The Session Description Protocol (SDP) Application Token Attribute
draft-even-mmusic-application-token-01
IETF 88

Roni Even
Jonathan Lennox
Qin Wu
Introduction

• SSRC identifies an RTP stream in an RTP session.
  • SSRC is in the RTP header.
  • SSRC may change during a session

• In SDP media streams are represented in m-lines.
  • An m-line can represent a media source (unified) with multiple media streams (e.g. simulcast, FEC)
  • An m-line can represent a single media stream.
  • Multiple m-lines can be bundled to allow multiplexing of multiple media streams. (from one or multiple sources)

• Question:
  • How to map media streams in SDP m-lines to RTP streams identified by SSRC.
  • How to map media streams in non SDP application protocols (CLUE capture encoding, RTCmediastreamtrack) to SDP and RTP streams
Mapping option 1 – SSRC based

• Mapping of RTP media streams to stream description in an m-line
  • Specifying the SSRCs of the RTP stream in the m-line and binding to a stream description when there is more then one RTP stream specified by the m-line.
    • Using a=ssrc and srcname attributes. (example in simulcast draft)
    • The SSRC must be defined in the SDP and not by the RTP layer

• How to map media streams in SDP (Simulcast, FEC) and non SDP application protocols (CLUE capture encoding, RTCmediastreamtrack) streams
  • Each application defines its own way
    • Config-id, SDP group, SDP label, msid.
Option 2 – application token

- Define a token “appID” associated with an RTP stream, allowing the semantics of the stream with a token to be defined by the application.
- The binding to SSRC will be done using RTP header extension and RTCP SDES but may also be done in the SDP.
- An application may receive a new RTP stream replacing an existing RTP stream having the same appID, or a new RTP stream with a new appID.
- The appID can be used for an m-line
  - a=appID:2 (a=SSRC is not required)
  - Declare that this appID is associated with this m-line.
- In SDP unified case, the appID can be tied to a specific attribute
  - a= appID:1 imageattr:98 send [x=480,y=320]
Advantages of application token

• Leaves SSRC values to the RTP stack, when desired
  • Robust to SSRC collision
  • Keeps protocol layering cleaner – don’t need to know SSRC when making an offer

• Avoids early-media race conditions
  • SSRC values can only be specified by a sender

• Allows dynamic mappings between sources
  • E.g., loudest-speaker switching
  • Appid moves from one source to another
  • E.g., “Selective Forwarding Middlebox” RTP topology
RTP / SDP synchronization

• When mapping a specific SSRC to an appID in SDP, need to keep consistency when the mapping is changed using SDES or RTP header extension.
  • Propose that RTP always wins (SDES / RTP Header extension)
    • I.e., once you’ve seen an RTP mapping, ignore subsequent SDP-based ones
  • Other option – never use a=appid:x SSRC:value, just use a=appId and the RTP SDES and header extensions
Open issue: are header extensions/RTCP reliable?

• Argument’s been made that we need SDP as backup, because header extensions and RTCP might be dropped.

• Is this really possible?
  
  • Possible to place header extensions such that if packets carrying them are lost, stream is useless anyway (e.g. on I-Frames).
  
  • Or just always send them, if you’re paranoid.

• Middleboxes that strip header extensions and RTCP – but don’t otherwise interfere in RTP – seem very unlikely.
  
  • Remember these are multi-SSRC sessions, and probably SRTP encrypted.

• Support for this mechanism is negotiated, so can always be negotiated off. If middleboxes participate in signaling, there’s no problem.