Lab 3 Multimedia Systems

Group

Marc Dahlem
Jon Martin Mikalsen

1 Problem Specification

1.1 Description

In this last lab there will be a system that builds on top of the two previous labs and will contain one multimedia server, web-camera clients and mobile phones with the Android OS. The overall idea here is that there is a motion detection server, that manages the streams of multiple web camera clients, and Android phones that can subscribe to these cameras and get updates in form of a message when there is movement on a subscribed camera.

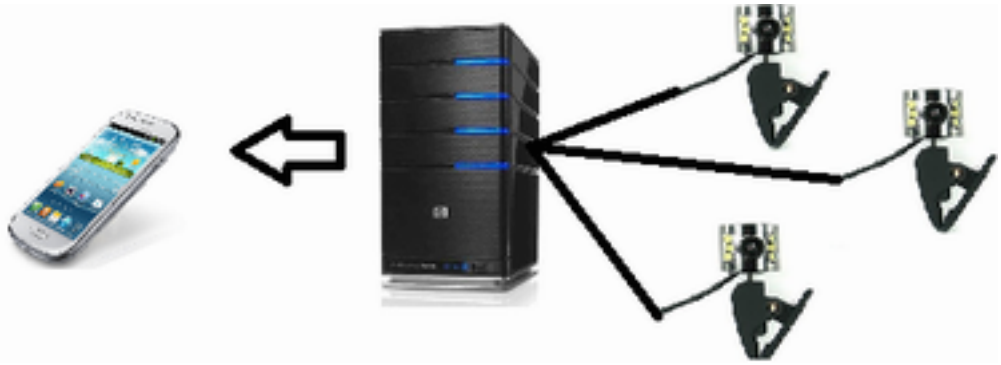


Illustration 1: Overall system overview

In the lab description it's pointed out that this lab shall focus on the use of SIP[1], that is a protocol for controlling communication sessions such as audio and video over Internet. Although it's described to use SIP, this project has focused on using GCM(Google Cloud Messaging)[2] and the reason for this is described later in this document.

1.2 Assumptions

To create a system like this several things has to be examined.

1. Android is able to play the recorded files with the embedded media player

[1] http://en.wikipedia.org/wiki/Session_Initiation_Protocol

[2] http://support.google.com/googleplay/android-developer/bin/answer.py?hl=en&answer=2663268

2 Usage and user's guide

Here the instructions on how to get the application running is stated.

2.1 Requirements

- Need Android phone or Android SDK working on computer
- Have the knowledge on how to run a given java Eclipse project/program
- Have GStreamer installed on computer
- Have the ability to run other GStreamer applications developed in Java
- Have Eclipse running and working on the computer
- Android phone has to have VLC installed to ensure 100% functionality

2.2 How to get it running

2.2.1 Runnable .jar file

To run the application one can use pre-build jar files delivered with this report.
To execute them run in the command line 'java -jar client.jar' or 'java -jar server.jar'.
The order of this two executions doesn't matter.
Hint: If one doesn't want to connect to localhost one has to open port 5000 on the server side.

2.2.2 Git repository with code

- http://github.com/MarcDahlem/ANSUR

2.2.3 Android application

To be able to use the Android application an Android SDK has to be installed on the computer. How to do this is explained in detail on Googles own pages http://developer.android.com/sdk/index.html

When the SDK is installed the application can be launched on an actual Android phone, or in the built-in emulator. How to launch application on phone is described here: http://developer.android.com/tools/device.html

To manage the emulator this link can be useful http://developer.android.com/tools/devices/managing-avds.html

2.3 Result

2.3.1 Web-camera clients

When starting up the client the screen shown in illustration 2 will appear.

Illustration 2: Client start-up page

In the client there is a possibility to change the server ip address and it's port, and to do this just click the "settings" button, and the following page will show (Illustration 3).

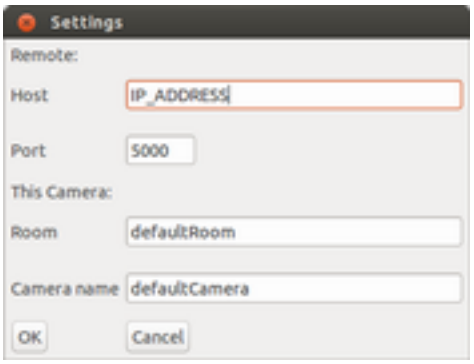


Illustration 3: Client settings

New for this lab is the possibility to define the cameras name and the room where the camera is placed. This is used in the android application to show and subscribe for whole rooms and in order to have it easier to recognize cameras. (Before was their ID the port on which their server gstreamer pipeline is running.)

2.3.2 Server

When starting the server, the application should look like in the last lab report. From the graphical layout it is only the connected client-list that is improved, since they have rooms and names now.

The most parts of the server for this part of the lab are hidden.
There were of course some small bugfixes regarding to the previous version, but the main parts are handled in the background. The most important part was the connection between the Android phone and the server, for which the ServerConnectionManager was improved and totally restructured.
It is now possible to register, unregister, subscribe, unsubscribe, getConnectedCameras and downloadMovie in addition to the already known Camera-connect command.

It has also a mechanism to notify clients for their subscribed camera events and furthermore commands to handle the connection like a server shutdown.

2.3.3 Android device

The other part of the communication is handled on the Android device, which has an own AppConnectionManager.
The android device has to display all the information accordingly and should be able to invoke and handle all the options from the connection.

It consist of the main page where one needs to connect to the server. When connection a messege will be presented on the screen whether the connection failed or succeeded.

Illustration 4:Main page

Illustration 5: Message when registering to the server

A second page shows a tree of all rooms and cameras and has options to update the subscription status for them. If something is marked orange like shown in illustraton 6, it means that that this room or this camera would raise in the current selection status a subscription update.

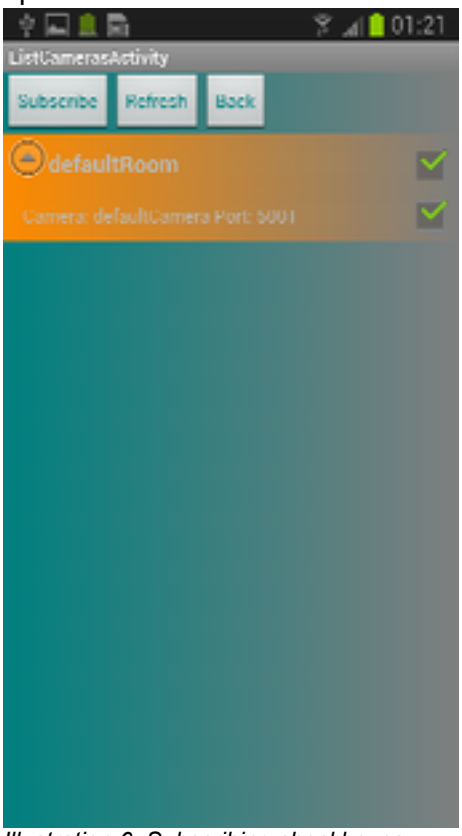


Illustration 6: Subscribing checkboxes

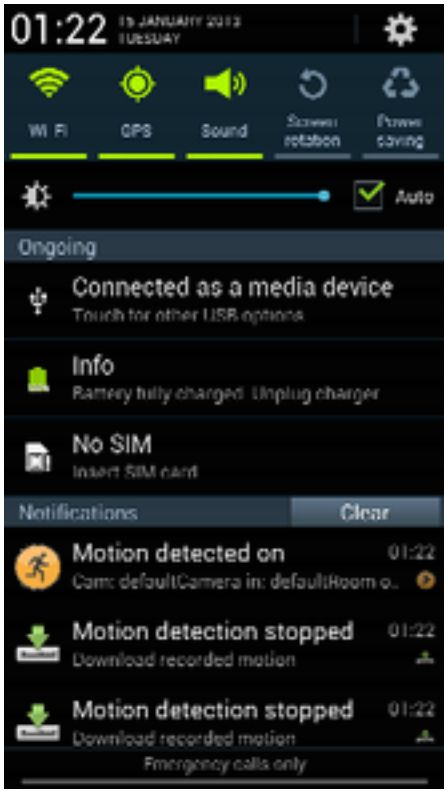


Illustration 7: Notifications

Android notifications are used to display and handle the motion events, and if a motion-event is finished one can download the file via clicking on the download link.
That will open a third page which should normally show the video after it is downloaded. Because of problems with the inbuilt android media player and the used codecs on the server side is the application at the moment not able to show these videos. But one can open them after downloading manually in media players that support the ogg video format, for example. the VLC application. The application also includes a settingsactivity which is accessible through the menu button, which lets the user set the server address and the port (*Note: This only works in the main menu*).

3 System description

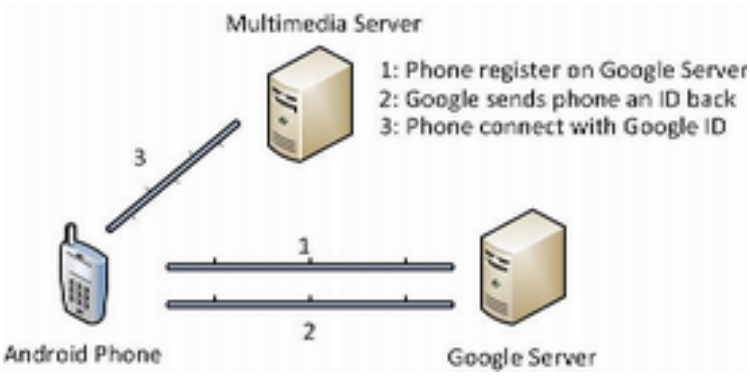
This chapter describes the design and the structure of the System in more detail.
The first part will take a deeper look into the session handling over the Google Cloud Messaging service. Chapter two will show the internal structure of the new server parts and the android application.

3.1 Session handling over Google Cloud Messaging

Google cloud messages are used in this application to establish and manage the subscription sessions. They inform the users over motions, camera updates like disconnect or rename, and server shutdowns.
This chapter will describe the registration and notification mechanism in more detail and describe some examples how they have been used.

3.1.1 Registration

When an Android device wants to subscribe to one or multiple web cameras on the server, it first (1) has to register on the Google Cloud Messages[3] server. This is done usually once in an application lifetime and needs a google project id of the application so that the google server can distinguish between different applications for one device. We decided to implement it as a manually registering since we want the user to see if he is connected or disconnected.



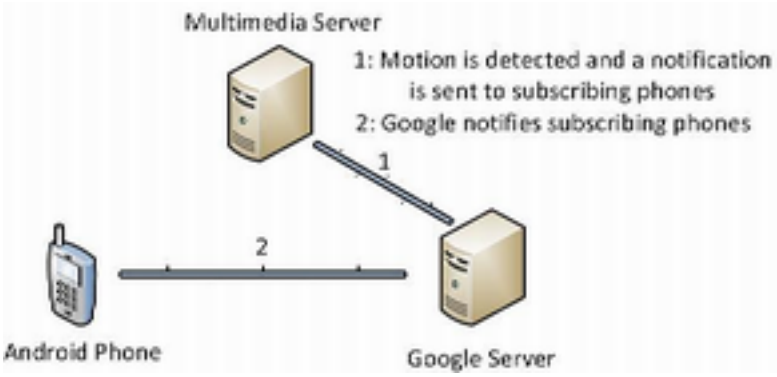
As a result of this registration will the Google Cloud Messaging Server return an GCM ID (2) which is only valid for the registered device and application.

To finish the registration a device has to send his GCM ID to the own server, which then can use this ID to send messages to the client (3).
In our case means that, that the application registers itself on the multimedia server so that it will receive server messages and that it can subscribe to cameras.

[3] http://developer.android.com/google/gcm/index.html

3.1.2 Notification

The notification from the multimedia server to all interested phones is performed through the Google Cloud Messaging Server. Since smartphones change often their ip or are sometimes for unkown reasons not connected to the internet is the Googles server used to get informed on an internet reconnect and it will store all messages for a phone for one in this message defined time. Besides of getting less errors in the connection handling it also saves energy on the phone, since they don't have to reconnect to the server all the time and since the messages are pushed to the phone (and not requested) as soon as a phone as reachable (2).



In our case is this message delivering used from the motion-detection server which sends mainly motion detection or motion detection ended messages to the google server (1). To address the phones it uses the GCM IDs of all phones which are connected to the cameras on which the motion events are detected. This server will retry 5 times to deliver the messages to the android phone as soon as it is connected. This message (2) from the google server to the phone creates on the android system an Intent with the project ID to find the right Activity on the right application.

3.1.3 Deregistration and other messages

Besides of the use of the Google Cloud Messaging service for the notifications is it also used for session updates and special server messages. Session updates means that the server sends a notification if he wants to shut down. Then can all the clients set their registeredOnServer field back and in our case can they also disconnect from the Google Messaging Server.

The usual deregistration when clicking on the button will send a deregistration command with the GCM ID to the multimedia server. The server will delete the ID from all subscriptions and from the known GCM IDs. After that will the client disconnect from the Google Cloud Messaging Server. If the multimedia-server was not connectable will he get informations about the client disconnect the next time when he tries to send a file to the client.

The Google Cloud Messages are furthermore used to inform on Camera disconnect events, so that subscribed applications can handle this deletion of the Camera and maybe also the deletion of the room.

3.2 Classes

An overview of all the classes in the project can be found in with the code that is on Github. Here is a link to the class diagrams. ***P.S. The pictures has to be downloaded.***

- **http://tinyurl.com/ClassDiagrams-Lab3**

4 Problems and reflections

4.1 Limitations

4.1.1 No live streaming with Android media player

When working out the idea for the third lab we wanted to implement a live streaming to the phone if the user presses on the notification of an motion event, and the motion is not finished with the recording. But since the inbuilt Android Media Player is not able to play video over rdp we tried out different ways how to solve it.

4.2 How to avoid limitations

4.2.1 VLC

Since there wasn't any solution that wouldn't take too much of our time, we decided to use Videolan player that already support live video/audio stream. Only thing we had to do was to download VLC from Google Play and find a way to use it in the Android application.
We figured finally out how to start it he application, but it is not possible to pass the network stream as parameter in the current vlc version. This comes from the fact, that it uses an internal callback mechanism to start files instead of using android intents like recommended. Finally we copied a the streams url into the clipboard and launched vlc without a parameter. One can paste the stream address in the usual opening dialog. But at the moment is the streaming not started on the server. We focused on more important parts as we realized that the streaming is only poorly possible. That would be one update to add in future releases.

4.3 Problems during the lab

4.3.1 Downloading motion detection file to phone

To implement the part where a motion detection movie was downloaded from the server, to the Android phone caused a lot of problems. At first there was a problem where the file that was about to be downloaded couldn't be saved in directory on the phone, because of missing write-access rules. When this was fixed there were problems with sending the entire movie. Sometimes the movie transfer stopped very fast and the movie would not work. After a lot of work and many hours of testing, it finally worked.

4.3.2 Launching VideoLan from application

Trying to implement the automatic startup of Videolan was some sort problematic at first. Mostly because it was hard to find the package and class name that the VLC player used. After a lot of research both were found and the application could be launched. But like said in limitations, it is at the moment not possible to parse parameters to it so that it cannot start a stream directly.

Package name: org.videolan.vlc.betav7neon
Class name: org.videolan.vlc.betav7neon.gui.MainActivity

4.3.3 Displaying room-list in Android

On the Android phone it is possible to get a list over all the rooms that are registered on the server, and all the cameras that are in each room. To create this list we spend many hours of research because the actual setup of the expandable list was poorly documented and not designed for custom data structures. After a lot of different tries did we finally succeed in implementing an own adapter which creates all the views for rooms and childs and handles all updates itself.

4.4 SIP versus GCM

In the planning phase for this lab, there had to be decided if it needed to contain the usage of SIP. Although the project should include SIP, it was decided to use GCM for several reasons.

GCM has a very good setup to support subscription, where a client application connects to a server-event. When using SIP on phones it has a big disadvantage. Since phones easily are disconnected, the phone needs to update on every reconnect its address again. It also has to call the server to get all updates since he was offline and so on.
The advantage of GCM is that it tells the client if there is anything new on every connection. The messages are pushed through this algorithm as soon as possible to the phone and google spend a lot of time in improving this resending by saving at the same time energy.
Especially for the eventually, but pressurable events like motion detection a customer wants to know it as soon as possible and one can gain time by the use of such improved systems.

One other thought was, that the session is very weak. It exists no strong coupling between the phone and the server, since the server just send sometimes some updates. Therefore one doesn't need a session which needs to be managed and up to date all the time. It is sufficient to know the Id of the client and then store the data on a server which sends it as soon as the client is connected.

So we decided to use the always recommended GCM, as it is also kind of a session initialization protocol, but can be used for other purposes as well.

4.5 Possible improvements

Besides of all the improvements descriped in the previous lab reports did we found a lot of other imrpovents of this framework which could be implemented in future realeses.

4.5.1 Multiple video screens on Server

At this moment the server only has one big screen where it shows the video-stream. Per now it is changing based on what camera the movement is detected on, but a better thing would to have multiple small windows of all connected streams, and that could be clicked on to be enlarged.

4.5.2 Better handling of live-stream on Android

At the moment the live-stream is just a theory with have complete implementation. The things that are missing is to get the server to support tcp or udp live streaming on ip and port. Now when there is a notification about a new movement on a camera, it is possible to click on that notification and VLC player would launch without showing stream. A good thing would be to implement so that a live stream is showed at once when clicking on notification.
Also could one implement a new activity as a detail page of every camera, where one could connect to its lifestream.

4.5.3 Better handling of file-download on Android

When it comes to downloading the motion detection movies from the server, this is done without any question to the user if he really want to download it. Now it is just to press on the notification about Motion stopped, and the movie will be downloaded to the phone. This could be greatly improved with dialogboxes and prompting the user about the download.

4.5.4 Better motion detection video playback on Android

At the moment when a movie is downloaded from the server, and the downloading page appears, the “play movie” button has a hardcoded path to the given file, and this is not optimal and should be fixed in later updates.

4.5.5 Security

The created framework is just to see as a prototypical implementation. It has no security mechanisms implied. When knowing the command and a filename a user could for example download at the moment every file from the servers filesystem, or motions to cameras he isn’t subscribed to. There can be added a lot of improvements like for example a handling all recorded movies in a media database.

4.5.6 User Management and Privacy

Privacy in data, media and the internet is a big topic. In the current state could the application not be published with a public server since everyone could access every camera registered on the server. To avoid this one should think about a user management system with accessrights for different cameras, rooms etc.

4.5.7 Bugs

Since this is a quite big project it is hard to remove all bugs that are in the system. With more testing-time it would be possible to remove more of the potentially bugs that might be there.

5 Contribution

In this lab Jon Martin and Marc has worked closely together when creating the different aspects of the overall system. Both have been sitting at their own places, but communicating over Skype and Facebook and on school when both had the time to meet. In this lab both have worked on different things across the project, but Marc focused most on the Server part and GCM communication, while Jon Martin worked with Android. Both have worked closely together and communication so that most of the work would be valid.

We want to thank with a big thumbs up Google for all the good documentation they have for the Android platform[4].

This project was that big for two people which were totally new in all this technologies.
So we had to set boundaries for what should be in and what not. Since it was more about learning new technologies than building a stable program it has to be clear that the application is not in a publishable state. But it does a good job in the final state now and we are very proud of it.

All in all came up a very big framework on its creation we learned a lot, not only about media handling, but also about Android, Java, GCM, SIP, Gstreamer and of course about a good teamwork

Lets detect the motions!
Jon Martin and Marc

[4] http://developer.android.com/index.html