Concurrent and Distributed Programming
- Assigned Project -
Document Version 1.0

Lecturer: Mauro Conti, PhD - conti@math.unipd.it
Teaching Assistant: Eyüp Serdar Canlar - canlar@math.unipd.it

Academic Year: 2011-2012

1 Project Overview

This document describes the specifications of the project. Each group (of at most two students) has to develop a Java application to simulate two clone detection protocols for Ad-Hoc Network: LSM [1], and RED [2].

We expect the developed system to involve the following components:

- A configuration file available on a http server (the file URL is a parameter that the user can set).
- A simulator (client), that actually runs the simulations as specified in the configuration file.
- An RMI server which gets the results of the simulations from the client(s).

The behaviour of the system should be as follows:

1. (Start the RMI server)
2. (Start the client simulator)
3. The client has to provide a simple GUI where the user should be able to specify: (1) the URL for the configuration file; and (2) the URL for the RMI server. The corresponding text fields, while modifiable, should automatically propose to the user: (1) the URL provided in Section 2 as an example; (2) the URL for the RMI as if it was running in the localhost. The GUI must have the following buttons: `START` and `STOP`. Pressing `START` will cause the simulation to start as specified in the configuration file. Pressing `STOP` (at any moment) will cause the simulation to stop and exit the client application. The GUI also has to include a text area where it is shown each information that has been sent to the RMI server. Other information about the progress of the ongoing simulation can be shown (this is not strictly required) in another separate text area.
4. Once the client obtained the configuration file, it has to process it and run the simulations accordingly to the information in the configuration file (as described in Section 3).

5. For each of the `NSIM` simulations, the client has to send the RMI server:
   - The list of the value of the variables (only the values, not their names), considering the order of variables given in Table 1.
   - *Min, max, average, and standard deviation* of *each* of the following variables referred to the network nodes: *number of sent messages, number of received messages, number of signature checks, amount of consumed energy, number of messages stored in the node.*
   - A value 0 (false) or 1 (true), depending whether during the simulation at least one node detected the presence of the clone.
6. For each list of information received in the previous point, the RMI server created a string line and *append* it to the file `output.txt`.

2 Configuration File

The configuration file is a text file where each line can have one of the following formats:

- A blank line.
- A comment line (starting with `%`).
- `[VARIABLE] = [VALUE]`

The list of all the variables `VARIABLE`, their meaning, and their possible corresponding values `VALUE` are specified in Table 1. The use of these variables in the simulator will be clear in Section 3. Blank spaces should be ignored. File not complying with these format will not be accepted: a proper message should be shown in the simulator client application.

Table 1: Table of variables.

Symbol	Meaning
$PROTO$	Protocol to be simulated (either `LSM` or `RED`)
$NSIM$	Number of simulation to be run (int)
n	Number of nodes in the network (int)
r	Communication radius of a node as a fraction of the side of the unit-area (float)
p	*Probability* for a neighbour node (float) to process a *location claim*
g	Number of destination location (int)
E	Total energy available on a node (int)
E_send	Energy spent for sending a message (int)
$E\_receive$	Energy spent for receiving a message (int)
$E\_signature$	Energy spent for verifying a signature (int)

An example of the configuration file is available at:
`http://www.math.unipd.it/~conti/teaching/PCD1112/project_config.txt`

3 Simulator

An *hypervisor* of the simulation will set up the simulation. In particular, *for each of the `NSIM` simulations, the hypervisor has to do what follows.* Simulate (the word "simulate" will be omitted from now on when it is clear from the context) the creation of the `n` nodes. Each node will be assigned a *unique* ID (e.g. just a number), and it will be "placed" in a random position (x,y) within a *unit-square area* (a square with side of size 1). Each node has a communication radius `r`. That means, when the network is "created", each node has the knowledge of his neighborhood (the list of nodes within the radius `r`, and their position). Once the nodes have been deployed, the hypervisor has also to simulate the clone attack. This is done as follow. A random (attacked) node in the deployed network is selected. A *new* nodes (the *clone*) is hence created and randomly placed in the network. Note that the *clone* has the same ID as the *attacked* node.

Once the attack has been done, the simulation with that specific setting (considering the specific deployment, and the specific attack) will start. Note that in the behaviour described in the following each network node is implemented via a separate thread. The behaviour of the simulation in the case `PROTO` is equal to LSM is specified in Section 3.1. The behaviour of the simulation in case `PROTO` is equal to RED is specified in Section 3.1.

3.1 The The Line-Selected Multicast (LSM) Protocol

1. At the beginning of the simulation, *each* node has to broadcast a *location claim* (the sender sends a message; each of its neighbours receives a message). A location claim message contains the ID of the claiming node, and the node position (x,y).

2. Each node, say B, that receives a *location claim*: with probability 1-`p` just ignores the message (this implies no other actions); with probability `p` the following steps are performed.

3. B selects a random point in the unit-square area and follows the routing procedure as specified in Section 3.3.

4. The point as selected in the previous step is set as the destination of a control message. The control message is then forwarded according to the routing procedure (Section 3.3).

5. For each node involved in the routing of a message: it stores the message, and it checks for clones considering the stored messages (the check also involves a signature verification).

The steps from Step 2 to the end are repeated g times.

3.2 The Randomized, Efficient, and Distributed (RED) Protocol

1. The hypervisor randomly selects an int number (`rand`) and let each node know this number.

2. As Step 1 of LSM.

3. As Step 2 of LSM.

4. B selects a point in the unit-square area. The point is selected using a pseudo random function (e.g. a hash function) that takes as input: the ID of the node that sent the location claim, the `rand` value for the specific simulation, and a counter value (that starts from 0 and it is incremented for each of the g points generation).

5. As Step 4 of LSM.

6. For each node involved in the routing of a message, it will only forward the message, and check for clones only if according to the routing it is the destinatary of the forwarded message.

The steps from Step 2 to the end are repeated g times (varying the value of the counter in Step 4 accordingly).

3.3 Routing procedure

Each message has a point as its destination. Each node in charge to forward a message will send the message to the neighbour that is closest to the destination. This behaviour implies the corresponding energy consumption on the sending and on the receiving node. If a node in charge of forwarding a message is more close to the destination point than any of its neighbours, it will not forward the message (and define itself as the destinatary of the message).

3.4 Clone Detection

The clone detection is done by verifying whether the node has stored in its memory two claim messages M and M', where the ID in M and M' is the same, while the position claimed in M and M' is different.

4 Non-Specified Aspects

For all the aspects not specified in this document and not clarified in the discussion in the Mailing List of the course, we leave to the group the freedom to take their (motivated) choices. The motivation should be discussed in the Project Report (see next section). We strongly suggest the students to raise and discuss in the Mailing List any unclear matter that they would consider very relevant for the system behaviour.

5 Project Report and Submission

The submission must be done within the deadline indicated on the web page of the course. The student has to submit a single `.tar` (or `.zip`) file via email (to **both** `conti@math.unipd.it` and `canlar@math.unipd.it`). The file has to contain:

- All the code written for the project.

- A `readme.txt` file, containing the instruction on how to start the simulator and the server.
- A project report (pdf file of at most 5 pages, font size not bigger than 11pt, written in English). The report should describe the main working of the system (at most 2 pages), and the motivations of the choices for the issues not specified in the present document (at most 3 pages).

References

[1] B. Parno, A. Perrig, and V. D. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE Security and Privacy*, 2005, pp. 49–63.

[2] M. Conti, R. Di Pietro, L. Mancini, and A. Mei, "Distributed detection of clone attacks in wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, pp. 685–698, September 2011.