Agenda

- Floor control
- Putting it together
- Case study
Floor Control

- Definition
- Architecture
- Protocols
Definition

Floor control: a mechanism that enables the management of the joint or exclusive access to the shared resources inside a conference

.e.g. audio/video channels, slide bar presentation

Floor: “A temporary permission to access or manipulate a specific shared resource or set of resources”.
Architecture

- Three entities are involved in floor control:
  - Floor participant
  - Floor chair
  - Floor Control Server (FCS)
Architecture

- Two main algorithms
  - First come First Serve (FCFS)
  - Chair moderated
Protocols

1. Establish floor control connections between the different entities
   - Floor Chair
   - Floor Participant
   - Floor Control Server

2. Coordinate access to shared resources
   - Application Server
     - Manage
     - Notify
     - Floor Control Server

3. Control the FCS
   - Create/terminate floor
   - Add participant/resource to floor
   - Remove participant/resource from floor
Protocols

• Establish floor control connections between the different entities
  – SIP/SDP (RFC 4583, RFC 5239)

• Coordinate access to shared resources
  – Binary Floor Control Protocol (BFCP)
  – Talk Burst Control Protocol (TBCP)

• Control the FCS
  – Megaco/H.248
  – SIP Floor Server Control Markup Language (SIP-FSCML)
Establish floor control connections between the different entities

- **Examples** of an offer sent by a conference server to a client

  ```
  m=application 50000 TCP/TLS/BFCP *
  a=setup:passive
  a=connection:new
  a=floorctrl:s-only
  a=confid:4321
  a=userid:1234
  a=floorid:1 m-stream:10
  a=floorid:2 m-stream:11
  m=audio 50002 RTP/AVP 0
  a=label:10
  m=video 50004 RTP/AVP 31
  a=label:11
  ```
Establish floor control connections between the different entities

• **Examples** of an answer returned by the client
  
m=application 9 TCP/TLS/BFCP *
a=setup:active
a=connection:new
a=floorctrl:c-only
m=audio 55000 RTP/AVP 0
m=video 55002 RTP/AVP 31
Coordinate access to shared resources

- **Binary Floor Control Protocol (BFCP)**
  - Standardized in RFC 4582

- Negotiation of BFCP connections within SIP/SDP
  - Standardized in RFC 4583

- **Advantages**
  - Fast (binary encoded)
  - Secure
  - Reliable (over TCP)
  - Provides all the floor control functionalities
Binary Floor Control Protocol

- Protocol operations and messages/primitives
  - Participant operations
    - Request a floor (FloorRequest)
    - Cancel a floor request (FloorRelease)
    - Release a Floor (FloorRelease)
  - Chair operations
    - Grant a floor (ChairAction)
    - Deny a floor (ChairAction)
    - Revoke a floor (ChairAction)
  - Participant/Chair
    - Requesting Information about Floors (FloorQuery)
    - Requesting Information about Floor Requests (FloorRequestQuery)
    - Requesting Information about a User (UserQuery)
    - Obtaining the Capabilities of a Floor Control Server (Hello)
  - FCS operations
    - Handles the participant and chair requests (FloorRequestStatus, FloorStatus, UserStatus, ChairActionAck, HelloAck, Error)
Binary Floor Control Protocol

5- FloorRelease
  Transaction ID: 154
  User ID: 234
  FLOOR-REQUEST-ID: 789

6- FloorRequestStatus
  Transaction ID: 154
  User ID: 234
  Floor Request ID: 789
  Request Status: Released
  Floor ID: 543
Binary Floor Control Protocol

• Packet Format
  – BFCP messages
    • Consist of a common header followed by a set of attributes.
    • Use a TLV (Type-Length-Value) binary encoding
    • Floor participants, media participants, and floor chairs are identified by 16-bit user identifiers.
    • BFCP supports nested attributes (i.e., attributes that contain attributes).
      – Referred to as grouped attributes.
Binary Floor Control Protocol

- **Packet Format**

  **Common-header format**

<table>
<thead>
<tr>
<th>Ver</th>
<th>Reserved</th>
<th>Primitive</th>
<th>Payload Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conference ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction ID</td>
<td>User ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  32 Bits

  **Attribute format**

<table>
<thead>
<tr>
<th>Type</th>
<th>M</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Contents</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Binary Floor Control Protocol

- **Primitives**

<table>
<thead>
<tr>
<th>Value</th>
<th>Primitive</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FloorRequest</td>
<td>P -&gt; S</td>
</tr>
<tr>
<td>2</td>
<td>FloorRelease</td>
<td>P -&gt; S</td>
</tr>
<tr>
<td>3</td>
<td>FloorRequestQuery</td>
<td>P -&gt; S ; Ch -&gt; S</td>
</tr>
<tr>
<td>4</td>
<td>FloorRequestStatus</td>
<td>P &lt;- S ; Ch &lt;- S</td>
</tr>
<tr>
<td>5</td>
<td>UserQuery</td>
<td>P -&gt; S ; Ch -&gt; S</td>
</tr>
<tr>
<td>6</td>
<td>UserStatus</td>
<td>P &lt;- S ; Ch &lt;- S</td>
</tr>
<tr>
<td>7</td>
<td>FloorQuery</td>
<td>P -&gt; S ; Ch -&gt; S</td>
</tr>
<tr>
<td>8</td>
<td>FloorStatus</td>
<td>P &lt;- S ; Ch &lt;- S</td>
</tr>
<tr>
<td>9</td>
<td>ChairAction</td>
<td>Ch -&gt; S</td>
</tr>
<tr>
<td>10</td>
<td>ChairActionAck</td>
<td>Ch &lt;- S</td>
</tr>
<tr>
<td>11</td>
<td>Hello</td>
<td>P -&gt; S ; Ch -&gt; S</td>
</tr>
<tr>
<td>12</td>
<td>HelloAck</td>
<td>P &lt;- S ; Ch &lt;- S</td>
</tr>
<tr>
<td>13</td>
<td>Error</td>
<td>P &lt;- S ; Ch &lt;- S</td>
</tr>
</tbody>
</table>

S: Floor Control Server  P: Floor Participant  Ch: Floor Chair
Talk Burst Control Protocol

- **TBCP**
  - Defined by the OMA (Open Mobile Alliance)
  - Uses the application extension features of RTCP (RTP Control Protocol) in order to invoke floor control within the POC (Push to talk Over Cellular) environment.

- **Typical TBCP messages include:**
  - Talk Burst Granted
  - Talk Burst Request Message
  - Talk Burst Deny Message
  - Talk Burst Release Message
  - Talk Burst Taken
  - Talk Burst Idle
  - Talk Burst Revoke

- **Advantages**
  - Fast
  - Secure

- **Disadvantages**
  - Only provides basic floor control functionalities (e.g. no chair supported).
Control the FCS

- SIP-FSCML is a non-standard alternative to H.248
  - Less complex
  - Easy to understand and use by SIP application developers.

- It follows SIP and XML paradigms.

- It enables a peer-to-peer communication model between the AS and the FCS.
  - This allows the FCS to be simultaneously used by multiple ASs.
SIP Floor Server Control Markup Language

- Conceptual view
  - Floor
  - Floor Connection
  - Floor Session
SIP Floor Server Control Markup Language

- Conceptual view
  - Floor
  - Floor Connection
  - Floor Session
SIP Floor Server Control Markup Language

• The control session between the application and the FCS is opened through a SIP INVITE message.

• FSCML requests to the FCS are carried in SIP INFO messages
  – Each INFO message includes a single FSCML body
  – An FSCML body can carry any number of FSCML requests

• SIP-FSCML responses are transported in a separate INFO message

• SIP-FSCML is a request-response protocol; with only final responses

• SIP-FSCML relies on SIP subscribe/notify mechanism, to allow applications subscribe to floor control related events
SIP Floor Server Control Markup Language

- SIP-FSCML operations
  - Open/close control connection
  - Create floor
  - Create floor Connection
  - Add/remove floor to/from a conference
  - Set/update Chair for a floor
  - Add/remove floor participant(s)
  - Set floor algorithm
  - add/remove media to/from a floor
  - Set maximum floor holders
  - Set maximum floor holding time
SIP Floor Server Control Markup Language

- Example of FSCML body

```
<FloorServerControl>
  <conferenceid>the conference ID</conferenceid>
  <request type="CreateFloor">
    <floorid>the floor ID</floorid> (mandatory)
    <algorithm>the floor control algorithm</algorithm> (mandatory)
    <maxholders>max number of floor holders</maxholders> (optional, default=1)
    <maxholdingtime>max time (in seconds) a participant can hold a floor, in case someone else asked for it</maxholdingtime> (optional, default 0=unlimited)
  </request>
  <request type="SetChair">
    <floorid>floor whose chair should be set</floorid> (mandatory)
    <chairid>the chair ID</chairid>
  </request>
  <request type="AddParticipant">
    <floorid>id of the floor to which to add</floorid> (mandatory)
    <participantid>the participant ID</participantid> (mandatory)
  </request>
</FloorServerControl>
```
SIP FSCML - A Scenario

1: INVITE
2: OK
3: ACK
4: INFO(request:createFloor)
5: OK
6: INFO(response)
7: OK
8: INVITE(noSDP)
9: OK(remoteBFCP_SD)
10: INFO(CreateFloorConnection)
11: OK
12: INFO(response:localBFCP_SD)
13: Ack(localBFCP_SD)
14: OK
15: INFO(request:addParticipant)
16: OK
17: INFO(response)
18: OK

Open a control connection
Create floor
Create floor connection
Add participant to floor
Putting it together

- Architecture
- Server side scenario
- Client side scenario
Architecture

- 3GPP conferencing architecture

- Extension to 3GPP conferencing architecture
3GPP conferencing architecture

- The 3GPP IMS conferencing architecture uses the tightly-coupled (centralized) conference model

- Conferencing technical components
  - Signaling
    - SIP
  - Media control
    - H.248
  - Floor control (optional)
    - Floor control connections’ establishment
      - SIP/SDP
    - Coordination of access to shared resources
      - Binary Floor Control Protocol (BFCP)
    - Control of the FCS
      - H.248
3GPP conferencing architecture

- **AS**: Cr: Mediactrl command
- **MRFC**: Mp: H.248
- **MRFP Mixer/FCS (FCS optional)**
- **S-CSCF**: ISC: SIP (2)
- **P-CSCF**: Mw: SIP
- **Gm: SIP**
- **Mb: RTP/ BFCP**
- **MGW**
- **UE**

Network components:
- 3G Network
- MRFP Mixer/FCS (FCS optional)
- P-CSCF
- S-CSCF
- MGW
- UE

Service components:
- Parlay/ OSA AS
- Parlay/ OSA SCS
- Parlay X API
- Parlay X WS

Protocols:
- Gm: SIP
- ISC: SIP
- Mr: SIP (2)
- ISC: SIP
- Mr: SIP (2)
- ISC: SIP
3GPP conferencing architecture

• Limitations
  – The FCS is located inside the MRFP
    • The MRFP has to host a brand new functional entity
  – There is no interface between the two MRFP and the FCS
    • MRFP and floor control node have to be bought from the same supplier
  – No API is provided for application development
Extension to 3GPP conferencing architecture

- Separate FCS from media server
- Enables the use of emerging industry standards such as SIP-MSCML for Media server control.
- Uses SIP–FSCML as an alternative to Megaco/H.248 for FCS control
- Includes a comprehensive set of server side and client side API that exposes the floor control capabilities
Server side scenario

1: INVITE(noSDP)
2: OK(remoteSDP)
3: Ack(localBFCP_SD)

- Open a floor control connection
- Create a conference and reserve resources
- Negotiate SDP
- Add participant to the conference
- Get MS SDP
- createFloor, addParticipant
- get FCS BFCP_SD

Subscribe to floor events
Server side scenario

1: INVITE
2: OK
3: Ack
4: INVITE
5: OK
6: Ack
7: INVITE(noSDP)
8: OK(remoteSDP)
9: INVITE(remoteSDP)
10: OK(localSDP)
11: Ack
12: INFO(request:createFloor, addParticipant)
13: OK
14: INFO(response:localBFCP)
15: Ack(localBFCP_SDP)
16: OK
17: SUBSCRIBE
18: OK

Open a floor control connection
Create a conference and reserve resources
- Negotiate SDP
- Add participant to the conference
Server side scenario

1: BFCP: FloorRequest

2: NOTIFY(GrantMedia)

3: OK

4: INFO(conf,<config_leg type=talker>)

5: OK

6: INFO(response)

7: OK

8: INFO(response)

9: OK

10: BFCP: floorRequestStatus(granted)

After some time

Ask the MS to grant media to the user to who the floor is granted

The application inform the FCS about the results of granting media
Client side scenario

1: localBFCP_SDP

2: FloorRequest(confid, userid, floorid)
3: FloorRequestStatus(Pending)
4: FloorRequestStatus(Granted)
5: FloorRelease(confid, userid, floorid)
6: FloorRequestStatus(Released)
Case study

- Multiparty Multimedia applications’ development
- Game semantics
- Implementation architecture
- Prototype
- Performance evaluation
Multiparty Multimedia applications’ development

• APIs
  – Standard APIs
    • 3GPP & Parlay forum
      – Parlay/OSA
      – Parlay X web services
    • Java Community Process (JCP)
      – A set of JSRs
        » E.g. JSR 289 (SIP Servlet), JSR 309
  – Non standard APIs
    • E.g. A comprehensive set of server side and client side floor control APIs

• Tool kits
  – Ericsson Service Development Studio (SDS)
  – Open IMS Core from Fraunhofer Institute for Open Communication Systems (FOKUS)
Game semantics

• “Capture the Flag” is
  – A multi-party game
  – Played by 2 teams
  – Each team has a base and a flag inside the base
  – Teams fight to kill the opponents and capture their flag
  – The first team that captures the flag wins the game

• Revised “Capture the Flag”
  – Players can communicate
  – Each team has one chance to bomb a zone, i.e. kill everybody within the range except the one that bombs
  – Each team has a captain (which can manage talking and bomb floors)
  – The team should at least kill one opponent before capturing the flag
  – Some announcements are played during the game
    • E.g. start game, enemy killed, game over
Implementation architecture

• The game server uses a high-level conference API we designed
  – We have designed this API because of the unavailability of high level API for conferencing in IMS.
  – The API is built on:
    • JSR 289
    • JSR 309
    • floor control API

• The game client uses ICP API and the client side floor control API.
Implementation architecture

- There are two types of interfaces between the client and the server:
  - Game control interface
    - The game control messages are carried in the body of SIP Message in an XML format.
    - The game control messages:
      - startGame,
      - playerMove
      - zoneUpdate
      - shoot
      - Bomb
      - shoot and bomb responses
      - enemyDead
      - capture
      - gameOver
  - Conferencing interface
    - INVITE, BYE, ...

```xml
<startGame>
  <maxBullets>6</maxBullets>
  <localPlayerId>123</localPlayerId>
  <teamPlayers>
    <playerId>456</playerId>
    <playerId>356</playerId>
  </teamPlayers>
  <teamId>Angels</teamId>
  <zoneId x=0, y=0 />
</startGame>
```
Implementation architecture

- Scenario
Prototype

• Network nodes
  – IMS network: simulated using SDS
  – SIP AS: GlassFish/Sailfin
  – FCS implement in a previous work
  – Media server: MRFP prototype from Ericsson

• APIs
  – We implemented the high-level conference API using a subset of JSR 309 implementation
  – Reused the server-side floor control API implemented before
  – We used the SDS implementation of ICP API

• The game server is deployed as a SIP Servlet in the AS
Prototype

• Server GUI

• Client GUI
Performance evaluation

Experiment setup

Sony Ericsson P1

Game Server

IMS Network

Ericsson MRFP

AS
CSCF/HSS (SDS)

FCS

• Intel Dual Core
• 2G RAM
• Windows 2000

• Intel Dual Core
• 2G RAM
• Red Hat Linux
Performance evaluation

Conference establishment code footprint: with and without using IMS high-level API

<table>
<thead>
<tr>
<th>JSR 289+JSR 309</th>
<th>high-level IMS API</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 20 lines</td>
<td>~ 3 lines</td>
</tr>
</tbody>
</table>

// create conference service
ImsConferenceService confService = ImsServiceManager.getInstance().createService(id, "Conferencing");

// create conference policy
ImsConferencePolicy confPolicy = service.createConferencePolicy(startTime,duration, maxParties,..);

// create a conference with participants
ImsConference conference = confService.createConference(confPolicy, partyList);
Performance evaluation

Conference establishment delays: with and without using IMS high-level API

<table>
<thead>
<tr>
<th>Participants</th>
<th>jsr289+jsr309</th>
<th>ims API</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 laptops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 cell phones</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 cell phones</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average Game start and stop delay in server side

<table>
<thead>
<tr>
<th>Delay (ms)</th>
<th>3 laptops</th>
<th>3 cell phones</th>
<th>4 cell phones</th>
<th>5 cell Phones</th>
<th>5 cells +1 laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Start delay</td>
<td>1439</td>
<td>2902</td>
<td>3311</td>
<td>3786</td>
<td>3057</td>
</tr>
<tr>
<td>Game Stop Delay</td>
<td>151</td>
<td>900</td>
<td>757</td>
<td>868</td>
<td>963</td>
</tr>
</tbody>
</table>

Average game response time in client side

<table>
<thead>
<tr>
<th>Response Time (ms)</th>
<th>Talk</th>
<th>Req Bomb</th>
<th>Shoot</th>
<th>Capture</th>
<th>Bomb</th>
<th>Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop</td>
<td>1746</td>
<td>1810</td>
<td>133</td>
<td>172</td>
<td>178</td>
<td>31</td>
</tr>
<tr>
<td>Cellular</td>
<td>2368</td>
<td>3001</td>
<td>1008</td>
<td>2453</td>
<td>1661</td>
<td>902</td>
</tr>
</tbody>
</table>
References


