PROCEEDINGS OF SPIE

Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018

Ryszard S. Romaniuk
Maciej Linczuk
Editors

3–10 June 2018
Wilga, Poland

Organized by
Institute of Electronic Systems, Faculty of Electronics and Information Technologies, Warsaw University of Technology (Poland)

Sponsored by
PSP—Photonics Society of Poland • Committee of Electronics and Telecommunications, Polish Academy of Sciences • ARIES—Accelerator Research and Innovation for European Science and Society (CERN, EU H2020) • PKOpto—Polish Committee of Optoelectronics of SEP—The Association of Polish Electrical Engineers • EuroFusion Collaboration • EuroFusion Poland

Published by
SPIE

Volume 10808

Part Two of Three Parts

Proceedings of SPIE 0277-786X, V. 10808

The papers in this volume were part of the technical conference cited on the cover and title page. Papers were selected and subject to review by the editors and conference program committee. Some conference presentations may not be available for publication. Additional papers and presentation recordings may be available online in the SPIE Digital Library at SPIEDigitalLibrary.org.

The papers reflect the work and thoughts of the authors and are published herein as submitted. The publisher is not responsible for the validity of the information or for any outcomes resulting from reliance thereon.

Please use the following format to cite material from these proceedings:
Author(s), "Title of Paper," in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018*, edited by Ryszard S. Romaniuk, Maciej Linczuk, Proceedings of SPIE Vol. 10808 (SPIE, Bellingham, WA, 2018) Seven-digit Article CID Number.

ISSN: 0277-786X
ISSN: 1996-756X (electronic)

ISBN: 9781510622036
ISBN: 9781510622043 (electronic)

Published by
SPIE
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone +1 360 676 3290 (Pacific Time) · Fax +1 360 647 1445
SPIE.org

Printed in the United States of America.

Publication of record for individual papers is online in the SPIE Digital Library.

SPIEDigitalLibrary.org

Paper Numbering: *Proceedings of SPIE* follow an e-First publication model. A unique citation identifier (CID) number is assigned to each article at the time of publication. Utilization of CIDs allows articles to be fully citable as soon as they are published online, and connects the same identifier to all online and print versions of the publication. SPIE uses a seven-digit CID article numbering system structured as follows:

- The first five digits correspond to the SPIE volume number.
- The last two digits indicate publication order within the volume using a Base 36 numbering system employing both numerals and letters. These two-number sets start with 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B … 0Z, followed by 10-1Z, 20-2Z, etc. The CID Number appears on each page of the manuscript.

Part Two

Solution of travelling salesman problem applied to Wireless Sensor Networks (WSN) through the MST and B&B methods

Jonathan E. Tito[a], Marco E. Yacelga[a], Martha C. Paredes[a],
Andres J. Utreras[a], Waldemar Wójcik[b], Olga Ussatova[c]
[a]Escuela Politécnica Nacional, Ladrón de Guevara E11-253, Quito, Ecuador; [b]Lublin University of Technology, 38A Nadbystrzycka Str., Lublin, Poland; [c]Al-Farabi Kazakh National University, 71 Al-Farabi, Almaty, Kazakhstan

ABSTRACT

During this investigation, Traveling Salesman Problem or TSP is applied in a Wireless Sensor Network (WSN), through a free simulator named Castalia and programming codes on JAVA and GNU/Linux Scripting in order to implement two methods for solving the TSP. First method, consist of Minimum Spanning Tree (MST) with the 2-opt algorithm and the second one is Branch and Bound (B&B) method related to the Held-Karp lower bound. Likewise, the Prim, Borůvka and Kruskal algorithms will be compared in order to determine, which of them solves the MST problem in less time, through the simulator which defines two scenarios for three models of motas: TelosB, Imote2, and Zolertia. Finally, some parameters will be also compared, such as throughput and energy consumption for each scenario, node model and solving method of the TSP, and conclude what is the best method that could be applied to a WSN.

Keywords: wireless sensor networks, embedded systems, wireless communications, internet of things, free software, travelling salesman problem, graph schema theory, java.

1. INTRODUCTION

WSNs have become ubiquitous and are used in a wide range of applications[1], where their importance is reflected by their great number of applications due to many parameters that can be measured by sensors. According to[2], the following projects can be mentioned: Wisevine (gathering of data of interest in vineyards through WSN), Great Duck Island (Underground bird´s nests monitoring through WSN), Vital Sign (Vital signs monitoring through WSN). TSP is maybe the combinatory optimization problem most studied during the history, and this problem consists of a traveling salesman which must move towards all the cities once until return to the starting city of its journey.

However, there is a conditioning of using the shortest possible path because there can be several traveling salesmen or different metrics can be considered, which extend practical applications in real life. The parameters measured by the WSN nodes must be collected and transported depending on the application. According to[3,4], it is possible to reduce the power consumption of the sensor nodes and, therefore, extend the WSN lifetime by reducing the number and size of the communicated data. Equally, establish minimum cost paths for data interchange reduces the number and size of communicated data. One of this outlines for finding a minimum cost path is defined with the resolution of TSP. The solving of TSP could be used for the efficient diffusion of information in a data network, which be complemented with applications that make use of diffusion, through one of the mentioned examples in[5,6] called Direct Diffusion (DD).

2. THE TSP FROM THE GRAPH SCHEMA THEORY PERSPECTIVE

There are algorithms which have generated protocols that possess a mathematic origin based on the graph schema theory. For example, the Dijkstra or Bellaman-Forman´s algorithms are essential part of routing protocols, such as OSPF or RIP respectively[7]. This occurs because the information streams can be mathematically modeled from the graph theory perspective, where relevant calculations establish the path that information streams must follow to accomplish certain defined conditions.

*jhonathan.e.tito.o@gmail.com

Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018, edited by Ryszard S. Romaniuk, Maciej Linczuk, Proc. of SPIE Vol. 10808, 108082F
 · doi: 10.1117/12.2501579

2.1 Graph schema theory definitions.

According to Vitaly Voloshin[7] a graph, is a set of lines and dots which connects some pairs of dots, where graph theory consider dots as nodes or vertices and lines as edges. In[8] nodes or vertices are processing beings or structures that contain some type of information, the edges, on the other hand, represent connections or relationships between the nodes. For example, one of this structures could be a set of radioactive waste containers, whose information of interest is the amount of Sieverts (a unit of measure of radioactivity level) received at a certain distance, the relationships or edges would come to be the measured Sieverts on the workplaces of the employees near this container.

A road is a kind of graph in which all the vertices can be arranged (from left to right) in such a way that there exist precisely one edge connecting two consecutive vertices without other existing edges, instead, a cycle is a path that ends in the same node where it began. If the path of this cycle goes through all the nodes in the graph it is called tour[9]. It is well known that a graph is connected when it can be reached from any node to any other one by a path. If the previous case is not met, then it is concluded that the graph is not connected, but it can be divided into connected components[9]. The concept of a connected graph is usually applied to non-directed graphs, if there is an edge for each pair of nodes in this graphs then the graph is called complete[10].

2.2 About the complexity with graph problems

According to Ernesto Coto in[11], there is a great variety of problems related with graphs and variety of algorithms for processing graphs. However, not every graph problem is simple to solve, and in many cases it is not easy to determine how difficult will it be to solve. For that reason, the author makes a classification according to the difficulty and cites some of the most representative problems in graph theory.

An easy graph processing problem is one that can be solved using an efficient program. Generally, it can be stablished that the problem is easy if we can develop brute force algorithms which allow us to find the desired solution on a linear execution time (for the worst case), or limited by a low degree polynomial in the number of nodes or the number of edges. Next, a treatable graph processing problem occurs when it is known an algorithm which guarantees that its requirements in time and space are limited by a polynomial function in the size of the graph (number of nodes plus number of edges), thus, any easy problem is treatable. On other side, an intractable graph processing problem is characterized when it is not known some algorithm that guarantees the solution of the problem in a reasonable amount of time, in fact if a brute force method is used to try all the possibilities for calculating the solution, there would exist many possibilities to be considered.

2.3 TSP Definition

Finding a Hamiltonian Tour for any graph is an intractable problem because the only known solution for the general problem consists of finding the solution between all the possibilities of cycles in the graph. As the great majority of intractable problems, finding a Hamiltonian tour is a Hard-NP problem while other problems that are variations of the Hamiltonian tour inherit the same difficulty level, so this is the problem of the traveling salesman (TSP).

We extracted its mathematic definition from[12,13]:

$$G = (X,E) \tag{1}$$

Where G is a graph (directed or non-directed), X is the set of vertices or nodes which confirm the graph G, and E is the set of edges which confirm the graph G. For each edge $e \in E$, exists a cost (weight) C_{ei}, which is prescribed. Then the TSP consists on finding a Hamiltonian tour on G such that the sum of all the costs of the edges in the tour is the smallest possible.

Without losing generality, it can be assumed that G is a complete graph (this is a condition for the TSP being solvable on G), in other way, the missing edges could be replaced with edges of a higher cost, being:

$$E = \{e_1, e_2, \ldots. e_m\} \tag{2}$$

Where e_1, e_2, e_3 and e_m are the edges that belong to the graph G. Also, being a matrix of costs, a matrix of distances, a matrix of weights or matrix of adjacencies:

$$C = (C_{ij})_{nn} \tag{3}$$

Where C is the matrix of costs, then the *ij*-nth input C_{ij} corresponds to the cost of the edge that connects the vertex *i* and the vertex *j* in *G*.

Also, depending on the nature of the costs matrix (equivalent to the nature of G), the TSP is divided into two classes. If C is symmetrical (that means, G is non-directed) then the TSP is called STSP or Symmetric Traveling Salesman Problem. If C is not necessarily symmetrical (equivalent to the graph C being directed) then it is called ATSP or Asymmetric Traveling Salesman Problem.

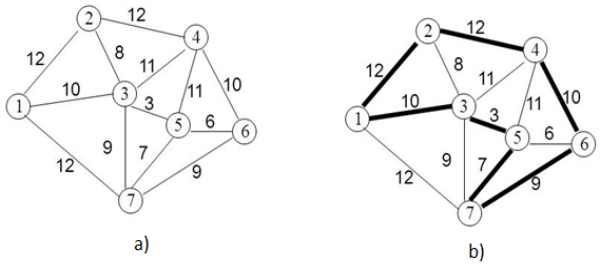

Figure 1. In a) a non-directed graph and in b) its TSP solved is highlighted in bold[14].

As it can be appreciated in the Fig. 1, solving the TSP for a reduced number of nodes can be done without difficulty evaluating all the alternatives manually, but when the number of nodes in the problem grows it is necessary to go to one of the approximated methods or some of the exact methods for obtaining an answer in a reasonable amount of time.

3. SIMULATING THE TSP ON A WSN

3.1 Definitions about WSN

A wireless sensor network according to[3] is an autonomous, self-organized sensor node system, distributed over a given area, to measure data of its surroundings and group them to a source node for their future processing. Once the sensor nodes are deployed, they are able to capture data of some physical magnitude, such as temperature, atmospheric pressure or some contaminant concentration. The sensor´s detections are usually reported to a central server, and also called source node, where the lectures will be later processed according to the requirements of the application.

Generally, the sensor nodes are spread on an area called sensor field, then the sensor nodes collect and transmit wirelessly the sensed data to the source node (also known as Base Station, Gateway or Access Point, besides of central node or sink as mentioned in[2]). Finally, the source node will send the sensed data to the users through other communication links (for example, a satellite link). A source node can be any of a wide variety of devices, for example, a laptop, a PDA (Personal Digital Assistant), an earth station, etc.

Depending on the application of the WSN, it can be more than a source node, and[15,16] stablishes a study about the topic of the WSN with multiple source nodes.

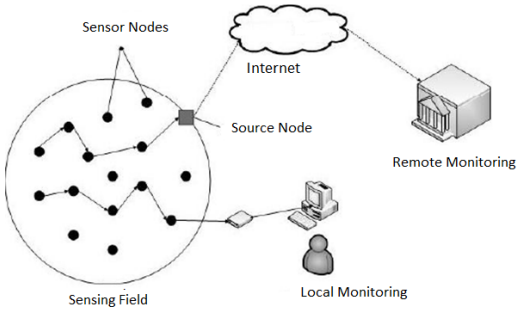

Figure 2. Structure of a WSN.

According to[17], a sensor node is formed by the sensing, processing, communications and power subsystems. The designer has many options to decide how to build and dispose all these subsystems together into a programmable node.

On Figure 3, starting from the upper left corner it can be seen a light sensor which sends data to the processing subsystem. Next, it can be observed two sensors, one of acceleration and one of temperature connected to an ADC (Analog-to-digital converter). Some sensors have its own built-in ADC, which can be communicated directly to the processor, and the processed data are sent due to the communications subsystem.

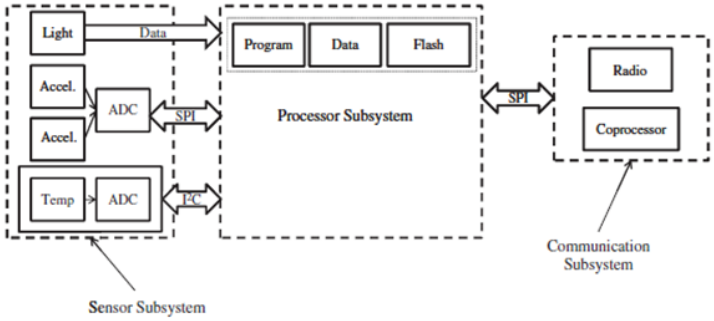

Figure 3. Architecture of a wireless sensor node.

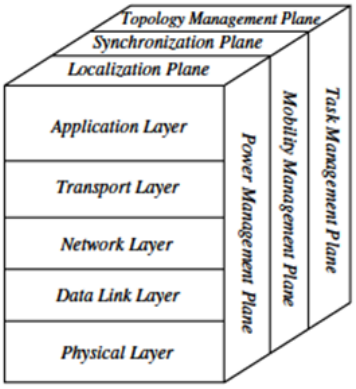

Figure 4. Architecture of a wireless sensor node[18].

The physical layer embraces techniques of modulation, transmission, and reception. Due to the noise of the environment and motion of nodes, data link layer is the responsibility of ensuring reliable communications through error control techniques and channel access management through MAC in order to minimize the collision for diffusion with neighbor nodes. In fact, depending on the sensing tasks, different kinds of application software can be built and used on the application layer. On other hand, the network layer is in charge of routing of data that are provided by the transport layer and transport layer helps to maintain the flow of data if it is required by the WSN.

Besides, power, mobility, and task management planes monitor the power, movement, and distribution of tasks between the sensor nodes. These planes help the sensor nodes to coordinate the tasks of sensing and minimize the average energy consumption. The power management plane is committed to managing the energy consumption of the node. For example, the sensor node can turn off its receptor after receiving a message from one of its neighbors to avoid having duplicated messages. Also, when the power level of a sensor node is low, sensor warns their neighbors that it cannot participate in the routing of messages, in fact, the leftover power is reserved for the sensing. The mobility management plane detects and registers the movements of the sensor nodes, in the way that a return path to the user is always maintained, and the sensor nodes can keep track of their neighbors. Through the knowledge of this neighbors sensor nodes, the sensors can balance their power and tasks, where the task management plane balances and plans the sensing tasks given to a specific region. In other words, not all sensor nodes in that region are required to do sensing tasks at the same time.

3.2 TSP applied to the WSN

When the problem of the travelling salesman is applied to a WSN, the cities become sensor nodes, the travelling salesman comes to be a package of data, the city of origin of the path comes to be the sink node and the costs of the paths

to be traveled to solve the problem could be quantified at evaluating the quality of the link according to the parameters of the application that the WSN considers critical (energy consumption, BER, etc).

Due to TSP is only applied on complete graphs, this implies that the WSN must have the greatest connectivity that it can quantify the majority of their links, otherwise, TSP will end taking an infinite cost link (a link non-existent in the WSN), this is because the condition must be met that the tour must pass through every node once.

3.3 Castalia Simulator

It was developed by the NICTA (National Information Communication Technology Australia) to simulate Wireless Sensor Networks (WSN), Body Area Networks (BAN) and in general of low power embedded devices networks. It is used by researchers and developers to probe their distributed algorithms and protocols in channel models and realistic wireless radio, with an objective node behavior specially related in what respects to the environment access[19]. Castalia is a generic open source tool that cannot test codes or hardware for replying on a specific sensor node model[20].

Castalia simulates all the components of real sensor nodes, on Fig. 5, where the arrows of continuous line means there exists a pass of messages, while the dotted arrows means there only exists a simple function calling.

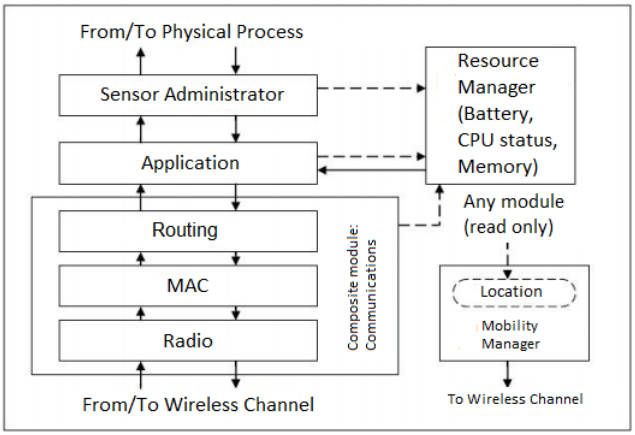

Figure 5. Modules which conforms the node on the Castalia simulator[20].

Application Module: It is the module that the user can commonly change, usually creating a new module in order to implement a new algorithm.

Communication Modules: MAC and Enrouting. – They will be usually modified by the user in order to implement a new protocol.

Mobility Manager Module: The user generally will modify this module to create a new mobility pattern.

3.4 Mota models

Table 1. Summary of the technical data of used mota models[18-21]

Name	Cpu speed [MHz]	Mem. Prog.	RAM [kB]	Radio freq. [GHz]	Tx speed [kbps]	Chip	Tx Power
TelosB	16	48 kB	10	2,4	250	CC2420	0 dBm
Imote2	13-416	32 MB	256	2,4	250	CC2420	0 dBm
Zolertia	16	16 MB	96	2,4	250	CC2420	0 dBm

Mota models which work with the CC2420 chip were selected because this is the default chip supported by the simulator Castalia. According to Table I, the transmission parameters are identical, while the storage and information processing parameters are the ones that differentiate the mota models.

3.5 Simulation Settings

Setting A - The sensor nodes are uniformly placed and distributed randomly, with the source node placed on the left most of the simulation setting.

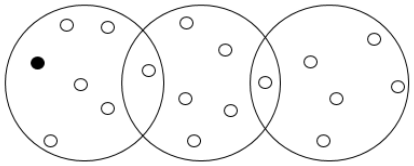

Figure 6. Example of the placing of the sensor nodes in setting "A", the source node is highlighted in black.

Setting B - The sensor nodes are uniformly placed and distributed randomly, with the source node placed on the center of the simulation setting.

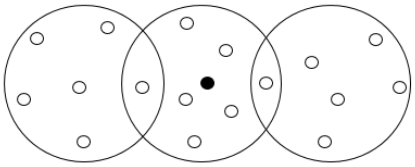

Figure 7. Example of the disposition of the sensor nodes in setting "B", with the source node highlighted in black.

4. SOLVING OF THE TSP: HEURISTIC METHOD

At the Network Design and Optimization class given at Melbourne University[22], one of the heuristic methods exposed for solving TSP was founded in the MST or Minimum Spanning Tree. In an article published by the University of Buenos Aires in[23], the heuristic method is defined in computer sciences as a kind of algorithm that, ignoring certain information, gets rid of great part of the effort that could be required to read the data and perform calculations with them. On the other hand, the heuristic solution is independent of the ignored information, and is not affected by the changes of that information. Ideally, it ignores information that results too expensive to collect or maintain or which contributes in minor degree to the precision of the solution. Being the TSP an NP problem, one of the flanks from which the solution is attacked consists of a set of heuristic techniques, from which the solving of the MST is taken as one of them.

4.1 Minimum Spanning Tree

A spanning tree of a connected graph consists of a subgraph that contains all the nodes in the graph and has no cycles. The minimum spanning tree of a non-directed heavy graph is the tree of expansion which weight (the sum of the weights of all of its edges) is not greater than the weight of another spanning tree. In fact, the problem of finding the MST of an arbitrary non-directed has plenty of important applications and it disposes of (since 1920) of algorithms to find it; however, the efficiency of the implementations varies widely and the researchers are still looking for better methods[24]

4.2 Comparison of Prim, Borůvka, Kruskal algorithms to obtain the MST

The Prim algorithm is the easiest to implement and thus is the less complex because it does not need structures or complex data like Borůvka, which uses lists or Kruskal that uses union-find. According to[25] Zolertia[26], TelosB[27] and Imote2 that support the C programming language, the usage of an algorithm that requires complex data structures could complicate the development of the solution. The difficulty will depend on the cleverness and experience of the programmer, because it is possible to develop data structures in C, as it can be seen in[28].

The execution complexity of the algorithm and its complexity on time resulted to be inversely proportional for the case of the Prim algorithm, because if the "heap" data structure is used, its complexity on time is the same as the Borůvka and Kruskal algorithms. That is to say: $O_{(m \log n)}$ where m is the number of iterations and n is the number of vertices. With respect to the faster execution algorithm[29], stablishes an analysis of the three algorithms (using the Prim algorithm in its basic version) and the results obtained are summarized in Fig. 8.

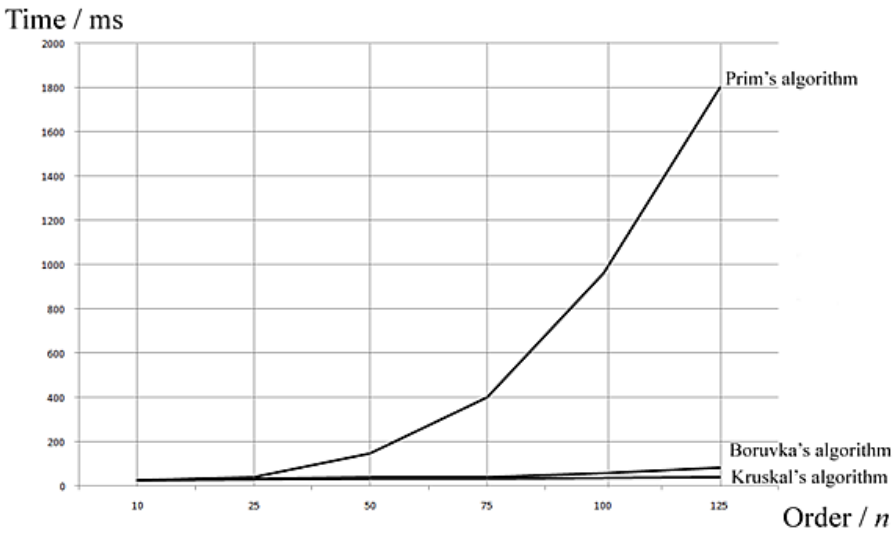

Figure 8. Number of nodes vs Execution time for the Prim (basic version), Borůvka and Kruskal algorithms[29].

According to Fig. 8, when the number of nodes is less than 25 the tree algorithms have a similar behavior. However, the Prim algorithm has an advantage in its basic version for being of lesser complexity. Hence, when defining the use of an 18 sensor nodes WSN in this work, it is used the Prim algorithm in its basic version for being the best that adapts to the proposed implementation requirements and with this algorithm it will proceed to solve the TSP by heuristic method.

4.3 MST and 2-opt for solving the TSP

For Dallaali in[22], two ways can be taken to solve the TSP heuristically:

- Use of the MST as starting point.
- Use of the MST as a "good" feasible solution.

After going through all the possible feasible permutations:

- The permutations must turn the MST into a tour of the original graph (G).
- The restrictions of the subtour must be always checked, in order ensure that the solutions are feasible. In other words, the restrictions of the subtour must be satisfied.

The MST is taken as starting point, and then the algorithm 2-opt is applied. This algorithm stablishes that, if a Hamiltonian cycle crosses itself, it can be easily shorted, so it is enough with eliminating the two edges that cross themselves and reconnecting the resulting paths through edges that do not cross each other. As a result, a final cycle is obtained which is smaller than the starting one. A 2-opt movement consists of eliminating two edges, for later connecting the four vertices that remained in a distinct way, obtaining a new route.

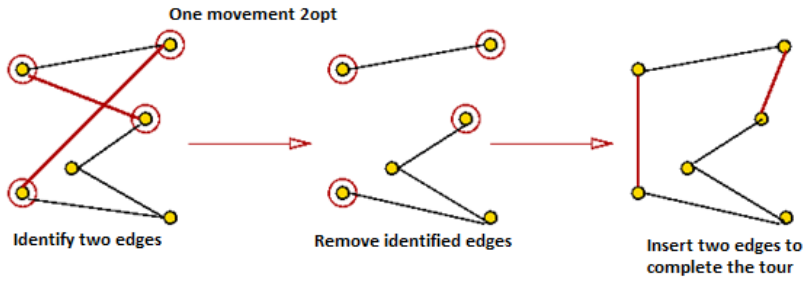

Figure 9. 2-opt algorithm for making permutations, looking for the optimal tour[31].

In Depth Last[30], is proved that for a graph G, the tour which solves the TSP approximately in G, is less or equal to the double of the total cost in G. For that reason, taking the MST as starting point, it is necessary to apply the 2-opt algorithm to validate the building of a tour that solves the MST and it is taken as solution the first value of the total cost of the tour that is less to the cost of going through the MST twice.

5. SOLVING OF TSP: EXACT METHOD

According to[32], Branch and Bound (B&B) is a well-known searching method that has its origins in TSP. B&B is organized in such way that it does an exhaustive searching of the best solution for a specified group. Each branch step divides the searching space into one or more subgroups, on an attempt at creating subproblems that can be easier to solve than the original one. Through the repetitive branching steps, a collection of subproblems that are needed to solve are created, where each one of them defined by a subgroup of tours that include certain edges and exclude certain others.

Before looking for a problem and possibly start dividing it, a limit is computed for the cost of the tours.

The purpose of the dimension step consist of trying to avoid an unsuccessful searching of a sub problem that not contains a better solution than one already discovered. In addition, the idea analyze if the limit is bigger than or equal to the cost of a tour already found, so it is possible to discard the sub problem without any danger of losing the best tour.

5.1 The Held – Karp´s lower limit

According to a publication of the George Washington University in[31], the best optimal known algorithm, is the Held – Karp algorithm. This is proved with the stated in[33], where it is affirmed that Held – Karp´s lower limit is used to judge the performance of any new proposed heuristic solution to solve the TSP. Due to this precision level that presents the relaxation of the TSP through the Held –Karp´s lower limit, this lower limit has been chosen as the lower bound for the solution of the TSP using the Branch and Bound method.

When considering a graph which vertices are numbered from zero to "n", in such way that it defines a 1-tree (one tree) as a subgraph formed by a vertex spanning tree from one to "n" and the two minor cost edges that come from the vertex zero. Is important to note that each tour (including the optimal) is a 1 – tree. The minimal 1-tree is defined as the lesser cost 1 – tree from between all the possible 1 – tree for the graph.

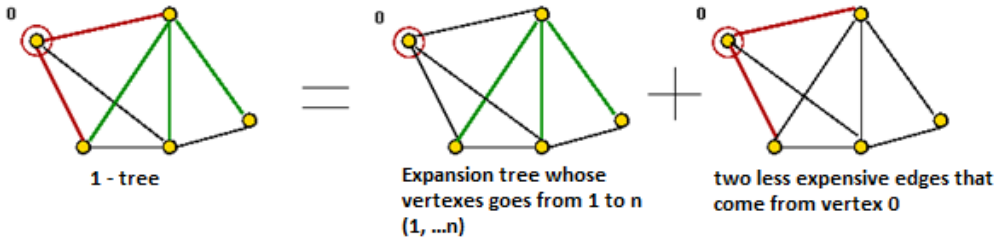

Figure 10. Summary of the making of a 1 – tree[31].

The idea of the Held – Karp´s lower limit consists in altering the original graph (G), in such way that the modified graph (G') generates different minimum 1-tree than the original graph. The way in which the original graph (G) is modified to obtain the modified graph (G') is realized through the addition of weights at the vertices of the original graph. The weights that exist to be added are calculated repeatedly using the subgradient method, in such way that this lower limit is being refined.

On a first approximation, after obtained the cost matrix corresponding to the graph formed by the nodes of the WSN to be simulated, the next step is to find the total cost of the minimal 1-tree without using Held-Karp (that is to say, without the nodes having associated weights). When the algorithm´s variables are initialized, the optimal lower limit by default is infinite. The objective of finding the minimal 1-tree, without using Held-Karp, it to establish the first approximated lower limit from which start to work, because a minimal 1-tree is by default a lower limit in the solving of the TSP. It´s worth saying that besides not working with Held-Karp at this point, the starting of the Branch and Bound algorithm is not started either at the moment.

6. SIMULATION AND RESULTS

All the simulation process is controlled and visualized from a Bash Script that works in Ubuntu. This script was developed in such way that it avoids the introduction of foreign characters from the user because menu screens are shown and always that the process is canceled, this returns to the starting menu. Three parameters must be chosen: mota model, the simulation setting and the TSP solving method in the WSN.

This process has been divided into three threads that are executed in order that the first and third thread imply modified code from the simulator in order to read the results of the TSP solving, which is implemented in the second thread through JAVA applications.

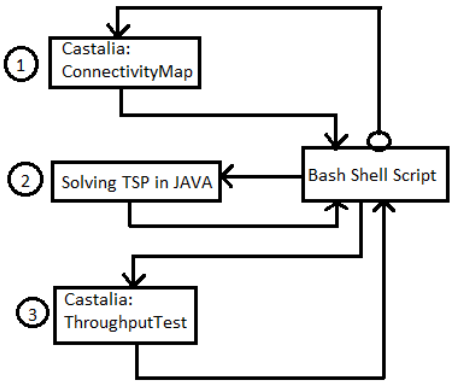

Figure 11. Summary of the implemented subprocess.

The Castalia simulator uses omnetpp.ini to get all the parameters ready to run the simulation, when parameters are chosen in the Bash Shell Script, the script will run the configuration with the mota model and setting specified.

For the mota models the size of the RAM memory, size of the flash memory and initial energy are configured in the simulation, while for setting "A" the sink node is placed in the left middle of the sensing area, this area is 50 meters by 50 meters and for setting "B" the sink node is placed in the center of the sensing area. For setting "A" and "B" all nodes are uniform distributed in the sensing area. By default each node is programmed to transmit 100 packets in a single time interval (timeslot), so that there are no collisions with other nodes. A node (when not transmitting) constantly listens to incoming packets, and when a packet is received by the node in question, it increments the counter of received packets by identifying the transmitting node.

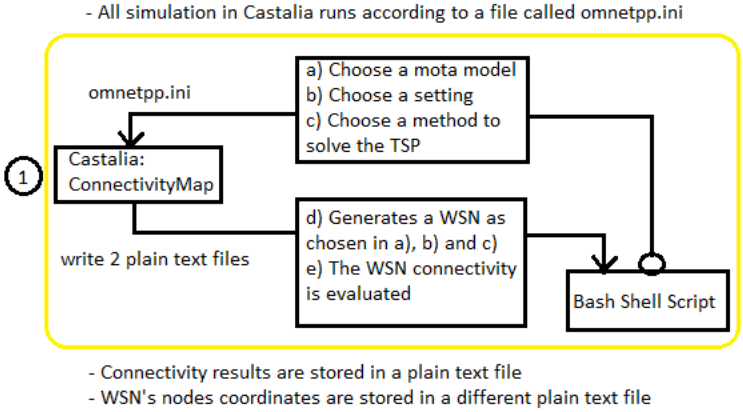

Figure 12. Structure of ConnectivityMap.

The source code of the Castalia *ConnectivityMap* application was modified to write two plain text files, where the logic of this application is in the file ConnnectivityMap.cc. The first modification consist of getting the costs matrix as the number of packets received successfully from each node to each node.

When the simulation ends, it means that simulation arrives to the final node (17th node), the cost matrix is written in the plain text file "ArchPlanoMatCostsCastalia_jtito.txt" and then the coordinates of all the eighteen nodes in the simulations are written to the file "ArchPlanoCoordCastalia_jtito.txt"

Thanks to the plain text file "*ArchPlanoMatCostsCastalia_jtito.txt*" the Castalia simulator can communicate the results of their simulation to the java programs, and then java programs generate a file which contains the route that solves the TSP in the WSN.

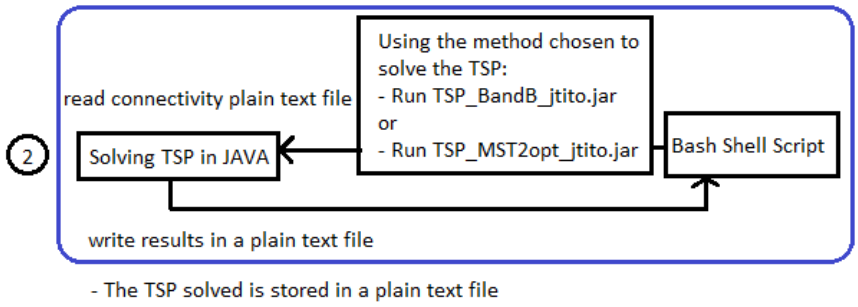

Figure 13. Structure of Solving TSP in JAVA.

For the implementation of the TSP resolution method by the Branch and Bound algorithm applying the lower limit of Held-Karp two equations are primary used:

$$\pi_I^{(m+1)} = \pi_I^{(m)} + t^{(m)}(d_i - 2) \tag{4}$$

Where each weight of each vertex on the graph representing the WSN is represented by π, at each iteration the present value is represented by m an next value is represented by $m+1$, while di represents the grade of the vertex. Hence, the weights for vertices with a minimum 1 - tree of degree greater than 2 are increased and the weights for the vertices with a minimum 1 - tree of degree greater than 2 are decremented. Therefore, through iterations, the aim is to force the minimum 1 - tree to be as close as possible to a tour.

In the program a node object is partial solution of the Branch and Bound method. All the iterations results are ordered in a priority queue and are filtered by the following code till the optimal solutions is discovered (or till is determined that the TSP has no solutions).

Finally the results of solving the TSP are stored in a text plain file to be used later for the third subprocess. For the implementation of the TSP resolution method by MST and 2-opt method, the MST is solved by using the simple Prim's algorithm through the method "*solveMSTPrimParaTSP*".

Then, the results of solving the TSP are stored in a text plain file to be used later for the third subprocess, and like a static route table, the results of solving the TSP are simulated in the *ThroughputTest* application in Castalia.

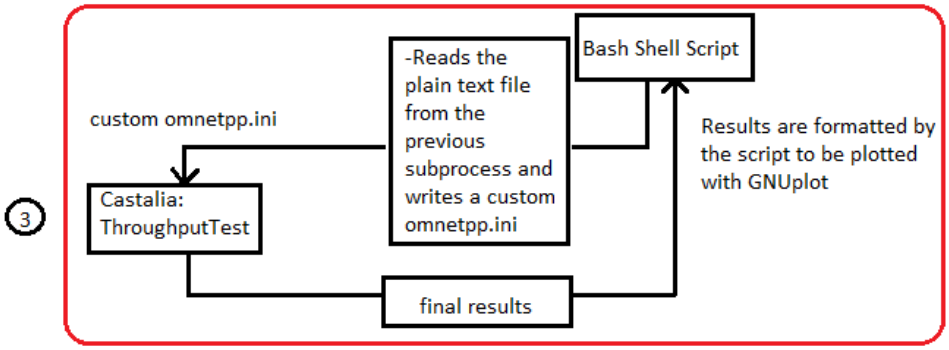

Figure 14. Simulation of ThroughputTest.

Finally, Castalia is able to tabulate the data and show them statistically:

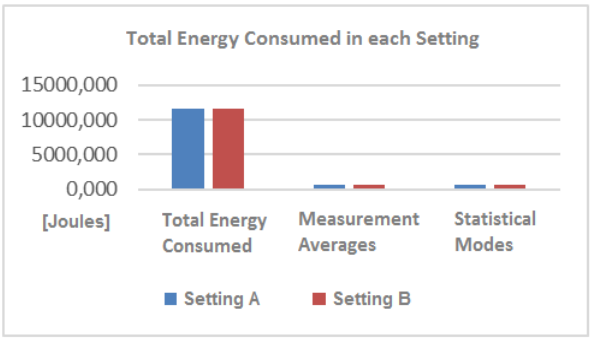

Figure 15. Total Energy Consumed by the WSN (Setting).

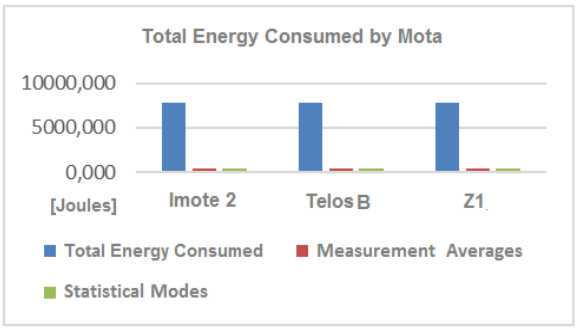

Figure 16. Total Energy Consumed by the WSN (Mota).

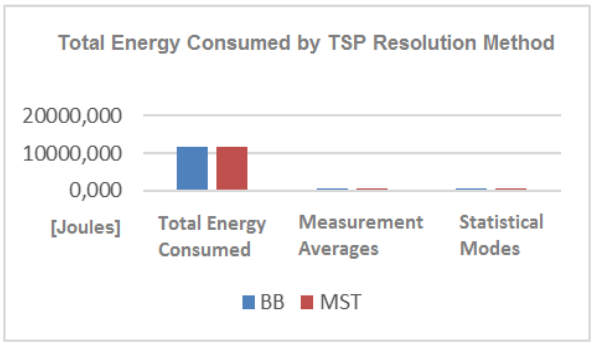

Figure 17. Total Energy Consumed by the WSN (Resolution method of the TSP).

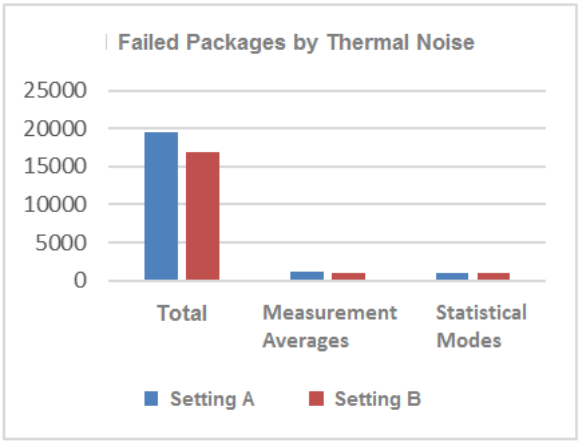

Figure 18. Failed packages due to thermical noise in the WSN.

7. CONCLUSIONS

Based on the analysis, it is concluded that the best method (of the two proposed methods) to solve the TSP in a WSN corresponds to the Branch and Bound method. Indeed, B & B method that uses the lower limit of Held-Karp takes 1 second to solve the TSP, while the approximate method by the MST takes 12 seconds to carry out the same task. Therefore, although B & B is more complex to implement, it is also faster, although this will depend on the programming facilities of the specific WSN. Also, with the resolution of the TSP it is verified that the minimization of the energy consumption in the WSN is fulfilled when observing that each node consumes about 53 Joules in carrying out the implementation of the TSP for all cases of WSNs raised. Finally, Castalia stands out among WSN simulators because it has a very detailed documentation, and for being completely open source it can be modified at convenience as required under terms related to free software licenses, so there is an active community raising / answering concerns about the simulator.

REFERENCES

[1] Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M. and Polakos, P., "Wireless Sensor Network Virtualization: Early Architecture and Research Perspectives," IEEE Network, 104 (2015).

[2] Ortiz, A. M., "Técnicas de enrutamiento inteligente para redes de sensores inalámbricas," Albacete: Universidad de Castilla-La Mancha, 38-37 (2011).

[3] Farrugia, L., [Computer Science, Technology and Applications: Wireless Sensor Networks], Nova Science Publishers, Hauppauge (2011).
[4] Kravchuk, S., Minochkin, D., Omiotek, Z., Bainazarov, U., Weryńska-Bieniasz, R. and Iskakova, A., "Cloud-based mobility management in heterogeneous wireless networks," Proc. SPIE 10445, (2017).
[5] Jumira, O. and Zeadally, S., [FOCUS Series, Volume 1: Energy Efficiency in Wireless Networks], John Wiley & Sons, Somerset (2013)
[6] Hart, C., [Graph Theory Topics in Computer Networking], Department of Computer and Mathematical Sciences (2018).
[7] Voloshin, V., [Introduction to graph theory], Nova, New York, 1-7 (2009).
[8] Coto, E., [Algoritmos Básicos de Grafos], Universidad Central de Venezuela, Caracas (2003).
[9] Voloshin, V., [Introduction to graph theory], Nova, New York, 14-15 (2009).
[10] Moşoi, A., "Difference between connected vs strongly connected vs complete graphs," 25 November 2009. http://mathoverflow.net/questions/6833/difference-between-connected-vs-strongly-connected-vs-complete-graphs
[11] Coto, E., [Algoritmos Básicos de Grafos], Universidad Central de Venezuela, Caracas, 6-9 (2013).
[12] Jiménez, A., "Graph Theory Topics in Computer Networking," https://www.xatakaciencia.com/matematicas/p-versus-np-nunca-lo-entendiste (26 April 2018).
[13] Gutin, G. and Punnen, A., [The traveling salesman problem and its variations], Kluwer Academic Publisher, New York, 3-4 (2004).
[14] Yifu, L., "Traveling Salesman Problem (TSP)," http://slideplayer.com/slide/8783733/. (December 2015).
[15] Colesanti, U. and Santini, S., "A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator," Proc. ETH Zurich, 2 (2010).
[16] Chen, S., Coolbeth, M., Dinh, H., Kim, Y. A. and Wang, B., [Data Collection with Multiple Sinks in Wireless Sensor Networks], University of Connecticut, Connecticut (2009).
[17] Dargie, W. and Poellabauer, C. [Fundamentals of Wireless Sensor Networks], Wiley, Singapur, 47-51 (2010).
[18] Akyildiz, I. and Can Vuran, M., [Wireless Sensor Networks], Wiley, Singapur, 2-15 (2010).
[19] Herrera, D. C., [Evaluación de redes de sensores inalámbricos mediante el Simulador OMNeT++], Universidad Politécnica de Valencia, Valencia, 11-12 (2014).
[20] Boulis, A., [Castalia: A simulator for Wireless Sensor Networks and Body Area Networks], NICTA, Sydney (2011).
[21] Linares Arenas, J. M., [Simulación e implementación de una red de sensores inalámbrica multisalto para la medición de consumo energético en un edificio], Universidad Politécnica de Cataluña, Cataluña, 9-12 (2014).
[22] Dallaali, M. A, [Network Design and Optimization], The University of Melbourne, Melbourne (2013).
[23] Ramos, S., "Heurísticas y Problemas Combinatorios," Universidad de Buenos Aires (2007).
[24] Coto, E., [Algoritmos Básicos de Grafos], Universidad Central de Venezuela, Caracas, 18-20 (2003).
[25] "Mainpage: Contiki Lesson 0," http://zolertia.sourceforge.net/wiki/index.php/Mainpage:Contiki_Lesson_0 (5 February 2018).
[26] "TelosB," 30.11.2012, http://smartsantander.eu/wiki/index.php/Main/TelosB (5 February 2018).
[27] Lorincz, K., "IMote2 Installation Instructions," Intel Research, Berkeley, 14 June 2005, http://www.eecs.harvard.edu/~konrad/projects/imote2Camera/IMote2-Installation-Instructions.html (5 February 2018).
[28] Straub, J., [C Programming: Data structures and Algorithms], University of Washington, Washington. (2006).
[29] Podsechin, I., "EURAXESS 2008," http://www.euraxess.fi/Tiedostot /Tiedostot/Viksu/Viksu%202008/IgorPodsechin.pdf (5 February 2018).
[30] Paterson, A., "Performing 2-OPT Updates To TSP Solution Approximation In Java," 6 April 2016, http://depthlast.com/2016/04/06/performing-2-opt-updates-to-tsp-solution-approximation-in-java/ (5 February 2018).
[31] Simha, R., "The Traveling Salesman Problem (TSP)," https://www.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html (5 February 2018).
[32] Applegate, L. D., Bixby, E. R., Chvátal, V. and Cook, J. W., [The Traveling Salesman Problem, A Computational Study], Princeton University Press, New Jersey, 41-42 (2006).
[33] Davendra, D., [Traveling Salesman Problem, Theory and Applications], InTech, Rijeka, 1-17 (2010).

Compromising an IoT device based on Harvard-architecture microcontroller

Krzysztof Cabaj[a], Grzegorz Mazur*[a], Mateusz Nosek[a]
[a]Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warszawa, Poland

ABSTRACT

The paper describes the concept and implementation of an attack technique, targeting an Internet-connected device based on Arduino family board and modules with an ATmega microcontoller. Due to Harvard-like architecture of the microcontroller, the attack uses return-oriented programming principle, utilizing the pieces of firmware already contained in the memory of target device. We show that the routines present in the device are sufficient to convey a successful attack and change the device operation in the presence of buffer overflow backdoor to the firmware.

Keywords: embedded system, microcontroller, Harvard architecture, AVR architecture, IoT device, buffer overflow attack, return-oriented programming, gadgets

1. INTRODUCTION

Currently we can observe a rapid rise of so called IoT devices, which are used for smart home, agriculture, environmental monitoring, transportation or medical and healthcare, to mention few. Great popularity of the devices, low cost and popularity of electronics used for developing these devices allows easy, rapid development and deploying new ones. Moreover, this ease introduces new type of devices - Do-It-Yourself (DiY), which can be developed by almost anyone at home. One of the popular family of development boards used for this kind of project is called Arduino, which is used during our work on proof-of-concept exploit.

However, current studies reveal that these devices have very poor security, for example, lack of encryption, easily guessable management passwords and various vulnerabilities in the code. This is even greater problem if we realize that most of these devices are directly connected to Internet. Moreover, we should emphasized fact that nowadays user are familiar with general security for desktop computers, but how many users know that new box or knob with few LED diodes should be monitored, virus cleaned or upgraded? How many users know that their IoT device can be used for attacking other computers in the home or office or become a part of the botnet?

In this paper we presented our research concerning IoT security. In the following paragraphs we describe how starting with analysis of results gathered from Shodan service, we discovered vulnerable software, analyze it source code and using this information successfully perform attack on device in our testbed.

The remaining of the paper is organized as follows. Section 2 presents state of the art, concerning general attack techniques and these specially crafted for the IoT. Section 3 describes the principles of buffer overflow attack scenarios. The next section describes our target, and provides information how it was discovered. Section 5, presents details of stack buffer overflow attack utilizing Return Oriented Programming, prepared during our research. Section 6 describes prepared Proof-of-Concept exploit and confirms that it really works. The paper ends with conclusions and direction for future work.

* g.mazur@ii.pw.edu.pl; www.ii.pw.edu.pl

Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018,
edited by Ryszard S. Romaniuk, Maciej Linczuk, Proc. of SPIE Vol. 10808, 108082G
 · doi: 10.1117/12.2501580