Network Systems integration project - Scuba chat

2/4/2019

1 Integration project assignment

Design and implement a fully distributed multi-hop ad-hoc chat application using wireless sound communication between at least 4 devices.

1.1 Details

The final integration project of the module Network Systems is performed in groups of 4 students. With this group of 4 students, you design and implement software that runs on each of your four laptops, and provides the networking and application functionality for a chat application. The chat application should be distributed and ad-hoc, i.e., it should not rely on a server to manage connections and routing data packets between clients. The underlying physical layer that you have to use is one based on wireless sound communications. In some situations, e.g., under water, propagation of (electromagnetic) radio waves is extremely poor. In some of these situations, wireless communications using sound may be possible, be it at rather low data rates. Your chat application should use wireless communications using sound, utilising a physical layer provided by us (hence the name mermaid chat).

Next to the design and implementation of the chat application, you should write a report describing the design of your system and its testing.

Please consider the following issues while designing your solution:

- You need to design and implement a multi-hop ad-hoc network between the 4 devices (laptops) of your group using the sound-based physical layer that we provide.
- You can download a framework providing you access to the physical layer that we provide from Canvas. Please note that your system should run on a single sound channel. Multiple channels are defined, so that different groups can run their systems without interfering each other. Please follow the instructions given w.r.t. which channel to use for your group.
- You will have to design a mechanism for medium access control.
- You will have to design a mechanism for addressing.
- As the sound-based physical layer is quite error-prone, you will have to design an protocol for reliable data transfer.

- As the range of the sound-based physical layer is rather limited, you will have to come up with some forwarding and routing mechanism to also exchange packets between nodes which can reach each other only via one or more intermediate nodes.
- Make sure your routing/forwarding algorithm is robust, so that even when the availability of wireless links changes, connectivity is maintained and packets cannot run around endlessly.
- You need to ensure the order of sent messages at the receiving device.
- the user of the chat application should be able to see which other users / devices are present and reachable.
- You need to decide what type of messages you support in your chat application. Think of short text messages, audio messages, pictures, etc. However, please take into account the the available data rate from the sound-based physical layer is very low, typically around a few hundreds of bit/s.
- Think how you can make the chat secure. You may consider for instance encryption.
- Think about the user interface.

1.2 Hints for getting started

You have to design and implement the system above using a sound-based physical layer that we provide. There are two variants of this physical layer. The first one will use your laptop's loudspeaker and microphone to wirelessly send and receive information using sound. Because sitting in a classroom with many groups using sound transmissions can be quite annoying and because the sound transfer is quite sensitive to disturbances, we will also provide an emulated environment to develop and test in. This latter variant is using a server we provide, and which you communicate to using the standard networking facilities of your laptop.

Your program will have to connect to either the audio software or emulation server. This will be handled for you by a small framework you can download from Canvas (in either Java or C++). Both frameworks provide you with a queue in which received messages will appear and a queue for data frames you want to transmit. There are 2 messages you can send and 6 you can receive. These messages are specified below.

Sending:
DATA: With this message you can send a frame of 16 bytes of data. The framework will handle sending this to the audio interface or emulation server, you just have to provide the bytes. If the number of bytes provided is less than the frame length (16 bytes), the frame will be padded with random bytes. If the number of bytes provided is more than the frame length, the excess bytes will be discarded.
DATA_SHORT: With this message you can send a short frame of 2 bytes of data. If the number of bytes provided is less than 2 bytes, the frame will be padded with random bytes. If the number of bytes provided is more than 2 bytes, the excess bytes will be discarded.

Receiving:
BUSY: You will receive this when the channel becomes busy.
FREE: You will receive this when the channel becomes free.
DATA: You received a data frame, this message will contain data as bytes (specifics depending on your programming language).
note: a data frame will be received by all nodes within the transmission range and listening to the channel of a transmitter, only if no other node within the interference range does a transmission on the same channel overlapping in time.
DATA_SHORT: You received a short data frame, this message will contain data as bytes (specifics depending on your programming language).
SENDING: A frame has started being transmitted. You can send multiple frames during transmission, which will be queued. You will receive this message when each of them starts transmitting.
DONE_SENDING: You will receive this when all queued frames have been transmitted.

Example:
We have node A and node B, they are in range of each other. A starts sending a frame (DATA), it will receive SENDING and BUSY. When the data has been transmitted it will receive DONE_SENDING and FREE. During the same time node B will have seen BUSY around the same time node A received this. B will receive DATA and FREE once A's transmission is complete.

When using the emulator, you can use the webpage `http://netsys2.ewi.utwente.nl/integrationproject/` to view and manipulate the (emulated) positions of your nodes and see if they are transmitting. The help page `http://netsys2.ewi.utwente.nl/integrationproject/help/` explains the page.

2 What to deliver?

Each group needs to deliver the following for the integration project of the module Network Systems:

- submit a planning

- give a demo of your designed system
- submit the source code of your system
- submit a final report on your project

Before being able to submit your planning and reserve a time-slot for the demo, you will have to register as a group. Please do so on Canvas (People –> Groups), choosing a free group number, and each student registering for that group number.

2.1 Planning

Before you start with implementation, each group needs to make a planning document. The planning should include (i) a global overview of the system to be designed, (ii) tasks to be performed, (iii) responsibilities of each group member for each task, and (iv) timeline. This document has to be submitted on Canvas, and will be checked and discussed with you by the TA's. After your planning has been signed off by a TA, you can reserve a time-slot for the final demo with them.

2.2 Demo

During the final demo, you demonstrate the features of your chat application. It is expected that you will be able to at least demo the following features of the application:

- presence / reachability information
- addressing messages to a single device
- reliable text messaging
- ad-hoc multihop routing and forwarding
- a simple user interface

Additional features such as security, group chat, an advanced graphical user interface are appreciated, and will give additional points.

Please note that in order to avoid unnecessary waiting times for all students, we have a very strict organisation of the final demo's. Each group gets assigned a 15-minute time-slot for their demo. Make sure you have your system to demo up and running in the demo room, at the beginning of your time-slot. If for any reason, your demo during the reserved time-slot fails, there will be an opportunity for a resit / retry on a later date. Time-slots for the demo are assigned, after your group has its planning signed off, on a first-come first serve basis. So, if you have your planning signed off early, you have more choice of time-slots.

In order to get early feedback, each group also has to give an initial demo to one of the TA's, a few days before the final demo.

2.3 Source code

The source code of your project has to be submitted using Canvas.

2.4 Final report

The final report should be submitted via Canvas. Please hand in your report as PDF and name it according to the following convention: NS-PROJECT-REPORT-GROUP##.pdf (For example: NS-PROJECT-REPORT-GROUP20.pdf) Do not forget to mention names and student numbers of the team members and the group number in the document itself.

A template of the final report can be found on Canvas under NS final project. It has the following sections:

- System architecture and diagram
- Description of designed (and implemented) system functionalities as well as methods/algorithms designed/used.
- Description of test setup, test scenarios, and performance metrics for individual system functionality and overall system
- Test results

The final report should NOT exceed 4 A4 pages. Extra pages will NOT be evaluated.

3 Evaluation criteria

The result of your integration project will be evaluated based on the demo, the final report, and where necessary, the submitted source code. Please note that each of these 3 components needs to be present in order to give a grade. In particular, a working demo with the mandatory components mentioned above is essential for getting a grade. Both the demo and the final report will count for 50% of the grade of the integration project.

For the demo, the following elements will be evaluated (based on the demo, code, and questions to be asked during the demo):

Basic features(if well-designed and implemented sufficient for a passing grade):

- medium access control
- reliable text messaging
- ad-hoc multi-hop routing and forwarding

Advanced features, e.g.,

- Security
- Group chat

- Advanced GUI
- other advanced features

The final report will be evaluated, taking into account both what it describes (the designed system, tests, etc.), and how it is described:

- System architecture and diagram, and overall readability
- Description of system and its functionalities
- Test setup and scenarios
- Test results

4 Timeline and important dates

Please pay special attention to the following dates: ...to be updated....

- 2 April, 10:15: kick-off integration project with a short introduction to the project, requirements, and deadlines
- 4 April, 11:59 (am): deadline for registering your group in Canvas
- 5 April, 23:59: deadline for submitting project planning (submitting earlier is recommended)
- 4 & 8 April: each group should discuss its planning including timeline, task division, and responsibilities with one of the TA's, have it signed off, and subsequently plan a time-slot for the final demo
- 12 April, 10:00 - 17:00: each group should show an initial demo of its system to a TA
- *16 April, 12:00-17:30: final demo*
- *18 April, 23:59: deadline to submit final report and code*
- *18 April, 10:00-12:00: demo of groups who failed on April 16th*