Masaryk University
Faculty of Informatics

Edu-hoc: Experimental and educational platform for wireless ad-hoc networking

Master's Thesis

Lukáš Němec

Brno, Fall 2016

Masaryk University
Faculty of Informatics

Edu-hoc: Experimental and educational platform for wireless ad-hoc networking

Master's Thesis

Lukáš Němec

Brno, Fall 2016

Declaration

Lukáš Němec

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgement

TODO thanks

Abstract

TODO abstract

Keywords

keyword1, keyword2, . . .

Contents

1 Introduction

2 Problem analysis (testbed, not general WSN)

2.1 Creating WSN network

2.2 Possible challenges

3 TESTBED deployment

3.1 Network design

3.2 JeeTool

Motivation Few nodes can be managed manually without any major problems, most frequently using directly Arduino IDE [1]. This approach is works flawlesly for up to three devices and it is possible to manage up to 10 devices this way, but with great effort and sacrificing the usability.

Either intentionaly or not, there is no doubt that the official Arduino IDE is designed with simplicity in mind, therefore it is usable for person without solid background in IT [2]. On the other hand, this simplicity comes at the cost of many features, that might have been usefull for more advanced use (e.g. managment of multiple devices).

MoteTool inspiration To remedy this problem, jeeTool [3] was created, deployment tool that uses Arduino Makefile project [4] to manage the network of 25 Arduino devices. This tool was inpired by motetool project [5] which was developed by Dusan Klinec in order to manage simmilar task, only for different type of devices. The basic functionality is the same, it allows users to detect all connected devices and upload applications to the selected ones. In additon jeeTool also supports bidirectional communication with individual devices via serial port.

Implementation details As has been stated, Arduino makefile is used for the actual application upload. In order to make Makefile call for each of the connected devices, jeeTool creates temporary bash script file, which is used to set the enviroment for the specific node and execute the upload via Makefile. Temporary script itself is then executed with system call directly from jeeTool. If multiple nodes are selected for upload, jeeTool enables either serial deployment (mostly for debugging purposes) and parallel deployment, where all Makefile calls are made within dictrete threads, thus all of them are executed simultaneously.

Communication over serial port In addition to the upload features, jeeTool also enables communication with devices over serial port. This is done via jSerialComm library [6] at the moment, in the past RxTx library [7] was used, unfortunately this library is currently not being developed[1] and decision was made to change library to more up to date one, instead of using problematic old code.

Individual nodes are matched with files, whicha are named after the nodes (name of the device in the linux file system) and jeeTool is capable of writing all inboud communication in such file or read such file and write it over to the serial port. For the inbound communication there is an option of delayed termination of the program, thus jeeTool can be left runnning for specified amount of time before the ports to devices are closed (e.g. collect results after the experiment). For the outbound communication jeeTool can either send whole content of file at once or there is an option of delay between individual lines of the file (can be used for periodical sending of messages etc.).

3.3 Hardware selection

New network was designed with both the research 4 and educational 5 use in mind. Therefore the selection of hardware for individual nodes was limited by these intended usecases. Since there already was experience with the research type network [8] and its pitfalls related to the complicated use of specialised hardware, we opted for the developer friendly approach; devices based on Arduino platform 3.3, which are designed with simplicity in mind, as Arduino focus group are not only programmers, but also artists and generally people without any advanced IT skills.

JeeNode USB and JeeLink Devices selected for the network are Arduino clones JeeLink nodes [9]. In addition to that, also several JeeNode devices [10] were obtained, these two are similar in many

1. last update is from 2011

ways and can interact with each other, JeeNode USB is only a little bit more versatile in its use options.

Both of these devices are based on Arduino Mini [11] with ATmega328P processor. Wireless communication is enabled by the RF12B chip 3.3 [12] which is permanently mounted to the device itself. Both devices can be connected directly to computer via USB[2]. The other most important difference between these two types is the fact that JeeLink has plastic cover, therefore there is no additional conectivity. On the other hand, JeeNode is not covered and there are pins available fo additional sensors or any other devices, that one might need to connect. Last, but certainly not least, JeeLink has additional 16 Mbit of flash memmory [9] available, therefore this can be used for convenient storage.

For both of these devices there is acompaniing library JeeLib [13] that contains all necessary functions for controlling the radio and any ther function that is different from standart Arduino Mini.

RF12B radio module Simple and quite cheap radio module from HopeRF [12] is present on all JeeMote and JeeLink devices. Version sold in Europe is using 868 MHz frequency for the radio[3]. As many other common radio modules, this radio is half-duplex, therefore it is able either to receive or to transmit messages, but cannot do both at one time.

Radio module is equipped with omnidirectional whip antena ($\approx$ 8 cm long wire) by default. It is possible to modify the propperties of such antena by shaping it in various ways, e.g. straight wire compared to coiled one. There might also be many other versions and shapes of antenae, which might improve radio properties, but this would be subject to whole another research in quite a different field.

2. the USB port type is different
3. other version can use 433 MHz or 915 MHz

4 Research use

One of the intentended uses for the created testbed was its use in research which can be of various nature. In this particular case, we would like to make a case study for key generation mechanism without any pre-shared information between individual nodes.

In todays aplicсation this would be quite common problem, since the amount of devices equiped with some kind of radio module is constantly groving, let it be wifi, bluetooth, or any other frequency and specification. However, we cannot expect for these devices to have any shared data nor capability to exchange them over already secure channel, since the secure channel is what we are trying to acomplish.

What we would like to show, is sheme for key derivation without any shared secret, only using properties of radio channel shared between these two devices and the nature of radio wave propagation through such shared channel.

4.1 Keys from radio signal

Due to a nature of an wireless channel whcih is not perfect, we can observe fluctuations and disturbances in the wave propagation. It might seem difficult at first to measure these, since by no means we are able to observe all such events. Despite that there is a way, almost every radio module is able to measure RSSI[1], which can be described as a amount of power in the received signal. There are many arbitrary scales and absolute values do not match on devices from different manufacturers

However, our alghoritm can work regardless to the differences in devices, even two devices with same configuration can produce different absolute values due to antena position and overall device manufacturing differences, since some of these devices are partially assembled by hand [2].

1. received signal strength indication
2. e.g. JeeNode motes come as a ready to assemble kit

4.1.1 Quantization principle (bits from signal strength)

Quantization enables to extract bits from individual values of signal strength. There are many different approaches to this problem two main approaches here are: lossless quantization and contrary to that we have lossy quantization.

Main difference between these is number of generated bits per original signal strength measurements, while lossless quantization produces bit value from every measurement of signal strength, which is useful for high-performance demands, but it requires guaranteed variance in the radio channel (e.g. the nodes are constantly moving, or the environment is changing) during the key establishment phase. Otherwise, the resulting keys could possibly be very weak.

Lossy quantization, on the other hand, does not have guaranteed output length per number of measured values, which can lead to a very limited length of output. However, this kind of quantization is expected to have better results in static environments because its nature is to drop such bits, that fail to differ from others.

Also there are few different measurements of the channel one can use to produce bits. One would be RSSI measurements on received messages, other measure

Since our network is static and without any moving nodes, we implemented lossy quantizer algorithm designed by Mathur et.al., which showed promising results for the of the shelf wireless devices similar to ours, and also contained experimental results from several different scenarios, where some of these were comparable to our conditions.

4.1.2 RSSI Quantization

Quantization princeple designed by Mathur et.al. works as follows:

1. both nodes send *n* messages to each other in alternating pattern, both nodes send counter value inside these messages, which is used to synchronise messages on individual nodes. For every one of these messages, signal strength is measured upon reception.

2. when *n* messages have been successfully exchanged, both nodes can proceed to the computational part.

3. both nodes calculate mean m and standard deviation sd for signal strength values of all received messages.
4. both nodes calculate q^+ and q^- values, which are upper and lower quantizer bin boundaries, as follows:

$$q^+ = m + \alpha \cdot sd$$

$$q^- = m - \alpha \cdot sd$$

5. every signal strength measurement is then processed and it is rejected, if it lies within q^+ and q^- boundaries, values above this range are assigned a bit value of one, values below are assigned a bit value of 0.
6. nodes then synchronise their measurement by exchanging counter values associated with those messages, where signal strength measurements were assigned either one or zero bit values.
7. those counter values that match on both nodes are expected to be excursions in the same direction and are used in the final outcome.

4.2 Cooperative jamming (can it improve our situation?)

4.3 Performance Evaluation (results from experiments)

4.3.1 Enthropy of data

4.3.2 Speed (bits of key per time)

4.3.3 Possible errors

4.4 Discussion, is it achievable and under what conditions?

5 Education use

5.1 motivation for educational WSN network

The current state of the art WSN devices usually uses specialised hardware and software in order to achieve the best performance available. This, unfortunately, is not the ideal prerequisite for an easy to learn matter. In fact, most of WSN devices have rather complicated setup and are quite challenging for novices.

Because of such discouragement, it is difficult to teach how to work with WSN's; few hours (at least) are usually required to explain the basics, which is reasonable for a research project or something similar, but for a class exercise, this would turn out to be not the most effective use of time, if it would be achievable at all. And we have not yet mentioned more advanced topics in this area, such as common techniques for encryption or message authentication.

Issue of this nature can be solved in various ways, in the case of Edu-hoc, we decided to sacrifice performance 3.3; which is not that much important for a network with an educational purpose. On the other hand, using hardware that is really easy to comprehend and use is of a great benefit here. Also having less powerful, but relatively cheap devices (in the range of $30 rather than $100 or more) gives the opportunity to lend each the students one of the devices, so that they can try the basics on their own, and also use this device for interactions with the network.

5.2 Scenario approach (attack and repair) + iterative higher difficulty

In order to make learning more enjoyable experience and also to add some challenging part to the learning process, we decided to make Edu-hoc scenario based; each scenario being composed of two distinct parts: first part in the role of an attacker (both enjoyable and educational) and the second part as a code reviewer or developer (primarily educational).

attacker part of a scenario Students are presented with network application which has known, or easily detectable, vulnerability. Task is to take advantage of such vulnerability and exploit it using own nodes and carefully executed interactions with the network. How successful these efforts were can be easily evaluated by percentage difference from the expected traffic of the network or by presenting learned secret in the submission of the solution. Exact methods of evaluation are described in 5.3.

reviewer part of a scenario An important part of the educational process is not only to find mistakes, but also to be able to correct them. The task in this part of the scenario is then to take the current source code of an application, which has been deployed on the network during the attacker part 5.2, correctly identify which mistakes were made and propose changes in the code, which would make the application secure against such kind of an attack. Exact evaluation methods are again described in 5.3.

5.2.1 1st scenario - Eavesdropping

The first scenario can be considered really simple and it is by design. Since this scenario is the first encounter with Edu-hoc, we opted for passive attacker approach, without any actual interaction with the network, just listening for any traffic.

Eavesdropping is also the first thing one would do if one would be about to attack some network because it does not compromise the presence of an attacker and intercepted messages may provide many useful pieces of information. This is another important reason why this task was selected as a first one, not only it is rather easy to do, but it provides vital information for the rest of the exercises (e.g. which nodes are present in the network, which frequency is used and what are the settings of the radio)

scenario setting Each node in the network sends messages, after each message, there is a delay of 1000 ms, therefore any receiving node has time to process the transmitted messages. All messages are sent as a broadcast and nodes within the network do not receive nor process them since it is not needed for successful scenario run. If this would be the case, transmission rates would have been updated accordingly and the scenario would be much more likely to fail on its own, because of single node malfunction, while with current settings, the network can operate without any problems even is several of the nodes would fail.

attack principle The attack is very simple, attacker node only has to listen for any traffic present in the network. For best results application has to be able to process as many messages as possible, however any performance changes will affect the end result only slightly (final percentage will be better, but the bottleneck is in the processor of the radio module, so better optimised application will increase the end result only by approximately 10-20%)

application and securing it The main issue with the application is unencrypted broadcast. This can be fixed quite easily, just by the addition of simple encryption and for simplicity using common shared key. We have to be aware of the fact, that attacker might be able to collect some of the nodes and thus learn such key and compromise it and there are techniques which deal with this problem However, in our scenario, this is sufficient solution because it is adequate to measure against eavesdropping.

5.2.2 2nd scenario - Black hole attack

This scenario presents more advanced concepts, dynamic routing in particular. The task is to attack the routing algorithm and diverse all traffic so that the central node does not receive any messages. This scenario requires active attacker, thus some basic interaction with the network.

scenario setting The scenario is divided into two parts, first parts are shorter and are used to establish routes for this particular run of scenario. Without any attack or interference outside from the network, these routes should be always the same.

In the following part of scenario each node periodically sends messages to the parent node, therefore all messages are routed to the central node. This node then counts all received messages and the final outcome is the number of expected messages compared to the number of actual messages received.

attack principle This attack requires some prior knowledge before execution which can be obtained simply by using eavesdropping technique from the previous exercise. This way an attacker can learn how the routing algorithm works (nodes broadcast their current distance to the central node, where central node has distance of 0, all nodes that hear distance announcement message then update their distance and broadcast this updated distance)

With this knowledge attacker than try to inject message with the same content as the central node does, but earlier than the central node is scheduled to do so. After this, the message from a central node will not improve the position of other nodes and they will not update their distance. Another possible approach is to find an intentional weakness in the algorithm implementation (application allows negative distance values).

The only thing that remains is to make sure to use nonexisting ID within the network and after the routing tree is established, attacker node can simply disappear, because it is no longer needed.

application and securing it In order to fix the application one has to fix two issues; check for negative counter values (which is the easy part and requires usually addition of one condition within the code); and what is more important, implement some kind of authentication mechanism for messages from the central node, or otherwise make sure that the attacker cannot diverse all the traffic only by being a bit faster than the central node.

Many options are available, the easiest one being preshared random authentication tag, which would be added to the messages. This is really simple countermeasure, but it guarantees that the attacker can't send a valid message before the central node sends it.

5.2.3 3rd scenario - Sinkhole attack

Although very similar to the second scenario, this poses far greater challenge than the previous one. The goal is not only to divert the traffic but also modify it on the fly and send it to the original recipient.

The greatest issue here is related to the performance, since all messages from the entire network will be routed through single attacker node, therefore most of the messages are very likely to be dropped somewhere in the process, because single node is not capable of receiving and processing messages from every other node in the network, if they do not transmit on very slow rate.

Task for this scenario is to deliver a modified message to the central node before the original message is delivered; messages with the same identifier but different content are discarded and only the one which was received the earliest is kept at the central node.

The reason why two very similar scenarios exist, while it could have been only this one, is to present scenarios in a manner of slowly increasing difficulty. If we were to skip the second scenario 5.2.2, the task might be too complicated to some, while this way, required task in the third scenario 5.2.3 is only a minor extension to the second one, however, the achieved result is far greater impact.

scenario setting Runtime of this scenario is the same as for the previous one 5.2.2; initial phase of route establishment and after that messages are routed from nodes to the central node. The expected outcome is to divert ad modify as many messages as possible.

attack principle The first part of the attack can be executed in the same fashion as the previous one 5.2.2 although it is not the ideal way. As the goal is not to prevent all messages from reaching its destination, but to deliver as many modified messages as possible. Therefore it is not a good idea to modify all of the routes in the network, but only as much as the attacker node is capable of handling (i.e. only third of the nodes in the network). This can be done by sending the distance message only to selected nodes instead of broadcasting it.

Another part of the attack is rather simple, attacker node has to receive all messages which are addressed to it, modify these messages as it is required and send these messages to the central node. Modification usually requires simple text substitution inside the body of the message or task very similar to this one.

application and securing it Again as many times with this scenario, the fix is partly the same as per second scenario 5.2.2, which deals with the attack on the routing; either preventing it or detecting it.

The another part of this fix is primarily an authentication problem, therefore any kind of MAC will be able to secure this application. Therefore the problem of shared key arises again, but as before in first scenario 5.2.1 one master key will be able to do the job, if we assume, that the attacker does not have access to individual devices, only to radio communication between them.

The point of the exercise is to deal the problem at hand without complicating it with anything else. Of course, we could add any advanced key distribution schemes to the application in order to add countermeasures against master key compromise, but that would make completely different application without any real added value to the original educational purpose.

5.2.4 4th scenario - Jamming

Evey once a while attacker comes around a network, application or protocol which is designed with security in mind, meaning there are no backdoors or weak spots in the design or implementation (really idealistic scenario, however, it is very likely that the attacker does not resource to find or use existing vulnerabilities, thus for such attacker there is no real way how to exploit such application).

Nevertheless, it is very likely that such attacker would like to cause any kind of damage he is capable of doing. Most likely to make the application, protocol, or network unavailable; most likely by DoS attack or something similar. For wireless communication it is quite easy to do, since the wireless medium, air to be specific, can be used by only one device at a time. If an attacker is able to utilise all ate available slot for transmissions, then the legitimate user would not be able to transmit anything. Such technique is called jamming and it is quite easy to deploy.

scenario setting The setting of this scenario is really simple, all nodes send a message via the fixed routing tree to the central node, who the counts the number of delivered messages. There is no phase of route establishment, only the message delivery phase. The assignment is to prevent delivery of as many messages as possible. Since there is no option to modify the routes etc. the only reasonable way is jamming.

attack principle As has been stated before, jamming is the expected way of solving this scenario, although there might be many creative solutions created. With jamming one can either bypass the media access protocol and send data continually, thus preventing everyone else from sending any messages, or the another viable option is to send many valid messages to the central node, thus overloading it and forcing this node to drop many incoming messages, including the valid ones.

application and securing it Since this attack does not exploit any weakness, only the actual property of wireless communication and network design, there is not much one can do to prevent such attack, while only modifying the application source code. However, there are ways of dealing with such kind of an attack. Countermeasures are usually based on multiplication of the devices and decentralisation of the whole network. Therefore give two or even more nodes the ability to become the central node, use multiple frequencies etc.

5.3 Evaluation principle

Measuring the success can be done in various different ways, Eduhoc utilises few different methods to decide, whether the attack was successful and how big was the impact of an attack.

Evaluation for each scenario is a little bit different because there are different objectives. The task of first scenario 5.2.1 is to capture as many messages as possible, therefore the evaluation is done as a simple comparison between the set of captured messages and set of pre-computed messages that are known to be transmitted in the time period of scenario run. Comparing these two inputs line by line we get intersect of these two sets, that contains all successfully captured messages and is stripped of all the duplicates and possible fraud messages. A number of messages in the resulting is then compared to the expected number of messages in order to receive the final percentage evaluation, the higher the number, the better the result.

The second and the fourth scenario 5.2.2 5.2.4 have the same goal, only differ in the scenario settings, which can be omitted for the evaluation purposes. The common goal is to prevent as many messages as possible from delivery to the central node. The way it is done is rather simple, the central node counts every delivered message and the resulting number is compared with the expected amount of messages.

Unfortunately, this design of scenario lacks in the evaluation one important thing, the credit for the attack cannot be given to any participant, because we have no identifier involved. Messages only vanish, but we do not know, whether it was by joint work of multiple attackers, one really successful one or by a fault in the network itself. Therefore there is not much of use for such scenario in the courses, and because of this, modified versions of scenarios have been used 5.5.

The third scenario 5.2.3 however utilises message modification, which can be used for identification purposes. Therefore messages on the central node are not only counted, but alco compared to the expected ones and differences are noted[1].

Since students are assigned a unique identifier to modify the messages with, we can easily distinguish between participants and evaluate how successful they were in this task.

The last option which can be utilised in evaluation is a submission of secret, which was received from interaction with the network, node either responds to received message with such secret or the secret has to be extracted from the communication in the network. Such secret can be a short word, number or something of similar nature and again such secret can be unique for each participant, which makes te evaluation an easy task.

1. the amount of computation on this level would limit the usability of the node, thus the node only sends received messages over the serial port to the server, where the actual evaluation is done

5.4 Web interface and auto run

Edu-hoc was designed as a long term excercise, thus single scenarios are expected to run for days, possibly week or longer. However, for some it would be extremly impractical to repeat only once during the whole time, because we would like to present fresh scenario to everyone involved, and also we would like to reset the scenario once a while. This will ensure correct initial settings and if we set the intervals right, then everyone will be able to test out several solutions within reasonably short timeframe.

To achieve such kind of behaviour, we can utilise various approaches, the most easy one being set of bash scripts and setting the cron service acordingly. The other option might be modified JeeTool, which would be running continuisly and in set intervals would trigger corresponding action.

The last option is the most complicated one, but offers the most

5.5 PA197 use and results

6 Future and related work

6.1 RSSI

6.2 Edu-hoc

7 Summary

Bibliography

[1] *Arduino*. https://www.arduino.cc/. 2016.

[2] *Arduino Introduction*. https://www.arduino.cc/en/guide/introduction. 2016.

[3] L. Němec. *JeeTool*. https://github.com/crocs-muni/JeeTool. 2016.

[4] *Arduino-Makefile*. https://github.com/weareleka/Arduino-Makefile. 2016.

[5] D. Klinec. *WSNmotelist*. https://github.com/ph4r05/WSNmotelist. 2014.

[6] W. Hedgecock. *jSerialComm library*. http://fazecast.github.io/jSerialComm/. 2016.

[7] *RXTX library*. http://rxtx.qbang.org. 2011.

[8] Vashek Matyáš et al. "WSNProtectLayer: Security Middleware for Wireless Sensor Networks". In: *Securing Cyber-Physical Systems*. CRC Press, Sept. 2015, pp. 119–162. ISBN: 978-1-4987-0098-6. DOI: doi:10.1201/b19311-6. URL: http://dx.doi.org/10.1201/b19311-6.

[9] JeeLabs. *JeeLink product page*. http://jeelabs.net/projects/hardware/wiki/JeeLink. 2016.

[10] JeeLabs. *JeeNode product page*. http://jeelabs.net/projects/hardware/wiki/JeeNode_USB. 2016.

[11] *Arduino Mini*. https://www.arduino.cc/en/Main/ArduinoBoardMini. 2016.

[12] HOPE RF. *RF12B Datasheet*. https://www.sparkfun.com/datasheets/Wireless/General/RF12B-IC.pdf. 2016.

[13] JeeLabs. *JeeLib library*. http://jeelabs.net/projects/jeelib/wiki. 2016.

A An appendix