Architecture of data exchange with minimal client-server interaction in multipoint video conferencing

Anton Saveliev
SPIIRAS

28 August 2014, NEW2AN 2014
St. Petersburg, Russia
Main problems associated with streaming and processing of video data in video conferencing systems

- Need for greater channel bandwidth for video transmission
- High server load in client-server video conferencing systems
- High client load in peer-to-peer video conferencing systems
- Network Address Translation
Main advantages and disadvantages of peer-to-peer video conferencing systems

- **Advantages:**
  
  - Low requirements for server hardware
  
  - Scalability

- **Disadvantages:**
  
  - High client load
  
  - The complexity of the management of connected clients
Architecture of data exchange in the video conferencing application

Client$_1$

- Video Camera
- Microphone

Web-page

- Peer
- Socket
- Ajax
- CSS
- HTML

WebRTC

WebSocket

HTTP

Socket

Node.js

Server

Database

...
The main web page functions:

- To create connections to the server
- To create connections to the clients
- To create, to process and to stream audio and video data
Server part

The main server functions:

- Building of the client-side applications
- Client registration
- Client authorization
- Chat rooms creation
- The ‘configuration message’ exchange between clients
- Working with the database
Protocols

- HTTP - transmission of web pages.
- WebScoket – transmission of ‘configuration messages’ between the clients and the server.
- WebRTC - streaming audio and video data between the clients.
The algorithm of preparation of client parts before forming the 'configuration message'

Start

The ‘caller’ client sends ‘start the call’ to the server

client name

Is the ‘responder’ client connected to server?

Yes

Send the call to the client

Answer the call or not?

Yes

Send ‘answer the call’ to the server

No

Send ‘reject the call’

End

Get ‘caller’ and ‘responder’ clients sockets

Get sockets from ‘chat rooms’ associated with ‘caller’ and ‘responder’ sockets
The algorithm of allocation of sockets and processing their buffers on the server

Start

Get socket from ‘responder’ socket ‘chat room’

name, socket

Get or create buffer with ‘responder’ socket name

buffer, socket

Get socket from ‘caller’ socket ‘chat room’

socket

Add socket to buffer

socket

Is this the last socket in the ‘chat room’?

Yes

End

No

Socket from ‘responder’ socket ‘chat room’ leaves its own ‘chat room’ and joins to ‘caller’ socket ‘chat room’

Buffer process

Is ‘chat room’ with ‘responder’ socket empty?

Yes

No

SPIIRAS
The algorithm of working of data buffer on the client

Start
Request or service data
Add service data or request to buffer
Are buffers busy?
No
Are buffers empty?
Yes
Send message ‘Buffer empty’ to the server
No
Get data or request from buffer
Service data
Connection is established
Create new connection with client using WebRTC protocol
End
Comparison of performance of the developed application and Skype

<table>
<thead>
<tr>
<th>Modes of work of application client’s part</th>
<th>RAM usage by client’s part of the developed application</th>
<th>RAM usage by client part of the application “Skype”</th>
<th>CPU usage by client part of the developed application</th>
<th>CPU usage by client part of the “Skype” application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verifying username and password mode</td>
<td>~ 8 000 Kb</td>
<td>~ 10 000 Kb</td>
<td>0%</td>
<td>1-2%</td>
</tr>
<tr>
<td>Standby mode</td>
<td>~ 11 000 Kb</td>
<td>~ 15 000 Kb</td>
<td>2-3%</td>
<td>2-4%</td>
</tr>
<tr>
<td>Mode of sending and receiving one audio and video stream</td>
<td>~ 17 000 Kb</td>
<td>~ 60 000 Kb</td>
<td>10-14%</td>
<td>12-15%</td>
</tr>
<tr>
<td>Mode of receiving four audio-video data streams and sending one audio-video data stream</td>
<td>~ 80 000 Kb</td>
<td>~ 250 000 Kb</td>
<td>25-40%</td>
<td>30-60%</td>
</tr>
</tbody>
</table>
Conclusion

- The architecture of the video conferencing application is asynchronous. It requires new data control algorithms for the connection via WebRTC protocol. During our research these algorithms were successfully developed and implemented in the application, which is capable:
  - to monitor the status of clients connected to the group;
  - to support the group calls.

Further research will be focused on simplifying and improving the audio and video processing and transmission using peer-to-peer connections to optimize the data processing load between clients.
Thank you!

- Address: 39, 14 Line,
  St. Petersburg, Russia, 199178
- Phone/Fax: +7 (812) 3287081
- E-Mail: saveliev@iias.spb.su
- Web: www.spiiras.nw.ru/speech