# Bluetooth Communication

## Classes

Multiplayer version of the game uses Bluetooth for the communication between devices. Following graphs show the class diagrams for the package de.tum.multiplayer and de.tum.multiplayer.bluetooth, which are responsible for the multiplayer version of the game.

### MultiplayerActivity

After user selects the Multiplayer version of the game from the WelcomeActivity screen, MultiplayerActivity is created. Upon creation of the activity, board and dice object are drawn to allow the upcoming dialog activities use it as background view for better user experience. Other related classes and threads communicate with MultiplayerActivity via onActivityResult() and the Handler object, respectively. Under the titles named “Communication with the Main Activity” and “Actions According to the Activity Results”, you can find the details of this communication written on the code snippet.

### ModeSelectionActivity

Right after the MultiplayerActivity is created, another activity named ModeSelectionActivity comes as a dialog screen. This activity allows the user to select the device mode, and make the current device discoverable for other devices to allow pairing if they haven’t been paired before. One device might be server or client.

### ClientNumberPicker

If user selected to make his device as server device at the ModeSelectionActivity, next activity will be ClientNumberPicker activity. This allows the server device user to select how many clients will connect it to play the multiplayer mode. Game application supports 7 more devices to connect to server device. However, only 3 of them can play the game while the rest can follow/watch the game on their devices’ screen.

|  |
| --- |
| **Note:** During our tests with different devices and different Android OS versions, we have seen that only the devices whose Android version is 2.3.5 or above can accommodate up to 7 devices as client, while earlier versions of Android only able to accept 2 client devices. Therefore, it’s suggested to make the device whose Android version is 2.3.5 or above be as client if the users are planning to play with more than 3 devices total. |

### DeviceListActivity

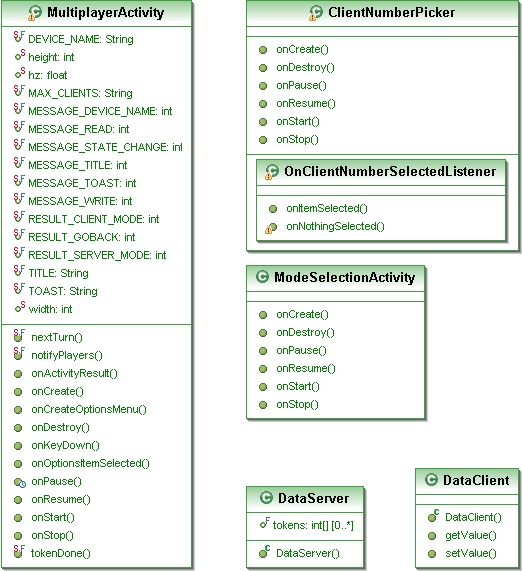
If user selected to make his device as client device at the ModeSelectionActivity, next activity will be DeviceListActivity displaying the already paired devices and allows scan more devices which are discoverable. Therefore, user won’t need to switch from game to his device’s Bluetooth settings to pair his device with others. Upon selection of device name form the list. MAC address will be passed to already initialized BluetoothMPService, and client’s connection attempt to server device will start.

### BluetoothMPService

This is the class where all the connection is handled via Threads. You can find the detail explanation at the following section with its sequence diagrams.

### DataServer / DataClient

These classes’ objcets are represents the data sending to other devices. It has fields to accommodate game’s state.



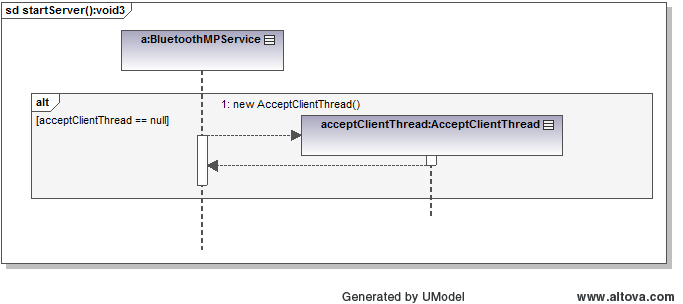
Class Diagram for the package de.tum.multiplayer



Class Diagram for the package de.tum.multiplayer.bluetooth

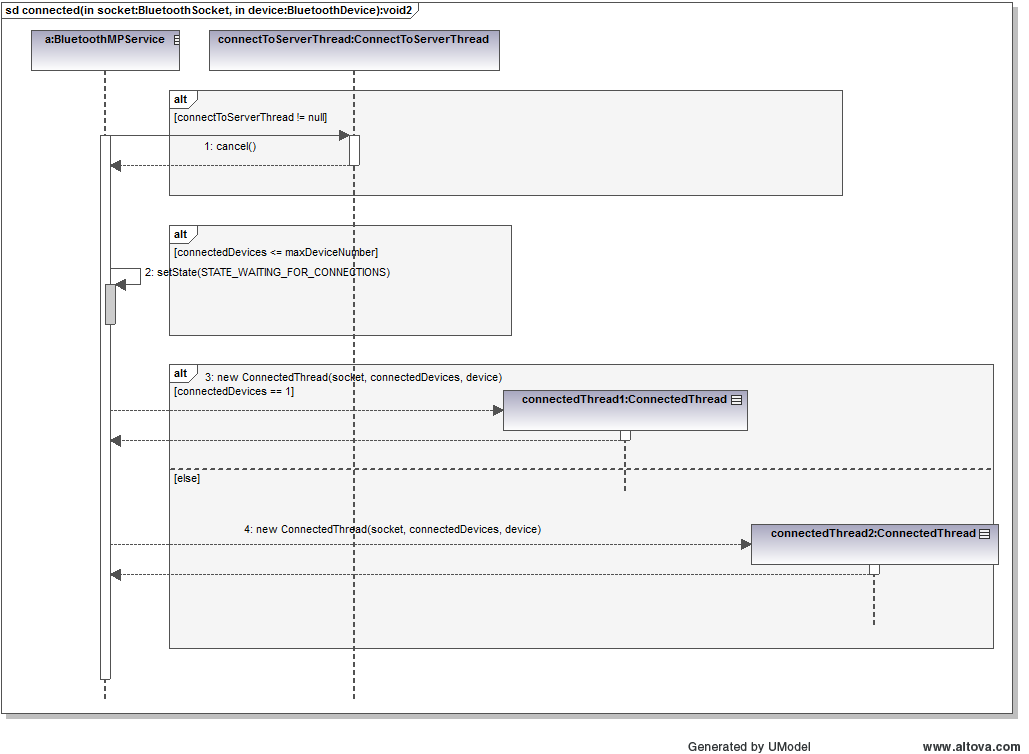
## Sequence Diagrams for the Bluetooth Communication

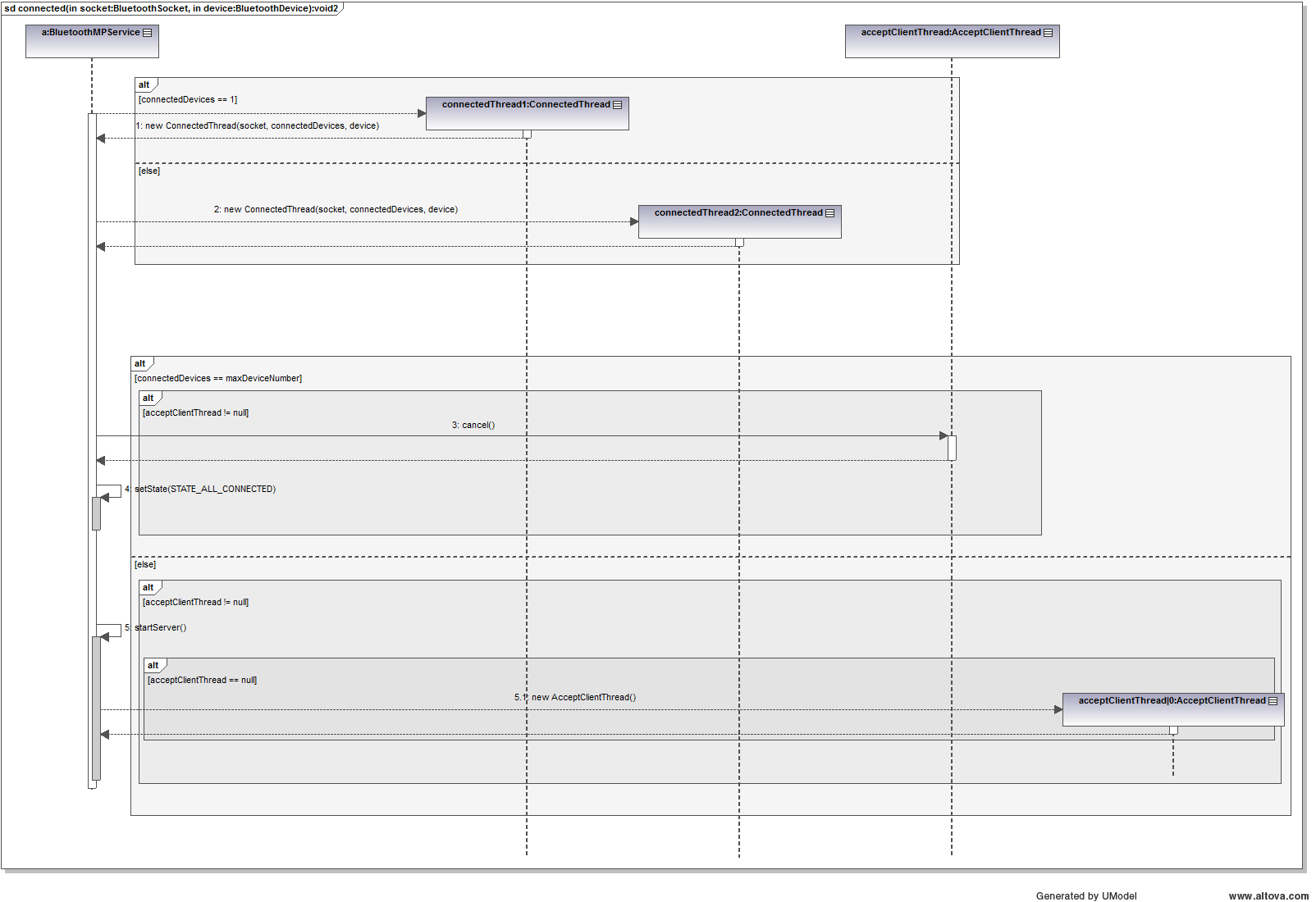
### Server Side

After user, who is using the server device, decides how many clients devices will be available for the game, startServer() method is called from the MultiplayerActivity class. Method creates a thread for listening upcoming client connections to the server device. 

Creating a thread for listening upcoming client connections to the server device.

After a client requestes a connection to the server device, who was listening for upcoming connections, another thread (ConnectedThread) is created for dealing with the communication between each other. If the maximum number for connected devices is not reached, thread for accepting new connections (AcceptClientThread) is recreated. Following sequence diagram is the simplified version for two client devices: After the BluetoothMPServer is started, it sets the state to STATE\_WAITING\_FOR\_CONNECTIONS after the first device is connected.





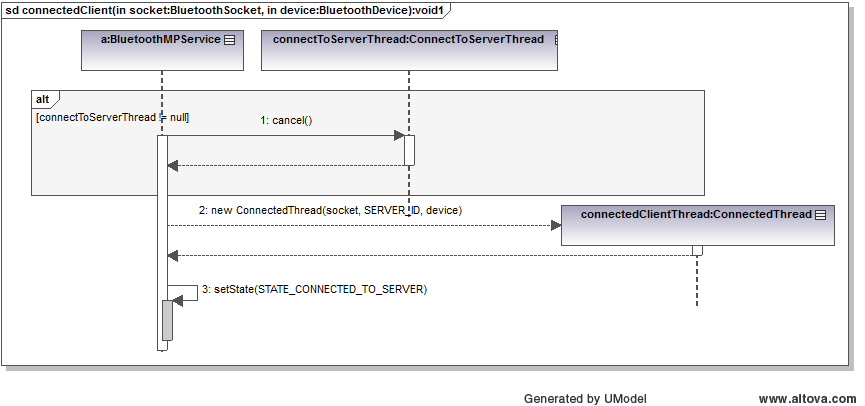
Following table shows the states while communicating with the main activity which is MultiplayerActivity. Messages are sent via Handler initiated in the MultiplayerActivity, and handled to the BluetoothMPService when the Bluetooth service is created.

|  |  |
| --- | --- |
| **STATE\_NONE** | When BluetoothMPService class is initiated |
| **STATE\_LISTEN** | When server is listening for incoming client connections |
| **STATE\_WAITING\_FOR\_CONNECTIONS** | When one or more clients are already connected to server, but waiting for more devices |
| **STATE\_ALL\_CONNECTED** | When all the devices are connected to server, and ready to start the game. |

## Client Side

When the user of the device selects the client mode, the next activity will be the screen where the paired devices list and unpaired devices list appear. User can also select to scan for more devices by clicking the button named “scan for devices”. If both devices are not paired before, the server machine user can select the option “Make This Device Discoverable” before switching to server mode. Hence server machine will be discoverable for 300 seconds, afterwards client device user can pick the server name from the list to be paired and complete the device selection activity (DeviceListActivity).

Following sequence graph reflects the activities done after the user decides his machine to be client. Selected server device’s MAC address is passed to the thread which is establishing connection by probing the socket with corresponding UUID. If one of the UUID is failed due to already reserved for another device on the server side, clients continue with probing the next UUID on the Socket. After the connection is established, another thread is created (connectedClientThread:ConnectedThread) to start the data transfer between the server and itself.



Following table shows the states to communicating with the main activity which is MultiplayerActivity. Messages are sent via Handler initiated in the MultiplayerActivity, and handled to the BluetoothMPService when the Bluetooth service is created.

|  |  |
| --- | --- |
| **STATE\_NONE** | When BluetoothMPService class is initiated |
| **STATE\_CONNECTING\_TO\_SERVER** | When probing the UUID on the opened Socket |
| **STATE\_CONNECTED\_TO\_SERVER** | When Socket is accepted and connected to the Server device |

## Communication with the Main Activity

Following table describes the messages arriving from BluetoothMPService, and the actions according to message.

|  |
| --- |
| **private** **final** Handler mHandler = **new** Handler() {  @Override  **public** **void** handleMessage(Message msg) {  **switch** (msg.what) {  **case** *MESSAGE\_STATE\_CHANGE*:  Connection state is changed  **case** *STATE\_ALL\_CONNECTED*:    All the devices are connected to the server, so start the game, and set the title to “All connected”  **break**;  **case** *STATE\_WAITING\_FOR\_CONNECTIONS*:    Toast the connected device number (Server side)  **break**;  **case** *STATE\_CONNECTED\_TO\_SERVER*:  **break**;  **case** *STATE\_CONNECTING\_TO\_SERVER*:  Set the title on the client side to "Connecting to server..."  **break**;  **case** *STATE\_LISTEN*:  **break**;  **case** *STATE\_NONE*:  **break**;  }  **break**;  **case** *MESSAGE\_WRITE*:  The message that this device has been sent to other(s)  **break**;  **case** *MESSAGE\_READ*:  Message has arrived to this device from other(s). Convert it to do the corresponding object depending on if this is client or server device  **break**;  **case** *MESSAGE\_DEVICE\_NAME*:  **Server Side:** A client connected to the server. Change the value of the progress bar on the server side according to total number of connected devices.  **Client Side:** Toast the name of the server device/  **break**;  **case** *MESSAGE\_TOAST*:  Game state has been changed, and letting the player know about the command that he has to do.  **break**;  **case** *MESSAGE\_TOAST\_WARNING*:    Warn the user about the connection lost or connection failed situation, and bringing options menu to the screen to let the user continue to the game as client or server again.  **break**;  **case** *MESSAGE\_TITLE*:    Change the text on the title bar with message arrived from BluetoohMPService  **break**;  }  }  }; |

## Actions According to the Activity Results

Following table describes the actions after the results come back from the Activities started in the MultiplayerActivity.

|  |
| --- |
| **public** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  **if** (*D*)  Log.*d*(*TAG*, "onActivityResult " + resultCode);  **switch** (requestCode) {  **case** *REQUEST\_CONNECT\_SERVER*:    Result comes from ModeSelectionActivity. MAC address of the server selected device will be used to probe the connection in the BluetoothMPServer’s connectServer() method.  }  **break**;  **case** *REQUEST\_ENABLE\_BT*:    If Bluetooth is not enabled on the device, application will enable the Bluetooth. If result is not positive, then user will be prompt to continue with single player mode.  **case** *REQUEST\_MODE\_TYPE*:  **if** (resultCode == *RESULT\_CLIENT\_MODE*) {  User selected to be client on his device at the ModeSelectionActivity. Now, another activity named DeviceListActivity will let the user to select the name of the server device to connect.  } **else** **if** (resultCode == *RESULT\_SERVER\_MODE*){  **if** (clientNumberPickerIntent == **null**) {  User selected to be server on his device at the ModeSelectionActivity. Now, another activity named ModeSelectionActivity will let the user to select the number of the client devices to play with.  } **else** {  Server device’s user selected how many clients will be available. Now, this information will be set in the BluetoothMPService as well, and it’s server process will be started. Then a progress bar will be created to show the states of the connections to the server device,/  } **else** **if** (resultCode == *RESULT\_CANCELED*) {  } **else** **if** (resultCode == *RESULT\_GOBACK*) {    Result when an Activity is ended with pressing back button before it does its job. So, the MultiplayerActivity will be ended as well.  }  }  } |