Our Little Multiuser Dungeon (OLMUD)

Table of Contents

1 SrrTrains is Dead, Long Live SrrTrains!!!

Dear Reader,

Well, I had invested a lot of sweat into the SrrTrains project (between the years 2008 and 2019), in the end it has been frozen somehow.

Nevertheless, I can still remember a lot of the findings that I was granted during that project, so now it makes totally sense, when I try to write down my thoughts about John's projects.

John is currently doing his JSONverse, which is – as far as I understood so far – a WebGL based, X3D based experimental implementation of some Network Sensor Prototype (NSP) and its application within one (or more than one) exemplary multiuser scene(s).

You can find the JSONverse at following link: https://github.com/coderextreme/JSONverse.

1.1 Scope of this Hobby Report

This Hobby Report cannot be an official documentation of the JSONverse (JV), but it sketches a few concepts that have been, shall be, should be, might be or can be implemented in some past, present or future development step of the JV.

I cannot be more specific, because I am not a member of the JV, I am just a friend.

However, the first demo scene of John's JSONverse has been installed on my vServer and John is trying to keep it up to date at following address:

https://lc-soc-lc.at:8443/yottzumm/public/index.html

This address is protected by a simple password, to mitigate DDOS attacks

u=doug

p=freewrl

1.2 Restrictions of the Demo Scene

As written above, the demo scene (DS) is maintained at following address:

https://lc-soc-lc.at:8443/yottzumm/public/index.html

This address is protected by a simple password, to mitigate DDOS attacks

u=doug

p=freewrl

It might be obvious to the reader, that the DS cannot support (yet) all the concepts that are described in the present Hobby Report, and it might support additional concepts that are not (yet) described in the present Hobby Report.

Such discrepancies, as far as I am aware of them, are marked in grey blocks, like the following.

DS Restriction: The demo scene does not (yet) support ALL concepts that are described in the present Hobby Report, but it might support additional concepts that are not (yet) described here.

2 Multiplexing

Before we dive a little bit into the needed structure of such multiuser scenes (what I called SMS, some time ago), we need to talk about the technical provisions that are provided by the Internet.

Often, we refer to the Internet as a "Network of Computers", but actually it is a "Network of Processes", as I will elaborate shortly.

The Internet addresses, the @IP, are sometimes referred to as revealing your identity.

Well, that's not completely true. Actually, an @IP reveals the identity of a machine. Only, if it is somehow proven that you are using (have been using) this machine, then the @IP reveals your identity, too.

Second, a machine, sometimes referred to as a host, hosts one or more processes. Now, it's the processes that actually run the software of your computer. We can think of the processes as of little busy dwarves that are inhabiting the machine.

Now, if one of these dwarves, who inhabits one machine, wants to send a letter to a dwarf, who inhabits another machine, he needs to ADDRESS the dwarf.

Like in real mail, where the address consists of NAME + ADDRESS of a person, when sending an Internet DATAGRAM to another dwarf, the address consists of PORT + @IP.

Therefore, the first dwarf, let's call him "Dwarf A" creates a connection to "Dwarf B", and then the datagram is sent via the connection.

Hence, such a connection is identified by a quadruple:

Network Connection is identified by: @IP A + PORT A + @IP B + PORT B

as depicted in following Abbildung 2.1.

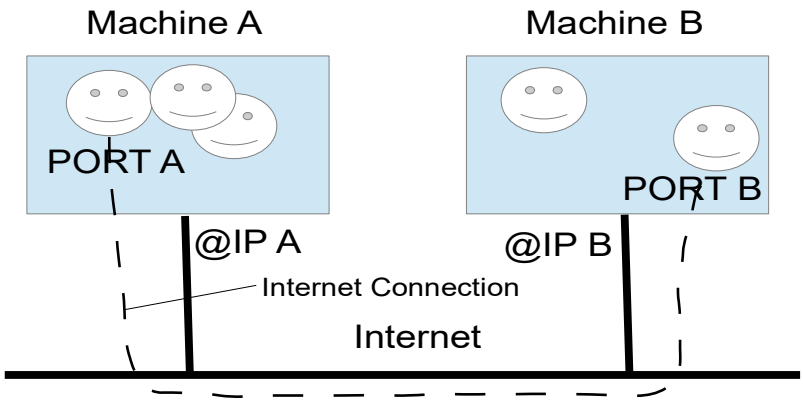

Abbildung 2.1: Basic Multiplexing in the Internet

So, this figure depicts the communication among processes in the Internet, where always two processes are connected by an Internet Connection.

3 Additional Multiplexing (Sub-Multiplexing)

However, when we talk about X3D scenes, then the processes – **what we call the Client Applications** – do not provide sufficient granularity for the network communication.

When a JV Client Application communicates with an X3D Collaboration Server – or even directly with another JV Client Application –, then we have to consider **the fact that the scenes will consist of what-i-call "objects"** (in the SrrTrains project, I called such objects the "facilities").

So, although there is only one Internet Connection for each client application (i.e. for each human user) between each client application and the X3D Collaboration Server, **we have to multiplex traffic for several (or many) objects over each of those connections**.

Additionally, **we have some session level control messages (e.g. for chat)**.

In John's implementation, the multiplexing is implemented by what is called "namespaces" of the socket.io software. There we use

- The default namespace for session level control: /
- One namespace for each object of the scene: /obj1, /obj2, /obj3, ...

This is depicted in the following figure.

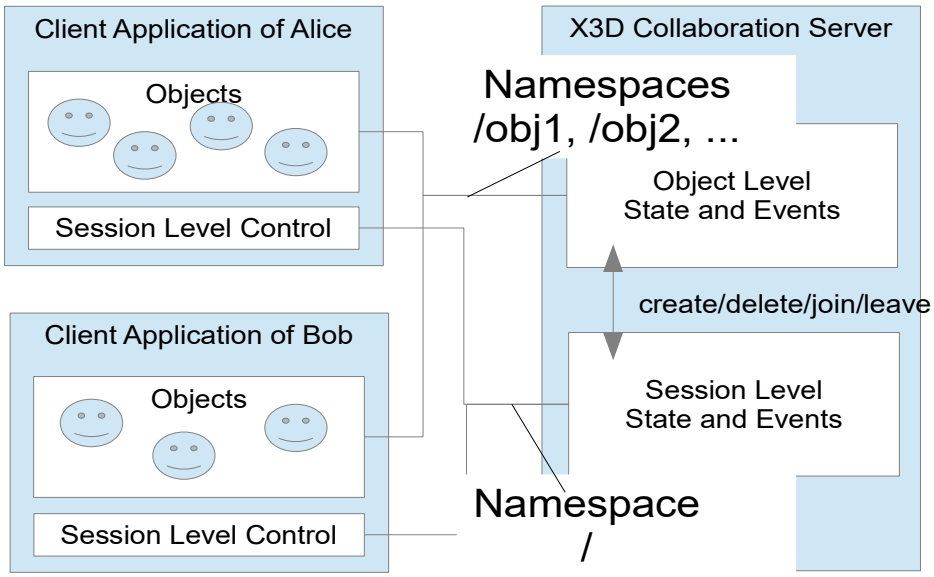

Abbildung 3.1: sub-multiplexing per namespace for objects and session level

Now, the socket.io library needs a global identifier, and additionally the server can handle more than one session and additionally each session is provided by a user of the server.

Hence the namespaces might actually look like the following:

- **Session Level Namespace: /socket.io/<user>/<sessionid>/**
- **Object Level Namespace: /socket.io/<user>/<sessionid>/<objId>**

4 The Objects of a Multiuser Scene – Shared State

The term object is one of the most general terms that we have in information technology and information science. So, what do we mean with the term object in the context of a multiuser scene?

Well, an object can be any part of an X3D scene, as long as it needs to share some information among its instances in the different client applications, on behalf of that part of the X3D scene.

This can be a model (which can be rendered), e.g. a multiuser capable slider as in the demo scene (DS), it can even be a rather complex model, like a locomotive in a trains game, or it can be an avatar[1].

Also, it can be a non-rendered part of the scene, e.g. an X3D <Script> node that shares some information among the client applications, while it calculates values for a simulation that are input for other, rendered objects.

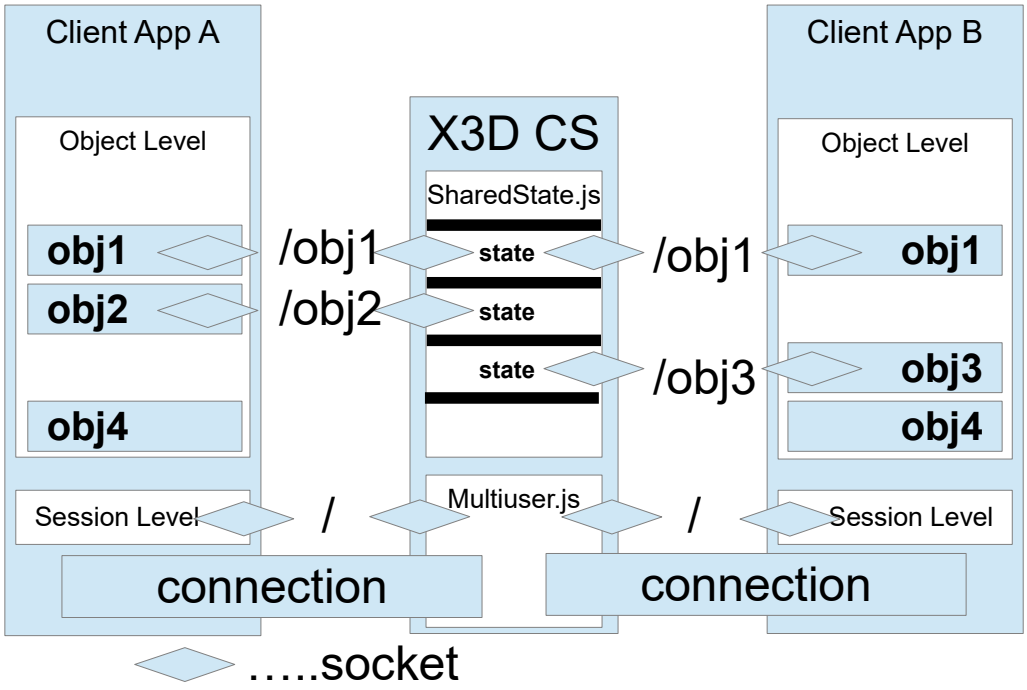

Abbildung 4.1: Example – one session with four objects

As far, as the JSONverse is involved, an object is defined by an 1:1 relation to an instance of a network sensor, i.e. by an 1:1 relationship to a socket.io socket and its namespace.

In the above example, obj1, obj2 and obj3 use the namespaces /obj1, /obj2 and /obj3 for the communication with the X3D Collaboration Server.

Acoording to our definition, obj4 is not really an object, because it does not share any information (states or events) among its instances.

1 An avatar is a very specific model, because it has a close relationship to a user of the multiuser session. It REPRESENTS a virtual identity of a user.

5 Actors

Tbd.

5.1 Scene Author

Tbd.

5.2 Model Author

Tbd.

5.3 Session Operator

Tbd.

5.4 Session Participant

Tbd.

6 Runtime Procedures

Tbd.

6.1 Bootstrapping the Scene / Session Description

6.1.1 Group Management

Tbd.

6.1.2 Session Description

Tbd.

6.1.3 Bootstrapping the Scene Instance

Tbd.

6.2 Login and Avatar Selection

Tbd.

6.3 Event Distribution

Tbd.

6.4 Shared State

Tbd.

6.5 Server Side Calculations

Tbd.

6.6 Flexible Client Side Calculations by the Author

Tbd.

7 Used Terms and Abbreviations

This Hobby Report has used following terms:

JSONverse (JV) The JSONverse is a collection of HTML/CSS/JavaScript/X3D software by John C., that aims for the experimental support of various types of 3D multiuser sessions, based on some prototype of a network sensor.
Following 3rd party software is used: socket.io as basis for the network sensor and other network communication with the collaboration server, node.js is used as platform for the server software.
The goal is collaboration within closed groups of users. Those groups of users might be small. Global collaboration is not the goal of this project.

Demo Scene (DS) The Demo Scene comes with the JSONverse. It represents the current state of implementation of the JSONverse.

Management App The JSONverse comes with a management app, which allows a managing user to manage JV groups, JV sessions and JV rooms.

Managing User A managing user is a user, who uses the management app in order to create, distribute and withdraw group tokens, as well as session tokens. Also, the management app supports the managing user with the creation, update and distribution of Session Descriptions.

Session Description (SD) A Session Description is a (JSON) data object that describes a session. I would strive for compatibility with RFC 8866, however the current implementation of the JSONverse needs more information than is standardized in RFC 8866.

Client Application Some HTML/CSS/JavaScript/X3D application that implements the UI of the multiuser session for one user (participant).

Object An object can be any part of an X3D scene, as long as it needs to share some information among its instances in the different client applications, on behalf of that part of the X3D scene

8 Appendix A – Using SDP for the Session Description

JSONverse needs some means to describe 3D Multiuser Sessions, such that a user, who posesses the Session Description (SD) and an Access Token, can easily access the Multiuser Session.

I did a small research in the IETF RFCs that describe the SDP and came to following conclusion: almost everything that is needed for the description of 3D Multiuser Sessions, is already in place. Take e.g. the following example:

```
v=0
o=yottzumm 123456789 123459999 IN IP4 lc-soc-lc.at
s=group1-petname
i=The Review of the Rose, Session#1
c=IN IP4 lc-soc-lc.at
t=3944246340  3944249940
m=application 8443 TCP/WSS/C3P *
a=setup:passive
a=connection:new
a=websocket-uri:wss://lc-soc-lc.at:8443/yottzumm/123456789
a=c3p-object:obj1
a=c3p-object:obj2
```

All these syntaxes are already specified in several RFCs, which are listed below, just

- a new proto field TCP/WSS/C3P (or TCP/WS/C3P) for the m-line and
- a new media-level attribute c3p-object

need to be defined specifically for our use case.

8.1 RFC 8866 – SDP: Session Description Protocol

The syntax is basically defined in RFC 8866, follow some citings from RFC 8866:

> When initiating multimedia teleconferences, voice-over-IP calls, streaming video, or other sessions, there is a requirement to convey media details, transport addresses, and other session description metadata to the participants.
>
> SDP provides a standard representation for such information, irrespective of how that information is transported. SDP is purely a format for session description -- it does not incorporate a transport protocol, and it is intended to use different transport protocols as appropriate, including the Session Announcement Protocol (SAP) [RFC2974], Session Initiation Protocol (SIP) [RFC3261], Real-Time Streaming Protocol (RTSP) [RFC7826], electronic mail [RFC5322] using the MIME extensions [RFC2045], and the Hypertext Transport Protocol (HTTP) [RFC7230].
>
> **SDP is intended to be general purpose** so that it can be used in a wide range of network environments and applications. **However, it is not intended to support negotiation of session content or media encodings**: this is viewed as outside the scope of session description.

So, our use case is the session announcement, where ONE session description is sent unchanged to all potential participants of the session.

The Session Description (SD) shall contain ALL of the information that is necessary to

A. download the scene and

B. connect the scene to the X3D Collaboration Server for the synchronization of the shared state and for the distribution of events.

As an exception to this idea, the access tokens (group access token GAT, session access token SAT, personal access token PAT) shall be transported outside of the session description.

Following additional citings:

```
Some lines in each description are required and some are optional,
but when present, they must appear in exactly the order given here.
(The fixed order greatly enhances error detection and allows for a
simple parser).  In the following overview, optional items are marked
with a "*".

   Session description
      v=  (protocol version)
      o=  (originator and session identifier)
      s=  (session name)
      i=* (session information)
      u=* (URI of description)
      e=* (email address)
      p=* (phone number)
      c=* (connection information -- not required if included in
           all media descriptions)
      b=* (zero or more bandwidth information lines)
      One or more time descriptions:
        ("t=", "r=" and "z=" lines; see below)
      k=* (obsolete)
      a=* (zero or more session attribute lines)
      Zero or more media descriptions

   Time description
      t=  (time the session is active)
      r=* (zero or more repeat times)
      z=* (optional time zone offset line)

   Media description, if present
      m=  (media name and transport address)
      i=* (media title)
      c=* (connection information -- optional if included at
           session level)
      b=* (zero or more bandwidth information lines)
      k=* (obsolete)
      a=* (zero or more media attribute lines)
```

The v-line

```
      v=0

The "v=" line (version-field) gives the version of the Session
Description Protocol.  This memo defines version 0.  There is no
minor version number.
```

The o-line

```
      o=<username> <sess-id> <sess-version> <nettype> <addrtype>
      <unicast-address>
```

The "o=" line (origin-field) gives the originator of the session (her username and the address of the user's host) plus a session identifier and version number:

<username> is the user's login on the originating host, or it is "-" if the originating host does not support the concept of user IDs. The <username> MUST NOT contain spaces.

<sess-id> is a numeric string such that the tuple of <username>, <sess-id>, <nettype>, <addrtype>, and <unicast-address> forms a globally unique identifier for the session. The method of <sess-id> allocation is up to the creating tool, but a timestamp, in seconds since January 1, 1900 UTC, is recommended to ensure uniqueness.

<sess-version> is a version number for this session description. Its usage is up to the creating tool, so long as <sess-version> is increased when a modification is made to the session description. Again, as with <sess-id> it is RECOMMENDED that a timestamp be used.

<nettype> is a text string giving the type of network. Initially, "IN" is defined to have the meaning "Internet", but other values MAY be registered in the future (see Section 8).

<addrtype> is a text string giving the type of the address that follows. Initially, "IP4" and "IP6" are defined, but other values MAY be registered in the future (see Section 8).

<unicast-address> is an address of the machine from which the session was created. For an address type of "IP4", this is either a fully qualified domain name of the machine or the dotted-decimal representation of an IP version 4 address of the machine. For an address type of "IP6", this is either a fully qualified domain name of the machine or the address of the machine represented as specified in Section 4 of [RFC5952]. For both "IP4" and "IP6", the fully qualified domain name is the form that SHOULD be given unless this is unavailable, in which case a globally unique address MAY be substituted.

In general, the "o=" line serves as a globally unique identifier for this version of the session description, and the subfields excepting the version, taken together identify the session irrespective of any modifications.

The s-, i- and c-line

```
      s=<session name>

   The "s=" line (session-name-field) is the textual session name.
   There MUST be one and only one "s=" line per session description.
   The "s=" line MUST NOT be empty.  If a session has no meaningful
   name, then "s= " or "s=-" (i.e., a single space or dash as the
   session name) is RECOMMENDED.  If a session-level "a=charset:"
   attribute is present, it specifies the character set used in the "s="
   field.  If a session-level "a=charset:" attribute is not present, the
   "s=" field MUST contain ISO 10646 characters in UTF-8 encoding.

      i=<session information>

   The "i=" line (information-field) provides textual information about
   the session.  There can be at most one session-level "i=" line per
   session description, and at most one "i=" line in each media
   description.  Unless a media-level "i=" line is provided, the
   session-level "i=" line applies to that media description.  If the
   "a=charset:" attribute is present, it specifies the character set
   used in the "i=" line.  If the "a=charset:" attribute is not present,
   the "i=" line MUST contain ISO 10646 characters in UTF-8 encoding.

......

   The "i=" line is intended to provide a free-form human-readable
   description of the session or the purpose of a media stream.  It is
   not suitable for parsing by automata.


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

   The "c=" line (connection-field) contains information necessary to
   establish a network connection.

   A session description MUST contain either at least one "c=" line in
   each media description or a single "c=" line at the session level.
   It MAY contain a single session-level "c=" line and additional media-
   level "c=" line(s) per-media-description, in which case the media-
   level values override the session-level settings for the respective
   media.
```

Attributes

```
a=<attribute-name>
a=<attribute-name>:<attribute-value>
```

Attributes are the primary means for extending SDP. Attributes may be defined to be used as session-level attributes, media-level attributes, or both. (Attribute scopes in addition to media-level and session-level scopes may also be defined in extensions to this document, e.g., [RFC5576] and [RFC8864].)

A media description may contain any number of "a=" lines (attribute-fields) that are media description specific. These are referred to as media-level attributes and add information about the media description. Attribute-fields can also be added before the first media description; these session-level attributes convey additional information that applies to the session as a whole rather than to individual media descriptions.

The time description

```
t=<start-time> <stop-time>
```

A "t=" line (time-field) begins a time description that specifies the start and stop times for a session. Multiple time descriptions MAY be used if a session is active at multiple irregularly spaced times; each additional time description specifies additional periods of time for which the session will be active. If the session is active at regular repeat times, a repeat description, begun by an "r=" line (see Section 5.10) can be included following the time-field -- in which case the time-field specifies the start and stop times of the entire repeat sequence.

The following example specifies two active intervals:

```
t=3724394400 3724398000 ; Mon 8-Jan-2018 10:00-11:00 UTC
t=3724484400 3724488000 ; Tue 9-Jan-2018 11:00-12:00 UTC
```

The first and second subfields of the time-field give the start and stop times, respectively, for the session. These are the decimal representation of time values in seconds since January 1, 1900 UTC. To convert these values to Unix time (UTC), subtract decimal 2208988800.

Some time representations will wrap in the year 2036. Because SDP uses an arbitrary length decimal representation, it does not have this issue. Implementations of SDP need to be prepared to handle these larger values.

If the <stop-time> is set to zero, then the session is not bounded, though it will not become active until after the <start-time>. If the <start-time> is also zero, the session is regarded as permanent.

User interfaces SHOULD strongly discourage the creation of unbounded and permanent sessions as they give no information about when the session is actually going to terminate, and so make scheduling difficult.

Media Descriptions

```
      m=<media> <port> <proto> <fmt> ...

   A session description may contain a number of media descriptions.
   Each media description starts with an "m=" line (media-field) and is
   terminated by either the next "m=" line or by the end of the session
   description.  A media-field has several subfields:

   <media>  is the media type.  This document defines the values
      "audio", "video", "text", "application", and "message".  This list
      is extended by other memos and may be further extended by
      additional memos registering media types in the future (see
      Section 8).  For example, [RFC6466] defined the "image" media
      type.

   <port>  is the transport port to which the media stream is sent.  The
      meaning of the transport port depends on the network being used as
      specified in the relevant "c=" line, and on the transport protocol
      defined in the <proto> subfield of the media-field.  Other ports
      used by the media application (such as the RTP Control Protocol
      (RTCP) port [RFC3550]) MAY be derived algorithmically from the
      base media port or MAY be specified in a separate attribute (for
      example, the "a=rtcp:" attribute as defined in [RFC3605]).


......
   <proto>  is the transport protocol.  The meaning of the transport
      protocol is dependent on the address type subfield in the relevant
      "c=" line.  Thus a "c=" line with an address type of "IP4"
      indicates that the transport protocol runs over IPv4.  The
      following transport protocols are defined, but may be extended
      through registration of new protocols with IANA (see Section 8):

      *  udp: denotes that the data is transported directly in UDP with
         no additional framing.

      *  RTP/AVP: denotes RTP [RFC3550] used under the RTP Profile for
         Audio and Video Conferences with Minimal Control [RFC3551]
         running over UDP.

      *  RTP/SAVP: denotes the Secure Real-time Transport Protocol
         [RFC3711] running over UDP.

      *  RTP/SAVPF: denotes SRTP with the Extended SRTP Profile for
         RTCP-Based Feedback [RFC5124] running over UDP.

      The main reason to specify the transport protocol in addition to
      the media format is that the same standard media formats may be
      carried over different transport protocols even when the network
      protocol is the same -- a historical example is vat (MBone's
      popular multimedia audio tool) Pulse Code Modulation (PCM) audio
      and RTP PCM audio; another might be TCP/RTP PCM audio.  In
      addition, relays and monitoring tools that are transport-protocol-
      specific but format-independent are possible.

   <fmt>  is a media format description.  The fourth and any subsequent
      subfields describe the format of the media.  The interpretation of
      the media format depends on the value of the <proto> subfield.
```

We use a media description with the values

- <media> = application
- <proto> = TCP/WS/C3P or <proto> = TCP/WSS/C3P
- <fmt> = *

This decision is based on following considerations.

1. The implementation of the JSONverse is based on socket.io, which uses WebSocket transport
2. A similar media type has been defined in RFC 8856 - Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams and RFC 8857 - Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams
   a) <media> = application shall be used
   b) RFC 8857 defines <proto> = TCP/WS/BFCP or <proto> = TCP/WSS/BFCP
   c) <fmt> = *
3. We call our WebSocket sub-protocol the C3P = Collaborative 3D Profile

8.2 RFC 4145 – TCP-Based Media Transport in the Session Description Protocol (SDP)

The attributes connection and setup are defined in this RFC. Follow some citings from RFC 4145

```
      setup-attr           =  "a=setup:" role
      role                 =  "active" / "passive" / "actpass"
                              / "holdconn"

   'active': The endpoint will initiate an outgoing connection.

   'passive': The endpoint will accept an incoming connection.

   'actpass': The endpoint is willing to accept an incoming
   connection or to initiate an outgoing connection.

   'holdconn': The endpoint does not want the connection to be
   established for the time being.


......

      connection-attr        = "a=connection:" conn-value
      conn-value             = "new" / "existing"
```

8.3 RFC 8124 - The Session Description Protocol (SDP) WebSocket Connection URI Attribute

The websocket-uri attribute is defined in RFC 8124, follows a citing:

```
This section defines a new SDP media-level attribute, "websocket-
uri", which can appear in any of the media sections.

Example:

   a=websocket-uri:wss://example.com/chat

Where "wss://example.com/chat" is the ws-URI defined in Section 3 of
[RFC6455].

When the "websocket-uri" attribute is present in the media section of
the SDP, the IP address in "c=" line SHALL be ignored and the full
URI SHALL be used instead to open the WebSocket connection.  The
clients MUST ensure that they use the URI to open the WebSocket
connection and ignore the IP address in the "c=" line and the port in
the "m=" line.
```