D6.4

Version	0.3
Author	Noam Amram
Dissemination	PU
Date	30/12/2016
Status	Final

D6.4: NUBOMEDIA TV Broadcast Connector Demonstrator

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2017-01-31
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP6: Demonstration
WP leader	Noam Amram
Deliverable nature:	Report
Lead editor:	Teo Redondo / Paula Collazos (ZED)
Planned delivery date	10/2016
Actual delivery date	31/12/2016
Keywords	NUBOMEDIA, Demonstrator

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 610576

FP7 ICT-2013.1.6. Connected and Social Media

DISCLAIMER

All intellectual property rights are owned by the NUBOMEDIA consortium members and are protected by the applicable laws. Except where otherwise specified, all document contents are: “© NUBOMEDIA project - All rights reserved”. Reproduction is not authorized without prior written agreement. All NUBOMEDIA consortium members have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the owner of that information.

All NUBOMEDIA consortium members are also committed to publish accurate and up to date information and take the greatest care to do so. However, the NUBOMEDIA consortium members cannot accept liability for any inaccuracies or omissions nor do they accept liability for any direct, indirect, special, consequential or other losses or damages of any kind arising out of the use of this information.

Contributors:

Teo Redondo (ZED)
Luis López Fernández (URJC)
Noam Amram (LiveU)

Internal Reviewer(s):

Luis López Fernández (URJC)

Version History

Version	Date	Authors	Sections Affected
0.1	14/10/2016	Teo Redondo / Paula Collazos	All (Structure of the Content)
0.2	30/12/2016	Noam Amram	All (Content)
0.3	31/12/2016	Noam Amram	3,4
0.4	29/1/2017	Noam Amram	Overall improvements

Table of contents

Executive summary

This demonstrator provides News Room application deployed on Nubomedia PaaS. News Room is a WebRTC-based application with a Room management, user registry and authentication, streaming media to LiveU cloud services. Given application is based on NUBOMEDIA Media Server API and LiveU TV Broadcasting Connector API.

The objective of the TV Broadcasting Connector API is to hide the complexity of the service media path by providing an easy interface to connect among various stakeholders. This means that anyone would be able to connect to a LiveU using this connector and to provide services to over 2000 broadcasters.

The TV Broadcasting Connector will connect streams coming from such services to a dedicated delivery system that will handle the media distribution in an effective manner to the broadcaster's studio. The impact could bring novel social and connected TV services to be used by many users.

Introduction

LiveU offers to his customers high-quality and reliable live video acquisition, management and distribution over IP. Its award-winning technology enables live, wireless video transmission from any location around the world with lightweight, easy-to-use equipment. From transmission backpacks to smartphones, and satellite/cellular hybrid to external antenna solutions, LiveU offers a complete range of devices for live, mobile video transmission anytime, anywhere. In addition, LiveU offers extensive cloud-based management and video distribution solutions.

LiveU's future offering can be enhanced by NUBOMEDIA infrastructures and with a clear path for exploitation of our work within the project to make new sources of revenues. Our main aim is to enhance the richness and liveness of available content to the broadcasters as well as to the end users, by allowing broader content/live and stored feeds generation models not available today. One important aspect in our future offering will be moving from the old paradigm of Broadcasters creating content for themselves (i.e. a reporter working for CNN creating content to CNN only) into broader service models as follows:

Consumers → Broadcasters (Main use case) "NewsRoom": this scenario goal is to connect the broadcasters with the consumers in order to increase further the richness and freshness of the content, when an unplanned event with public interest occur, consumers are the first to be there (Service name is "NewsRoom" and in some forms "BeFirst"), LiveU are now developing technologies jointly in NUBOMEDIA and with FP7 COMPEIT project to allow any user using their mobile browser and Web RTC streaming to send video to broadcasters through LiveU cloud services deployed in NUBOMEDIA cloud, this scenario could be seen as a broadened of the first case shifting from professional reporters to armature consumers while supporting of NUBOMEDIA main goals of using open API's and WebRTC while reaching the mass market of consumers and through technology collaboration with brother project.

1 Technological objectives of the demonstrator

The demonstrator's objectives created by LiveU are following:

Real-time communication including:

- Conferencing between TV Channel room participants;
- WebRTC media streaming, used as participant endpoints;
- URI media streaming (RTSP, files) used for Room;
- RTP media streaming, for LiveU network connectivity;
- Chat messaging between participants;
- Geographic location sharing.

Participant roles

There 3 main participants categories:

- Public - anyone who has access to WWW. They have access to demonstrator channel rooms in read-only way;
- Reporter – people who share live content to TV Broadcaster. They can be paid reporters or freelancers who want to either get recognition or perhaps allow broadcasters to purchase their content through content exchange platforms;
- Producer – TV Broadcaster stuff, they main function is reporter management and administration.

2 Requirements that the demonstrator must satisfy

We use the following terminology to classify the order of importance of requirements.

- Must: Shows a feature considered essential for the success of the applications. Without this feature the application success could be in risk.
- Relevant: Shows a feature considered important for the application.
- Desirable: Shows a feature that might be interesting for some scenarios, but whose real interest would need further analysis.

1. Scalability
    a. MUST: scale up to 100 simultaneous users (100 simultaneous video feeds from freelancers, not necessarily from the same event)
    b. RELEVANT: scale up to 1,000 simultaneous users
    c. DESIRABLE: scale up to 10,000 simultaneous users

2. Communication capabilities
    a. MUST: Support of WebRTC input stream from consumers browser
    b. MUST: Support transcoding of WebRTC encoded video stream to a lower bandwidth to support preview mode.
    c. MUST: Point to Multipoint Distribution of up to 100 streams over WebRTC.
    d. RELEVANT: Connectivity with at least 1 different Content Distribution Network to support mass point to multipoint distributions (perhaps millions of simultaneously consumers)
    e. DESIRABLE: Bonding filters to merge the video stream delivered over multiple channels.
    f. DESIRABLE: FEC filters to reconstruct the original video stream in case of packet loss.
    g. DESIRABLE: low delay feedback stream to enable fast adaptive video generation at the source

3. VCA capabilities
    a. DESIRABLE: object tracking for AR marking, to help mark an object area of interest.
    b. DESIRABLE: automatic lip reading, to be used for subtitles

4. AR capabilities
    a. DESIRABLE: Adding subtitles (based on lip reading)
    b. DESIRABLE: Adding sign language translation (based on video analysis of the sign language)
    c. DESIRABLE: Marking interesting objects in the scene.

5. Management capabilities
    a. DESIRABLE: user management (registration, banning, etc.)

6. Other features

a. RELEVANT: Storing the video streams.
b. DESIRABLE: Security, to ensure only authorized people can view the video streams.
c. DESIRABLE: Multiplatform (availability on Android, iOS and WWW)
d. DESIRABLE: Multidevice (same user may connect to its account from different devices)

3 Architecture of the demonstrator (related to NUBOMEDIA architecture)

Architecture of the demonstrator is shown below:

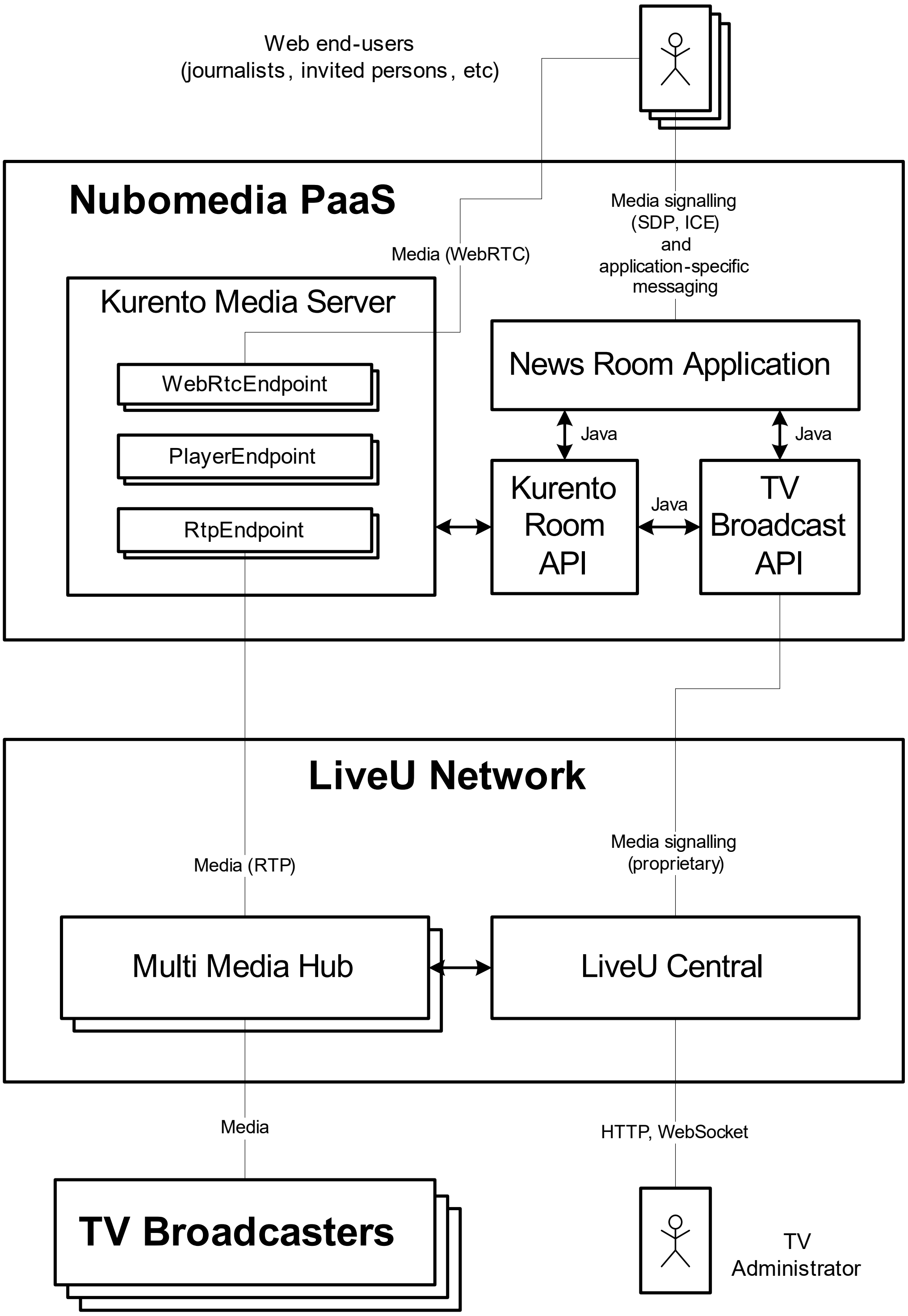


Figure 1. Demonstrator Architecture

News Room application can be deployed to Nubomedia PaaS or to any platform which supports Java SE 1.7 and greater. Connection to Kurento Media Server is required.

Application depends on such NUBOMEDIA facilities:

- Kurento Media Server;
- Kurento Room API;
- WebRtcEndpoint for web browser support;
- PlayerEndpoint for playing media streams by URI and RTSP;
- RtpEndpoint for media streaming to LiveU Multi Media Hub.

News Room application can be logically divided to server and client parts.

Client part is a Single-Page Application utilizing following:

- WebRTC stack in browser stack (FireFox or Chrome recommended);
- Google JavaScript WebRTC adapter library;
- Kurento JavaScript util library;
- News Room specific HTML pages and JavaScript scripts for end-user GUI and custom News Room application logic.

Server part of the News Room application is a Java-based Web Application. It is utilizing following:

- Java Spring (Boot) framework;
- Kurento Client Java library;
- TV Broadcast Connector library;
- News Room application classes with application logic.

Client-server interaction is based on JSON-RPC protocol over WebSocket protocol. It is used for media signaling, Room API messages and custom messages for application logic.

4 Relevant implementation details

News Room Demonstrators acts as Kurento Room application with following custom features added:

- User must be authenticated by New Room application;
- There are 2 kind or rooms:
    - **News Room**, accessed by participants, who are sharing media;
    - **Producer Room** – private room for selected for live TV streaming.
- Participants can be moved by Producers from News Room to Producer Room;
- Participants can be invited to Producer Room by URL shared by Producer;
- Participants media can be directed by Producer to LiveU network;
- RTSP or file media stream can be added to News Room;
- Participant geographic location can be shared between Participants;
- Chat messaging support between News Room participants.

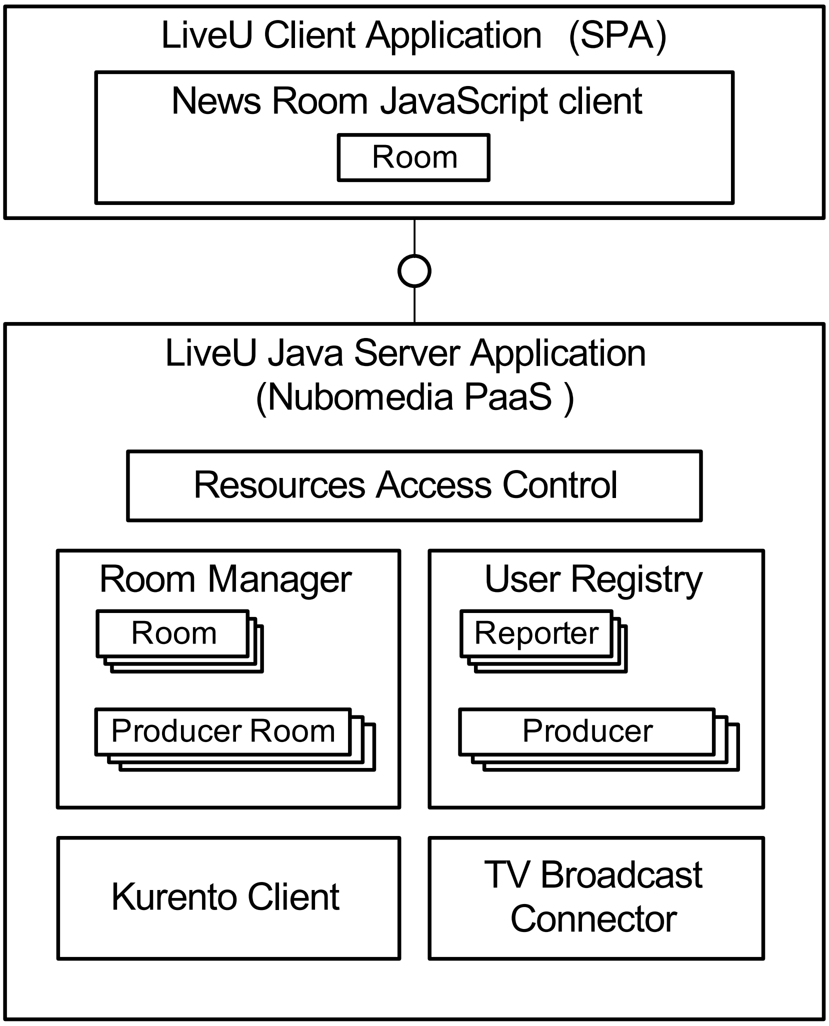
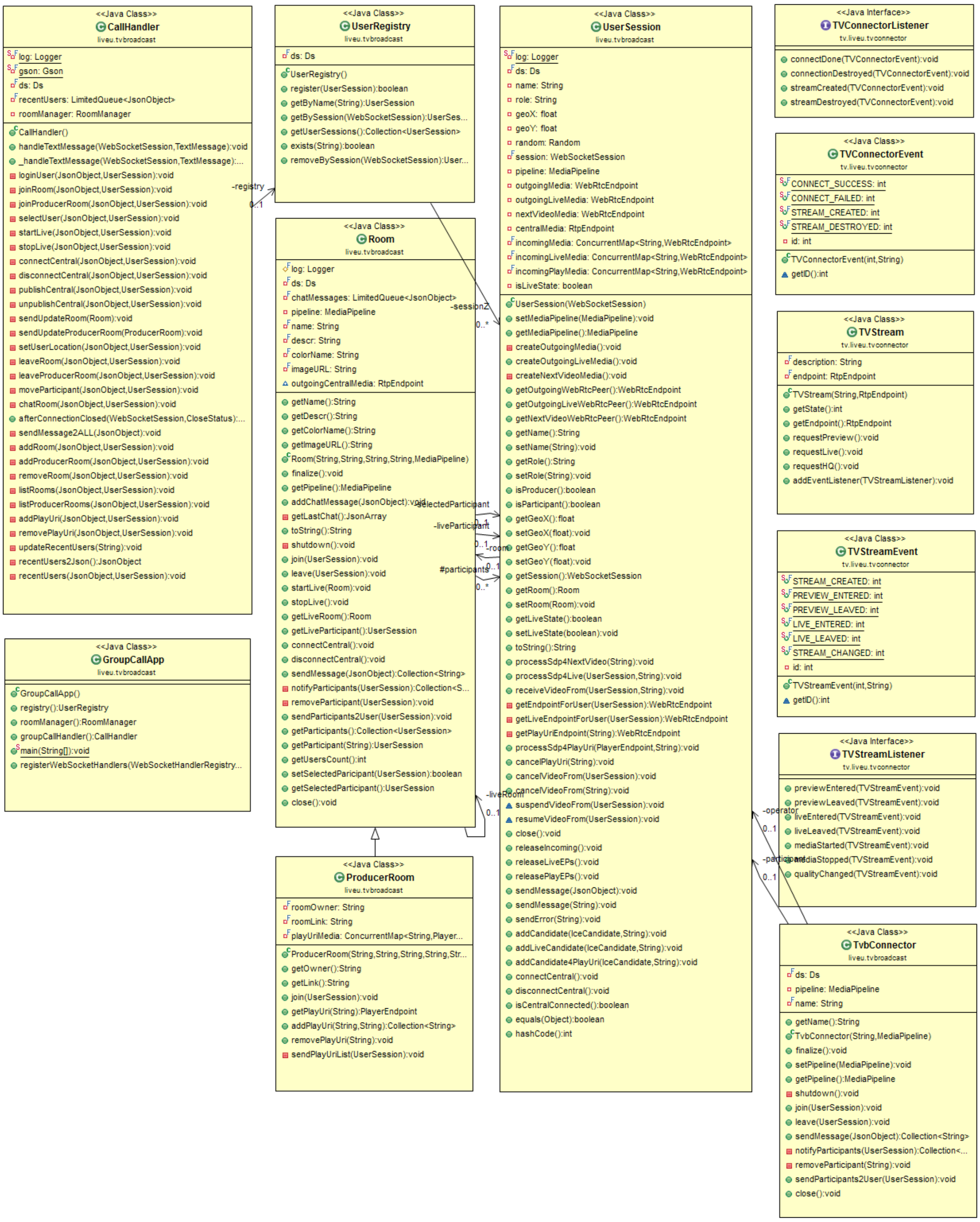


Figure 2. News Room application architecture

4.1 News Room application messages

Application logic extends Kurento WebSocket Room API by a set of JSON-RPC messages. Main of them are:

- **login** – log participant into system;
- **joinProducerRoom** – participant intended to enter to producer room;
- **leaveProducerRoom** – participant leaves producer room;
- **removeProducerRoom** – remove Producer private room;
- **geo** – send user geographical coordinates;
- **chat** – send message to Room chat;
- **connectCentral** – connect to LiveU Central;
- **disconnectCentral** – disconnect from LiveU Central;
- **publishCentral** – publish media to LiveU Media Network;
- **unpublishCentral** – unpublish media from LiveU Media Network;
- **addPlayUri** – add media by URI (PlayerEndpoint) to room;
- **removePlayUri** – remove media by URI from room;
- **startLive** – starts live with LiveU Media Network;
- **stopLive** – stops live with LiveU Media Network;
- **moveParticipant** – move participant from one room to another.


<<Java Class>>
CallHandler
liveu.tvbroadcast
log: Logger
gson: Gson
ds: Ds
recentUsers: LimitedQueue<JsonObject>
roomManager: RoomManager
CallHandler()
handleTextMessage(WebSocketSession,TextMessage):void
_handleTextMessage(WebSocketSession,TextMessage):...
loginUser(JsonObject,UserSession):void
joinRoom(JsonObject,UserSession):void
joinProducerRoom(JsonObject,UserSession):void
selectUser(JsonObject,UserSession):void
startLive(JsonObject,UserSession):void
stopLive(JsonObject,UserSession):void
connectCentral(JsonObject,UserSession):void
disconnectCentral(JsonObject,UserSession):void
publishCentral(JsonObject,UserSession):void
unpublishCentral(JsonObject,UserSession):void
sendUpdateRoom(Room):void
sendUpdateProducerRoom(ProducerRoom):void
setUserLocation(JsonObject,UserSession):void
leaveRoom(JsonObject,UserSession):void
leaveProducerRoom(JsonObject,UserSession):void
moveParticipant(JsonObject,UserSession):void
chatRoom(JsonObject,UserSession):void
afterConnectionClosed(WebSocketSession,CloseStatus):...
sendMessage2ALL(JsonObject):void
addRoom(JsonObject,UserSession):void
addProducerRoom(JsonObject,UserSession):void
removeRoom(JsonObject,UserSession):void
listRooms(JsonObject,UserSession):void
listProducerRooms(JsonObject,UserSession):void
addPlayUri(JsonObject,UserSession):void
removePlayUri(JsonObject,UserSession):void
updateRecentUsers(String):void
recentUsers2Json():JsonObject
recentUsers(JsonObject,UserSession):void
-registry
0..1
<<Java Class>>
GroupCallApp
liveu.tvbroadcast
GroupCallApp()
registry():UserRegistry
roomManager():RoomManager
groupCallHandler():CallHandler
main(String[]):void
registerWebSocketHandlers(WebSocketHandlerRegistry...
<<Java Class>>
UserRegistry
liveu.tvbroadcast
ds: Ds
UserRegistry()
register(UserSession):boolean
getByName(String):UserSession
getBySession(WebSocketSession):UserSes...
getUserSessions():Collection<UserSession>
exists(String):boolean
removeBySession(WebSocketSession):User...
<<Java Class>>
Room
liveu.tvbroadcast
log: Logger
ds: Ds
chatMessages: LimitedQueue<JsonObject>
pipeline: MediaPipeline
name: String
descr: String
colorName: String
imageURL: String
outgoingCentralMedia: RtpEndpoint
getName():String
getDescr():String
getColorName():String
getImageURL():String
Room(String,String,String,String,MediaPipeline)
finalize():void
getPipeline():MediaPipeline
addChatMessage(JsonObject):void
getLastChat():JsonArray
toString():String
shutdown():void
join(UserSession):void
leave(UserSession):void
startLive(Room):void
stopLive():void
getLiveRoom():Room
getLiveParticipant():UserSession
connectCentral():void
disconnectCentral():void
sendMessage(JsonObject):Collection<String>
notifyParticipants(UserSession):Collection<S...
removeParticipant(UserSession):void
sendParticipants2User(UserSession):void
getParticipants():Collection<UserSession>
getParticipant(String):UserSession
getUsersCount():int
setSelectedParicipant(UserSession):boolean
getSelectedParticipant():UserSession
close():void
-sessionZ
0..*
-selectedParticipant
0..1
-liveParticipant
0..1
-room
0..1
#participants
0..*
-liveRoom
0..1
<<Java Class>>
ProducerRoom
liveu.tvbroadcast
roomOwner: String
roomLink: String
playUriMedia: ConcurrentMap<String,Player...
ProducerRoom(String,String,String,String,Str...
getOwner():String
getLink():String
join(UserSession):void
getPlayUri(String):PlayerEndpoint
addPlayUri(String,String):Collection<String>
removePlayUri(String):void
sendPlayUriList(UserSession):void
<<Java Class>>
UserSession
liveu.tvbroadcast
log: Logger
ds: Ds
name: String
role: String
geoX: float
geoY: float
random: Random
session: WebSocketSession
pipeline: MediaPipeline
outgoingMedia: WebRtcEndpoint
outgoingLiveMedia: WebRtcEndpoint
nextVideoMedia: WebRtcEndpoint
centralMedia: RtpEndpoint
incomingMedia: ConcurrentMap<String,WebRtcEndpoint>
incomingLiveMedia: ConcurrentMap<String,WebRtcEndpoint>
incomingPlayMedia: ConcurrentMap<String,WebRtcEndpoint>
isLiveState: boolean
UserSession(WebSocketSession)
setMediaPipeline(MediaPipeline):void
getMediaPipeline():MediaPipeline
createOutgoingMedia():void
createOutgoingLiveMedia():void
createNextVideoMedia():void
getOutgoingWebRtcPeer():WebRtcEndpoint
getOutgoingLiveWebRtcPeer():WebRtcEndpoint
getNextVideoWebRtcPeer():WebRtcEndpoint
getName():String
setName(String):void
getRole():String
setRole(String):void
isProducer():boolean
isParticipant():boolean
getGeoX():float
setGeoX(float):void
getGeoY():float
setGeoY(float):void
getSession():WebSocketSession
getRoom():Room
setRoom(Room):void
getLiveState():boolean
setLiveState(boolean):void
toString():String
processSdp4NextVideo(String):void
processSdp4Live(UserSession,String):void
receiveVideoFrom(UserSession,String):void
getEndpointForUser(UserSession):WebRtcEndpoint
getLiveEndpointForUser(UserSession):WebRtcEndpoint
getPlayUriEndpoint(String):WebRtcEndpoint
processSdp4PlayUri(PlayerEndpoint,String):void
cancelPlayUri(String):void
cancelVideoFrom(UserSession):void
cancelVideoFrom(String):void
suspendVideoFrom(UserSession):void
resumeVideoFrom(UserSession):void
close():void
releaseIncoming():void
releaseLiveEPs():void
releasePlayEPs():void
sendMessage(JsonObject):void
sendMessage(String):void
sendError(String):void
addCandidate(IceCandidate,String):void
addLiveCandidate(IceCandidate,String):void
addCandidate4PlayUri(IceCandidate,String):void
connectCentral():void
disconnectCentral():void
isCentralConnected():boolean
equals(Object):boolean
hashCode():int
<<Java Interface>>
TVConnectorListener
tv.liveu.tvconnector
connectDone(TVConnectorEvent):void
connectionDestroyed(TVConnectorEvent):void
streamCreated(TVConnectorEvent):void
streamDestroyed(TVConnectorEvent):void
<<Java Class>>
TVConnectorEvent
tv.liveu.tvconnector
CONNECT_SUCCESS: int
CONNECT_FAILED: int
STREAM_CREATED: int
STREAM_DESTROYED: int
id: int
TVConnectorEvent(int,String)
getID():int
<<Java Class>>
TVStream
tv.liveu.tvconnector
description: String
endpoint: RtpEndpoint
TVStream(String,RtpEndpoint)
getState():int
getEndpoint():RtpEndpoint
requestPreview():void
requestLive():void
requestHQ():void
addEventListener(TVStreamListener):void
<<Java Class>>
TVStreamEvent
tv.liveu.tvconnector
STREAM_CREATED: int
PREVIEW_ENTERED: int
PREVIEW_LEAVED: int
LIVE_ENTERED: int
LIVE_LEAVED: int
STREAM_CHANGED: int
id: int
TVStreamEvent(int,String)
getID():int
<<Java Interface>>
TVStreamListener
tv.liveu.tvconnector
previewEntered(TVStreamEvent):void
previewLeaved(TVStreamEvent):void
liveEntered(TVStreamEvent):void
liveLeaved(TVStreamEvent):void
mediaStarted(TVStreamEvent):void
mediaStopped(TVStreamEvent):void
qualityChanged(TVStreamEvent):void
-operator
0..1
-participant
0..1
<<Java Class>>
TvbConnector
liveu.tvbroadcast
ds: Ds
pipeline: MediaPipeline
name: String
getName():String
TvbConnector(String,MediaPipeline)
finalize():void
setPipeline(MediaPipeline):void
getPipeline():MediaPipeline
shutdown():void
join(UserSession):void
leave(UserSession):void
sendMessage(JsonObject):Collection<String>
notifyParticipants(UserSession):Collection<...
removeParticipant(String):void
sendParticipants2User(UserSession):void
close():void

4.2 TV Broadcast Connectivity

TV Broadcast Connector is out of scope of this document and described in WP4 deliverables, however we need to elaborate here a bit to complete our implementation description. For media streaming from Kurento Media Server to LiveU's multimedia hub RTP protocol is used, which is provided by RtpEndpoint abstraction from NUBOMEDIA side. For media signaling proprietary LiveU's protocol is used between the Connector and the LiveU Central. Thus the connector receives signaling requests from the Newsroom application, to add and remove sessions from the connector, the connector is receiving WebRTC streams from users browser and connect to the Central via proprietary protocol for register the stream, in parallel, the stream is transferred to the MMH- Multi-media hub, in RTP protocol.

5 Testing mechanisms and methodologies

We have made the necessary tests to meet with the MUST requirements specified in section 2. In particular we have tested the scalability by using a single PC instalment of the server, and up to 30 users where supported, taking into considerations the scalability of the PaaS that was tested with up to 200 machines, we can conclude that the scalability requirement for 100 streams, including distribution of 100 streams to 100 clients is fully achieved.

The WebRTC itself provides the bandwidth modifications and thus support the lower channel condition of some of the participants, in that case we can conclude that all our MUST requirements are fully fulfilled by the platform.

In Addition we have started to collaborate with MONROE EU Project that provides a mobile broadband network, with 400 mobile nodes deployed on busses, on trains and also static nodes in various locations, we are testing the implementation of the Newsroom in various channel conditions, to check the adaptively. As part of this collaboration we are also disseminating NUBOMEDIA and we are installing on the nodes NUBOMEDIA Client API's to be used by other experimenters.

Scalability Test Setup:

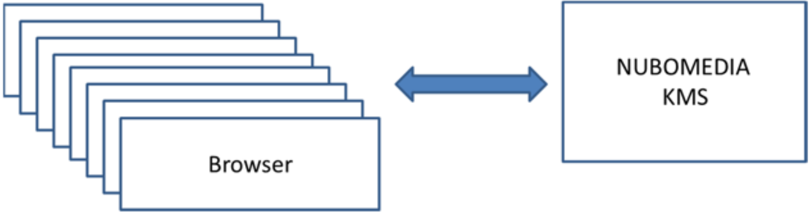
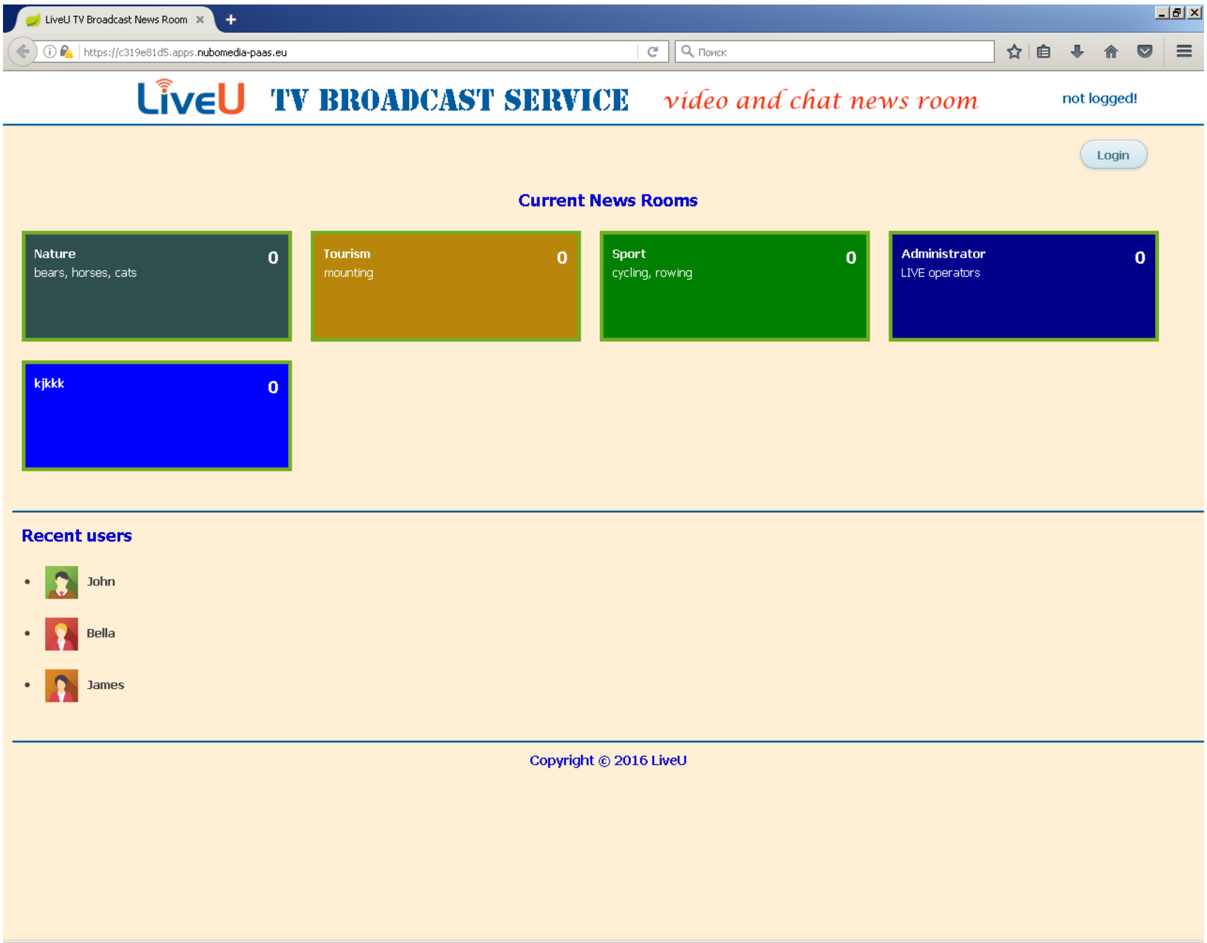


Figure 1: Scalability Test for NewsRoom

6 End-user's guide and tutorial: how to install and use

Test application can be deployed at Nubomedia PaaS. You can follow instructions at http://nubomedia.readthedocs.io/en/latest/paas/paas-gui/. GitURL must be https://github.com/LiveU-TV/tv-broadcast-jar.git. Demonstrator application screens with short description are below:

6.1 Room list



LiveU TV Broadcast News Room
https://c319e81d5.apps.nubomedia-paas.eu
Поиск

LiveU TV BROADCAST SERVICE *video and chat news room* not logged!

Login

Current News Rooms

Nature 0
bears, horses, cats

Tourism 0
mounting

Sport 0
cycling, rowing

Administrator 0
LIVE operators

kjkkk 0

Recent users

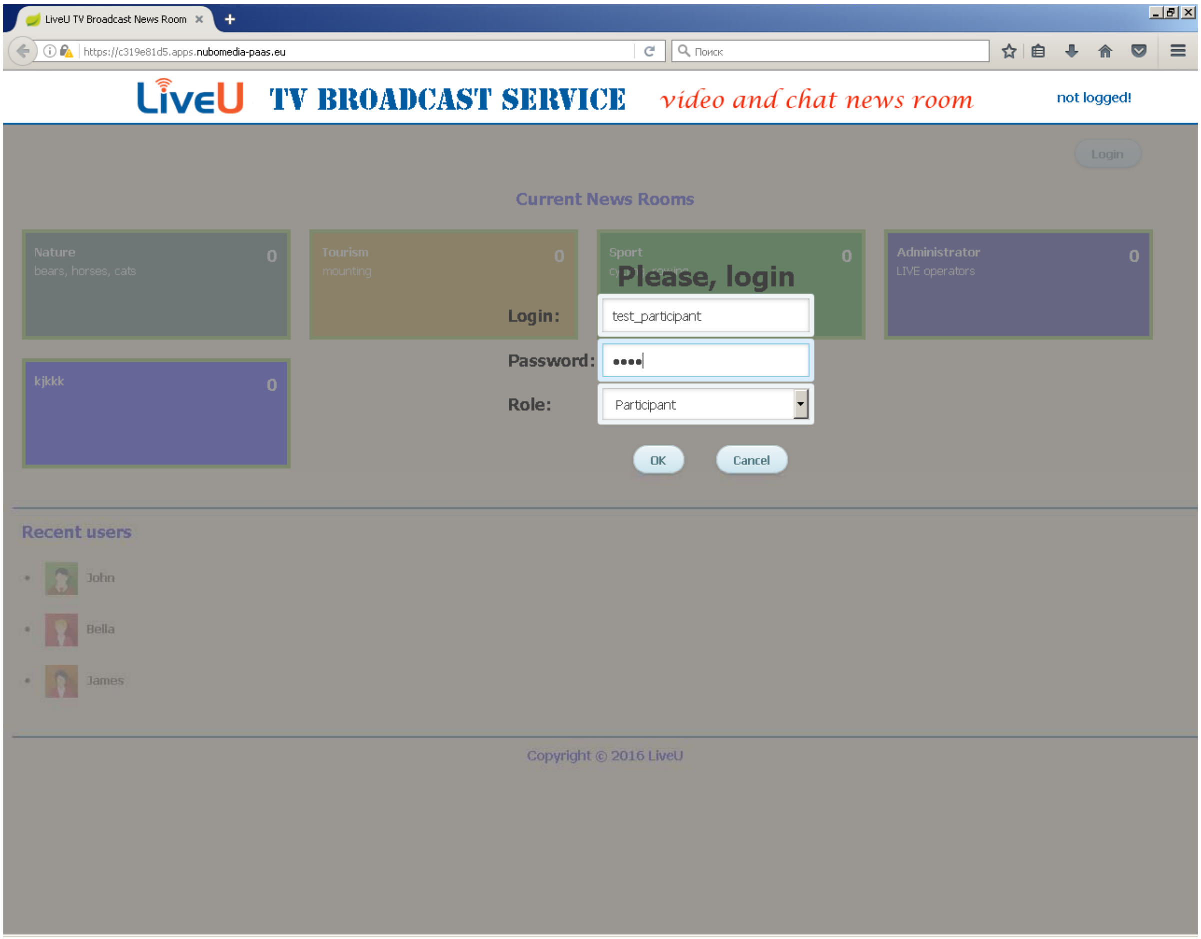
- John
- Bella
- James

Copyright © 2016 LiveU

Figure 3. List of News Rooms and number of participants in each News Room

User must logon to enter News Room.

6.2 User Login



Please, login

Login: test_participant

Password: ••••

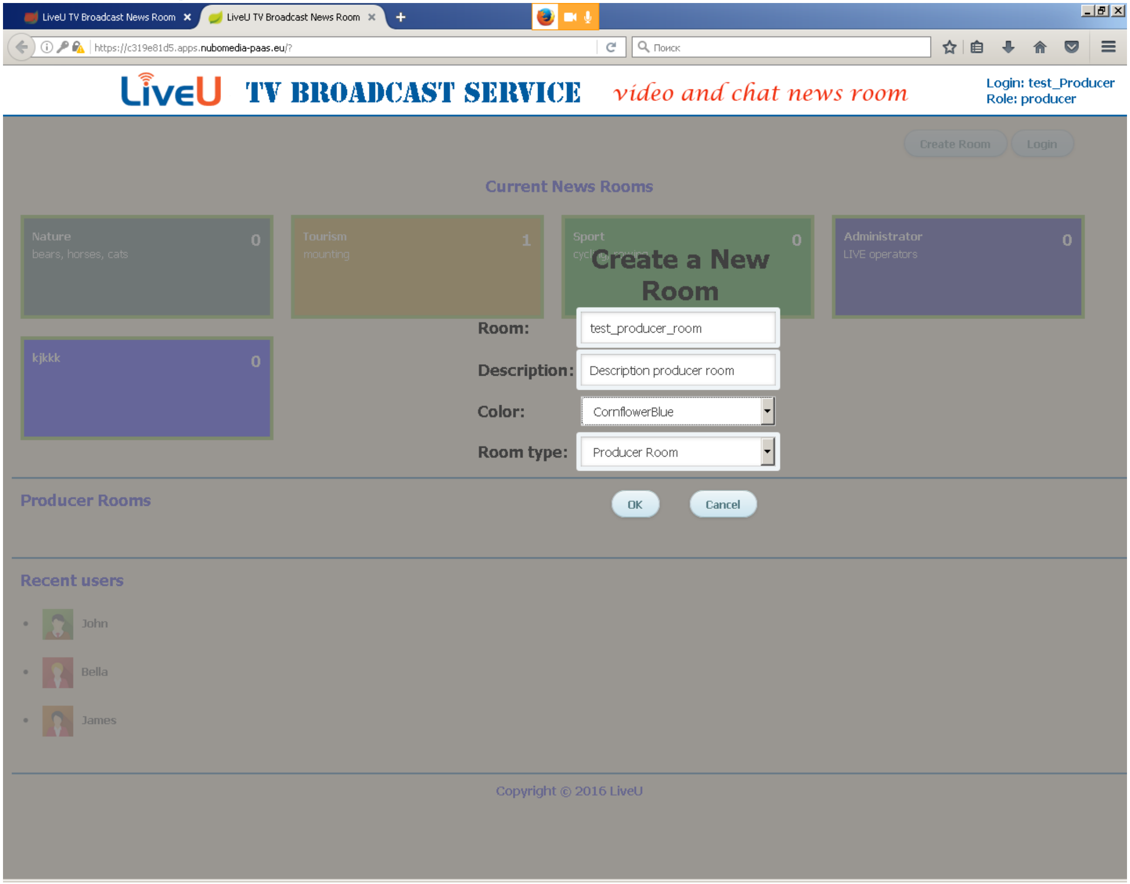
Role: Participant

Figure 4. Login form

User must enter login name (password field now is optional) and choose Role. Role must be one of:

- Participant – person who shares his video and audio and can be stream to LiveU media network;
- Producer – person who shares his audio and can move Participants to Producer Rooms and direct Participant media to LiveU media network.

6.3 Adding room



Room: test_producer_room

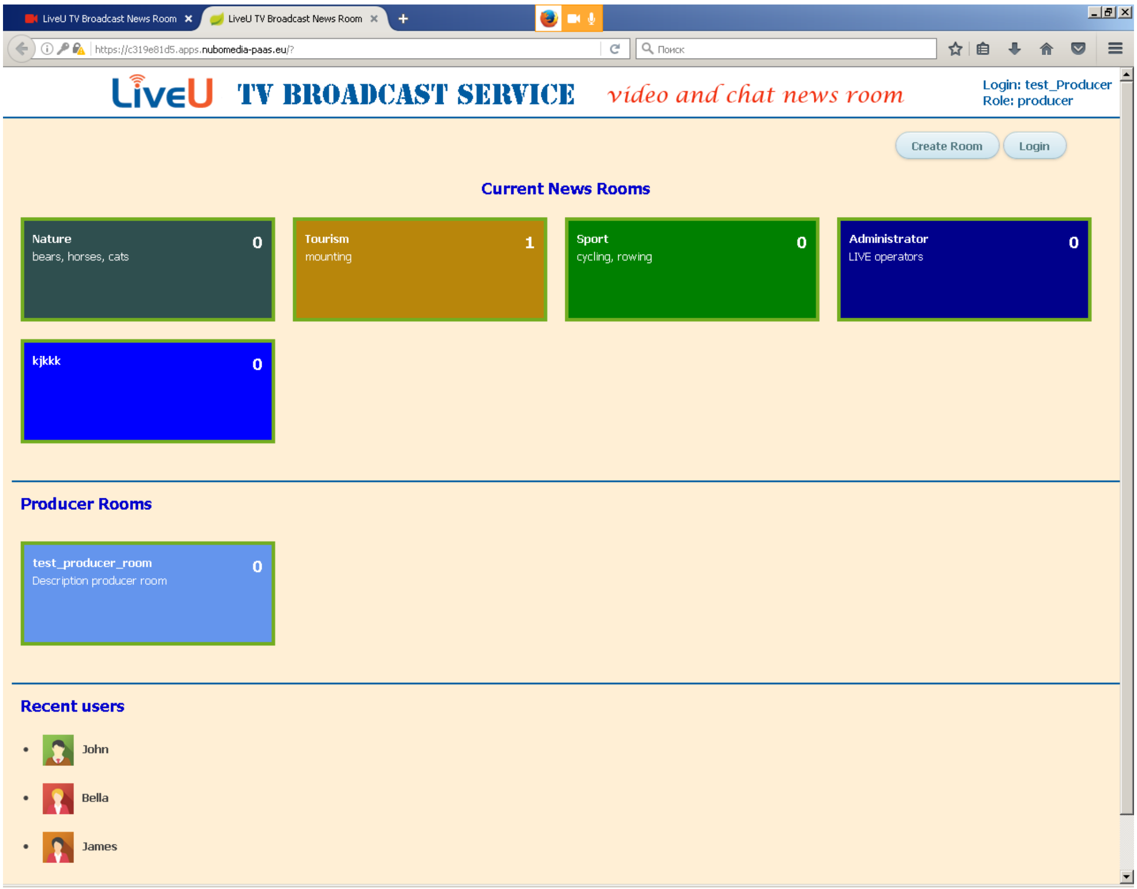
Description: Description producer room

Color: CornflowerBlue

Room type: Producer Room

Figure 5. Room creation form

After successful login Participant can create NewsRoom, Producer can create NewsRoom or private Producer Room. Producer Room is visible only to owner at Room list page.



LiveU TV Broadcast News Room ✕ | LiveU TV Broadcast News Room ✕ | +

https://c319e81d5.apps.nubomedia-paas.eu/?

Поиск

LiveU TV BROADCAST SERVICE *video and chat news room*

Login: test_Producer
Role: producer

Create Room | Login

Current News Rooms

Nature 0
bears, horses, cats

Tourism 1
mounting

Sport 0
cycling, rowing

Administrator 0
LIVE operators

kjkkk 0

Producer Rooms

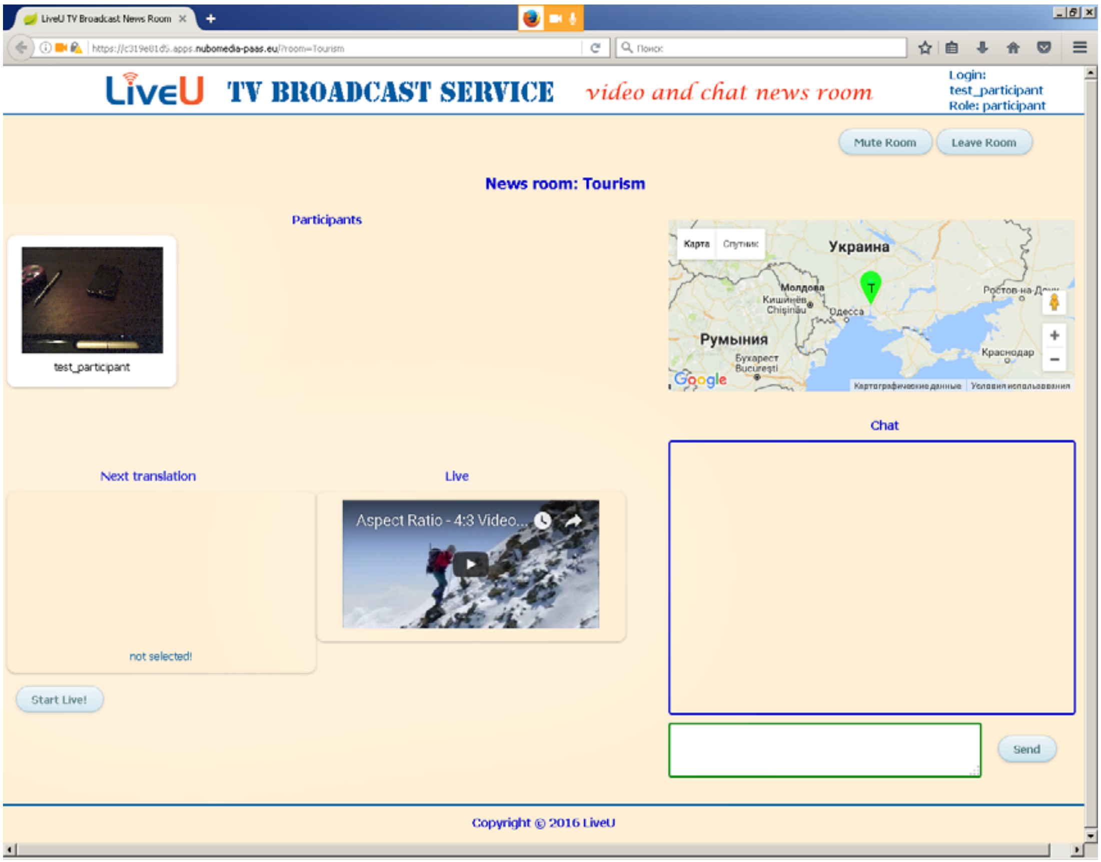
test_producer_room 0
Description producer room

Recent users

- John
- Bella
- James

Figure 6. Producer Room List after Producer Room creation

6.4 News Room



LiveU TV Broadcast News Room ✕ | +

https://c319e81d5.apps.nubomedia-paas.eu/?room=Tourism

Поиск

LiveU TV BROADCAST SERVICE *video and chat news room*

Login:
test_participant
Role: participant

Mute Room | Leave Room

News room: Tourism

Participants

test_participant

Chat

Next translation

not selected!

Live

Start Live!

Send

Copyright © 2016 LiveU

Figure 7. News Room after enter

Participant or Producer can see herself and other Participants. There is also map and chat.

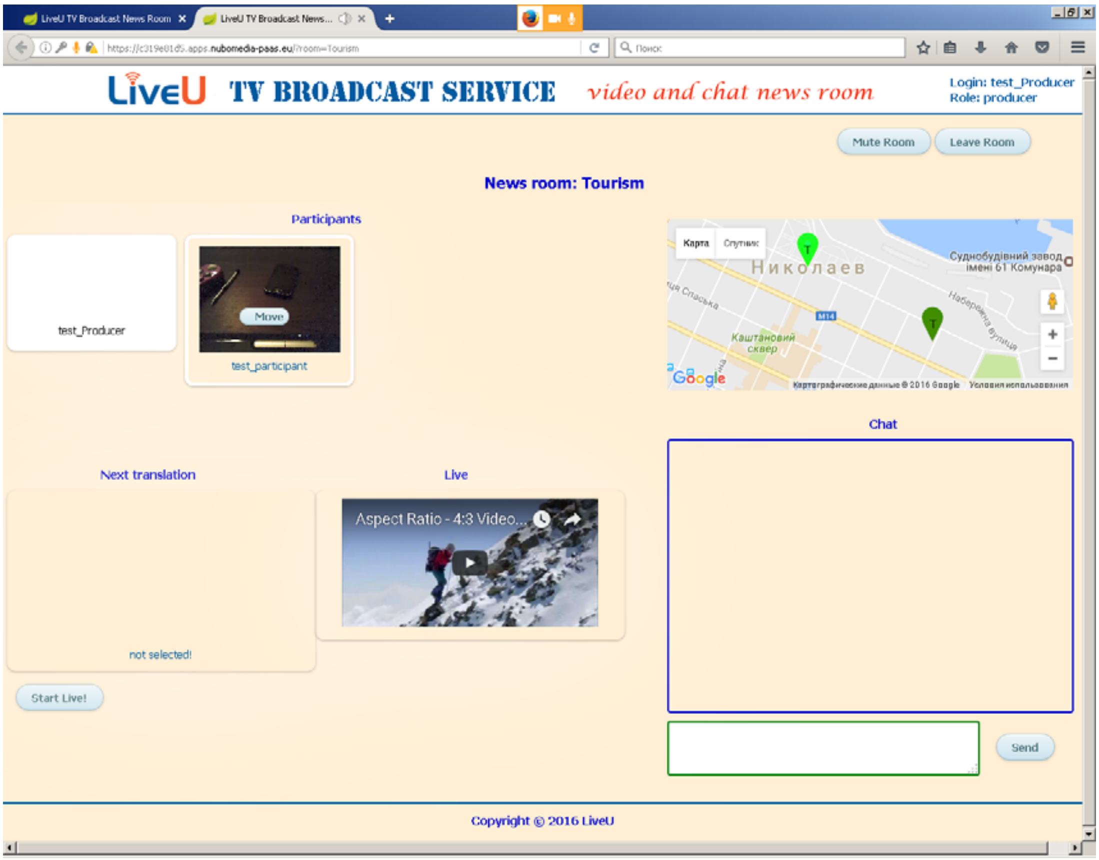
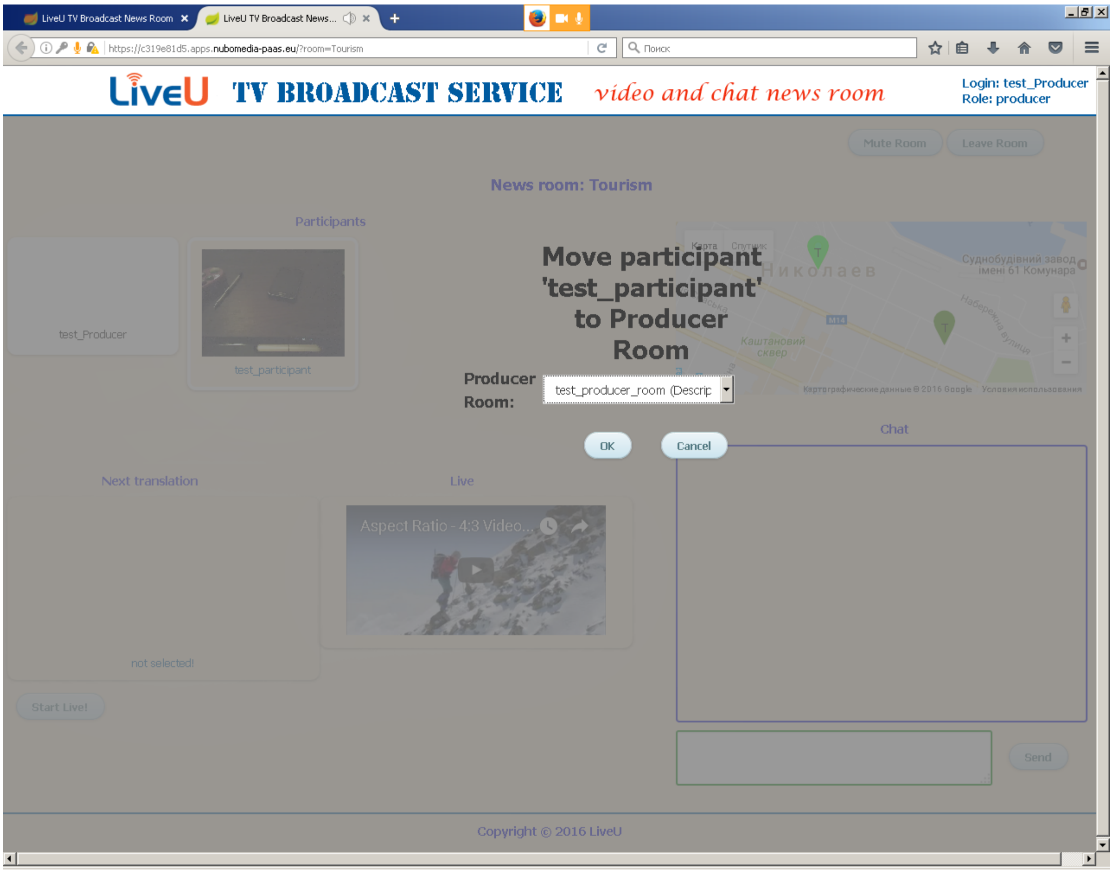


Figure 8. News Room after Producer entering

Producer can move Participant to his Private Producer Room.

6.5 Producer Room



Move participant 'test_participant' to Producer Room

Producer Room: test_producer_room (Descrip

OK Cancel

Figure 9. Producer moves Participant to Producer Room

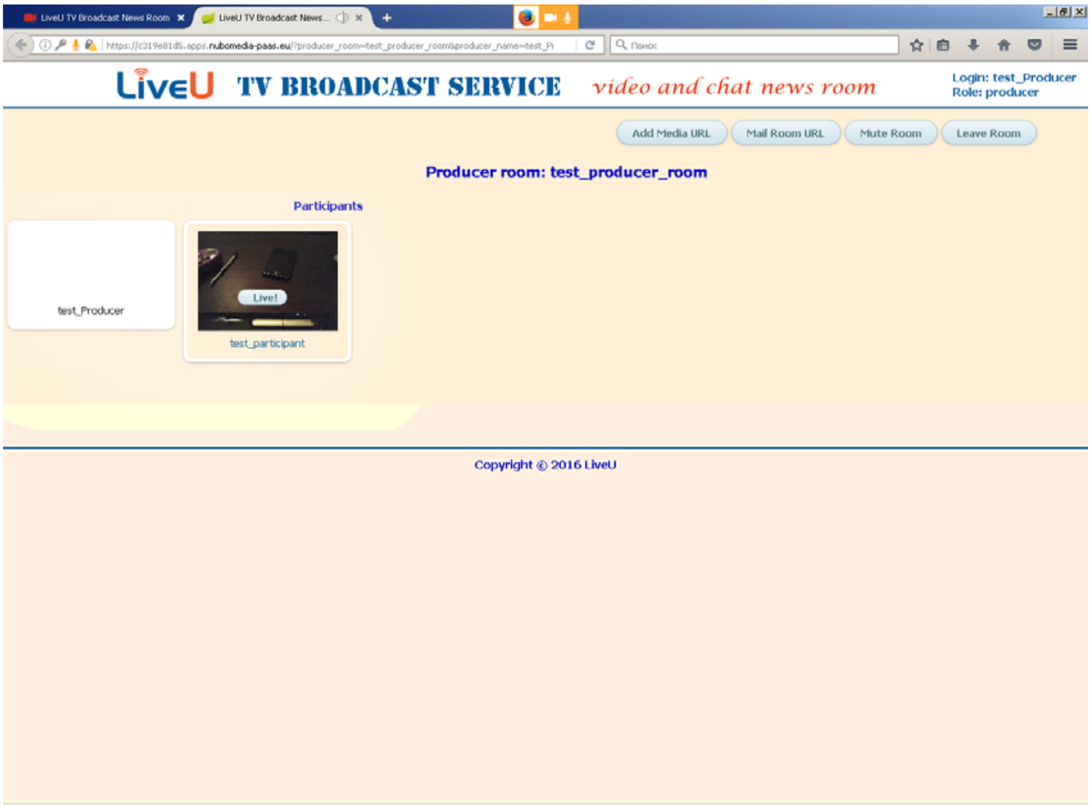
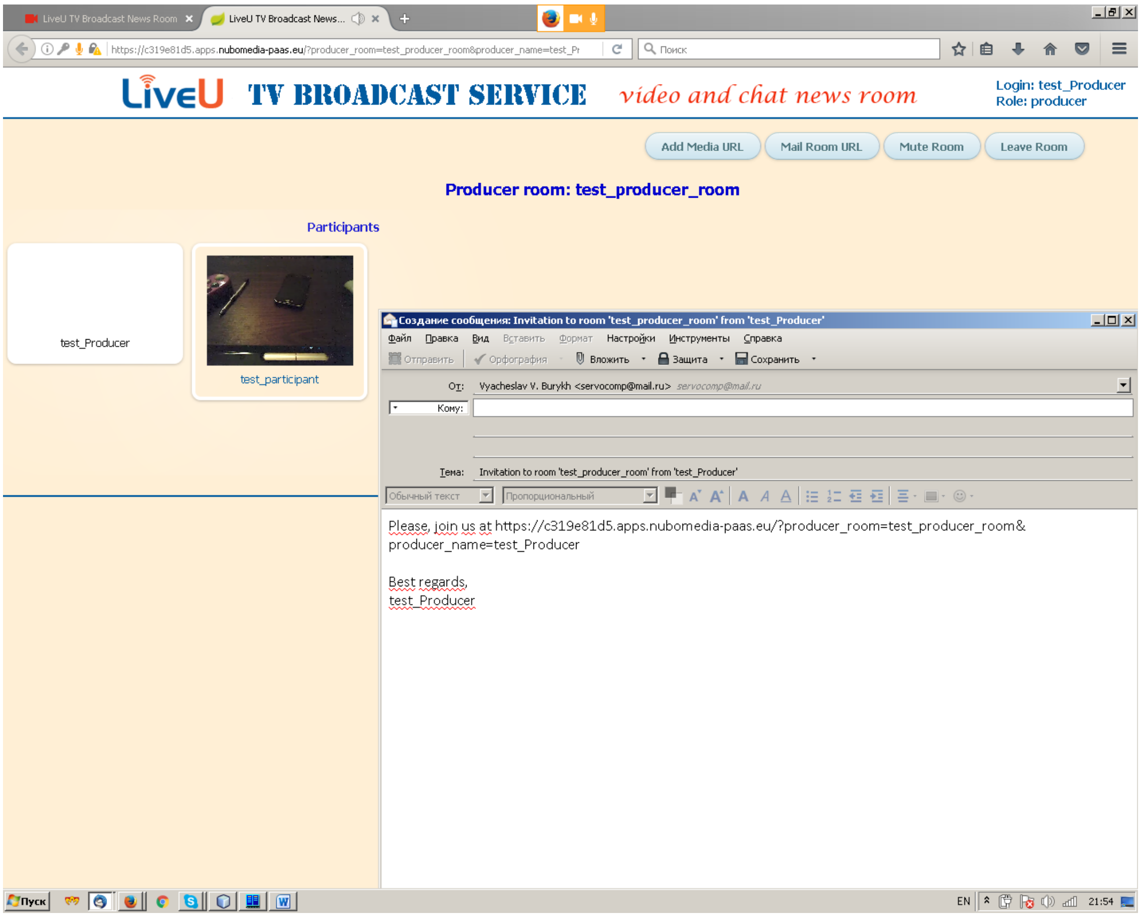


Figure 10. Producer Room after Producer enter


LiveU TV Broadcast News Room
LiveU TV Broadcast News...
https://c319e81d5.apps.nubomedia-paas.eu/?producer_room=test_producer_room&producer_name=test_Pr
Поиск
LiveU TV BROADCAST SERVICE
video and chat news room
Login: test_Producer
Role: producer
Add Media URL
Mail Room URL
Mute Room
Leave Room
Producer room: test_producer_room
Participants
test_Producer
test_participant
Файл Правка Вид Вставить Формат Настройки Инструменты Справка
Отправить Орфография Вложить Защита Сохранить
Обычный текст
Пропорциональный
Пуск
EN
21:54

Создание сообщения: Invitation to room 'test_producer_room' from 'test_Producer'

От: Vyacheslav V. Burykh <servocomp@mail.ru> servocomp@mail.ru

Кому:

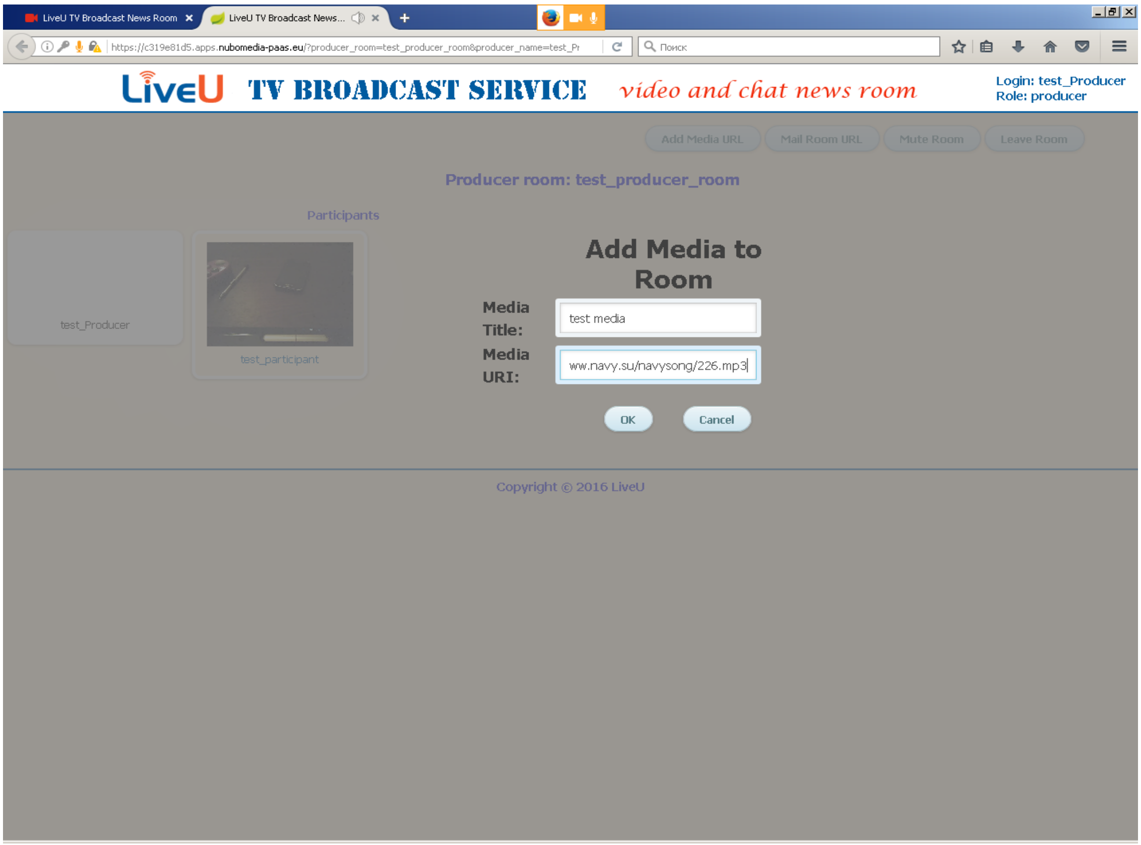
Тема: Invitation to room 'test_producer_room' from 'test_Producer'

Please, join us at https://c319e81d5.apps.nubomedia-paas.eu/?producer_room=test_producer_room&
producer_name=test_Producer

Best regards,
test_Producer

Fig11. Mail Room link

Person who receives mail message and follows link can enter Private Producer Room after successful login.



https://c319e81d5.apps.nubomedia-paas.eu/?producer_room=test_producer_room&producer_name=test_Pr

Login: test_Producer
Role: producer

Producer room: test_producer_room

Participants

test_Producer

test_participant

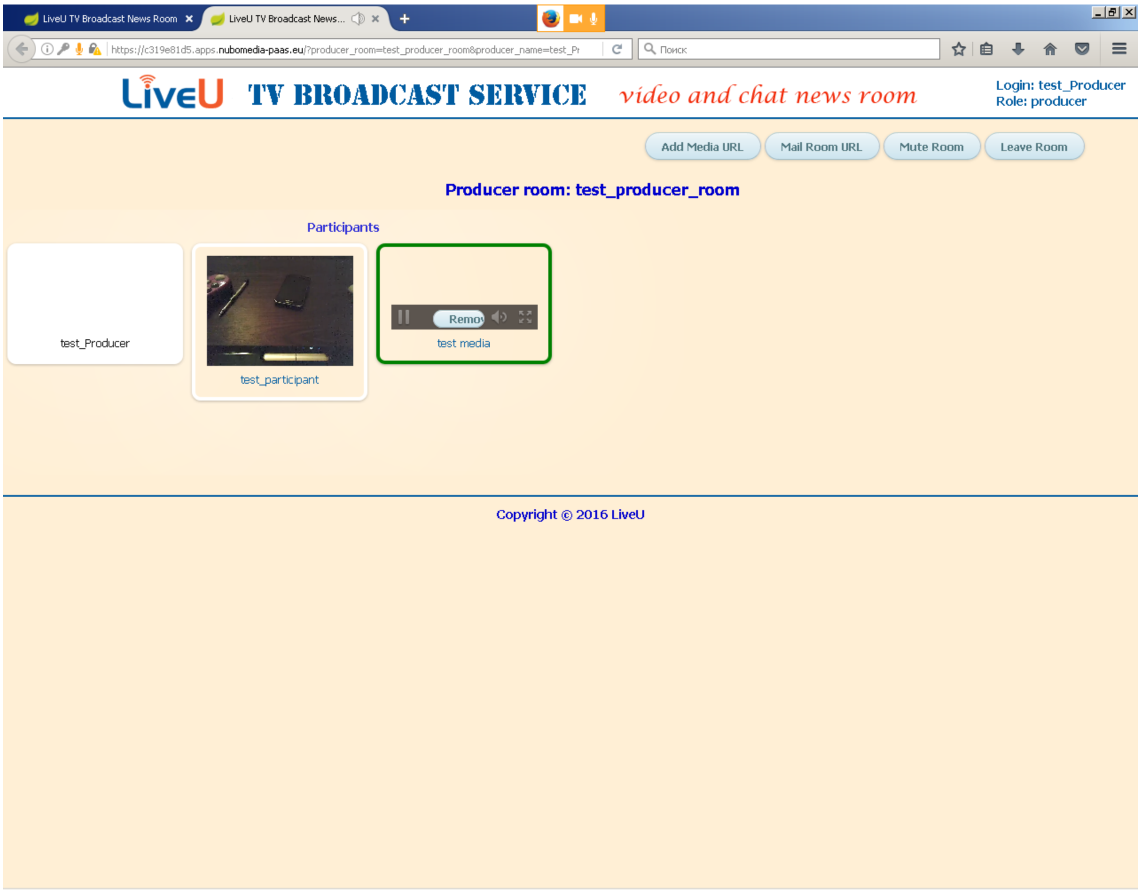
Add Media to Room

Media Title: test media

Media URI: ww.navy.su/navysong/226.mp3

Copyright © 2016 LiveU

Figure12. Alternate media can be added in Producer Room (HTML, RTSP, mp4)).



https://c319e81d5.apps.nubomedia-paas.eu/?producer_room=test_producer_room&producer_name=test_Pr

Login: test_Producer
Role: producer

Producer room: test_producer_room

Participants

test_Producer

test_participant

test media

Copyright © 2016 LiveU

Fig13. Added Media can be removed

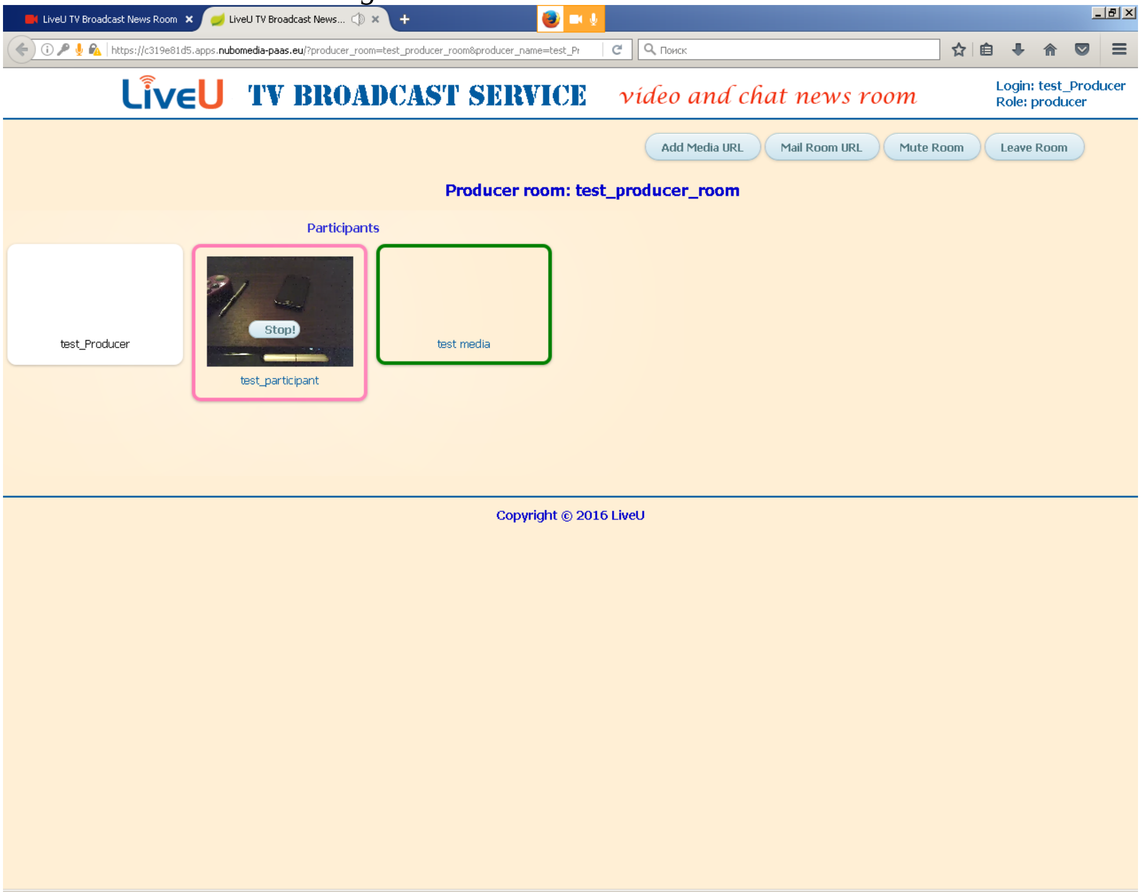


Fig14. Participant can be selected and streamed to LiveU media network

7 Total effort consumed in creating the demonstrator

LiveU real effort devoted to creating the demonstrator is about **8** Person/Month.

8 Developer's feedback in relation to using NUBOMEDIA

LiveU's developer feedback is positive when using NUBOMEDIA Media Server and its APIs. As a company with expertise in video communication and media technologies, we already knew the difficulty behind the development of reliable sessions with many audio and video streams. Because of this, when we started using NUBOMEDIA APIs, we was really surprised about the simple use of it, saving a lot of time to developers to receive acceptable results. NUBOMEDIA APIs are easy to understand, so we don't need to spend a lot of time reading documentation to make it work, just after a couple of hours reading how to use it we started developing our demonstrator. To sum up, LiveU's experience is really positive developing with NUBOMEDIA Media Server APIs in such complicated field as media networking. However, debugging wasn't always easy, and stability of the PaaS was weak during the development, nevertheless, it got better toward the end of the project and the potential is huge.