Aalto University
School of Science
and Technology

Performance analysis of topologies for Web-based Real-Time Communication

Albert Abello Lozano

Supervisor: Jörg Ott
Instructor: Varun Singh
Communication and Networking Department
Aalto University, School of Science and Technology
albert.abello.lozano@aalto.fi

Espoo , May 22, 2013

Outline

- Background
- WebRTC
- Topologies
- Evaluation environment
- Tests and results
- Conclusions

Background

- Open sourced by Google
- First introduced in May 2011
- Designed in conjunction by the IETF and W3C
- Still in ongoing discussion
- Works using high level JavaScript APIs
- First version to be delivered by Q4 2013
- Uses existing technologies derived from SIP

Support

Browser vendors that are actively supporting WebRTC:

- Google
- Mozilla Foundation
- Opera
- Microsoft (separate spec)

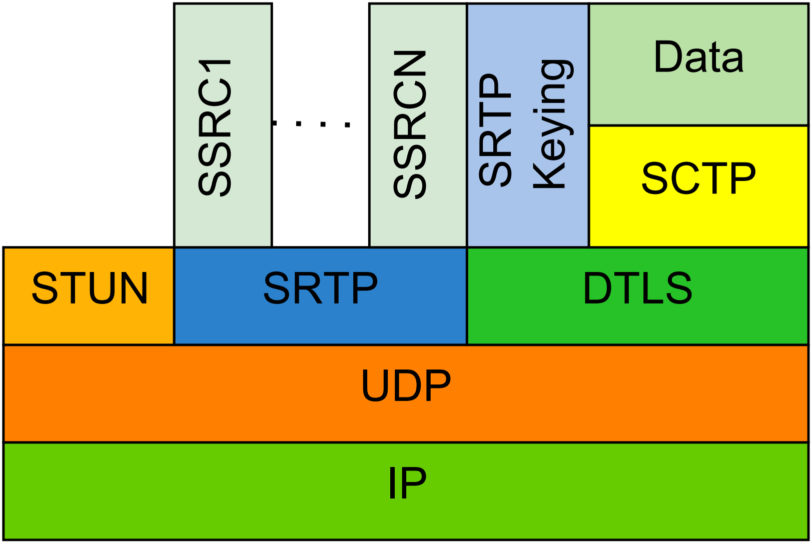
Equipment and service providers involved:

- Ericsson
- Cisco

WebRTC

- Provides real-time peer-to-peer media and data transport using browsers
- Plugin-free, mechanisms are integrated on the browsers
- Interoperable between vendors
- Uses JavaScript APIs to enable the features
- Federated domain video calls
- *Google Chrome* and *Mozilla Firefox* implement all the APIs
- *Opera* only includes *GetUserMedia*

WebRTC protocol stack

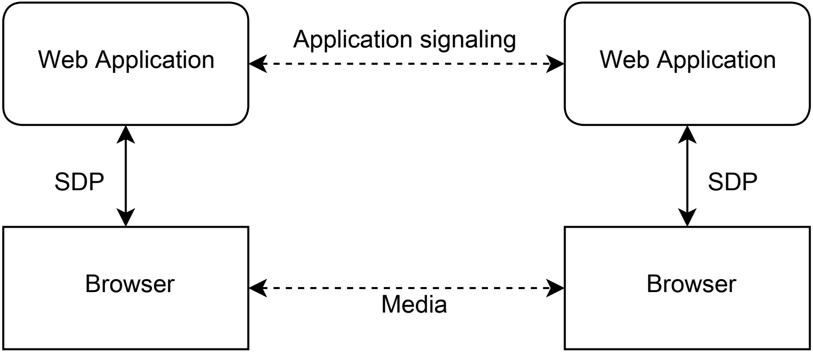


WebRTC implementation

- VP8 as de facto video codec
- G711 and Opus as audio codec
- H.264 video codec in some browser vendors
- Uses JSEP for signaling

Media codecs are still on an ongoing discussion

JSEP signaling model



WebRTC APIs

- Combines two high-level JavaScript APIs:
  - *GetUserMedia()*
  - *PeerConnection()*
- Uses different specific objects:
  - *MediaSream*
  - *MediaStreamTrack*
  - *DataChannel*
- Control and monitoring:
  - *getStats()* method is used for monitoring
  - JSON objects change the constraints for video and networking
- *Google Chrome* and *Mozilla Firefox* implement all the APIs
- *Opera* only includes *GetUserMedia*

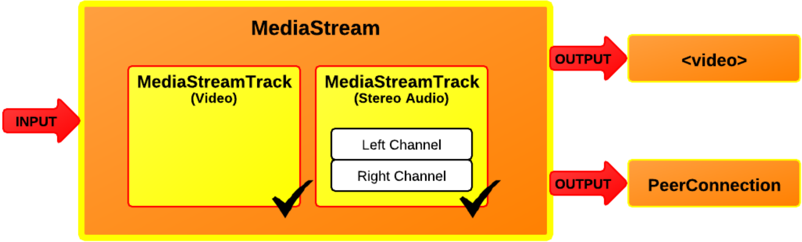
GetUserMedia()

Provides access to media devices and returns a *MediaStream* object.

```
navigator.webkitGetUserMedia(cameraConstraints(),
    gotStream, function() {
   console.log("GetUserMedia failed");
});

function gotStream(stream) {
   //Stream is the MediaStream object returned by the API
       and played in HTML local-video element
   console.log("GetUserMedia succeeded");
   document.getElementById("local-video").src =
       webkitURL.createObjectURL(stream);
}
```

MediaStream and *MediaStreamTrack*



PeerConnection()

Builds a point-to-point connection

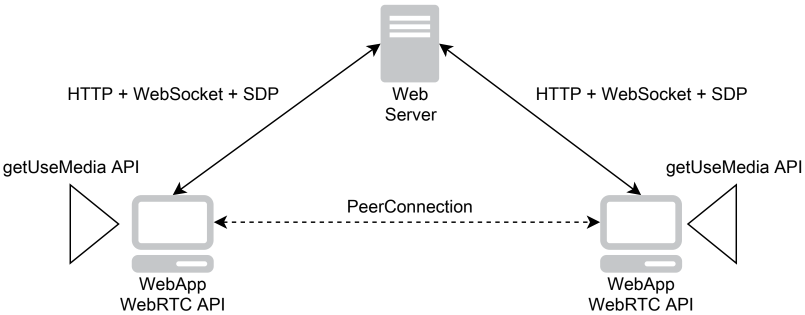
```
pc = new webkitRTCPeerConnection(pc_config);
pc.onicecandidate = iceCallback;
pc.addStream(localstream);
function iceCallback(event){
   if (event.candidate) {
      sendMessage(event.candidate);
   }
}
pc.addIceCandidate(new RTCIceCandidate(event.candidate));
pc.onaddstream = gotRemoteStream;
function gotRemoteStream(e){
   document.getElementById("remote-video").src =
       URL.createObjectURL(e.stream);
}
```

Topologies

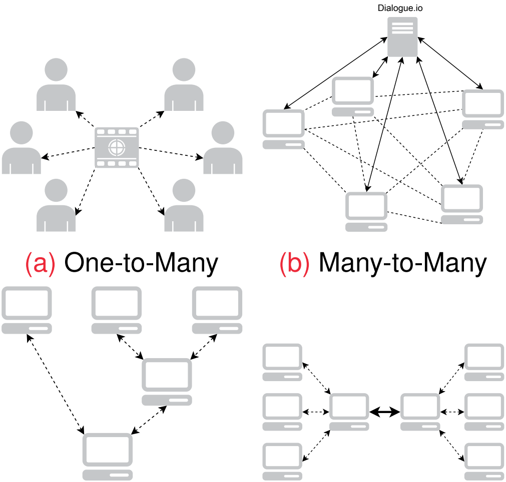
Possible topologies for real-time communication:

- Point-to-Point
- One-to-Many
- Many-to-Many
- MCU
- Overlay
  - Spoke-hub
  - Tree

Point-to-Point



Topologies



(a) One-to-Many (b) Many-to-Many

(c) Tree (d) Spoke-Hub

Performance Metrics

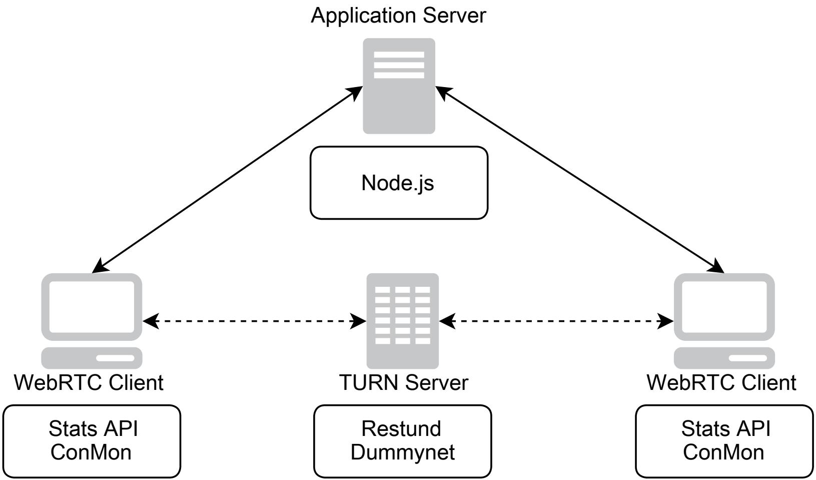
Network:

- Packet loss
- Round-Trip Time and One-Way Delay
- Throughput
- Inter-Arrival Time and Jitter

Host:

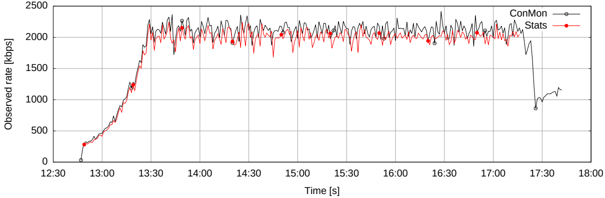
- Resources
- Setup time
- Call failure rate
- Encoding and decoding

Evaluation Environment

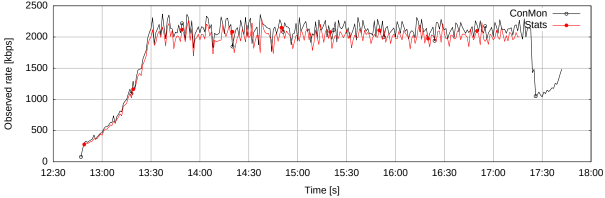


ConMon vs StatsAPI

- Incoming stream



- Outgoing stream



Congestion mechanisms in WebRTC

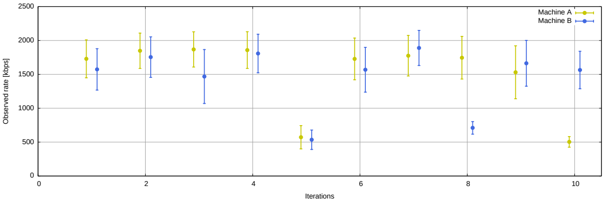
- UDP has difficulties for rate adaption
- Sender and Receiver RTCP reports used for rate adjustment
- Receiver Estimated Maximum Bitrate (REMB) extension for RTCP
- Utilizes specific Google algorithm for rate adaption based on packet loss
  - Under 2% increases rate
  - 2-10% reminds unchanged
  - +10% adapts rate based on packet loss ratio
- Rate limit by the TCP Friendly Rate Control (TFRC) formula

Benchmarks

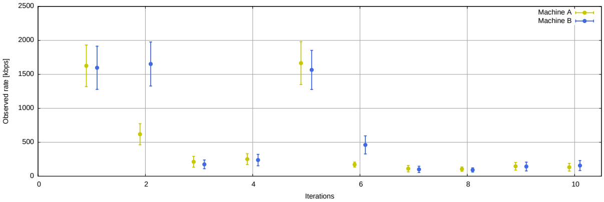
- Evaluation
  - Lossy environment
  - Delayed networks
  - Loss and delay combination
  - Varying bandwidth and queue size
- Real scenarios
  - Loaded networks
  - Parallel calls
  - Mesh topology
  - Mobile environments
  - Interoperability between browsers

Delayed network

- 100ms

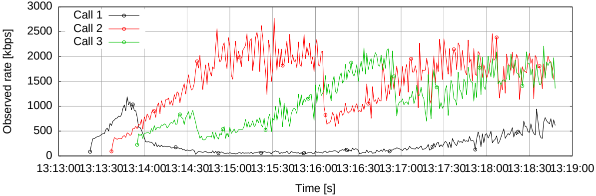


- 200ms

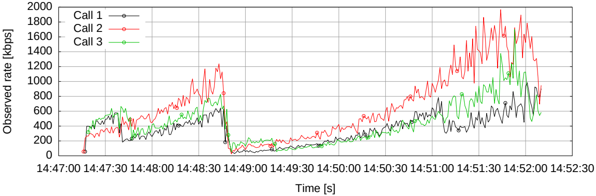


Three parallel calls - rate

- Asynchronous

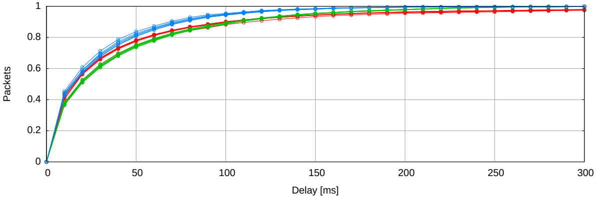


- Synchronous

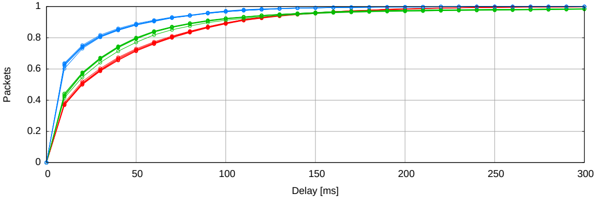


Three parallel calls - delay

- Asynchronous

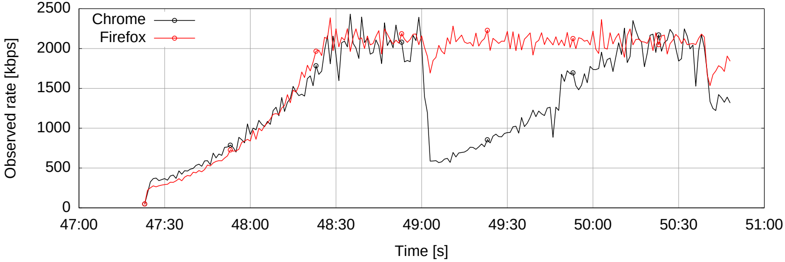


- Synchronous



Interoperability

This test evaluates the congestion mechanisms between *Chrome* and *Firefox*.



Firefox is not running any congestion mechanisms to adapt the rate based on the RTCP feedback, thus is only sending Receiver Reports to the sender.

Conclusion

- Congestion mechanisms have very bad response in delayed networks
- Interoperability cannot be achieved for low latency scenarios until *Firefox* enables all congestion mechanisms
- WebRTC has a very bad performance in multiple *PeerConnection* scenario regarding CPU managment
- Different features should be enabled in the existing APIs (overlay and media track forwarding)

Thank you!