used to advertise and discover multimedia sessions. The sessions are described using the Session Description Protocol (SDP). An SDP session description is entirely textual and consists of a number of lines of text of the form <type>=<value>. The type is always exactly one character and is case-significant, the value is a structured text string whose format depends on the type. An example of an SDP session description is given below. Note that the text in brackets are meant as explanations and should not appear in the actual description.

```
v=0 (protocol version)
o=srudkin  2890844526 0 IN IP4 132.146.107.67 (owner,
session identifier, version, network type, address
type, address)
s=SDP Seminar (session name)
i=A Seminar on the session description protocol (ses-
sion information)
u=http://www.labs.bt.com/people/rudkins (URI of de-
scription)
e=srudkin@jungle.bt.co.uk(email address)
c=IN IP4 224.2.17.12/127 (network type, network ad-
dress/TTL)
t=2873397496 2873404696 (start and stop time)
a=recvonly (media attributes set to receive only)
m=audio 3456 RTP/AVP 0 (media type, port number, trans-
port, media format)
m=video 2232 RTP/AVP 31
```

It can be seen that an SDP session description both identifies the application and defines the session specific parameters such as IP addresses to be used by the application. SDP uses the media field to declare the relevant media types and formats, and thereby implicitly identifies the application that should be used to participate in the associated session. In the case of the above example, SDR would launch the Mbone applications vat (for audio conferencing) and vic (for video conferencing). SDR can do this since it knows they support the specified formats.

[1] The Mbone is the part of the Internet that supports IP multicast which permits efficient many-to-many communication.

We are investigating replacing SDP with an XML [9] based session description framework. We believe that XML can be used to define a wider range of session specific parameters including QoS, charging and security policies. Moreover we show, in this paper, how the media fields of a session description can be replaced by a BML specification of the application(s) that are to be dynamically configured. BML is an XML-based scripting language developed by IBM for describing structures of nested and interconnected JavaBeans [10]. BML is directly executable, that is to say, processing a BML script results in an application program configured according to the script.

3. The Service Creation and Deployment Process

This section briefly describes the process of creating and deploying a new service. Figure 1 illustrates the main elements and their role in this process.

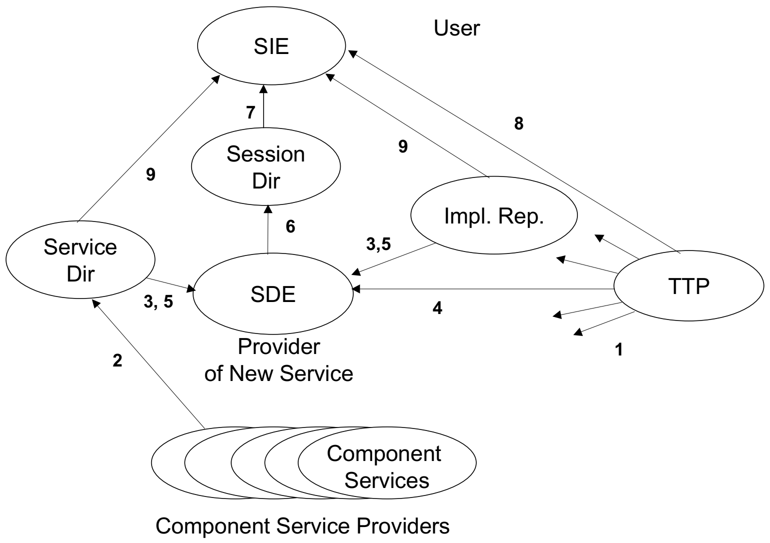


Figure 1: The process of creating and deploying a service

3.1 The Main Elements

Service Directory: A service directory is used by component service providers to advertise their component services to other service providers. It also allows users to download any client components which are needed as part of a third party service to interact with the component service. The service directory is outside the scope of this paper.

Implementation Repository: The implementation repository is a store of components that may be used in the implementation of applications that reside on the CPE device. These are components that are not part of an actual service, but which offer functionality in their own right (e.g. codecs, GUI elements, etc.). The implementation repository is outside the scope of this paper.

Service Definition Environment (SDE): This comprises an Application Development tool (for example supporting drag and drop development of components) combined with a tool for adding session specific characteristics (such as title of the session, purpose of the session, details of the session owner, IP addresses of the different media channels etc.) The practical use of the approach described in this paper depends on the existence of an Application Development tool that generates a suitable application description based on BML. In the absence of such a tool we have simply handwritten the BML for a number of component-based applications we have previously developed.

Session Directory: The session directory allows users to view details of current and future sessions and allows users and service providers to announce sessions. Currently we have implemented a complete multicast session directory that handles SDP session descriptions, and a very basic web-based session directory that handles the simple XML-based session descriptions described in this paper.

Service Instantiation Environment (SIE): This is resident on the User's CPE device. It is used by the Third Party Service Provider to build the required application and to launch it with the specified session parameters. The Service Instantiation Environment is the subject of section 4.

Trusted Third Party (TTP): It is the role of a Trusted Third Party to dynamically establish sufficient trust between two or more parties to give them the confidence to transact business. The Trusted Third Party issues and revokes certificates which are statements signed by the TTP about the certificate subject. Typically these statements include the subject's public key and identity, but they may also include higher value statements such as the subject's credit rating [11].

3.2 The Process

The process of creating and deploying a new service involves the following steps:

Bootstrap

1. The various parties obtain certificates from the Trusted Third Party.
2. The component service providers register their services with the Service Directory. This involves generating service offers describing the service itself, and any obligations on the users of the service (including for example payment details). The client components are passed to the Service Directory. It is a decision of the component service provider as to how much functionality is implemented in the client component and how much should be accessed remotely.

Service Definition

3. The Third Party Service Provider uses the Service Directory to find appropriate component services that may be used to support a new service they wish to offer. It also uses the Implementation Repository to find any functionality which will reside solely on the CPE device (e.g. codecs).
4. The Third Party Service Provider verifies, that the component service providers and the providers of components found in the implementation repository can be trusted, by obtaining the relevant certificates from the TTP.
5. The Third Party Service Provider downloads the client components for the component services, downloads any required components for the implementation repository and checks they are all correctly signed (using the public keys from the certificates that have just been obtained from the TTP).
6. The Third Party Service Provider builds an appropriate CPE application (using a suitable application development environment) and generates a BML application description (currently we are using conventional development and test to ensure the components work together). The Third Party Service Provider defines the necessary session specific parameters and advertises the whole session description (including the application description and session specific parameters) in the session directory.

Service Instantiation

7. The Customer downloads a session description from the Session Directory.
8. The Customer verifies with the TTP that the component service providers can be trusted.
9. The Customer downloads the relevant client components from the Service Directory and any other components from the Implementation Repository, checking they are correctly signed. The application can then be built on the CPE device and the relevant session specific parameters are instantiated. Service Instantiation is discussed in detail in the next section.

4 Service Instantiation Environment

Figure 2 shows the process for building applications dynamically on the CPE device; this is the Service Instantiation Environment. It also shows the inputs into the process. The inputs and the process are discussed in turn below.

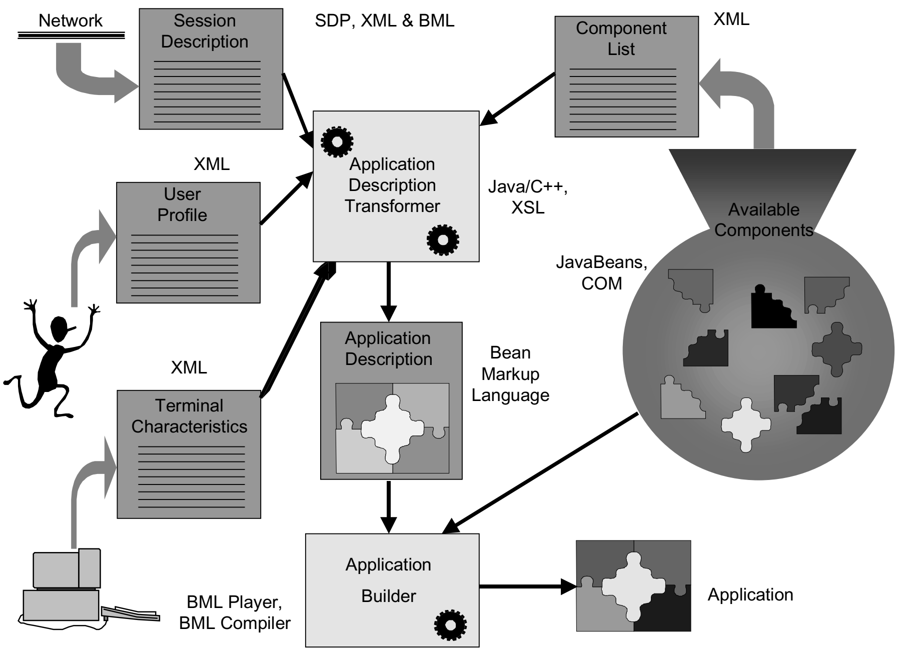


Figure 2: Dynamic application building process

4.1 Inputs into Application Building

There are several inputs into the application building process. All of these can be implemented in XML.
Session Description: The base input is the Session Description. A session description comprises user directed information (e.g. session owner, purpose of the session), session parameters (e.g. IP addresses) and an application description (in BML).
User Profile: This contains user preferences. For example, in the case of an aurally impaired user a preference may be to omit the audio component from an application and instead receive sub-titles. Whilst someone with unimpaired hearing might elect to receive the audio, but ignore sub-titles.
Available Components: Components can be stored on a user's own machine, or they can be downloaded from the network. Using components stored locally could offer the benefit of reducing time to start the application. Also, a local component may already have been paid for, whereas one to be downloaded may have to be rented (how components are charged for has not yet been addressed).
Terminal Characteristics: This has a critical influence on the application building process. A low resourced device is less able to show certain media-types. A mobile device will probably not be capable of displaying high resolution video. In the case of

a session containing audio and video the mobile device could still join the session, but would only instantiate an audio component.

Network Characteristics: This has a very similar effect to the terminal characteristics. For example, a GSM network is not capable of relaying high resolution video

4.2 Dynamic Application Building Process

The process starts with a user browsing a list of available sessions using a Session Directory tool. Once the user has decided to join the session the session is passed to the Application Description Transformer for the application building to start.

Application Description Transformer: This takes the Session Description as primary input. As described above the Session Description will contain details of the media-types that make up the session, connection information, QoS data, charging data, etc. In order to adapt the application, this process may also use the other inputs described in section 4.1 namely user profile, available components and terminal (and network) characteristics. There are currently a number of different options for implementing this process. The choice is basically between XSL transformations or code written in a conventional programming language. Currently implementations have been tried using XSL and Java. The output of this process is an **Application Description**.

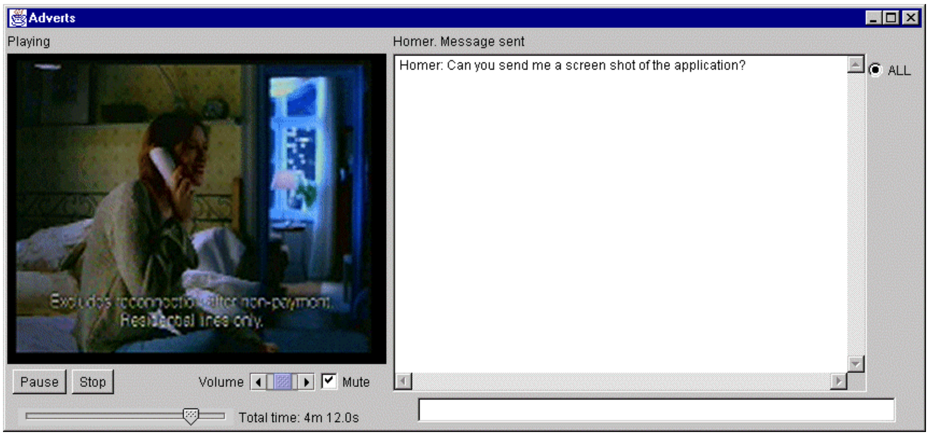
The application description uses IBM's Bean Markup Language (BML) from IBM's Alphaworks programme. BML is itself an instance of XML. It is used to describe the structure of a set of nested and interconnected beans. It is a language whose only functions are to describe how components relate to one another and how each of those components is configured. In fact Bean Markup Language is a bit of a misnomer as any Java classes can be included in the BML. It is just that special consideration is given to JavaBeans.

Application Builder: Once an application description has been produced it must be turned into a running application. For this stage we have used IBM's BML Player and compiler. The BML Player takes a BML script and produces a running program from it. The BML Compiler will compile a BML file into a class file which can then be run like a normal Java program.

4.3 Adaptation

Section 4.1 detailed the inputs to the application building process. The simplest form of adaptation will be to include or exclude particular media-types from the session and which components are selected in order to display those media-types. Examples include not selecting a media-type because of user preference or because of the type of terminal being used. Here the media-type will be removed from the BML. Presentation characteristics such as style of window objects used are also determined by the user profile. The session description could be extended to include alternative sources for data such as video at different bandwidths. A low resourced machine would select the video at the lower bandwidth.

5 An Example Implementation



Homer: Can you send me a screen shot of the application?

Figure 3 Audio-video-chat Application

As an example application we will describe an integrated chat and audio-video application (Figure 3).

This example combines two major components, namely chat and a JMF (Java Media Framework) implementation of Real Player. The Real Player component is used to access a remote component service streaming audio and video. There are a number of smaller scale components representing the GUI components, e.g. buttons and frames. The chat and the audio-video components have been used in other applications where they provide the sole functionality.

In the example implementation the only input to the Application Description Transformer is the Session Description; the other inputs remain to be implemented.

5.1 Session Description

```
      <?xml version="1.0"?>

      <session>

    <title>Adverts</title>
    <media-type id="audio-video">
  <address>sherekhan.jungle.bt.co.uk</address>
  <port>1545</port>
    </media-type>
    <media-type id="chat">
        <address>ferao.jungle.bt.co.uk</address>
        <port>1546</port>
    </media-type>
    <application-description>
<url>http://www.jungle.bt.co.uk/projects/internet-
```

```
middleware/bml/audio-video-chat.bml</url>
            </application-description>
   </session>
```

In this example a session description consists of two media-types audio-video and chat. Each media-type information includes the address of the server and the port number (other location information such as a URL could be provided instead). The media-type tags serve to scope the address data so that they can be matched with the appropriate placeholders in the BML. In practice the media-type information would include other data such as QoS parameters and user directed information. QoS parameters would be used in order to adapt a session, e.g. by excluding a particular media-type because the terminal is incapable of meeting the QoS requirements. The user directed information would include such things as the owner of the content or comments, e.g. that the audio-video is the directors cut.

The application-description tag includes an address where a BML file representing a description of the application can be found. Note that the BML file is generic description for a class of applications that implement a combined real player and chat. It does not include session specific information. These generic BML files can be stored in a cache so that they only need to be downloaded when first needed.

We have produced a Session Description Parser in Java. The parser takes the input session description in XML and creates a session description object. This is used to display user oriented content to the user. This parsing is done using the XML for Java parser from IBM. A DOM (Document Object Model) tree is created then a separate Java object is instantiated to handle each tag and transfers the data from the tag to the session description object. The user is then presented with the title of the session and a list of media-types in the session.

5.2 BML

The BML represents a class of application. The BML produced at application creation time includes place-holders for the session parameters. Rather than show the entire BML script, excerpts will be used to illustrate key points

```
          <?xml version=”1.0”?>

   <bean class=”java.util.Vector”>
            <add>
   <bean class=”java.awt.Frame” id=”mainFrame”>
                                                            <property name=”title”>
   <SD-session-name/>
        </property>
        <property name=”background”>
   <field target=”class:java.awt.Color”
   name=”lightGray”/>
                                                            </property>
```

At the topmost level of BML there is a Java Vector which contains all the JavaBean components in the application. The first of these is a Java Frame class. There follow a number of property setters. The second of these is standard BML; the background colour of the frame is set to light grey. The first property setter has a place holder for the session name; identified by the tag <SD-session-name>. This must be filled in with details taken from the session description.

BML Bindings

```
<bean source=”chatFrame”>
          <event-binding name=”window” filter=”windowClosing”>
<script>
                                                          <call-method          tar-
                                                       get=”chatClient” name=”quit”/>
</script>
          </event-binding></bean>
```

JavaBeans in BML are connected together via event-bindings. An event from one component is mapped to call a method or set a property on another. In this simple example the chatClient component responds to a windowClosing event from the chat-Frame component by closing.

5.3 Application Description Transformation

If the user elects to join the session, then the BML is retrieved from either the cache or from the address specified in the session description. The BML is then parsed and the place-holder tags replaced by the session specific data. This is again achieved using XML for Java. The “Element Handler” event based parser is used. An element handler is associated with each placeholder tag in the BML. Each time the placeholder is encountered the element handler is called. The placeholder tag is replaced by data from the session description object.

A local component cache is checked to see if the components are already loaded onto the user’s terminal. If not, they are downloaded over the network. Currently the version is not checked. In the future a scheme such as COM GUIDs could be used.

The initial three applications (chat, audio-video and the combined application) have been extended to include a payment component. If the session creator specifies that payment is required, then a payment component will be included by the application transformer. The amount to be charged for the session and the location of the payment gateway is taken from the session description and passed to the payment component. The user is then presented with the amount to pay and requested to provide credit card details. The payment component contacts a payment server for authorisation. Only when payment has been authorised will the other application components start.

We have also implemented an Application Writer using an XSL (XML Style Language) style sheet. The XSL processor (Lotus XSL) takes the session description XML file and XSL style sheet as inputs and outputs the complete BML file. The XSL style sheet contains snippets of BML which are output to the final BML file.

6 Future Work

Probably the most important task to complete is the definition of a session description DTD that is flexible enough to address a range of application types and service policies. The session directory will need to be adapted to handle these new session types. We are seeking to broaden the approaches to wiring giving alternative methods to BML.

On the server side, service creation would be improved if existing application development tools were securely coupled with the service directory, and implementation repository. Application developers should be able to securely drag and drop client components from the service directory and implementation repository and automatically generate BML describing the CPE application. It should be a simple further step for service providers to add session parameters and advertise the session description .

In the longer term we wish to explore the possibility of selecting and configuring components based on high level declarations of requirements of the kind described in a Taxonomy of Communication Requirements [12].

7 Conclusion

The use of a session description containing a recipe to dynamically build the CPE part of a new service offers a number of benefits.

- By using a session announcement to “deploy” a recipe for building the application, barriers to deployment are reduced.
- By overcoming deployment barriers, it becomes easier for service providers to offer innovative new services.
- By dynamically building the application from components on the client, only those components that are not available locally need to be downloaded.
- During dynamic construction, the application can be adapted to suit the local environment.

Acknowledgements

The authors would like to thank Ian Fairman for the contribution of his ideas to this work. We would also like to thank Kashaf Khan and Luca Tavanti for their implementation of components (under Eurescom project P925: Internet Middleware).

References

[1] B. Carpenter, Editor, Architectural Principles of the Internet, IETF Network Working Group RFC1958, June 1996, ftp://ftp.isi.edu/in-notes/rfc1958.txt

[2] Clemens Szyperski, Component Software - Beyond Object-Oriented Programming, Addison-Wesley; ISBN: 0201178885

[3] Sarom Ing, Steve Rudkin. Simplifying Real-Time Multimedia Application Develop ment Using Session Descriptions, IS&N 99, Barcelona, Spain, April 1999

[4] M. Handley. and M. Jacobson, SDP: Session Description Protocol. IETF MMUSIC Working Group RFC 2327 (1998). ftp://ftp.isi.edu/in-notes/rfc2327.txt .

[5] M. Bagley, I. Marshal, et al, The Information Services Supermarket - a trial TINA-C Design, TINA Conference, Melbourne Australia. Feb 1995

[6] Mark Johnson, ”Cover Story: Bean Markup Language, Part 1”, http://www.javaworld.com/javaworld/jw-08-1999/jw-08-beans.html .

[7] Mark Johnson, “Cover Story: Bean Markup Language, Part 2”, http://www.javaworld.com/javaworld/jw-10-1999/jw-10-beans.html .

[8] M. Handley, The Session Directory Tool (SDR) http://mice.ed.ac.uk/archive.srd/html

[9] Extensible Markup Language (XML) 1.0. W3C Recomm. http://www.w3.org/XML/

[10] JavaBeans, http://web2.java.sun.com/beans/docs/beans.101.pdf .

[11] T. Rea, High Value certification - trust services for complex eCommerce transactions, BT Technology Journal Vol 17 No.3 July 1999.

[12] P. Bagnall, R. Briscoe, A. Poppitt, A Taxonomy of Communication Requirements for Large Scale Multicast Applications, IETF Large Scale Multicast Applications Working Group RFC2729, December 1999, ftp://ftp.isi.edu/in-notes/rfc2729.txt .