Uppsala University

Information Technology

Wireless Communication and Built-in Systems

Cooja Lab - An Intrusion Detection System

Student: _____

Email: _____

Date: _____

February 18, 2024

Formalities

For lab 3, you should sign up under a different group set than lab 1 and 2. New groups are created and their names indicate lab 3. You are encouraged to use available resources such as online code examples. However, you are not allowed to use code from other students doing the assignment. If you discuss with another group, please state so on the front page to give them credit and avoid suspicion of cheating. The assignment is divided into two tasks, and you need to pass both tasks to pass this lab.

During the examination session, you need to demonstrate the solution and be able to answer questions to pass this lab. In order to pass, your code needs to perform accordingly to the task descriptions. Test cases are provided in studium to test your code.

Setting up the Cooja simulator

The Cooja simulator is part of the Contiki distribution that is installed on the virtual machine image. To start Cooja go to the folder /contiki-ng-wcnes/tools/cooja and in the terminal run the command *ant run*.

In the lab3 folder, you will find a Cooja simulation files called *simu_lab3_task1.csc* and *simu_lab3_task2.csc*. Those files defines a simple scenario containing a few sensor nodes that runs the codes of each task. Open the simulation file in Cooja and start a simulation by clicking start in the Control Panel. It can be a good idea to set the speed of the simulation in the Control Panel under the drop down menu "speed limit".

Take a few minutes to get familiar with the Cooja simulator. Find out how to reload the simulation, where to read the serial output for each node, and how to control the simulation speed. Note that the nodes can take a while to boot up. The easiest way to see when they are done is to look at the serial output.

You can click a button on a simulated sensor node by right clicking on the node in the 'Network' pane and selecting "Click button on node X".

Take some time to get familiar with the Cooja simulator before start with the Tasks. Make sure that you know how to start Cooja, load the simulation file and know how to start and stop a simulation.

Setup OK:_____

Introduction

The goal of the assignment is to get to know the simulator Cooja. You will start with a simple version of the clicker program you know quite well by now. In this simplified version, there is no difference between client and base station nodes: all nodes run the same program. If a button is clicked on a node, it will send a broadcast packet to all its neighbors. If a node receives a packet, it will toggle its green LED.

Your task is to change the clicker application into a distributed intrusion detection system. This distributed system will reduce the number of false positives, which will reduce the network traffic and extend the lifetime of the system. In the examination, you need to demonstrate to a teacher that your application works and explain the essential steps.

Make yourself ready for the lab

Task 1

Modify the simplified clicker application such that an alarm is only triggered when three distinct nodes that are close to each other have detected an event in the last 30 seconds. Nodes are considered "close" to each other if they are direct neighbors, i.e., if they can communicate with each other over one hop. For our scenario, we consider button clicks as events. In addition, if there is no upcoming events for a certain period, the LED which indicated the alarm should be turned off. If you reload the simulation, the clicker.c file is recompiled and the image is put on the simulated nodes.

Here are some hints:

- To visualize that an alarm is triggered – i.e., that at least three nodes have detected an event – you can for example toggle the blue LED.
- You can use the function `clock_time()` to get the current time of a node. The result is measured in clock ticks and has the type `clock_time_t`.

The constant `CLOCK_SECOND` defines how many clock ticks there are in one second.

- A good way to keep track of events is to keep recent events in an array of structs, for example

```
struct event {
    clock_time_t time;
    linkaddr_t addr;
};
#define MAX_NUMBER_OF_EVENTS 3
struct event event_history[MAX_NUMBER_OF_EVENTS];
```

- For debug purposes it can be a good idea to write a function that prints out the current event_history structure.
- One possible approach to this problem is to implement a function `void handle_event(const linkaddr_t *src)` that updates the event history and checks if an alarm should be triggered. This function would be called when a broadcast packet is received, or when the button on the local node is clicked. It may be helpful to sketch what this function should do on paper before starting to program.
- Your program should not make any static assumptions about the network topology. E.g., the node with the Link address 2.0 should not assume that the nodes with addresses 6.0 and 1.0 are in its neighborhood.
- There is a function `linkaddr_cmp()` that takes two Rime addresses and checks if they are the same or not.

Task 1 OK:_____

Task 2

The goal of Task 2 is to understand the behavior of of nodes and its positioning in the routing table of the network. For this part the lab, there is a new code in the *../Lab3/Task2* folder. This code creates a wireless network that operates on RPL Classic protocol of Contiki-NG. Go through the code to understand what it is doing. This is a simple code that prints out the

routes of the network. Every node will print the routes that a message would take from a specific node until the top of the routing table.

Your Task is to modify the code to turn on the LEDs according to the nodes position inside the routing table. The LED signaling need to be the following:

- LED GREEN: Root Node
- LED BLUE: Intermediate Nodes
- LED RED: Endpoint Nodes
- All LEDs OFF: Node Outside the Network

After implementing this LED singling, try different topologies by moving the nodes around the network window. Try to predict the print out messages for each topology you assemble, verify the LEDs signaling and obseve the time it takes for the routing table update.

Hints:

- In order to move any node, just click, hold and drag it.
- Remember that the time it takes to print the routing information is bounded to the timer event.
- Observe the Radio Range of the node marked by the 2 concentric circles (green and Gray) when building your topology.

Task 2 OK:_____