Smart Grid Communication and Co-Simulation

Vincenzo Liberatore, *Member, IEEE Computer,* Ahmad Al-Hammouri

***Abstract*—The smart power grid will extensively rely on networked control to increase efficiency, reliability, and safety; to enable plug-and-play asset integration, such as in the case of distributed generation and alternative energy sources; to support market dynamics as well as reduce peak prices and stabilize costs when supply is limited. In turn, network control requires an advanced communication infrastructure with support for security and real-time communication.**

This paper reviews the major challenges in smart grid communication, and proposes *PowerNet*, a system of heterogeneous yet interoperable networks that provides adequate levels of real-time performance, reliability, and security, and that exploits current investments in software and hardware. Smart grid communication involves disparate designs and complex issues, and it can be effectively evaluated through co-simulation. The paper describes a co-simulator that combines extensive support for power device models and for communication models, and highlights current work in the area.

I. Introduction

The smart power grid will consist of intelligent devices that use advanced communication methods, which in turn will improve situational awareness and enable networked control. Networked control will increase efficiency, reliability, and safety; will enable plug-and-play asset integration, such as in the case of distributed generation and alternative energy sources; will support market dynamics as well as reduce peak prices and stabilize costs when supply is limited.

Networked smart devices currently include smart meters at the customer site and phasor measurement units (syncronophasors) in the transmission and distribution grids. In the future, smart appliances will adjust their functions depending on instantaneous electricity cost, and a host of communicating devices will enable the feedback-loop networked control of the smart grid.

In this paper, Section II will briefly review the main objectives and technical challenges for the communication infrastructure that will enable networked control of the smart grid; propose *PowerNet*, a communication framework for the smart power grid; Section III will discuss metrics and objectives. Section IV will outline an evaluation methodology based on the co-simulation of the grid physics with the communication infrastructure. We will then outline future directions.

II. PowerNet: A Smart Grid Communication Network

Smart grid communication should protect users' privacy and security. Networked control requires adequate real-time performance and reliability. Communication should use existing infrastructure whenever possible, due to (1) the large cost of deploying a new communication infrastructure, (2) the substantial investments made in the past (e.g., dark fiber) or currently underway in hardware (e.g., universal broadband) and software (e.g., secure real-time middleware), and (3) the potential for widespread technology adoption, which will result into increased grid responsiveness and larger economic benefits.

In the end, smart grid devices can communicate over a variety of substrates with different properties. Furthermore, different media may be appropriate in different circumstances. For example, smart appliances in the home can exploit existing Ethernet home local area networks (LANs), 802.11 home wireless LANs (WLANs), or powerline home networks. At the other end, if a synchronophasor is deployed in the transmission grid, it may be better served by wide-area wireless, such as a mesh network or a powerline wide-area network. Since the smart grid will exploit different types of data from a variety of devices communicating over heterogeneous networks, communication should be possible across various media, which in turn requires a convergence layer, such as the Internet Protocol (IP).

The smart grid will share pre-existing communication networks so as to exploit previous and current investments. For example, a smart dishwasher may use a home WLAN that had been previously installed to serve traditional types of network traffic (e.g., Web browsing, e-mail, and multi-media). A shared infrastructure will lead to substantial savings, to plug-and-play installation, and eventually should result into a deeper penetration of smart grid technology. However, a shared environment poses significant risks due to potential security vulnerabilities and to the potential lack of real-time Quality-of-Service (QoS). The problem can be addressed, for example, through *virtualization*, in which the communication resources are split into multiple and isolated virtual private networks (VPNs). For example, the same home network hardware (firewalls, network address translators, wireless gateways) may be logically subdivided into multiple virtual networks, each securely isolated from the others and providing its own QoS service level agreement (SLA). As a result, a smart grid communication network would preserve security and real-time performance while being completely hidden from a concurrent traditional home network even though both use the same physical infrastructure.

In conclusion, we envisage a communication network that we call *PowerNet* and that

1) Consists of heterogeneous networks that are interoperable via a convergence layer
2) Provides adequate levels of real-time Quality-of-Service (QoS), reliability, and security to support networked

V. Liberatore is with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106-7071. E-mail: vl@case.edu.

A. Al-Hammouri is with the Department of Network Engineering and Security, Jordan University of Science and Technology, Irbid, 22110, Jordan. E-mail: hammouri@just.edu.jo.

control
3) Maximally exploits pre-existing or concurrent investments in software (e.g., real-time secure middleware) and hardware (e.g., home networks), possibly through virtualization.

Since PowerNet involves heterogeneous designs and complex interoperability and security issues, it is essential that solutions be designed and evaluated carefully prior to large scale deployment.

III. Evaluation Methodology: Metrics

Real-time distributed systems have been analyzed assuming (1) purely network-related metrics, such as delays or jitter, or with (2) control-theoretical objectives, such as stability, but under simplified network characteristics. These evaluation approaches promote the isolation of different areas of expertise. The compartmentalization of research makes intricate problems more tractable, but at the same its limitation is that it can decrease the potential effectiveness of a networked smart grid, and reduces the opportunities for multi-disciplinary research. The challenge is to formulate a methodology that maps networked-oriented metrics into systems performance. An analogous process has been accomplished in Voice-over-IP (VoIP), where ITU standards associate delays, jitter, loss rates, and bandwidth with MOS (Mean-Opinion Score) ratings that denote the user's satisfaction with a certain level of voice quality. A milestone for the next 5–10 years is to direct research in disparate communities toward a shared set of metrics and goals. The objective is to create an evaluation method that is analogous to an ITU standard for VoIP and that translates network-related metrics into user-perceived performance. A promising direction is the set of benchmarks and metrics for distributed haptics: the evaluation framework aims at the semi-automatic evaluation of distributed haptics over various network configurations [1]. Another promising direction is the emphasis on disturbance rejection to evaluate play-back buffer effectiveness [2]. Additional issues include metrics for scalability, security, interoperability, and metrics' impact on requirements-services interfaces.

IV. Co-Simulation

The smart grid will consist of heterogeneous communicating devices, and it will involve complex issues in the areas of safety, security, interoperability, and performance. Consequently, it is essential that smart grid communication and control be designed and evaluated carefully prior to demonstration and large scale deployment. However, current simulation software is incapable of accurate smart grid evaluation because it lacks fine-grained models of complex grid devices or of communication protocols. On-going work is developing a co-simulation platform for the smart grid that tightly integrates fine-grained models of power grid devices and of communication dynamics.

A. Background

Several simulators make it possible to evaluate power systems, but only with minimal communication models (e.g., Modelica's Spot, Optimal's Smartgrid). Others simulators enable extensive communication protocol simulations but with no power grid component (e.g., ns-2, Opnet).

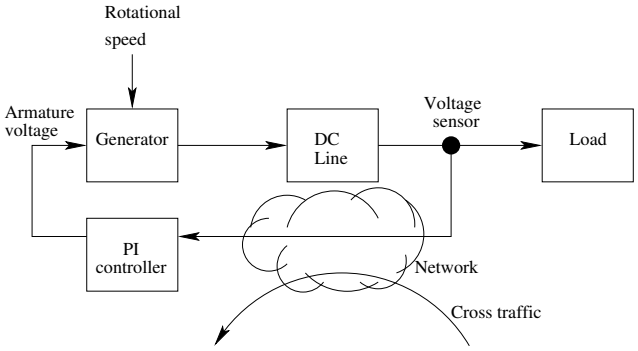

Fig. 1. Example of information flow in a simple model of a power system with networked controllers.

A *co-simulator* enables system evaluation that encompasses network communication properties (such as real-time QoS), networked control methods, sensors, actuators, and their effects on the physical world [3]. Co-simulations are essential for the evaluation of the smart grid in that the overall system dynamics depend both on network and control and on the physical properties of the power system. The Thyme co-simulator [4] has been used to test dynamic electricity pricing methods [5]. In another early project, we have built a prototype co-simulator and used it to evaluate the impact of networked control on simple smart grid scenarios [6]. Our co-simulator combines the ns-2 network simulator with a power grid simulator based on Modelica. The co-simulator benefits from the extensive array of power devices that are available in the Modelica libraries, and from the communication protocols that are available in ns-2, as well as from the extensibility and graphical interfaces of both simulators.

B. An Example

Figure 1 shows an example of simulations that are enabled by our co-simulation tool. In this particular simulation, a time-varying stochastic rotation speed is applied to a generator (i.e., a wind turbine). The generator can supply varying amounts of electric power depending on the armature voltage, which is a tunable quantity. The resulting electric power is transmitted over a DC line, that includes a model of capacitance effects. The load is also time varying. A sensor measures the voltage at an intermediate point of the transmission line. Sensor readings are transmitted back to the generation site through a communication network, which is potentially subject to failures and to congestion caused by cross-traffic. A PI controller uses the voltage readings to dynamically set the voltage on the generator armature, thereby regulating the amount of generated power. The objective of the controller is to keep the measured voltage constant, which effectively means that the generated power matches the load. A main obstacle is that communication may be subject to delays and to time-varying jitter, which make it hard to achieve real-time data delivery.

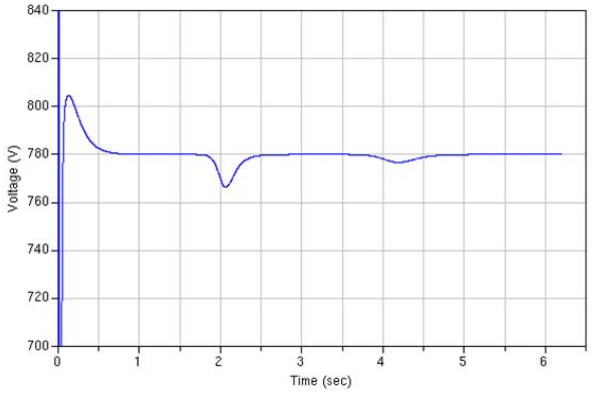

Fig. 2. Output voltage when the communication network is uncongested.

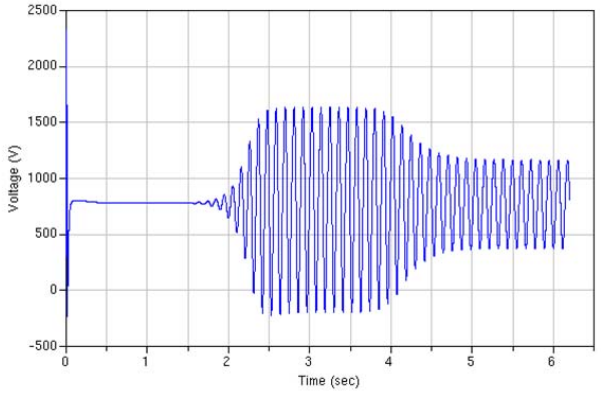

Fig. 3. Output voltage when the communication network is congested. (Vertical scale differs from Figure 2.)

Figures 2 and 3 show voltage as a function of time. The voltage is expected to track the 780V set point as the load undergoes large step changes. Figure 2 is the scenario when cross-traffic consumes less than 10% of available bandwidth in the networked control path. In this case, the voltage tracks reasonably well the set point, although some transients are visible at the time instants when the load undergoes step changes. Figure 3 is the scenario when cross-traffic uses more than 80% of the available bandwidth. In this case, the system is effectively unstable and voltage exhibits persistent oscillation of 1000-2000V.

C. Base Simulators

For the network, we chose `ns-2` [7] among other alternatives because

- `ns-2` is a free, widespread packet-level simulator.
- `ns-2` simulates the exact dynamics and events of individual packets while traversing network elements, e.g., communication links and routers.
- `ns-2` supports various routing, transport, and application protocols.
- `ns-2` is capable of simulating wired, wireless, local- and wide-area networks.
- `ns-2` is evolvable by exposing well-defined APIs that greatly facilitate developing new protocols and algorithms.

As for the electric grid, we chose Modelica [8] among other candidates because

- Modelica is a modeling language for large-scale complex physical systems.
- Modelica is an object-oriented language and supports model construction and reusability.
- Modelica allows acausal modeling.
- Several ready Modelica libraries are available for different domains, including electrical, electronic, and electric power systems.
- Several commercial and open source simulation environments are available for Modelica.

D. Inter-process Communication

1) Basic Synchronization: Modelica and `ns-2` run as two separate processes. The communication between the two processes is achieved via UNIX named pipes. In general, when combining two or more simulators, where each runs as a separate process, the major issue becomes how to synchronize their simulated clocks. To illustrate, consider the following example of only two simulators, S_1 and S_2. Both S_1 and S_2 start at simulation time t_0 and, at simulation time t_1, S_1 needs to convey data to S_2. Intuitively, we cannot let S_1 and S_2 freely run because it might happen that S_2 passes t_1 before S_1, which will lead to erroneous operations. Next, we discuss three mechanisms to synchronize S_1 and S_2.

a) Predetermined Communication Time: The race condition between S_1 and S_2 can be solved easily if the time instant t_1 is predetermined and known beforehand. In such case, we allow S_1 and S_2 to run freely until t_1, at which point they both pause, exchange data, and then resume execution until the next predetermined communication time instant t_2, and so on. However, in realistic simulations, not all time instants at which communication between the two simulators must occur are known *a priori*. For example, in most cases, the communication between the two simulators is triggered by internal events inside one or both simulators depending on meeting some conditions, which might be stochastic or even deterministic, but not known *a priori*.

b) Real-time Synchronization: Several simulators, including `ns-2`, possess the capability of synchronizing their simulated clocks with real-life time, i.e., the wall-clock time. If the two simulators have this real-time synchronization feature, the race condition and synchronization between the two simulators is completely resolved because both simulators will advance at the same rate (the wall-clock rate) and they will never outpace one another. Effectively, each simulator sees the other as a (simulated) hardware in the loop. However, this functionality is unimplemented in some simulators and is experimental and correct operation is not guaranteed in others, e.g., `ns-2` [7], [9]. Another disadvantage of real-time synchronization is that simulations will progress at the pace of real-life clocks and thus may take a long time to finish, i.e., an hour-long simulation will take exactly one real hour to finish no matter how powerful the hardware on which it is running. Consequently, this removes a major advantage of

simulation—the ability to compress long time into a shorter period [10].

2) Synchronization: In general, achieving synchronization between distributed simulators is a challenging issue. Therefore, in [6], [3], we relaxed some constraints on the requirements the combined tool must meet. In particular, while the tool fully supports communication that depend on ns−2's internal events, it does not support communication events that depend on internal events inside Modelica.

To elaborate, consider the following example. Suppose that this tool is used to simulate the simplest form of a system consisting of a single physical plant and a remote controller, such as the one in Figure 1. The sensor samples the values of physical quantities, writes them in a packet, and sends the packet over the network to the controller. The controller examines the received sample to generate a control signal that is then sent to the actuator (in this case, the generator). When we first developed the tool, we based it on common assumptions: the plant is time driven (and most often the sampling times are uniform) and the controller is event driven and its computation time is negligible or constant; see for example [11] and the references therein. So, in reference to the example of Figure 1:

- The voltage sampling events are dealt with inside ns−2 not inside Modelica. The sampling intervals can be regular or irregular. However, the next sampling time cannot depend on some quantity or variable inside a Modelica model (e.g., sending a packet once a Modelica variable crosses some threshold).
- The computation delay of a controller is assumed to be zero. Then, when ns−2 delivers the packet carrying the sampled data to the controller modeled inside Modelica, ns−2 collects the output of the controller instantly. Notice that the tool can still support the case in which the controller has a delay independent of any Modelica variable, for example, a known constant delay or a delay that changes based on a predefined trend. This delay should be "coded" inside ns−2, though.

Due to such assumptions, the design choice has been to enslave Modelica to ns−2 in that ns−2 controls and determines *all* time instants at which the communication between Modelica and ns−2 should occur. However, the converse, i.e., enslaving ns−2 inside Modelica, was ruled out because if Modelica were to control the communication events between the two simulators, the communication between Modelica and ns−2 could not depend on internal events inside ns−2. In turn, unpredictability and nondeterminisim of communication networks would be eliminated and unsupported by such design choice. In the adopted choice, on the other hand, the unpredictability of physical systems is unsupported, at least as far as inter-simulator events are concerned. We opted to support nondeterminisim in networks over nondeterminisim in physical systems for the following two reasons:

1) The time instants of data delivery from Modelica to ns−2 (e.g., a voltage reading or collecting a control signal) are often assumed predictable and can be coded inside ns−2.

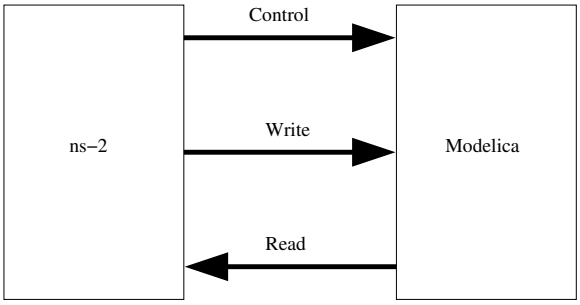

Fig. 4. The read and the write operations between Modelica and ns−2.

2) In general, the time instants when data needs to be delivered from ns−2 to Modelica (e.g., arrival of packets) are unpredictable due to random delays, presence or absence of cross traffic, uncontrolled losses that the simulated packets incur.

To make the inter-simulator communication feasible, the following detailed steps are executed. It is realized that the flow of data between Modelica and ns−2 occurs in both directions: data flows from Modelica to ns−2 and vice versa. When data is to be sent from ns−2 to Modelica, we refer to this process as a *write event* (using the ns−2 side convention). Conversely, a *read event* is when data is sent from Modelica to ns−2; see Figure 4.

1) Without loss of generality, we assume that both Modelica and ns−2 start from a common time, t_i.
2) While Modelica is pausing at t_i, ns−2 runs until the time of the first event that mandates communication with Modelica, $t_{i+1} \geq t_i$.
3) ns−2 executes the respective event, it pauses at t_{i+1}, and it instructs Modelica to run until t_{i+1}, at which point
   - If the event is a read operation, ns−2 instructs Modelica to write to the named pipe. Then, ns−2 reads the data.
   - If the event is a write operation, ns−2 instructs Modelica to read from a named pipe.
4) Steps 2 and 3 are then repeated with t_{i+1} until the end of simulation. These step are illustrated in Figure 5.

The simulation time of ns−2 is always leading that of Modelica. Note that when ns−2 is progressing in time (i.e., running), Modelica is pausing; and when Modelica is progressing, ns−2 is pausing. So, at a given time either ns−2 or Modelica is running and the other is pausing. One way to interpret this method is that, while Modelica is pausing, ns−2 is searching for events that require communication with Modelica. Modelica and ns−2 pausing mechanism is achieved by blocking reading from an empty named pipe; see Figure 4. Additionally, the method reduces the amount of parallelism potentially available in a co-simulation.

E. Current Work

Due to the fact that ns−2 determines the communication times between the two simulators, Modelica cannot determine when to deliver data to ns−2 and it is ns−2 that determines on behalf of Modelica when it should communicate with

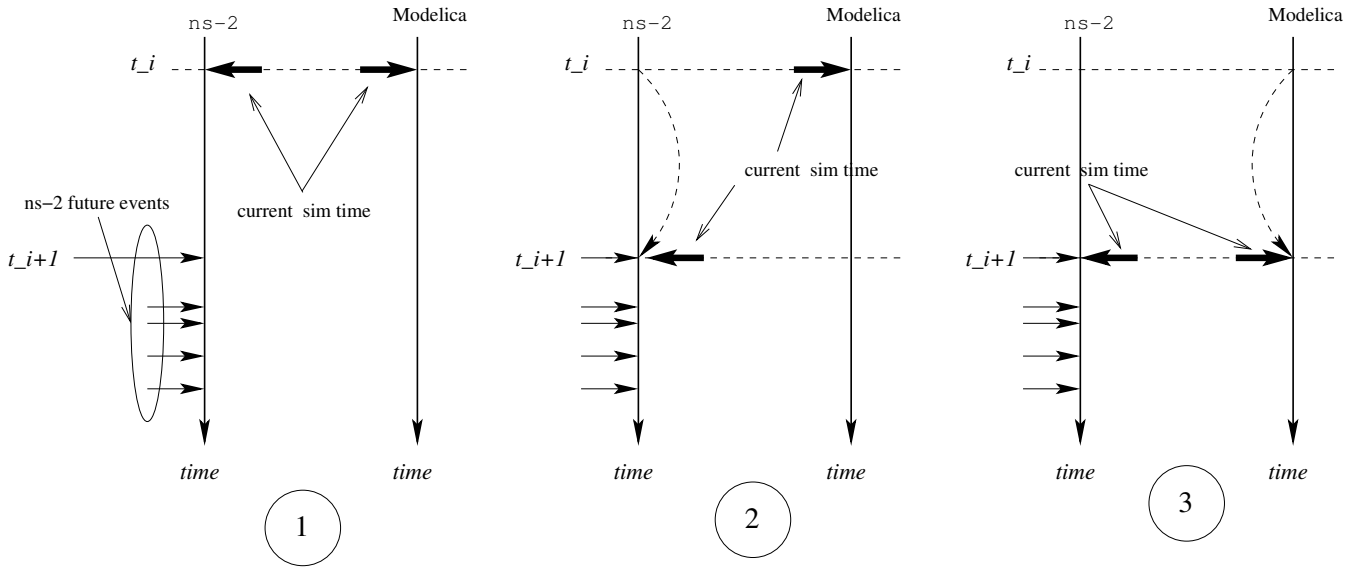

Fig. 5. Illustration of Modelica and `ns-2` pausing and progressing steps.

`ns-2`. Therefore, sending data between Modelica and ns-2 in response to events generated inside Modelica is not supported. That is, aperiodic control and alarm signals that are generated in response to events triggered exclusively inside the physical system (i.e., inside Modelica) are not accounted for. Further, the current methodology fails to fully exploit the parallelism inherent in the co-simulation approach.

Although the tool in [3] provided a very general tool that is sufficient to simulate myriad of smart grid scenarios, there is still a need for a more comprehensive tool that addresses all these limitations.

ACKNOWLEDGMENTS

Research supported in part by NSF CCR-0329910, Department of Commerce TOP 39-60-04003, and Department of Energy DE-FC26-06NT42853.

REFERENCES

[1] V. Liberatore, M. C. Cavusoglu, Q. Cai, and Y. Yoo, "Evaluation methods of a middleware for networked surgical simulations," in *14th annual Medicine Meets Virtual Reality (MMVR) conference*, 2006.

[2] V. Liberatore, "A play-back algorithm for networked control," in *Proceedings of the Twentififth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006)*, 2006.

[3] A. Al-Hammouri, M. Branicky, and V. Liberatore, "Co-simulation for networked control systems," in *2008 Hybrid Systems: Computation and Control (HSCC 2008)*, pp. 16–29.

[4] J. Nutaro, P. T. Kuruganti, M. Shankar, L. Miller, and S. Mullen, "Integrated modeling of the electric grid, communications, and control," *International Journal of Energy Sector Management*, vol. 2, no. 3, pp. 420–438, 2008.

[5] J. Nutaro and V. Protopopescu, "The impact of market clearing time and price signal delay on the stability of electric power markets," *Transactions on Power Systems*.

[6] A. Al-Hammouri, D. Agrawal, V. Liberatore, H. Al-Omari, Z. Al-Qudah, and M. S. Branicky, "Demo abstract: A co-simulation platform for actuator networks," in *Sensys 2007*.

[7] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–68, May 2000.

[8] M. M. Tiller, *Introduction to Physical Modeling with Modelica*. Springer, 2001.

[9] T. Kohtamäki, M. Pohjola, J. Brand, and L. M. Eriksson, "Piccsim toolchain—design, simulation and automatic implementation of wireless networked control systems," in *Proc. of the 2009 IEEE International Conference on Networking, Sensing and Control*, Okayama, Japan, Mar. 2009.

[10] J. Banks, J. Carson, B. L. Nelson, and D. Nicol, *Discrete-Event System Simulation*. Pearson Prentice Hall, 2005.

[11] W. Zhang, "Stability analysis of networked control systems," Ph.D. dissertation, Case Western Reserve, Aug. 2001.

Vincenzo Liberatore is an Associate Professor in CS at CWRU. He holds a Laurea degree in EE from Univ. of Rome "La Sapienza" and a Ph.D. in CS from Rutgers Univ. Previous appointments include research positions at Bell Labs and Univ. of Maryland, College Park. He has published extensively in networking, theoretical computer science, compilers, and security. Research interests include networked control systems, with applications to Internet robotics and distributed simulations. He served on the program committees of the Workshop on Factory Communication Systems and the Intl. Conf. on Mobile Data Management.

Ahmad Al-Hammouri is an Assistant Professor in the Department of Network Engineering and Security at Jordan University of Science and Technology. He received the B.S. degree with first-class honors in Electrical Engineering from Jordan University of Science and Technology, Irbid, Jordan, in 2001; and the M.S. and the Ph.D. degrees in Computer Engineering from Case Western Reserve University, Cleveland, Ohio, in 2004 and 2008, respectively. He has held research positions at Case Western Reserve University's Netlab. His Research interests are in cyber-physical systems, congestion control, and middleware for real-time sense-and-respond systems.