Network Working Group
Internet-Draft
Expires: December 31, 2005

C. Boulton
Ubiquity Software Corporation
T. Melanchuk
Convedia
June 29, 2005

Media Server Request Protocol
draft-boulton-media-server-control-00

Status of this Memo

Copyright Notice

Abstract

This document describes a protocol for application deployment where the application logic and media processing are distributed. The framework uses the Session Initiation Protocol (SIP) to establish an application-level control mechanism between Application Servers and Media Servers.

The motivation for this protocol is to provide an interface suitable

to meet the requirements of a distributed, centralized conference system, as defined by the XCON work group of the IETF.

Table of Contents

1. Introduction

Applications are often developed using an architecture where the application logic and media processing are distributed. Commonly, the application logic runs on "application servers" whilst the media processing runs on "media servers". This document focuses on the protocol between the application server and media server. A detailed set of requirements for Media Server Control can be found in the 'Requirements for a Media Server Control Protocol' document[9]

Currently the document describes the model for media server control. Subsequent versions will build on the model and address the specific application requirements from [9]

While the primary motivation for the work is to meet the XCON requirements, there are many other application scenarios that require media processing services within a SIP centric network. Application developers want to continue to leverage SIP for establishing and managing media sessions, while only adding the protocol machinery necessary to allow application control of media server operation.

Current IETF transport device control protocols, such as megaco [7], while excellent for controlling media gateways which bridge separate networks are troublesome for supporting media-rich applications in SIP networks as they duplicate many of the functions inherent in SIP. Rather than relying on single protocol session establishment, application developers need to translate between two separate mechanisms.

Application servers traditionally use SIP third party call control RFC 3725 [12] to establish media sessions from SIP user agents to a media server. SIP, as defined in RFC 3261 [2], also provides the ideal rendezvous mechanism for establishing and maintaining control connections to Media Server components. The control connections can then be used to exchange explicit command/response interactions that allow for media control and associated command response results.

At first glance, it may appear that the session establishment procedures here look similar to MRCPv2 [6]. Like MRCPv2, the protocol described here uses SIP for resource location, leverages SIP for availability and redundancy, can tap into caller preferences [14], and, as this document will describe below, establish an application channel for the exchange of media processing commands.

However, the protocol entities of MRCPv2 and the protocol described herein are entirely different. Moreover, it is highly unlikely for a MRCPv2 server to implement the primitives described in this document. Likewise, it is exceedingly unlikely for a server implementing this

protocol to implement speech services such as speech recognition, speaker identification, or text-to-speech.

We could institute a complex capabilities negotiation mechanism and a large set of non-overlapping, optional methods. However, it is much cleaner to simply use the proven MRCPv2 session establishment model with a wire protocol that is appropriate for the task at hand.

2. Conventions and Terminology

In this document, BCP 14/RFC 2119 [1] defines the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL". In addition, BCP 15 indicates requirement levels for compliant implementations.

The following additional terms are defined for use in this document:

B2BUA : A B2BUA is a Back-to-Back SIP User Agent.

Media Server : A Media Server is an entity that performs media processing on behalf of a requesting agent or Media Control Client. In particular, a Media Server offers mixing, announcement, tone detection and generation, and object play and record services. The Media Server has a direct RTP [15] relationship with the source or sink of the media flow.

Media Control Client : A Media Control Client is an entity that requests media processing from a Media Server. Note that the Media Control Client may not have any media capabilities whatsoever. For example, the Media Control Client may be an Application Server (B2BUA) or other endpoint requesting manipulation of a third-party's media stream. In the document, we often refer to this entity simply as "the Client".

3. Overview

This document details mechanisms for establishing, using, and terminating a reliable channel using SIP for the purpose of controlling a Media Server. The following text provides a non-normative overview of the mechanisms used. Detailed, normative guidelines are provided later in the document.

Media control channels are negotiated using standard SIP mechanisms that would be used in a similar manner to creating a voice session. Figure 1 illustrates a simplified view of the proposed mechanism. It highlights a separation of the SIP signaling traffic and the associated control channel that is established as a result of the SIP interactions.

The use of SIP for the specified mechanism provides many inherent

capabilities which include:-

- Service location - Use SIP Proxies or Back-to-Back User Agents for discovering Media Servers.
- Security mechanisms - Leverage established security mechanisms such as TLS and Client Authentication.
- Connection Maintenance - The ability to re-negotiate a connection, ensure it is active, audit parameters, etc.
- Media Agnostic - Generic protocol allows for easy extension.

As mentioned in the previous list, one of the main benefits of using SIP as the session control protocol is the 'Service Location' facilities provided. This applies at both a routing level where RFC 3263 [4] provides the physical location of devices and at the Service level using Caller Preferences[13] and Callee Capabilities[14]. The ability to select a Media Server based on Service level capabilities is extremely powerful when considering a distributed, clustered architecture containing varying services (e.g. Voice, Video, IM). More detail on locating Media Server resources using these techniques is outlined in Section 5 of this document.

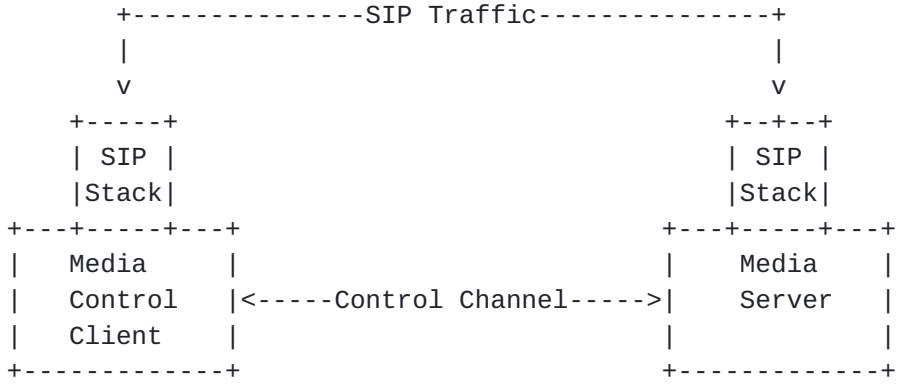

Figure 1: Basic Architecture

The example from Figure 1 conveys a 1:1 connection between the Media Control Client and the Media Server. It is be possible, if required, for multiple connections using separate SIP dialogs to be established between the Media Control Client and the Media Server entities. Any of the connections created between the two entities can then be used for Media Server control interactions. The control connections are agnostic to the overlying media sessions and specific session information is incorporated in the control interaction commands represented using the defined XML schema (as defined in Section 10). The ability to have multiple connections allows for stronger redundancy and the ability to manage high volumes of traffic in busy systems.

[Editors Note: Still under discussion. How does an app server know, when there are multiple media servers, which specific MS has any given media session? Next version of the draft will discuss the correlation procedures. The App server needs a control channel with the media server and needs to know which channel to use once the media session has been established. Sounds like a GRUU usage?

Consider the following simple example for session establishment between a Client and a Media Server (Note: Some lines in the examples are removed for clarity and brevity).

The Client constructs and sends a SIP INVITE request to the Media Server. The request contains the option tag 'mscs' in a SIP 'Require' header for the purpose of forcing the use of mechanism described in this document. The SDP payload includes the required information for control channel negotiation. The COMEDIA [8] specification for setting up and maintaining reliable connections is used (more detail available in later sections).

Client Sends to Media Server:

```
INVITE sip:Media-Server@example.com SIP/2.0
To: <sip:Media-Server@example.com>
From: <sip:Client@example.com>;tag=64823746
Require: mscs
Call-ID: 7823987HJHG6
Content-Type: application/sdp

v=0
o=originator 2890844526 2890842808 IN IP4 controller.example,com
s=-
c=IN IP4 controller.example.com
m=application 7575 TCP/MSCS
a=setup:active
a=connection:new
```

On receiving the INVITE requests, the Media Server supporting this mechanism generates a 200 OK response containing appropriate SDP.

Media Server Sends to Client:

```
SIP/2.0 200 OK
To: <sip:Media-Server@example.com>;tag=28943879
From: <sip:Client@example.com>;tag=64823746
Call-ID: 7823987HJHG6
Content-Type: application/sdp

v=0
o=originator 2890844526 2890842808 IN IP4 controller.example,com
s=-
c=IN IP4 mserver.example.com
m=application 7563 TCP/MSCS
a=setup:passive
a=connection:new
```

The Client receives the SIP 200 OK response and extracts the relevant information. It creates an outgoing (as specified by the SDP 'setup:' attribute) TCP connection to the Media server. The connection address (taken from 'c=') and port (taken from 'm=')are used to identify the remote part in the new connection.

Once established, the newly created connection can be used to exchange Media Server control language requests and responses. As well, after the control channel has been setup, media sessions can be established using standard SIP third party call control.

[Editors Note: See previous note:this is where we may need to mention how an App Server knows which Media Server is responsible for any given media session.]

Figure 4 provides a simplified view of a User Agent involved with the proposed architecture. (1) in brackets represents the SIP dialog and dedicated control channel previously described in this overview section.

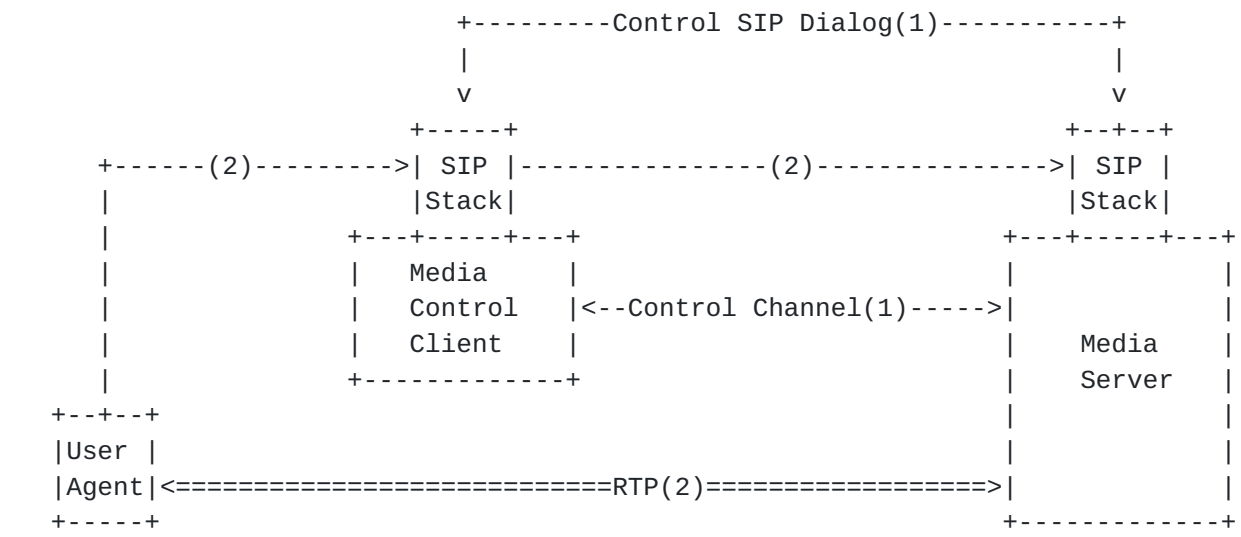

Figure 4: Participant Architecture

(2) from Figure 4 represents the User Agent SIP dialog interactions and associated media flow. A User Agent would create a SIP dialog with the Media Control Client entity. The Media Control Client entity will also create a related dialog to the Media Server (B2BUA type functionality). Using the interaction illustrated by (2), the User Agent is able to negotiate media capabilities using standard SIP mechanisms as defined in RFC 3261 [2] and RFC 3264 [5] with the Media Server. The Media Control Client will maintain relevant, unique, information associated with the User Agent media dialog. This is achieved using a concatenation of the dialog identifiers (SIP From-tag + SIP To-tag + Call-ID as defined in RFC 3261 [2] - [TBD and defined in later section]. Both Media Server and the Control Client carry out this process when a SIP media dialog from a User Agent is successful. The token produced from this concatenation process is then used by the Control Client and Media Server to directly identify a SIP dialog when Media Control commands using the XML defined in Section 10 and Section 9 are passed between the two entities.

If not present in the SDP received by the Media Control Client from the User Agent(2), a media label SDP attribute which is defined in [11] MAY be inserted for every media description (identified as m= line as defined in [10]). This provides flexibility for the Media Control Client as it can generate Media Server controls that specify a particular Media stream (between User Agent and Media Server) within a SIP media dialog. If a Media label is not included in the Media Control XML command it applies to all media associated with the dialog.

[Editors Note: TODO - Overview of Conference instance + control

commands.]

4. Locating Media Server Resources

Section will describe mechanisms for locating a Media server.

5. Controlling UAC Behavior - Control Channel Setup

On creating a new SIP INVITE request, a UAC can insist on using the mechanisms defined in this document. This is achieved by inserting a SIP Require header containing the option tag 'mscs'. A SIP Require header with the value 'mscs' SHOULD NOT be present in any other SIP request type, although extensions to SIP MAY allow its usage with other request methods.

If on creating a new SIP INVITE request, a UAC does not want to insist on the usage of the mechanisms defined in this document but merely that it supports them, a SIP Supported header MUST be included in the request with the option tag 'mscs'.

If a reliable response is received (as defined RFC 3261 [2] and RFC 3262 [3]) that contains a SIP Require header containing the option tag 'mscs', the mechanisms defined in this document are applicable to the newly created dialog.

Before the UAC can send a request, it MUST include a valid session description using the Session Description Protocol defined in . The following information defines the composition of some specific elements of the SDP payload that MUST be adhered to for compliancy to this specification.

The Connection Data line in the SDP payload is constructed as specified in [10]:

c=<nettype> <addrtype> <connection-address>

The first sub-field, <nettype>, MUST equal the value "IN". The second sub-field, <addrtype>, MUST equal either "IP4" or "IP6". The third sub-field for Connection Data is <connection-address>. This supplies a representation of the SDP originators address e.g. dns/IP representation. The address will be the network address used for connections in this specification.

Example:

c=IN IP4 controller.example.com

The SDP MUST contain a corresponding Media Description entry for

compliance to this specification:

m=<media> <port> <proto>

The first "sub-field" <media> MUST equal the value "application". The second sub-field <port> MUST represent a port on which the constructing client can receive an incoming connection if required. The port is used in combination with the address specified in the 'Connection Data line defined previously to supply connection details. If the constructing client can not receive incoming connections it MUST still enter a valid port range entry. The use of the port value '0' has the same meaning as defined in the SDP specification[10]. The third sub-field, <proto>, MUST equal the value "TCP/MSCS" as defined in Section 14.2.2 of this document.

[Editors note: Need to cover other protocols so not TCP specific]

The SDP MUST also contain a number of SDP media attributes(a=), that are specifically defined in the COMEDIA specification. The attributes provide connection negotiation and maintenance parameters. A client conforming to this specification SHOULD support all the possible values defined for media attributes from the COMEDIA [8] specification. It is RECOMMENDED that a Controlling UAC initiate a connection to a Media Server but a Media Server MAY negotiate and initiate a connection using COMEDIA, if network topology prohibits initiating connections in a certain direction. An example of the attributes might be:

```
a=setup:active
a=connection:new
```

This example demonstrates a new connection that will be initiated from the owner of the SDP payload. The connection details are contained in the SDP answer received from the UAS. A full example of an SDP payload compliant to this specification can be viewed in Section 3. Once the SDP has been constructed along with the remainder of the SIP INVITE request (as defined in RFC 3261 [2]), it can be sent to the appropriate location.

6. Media Server UAS Behavior - Control Channel Setup

On receiving a SIP INVITE request, a Media Server(UAS) inspects the message for indications of support for the mechanisms defined in this specification. This is achieved through the presence of the SIP Supported and Require headers containing the option tag 'mscs'. If the Media Server wishes to construct a reliable response that conveys

support for the extension, it should follow the mechanisms defined in RFC 3261 [2] for responding to SIP supported and Require headers. If support is conveyed in a reliable SIP provisional response, the mechanisms in RFC 3263 [4] MUST also be used.

When constructing a SIP success response, the SDP payload MUST be constructed using the semantics(Connection, Media and attribute) defined in Section 5 using valid local settings and also with full compliance to the COMEDIA[8] specification. For example, the SDP attributes included in the answer constructed for the example offer provided in Section 5 would look as illustrated below:

```
a=setup:passive
a=connection:new
```

Once the SIP success response has been constructed, it is sent using standard SIP mechanisms. Depending on the contents of the SDP payloads that were negotiated using the Offer/Answer exchange, a reliable connection will be established between the Controlling UAC and Media server UAS entities. The connection is now available to exchange XML commands, as defined in Section 9 and Section 10 of this document.

7. Media Dialog Operation

This section will describe in more detail the SIP interactions between User Agents-->Control client-->Media Server.

8. Control Command Construction

This section focuses on the construction of control commands that that are defined in the XML schemas provided later in this draft. It is expected that the draft might split dialog commands away from conference commands. This will enable simple implementations to just do IVR and advanced to implement conference control and IVR.

9. Media Control Elements

Included as a placeholder for Element definitions

10. XML Schema

Included as a placeholder for XML schema definition

11. Network Address Translator(NAT)

This section will look at geographically distributed systems where NAT traversal might be an issue. It will look at both the SIP media dialog traversal and the control channel traversal.

12. Examples

13. Security Considerations

Security Considerations to be included in later versions of this document.

14. IANA Considerations

14.1 IANA Registration of the 'mscs' Option Tag

14.2 SDP Transport Protocol

14.2.1 TCP/MSCS

14.2.2 TCP/TLS/MSCS

14.3 SDP Attribute Names

15. Acknowledgments

The authors would like to thank Ian Evans and Michael Bardzinski of Ubiquity Software for useful review and input to this work. Eric Burger contributed to the early phases of this work.

16. References

16.1 Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

16.2 Informative References

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[4] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[5] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[6] Shanmugham, S., "Media Resource Control Protocol Version 2(MRCPv2)", draft-ietf-speechsc-mrcpv2-06 (work in progress), February 2005.

[7] Groves, C., Pantaleo, M., Anderson, T., and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[8] Yon, D., "Connection-Oriented Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-comedia-10 (work in progress), November 2004.

[9] Even, R., "Requirements for a media server control protocol", draft-even-media-server-req-00 (work in progress), January 2005.

[10] Handley, M., "SDP: Session Description Protocol", draft-ietf-mmusic-sdp-new-24 (work in progress), February 2005.

[11] Levin, O. and G. Camarillo, "The SDP (Session Description Protocol) Label Attribute", draft-ietf-mmusic-sdp-media-label-01 (work in progress), January 2005.

[12] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[13] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[14] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

[15] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

Authors' Addresses

Chris Boulton
Ubiquity Software Corporation
Building 3
Wern Fawr Lane
St Mellons
Cardiff, South Wales CF3 5EA

Email: cboulton@ubiquitysoftware.com

Tim Melanchuk
Convedia
4190 Still Creek Drive, Suite 300
Vancouver, BC V5C 6C6
Canada

Email: timm@convedia.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.