Network Systems integration project - Scuba chat

2/4/2020

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 2 students. With this group of 2 students, you design and implement software of which at least 4 instances (4 processes) are running on your laptops, providing 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 waves may be possible, be it at rather low data rates. Your chat application should use wireless communications using sound waves, utilising a physical layer provided by us (hence the name scuba chat).

The project was initially defined for groups of 4 persons, requiring 4 processes on 4 different laptop. But as we think cooperating with a group of 4 students might be hard in times of Corona, we have limited the scope, required functionality, and the group size of the project, resulting in multiple instances of the chat running on one laptop. Furthermore, to allow for remote / online development, testing, and demoing, the physical layer framework we make available also supports virtual sound communications, using an Internet connection to our emulation server, which "transmits" messages to other nodes within emulated range, tuned to the same channel. We recommend that you use this emulator during development and testing. It also has to be used during your final demo.

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 at least 4 processes, running on your 2 laptops, and using the (emulated)

sound-based physical layer that we provide.

- Your chat service has to support reliable text broadcasting, i.e., text messages of arbitrary length typed in any of the instances should be delivered identically and in the correct order at all other instances.
- 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 (frequency). Multiple channels (frequencies) have been 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 addressing.
- You will have to design a mechanism for medium access control.
- As the range of the sound-based physical layer is rather limited, you will have to come up with some forwarding mechanism to also exchange packets between nodes (instances) which can reach each other only via one or more intermediate nodes. (Please note that in the emulator, the position, and hence connectivity of the nodes can be set using the web-interface we provide.)
- As the sound-based physical layer is quite error-prone, you will have to design a protocol for reliable transfer of broadcast messages.
- 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.
- Think about the user interface (but do not invest too much time in making it fancy).

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 we do not want you to sit together with your group mate in the same room (to avoid the risk of Corona infection), we also make available a variant for virtualized sound communication. This latter variant is using a server we provide, and which you communicate to using the standard networking facilities of your laptop. Using a special web-interface you can configure node (instance) locations and other settings of this virtual sound channel.

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 types you can send and 6 you can receive. These messages types are specified below.

Sending:
DATA: With this message you can send a frame of 32 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 (32 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 of the receiver 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://netsys.ewi.utwente.nl/integrationproject/` to view and manipulate the (emulated) positions of your nodes and see if they are transmitting. The help page `http://netsys.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:

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

2.1 Registration

Please start with forming a group (of 2 students), and registering as such. Registration is done through Canvas (People –> Groups) by choosing a free group number, and each student registering for that group number. Please note that the group numbering is also used for planning the final demos. During the demo afternoon, we will schedule all groups in ascending order, so a low group number will give you an early time slot, a high group number a (relatively) late time-slot. Please make sure you hand in source code and report also from this group (see below).

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:

- a simple user interface
- presence / reachability information
- reliable text messaging (broadcasting), even in the presence of packet drop (emulated by our server)
- multihop forwarding of messages to nodes only reachable via intermediate nodes

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 on one of your laptops (i.e., 4 windows for 4 instances/nodes) at the beginning of your time-slot.

2.3 Source code

The source code of your project has to be submitted in a .zip-file using Canvas.

2.4 Final report

The final report should be submitted via Canvas. Please hand in your report as .pdf-file. 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:

- Overview of the system. Which functionalities and protocols did you identify, how do they relate?
- Description of designed (and implemented) system functionalities, protocols, methods/algorithms designed/used.
- Description of the testing of your system, functionally, and possibly also in terms of performance.

The final report should be 2 (max. 3) A4 pages.

3 Evaluation criteria

The evaluation of the integration project will be either pass or fail. 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 and assessed to be sufficient in order to get a pass.

4 Timeline and important dates

Please pay special attention to the following dates:

- 6 April, 10:30: online kick-off integration project with a short introduction to the project, requirements, and deadlines (Through Canvas Conferences)
- 7 April, 11:59 (am): deadline for registering your group in Canvas
- 7 April, 14 April, 15 April am: each group works on the final project (incl. report)
- *15 April, 13:30-17:30: final demo*
- *15 April, 23:59: deadline to submit final report and code*