RTP usage in WebRTC Part 1: API and Topologies

draft-ietf-rtcweb-rtp-usage-03

CAPITALS

RTCWEB Interim June 2012

Magnus Westerlund / Ericsson
Colin Perkins / University of Glasgow
Jörg Ott / Aalto University

Introduction

- WebRTC's usage of RTP, Extension and related topics will be split into two presentations:
  1. WebRTC API and RTP Topologies (Magnus)
  2. RTP/RTCP usage, Extensions etc. Implementation requirements (Colin)

Goals

- The goals with these presentations are:
  - Increase your awareness of the content of the RTP specification
  - Highlight the Open Issues that need your input
  - Enable discussion of the document
    - Find additional Open Issues
    - Find disputed requirements

Outline

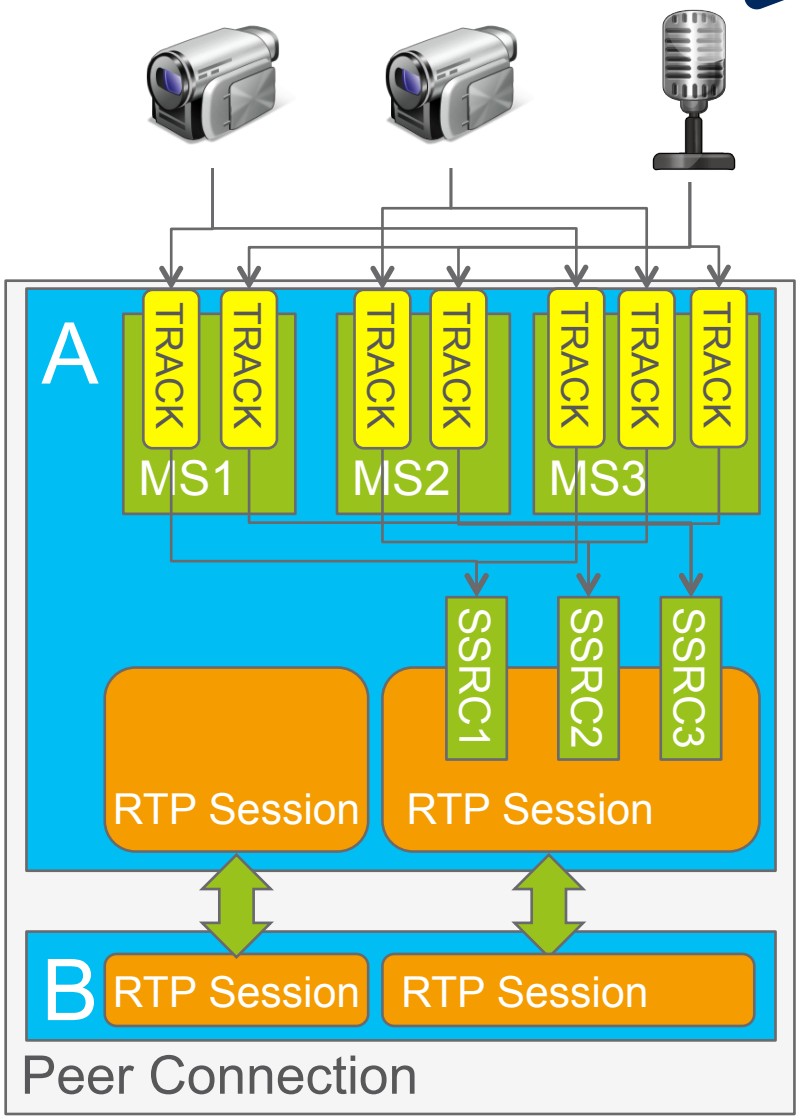
- Part 1
  - Goals
  - Definitions
  - WebRTC API
  - Topologies affects end-point functionality
  - Simulcast
- Part 2
  - Core RTP functionality
  - RTP/RTCP Extensions
  - Transport Robustness
  - Rate Control
  - Performance Monitoring

Definitions

- **RTP Session** – One SSRC space (32-bits); commonly identified by one or more address+port (destinations)
- **SSRC** – Sender Source (a 32-bit number),
  - a RTP stream source identifier,
  - independent Sequence number and Timestamp space
- **Media Stream:** A sequence of media fragments that together form a real-time experience of the media,
  - like a video sequence or an audio stream from a media source
- **RTP (Media) Stream** – A sequence of RTP packets with the same SSRC
  - providing the receiver with a encoded media stream from a media source
- **Media Source** – The source of a particular media type
  - Microphone
  - Video camera
  - Conceptual media source
    - Created from a set of other media sources, like a media mix, a selection between video cameras, etc.

WebRTC API

- **PeerConnection** – An Association between two peers
  - Containing one or more RTP sessions
  - Sent using one or more bi-directional UDP flow.
- **MediaStream** – An WebRTC API MediaStream
  - A set of MediaStreamTracks
  - Synchronized playback
- **MediaStreamTrack**
  - A Media Stream that over RTP will be represented by a SSRC



WebRTC API

- Things to Note:
  - MediaStream
    - More than one MediaStream may include the same Media Source
    - Multiple MediaStream:Tracks maps to the same Source and SSRC
    - MediaStream and tracks are unidirectional
    - Only proposal for how to establish MediaStream and Track mapping to RTP SSRC are in draft-alvestrand-rtcweb-msid-02
  - To provide synchronization in RTP all Tracks in a MS must be sent using a common CNAME
    - MediaStreamTracks may be from multiple different sources / end-points
      - Different synchronization contexts
      - A combing WebRTC node must then provide a common synchronization context
  - A PeerConnection can contain
    - multiple UDP Flows
    - RTP sessions
    - Still only one PeerConnection

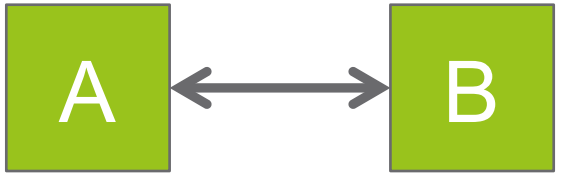
Topologies

- Topologies
  - Point-to-Point
  - Multi-unicast (MESH)
  - Mixers
  - Relay
  - End-point Forwarding
  - Simulcast
- Functionality groups
- Conclusions

Topologies

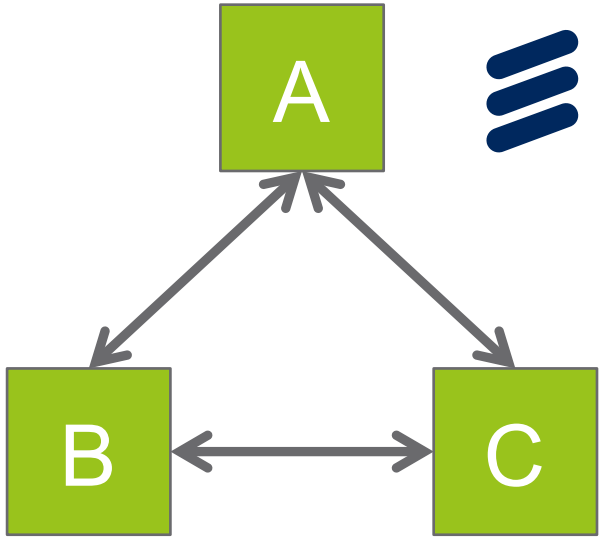
- The topologies created for a multi-media session affects end-point functionality
- This part of the presentation will:
  - Investigate a set of possible topologies in WebRTC
  - Discuss their main merits
  - Consider what functionality from an end-point they require
- How topologies relate to groups of functionality will be summarized
- Discuss recommendation on Topologies support

Point to Point



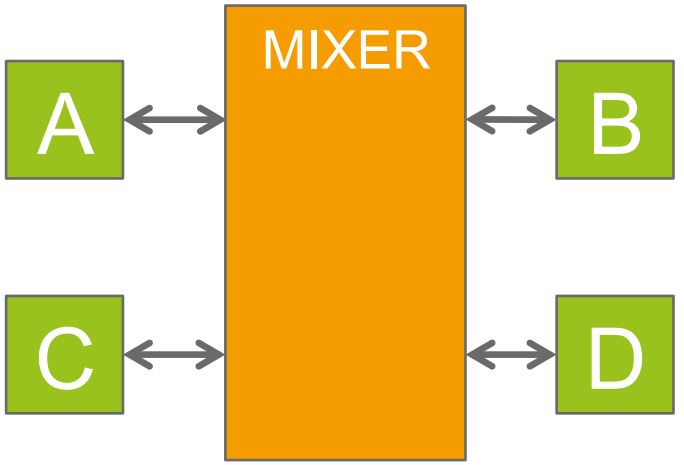
- The Point to Point is the basic topology
- A WebRTC end-point needs to support:
  - Multiple Sources (SSRCs) in one RTP session
  - One or More RTP sessions
    - Over one or more UDP flow (5-tuple)
  - Congestion Control
  - Codec Control of individual sources
  - Transport Robustifications
  - Common Security Functions
    - SRTP
    - DTLS-SRTP key management
  - Setup Signalling

Multi-Unicast (MESH)



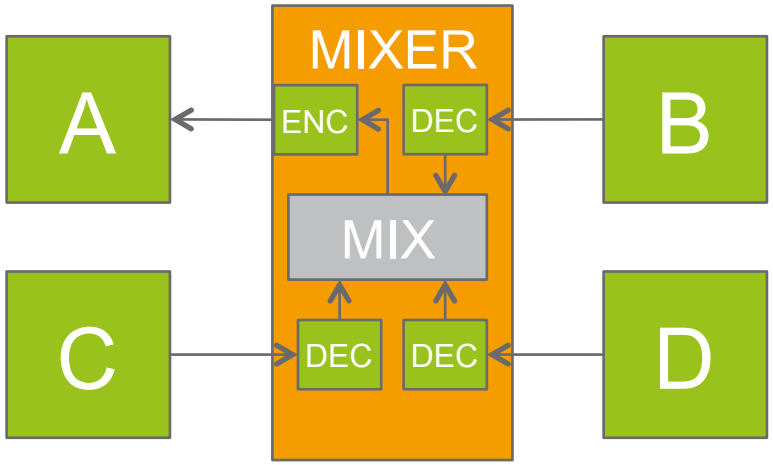
- A End-point establish multiple PC
  - Each PC has its own RTP session(s)
  - Common or Independent Media Encoders
  - Individual control and quality for each PC
- No Central Node
  - No need for media related infrastructure beyond NAT traversal
  - Increased bandwidth consumption in common path from end-point
- Controlling which media streams, bit-rate and quality
  - Distributed task as the independent PC affect each other
- An end-point must be capable of combing media from multiple PC for concurrent playout and audio mixing

Mixers



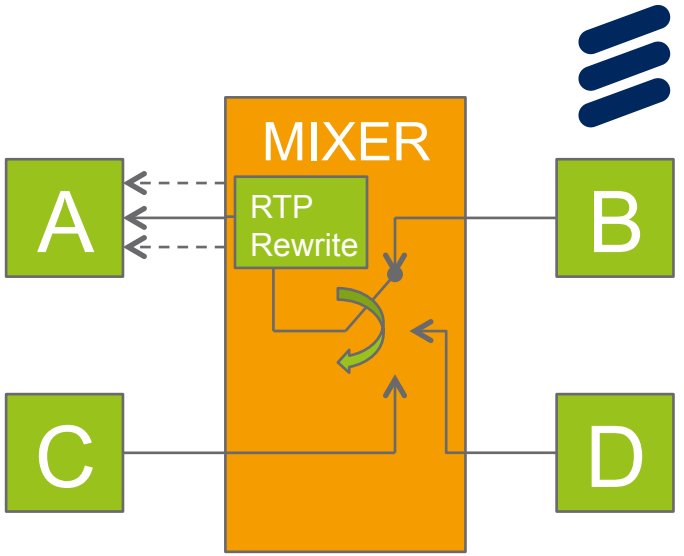
- There are several types of mixers
  - Media Mixers
  - Stream Switching
  - Source Projecting
- The have the following common properties
  - End-point communicates only with Mixer using a PC
    - The Mixer provides the other participants over that PC
  - Must be trusted devices and have media keys
    - Changes media or RTP headers
  - Tries to optimize the conference for each participant

Media Mixer



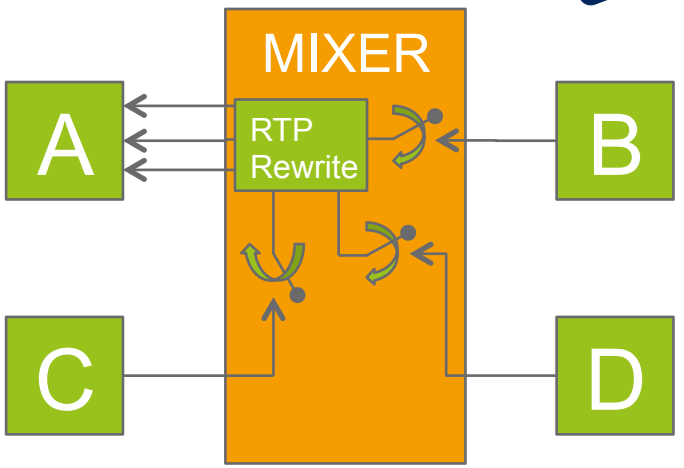
- A Media Mixer will commonly:
  - Decode incoming media streams
  - Mix or composite the selected media
  - Re-encode and transmit to the target
    - Encoding can be tailored to receivers capability and path
- Mixers will use their own SSRC when sending the encoded stream
  - Use CSRC field to provide receiver with contributing sources in mix
  - Only Source Descriptions (SDES) and BYE RTCP packets are forward between legs in RTCP
  - Mixer will have to control upstream media source based on what is most suitable for all receivers of the content in the conference

Stream Switching



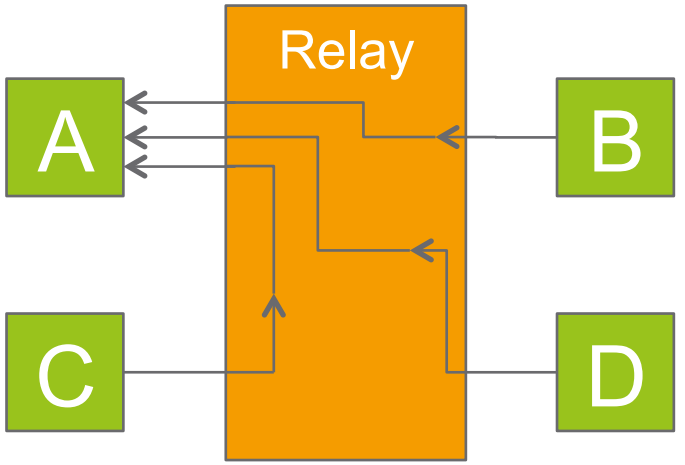
- Mixer uses conceptual SSRCs, e.g.
  - Video of the most important speaker
  - 4 SSRCs for Thumbnails of the last 4 speaker not included in most important speaker
- The Mixer constantly evaluates and selects which stream is selected to be forwarded by the Mixer's SSRC
  - RTP headers must be rewritten to ensure consistent streams
  - CSRC field can be used to indicate identity of source
- To enable switching between video streams
  - Full Intra Request are crucial
- Mixer must monitor congestion on the legs to the different receivers
  - Simulcast or scalability enables multiple quality tiers
  - To adjust a quality tier to better suite the set of receivers codec control and bit-rate adjustments are needed
- Receivers of the same stream will get the same content and quality

Source Projection



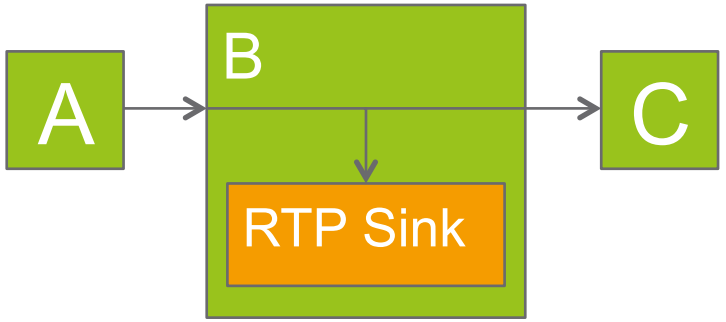
- Each participant and the mixer have their own RTP Session
- The sources in the other sessions are projected by the mixer into the other sessions
- There is a one to one mapping between SSRCs in the local session and the original media sources
- Mixer optimizes by selecting which sources are currently forwarded to this session
  - RTP headers must be rewritten to ensure consistent streams to receiver
  - Mixer needs to be able to both initiate and forward control requests between RTP sessions.
- All Receiver of particular stream gets the same content and quality

Relay (Transport Translator)



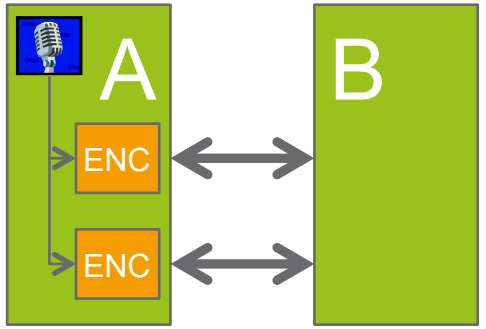
- A Relay is a media node that
  - Only rewrites transport headers (IP/UDP)
  - Functions without Crypto keys to media
  - Create a common RTP session between all participants
- End-point is required to handle multiple end-points in session
  - Merge feedback results into common adaptation decision
  - All receivers get the same content
  - Keying of session needs more than DTLS-SRTP, e.g. EKT
  - For cryptographic source authentication of individual sources extensions like TESLA are required

End-Point Forwarding



- A delivers MediaStream to B
  - B decides to forward it to C
- Simple on API level
- More complicated in Implementations
  - Forward the media stream received into other PC
    - Relay functionality
    - Maintain quality from source
    - Source Authentication of A possible
    - A must adapt media to all receivers
  - Transcode or rewrite stream before sending it to C
    - Mixer based functionality
    - Each transcoding reduces quality
    - B needs mixer logic and adaptation support
    - Trust on B to not modified A's content

Simulcast



- Simulcast, i.e. to provide multiple encodings of the same media source to the Peer
- The different encoding are used to
  - Provide different end-points with different codecs
  - Provide different quality tiers to be used in Stream Switching or Source Projection Mixers
- A way of achieving Simulcast are:
  - Establish two PeerConnections with different encoding parameters for the same MediaStreamTrack
  - Multiple MSTracks from one media source in the same PeerConnection
- A end-point could optimize local resources as discussed in Multi-unicast
  - Need to be able to ensure different encodings are provided if at all possible

Source Identity in Multiparty

- In the topologies that provides multiparty over a single PC:
  - Mixers
  - Relay
  - End-point Forwarding
- A receiver should be able to know and cross conference identities for media sources
  - Relay based solutions maintain SSRC space as common identity space that can be mapped to MediaStreamsTracks
  - Media Mixer and Stream Switching produce conceptual media streams with contributing sources
    - What level of identities of contributing sources are desired?
  - Source Projecting Mixer can maintain common identities
    - Must deal with SSRC collisions across the conference
    - Can map local SSRCs to common MediaStreamTrack identities

Functionality Groups

- Can benefit from CSRC:
  - Media Mixer
  - Stream Switching Mixer
  - End-point forwarding (Mixer based)
- Conference Extensions
  - Mixers
  - Relay
  - End-point Forwarding (both types)
- Multiple End-point handling:
  - Relay
  - End-point forwarding (Relay based)
- Multiple Simultaneous PeerConnections
  - Multi-Unicast (Mesh)
  - Simulcast?

Conclusions

- Need for Conference Extensions very well motivated
  - Question is what, see Presentation Part 2
- How to deal with Identity of contributing sources open Issue
  - CSRC handling is part of RTP core specification
  - Question more if JS application shall be provided with information
- Multiple End-point handling depends on the Use Cases
  - Core RTP has support for this
    - Some implementations may be lagging
  - Implementations complexities in adaptation and codec control logic
- Multiple Simultaneous PeerConnections
  - Have well established use cases
  - MUST be supported