GISOO: A Virtual Testbed for Wireless Networked Control Systems

BEHDAD AMINIAN

Master’s Degree Project
Stockholm, Sweden August 2012

XR-EE-RT 2012:021
GISOO: A Virtual Testbed for Wireless Networked Control Systems

BEHDAD AMINIAN

Master’s Thesis
Supervisor: José Araújo
Co-Supervisor: Fredrik Österlind
Examiner: Prof. Mikael Johansson

Stockholm November 8, 2012

XR-EE-RT 2012:021
Abstract

Networked control over wireless sensor and actuator networks is of growing importance in cyber-physical systems as in industrial process control and building management systems. The correct design of wireless communication solutions and control algorithms is therefore a major requirement in such systems. Simulators and emulators of wireless networked control systems (WNCSs) are tools that help the system designer to develop, optimize and validate solutions prior to the deployment in a real system. Without such tools the development and validation of complex systems would be highly costly, both financially and timely.

Considering the special characteristic of WNCSs, which is a complex combination of control systems and a wireless network, WNCSs’ simulators must be able to model and emulate/simulate both the control systems and wireless systems together.

In this thesis project, GISOO, a new simulation platform for WNCSs has been created. GISOO stands for Graphical Integration of Simulink and Cooja and is a hybrid simulation environment which simulates both the physical system to be controlled and the wireless devices and their networking behavior. Simulink, developed by Mathworks, is a tool for modeling, simulation and analysis of dynamic systems and is widely used by the control engineers. The Cooja network simulator, is able to emulate the operation of a real wireless device and its networking behavior. In this way, developers can set up simulations both to debug the developed software and the behavior of the system before running it in the target hardware. This specification makes GISOO a powerful virtual testbed for developing and evaluating novel communication protocols, control techniques and their interactions in large scale complex systems. We validate the correct operation of GISOO in the closed-loop control of a double tank system.
Acknowledgements

First of all, I would like to sincerely thank my supervisor José Araújo, who kindly provided me with the opportunity to work on this project in the dynamic and favorable atmosphere of the Automatic Control Laboratory. His great help and support was beyond my imagination and I am really happy that I could work with him. He has taught me much more than just technical points.

I also would like to thank Fredrik Österlind, my co-supervisor from Swedish Institute of Computer Science (SICS), who helped me with his valuable technical hints about Cooja. It was not possible to work on this project without his guidance and support.

In addition, I would like to appreciate my examiner, Prof. Mikael Johansson, for his valuable feedbacks on my work.

Moreover, I want to thank my colleague Anser Ahmed who helped me in Coupled Tanks experiments, and all of the people at Automatic Control Laboratory especially my friends in the research engineers’ room, Alireza, Altamash, Rashid and Navid for the wonderful moments that I had with them.

Finally I would like to express my gratitude to my family, to whom I owe every success that I have ever reached.
# Contents

## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNCS</td>
<td>1</td>
</tr>
<tr>
<td>NS-2</td>
<td>2</td>
</tr>
<tr>
<td>PNCS</td>
<td>3</td>
</tr>
<tr>
<td>Prowler</td>
<td>4</td>
</tr>
<tr>
<td>TrueTime</td>
<td>5</td>
</tr>
<tr>
<td>TmoteSky</td>
<td>6</td>
</tr>
<tr>
<td>Motes’</td>
<td>7</td>
</tr>
<tr>
<td>Operating</td>
<td>8</td>
</tr>
<tr>
<td>systems</td>
<td>9</td>
</tr>
<tr>
<td>Hybrid</td>
<td>10</td>
</tr>
<tr>
<td>simulators</td>
<td>11</td>
</tr>
<tr>
<td>MATLAB/NS-2</td>
<td>12</td>
</tr>
<tr>
<td>Modelica/NS-2</td>
<td>13</td>
</tr>
<tr>
<td>OPNET/SIMULINK</td>
<td>14</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Problem definition ........................................... 2
1.2 Previous work ................................................ 2
1.3 Thesis outline .............................................. 3

## 2 Wireless Network Control Systems

2.1 Networked Control Systems .................................. 6
2.2 Wireless Networked Control System ......................... 6
2.3 Wireless devices ............................................. 7
2.3.1 TmoteSky .................................................. 8
2.3.2 Motes’ Operating systems ................................ 9

## 3 WNCS Simulators

3.1 WNCSs’ simulators categories ............................... 12
3.2 Network simulators with extensions for control systems 12
3.2.1 NS-2 ..................................................... 12
3.3 Control system simulators with extensions for network simulation 13
3.3.1 Simulink network extensions and TrueTime ............. 14
3.3.2 Prowler ................................................. 15
3.3.3 WSN simulator by D.Andreu ............................. 15
3.3.4 Ptolemy II .............................................. 15
3.4 Hybrid simulators .......................................... 16
3.4.1 NMLab ................................................... 16
3.4.2 MATLAB/NS-2 integration by Soglo ........................ 16
3.4.3 Modelica/NS-2 .......................................... 16
3.4.4 OPNET/SIMULINK ....................................... 16
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GISOO</td>
<td>Graphical Integration of Simulink and Cooja. 6, 7, 23, 24, 27, 33–35, 37, 38, 41, 45, 49</td>
</tr>
<tr>
<td>MIG</td>
<td>Message Interface Generator. 40</td>
</tr>
<tr>
<td>NCS</td>
<td>Network Control System. 5, 9, 10, 16, 17, 19, 20, 24</td>
</tr>
<tr>
<td>NS-2</td>
<td>Network Simulator version 2. 6, 16, 19, 20, 33, 34</td>
</tr>
<tr>
<td>NSCSCController</td>
<td>Networked Sensing and Control System Controller. 16</td>
</tr>
<tr>
<td>NSCSPlant</td>
<td>Networked Sensing and Control System Plant. 16</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>Objective Modular Network Testbed in C++. 16</td>
</tr>
<tr>
<td>OPNET</td>
<td>Optimize Network Engineering Tools. 20</td>
</tr>
<tr>
<td>PIccSIM</td>
<td>Platform for integrated communications and control design, simulation, implementation and modeling. 6, 20, 21, 23, 28, 33, 34</td>
</tr>
<tr>
<td>SICS</td>
<td>Swedish Institute of Computer Science. 6, 13, 22</td>
</tr>
<tr>
<td>Tcl</td>
<td>Tool command language. 16</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol. 25, 26</td>
</tr>
<tr>
<td>TinyOS</td>
<td>Tiny Microthreading Operating System. 13, 21, 38, 40</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>Definition</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>TOSSIM</td>
<td>TinyOS Simulator</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>WNCS</td>
<td>Wireless Network Control System</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Although it is almost more than 100 years passed from the first radio transmission, the most significant growth in wireless communication has occurred during the recent decades. Now, wireless communication is much more trustworthy than before, and several different standards have been designed for it. Considerable improvements in the area in addition to interesting characteristics of wireless communications attract several attentions from the academia and the industry. In parallel with wireless communications, Network Control System (NCS) has improved and grown for several years being applied for different purposes even in very large scales such as industrial production plants, water and energy distribution networks or transportation. Need of node mobility and more flexibility in most of the control systems, lead them toward wireless networks.

Combination of wireless communications and control systems, known as "wireless automation" is considered as a revolutionary area in the automation industry. Wireless communications can lead to reducing cabling and improving flexibility of movements for controllable systems, by providing the possibility to locate wireless devices like sensors, controllers and actuators on mobile devices [23].

Such a beneficial aspect is enough for industrial investors and academic researchers to focus on this branch of knowledge to create and improve Wireless Network Control System (WNCS). Although, WNCSs have been applied in some industrial parts with low real time controlling requirements, extensive research is needed to expand the usage of this technology.

For research purposes, several experiments with different scenarios should be performed and repeated, while doing the real experiments with real hardware are difficult, time consuming, costly and not easy to be repeated under exactly the same conditions. To overcome such concerns, simulators and emulators are good replacements for real experiments and can virtually provide necessary devices and facilities and an environment for this regard.

Since WNCSs combines both wireless networks and control systems, creating a
simulator which can cover both sides properly is not easy. Most of the available simulators for WNCS are just a simulator for one of the control systems or wireless communication with an additional extension to cover the other part. These types of simulators are mostly simple and not appropriate for complicated scenarios, while integration of two separately developed advanced simulators for control and communication can result in convenient simulators for intricate scenarios.

This thesis aims to create a new simulation platform for WNCSs by integrating the Simulink, a simulation and model-based design tool from Mathworks and Cooja, a cross-level wireless sensor network simulator and wireless node emulator from Swedish Institute of Computer Science (SICS), which we call "GISOO". Graphical Integration of Simulink and Cooja (GISOO) is a new simulator for WNCS which is able to simulate the physical systems and wireless nodes’ communication in an integrated environment. Since Cooja is a wireless node emulator, actual code of wireless nodes is executed, which makes it possible to evaluate the real performance of wireless nodes’ program in the systems. Such attribute makes GISOO a unique simulator among others.

1.1 Problem definition

This project is mainly focused on the development of a new simulation environment, GISOO, by integrating the Simulink and Cooja, for which the main challenging requirements are provision of communication and synchronization. Such challenging requirements are strengthened considering that, Cooja is a mote emulator, not a simple wireless network simulator.

To evaluate the functionality of GISOO, a complete control scenario is simulated inside the environment and results has compared with the results of same experiment in real-world. Also the affect of imposed delay of each mote on the maximum number of motes in a control system with a specific sampling time is studied.

1.2 Previous work

So far, several different simulators for control systems and wireless communication have been developed. In the past years, the increasing attention to the WNCSs has raised many attempts to create different simulators for covering this area, but most of them are performed in the form of extensions to previous simulators. Recently, the requirement of advanced simulators for implementation of more complicated scenarios is more highlighted and some attempts towards creating these types of simulators including the hybrid ones have been shaped. One of latest ones, "Platform for integrated communications and control design, simulation, implementation and modeling (PIccSIM)" simulator, which has been partly used in this project, is an integration of Simulink and Network Simulator version 2 (NS-2) and has been
created at Helsinki University of Technology. Nevertheless, no project or study has been performed regarding the integration of the Simulink and Cooja. So, this project is the first attempt in this regard; and can be a basis for further advancements in the future.

1.3 Thesis outline

In this report, chapter 2 presents some basic information about Wireless Networked Control Systems. Besides, some common devices which can be used in WNCS are introduced and some of the available simulators for WNCS are reviewed.

Chapter 3 discusses the architecture of GISOO and the implementation of each part. Also, different parts of GISOO are introduced and finally the implementation will be evaluated.

In chapter 4 some experimental scenarios are explained. Besides the results of simulation of a Coupled Tank control system, as a complete WNCS, are presented.

The affect of communication delay on the maximum number of wireless nodes is studied and discussed in chapter 5.

And, chapter 6 encompasses explanation of the results of the project and potential future work for further advancements in GISOO.
Wireless Network Control Systems

This chapter provides basic information about Networked Control Systems and the usage of wireless communication for them. So section 2.1 and 2.2, focus on the structure of Networked Control Systems (NCS) and Wireless Networked Control Systems (WNCS) respectively. In section 2.3, some of the wireless devices that can be used in WNCS and their operating systems are introduced.
2.1 Networked Control Systems

"A control system is a device or set of devices to manage, command, direct or regulate the behavior of other devices." [46]. Significant improvements in communication networks, allows the control systems to exchange their information via networks to manage remote devices. This type of control systems are referred to Networked Control Systems (NCS).

In a feedback control systems, a control loop makes of a set of devices including:

- Sensors, which sense different parameters of the controlled environment
- Controllers, to manage the system by running the control algorithm.
- Actuators, which can affect the controlled environment.

Such set of devices together, makes a complete control loop to manage the controlled environment. Typically, controlled environments are known as plant dynamics. Figure 2.1 shows a basic graph of a feedback control system.

![Figure 2.1: A simple feedback control system.](image)

In the networked control systems, all the controlling parts, communicate to each other via networks. Among different types of communication mediums, wireless communication due to its interesting characteristics, attracts a lot of research and investment in academia and industry for enhancing NCS and creating WNCS.

2.2 Wireless Networked Control System

Wireless Networked Control System (WNCS), is a type of networked control systems that uses wireless medium for communication between controlling parts. Employing wireless communication in NCS introduces several advantages such as cabling reduction, easy and rapid deployment, increase of mobility and flexibility and improved freedom in placement of sensors, controller and actuators [43][21][30]. Figure 2.2 shows a general schema of a WNCS.
2.3. WIRELESS DEVICES

Different hardware or applications can be used in WNCS; in the following some of the related ones to this project are discussed.

2.3 Wireless devices

Sensor node or mote is a type of node in Wireless Sensor Network (WSN)s which contains different units for processing, data sensing and communication. In such a device, processing unit collects and processes the signals sensed by sensors and transmits data to the network [45]. Figure 2.3 shows the basic architecture of the motes.

![Basic architecture of a sensor node (Mote) [29].](image)

Being tiny, and having both low-cost with low power consumption are some of the most important challenges in design of motes. Nowadays they can be as
small as few millimeters or operate longer than a year with a number of regular small batteries [8]. These devices are used in control systems to perform wireless communication for information gathering from a plant, or act as a middle node to relay data. Considering their processing capabilities they are also able to act as small controllers. These mote can be found in different types and platforms but the most common types are: Tmote Sky[37], TelosB[27], MicaZ[26], IMote2[33], Cricket [36] and IRIS[44]. Figure 2.4 shows the picture of some of these motes.

![Figure 2.4: Four types of motes. From left to right: IRIS, MICAz, TelosB and Cricket.](image)

In this project TmoteSky has been used in all of the experiments and simulations, which is explained in the following.

### 2.3.1 TmoteSky

Tmote Sky is an ultra low power mote, for high data-rate sensor network applications. In addition to being programmable, this mote contains integrated sensors, radio, antenna and microcontroller. Low power consumption of the Tmote Sky is result of using the ultra low power TI MSP430 F1611 microcontroller. It also goes to sleep mode when it is not in use. This mote also has a Universal Serial Bus (USB) interface and is able to provide reliable wireless communication [37]. Figure 2.5 shows a Tmote Sky mote.

![Figure 2.5: General view of a Tmote Sky mote [37].](image)
2.3. WIRELESS DEVICES

2.3.2 Motes’ Operating systems

Different types of operating systems are available for motes such as MANTIS [11], Nano-RK [12], LiteOS [16], TinyOS [9] and Contiki [13], while the two latter ones which are the most popular are introduced in the following.

TinyOS

Tiny Microthreading Operating System (TinyOS) is an open source operating systems for low-power wireless devices. It has been written in the nesC programming language which is an optimization of the C languages for memory limits of sensor networks. But most of the supplementary software for this OS has been written in java. It is a result of collaboration between the University of California, Berkeley, Intel Research and Crossbow Technology [9]. In this project all of the real or simulated motes are using the TinyOS and their related programs are written in the nesC language.

Contiki

Contiki is an open source, highly portable and multi-tasking operating system for wireless sensor network devices and microcontrollers which have low amount of memory. It has been written in the C programming language, and developed at SICS [13][28].
This chapter presents an overview of available simulators for WNCS. The first section categorizes different types of simulators according to their structures; and the next sections, reviews each type of categorized simulators.
3.1 WNCSs’ simulators categories

Building a proper test bed for WNCS is usually costly and also running a practical experiment on a real test bed is both difficult and time consuming. It can be really challenging to run a real experiment when it is required to isolate some factors, and it become more problematic when repetition of an experiment with exactly the same situation is necessary. Finally, working with hardwires needs several settings and calibrations which should be repeated periodically. All of these factors show the importance and necessity of a proper simulators to study different aspects of WNCSs.

Since WNCSs are composed of both wireless communication and control systems, a simulator for such systems should be able to simulate both controlling and communication sides. Considering that there are different and separate simulators for network communication and control systems, the simulators for WNCSs or sensor networks can be divided to different categories [23]:

- Network simulators with an extension to simulate control systems.
- Control systems simulators with an extension to simulate network communication.
- Hybrid simulators which are combination of separate simulators for network communication and control systems.
- Node emulators.

3.2 Network simulators with extensions for control systems

So far several environments for simulation of network communications have been created, which can be used to simulate different communication mediums and variety of network protocols. NS-2 [3] and Objective Modular Network Testbed in C++ (OMNeT++) [17] are some of the common network simulators, where NS-2 has been widely used/extended to simulate network communication in different environments including WNCSs.

3.2.1 NS-2

NS-2 is an open source network simulator which has been developed since 1989. This simulator which is developed using the C++ language is a discreet-event and packet level simulator. It supports considerable range of protocols in different layers and mediums such as wired and wireless networks [3]. For instance, the wireless modeling in NS-2 can support signal propagation, fading models and radio model
with propagation time [23]. These specifications make NS-2 one of the most common network simulators. In addition NS-2 has been extended to simulate the control models inside itself to simulate NCSs and WNCSs. Some of the extensions and frameworks for control modeling in NS-2 are reviewed in the following.

Agent/plant extension

In order to support NCSs’ simulation in NS-2, different extensions such as agent/plant have been created. These extensions facilitate NS-2 to simulate dynamic systems and transmission of data packets between plants and controllers. The agent/plant is a type of NS-2 agents, which can act as an interface between physical system and network model. This agent can be used as a sensor, controller or an actuator in the simulation, and can simulate various types of plant such as continuous, discreet, linear and nonlinear. In order to simulate NCSs, Agent/Plant uses two functions in Tool command language (Tcl) code, which are Sysphy and Smplschd. Sysphy is used to simulate the physical system and Smplschd is employed to schedule sample invocation [18][19].

NSCSPlant and NSCSController

Networked Sensing and Control System Plant (NSCSPlant), and Networked Sensing and Control System Controller (NSCSController) are two NS-2 agents, based on agent/plant framework. NSCSPlant is an agent which is utilized to create several control systems to simulate different physical systems and can support adaptive sampling policies; while NSCSController, is used to simulate different controllers [18].

3.3 Control system simulators with extensions for network simulation

Different environments such as MATLAB [1], Simulink [7], Modelica [2] and NI LabVIEW [15] can be used to simulate control systems, while MATLAB and Simulink are widely used for WNCSs’ simulation.

Simulink is an environment for simulation and Model-Based Design for dynamic and embedded systems, which encompasses graphical interface with sets of block libraries to simulate different systems. Since Simulink is integrated with MATLAB, different facilities of MATLAB are usable inside it. A group of NCS simulators can be made by adding an extension to a control system simulator such as MATLAB or Simulink [7].

3.3.1 Simulink network extensions and TrueTime

To extend Simulink facilities, add it new functionalities to it, or create a novel simulation environment, new sets of blocks such as network simulation extensions,
can be created and added to Simulink library. Many different sets of extension blocks for Simulink have been created so far among which TrueTime is probably the most well-known Simulink network blocks that has been developed in the Lund University in Sweden [24]. TrueTime is employed to model the execution time of tasks and time of messages transmission. These times can be modeled as constant, random, or data dependent. TrueTime block library includes of computer and network blocks. User defined tasks such as I/O tasks or control algorithms are executed in computer blocks. The network block is event driven and is executed when a message enters and leaves the network. For a network block it is possible to specify the transmission rate, the medium access control protocol, and other transmission related parameter [24]. TrueTime can only model the physical and MAC layers [23]. Although some of the IEEE802.15.4 beacon-enabled mode protocols have been implemented in TrueTime, they have not been contrasted with theoretical models. Also since TrueTime is based on MATLAB-Simulink, it is not easy to add new functionalities to it [20]. Figure 3.1 Shows the TrueTime 2.0 block library.

![Figure 3.1: TrueTime 2.0 block library [24].](image)
3.3. CONTROL SYSTEM SIMULATORS WITH EXTENSIONS FOR NETWORK SIMULATION

3.3.2 Prowler

Prowler is a MATLAB based wireless network simulator which can simulate wireless distributed systems from application to the communication physical layer. Prowler is a generic simulation environment, but its current target is the Berkeley MICA mote running TinyOS [4]. While Prowler can simulate different layers, its MAC sublayer is very simplified and is not based on IEEE802.15.4 [20].

3.3.3 WSN simulator by D. Andreu

Another MATLAB based WSN simulator has recently developed at KTH Royal Institute of Technology in Sweden by David Andreu. This simulator which is based on IEEE802.15.4.PHY and MAC layer has been mainly developed to create an easily extendable simulation environment to generate reliable results contrasted with theoretical models. The GUI in this simulator which is based on TORSHE Graphedit Tool [42] makes it possible for the users to easily draw the network topology and run the simulation. It also contains a debugging tool which graphically shows the PHY and MAC event registration during the simulation [20]. Figure 3.2 shows the main control panel and Modified Graphedit Tool of this simulator.

![Figure 3.2: Simulator main control panel and Modified Graphedit Tool [20].](image)

3.3.4 Ptolemy II

Ptolemy II is an open source software framework for simulation of discrete events, especially in heterogeneous, hierarchical and synchronous systems [5][23]. It has been developed at UC Berkeley and has been extended for different purposes such as distributed detection with sensor networks, however it is no longer being developed [22].
3.4 Hybrid simulators

Although the discussed types of simulators are able to run both network part and control part in the same environment as an advantage, most of the extension parts, are not as powerful as the main simulator. Also, since the extension parts should be developed from scratch, they are usually simple and not very accurate. In contrast to such simulators, hybrid simulators, simulate the NCS in two different environments, while both simulators are advanced enough to be used in complicated scenarios.

3.4.1 NMLab

NMLab is one of the hybrid simulators for NCS, which is a combination of NS-2 for network simulation and MATLAB for modeling the control systems. NMLab combines flexible and powerful numerical operations with support of wide range of communication protocols, by means of which, it can simulate the NCS with a high level of complexity [32]. Although NMLab employs two advanced simulators, creating complex dynamic systems in this simulator can be complicated [23].

3.4.2 MATLAB/NS-2 integration by Soglo

Another attempt to integrate MATLAB and NS-2 has been done by A.B.Soglo. In this system, the simulator contains a network environment model in NS-2, control systems, which are modeled by MATLAB or C/C++, and an external application programming interface. Also, an especial User Datagram Protocol (UDP) [39] packet format has been implemented to carry the controlling data in this simulation environment [41].

3.4.3 Modelica/NS-2

Integration of Modelica/NS-2 creates another hybrid simulator for NCSs and WNCSs. Modelica is an object oriented equation based language for modeling the complex and large-scaled physical systems. In addition to supporting model construction and reusability, it contains several libraries for modeling different control systems. Different free or commercial simulation environments are available for Modelica; and some of them can conveniently export their models to other simulators like Simulink [2][19]. The integration of Modelica and NS-2 creates a proper simulation environment, which can simulate both physical systems and network communications.

3.4.4 OPNET/SIMULINK

Optimize Network Engineering Tools (OPNET) is an object oriented and multi-purpose network simulator [25]. OPNET can integrate communication hardware’s models, network control software, transmission effects and complex operating environments together [34]. In comparison with NS-2, OPNET is more advanced with
the ability of simulating physical links and antennas in addition to the regular parts. Also, its configuration and visualization capabilities are better than NS-2[23]. In order to simulate WNCS, OPNET has been integrated to Simulink and can use the facilities of both simulators [31].

3.4.5 Arena/NS

Arena is a simulator for multi-robot systems and has been developed at Southern University of California. NS is used for network communication simulation. Combination of Arena and NS creates an environment for simulating mobile multi-robots scenarios with inter-robot communications [47].

3.4.6 PIccSIM

PIccSIM is one of the latest hybrid simulators for NCS and WNCS, created by integration of Simulink and NS-2. It has been released as an open source simulator in the Helsinki University of Technology. PIccSIM Toolchain is a valuable toolset for design, modeling and implementation of WNCS and NCS. Using this Toolchain, makes it possible to perform the whole development of control system including design, simulation, and implementation in the same environment. PIccSIM is also facilitated with a graphical user interface, and an automatic code generation tool in the Simulink models for wireless sensor nodes [23].

3.5 Emulators

Mote emulators form an especial type of simulators that can simulate wireless nodes’ operating systems and run their actual programs inside the simulated node. They can also simulate the communications among the motes. Considering these specifications, node emulators are important environments for WSN simulation while they cannot support the control systems modeling, they are good options for creating hybrid simulators for WNCS. Two of the most important emulators, which are designed for two main mote operating systems are TOSSIM and Cooja.

3.5.1 TOSSIM

TinyOS Simulator (TOSSIM) is an emulator, particularly designed for TinyOS. It has been developed in 2003 by University of California Berkeley’s TinyOS team. This emulator is a bit-level discrete event network emulator built in Python[6]. It is open source, well documented with a good graphical interface [35][10][29]. In spite of its advantages, TOSSIM has some limitations. For instance it cannot simulate the performance metric of other new protocols, disables it to correctly simulate the energy consumption in WSN. Also, science as all of the TOSSIM applications should be designed on TinyOS, it can only emulate the homogeneous applications [29].
3.5.2 Cooja

Cooja is a cross-level simulator for the Contiki sensor nodes, which is able to simulate both of the low-level sensors’ hardware and high-level behaviors in a single environment. It allows simulation in three different levels including, networking or application level, the operating system level and the code instruction level. Figure 3.3 depicts different levels of simulation in which Cooja can operate.

Although Cooja has been developed for the Contiki operating system, it is able to simulate non-Contiki nodes such as the nodes implemented in java or other operating systems [38]. This emulator which is developed in SICS has been utilized in this project and is able to emulate several motes simultaneously with their communications. Since Cooja is implemented in Java, it is easy to develop new plug-ins for different functionalities and use them in simulations. Picture 3.4 shows a general view of the simulation environment in Cooja with several simulated motes inside it.
This chapter explains the integration of Simulink and Cooja as the main contribution of this project, which has resulted in a new co-simulator for wireless networked control systems, so called GISOO. Section 4.1 discusses the main reasons and motivations for this project. Section 4.2 explains the architecture of GISOO and the logic behind the designed methods for integration of Simulink and Cooja. Details of the implementation and introduction of the new components in Simulink and Cooja are explained in section 4.3; and finally, section 4.4, compares GISOO as a result of the implementation with PIccSIM, which is another simulator for WNCS.
CHAPTER 4. INTEGRATION OF SIMULINK AND COOJA

4.1 Motivation

Although several different simulators have been developed for NCS and WNCS simulation, none of them has the possibility to perform simulation by using the real code developed for wireless sensor nodes. So, for most of them the control systems and the communication networks should have been designed as a model inside the simulators; finally after the evaluation of the results, the necessary parts should be implemented inside the nodes which can be a fallible step. In the best situation it may be possible to generate the nodes’ codes automatically according to the model; however in these simulators one cannot simulate his own codes and implementations. Lack of an environment for simulation and evaluation of our developed codes in WNCSs was the main motivation for this project. This project aimed to create a virtual test bed for WNCSs, which enables us to execute our developed code in a simulated environment for both test and evaluation. Moreover to this target, Cooja which has been used in this project can provide us with many facilities which are not used in the previous simulation environments.

4.2 GISOO Architecture

GISOO, basically, has been made of two separate simulators while both of them can work on the same computer. Simulink simulates the physical system, while Cooja emulates wireless motes, executes their programs, and simulates the communication among them. These two simulators together, makes GISOO as a co-simulator for wireless network control systems. Picture 4.1 shows the general architecture of GISOO.

![GISOO Architecture Diagram](image)

**Figure 4.1:** GISOO architecture schema. In GISOO Simulink simulates the physical system and Cooja emulates sensors, relays, controllers and actuators.

In the above system, Simulink simulates the physical model and sends the generated results to the appropriate mote in Cooja. it also receives the necessary results of control system from Cooja and uses it as an input for the model of physical system. On the other side, Cooja simulates other parts of the control system such as sensors, controller and actuators inside different emulated motes. It receives the
4.2. GISOO ARCHITECTURE

sensing data from physical model in Simulink and generates the actuation values in the result.

To create this integration between Simulink and Cooja, proper methods for data communication and time synchronization are necessary. Details of implemented methods in this project are explained in the following sections.

4.2.1 Data communication

Data communication, which is required for both time synchronization and data exchange between simulators has been one of the most important parts of the implementation. Two options for data communication between two simulators were using the UDP [39] or Transmission Control Protocol (TCP) [40]. While TCP can provide a reliable data transmission by itself and UDP has not such possibility, the header size of TCP is much bigger than UDP causing more overhead; especially when the payload data for each packet is small. Figure 4.2 compares the size of the header part in UDP and TCP.

**TCP and UDP Headers**

**TCP Segment**

<table>
<thead>
<tr>
<th>Bit (0)</th>
<th>Bit (15) Bit (16)</th>
<th>Bit (31)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port (16)</td>
<td>Destination Port (16)</td>
<td></td>
</tr>
<tr>
<td>Sequence Number (32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number (32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length (4) Reserved (4) Code Bits (6)</td>
<td>Window (16)</td>
<td></td>
</tr>
<tr>
<td>Checksum (16)</td>
<td>Urgent (16)</td>
<td></td>
</tr>
<tr>
<td>Options (0 or 32 if any)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPLICATION LAYER DATA (Size varies)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20 Bytes

**UDP Datagram**

<table>
<thead>
<tr>
<th>Bit (0)</th>
<th>Bit (15) Bit (16)</th>
<th>Bit (31)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port (16)</td>
<td>Destination Port (16)</td>
<td></td>
</tr>
<tr>
<td>Length (16)</td>
<td>Checksum (16)</td>
<td></td>
</tr>
<tr>
<td>APPLICATION LAYER DATA (Size varies)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8 Bytes

*Figure 4.2: Structure and size of the header in UDP and TCP packets.*
Based on this fact, most of the real-time applications prefer UDP to TCP for their communications. Accordingly, as most of the packets contain small amount of data in this project, UDP has been selected for data communication between Simulink and Cooja.

For time synchronization both of the simulators should be aware of simulation time in the other simulator. Since all of the synchronization data should be transferred between Simulink and the main application of Cooja, the communication method between them is simple and straightforward. fulfilling such requirements both of simulators use especial UDP ports (44445-44446) to send and receive timing information to/from each other.

Except for synchronization data, rest of the data in sensing or controlling data which should be transferred between Simulink and a specific mote inside the Cooja. So, data communication for this part is not as simple as synchronization part, while data should be communicated via a middle application to reach a particular mote. Since Cooja is a mote emulator, communication to a mote inside Cooja is similar to communication to a real mote over the network, via its serial port. For this aim, ”Serial socket server” plug-in in Cooja allocates a port number (60000+Mote-ID) to each mote, via which one can communicate with a mote through its serial port. Since Simulink cannot send pure data, directly to the mote’s serial port, a middle application as an interface has been created. This interface provides a UDP port, which is utilized by Simulink to communicate with the specific mote in Cooja. Since such interface depends on the message structure, it can differ for sending and receiving. Figure 4.3, shows the communication procedure between Simulink and Cooja.
4.2.2 Time synchronization method

To obtain correct results in an integrated simulation environment, both simulators should work synchronously. GISOO uses an especial method for this aim. Since Cooja is relatively slower than Simulink, it will control the speed of simulation by working continuously and sending its simulation time to Simulink periodically, while Simulink is much faster it reach the Cooja’s simulation time and then waits to receive a new simulation time from Cooja.

4.3 Implementation and new components

In order to integrate Simulink and Cooja, different components have been implemented for each of them. Such new components are described in the following sections.

4.3.1 Simulink’s components

A new library with different blocks has been created in Simulink. One of the main blocks of this library is the “Synchronizer” block which performs the synchronization
between Cooja and Simulink. Another designed block, 'Enabling signal generator', is used to send data from Simulink to Cooja with especial sampling time. For sending data to an especial mote in Cooja a 'send' block has been designed; while preexisting 'UDP Receive' and 'Unpack' block in xPC Target library are used for data receiving. Figure 4.4 shows the usage of these blocks in a sample Simulink model. In this picture 'Synchronizer' and 'Enabling signal generator' have been shown in orange and yellow respectively. These two blocks are more explained in the following.

**Figure 4.4:** A Simulink sample model in GISOO. 'Synchronizer' and 'Enable Signal Generator' are shown in orange and yellow respectively.

### Synchronizer block

The time synchronization between Simulink and Cooja has been achieved by two different parts; a block in Simulink and a plug-in in Cooja. The Simulink block is the same block developed in the Helsinki University of Technology for PIccSIM simulator. The structure of this block is shown in Figure 4.5. The block receives the simulation time of Cooja over a UDP port '44445', (highlighted in blue color in the figure), compares the received time with the simulation time in Simulink (highlighted in green in the figure); and if Cooja is behind, it uses an empty loop to wait for Cooja (highlighted in red in the figure); otherwise it continues the simulation until the received time. Simulink also sends its simulation time (using UDP port number 4444) to Cooja periodically which can be used when Cooja is ahead (highlighted in yellow in the figure). In the case that Simulink’s waiting time exceeds a predetermined value, the simulation will be stopped (highlighted in pink in the figure).
4.3. IMPLEMENTATION AND NEW COMPONENTS

Figure 4.5: Structure of time synchronization block in Simulink.

Enabling signal generator

This block has been created to activate the send block in the Simulink model according to the designed sampling time in the control system. This block can receive the system sampling time via the parameter, 'Enabling period', and use the 'Start time' parameter to postpone sending samples for a specified while. Figure 4.6 shows the block parameter window of this block. In this figure the block will start to activate the send block immediately after the start of the simulation and it will activate it every 200 milliseconds.

Figure 4.6: General view of Enabling signal generator block parameter
4.3.2 Cooja’s component

The developed plug-in for Cooja is the time synchronization plug-in which can be found in the plug-in menu in Cooja’s simulation environment. This block receives the IP of Simulink machine for communication. So, if both Simulink and Cooja were running on a same computer the IP can be left blank. It also needs the UDP port number of synchronization block in Simulink which is 44445 by default. Picture 4.7 shows the synchronization plug-in in Cooja.

![General view of Synchronizer plug-in in Cooja.](image)

**Figure 4.7:** General view of Synchronizer plug-in in Cooja.

4.3.3 Implementation evaluation

In order to evaluate the implementation, a test scenario has been created for the complete system. In this test, Simulink generates a counter and sends it to a mote in Cooja. This counter data is then forwarded to the relay mote that passes it to another mote without any modification. Then the receiver mote sends back the counter to the Simulink. By creating this loop and comparing the sent data and received data one can evaluate the amount of delay in the system. Figure 4.8 shows the sent counter and received counter in a sample period in the same graph. In this graph the sent data and the received data are shown in red and blue respectively.
Figure 4.8: Sample results of counter delay test for implementation evaluation.

Figure 4.8 depicts that the sent data has been received with some amount of delay. Although may be possible to reduce the variation of delays with more accurate synchronization or improving the communication method, as long as the counter data received before the next sample generation, the simulation’s results are acceptable. In this regard selecting a greater sampling interval can increase the chance of data receiving in the correct time. Figure 4.9 shows the delay between sending and receiving of each counter data in Figure 4.8. To have a proper simulation the selected sampling time should be greater than the maximum delay time.

Figure 4.9: Sample results of delay variation in the counter delay test.
4.4 Comparison of Gisoo and PiccSIM

Since PIccSIM is one of the most important and recently developed co-simulators for WNCS comparing the GISOO and PIccSIM in different aspects can be valuable. The main difference between GISOO and PIccSIM is the type of combined simulators inside them. Both of them simulate physical systems by using Simulink. However for network simulation, PIccSIM uses NS-2 while GISOO, uses the Cooja which is a mote emulator. Based on this, PIccSIM simulates a modeled environment by modeling and simulating both networks and control systems. However thanks Cooja, GISOO can uses the exact code of different motes inside the simulator and evaluate the system with the real program of the motes. This significant difference makes GISOO not only a powerful simulator but even a test bed for new developed code for WNCSs. The other difference between these two simulators is related to their architectures. Figure 4.10 shows the architecture of PIccSIM.

As can be seen in figure 4.10, in PIccSIM all of the plant, sensors and controllers should be modeled in the Simulink and NS-2 only models the data transmission between them. But in GISOO, Simulink only models the physical systems and the rest, such as sensors, Controllers and actuators or relays as well as their communications, should be modeled inside the Cooja. Figure 4.11 shows the architecture of GISOO.
Figure 4.11: Architecture of GISOO. Simulink, simulates the plant and the rest are simulated in Cooja.
GISOO is a new simulation environment for wireless networked control systems. In order to clarify its usages, different experiments have been modeled using this new system which are explained in the following order in this chapter. The first section introduces a sample scenario of WNCS. Section 5.2 describes the procedure of performing an experiment in the real world and in the simulation environment. In the next section modeling of physical system and simulation of wireless devices with their communication are discussed. Then section 5.4 describes the necessary steps for connecting the Cooja and Simulink together and finally the last section demonstrates some sample scenarios that have been modeled in GISOO.
5.1 Coupled Tank

Coupled Tank, is one of the well-known and regular control lab instruments which aims to create a basic control scenario. This system is composed of two tanks with a hole at the bottom, while one tank is placed above the other one. Both tanks have separate sensors to sense the water level inside them. The system also contains a motor which can pump water from a water reservoir to the upper tank. The aim of this system is to configure a controller and apply a desired value of voltage as the actuation signal to the pump to stabilize the water level in the lower tank on an especial value. Figure 5.1 shows a picture of Coupled Tank which has been used in this project.

![Coupled Tank System Diagram]

**Figure 5.1:** A Coupled Tank system. To create a sample WNCS, different motes uses as sensor, controller, relay and actuator [21][14].

To create a wireless system to study the WNCS, different motes can be used to sense the water level, act as controller or actuator. In this system also several motes can be used as relay motes to transmit data between different parts.

5.2 Real-world experiments and Simulation procedure

In order to create a control system which can perform required controlling tasks like controlling the water level in the Coupled Tank system, different steps should be taken. First of all, the physical system should be modeled and in the next step, this model has to be used to design a proper controller for the control system. Based
on such design one can develop the necessary code for it. Finally, using the code a closed-loop for the real control experiment can be created.

In order to simulate the real experiment, firstly the physical system should be modeled to be used in Simulink. In the next step using the same developed code for the real experiment and the proper network settings in Cooja creates the simulated wireless devices and communication network. Finally, after simulation preparation steps which are needed for connecting Simulink and Cooja together, we can create a simulated closed loop for our control experiment. Figure 5.2 shows a diagram of creating a closed loop in both the real world and simulation environment.

The necessary steps to simulate a WNCS scenario in GISOO are described in the following sections. In this document, such steps are explained for this project however they are general and are the same for all scenarios.

### 5.3 Physical system modeling

In order to model the Coupled Tank, with different scenarios, first of all the physical system which is water level in this experiment, should be modeled in discrete mode in Simulink. Figure 5.3 shows a designed model of this system for the simulation. In this design the system receives the voltage of motor as an input and calculates the water levels in both tanks. Such water levels are then sent to the sensor mote.
(emulated in Cooja) with the same sampling period as of the sensor. For this aim we can use "Enabling signal generator" which has been created for GISOO.

In GISOO, Cooja emulates motes for different functionalities such as sensing, controlling, actuating and relaying. Cooja designates mote-ID to the motes based on their order in being created. So, the user should be aware of creation order, to have proper IDs for the motes. Figure 5.4 shows the steps for creating a new mote in Cooja. The executable file (main.exe) which has to be selected during the mote creation is as same as the file, created by TinyOS in the build folder using "make tmote" command.
5.4 GISOO preparation

In order to create time synchronization and proper data communication between Simulink and Cooja the following steps should be followed:

5.4.1 Synchronization

For time synchronization between Simulink and Cooja, the synchronizer block in Simulink should be added to the physical model, and the synchronizer plug-in with the proper settings should be used in Cooja as described in 4.3.2.

5.4.2 Data Communication

In the water tank model, Simulink should send water levels to the sensor motes in Cooja and actuator mote in Cooja should send back the voltage of motor to the Simulink. This communication loop between Simulink and Cooja has to be created via the interface application, which requires the following steps:
For all the motes that have to communicate with Simulink - sensor and actuator in this project - the serial socket server which is a Cooja’s plug-in for communication to the serial port of emulated motes should be run. Figure 5.5 shows the procedure for running this plug-in. This plug-in listen to an especial port number (6000+mote-ID) and makes a connection between that port number and the serial port of the desired mote.

![Figure 5.5: Creation procedure for SerialSocket-Server Plug-In and its general view.](image)

A message class should be created according to the message structure that has to be passed from Simulink to Cooja and vice versa by using Message Interface Generator (MIG). In this project the messages from Simulink to Cooja should be created based on the "SensorValues" structure, and from Cooja to Simulink according to the "EncMsg2SensorsAct" structures, where both are located in the "app_parameters.h". The complete explanation of MIG is available in TinyOS tutorial, however the mentioned structures for this project can be created by the two following commands:
5.5 Simulation of different scenarios

Previous steps prepare the simulation environment. Using this environment we can simulate different scenarios in WNCS.

5.5.1 Simple scenario for simulation evaluation

The first scenario which has been implemented in GISOO was simple Coupled Tank scenario with the target of stabilizing the water level in the lower tank at 5 units. This scenario was consisting of four motes: a sensor for receiving data from tanks’ model, a relay mote to forward data from sensor to the controller, a controller,
and an actuator which send the desired voltage level (between 0 -15) to the tanks’ model in Simulink. These motes had the mote-IDs 5,7,4,6 respectively. In order to evaluate the simulation results this scenario was firstly implemented in real-world using real wireless motes, from which, water level of both tanks as well as to the voltage of motor were recorded. After real experiment, for simulating the scenario, the physical system which is the water level was modeled in Simulink. Finally using this model and the motes’ code, the same experiment with the similar situation was simulated and the water levels and voltage of motor were recorded. Picture 5.6 shows the simulator environment for this scenario.

![Simulation environment in GISOO simple Coupled Tank scenario.](image)

**Figure 5.6:** Simulation environment in GISOO simple Coupled Tank scenario.

Comparison of the results in real experiment with the simulated one, proves that the simulated results match the real one’s. Figure 5.7 shows the water levels and voltage of motors in both simulated and real experiment.
5.5. SIMULATION OF DIFFERENT SCENARIOS

5.5.2 Under attack relay scenario

The second scenario was similar to the first one with only one difference: using a modified relay mote. In this scenario the relay mote stopped forwarding data for 15 seconds period, 10 seconds after the beginning of the experiment. Figure 5.8 shows the water levels and level of voltage in this scenario. As can be seen in the figure, the voltage is constant for 15 seconds after the 10th second because of not receiving any value from the relay node. After this period, when the controller receives a new correct data it immediately changes the voltage level to control the water level.

![Figure 5.7: Simulation evaluation. Comparison between voltage and water levels in Simulation and real experiment.](image1)

![Figure 5.8: Water levels and voltage for simulated 'under attack relay' scenario.](image2)
Communication delay and maximum number of motes

In this chapter GISOO will be used to study the affect of communication delay on the maximum number of relay motes for a specific sampling time. In this regard section 6.1 explains the concept of delay of relay. The next section discusses the simulation of a specific scenario to calculate the delay time between relay motes. For this aim we have used an especial plug-in in Cooja to record all the radio communications’ timing details. Finally using these data, the maximum number of relay motes which does not affect the controlling results is calculated.
6.1 Delay of relay

In our simulation, the delay of relay is amount of time which is passed from the data receiving instant by a relay until the data receiving instant by a mote, to which data is forwarded by the relay. Figure 6.1 tries to depict this concept. In the figure delay of relay is equal to $t_2-t_1$. In every system adding each relay mote, imposes a delay for data receiving in the last mote.

![Figure 6.1: Delay of relay. In this figure delay of relay is equal to $t_2 - t_1$](image)

6.2 Simple scenario with several relays

In order to calculate the effect of communication delay on maximum number of relay motes, a scenario with several relay motes should be used to be able to collect enough sample of relaying time. For this aim a simple Coupled Tank scenario with 10 relay motes, has been simulated. 6.2 shows the simulation environment for this scenario. During the simulation by using "Radio logger" plug-in in Cooja, all the simulated radio communications between different motes, with their timing details have been recorded. In the next step, using the recoded data and a MATLAB

![Figure 6.2: Simulation environment for simple Couple Tank scenario with 10 relay motes.](image)
function, communication delay between relay motes has been calculated based on which figure 6.3 that shows the relay delays in the simulation has been generated.

Figure 6.3: Delays of relays between relay motes in a Coupled Tank scenario with 10 relay motes.

According to the generated data, the imposed delay by relay motes has the following characteristics:

- Maximum delay: 14 ms
- Minimum delay: 3 ms
- Average of delay: 8.512 ms

Based on the above data and in order to reach a trustworthy system, maximum delay time has been considered as the relay’s delay in this scenario.

In a control system, communication time should not take longer than sampling time; otherwise, the imposed delay can affect the controlling results. So considering that the sampling time in this scenario has been 200 ms, and in order to create an accurate control loop for this system, less than 14 relay motes should be used.
Conclusion and future works

7.1 Conclusion

Considering the necessity of having a simulation environment to study the structure and different aspects of WNCS, which requires to cover simulation of both control systems and wireless communications, has resulted in creation of GISOO.

This new simulation environment for WNCSs has been developed by integrating the Simulink as a physical system simulator, with Cooja, which is a wireless mote emulator. Since GISOO is a co-simulator, it can simulate both sides of control system and wireless networks in an appropriate way. Moreover, using the Cooja as a mote emulator inside this simulation environment makes GISOO a unique simulator for WNCS, which can simulate different scenarios by executing the actual code of wireless motes. Such significant specification of GISOO, not only makes it a powerful simulator, but also renders it as a proper simulated test bed to test the new developed code for wireless motes.

To achieve this aim, different challenges in regard of synchronization and communication between Simulink and Cooja have been solved and some sample scenarios have been simulated in the new environment. Finally by using some of the facilities of GISOO the effect of communication delay on the number of motes in a system has been studied.

7.2 Future works

Although GISOO is now available, it still has a high potential for improvement. Some of its potential enhancements are listed in the following.

- Communication: currently communication between Simulink and Cooja, requires a middle interface that has to be executed which can increase both
delay and inaccuracy. Improving such communication system by enhancing this interface or even making a direct connection between Simulink and Cooja will be a considerable improvement for the GISOO’s performance.

- Synchronization: although the current synchronization plug-in in Cooja works fine, a new version of this plug-in which that can stop the Cooja until it receives a new time from Simulink can improve the accuracy of the simulation.

- User Interface: a new user interface which covers all different parts of GISOO can facilitate the users to deal better with the environment.

- Mote movement facility: the current version of GISOO does not support mote mobility, for which the new position of a mote, calculated by the physical system Simulator, should be received by Cooja. Adding the possibility of receiving such data to Cooja can improve GISOO’s functionality to Simulate more advanced scenarios.

- Integration of GISOO or Simulink with NI LabVIEW: such integration will provide the possibility of using GISOO in many applications because of the widespread usage of the NI LabVIEW.
References


REFERENCES


REFERENCES


