Performance Evaluation of WebRTC-based Video Conferencing

Delft University of Technology
- Bart Jansen
- Fernando Kuipers

Columbia University
- Timothy Goodwin
- Varun Gupta
- Gil Zussman

IFIP WG 7.3 Performance 2017
Bart Jansen
WebRTC

- Web Real Time Communications
- Audio, Video and Data communication
- Standard by W3C and IETF since 2012

Benefits
- Plugin less
- Peer to peer
- Cross browser/platform
- Easy to use (API)
Research motivations

- WebRTC is a new emerging technology
  - Easy to use
  - Adobe Flash dying out
- Fast changing protocol under active development
- Study WebRTC’s performance for both simulated and real world conditions
WebRTC – Topologies
The way network nodes are arranged in a network

Peer to Peer

Server to Client

Meshed

SFU
WebRTC – Interoperability

- Adoption

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop</td>
<td>2,01%</td>
<td>22,06%</td>
<td>30,90%</td>
<td>36,53%</td>
<td>68,51%</td>
</tr>
<tr>
<td>Mobile and Tablet</td>
<td>0%</td>
<td>1,75%</td>
<td>12,32%</td>
<td>28,94%</td>
<td>41,59%</td>
</tr>
</tbody>
</table>

- Media codecs

- Browser compatibility
  - Support for older browsers with plugin
Congestion Control

Congestion occurs when a node carries more data than it can handle

- **Google Congestion Control Algorithm**
  - Dynamically adjusts send and receive rate

- **Receiver:** delay based

- **Sender:** loss based

![Image of congestion control diagram]

Sender-side controller

RTCP REMB

RTP

Arrival-time filter

Over-use detector

Adaptive threshold

Signal

Rate controller

\( A_s \)

\( A_r \)

\( m_i \)
Congestion Control

• Receiver side (delay based):

\[ A_r(i) = \begin{cases} 
\eta A_r(i - 1) & \text{Increase} \\
\alpha R(i) & \text{Decrease} \\
A_r(i - 1) & \text{Hold}
\end{cases} \]

\[ \eta = 1.05 \]
\[ \alpha = 0.85 \]

• Sender side (packet loss based):

\[ A_s(i) = \begin{cases} 
A_s(i - 1)(1 - 0.5f_l(i)) & f_l(i) > 0.1 \\
1.05A_s(i - 1) & f_l(i) < 0.02 \\
A_s(i - 1) & \text{otherwise}
\end{cases} \]

• Send rate never exceeds receive rate:

\[ A_s(i) = \min(A_s(i), A_r(k)) \]
Experimental test setup
Synthetic network conditions

- Custom video stream
- Network limiter
- Statistics via RTCStatsReport
Experimental test setup

• Avoid CPU and memory limitations
• 5 tests averaged

• Statistics:
  – Data rate (kbps)
  – Framerate (fps)
  – Resolution (width * height) (pixels)
  – Round-trip-time (ms)

• Source code available at:
  https://github.com/Wimnet/webrtc_performance
Performance Evaluation

- Baseline experiments
- Cross traffic
- Multi-party topologies
- Video codecs
- Mobile browsers
- Real wireless networks
Performance Evaluation

Network characteristics – Packet loss

- As expected
  - 5% converges to maximum data rate
  - > 10% converge to minimum data rate

![Data rate vs Time graph showing no loss, 5% packet loss, 10% packet loss, and 20% packet loss.]
Performance Evaluation
Network adaptability – Bandwidth

- 77% utilization
- Follows GCC rates after minute 1:
  \[1000 \times 1.05^{17} = 2300 \text{ kbps}\]
- Also after minute 3:
  \[500 \times 1.05^{18} = 1200 \text{ kbps}\]
Performance Evaluation
Cross traffic – Intra-protocol fairness

- Three separate calls with 2Mbps limit
- Fairness is reached with delay
Performance Evaluation

Cross traffic – Inter-protocol fairness

- 2Mbps limit
- Competing TCP flows
- Fairness can be improved
Performance Evaluation
Multi-party - Meshed

- 2, 3 and 4 person calls
- Results averaged on uplink and downlink
- CPU limitations
Performance Evaluation
Multi-party - SFU

- Less uplink bandwidth required
- Start up delay

![Graphs showing data rate over time for different SFU configurations]
Performance Evaluation

Video codec comparison

- VP8 (default), VP9 and H.264
- Room for improvement:
  - H.264 needs to balance between framerate and resolution
  - VP9 needs to scale up when congestion disappears
Experimental test setup

Wireless performance

• 3 nodes:
  – Local wireless node (NYC)
  – Local wired node (NYC)
  – Remote wired nodes (Oregon or Sydney)

• Varying:
  – AP transmission power (to 1mW)
  – Distance from AP (5ft – 25ft)
  – MAC retry limit
Performance Evaluation

Wireless network conditions

• 5ft vs 25ft AP distance

• Results
  – Higher RTTs results in more packet loss and lower quality
Performance Evaluation

Wireless network conditions

• Limit retransmissions in AP
  – From MAC retry limit 1 (min) and 15 (max)

• Results
  – Trade off between RTT and packet loss
  – Less video freezes
  – GCC relies too heavy on packet loss
Summary

• Improved browser interoperability

• Thorough evaluation of WebRTC
  – Custom video stream
  – Similar results

• Improved cross traffic fairness

• Poor performance over wireless due to:
  – Bursty losses
  – Packet retransmissions
Conclusions

- Heavily reliant on packet loss

- Multi-party
  - When more than 3 people: Add SFUs

- Room for improvement
  - Mobile performance
  - Video codecs
Future Work

• Take CPU limitations into account
  – Energy consumption (battery)
  – Call characteristics when CPU is exhausted

• Simulate Google Congestion Control

• Tests with cellular connections