GISOO: a virtual testbed for wireless cyber-physical systems

Behdad Aminian, José Araújo, Mikael Johansson, Karl H. Johansson

ACCESS Linnaeus Centre
KTH Royal Institute of Technology
Stockholm, Sweden
Email: {behdad, araujo, mikaelj, kallej}@kth.se

***Abstract*—The increasing demand for wireless cyber-physical systems requires correct design, implementation and validation of computation, communication and control methods. Traditional simulation tools, which focus on either computation, communication or control, are insufficient when the three aspects interact. Efforts to extend the traditional tools to cover multiple domains, e.g., from simulating only control aspects to simulating both control and communication, often rely on simplistic models of a small subset of possible communication solutions. We introduce GISOO, a virtual testbed for simulation of wireless cyber-physical systems that integrates two state-of-the art simulators, Simulink and COOJA. GISOO enables users to evaluate actual embedded code for the wireless nodes in realistic cyber-physical experiments, observing the effects of both the control and communication components. In this way, a wide range of communication solutions can be evaluated without developing abstract models of their control-relevant aspects, and changes made to the networking code in simulations is guaranteed to be translated into production code without errors. A double-tank laboratory experimental setup controlled over a multi-hop relay wireless network is used to validate GISOO and demonstrate its features.**

I. Introduction

The integration of wireless communications in cyber-physical systems (CPSs) such as process automation, building automation and intelligent transportation systems, poses many challenges and has become an area of intense research [28], [3], [27]. Wireless CPSs are complex systems characterized by dynamic interactions between different devices (sensors, controllers, actuators, radio transmitters and receivers) and their environment (the physical process under control as well as the communication medium). The correct design, development, and validation of cyber-physical systems is recognized to be an extremely hard task, requiring deep knowledge of communication, computation and control [3], [27], [4]. Simulation is an important tool for evaluating various designs and for developing a thorough understanding of the interactions among the different components in wireless CPSs.

In this paper, we introduce GISOO, a virtual testbed designed to provide realistic simulations of all the components of a wireless cyber-physical system. GISOO integrates Simulink [2], the most widely used tool for simulation and model-based control design, and COOJA [25], a comprehensive wireless sensor network simulator capable of emulating real embedded code under realistic models of timing and wireless communication [11]. The virtual testbed that we propose has the following features: a) runs real embedded code including the full wireless communication stack, combining medium-access control (MAC), routing and application layer and reproduces timing and packet loss rates without the need for building abstract simulation models, b) it allows embedded wireless communication code to be emulated in GISOO without any changes, so that the same code that has been evaluated in simulations can be executed directly on the target platform, c) it provides full flexibility in the design of the CPS architecture, where computation, control and actuation may be implemented directly in wireless devices or in Simulink, d) it allows a comprehensive analysis of the interactions of communication, computation and control components of the cyber-physical system and, e) it supports various widely adopted wireless platforms in both TinyOS and ContikiOS operating systems through COOJA. The code and manual of GISOO, together with multiple example scenarios is available in the GISOO page [19].

II. Related work

Many efforts have been made to simulate and validate wireless CPSs with a variety of simulation tools. However, various limitations exist in the currently available tools. Matlab/Simulink is the most widely used tool for modelling and simulation of control systems. The Truetime simulator [9] is a toolbox implemented in Matlab/Simulink which provides link layer level simulation through various models. Packet-level network simulation can be achieved through the usage of ns-2 and OMNet++ which can provide high-level protocol stack implementation. The PiccSIM tool chain [8] provides an integration of Matlab/Simulink models with ns-2, allowing for control system design and automatic code generation. Similarly, NCSWT [12] integrates Matlab/Simulink and ns-2 for simulating networked control system. A common denominator of all the above tools is that all require the application code and communication protocols to be modelled and developed for their native simulation tool. Moreover, the simulation accuracy is dependent on the network modelling capabilities of the simulator, and the wireless models in ns-2 are not able to capture the typical communication interactions among nodes in a network. Emulation of real code with realistic and detailed network behavior has become a priority in recent developments. Such feature is provided in the wireless sensor network tools TOSSIM [21], COOJA [25] and recently in [23], providing realistic network emulation environment for CPS security experiments. Both TOSSIM and COOJA are able to capture the complex communication dynamics and emulate real code, which are strong requirements for faithful development, test and validation of wireless CPSs. The WCPS tool developed in [29] integrates Simulink and TOSSIM and has been applied on wireless structural control applications [22]. The TOSSIM simulator emulates the TinyOS operating system [20], solely for micaZ wireless platform and is designed to perform operating system level simulations. On the other hand, COOJA is able to emulate additional wireless

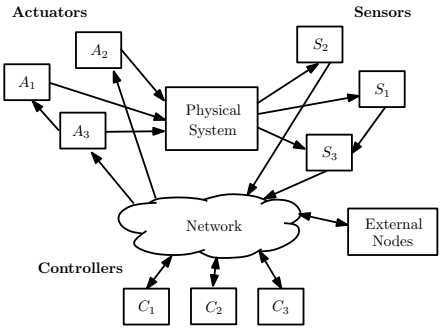


Fig. 1: A general wireless cyber-physical system architecture.

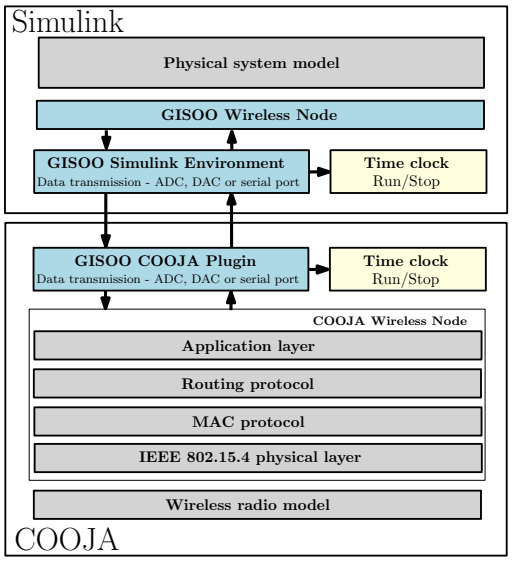


Fig. 2: Detailed illustration of the GISOO architecture.

platforms using both TinyOS and ContikiOS [10] and works at networking, operating system and machine code instruction set level. Additionally, COOJA is flexible, easily extendible and with many extensions/plugins available off-the-shelf. While GISOO shares the same real code emulation goal for wireless CPSs as WCPS, we choose to integrate Simulink with COOJA because of all its advantages in comparison to TOSSIM. The architecture of the CPS provided by the WCPS simulator is fixed and tailored only for data collection applications, where only the sensor devices and relay nodes are wireless, while the controller and actuators are required to be implemented solely in Simulink. In GISOO, any kind of wireless CPS architecture can be implemented and every CPS component is allowed to be a wireless device. Additionally, in WCPS, communication between the nodes and Simulink is performed through WCPS-specific functions and not using the native peripherals of the wireless platform as provided in GISOO. Such functions do not allow the wireless devices code to be utilized in a real scenario without requiring code modifications.

III. GISOO: A VIRTUAL TESTBED FOR WIRELESS CPSS

GISOO supports a general wireless CPS as shown in Fig. 1. This system consists of several wireless sensors and actuators collocated with a single or multiple physical system(s), communicating with a set of controllers through a wireless network. In our architecture the controllers may be centralized or distributed and can be implemented inside wireless devices or simulated in an external CPU. The wireless network uses the IEEE 802.15.4 standard as the physical layer and is open for any specification of the other communication layers. Particularly, MAC and routing protocols may be designed by the user, or available protocols may be used off-the-shelf. Network transmissions may be scheduled by a centralized networked manager or decided locally by each individual node in the network.

A. Architecture

GISOO is an integrated simulation and emulation platform for wireless CPSs which integrates Simulink for simulating the physical system dynamics and perform controller designs, and COOJA for simulating wireless devices in a wireless sensor and actuator network. Simulink has been the most widely used tool to design and study control systems by control engineers, while COOJA is a simulator that provides simultaneous cross-level simulation at application, operating system and machine code instruction set level in a single framework, being specifically designed to emulate wireless sensor and actuator networks using realistic wireless radio mediums [11]. COOJA is flexible and extensible in that all levels of the system can be modified or replaced, e.g, sensor node platforms, operating system software, radio transceivers, and radio propagation models. Moreover, COOJA is able to emulate wireless devices in both Contiki OS [10] and TinyOS [20], which are the most widely used operating systems for wireless sensor and actuator networks. With the combination of Simulink and COOJA we are able to provide an environment to simulate wireless CPSs in a realistic manner.

The GISOO architecture is illustrated in Fig. 2. A substantial effort has been devoted to making sure that embedded wireless code can be used in GISOO without any manual intervention or changes to the code. This is achieved by the GISOO Plugin implemented in COOJA which monitors any calls made by the native Analog-to-Digital converter (ADC), Digital-to-Analog converter (DAC) and serial port functions in the real wireless nodes. Whenever these functions are called, the plugin is responsible for exchanging data with Simulink, retrieving Simulink data to the wireless node in COOJA or delivering data to Simulink from the wireless node.

A typical wireless CPS scenario with communication between sensors, controller and actuators through a wireless network has the following flow in the GISOO simulator. The sensor data is the output of the physical system modelled in Simulink and is to be read by the ADC or serial port of the wireless sensor node. A COOJA GISOO plugin is activated in an event-driven fashion, whenever a request for an ADC reading or serial port communication is issued by the wireless node. Communication then occurs between COOJA GISOO plugin and Simulink GISOO plugin where a reading request at the GISOO wireless node in Simulink is issued at the specific simulation time. These readings may be periodic or be generated in an event-based/aperiodic fashion. Incoming data from Simulink is fed through the COOJA GISOO plugin into the wireless mote in COOJA. This data may then be transmitted by the sensor node through the wireless network, which may be composed of many relaying nodes (pure relay nodes or actual sensor nodes), until it reaches the specific

controller node.
The controller algorithm may be implemented in a wireless node or in Simulink, using appropriate Simulink blocks. In the former case, after the computation of a new control input, the controller node transmits this data to an actuator node through the wireless network. In the latter scenario, the sensor data can be communicated to the Simulink controller block by a wireless basestation/controller mote through the serial bus. This scenario occurs in reality when a controller is implemented in Simulink and the CPU interfaces the wireless network through a wireless node connected to the CPU's USB port [5]. The advantage of allowing for controller design in Simulink is that changes in the controller algorithm do not require a modification of the wireless devices code, increasing the speed of evaluating new controller designs. Finally, the actuator node is applied the control input to the specific actuator through the Digital-to-Analog converter (DAC) port or serial interface. The request for the actuation input is issued at the wireless node and is received by the COOJA GISOO plugin. This plugin communicates with the Simulink GISOO plugin, which properly sets the actuation value at the requested GISOO wireless node in Simulink.

Synchronization between COOJA and Simulink is mantained by a stop and run mechanism where GISOO's COOJA plugin controls the time clocks. Whenever an event happens at a wireless node, e.g., performing an ADC reading or a DAC writing, the action and the time at which this takes place in COOJA are transmitted to the GISOO node in Simulink. At this moment, the time clock of Simulink has been stopped since the last event has occurred. Then, Simulink time clock is allowed to run until the time requested for the action to take place. After the action is completed, the time clock in Simulink stops again. After the action and time are transmitted between COOJA and Simulink, the time clock of COOJA also stops and starts running again when the action to be performed in Simulink is completed.

B. Communication protocols and sensor nodes

Several MAC and routing protocols are currently available in Contiki OS and TinyOS from which we highlight, the IEEE 802.15.4 MAC which is the current MAC standard for WSNs [17], [15], [16], the routing protocol for low power networks (RPL) proposed by the internet engineering task force (IETF) [30], [18] and the Collection Tree Protocol (CTP) [14]. Through COOJA, one has also access to many debugging features such as break points, watches, logging, and single stepping for the wireless code developments. Additionally, full access to the wireless transmissions including data and acknowledgement messages is provided, along with *printf* logging from all nodes and node statistics such as energy consumption. Moreover, several wireless sensor platforms are supported by both ContikiOS and TinyOS [1], [13].

IV. Case study - Networked control of double-tank systems

In order to validate GISOO and demonstrate its capabilities, we study the networked control of double-tank systems [7], which have similar properties of a real industrial automation setting. We start by describing the double-tank system, followed by closed-loop control and communication performance analysis when controlling the double-tank system in both reality and in GISOO, using a simple network setup. Afterwards, we perform communication analyses on large-scale a closed-loop control experiment of ten double-tank systems through a multi-hop relay network of sixteen relay nodes.

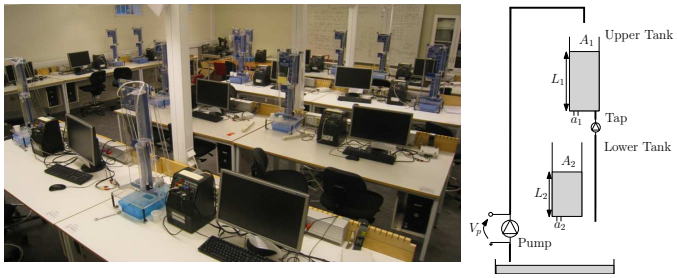


Fig. 3: Double-tank system testbed at KTH Royal Institute of Technology and the diagram of a double-tank system.

A. Double-tank system

The double-tank system consists of a pump, a water basin and two tanks of uniform cross sections. Fig. 3 depicts the experimental testbed and a diagram of the physical system. The liquid in the lower tank flows to the water basin. A pump fills the upper tank with water from the basin, which then flows to the lower tank. Sensing of the water levels is performed by pressure sensors placed under each tank. One wireless sensor node interfaces the sensors with an analog-to-digital converter (ADC), in order to sample the pressure sensor values for both tanks. The pump actuation is made through the digital-to-analog converter (DAC) of the wireless actuator node.

In this experiment, we chose to develop the control algorithm for a linearized model of the nonlinear double-tank system. However, we note that the usage of nonlinear models is available in GISOO through Simulink. The linearized model of the double-tank system is represented general linear state-space system

$$\dot{x}(t) = Ax(t) + BV_p(t) \tag{1}$$

where $x(t) = [L_1 \quad L_2]^T$, L_1 and L_2 are the height of the water in the upper and lower tanks and V_p is the voltage applied to the pump motor. A detailed description of the linear model is given in [5]. The control objective is to keep the water level of the lower tank L_2 at 5 cm by adjusting the motor voltage V_p accordingly. Tracking of a constant reference signal $r(t)$ can be achieved by using using a Proportional-Integral (PI) controller, with the control input $u(t)$ defined as,

$$\begin{aligned} u(t) &= K_p x(t) + K_i x_c(t), \\ \dot{x}_c(t) &= r(t) - Cx(t), \end{aligned} \tag{2}$$

where $x_c(t)$ is the controller integral state, $C = [0 \quad 1]$ and the controller gains (K_p, K_i) are chosen so that the closed-loop system is stable and achieves a desired closed-loop performance in terms of rise time, settling time and overshoot.

B. Wireless nodes and communication network

The wireless sensor platform chosen for this experiment is the Telos platform [26]. These nodes are equipped with a Texas Instruments MSP430 16-bit, 8 Mhz microcontroller with 48 kB of Flash and 10 kB of RAM memory and a 250 kbps 2.4GHz Chipcon CC2420 IEEE 802.15.4 compliant radio. All our

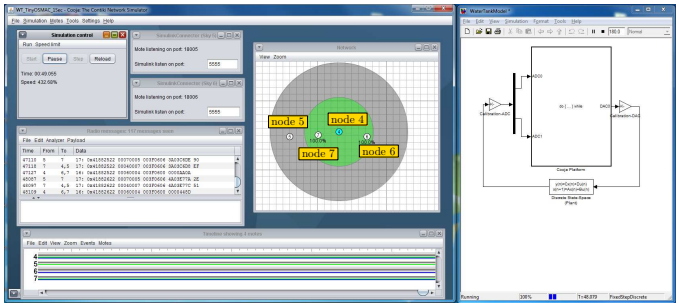


Fig. 4: GISOO environment, integrating COOJA (left) and Simulink (right) for the closed-loop control of a single double-tank system. Four wireless nodes are used in this experiment, a sensor (node 5), relay (node 7), controller (node 4) and actuator (node 6). The green region denotes the transmission range and the grey region the interference range of node 4.

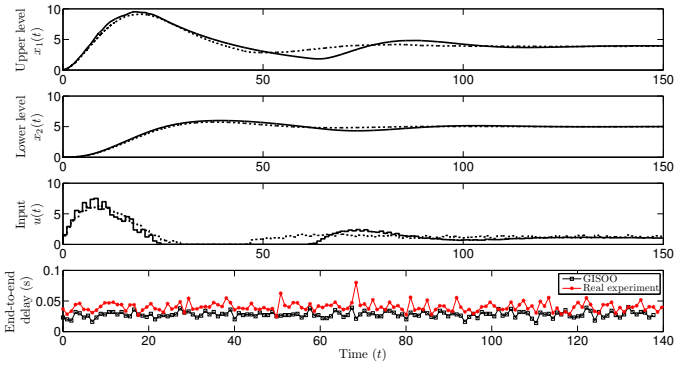


Fig. 5: Double-tank levels $x_1(t)$ and $x_2(t)$ and control input value $u(t)$ for a real experiment (dotted) and GISOO experiment (solid). The last plot depicts the end-to-end delay (ms) for both experiments.

devices are programmed using TinyOS. We used the TinyOS original MAC (BoX-MAC [24]) which is a CSMA/CA MAC with retransmissions in case of transmission failures. As the routing layer, we selected CTP [14].

C. Single process and simple network

We start by performing a comparison between a real closed-loop control experiment on the double-tank system and a closed-loop experiment simulation in GISOO simulation using the same wireless code.

In this setup, four wireless devices are used as the sensor, relay, controller and actuator. The GISOO environment for this simulation is presented in Fig. 4. The wireless sensor node (node 5 in GISOO) periodically samples every 1 s its ADC to acquire the tank level values, and transmits these values to an intermediate relay node (node 7 in GISOO). This node forwards the data packet to the wireless controller node (node 4 in GISOO), which computes the control action according to (2) and communicates the actuation input value to the wireless actuator (node 6 in GISOO). The wireless actuator applies the required voltage to the double-tank pump through the DAC.

The tank levels, the control input and the end-to-end delay for this experiment are shown in Fig. 5. The end-to-end delay is calculated as the time between the ADC sampling at the sensor, until a value is requested to be placed in the actuator's DAC, in both GISOO and in the real experiment. From Fig. 5 one can verify that the behavior of the linear double-tank model in GISOO, follows fairly close the real nonlinear double-tank system and a 5 cm set-point tracking on the lower tank is achieved. Additionally, the end-to-end delay average difference between the experiments is of only 1.27 ms. This is believed to be caused by the fact that the ADC reading and DAC writing action can be detected immediately in GISOO, while in a real experiment one has to rely on *printf* commands in order to accurately time-stamp the ADC reading and DAC writing actions.

D. Multiple processes and mesh network

We now aim at evaluating various typical scenarios in real large-scale wireless CPSs. For this, we deploy ten double-tank systems where its sensors communicate with a single wireless controller node through a sixteen wireless relay multi-hop network. Firstly, the case when sensors transmit periodically in the network is evaluated, followed by the analysis when the sensor transmission is event-based.

1) Periodic sensor transmissions: This scenario is depicted in Fig. 6 and the resuls on the end-to-end delay and packet loss are shown in Fig. 7a for nodes 1, 2, 5 and 8 which are at different hop levels in the network. As it can be seen, the distance to the controller greatly affects the end-to-end delay, while there is a large delay variability due to the high density of the network. Furthermore, packet losses exist in the network due to high transmission periods in the network.

2) Event-based sensor transmissions: The event-based sampler follows the proposal from [6]. The event condition which decides if the sensor node transmits the most recent tank levels is given by

$$|L_2(t) - L_2(t_s)| > e_{\text{lim}} \quad \text{OR} \quad h_{\text{act}} \geq h_{\text{lim}},$$

where $L_2(t_s)$ is the last transmitted lower tank level, e_{lim} the threshold value, h_{act} the time elapsed since the last transmission and h_{lim} is the maximum allowed inter-transmission time. The thresholds are set to $e_{\text{lim}} = 0.2$ and $h_{\text{lim}} = 10$ s, which on average generated 75 aperiodic transmissions on 150 s of simulation time, while achieving set-point tracking. The end-to-end delay results are depicted in Fig. 7b, where one can see that there is an increased variability on the delay value, as well as an increase of the average delay for sensors 1 and 2.

E. Impact of interference and node faults

The same scenario with multiple processes controlled over a mesh network is evaluated under 1) network interference and 2) complete deactivation of a set of four faulty relay nodes.

1) Interference: The end-to-end delay and packet loss statistics are shown in Fig. 8a. Interference is inflicted by an external node 59 in the region close to the controller node 41, which transmits periodically every 20 ms packets with 48 bytes of size, from $t = 40s$ onwards. As expected, the interference increases the end-to-end delay and the jitter. This is however not causing any packet losses in the network.

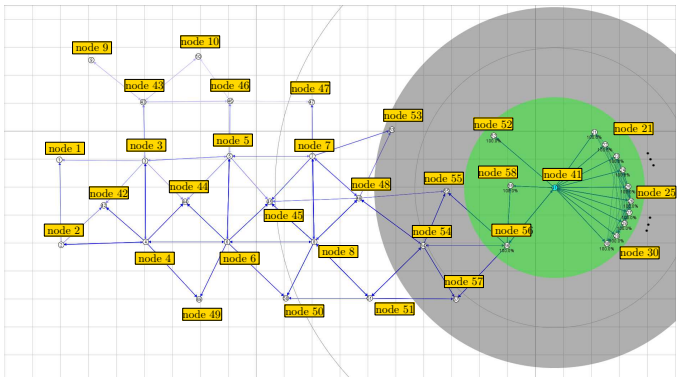
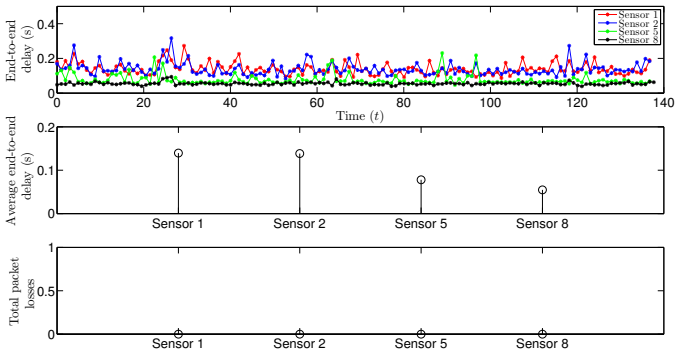
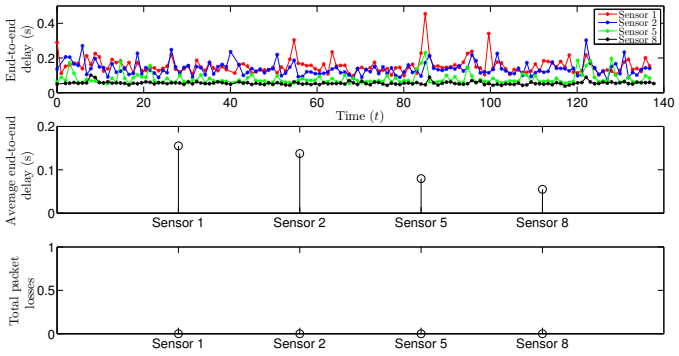


Fig. 6: Topology for the control of multiple double-tank systems over a mesh network with sixteen relay nodes. Sensors nodes are nodes 1 to 10, actuator nodes are nodes 21 to 30, controller is node 41 and the relay nodes range from 42 to 58. The transmission and interference range of all nodes is set as the controller node 41 in the figure.

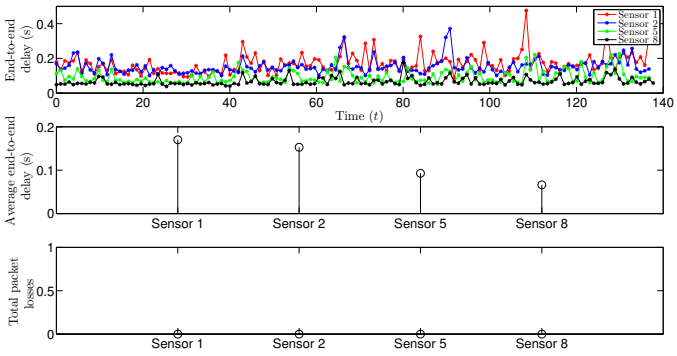


(a) Periodic transmission of sensor data with no interference or relay displacement as in Fig. 6.

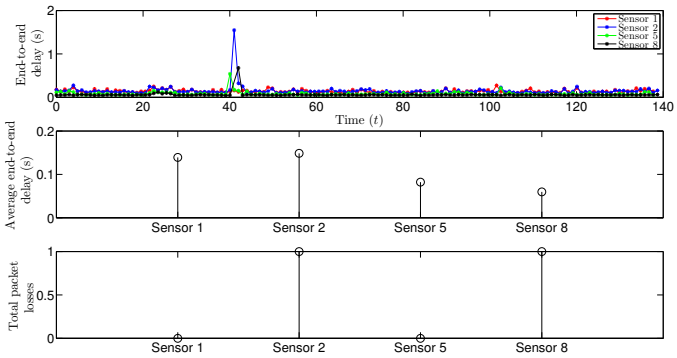


(b) Event-based transmission of sensor data with no interference or relay displacement as in Fig. 6.

Fig. 7: Evaluation of the end-to-end delay and the total number of packets lost for the closed-loop control of ten double-tank processes through a large mesh network of sixteen relays. Results for nodes 1, 2, 5 and 8 located at different hop levels in the network.



(a) Interference in the region close to the controller node starting at $t = 40\,\mathrm{s}$.



(b) Fault and removal of four relay nodes close to the controller node at $t = 40\,\mathrm{s}$.

Fig. 8: Evaluation under network interference and node faults.

TABLE I: Time efficiency of a $150\,\mathrm{s}$ wireless CPS simulation in GISOO

Scenario	Actual simulation time
Single plant (controller in node) - 1 relay	0 min 30 sec
Single plant (controller in PC) - 1 relay	2 min 33 sec
Single plant (controller in node) - 16 relays	0 min 36 sec
10 plants (controller in node)- 16 relays - no interference	4 min 38 sec
10 plants (controller in node) - 16 relays - faults	4 min 28 sec

2) Node faults: Fig. 8b depicts the end-to-end delay and packet loss statistics of the node fault experiment. Removal of relay nodes 50, 54, 55 and 58 which are close to the controller node 41 takes place at time $t = 40s$. Due to this failure, there is an end-to-end delay peak that occurs due to the fact that the faulty nodes were communicating the data from the selected sensors. The two packets lost, transmitted by sensor 2 and 8, were lost due to the fact that the relay nodes which were in their routing path failed, not allowing those packets to successfully arrive to the controller. Since the network topology offers many redundant paths, the nodes are able to quickly and distributively re-select a new parent from their parent list when their preferred parent becomes unavailable, using the CTP routing mechanism.

F. Time-efficiency of GISOO

Results on the total time required to run the above experiments in GISOO are presented in Table I. As it can be seen, the total simulation time increases greatly with the total

number of wireless devices that exchange data with Simulink. Additionally, the addition of wireless nodes that do not interact with Simulink have a very small impact on the simulation time.

V. Conclusion

In this paper, we have presented and evaluated a virtual testbed for wireless cyber-physical system simulation which we call GISOO. By providing the integration between Simulink and COOJA, we are able to model and simulate physical systems, design control algorithms and real wireless networks code, and evaluate their complete interaction in a single platform. A case-study of the wireless control of double-tank systems has provided insight on the capabilities of GISOO. Additionally, we validated GISOO's performance against a real wireless CPS experiment, using real wireless devices and the physical process. Future work will focus on the further development of GISOO capabilities. We aim at allowing for automated code generation of the wireless nodes code, mainly control algorithms, through Simulink. The automated code generation will be performed for TinyOS (nesC language) and for ContikiOS (C language). Additionally, we plan to create COOJA plugins to improve the speed of detection, diagnosis and analysis of relevant situations in wireless CPSs such as, ill deployments, node faults and interference devices. Furthermore, we intend to demonstrate GISOO's capabilities in other CPS scenarios. The code and manual of GISOO, together with multiple example scenarios is available in the GISOO page [19].

Acknowledgments

The authors would like to thank Fredrik Österlind for his initial inputs to this work and the help provided by Lin Wei, Antonio Gonga, Altamash Ahmed, Ziyang Li and Navid Hassanzadeh during the project is greatly acknowledged.

This work is supported by SSF through the PROMOS project, the Swedish Research Council, the Knut and Alice Wallenberg Foundation and the HYCON2 EU project

References

[1] Contiki and COOJA wiki. [Online]. Available: https://github.com/contiki-os/contiki/wiki

[2] Mathworks Simulink - Simulation and Model-Based Design. [Online]. Available: http://www.mathworks.se/products/simulink/

[3] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *9th IEEE International Conference on Industrial Informatics (INDIN)*, July 2011.

[4] A. Annaswamy, S. Chakraborty, D. Soudbakhsh, D. Goswami, and H. Voit, "The arbitrated networked control systems approach to designing cyber-physical systems," in *Estimation and Control of Networked Systems*, vol. 3, no. 1, 2012, pp. 174–179.

[5] J. Araujo, A. Anta, M. Mazo, J. Faria, A. Hernandez, P. Tabuada, and K. H. Johansson, "Self-triggered control over wireless sensor and actuator networks," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, june 2011.

[6] K. Årzén, "A simple event-based PID controller," *Preprints 14th World Congress of IFAC. Beijing, China*, 1999.

[7] K. J. Åström and M. Lundh, "Lund control program combines theory with hands-on experience," *Control Systems, IEEE*, vol. 12, no. 3, pp. 22 –30, jun 1992.

[8] M. Björkbom, S. Nethi, L. M. Eriksson, and R. Jäntti, "Wireless control system design and co-simulation," *Control Engineering Practice*, vol. 19, no. 9, pp. 1075–1086, 2011.

[9] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 16–30, Jun. 2003.

[10] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[11] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "Cooja/mspsim: interoperability testing for wireless sensor networks," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, 2009, p. 27.

[12] E. Eyisi, J. Bai, D. Riley, J. Weng, W. Yan, Y. Xue, X. Koutsoukos, and J. Sztipanovits, "Ncswt: An integrated modeling and simulation tool for networked control systems," *Simulation Modelling Practice and Theory*, vol. 27, pp. 90–111, 2012.

[13] D. Gay, P. Levis, and D. Culler, "Software design patterns for tinyos," *SIGPLAN Not.*, vol. 40, no. 7, pp. 40–49, 2005.

[14] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, November 2009.

[15] J. Hauer, "TKN15.4: An IEEE 802.15.4 MAC Implementation for TinyOS 2," *TKN Technical Report Series, Telecommunication Networks Group, TU-Berlin*, no. TKN-08-003, Mar 2009. [Online]. Available: http://www.tkn.tu-berlin.de/publications/papers/TKN154.pdf

[16] A. Hernandez and P. Park, "Ieee 802.15. 4 implementation based on tkn15. 4 using tinyos," Technical report, KTH Electrical Engineering, Stockholm, Tech. Rep., 2011.

[17] *IEEE 802.15.4 standard: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE, 2006. [Online]. Available: http://www.ieee802.org/15/pub/TG4.html

[18] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler, "Contikirpl and tinyrpl: Happy together," in *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN)*, 2011.

[19] KTH - GISOO: a virtual testbed for wireless cyber-physical systems. here you can find the simulator's code, examples and documentation. KTH - Automatic Control Lab. [Online]. Available: https://code.google.com/p/kth-gisoo/

[20] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "TinyOS: An operating system for wireless sensor networks," *Ambient Intelligence*, 2004.

[21] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[22] B. Li, Z. Sun, K. Mechitov, G. Hackmann, C. Lu, S. Dyke, G. Agha, and B. F. Spencer Jr, "Realistic case studies of wireless structural control," in *International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

[23] J. Mirkovic and T. Benzel, "Teaching cybersecurity with deterlab," *Security & Privacy, IEEE*, vol. 10, no. 1, pp. 73–76, 2012.

[24] D. Moss and P. Levis, "Box-macs: Exploiting physical and link layer boundaries in low-power networking," Technical Report SING-08-00, Stanford University, Tech. Rep., 2008.

[25] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 641–648.

[26] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," *Information Processing in Sensor Networks 2005. Fourth International Symposium on*, Apr. 2005.

[27] R. Poovendran, K. Sampigethaya, S. K. S. Gupta, I. Lee, K. V. Prasad, D. Corman, and J. Paunicka, "Special issue on cyber - physical systems [scanning the issue]," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 6 –12, jan. 2012.

[28] T. Samad, P. McLaughlin, and J. Lu, "System architecture for process automation: Review and trends," *Journal of Process Control*, vol. 17, no. 3, pp. 191 – 201, 2007.

[29] WCPS. (2013) Wireless cyber-physical simulator. [Online]. Available: http://wsn.cse.wustl.edu/

[30] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, "Rpl: Routing protocol for low power and lossy networks," Tech. Rep., March 2012. [Online]. Available: http://tools.ietf.org/html/rfc6550