

CLUE
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2013

R. Hansen
A. Krishna
A. Romanow
Cisco Systems
July 10, 2012

Call Flow for CLUE
draft-romanow-clue-call-flow-00

Abstract

This draft shows the CLUE call flow demonstrating the use of CLUE in SIP. It also provides information about the message and syntax for CLUE transport establishment and other extensions in SDP. In CLUE participants act as both providers and consumers. The draft includes a detailed example of a typical use case which includes both static and switched captures.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Background	3
2.	Terminology	4
3.	Solution Overview	4
4.	Transport for carrying CLUE signaling protocol	5
4.1.	Requirements	6
4.2.	UDP/DTLS/SCTP	6
5.	Initial Call Establishment	7
5.1.	First offer-answer exchange	7
5.2.	RTP-header extension and SDP support for CLUE calls	8
5.2.1.	RTP extension header for associating RTP stream with CLUE capture	9
5.2.2.	RTP extension header for audio rendering tag	11
5.3.	CLUE channel establishment using SDP	12
5.4.	Bandwidth considerations for CLUE	14
5.4.1.	Allocate as you go	14
5.4.2.	Pre-allocation of bandwidth	17
6.	CLUE message exchange following call setup	17
7.	Sending and receiving media	20
7.1.	RTP	20
7.2.	RTCP	21
8.	Interoperability	21
8.1.	Backward compatibility with non-CLUE endpoints	22
9.	Mid-call provider advertisement and re-configuration	22
10.	Example call flow	25
11.	Security Considerations	27
12.	Acknowledgements	28
13.	IANA Considerations	28
14.	References	28
14.1.	Normative References	28
14.2.	Informative References	28
Appendix A.	Example call flow messages	29
A.1.	INVITE from Alice	30
A.2.	200 OK from Bob	31
A.3.	CLUE advertisement A (from Alice)	31
A.4.	CLUE advertisement B (from Bob)	33
A.5.	CLUE response A (from Alice)	39
A.6.	CLUE response B (from Bob)	40
	Authors' Addresses	40

1. Background

The purpose of this draft is to examine the call flow for implementing the CLUE framework using SIP and other IETF protocols. Although much of the CLUE framework [ref] is reasonably well agreed upon in the CLUE WG, in order to develop a feasible call flow, we had to make assumptions about many aspects of CLUE that have not yet been decided: what the transport will be, what the message format will be, etc. While these assumptions are not definitive final answers, we have tried to make sensible decisions based on existing drafts and common sense. Perhaps by examining in more detail how the call flow could be handled some light will be shed on some of the proposals.

The following assumptions related to UDP are being made: [these are also stated in [draft-romanow-clue-sdp-usage](#)]

- o For each CLUE media type and content type (audio, video-main, or video-slides) [[draft-romanow-clue-data](#) model], there will be at most one corresponding SDP media stream ("m=" line). Note: video-main and video-slides are both using SDP "video" media type.
- o Multiple media captures of the same CLUE media and content type, or different encodings of a given media capture will all use the same SDP media stream (i. e., via RTP multiplexing with a different SSRC for each).
- o It is acceptable to use more than one offer/answer exchange to setup the whole Telepresence session.
- o The Telepresence session is established by setting up sets of unidirectional streams (not bidirectional streams).

Today telepresence endpoints actively negotiate audio and video SDP media streams using the SDP offer/answer model during call establishment and during mid-call features, such as hold resume. The CLUE framework adds additional parameters to a telepresence session. Each party in the CLUE conference is usually both a provider and a consumer, whether the conference is point-to-point or multipoint. The CLUE parameter values for one provider may well not be the same values for the other provider. Thus each party needs to advertise its provider encoding parameter values to the other side. This is not the typical communication model for SDP offer/answer, which is more often a bidirectional agreement on parameter values.

As specified in the CLUE framework, parties in a telepresence conference do not negotiate a single value between them; therefore offer/answer [[RFC3264](#)] is not used for the full negotiation of all encoding related values. Rather, we propose, sending CLUE parameter values via a new CLUE signaling protocol which will be negotiated via SIP. The CLUE messages consist of provider advertisements and

consumer requests.

2. Terminology

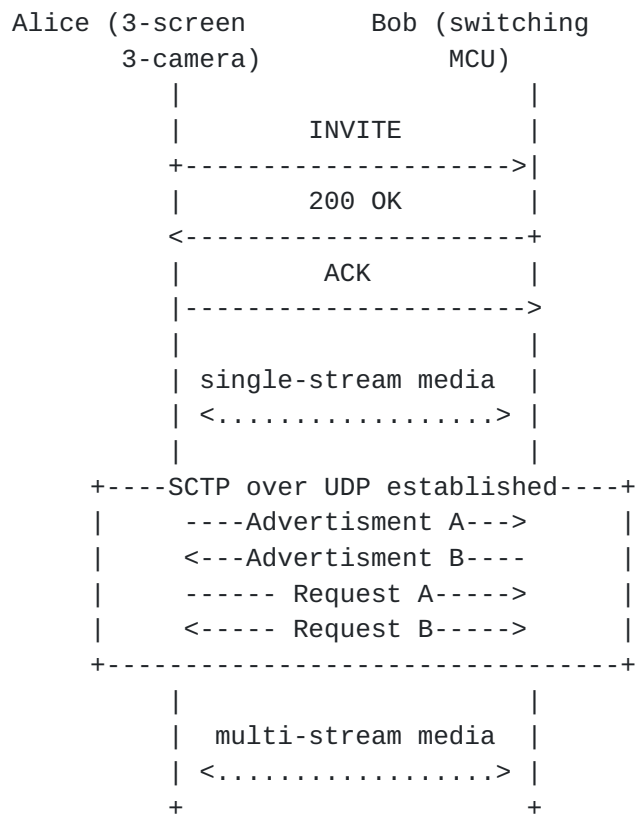
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)] and indicate requirement levels for compliant implementations.

3. Solution Overview

A number of aspects of CLUE remain to be fully defined, including major issues such as the transport mechanism and message structure. In order to provide illustrative examples, this draft makes a number of assumptions and guesses about the final format of CLUE. Where possible an attempt has been made to use existing drafts (such as for the multiplexing methodology), or similar developments in other working groups (such as the use of SCTP over UDP as a transport from RTCWeb). As such, this draft should not be taken as an attempt to specify such aspects, but merely to illustrate how they would be used to convey the necessary data: were other decisions to be made on aspects such as transport, the mechanisms would be different, but the data needing to be conveyed would remain the same.

We have assumed about the call flow that:

1. SIP is used to establish the telepresence conference, similarly to using SIP for non-CLUE video conferencing.
2. The media parameters for the call are established using SDP.
3. A new and separate CLUE signaling protocol is used for carrying CLUE messages. ([draft-romanow-clue-SDP-usage](#) looked at carrying CLUE messages in SDP and concluded it was infeasible.)
4. Usage of the CLUE protocol is established through SDP
5. The transport for the CLUE protocol is SCTP over UDP, following the RTCWeb specification.
6. The transport for CLUE protocol, SCTP/DTLS/UDP is setup in SDP, as illustrated in the following diagram and discussed below.



Call setup overview

The following section discusses the transport for carrying CLUE signaling protocol, following the direction that RTCWeb has taken. Then the telepresence call establishment using SDP is considered, including RTP header extension, initial offer answer exchange, and bandwidth considerations. The following section describes CLUE message exchange following the call setup, including provider advertisements and consumer requests. Then mid-call changes in provider advertisements and consumer requests are considered. [Section 10](#) describes a case study which includes both static and dynamic switched captures. Interoperability is then considered, followed by security.

4. Transport for carrying CLUE signaling protocol

This section discusses a feasible choice for the transport of CLUE signaling protocol. First the requirements are considered followed by a discussion of RTCWeb's choice of SCTP over DTLS over UDP.

[4.1. Requirements](#)

TBD Refer to draft wenger-clue-transport and mailing list discussion.

[4.2. UDP/DTLS/SCTP](#)

There have been several discussion in the CLUE WG suggesting that the transport requirements for CLUE and RTCWeb are sufficiently similar so that CLUE should consider following what RTCWeb has decided. This section briefly reviews RTCWeb's transport specification relevant for CLUE.

RTCWeb WG has reached a general consensus on using SCTP [[RFC4960](#)] encapsulated on DTLS [[RFC6347](#)] encapsulated on UDP for handling non-media data types in the context of RTCWeb. The approach is described in RTCWeb Datagram Connection, [draft-ietf-rtcweb-data-channel](#).

The transport protocol stack is illustrated in the diagram below, Protocol layers.

```
+-----+
|CLUE/RTCWeb|
+-----+
|  SCTP    |
+-----+
|  DTLS    |
+-----+
| ICE/UDP  |
+-----+
```

Protocol layers

The services offered by this stack are described in [draft-ietf-rtcweb-data-channel](#):

The encapsulation of SCTP over DTLS over ICE/UDP provides a NAT traversal solution together with confidentiality, source authenticated, integrity protected transfers. This data transport service operates in parallel to the media transports, and all of them can eventually share a single transport-layer port number. SCTP provides multiple streams natively with reliable, unreliable and partially-reliable delivery modes.

DTLS Encapsulation of SCTP Packets for RTCWEB, [draft-tuexen-tsvwg-sctp-dtls-encaps](#) describes the encapsulation of SCTP by DTLS.

The Stream Control Transmission Protocol (SCTP) is a transport protocol originally defined to run on top of the network protocols IPv4 or IPv6. This memo document specifies how SCTP can be used on top of the Datagram Transport Layer Security (DTLS) protocol. SCTP over DTLS is used by the RTCWeb protocol suite for transporting non-media data between browsers.

In addition a third RTCWeb related draft, Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP) [draft-ietf-mmusic-sctp-sdp](#) describes in detail how SDP can handle setting up SCTP and DTLS.

SCTP (Stream Control Transmission Protocol) is a transport protocol used to establish associations between two endpoints. This document describes how to express media transport over SCTP in SDP (Session Description Protocol). This document defines the 'SCTP', 'SCTP/DTLS' and 'DTLS/SCTP' protocol identifiers for SDP.

We have followed this syntax here.

5. Initial Call Establishment

This section is not intended to prescribe the flows and messages precisely as they are shown, but rather illustrates the principles.

5.1. First offer-answer exchange

The flow illustrated below shows Alice calling Bob offering SDP parameters that are typical of an offer. The new extensions related to CLUE are highlighted below.

Alice	Bob
(1) INVITE	
SDP-Offer {	
session-attributes	
<<< rtp-header extensions for CLUE >>> *new*	
m=audio	
m=video	
m=video	
m=application...UDP/BFCP *	
<<< m=application 6736 UDP/SCTP/CLUE * >>> *new*	
}	
----->	
(2) 200OK	
SDP-Answer {	
session attributes	
<<< rtp-header extensions for CLUE >>> *new*	
m=audio	
m=video	
m=video	
m=application...UDP/BFCP *	
<<< m=application 4534 UDP/SCTP/CLUE * >>> *new*	
<-----	

First SDP offer answer exchange

TODO : Discuss any changes to SIP header

Three issues relate to SDP support during initial call setup:

1. Extension of the RTP header for telepresence calls. As part of CLUE usage, two such potential requirements are consideration. Refer to sec 5.2.
2. Establishment of CLUE signaling channel using separate application m-lines. Sec 5.3
3. Bandwidth considerations during initial setup. Sec.5.4

5.2. RTP-header extension and SDP support for CLUE calls

Currently 2 proposals are under consideration for mechanisms that require using RTP extension headers [[RFC5285](#)]. Although neither proposal has been accepted, we are showing how each would be handled if they are added to the CLUE framework [[draft-ietf-clue-framework](#)].

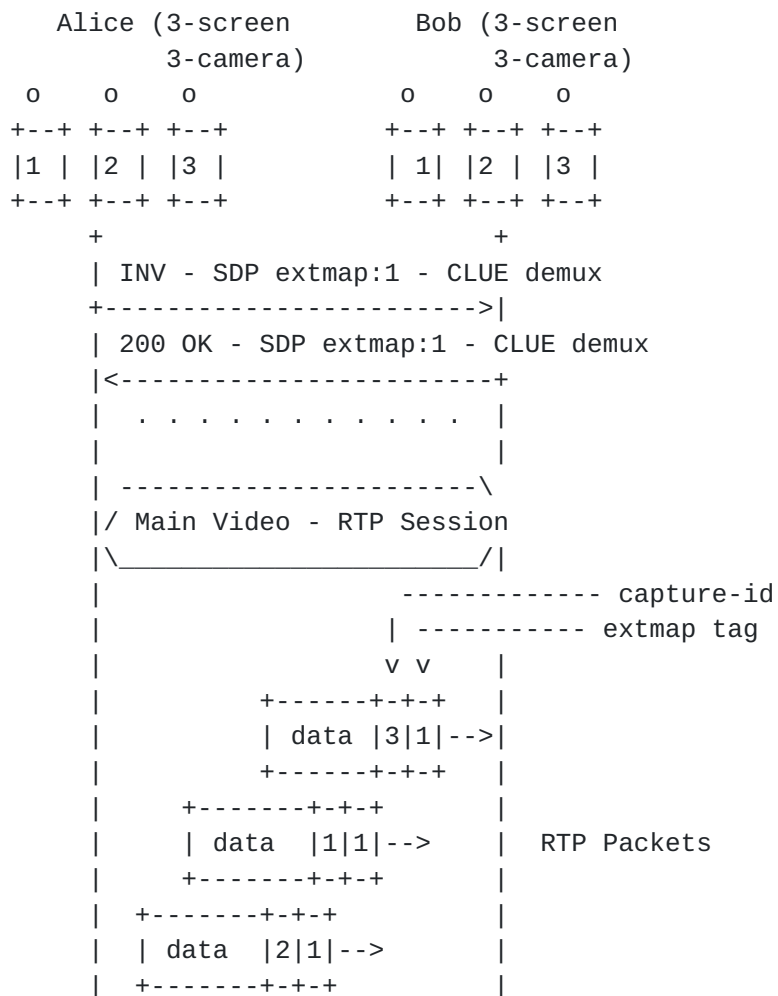
The first mechanism that requires an RTP header extension is for associating RTP streams with CLUE captures as described in [draft-lennox-clue-rtp-usage](#). The second mechanism is using an audio rendering tag to associate audio captures with video captures as described in [draft-romanow-clue-audio-rendering-tag](#). Both use session-level SDP parameters as recommended by [RFC5285](#).

5.2.1. RTP extension header for associating RTP stream with CLUE capture

Telepresence calls multiplex encoded streams from multiple similar media sources on a single RTP session for the same media-type/and content-type. For example, Alice has a 3 camera system but in SDP she negotiates a single RTP session for main video. How does she indicate these separate RTP streams with the CLUE capture ids to enable Bob, the receiver, to know the proper placement of the audio and video streams? How do the RTP streams get routed to the appropriate output devices? This is considered in detail in [draft-lennox-clue-rtp-usage](#).

The proposal is to extend the RTP header to embed the capture-id in each RTP packet. That requires negotiating a unique "id" out of band (in SDP) so that multiple such extensions could co-exist. This is achieved by using the extmap extension in SDP as specified in [RFC5285](#).

The extmap id-1 below depicts the identifier being used to signal the capture-id information in RTP extension header.



SDP example using extmap id-1

The extmap is a SDP session level attribute, as shown

```

v=0
o=3ss 40471 40471 IN IP4 10.47.197.103
c=IN IP4 10.47.197.103
t=0 0
a=extmap:1 urn:ietf:params:clue:mux
  
```

SDP showing extmap-1

An example of the RTP extension for a video packet is illustrated below:

[illegible]

RTP packet showing extension header for capture-id

5.2.2. RTP extension header for audio rendering tag

As explained in [draft-romanow-clue-audio-rendering-tag](#), to support directional audio the receiver needs to know where to render audio in relationship to video captures. In some circumstances the spatial information is not available in the provider advertisement, so the association cannot be made in that way. The proposal for these cases is that the consumer optionally tells the provider an audio tag value corresponding to each of its chosen video captures, which enables received audio to be associated with the correct video stream, even when the set of audible participants changes.

In the example provided below, the packet shows dual header extensions - one for associating RTP stream with capture as above, and one for associating audio capture with video capture.

The SDP session level attribute extmap id-3 below depicts the identifier being used to signal the audio tag field in the RTP extension header.

```
a=extmap:3 urn:ietf:params:clue:audiotag
```

An example of the RTP header extension for an audio packet is illustrated below:

								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
+-----+-----+-----+-----+-----+-----+-----+-----+																															
0xBE 0xDE length=2																															
+-----+-----+-----+-----+-----+-----+-----+-----+																															
ID=1 len=3 capture id																															
+-----+-----+-----+-----+-----+-----+-----+-----+																															
capture id ID=3 len=0 tag 0 (pad)																															
+-----+-----+-----+-----+-----+-----+-----+-----+																															

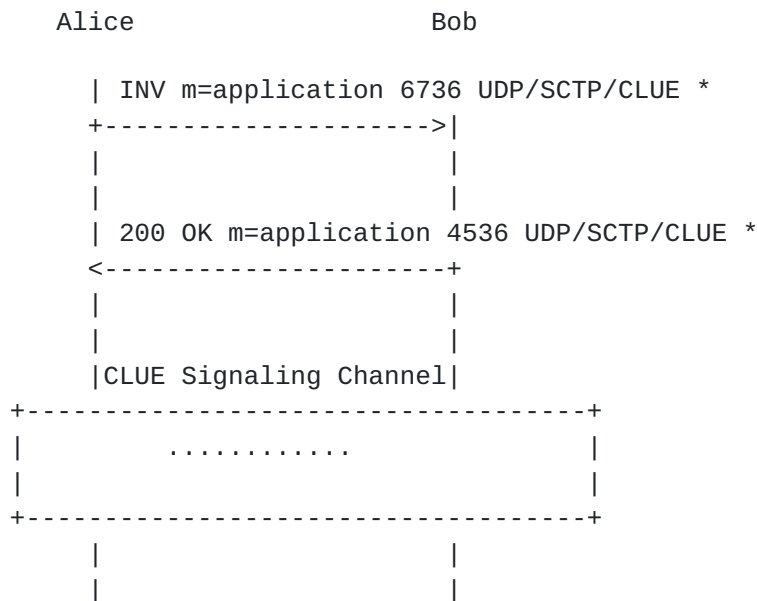
RTP packet showing extension header for capture-d and
audio
tag

The default value of these extmap attributes is to be agreed upon in case these lines were dropped by intermediaries or not sent by Alice in the first place.

5.3. CLUE channel establishment using SDP

Most importantly, the initial offer/answer negotiates the port and transport information for establishing a CLUE signaling channel. The new application m-line details the information for that. As explained in Sec.4, the CLUE signaling protocol will ride on top of SCTP transport.

The following illustrates an unsecure CLUE channel.



CLUE transport channel establishment


```
SDP Offer:
INVITE bob@biloxi.net SIP/2.0
...
Content-Length: 1000

v=0
....
m=audio...
...
m=video...
...
m=application 6736 UDP/SCTP/CLUE *
```

```
SDP Answer:
SIP/2.0 200 OK
...
Content-Length: 1000

v=0
....
m=audio...
m=video...
....
m=application 4534 UDP/SCTP/CLUE *
```

The same encrypted with DTLS shall look something as follows. This illustrates including fingerprint hash attribute in the session level of SDP in order for the other side to authenticate the identity while exchanging the certificate during the connection setup.

```
SDP Offer:
a=fingerprint:
SHA-1 64:02:2A:20:92:CD:DB:80:9F:68:0D:EF:AC:99:95:34:89:C6:7D:34
...
m=application 6736 UDP/DTLS/SCTP/CLUE *
a=setup:actpass
a=connection:new
```

```
SDP Answer:
a=fingerprint:
SHA-1 4B:AC:B7:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:BA
...
m=application 4534 UDP/DTLS/SCTP/CLUE *
a=setup:passive
a=connection:new
```


Secure CLUE channel establishment

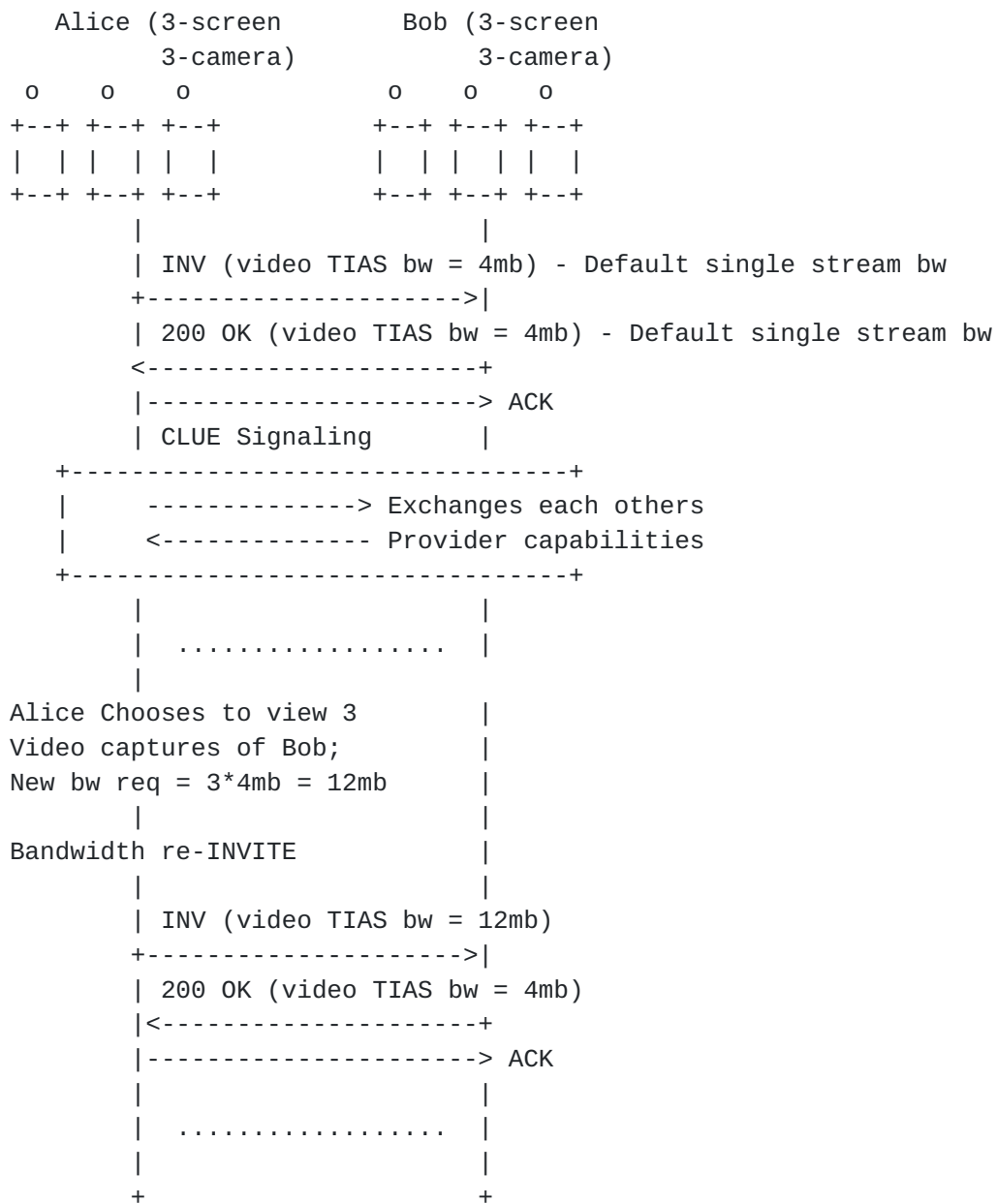
In an unencrypted call the 'setup' parameter in SDP is used to negotiate which participant will listen for an incoming SCTP connection, while the other participant initiates the connection. In an encrypted call using DTLS the 'setup' parameter negotiates which participant will establish the DTLS connection (the same participant then establishes SCTP over the DTLS channel once DTLS has been established).

5.4. Bandwidth considerations for CLUE

We discuss two models of bandwidth allocation for CLUE telepresence calls. In the first mechanism bandwidth is allocated as needed through successive re-INVITES. This has the advantage of being bandwidth efficient but may require multiple re-INVITES. In the second approach the maximum allowable bandwidth is allocated initially. This saves on re-INVITES but may allocate bandwidth not needed.

5.4.1. Allocate as you go

The bandwidth initially offered by Alice is set to some default value (say for a single-screen system). The bandwidth requirement is bound to change as the call progresses on learning the remote end capabilities and the selections made thereafter. Any increase in the usage of bandwidth based on the selected captures will result in that telepresence endpoint sending out a new re-INVITE requesting the intermediaries to allocate extra bandwidth from their pool.



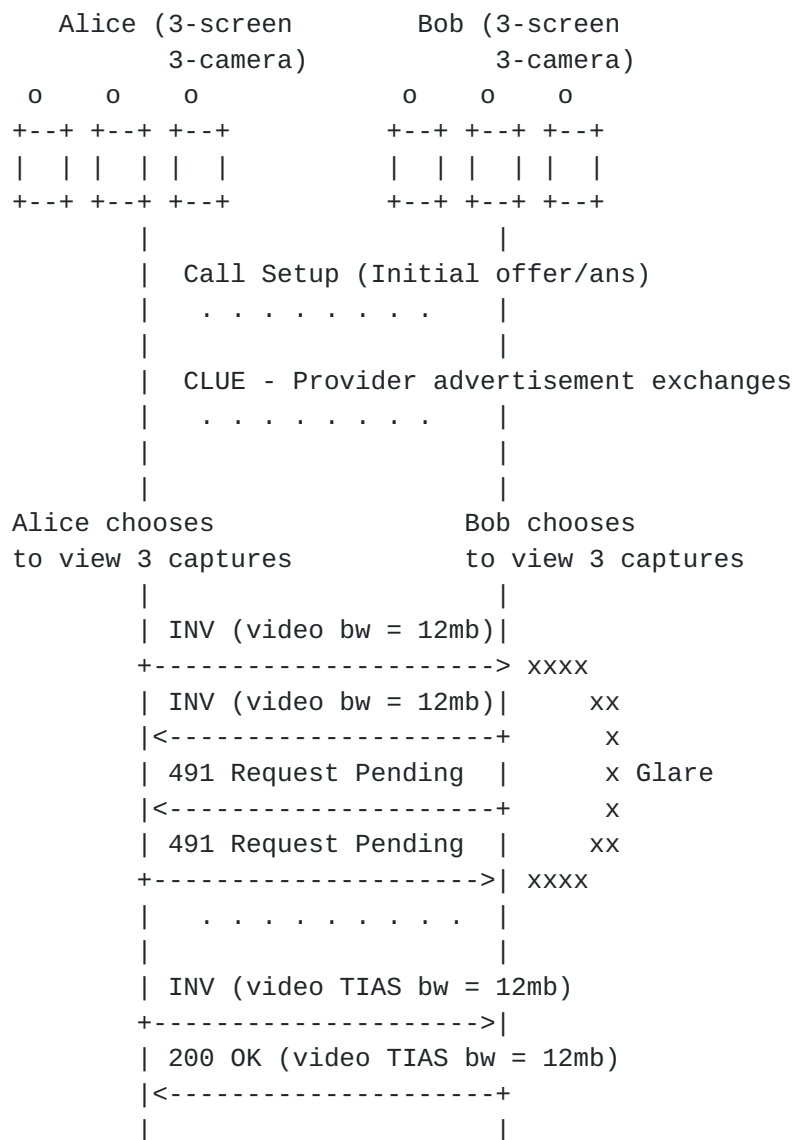
Allocate bandwidth as needed

The advantage of this approach is the usage efficiency, as during the initial call setup the remote endpoint's capabilities are still unknown. Also, even after learning the remote end's capabilities through CLUE, there is no guarantee that Alice shall decide to view all or a subset of the captures there after. So the reserved network bandwidth is the amount that is currently being used.

The drawback is the potential need for one or more re-INVITES for re-negotiating new bandwidth numbers from either side if they choose to

view additional captures or choose captures in additional encoding formats. Also due to the asynchronous nature of the consumer requests, the flow is susceptible to glare re-INVITES.

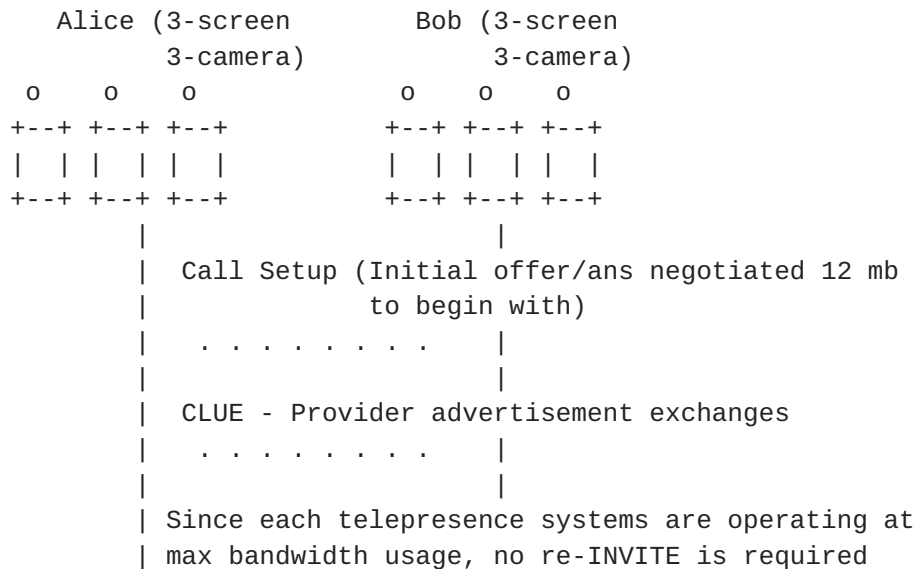
The glare would be more pronounced in the case after the first CLUE provider advertisement exchange. This may be due to these systems being pre-programmed to re-configure themselves immediately after learning each other's capabilities. The following figure illustrates the glare condition.



Glare condition in allocate bw as you go

5.4.2. Pre-allocation of bandwidth

An alternate method to avoid the need for multiple re-INVITES to increase bandwidth is to offer maximum bandwidth size that this device is capable of handling in the initial offer/answer. However this approach risks unnecessary call failures in the case the network bandwidth is provisioned and policed.



Pre-allocation of maximum bandwidth

Implementations must be able to cope with the remote endpoints operating in either mode.

6. CLUE message exchange following call setup

This section describes the call flow for CLUE messages.

Following the SIP and SDP call setup, CLUE messages are exchanged. There are only 2 types of CLUE messages: provider advertisements and consumer requests.

Each participant in a CLUE telepresence conference, whether an endpoint or an MCU, is most likely to act as both provider and consumer, as discussed above. Thus each will send provider advertisements and receive consumer requests for those advertisements; and each will also receive provider advertisements and send consumer requests.

The provider advertisements will consist of the following data elements:

- o captures
- o capture scenes
- o encoding groups
- o encodings
- o simultaneous transmission set

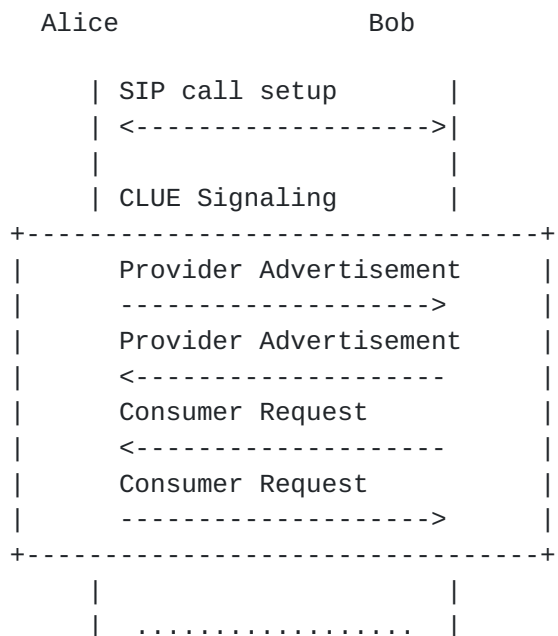
The consumer requestsconfigurations will consist of the following data elements:

- o captures
- o encodings

In considering the sequencing of messages, either participant could be first to send a provider advertisement. The next message, sent by the other participant, could be either a provider advertisement or a consumer request in response to the provider advertisement. This is illustrated in the 2 examples below. The first example shows Bob's provider advertisement sent before he responds to Alice's provider advertisement. The second example shows Bob sending a consumer request in response to Alice's provider advertisement before sending his own provider advertisement.

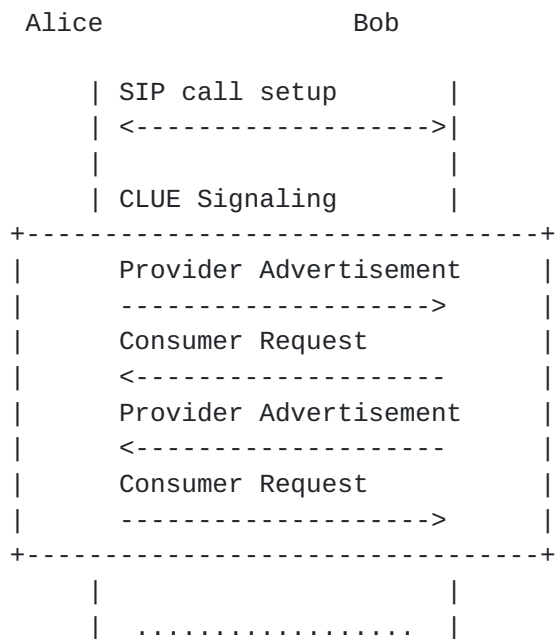
The important thing to keep in mind is that these messages are sent asynchronously and can be sent not only at the beginning of a call, but any time throughout the call when a parameter value has changed.

The diagram below shows an example of CLUE messages with provider advertisements back to back.



Back-to-back provider advertisements

Figure XX shows an example of CLUE messages with provider advertisement directly followed by consumer request.



Provider advertisement followed by consumer request

[Appendix A](#) shows example provider advertisement and consumer request messages related to [Section 10](#). Keep in mind these are not proposals, just examples using data elements taken from the current versions of the [draft-romanow-clue-data-model](#) and [draft-ietf-clue-framework](#).

7. Sending and receiving media

7.1. RTP

CLUE allows multiple media streams to be multiplexed onto a single RTP session, to reduce the complexity of negotiating independent ports for an asymmetric and variable number of media streams, and aid NAT and SBC traversal. Demultiplexing these media streams is achieved via the urn:ietf:params:clue:demux RTP header extension negotiated in the SIP messages. The provider MUST include this header extension in all media packets sent due to a successfully processed CLUE consumer request. The value of the header extension for a given stream MUST match the value of the 'multiplex-id' element for the associated stream in the consumer request. An example of the RTP header extension section of an RTP packet containing an RTP header extension is shown below:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|          0xBE          |   0xDE   |   length=2   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| ID=1 | len=3 |                capture id                |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| capture id |   0 (pad)   |   0 (pad)   |   0 (pad)   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

RTP packet showing extension header for capture-id

See [draft-lennox-clue-rtp-usage](#) for the specifics of the multiplexing process.

A receiving device which has sent at least one CLUE consumer request MUST examine RTP packet headers for the presence of this header extension. If it is not present then the packet should be treated as part of a single-stream media stream. If the extension header is present then the value of the capture id allows the various multiplexed streams to be differentiated. Packets with a capture id that does not correspond to one of the ids sent in the most recent

valid CLUE consumer request MUST be discarded.

7.2. RTCP

As described in the preceding session, multiple media streams are multiplexed onto a single RTP session. This RTP session SHOULD be matched by a single set of RTCP messages, sent in a conventional fashion. A device SHOULD send RTCP receiver reports, sender reports, SDP and other packets as it would in the single-stream case. However, when multiple media streams are present each RTCP packet SHOULD contain multiple chunks; each chunk can be used to identify a specific SSRC and include the information relevant to that media stream. This allows standard statistics, packet loss indications and other RTCP metrics to be used. CNAME, as part of the RTCP SDP message, can be used to indicate synchronization between multiple multiplexed media streams when they are generated by encoders sharing a common clock, in the same way it can be used to synchronize streams received on different ports.

8. Interoperability

CLUE support in a remote device is identified by the presence of a valid SDP media stream advertising the ability to negotiate the SCTP-based CLUE protocol - absence of this requirement indicates that either the remote device does not support CLUE, or that a middle box in the signaling path has suppressed the ability to do so. A device that receives an SDP offer or answer without a valid media stream advertising the CLUE protocol MUST fall back to a single-stream call until such time as new SDP messages are able to negotiate a CLUE protocol media stream between the devices - see [Section 10.1](#) for details on interoperability with non-CLUE devices. If both sides successfully support CLUE then the call flow proceeds as shown in this draft.

Note that as per [\[RFC3264\]](#), both devices MUST be prepared to receive media for any media streams they have advertised as recvonly or sendrecv. The devices SHOULD also begin sending single-stream media once the initial SDP negotiation is complete, while the SCTP channel is being set up and before any CLUE messages are sent or received. This minimizes the time in which a basic single-stream call is established to match that of a conventional non-CLUE system, and ensures a baseline level of interoperability. Once CLUE messages have been received and processed the call will then be 'upgraded' to multistream.

8.1. Backward compatibility with non-CLUE endpoints

One of the requirements of CLUE is that it allows backwards compatibility with devices that are not CLUE-aware, ensuring that a call between a CLUEful device and a conventional non-CLUEful SIP will result in a conventional, single-stream call.

The SIP messages sent by a CLUEful endpoint are highly compatible with conventional SIP devices, as all CLUE-specific changes are in the form of additions that will be ignored by a non-CLUE recipient. Audio and video remain specified on separate receive ports in the SDP, which not only allows non-CLUE devices to establish calls as normal, but allows separate QoS settings to be applied to the different media types if desired. A CLUEful SDP will contain one or more RTP header extmap extensions [[RFC5285](#)], which will be ignored by non-CLUE devices that do not support negotiation of RTP header extensions, or which do support such negotiation but do not recognise the CLUE-specific extensions. An offer from a CLUE device will contain an m-line specifying the CLUE transport, which the recipient will not understand and will reject (though as per the SDP offer-answer [[RFC3264](#)] the m-line itself will remain in the response and further SDPs). A CLUE device that receives an offer or a response either lacking a CLUE protocol m-line or with a rejected CLUE protocol m-line MUST fall back to a standard single-stream call.

9. Mid-call provider advertisement and re-configuration

A provider may send a new advertisement at any time, and a consumer may send a new request at any time (once it has received an initial advertisement). There is not a one-to-one mapping between advertisements and requests - a consumer may send multiple requests to a single advertisement. Advertisements contain a sequence number, while consumer requests contain both a sequence number and a reference to the sequence number of the advertisement to which they correspond - the provider can use these to detect and discard requests relating to a previous, now invalid advertisement.

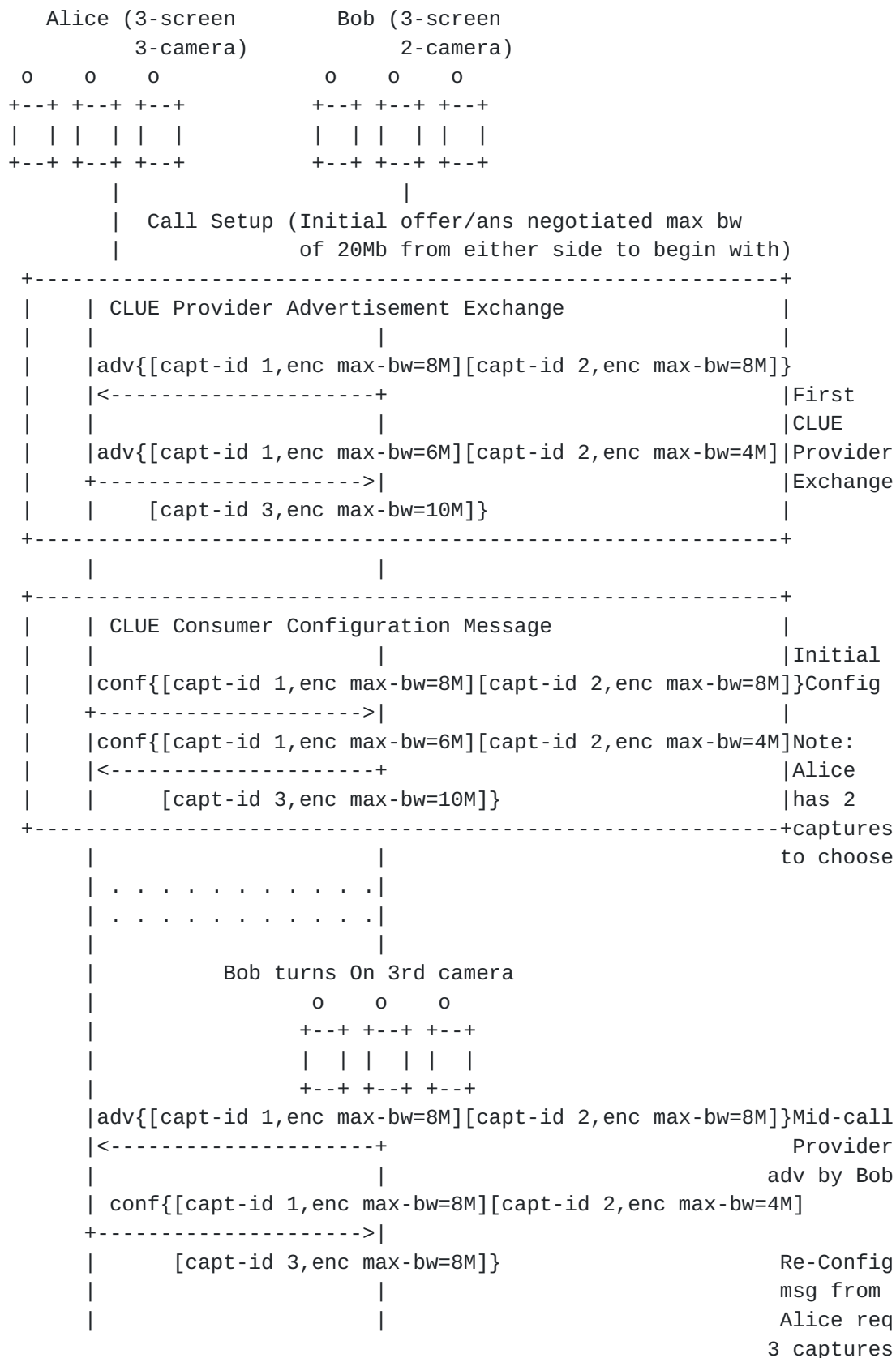
Consumer re-configuration can occur anytime during the course of a call. Possible triggers include but are not restricted to:

- o New provider advertisement to indicate a changes in captures or encodings.
- o Change in device configuration (such as availability of new screen) or user selection at the consumer end.

The bandwidth requirement and need for re-INVITE is based on the increase or decrease in usage. Implementations can choose

appropriate bandwidth as discussed previously. It is just that they have to ensure necessary bandwidth is in place before choosing any new captures.

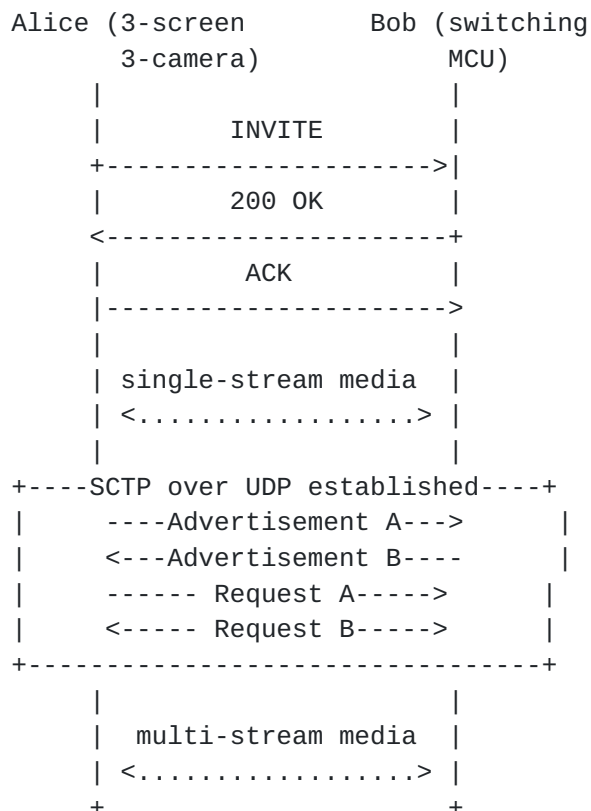
The following is a mid-call CLUE provider advertisement from Bob advertising capture-scene with 3 captures as opposed to 2 previously (could be triggered by turning on the 3rd camera. In order to keep the call-flow simple we assume both Alice and Bob allocate a large bandwidth (20Mb). The consumer request is within that bandwidth boundary.



Provider advertisement followed by consumer request

10. Example call flow

To help illustrate how all the aspects of a multistream call using CLUE tie together, a complete set of SIP and CLUE message is provided for a sample call between two participants. Alice in this example is a three-screen endpoint, with three cameras. She is able to send all three camera streams individually, or can send a single switched video stream (based on the current active speaker, or most recent speaker if no one in her room is speaking). Her system also has three microphones, but only advertises a single, composed audio stream made up of the streams of three microphones mixed together. Bob, in contrast, is an MCU that switches media between different participants in a conference. Bob advertises up to three video streams and three audio streams, with alternate offerings for one- and two-screen systems.



Example call flow

The flow begins with Alice sending an INVITE to Bob (Appendix A.1.)

The SDP includes an audio stream, offering AAC and G.711(u), and a video stream offering H264. Alice is limited to receiving 6M of bandwidth. Alice also includes a media line for CLUE over SCTP over UDP (for simplicity in this example the stream is unencrypted). The attributes for the stream show that this is a new connection, and that Alice is willing to be either active or passive with respect to establishment of the media session. Finally, Alice includes an SDP 'extmap' parameter, advertising that she understands the multiplex id RTP header extension, and that Bob should use an ID of 1 to identify such extensions; this parameter is included at the session level, as it applies to both audio and video media streams.

Bob responds with a 200 OK (Appendix A.2. - any prior 1xx messages have been elided for brevity). The SDP also includes an audio stream supporting AAC and G.711(u) and a video stream offering H264. Bob is able to receive up to 10M of media. Bob includes a media line for CLUE over SCTP over UDP that matches Alice's; the 'connection:new' attribute corresponds to hers, and Bob chooses to be passive in establishment of the stream. Bob also includes an SDP 'extmap' parameter, in this case specifying that Alice should send multiplexed media with an RTP extension header with an ID of 3.

Alice sends an ACK, not included in the appendix.

Having completed the initial SIP call setup, Alice and Bob begin sending H264 video and AAC audio in a conventional single-stream fashion. Further, Alice initiates a STCP connection to Bob on port 3782, encapsulated in UDP, as was negotiated in the SDP exchange.

With the CLUE transport established both sides are free to begin sending CLUE messages. The call flow shows Alice sending an advertisement before Bob, but in practise there is no correlation between the sending of these messages; Bob may send his message first, or both may be sent simultaneously. Both sides of the CLUE message exchange are independant of one another.

Alice's CLUE advertisement includes four video captures (Appendix A.3.) The first three are static captures that correspond to the three cameras of the system, while the final one is a switched capture to show the current active speaker. The static captures give the physical coordinates of the area of capture, in millimetres, while the switched capture does not include an area of capture. There is a single, mixed audio capture, again with no area of capture. The entries for the capture scene allow the recipient to understand which captures provide 'equivalent' views. In this case it illustrates that to get a 'complete' view of the scene the recipient should subscribe to either captures 1, 2 and 3, or to capture 4. Finally, the information in the encoding groups shows

that while Alice can supply any given video stream at up to 4M, she can only supply a total of 6M of video.

In contrast, as a switching MCU Bob has no static captures (Appendix A.4.) Instead, Bob wants to be able to supply one-, two- and three-screen switched capture views. He does this by advertising six video captures with non-physical area of capture coordinates; these coordinates are used to illustrate the fraction of the overall scene a capture represents (and how to render the captures to ensure their adjacency is correct and no multi-screen systems are rendered back-to-front). A similar approach is taken for audio. This is reinforced in the entries, which illustrate the captures to which a recipient should subscribe to receive a 'correct' view of the conference. The encoding information supplied by Bob shows that he is able to supply up to three video streams, with the only encoding maximum that of the 10M advertised in his SDP for the stream total.

Having received Bob's CLUE advertisement, Alice then sends her consumer request (Appendix A.5.) As she is a three-screen system she requests video captures 4, 5 and 6, which correspond to the switched captures for a three-screen system. She includes a 'max-bandwidth' element, limiting each stream to 2M. Having two speakers she subscribes to audio captures 8 and 9, which correspond to the two-channel audio advertised by Bob.

Having received Alice's CLUE advertisement, Bob sends his initial consumer request (Appendix A.6.) Bob's request is straightforward, requesting the three static camera streams and the mixed audio stream, and adds no additional encoder limitations beyond those already imposed by his SDP.

Both Bob and Alice now send multistream audio and video, the RTP packets including the multiplex extension header with capture ids specified in the consumer requests. Note again that, though the example illustrates Alice and Bob sending their CLUE messages in order, this is not a requirement; Alice might send her advertisement, receive Bob's request and begin sending multistream media before ever receiving Bob's advertisement. At any time Alice or Bob may send further CLUE advertisement or request messages, which invalidate previous messages. They may also send SIP messages to update or alter media parameters.

11. Security Considerations

This document provides an overview of the call-flow of a CLUE multiscreen telepresence call, and hence is not an appropriate place to definitively identify and deal with security considerations.

However, it should be clear from the above that CLUE does pose a number of security challenges: for example, the transport conveying CLUE messages must be encryptable in a fashion that limits the potential for man-in-the-middle attacks, while the multiplexing of RTP streams must not interfere with the ability to secure these streams against interception or spoofing. These security considerations will be addressed as part of the specification of these aspects of CLUE, but it is believed that well-understood methods (such as DTLS for the CLUE protocol and SRTP for the media) can be used to secure these channels.

12. Acknowledgements

13. IANA Considerations

This draft includes examples of a number of features which, if agreed on by consensus, would have IANA considerations as part of their specification. The examples used here are purely for illustrative purposes, and do not represent the final format of these features; as such this draft has no IANA considerations.

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", [RFC 3264](#), June 2002.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", [RFC 4960](#), September 2007.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", [RFC 5285](#), July 2008.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), January 2012.

14.2. Informative References

- [I-D.ietf-clue-framework]
Romanow, A., Duckworth, M., Pepperell, A., and B. Baldino,

"Framework for Telepresence Multi-Streams",
[draft-ietf-clue-framework-06](#) (work in progress),
July 2012.

[I-D.lennox-clue-rtp-usage]

Lennox, J., Witty, P., and A. Romanow, "Real-Time
Transport Protocol (RTP) Usage for Telepresence Sessions",
[draft-lennox-clue-rtp-usage-04](#) (work in progress),
June 2012.

[I-D.romanow-clue-audio-rendering-tag]

Romanow, A., Hansen, R., Pepperell, A., and B. Baldino,
"The need for audio rendering tag mechanism in the CLUE
Framework", [draft-romanow-clue-audio-rendering-tag-00](#)
(work in progress), May 2012.

[I-D.romanow-clue-data-model]

Romanow, A. and A. Pepperell, "Data model for the CLUE
Framework", [draft-romanow-clue-data-model-01](#) (work in
progress), June 2012.

[I-D.romanow-clue-sdp-usage]

Romanow, A., Andreassen, F., and A. Krishna, "Investigation
of Session Description Protocol (SDP) Usage for
Controlling multiple streams for telepresence (CLUE)",
[draft-romanow-clue-sdp-usage-01](#) (work in progress),
March 2012.

[I-D.wenger-clue-transport]

Wenger, S., Eubanks, M., Even, R., and G. Camarillo,
"Transport Options for Clue",
[draft-wenger-clue-transport-02](#) (work in progress),
March 2012.

[Appendix A](#). Example call flow messages

A.1. INVITE from Alice

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/TCP 10.47.197.7:5060;branch=z9hG4bK7251784nf
Call-ID: 3ebf09e73b83408f@10.47.197.7
CSeq: 1 INVITE
Contact: <sip:alice@example.com>
From: "Alice" <sip:alice@example.com>;tag=6d1e0bf6d948f0b8
To: <sip:bob@example.com>
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 525

v=0
o=alice 1 3 IN IP4 10.47.197.7
s=-
c=IN IP4 10.47.197.7
b=AS:6000
t=0 0
a=extmap:1 urn:ietf:params:clue:demux
m=audio 1000 RTP/AVP 101 0
b=TIAS:64000
a=rtpmap:101 MP4A-LATM/90000
a=fmtp:101 profile-level-id=25;object=23;bitrate=64000
a=rtpmap:0 PCMU/8000
a=sendrecv
m=video 1002 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42e016;max-mps=35000;max-fs=3600
a=sendrecv
m=application 1004 UDP/SCTP/CLUE *
a=setup:actpass
a=connection:new
```


A.2. 200 OK from Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP 10.47.197.7:5060;branch=z9hG4bK7251784nf
Call-ID: 3ebf09e73b83408f@10.47.197.7
CSeq: 1 INVITE
From: "Alice" <sip:alice@example.com>;tag=6d1e0bf6d948f0b8
To: <sip:bob@example.com>;tag=b91d0ea4
Contact: <sip:bob@example.com>;isfocus
Content-Type: application/sdp
Content-Length: 474

v=0
o=bob 45727 45727 IN IP4 10.47.196.161
s=-
c=IN IP4 10.47.196.161
b=AS:10000
t=0 0
a=extmap:3 urn:ietf:params:clue:demux
m=audio 3778 RTP/AVP 101 0
a=rtpmap:101 MP4A-LATM/90000
a=fmtp:101 profile-level-id=25;object=23;bitrate=64000
a=sendrecv
m=video 3780 RTP/AVP 98 99 34 31
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42e016;max-mps=244800;max-fs=8160
a=sendrecv
m=application 3782 UDP/SCTP/CLUE *
a=setup:passive
a=connection:new
```

A.3. CLUE advertisement A (from Alice)

```
<advertisement sequence='100'>
  <capture-scene id='1'>
    <spatial-description>
      <scale>millimeters</scale>
    </spatial-description>
    <capture id='1'>
      <media-type>video</media-type>
      <content-type>main</content-type>
      <encoding-group-id>1</encoding-group-id>
      <spatial-description>
        <point-of-capture>
          <point x='0' y='0' z='0' />
        </point-of-capture>
      </spatial-description>
    </capture>
  </capture-scene>
</advertisement>
```



```
</point-of-capture>
<capture-area>
  <point x='-1000' y='0' z='3000' />
  <point x='1000' y='-1125' z='3000' />
  <point x='-1000' y='-1125' z='3000' />
  <point x='1000' y='0' z='3000' />
</capture-area>
</spatial-description>
</capture>
<capture id='2'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <spatial-description>
    <point-of-capture>
      <point x='0' y='0' z='0' />
    </point-of-capture>
    <capture-area>
      <point x='1000' y='0' z='3000' />
      <point x='2600' y='-1125' z='1800' />
      <point x='1000' y='-1125' z='3000' />
      <point x='2600' y='0' z='1800' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='3'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <spatial-description>
    <point-of-capture>
      <point x='0' y='0' z='0' />
    </point-of-capture>
    <capture-area>
      <point x='-1000' y='0' z='3000' />
      <point x='-2600' y='-1125' z='1800' />
      <point x='-1000' y='-1125' z='3000' />
      <point x='-2600' y='0' z='1800' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='4'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
</capture>
<capture id='5'>
```



```
<media-type>audio</media-type>
<content-type>main</content-type>
<encoding-group-id>2</encoding-group-id>
<mixed>true</mixed>
</capture>
<entries>
  <entry>
    <capture id='1' />
    <capture id='2' />
    <capture id='3' />
  </entry>
  <entry>
    <capture id='4' />
  </entry>
  <entry>
    <capture id='5' />
  </entry>
</entries>
</capture-scene>
<simultaneous-transmission-set />
<encoding-group id='1'>
  <max-bandwidth>6000000</max-bandwidth>
  <encoding id='1'>
    <max-bandwidth>4000000</max-bandwidth>
  </encoding>
  <encoding id='2'>
    <max-bandwidth>4000000</max-bandwidth>
  </encoding>
  <encoding id='3'>
    <max-bandwidth>4000000</max-bandwidth>
  </encoding>
</encoding-group>
<encoding-group id='2'>
  <encoding id='4'>
    <max-bandwidth>64000</max-bandwidth>
  </encoding>
</encoding-group>
</advertisement>
```

[A.4.](#) CLUE advertisement B (from Bob)

```
<advertisement sequence='1'>
  <capture-scene id='1'>
    <spatial-description>
      <scale>No Scale</scale>
    <scene-area>
```



```
<point x='0' y='0' z='0' />
<point x='1' y='1' z='0' />
<point x='0' y='1' z='0' />
<point x='1' y='0' z='0' />
</scene-area>
</spatial-description>
<capture id='1'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0' y='0' z='0' />
      <point x='1' y='1' z='0' />
      <point x='0' y='1' z='0' />
      <point x='1' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='2'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0' y='0' z='0' />
      <point x='0.5' y='1' z='0' />
      <point x='0' y='1' z='0' />
      <point x='0.5' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='3'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.5' y='0' z='0' />
      <point x='1' y='1' z='0' />
      <point x='0.5' y='1' z='0' />
      <point x='1' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
```



```
<capture id='4'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0' y='0' z='0' />
      <point x='0.33' y='1' z='0' />
      <point x='0' y='1' z='0' />
      <point x='0.33' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='5'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.33' y='0' z='0' />
      <point x='0.67' y='1' z='0' />
      <point x='0.33' y='1' z='0' />
      <point x='0.67' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='6'>
  <media-type>video</media-type>
  <content-type>main</content-type>
  <encoding-group-id>1</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.67' y='0' z='0' />
      <point x='1' y='1' z='0' />
      <point x='0.67' y='1' z='0' />
      <point x='1' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='7'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
```



```
<capture-area>
  <point x='0' y='0' z='0' />
  <point x='1' y='1' z='0' />
  <point x='0' y='1' z='0' />
  <point x='1' y='0' z='0' />
</capture-area>
</spatial-description>
</capture>
<capture id='8'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0' y='0' z='0' />
      <point x='0.5' y='1' z='0' />
      <point x='0' y='1' z='0' />
      <point x='0.5' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='9'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.5' y='0' z='0' />
      <point x='1' y='1' z='0' />
      <point x='0.5' y='1' z='0' />
      <point x='1' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='10'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0' y='0' z='0' />
      <point x='0.33' y='1' z='0' />
      <point x='0' y='1' z='0' />
      <point x='0.33' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
```



```
</spatial-description>
</capture>
<capture id='11'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.33' y='0' z='0' />
      <point x='0.67' y='1' z='0' />
      <point x='0.33' y='1' z='0' />
      <point x='0.67' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<capture id='12'>
  <media-type>audio</media-type>
  <content-type>main</content-type>
  <encoding-group-id>2</encoding-group-id>
  <switched>true</switched>
  <spatial-description>
    <capture-area>
      <point x='0.67' y='0' z='0' />
      <point x='1' y='1' z='0' />
      <point x='0.67' y='1' z='0' />
      <point x='1' y='0' z='0' />
    </capture-area>
  </spatial-description>
</capture>
<entries>
  <entry>
    <capture id='1' />
  </entry>
  <entry>
    <capture id='2' />
    <capture id='3' />
  </entry>
  <entry>
    <capture id='4' />
    <capture id='5' />
    <capture id='6' />
  </entry>
  <entry>
    <capture id='7' />
  </entry>
  <entry>
    <capture id='8' />
  </entry>
```



```
        <capture id='9' />
    </entry>
    <entry>
        <capture id='10' />
        <capture id='11' />
        <capture id='12' />
    </entry>
</entries>
</capture-scene>
<simultaneous-transmission-set />
<encoding-group id='1'>
    <encoding id='1' />
    <encoding id='2' />
    <encoding id='3' />
</encoding-group>
<encoding-group id='2'>
    <encoding id='1' />
    <encoding id='2' />
    <encoding id='3' />
</encoding-group>
</advertisement>
```


A.5. CLUE response A (from Alice)

```
<configure sequence='10' advertise-sequence='1'>
  <capture id='4'>
    <multiplex-id>11</multiplex-id>
    <encoding-id>1</encoding-id>
    <max-bandwidth>2000000</max-bandwidth>
  </capture>
  <capture id='5'>
    <multiplex-id>22</multiplex-id>
    <encoding-id>2</encoding-id>
    <max-bandwidth>2000000</max-bandwidth>
  </capture>
  <capture id='6'>
    <multiplex-id>44</multiplex-id>
    <encoding-id>3</encoding-id>
    <max-bandwidth>2000000</max-bandwidth>
  </capture>
  <capture id='8'>
    <multiplex-id>111</multiplex-id>
    <encoding-id>1</encoding-id>
  </capture>
  <capture id='9'>
    <multiplex-id>222</multiplex-id>
    <encoding-id>2</encoding-id>
  </capture>
</configure>
```


A.6. CLUE response B (from Bob)

```
<configure sequence='1' advertise-sequence='100'>
  <capture id='1'>
    <multiplex-id>1</multiplex-id>
    <encoding-id>1</encoding-id>
  </capture>
  <capture id='2'>
    <multiplex-id>2</multiplex-id>
    <encoding-id>2</encoding-id>
  </capture>
  <capture id='3'>
    <multiplex-id>3</multiplex-id>
    <encoding-id>3</encoding-id>
  </capture>
  <capture id='5'>
    <multiplex-id>1</multiplex-id>
    <encoding-id>4</encoding-id>
  </capture>
</configure>
```

Authors' Addresses

Robert Hansen
Cisco Systems
Langley,
UK

Email: rohanse2@cisco.com

Arun Krishna
Cisco Systems
San Jose, CA
USA

Email: arukrish@cisco.com

Allyn Romanow
Cisco Systems
San Jose, CA 95134
USA

Email: allyn@cisco.com