Editors: Barry I. Schneider, bis@nist.gov | Gabriel A. Wainer, gwainer@sce.carleton.ca

Modeling and Simulating Internet-of-Things Systems: A Hybrid Agent-Oriented Approach

Giancarlo Fortino, Raffaele Gravina, Wilma Russo, and Claudio Savaglio | University of Calabria

Everyday objects are continuously augmented with novel communication, sensing, actuation, and computation capabilities to extend their conventional use. They're generically defined as "smart" and progressively exploited in a plethora of application domains (health, transportation, manufacturing, and so on). Both the massive proliferation and global networking of such heterogeneous smart objects (SOs)[1] are pushing an epochal paradigm shift from the current human-centered Internet to the so-called Internet of Things (IoT),[2] a global ecosystem in which cyber-physical and highly pervasive services interconnect SOs, mobile devices, places, and people. IoT's market value is expected to exceed 1 trillion euros in 2020 in the EU alone, where nearly 26 billion SOs will soon impact daily life.[3]

Although 2015 is the year in which IoT gained legitimacy (www.verizonenterprise.com/verizon-insights-lab/state-of-the-marketinternet-of-things/2016), several open challenges still face full-fledged IoT realization. Such issues are mostly related to IoT's intrinsic heterogeneity, complexity, and global scale, so proper modeling paradigms and simulation approaches are crucial to achieve robust implementation and effective deployment of IoT systems.[4,5] Indeed, modeling lets IoT designers express system features at different degrees of granularity and formalization, in different perspectives and languages, abstracting them from low-level details or specific constraints (such as technology). Meanwhile, simulation lets IoT designers validate their design choices and discloses unexpected behaviors before actual system deployment, which is often time-consuming and error-prone. Even though IoT is a well-established research area, interestingly, these two aspects haven't been deeply investigated.

Software agents[6] represent a very expressive paradigm for modeling dynamic distributed systems. Their primary features (autonomy, social ability, responsiveness, proactiveness, and mobility) perfectly fit both generic and specific requirements of IoT systems, and in our view, agent-based modeling could provide even more benefits in the IoT domain than in conventional distributed environments. Although agent-based modeling captures key characteristics of SOs and IoT systems, agent-based simulators aren't suitable for dealing with certain aspects of wireless network communications, such as device density, physical network design, and coverage overlap.

This work thus proposes a hybrid approach for modeling and simulating IoT systems that envisages the exploitation of the agent-based computing (ABC) paradigm for modeling IoT systems in terms of multiagent systems (MASs) and a network simulator to investigate communication issues related to SO deployment in IoT systems. We present here our hybrid approach, which exploits the Agent-based Cooperative Smart Object (ACOSO) framework[7] and the OMNeT++ simulator,[8] as well as key guidelines for mapping an ACOSO-based SO into an OMNeT++ node and driving IoT systems design and deployment.

Background

Software agent and IoT concepts arose in very different computer ages with very different initial purposes (collaborative computation and RFID-based object traceability, respectively). As the ABC paradigm has proved to be well-suited to support modeling, design, and implementation of autonomic and cognitive IoT systems, software agents and IoT are now being jointly exploited.[9]

Smart Object-Based IoT Systems

In the literature, there are several (sometimes deeply) different definitions of IoT. The lack of any standard definition is due to the overlapping of at least three technical visions outlining a common IoT scenario approached from different perspectives.[2] The "thing-oriented" vision emphasizes the importance of IoT devices as the joining link between physical and virtual worlds. The "network-oriented" vision focuses on communication aspects to provide anywhere, anytime, anything connectivity. Finally, the "semantics-oriented" vision concerns the scalable management and effective exploitation of the massive amounts of heterogeneous data generated by IoT devices. We adhere to the thing-oriented perspective, specifically, to the SO-based IoT vision, in which SOs are fundamental, basic IoT building blocks.[1] SOs are autonomous everyday things augmented with sensing/actuation, data management, and network capabilities, as depicted in Figure 1.

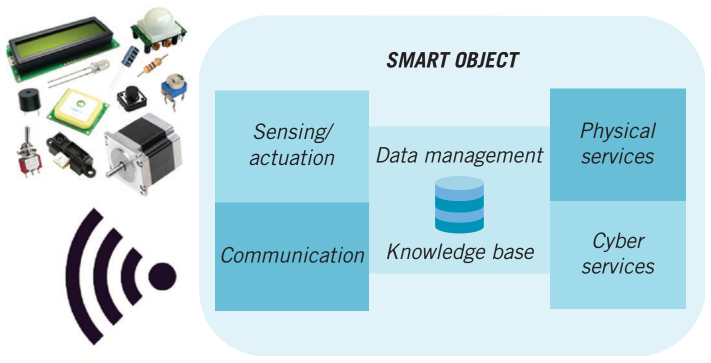

Figure 1. Smart object (SO) components.

Each of the aforementioned features is essential for making SOs both self-aware and context-aware in providing cyber-physical and ubiquitous services to human and digital users. However, with respect to traditional computer systems, SO design is notably more challenging. SOs can be constrained by limited hardware resources (RAM, storage, and CPU), physical dimensions, and market-imposed price politics. Moreover, they're technologically and functionally heterogeneous, so their clustering forms IoT systems of different scales (from a single smart home to a whole smart city), namely, loose collections of miscellaneous devices and subsystems. Before actual deployment, IoT systems must be carefully modeled and simulated even more than conventional computer systems.

Multiagent Systems

The ABC paradigm[6] is centered around the notion of agent, a well-established software artefact suitable to abstract intelligent, autonomous, adaptive, and social entities. An agent is characterized by its behavior, and it autonomously acts on behalf of its user to achieve certain goals. Moreover, agents are able to encapsulate complex functionalities hiding implementation details and to interoperate via different communication technologies simultaneously. Agents interacting in a shared environment to solve common tasks constitute loosely coupled, decentralized networks defined as MASs. Research and industrial experience in a wide range of application domains (logistics, economics, social

Table 1. Comparison of related work in modeling and simulation.

Goal	Approach	Perspective	Focus
Modeling	Service-oriented	Application level	SO service interaction
	Agent-oriented	Physical/Communication/ Application level	SO main components
	Actor-oriented	Physical/Communication/ Application level	SO interactions
Simulation	Agent-based	Application level	Collective dynamics
	Network-based	Physical/ Communication level	Networking performance
	Network & Agent-based	Physical/Communication/ Application level	Full-fledged evaluation

science, and automation science) have proven the advantages of using ABC in developing complex distributed systems as MASs.

IoT systems can be considered as loosely coupled, decentralized systems of cooperating SOs, so ABC is even more suitable for addressing their crucial requirements[5] both at system (scalability, robustness, standards compliance) and thing (interoperability, virtualization, embedded intelligence) levels. Coupling each SO with one agent (and hence treating the IoT system as a MAS) maximizes interoperability among heterogeneous subsystems and distributed resources, facilitating system modeling and development, increasing scalability and robustness, and reducing design time as well as time to market. This makes ABC suitable to effectively support IoT system development, from modeling to design to implementation.[4]

State of the Art

Most existing IoT system models result from the IoT-A project (www.iot-a.eu), which envisages a technological-agnostic and application-neutral reference architecture to be further specified in different domains (information domain, functional domain, and so on). The service-oriented computing paradigm has inspired a set of models focused on essential service-related IoT entity features.[10] In this approach, IoT entities are considered as black boxes whose functionalities can be discovered and interfaced through (textually described) services. In contrast, the ABC paradigm allows for fine-grained modeling because most SO features can be described through agent-related concepts. SO functionalities can also be expressed in terms of goals, SO working plans in terms of behaviors, SO augmentation devices in terms of dynamically bindable agent resources, and so on. In one approach,[11] an IoT-A-like agent-oriented SO model and a cognitive management framework focus on real-world object virtualization and functional composition. At a higher degree of detail,[12] actor-oriented models along with a well-defined semantic can describe cyber-physical systems in terms of their concurrent and parallel interactions. Indeed, such a fine-grained approach lets us focus on the intrinsic complexity, heterogeneity, and sensitivity to timing that characterize these systems.

Further in the development phase, simulation supports system evaluation, design verification, and optimization. Because IoT systems are collections of heterogeneous yet interacting SOs subjected to a variety of contingent factors, a wide range of possible design choices must be explored and validated before the actual deployment phase. However, no IoT-specific simulators are currently available. Agent-based simulators, being centered on high-level agent abstraction, don't directly address issues that characterize SO-to-SO interactions in a physical environment (such as limited computational and energetic resources, network congestion due to SO density, interference obstructing wireless communications, and so on). Agent-based simulations have been exploited[13] to inspect high-level issues, such as increased collective dynamics and behavioral patterns in IoT systems, assuming that agents are deployed in aseptic environments without any connectivity issues. In contrast, conventional network simulators allow an effective and detailed management of low-level communication features.[14] Specifically, such simulators let us validate network design choices and deeply analyze network performance, but they're usually exploited in application-agnostic scenarios. To exploit the benefits of both approaches, network and agent-based simulators have been jointly exploited,[15] but the actual validity of such combinations is hard to verify.

Table 1 highlights the related works presented in this section.

A Hybrid Agent-Oriented Approach

Our proposed hybrid IoT system modeling and simulation approach is based on ACOSO,[7] our agent-oriented middleware, and on the INET extension for the OMNeT++ network simulator.[8] ACOSO has been specifically conceived to model SO functionalities (sensing, actuation, communication, data management, and service provision) and requirements (abstraction of heterogeneous devices, communication patterns, knowledge bases), whereas INET allows us to simulate wireless sensor networks (WSNs) on the open source OMNeT++ platform. To support IoT system designers during the transition (see Figure 2) from ACOSO-based modeling to OMNeT++-based simulation, mapping guidelines are provided later.

ACOSO

ACOSO fully supports the development, management, and deployment of cooperating SOs in any IoT scenario that requires distributed computation, proactivity, knowledge management, and interaction. The ACOSO-based SO metamodel, depicted in Figure 3, represents one of the cornerstones of ACOSO. Each ACOSO-based SO is abstracted in a cooperating agent characterized by its set of tasks, namely, event-driven and state-based components modeling agent behaviors and goals. Tasks describe both operations required for the agent life-cycle management (SystemTask) and operations defining application-specific SO services (UserDefinedTask), and they're coordinated by a TaskManagement subsystem (TMS). The TMS exploits SO computation, communication, sensing/actuation, and data management components by interacting with the CommunicationManagement subsystem (CMS), DeviceManagement subsystem (DMS), and KnowledgeBaseManagement subsystem (KMS).

The DMS, through multiple DeviceAdapters, handles the SO augmentation devices that enable SOs to interact with the physical world. The CMS provides communication services between agents and external entities. Different kinds of interactions (for example, intra-agent FIPA-ACL-based or inter-entities UDP/TCP-based communications) are enabled via different CommunicationAdapter components. The KMS exploits local or remote knowledge bases to handle information pertaining to the SO, its current status, inference rules, and other useful data that can be shared among the agent tasks.

OMNeT++ and INET extension

OMNeT++ is an open source discrete event simulation platform that mainly aims to simulate communication among entities of distributed computer systems. OMNeT++ is fully programmable and modular, and it follows a reusable, component-based approach to build up complex and customizable network scenarios. An OMNeT++ node is composed of multiple compound and simple

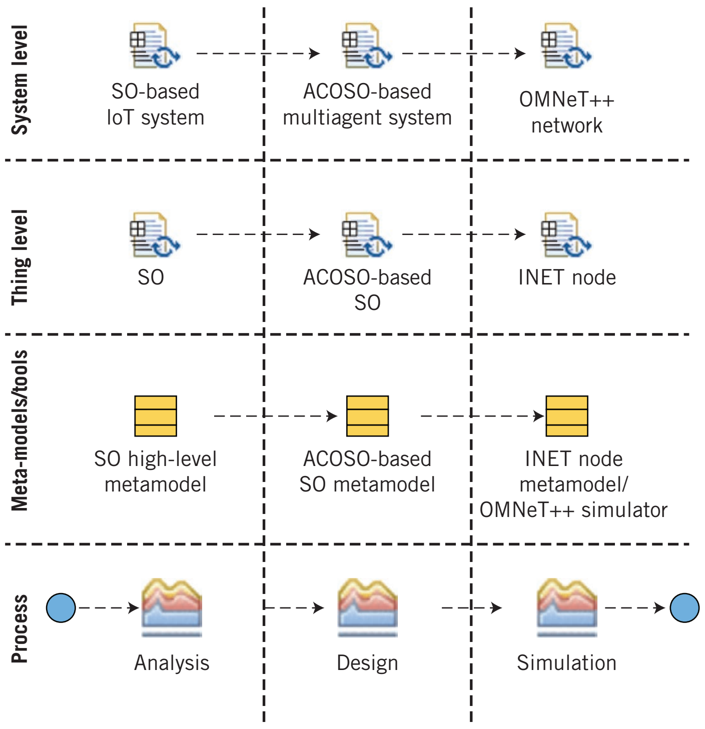

Figure 2. IoT systems agent-based modeling and network-based simulation.

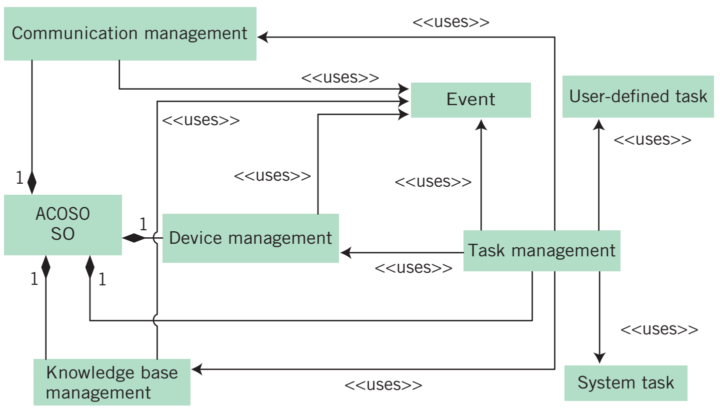

Figure 3. ACOSO SO metamodel.

modules interacting through the message-passing paradigm, while settings, properties, and data can be retrieved from different configuration files. Several extensions can be integrated to simulate specific typologies of networks. In particular, we exploited INET, an extension suite that introduces specific protocols and models for WSNs. An INET compound module representing a generic wireless node comprises modules that are grouped in packages, organized according to a simplified ISO OSI model and interacting through cMessages (a class

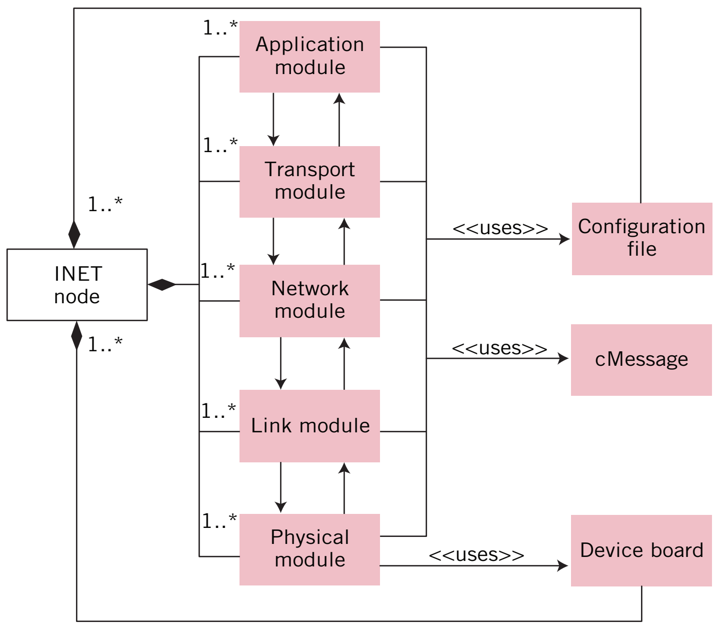

Figure 4. INET node metamodel.

of message objects representing events, messages, or jobs in a simulation), as depicted in Figure 4. In addition, different device boards can be plugged in at the physical level to manage INET node components ranging from wired/wireless physical interfaces and radio antenna to sensors and actuators.

Mapping Guidelines

SO features can be equivalently represented via the ACOSO-based SO and the INET node metamodels. However, the transition from an agent-based SO to an INET node must be guided by a preliminary mapping phase. Table 2 shows these guidelines for the transition from modeling to simulation for IoT system designers; the definition of automatic translation rules is beyond this article's scope.

Indeed, both ACOSO-based SOs and INET nodes are state-based entities whose evolution and interactions are driven by messages (ACOSO Events and INET cMessages, respectively) flowing among their components. In both cases, these components constitute modular and versatile architectures and allow the implementation of SO-based IoT systems of different scales in different application scenarios by just reusing or re-implementing some pluggable blocks. In particular, specific SO services can be implemented via UserDefinedTasks within ACOSO-based SOs as well as through application level modules in INET nodes. New SO communication protocols can be introduced by developing a related CommunicationAdapter in the ACOSO CMS or by adding a new transport/network/link module to the INET node stack. Likewise, SO sensing/actuation devices can be managed through the ACOSO DMS and its DeviceAdapters or through specific device boards connected to the INET node's physical level. Finally, SO data coming from computation or communication activities as well as SO configuration setups and parameters are managed by the ACOSO KMS while they're stored and queried into configuration files in the INET node.

IoT System Communication Analysis

The act of simulating IoT systems, notably complex and heterogeneous ones in terms of SO population, density, and adopted communication paradigms, is a challenging task. In addition, factors unrelated to applications but specifically associated to networking (such as traffic congestion, wireless signal attenuation and coverage) can influence SO interactions and service provision/fruition. To address this, and considering that we are specifically interested in the communication among SOs, we use INET to perform application-agnostic simulations, a choice that provides generality to the obtained results because they aren't bounded to any specific application contexts. Considering that network congestion can increase with SO population and that the proximity of multiple SOs can cause signal interferences in wireless communications, our simulations took into account the number of involved SOs (#SOs), their distribution in a different number of subnetworks (#subnet), and the deployment area (supposed to be squared) in which they're located. To better support simulations organization, we set three types of scenarios, referred to as small, medium, and large, as described in Table 3.

To exemplify, a small system can be a smart home or a body area network, a medium system can be a smart parking lot or a smart building, and large systems can be a smart farm or a smart port. In our simulations, in addition to the #SOs and #subnet, the essential difference between medium- and large-scale systems is the overlap in their subnetworks. It's worth noting that the simulation activity aims to provide useful indications driving system design and deployment choices rather than a punctual quantitative results evaluation. Our simulations focused on the information exchange phase, in which nodes send and receive messages following either a client/server (C/S) or a

Table 2. Mapping guidelines.

SO functionality	ACOSO-based SO	INET node	Rationale
Service provision	TaskManagement subsystem (TMS)	Application level module	Application logic defining specific SO services can be implemented within ACOSO UserDefinedTasks coordinated by the TMS or within INET modules located at the application level
Communication	CommunicationManagement subsystem (CMS)	Physical to transport level module	Communication with SO itself or external entities can be carried out through ACOSO events managed by the ACOSO CMS (and its CommunicationAdapters) or by INET cMessages among modules of every level
Sensing/ actuation	DeviceManagement subsystem (DMS)	Physical level module	SO sensing/actuation devices can be managed by the ACOSO DMS (and its DeviceAdapters) or at physical level module in the INET node
Data management	Knowledge Base Management subsystem (KMS)	Configuration files	SO data can be stored and queried through database managed by the ACOSO KMS or in configuration files in the INET node

Table 3. Simulation scenarios.

Scale	#SOs	#subnet	Overlap	Grid area (m^2)
Small	20 to 100	1	n/a	$<<$ 10.000
Medium	50 to 500	2 to 10	Yes	$\geq$ 10.000
Large	50 to 1,000	5 to 20	No	$>>$ 10.000

peer-to-peer (P2P) paradigm and use both reliable (TCP-based) and unreliable (UDP-based) transport protocols. The round-trip time (RTT) and packet delivery ratio (PDR) are measured via deterministic (1 pk/s and 10 pk/s) and stochastic normal (with 0.5/0.05 mean and 0.2/0.02 variance) data generation models. For the sake of clarity, however, the plots highlighted here have been selected to show only the most interesting results.

The plots, in particular, show PDR and RTT results obtained by varying the #SOs and the #subnets (for medium- and large-scale scenarios), considering the stochastic normal data generation model (0.5 mean and 0.2 variance); the nondeterministic stochastic transmissions model realistic situations better, while the choice of 1 pk/s is a reasonable tradeoff that fits the vast majority of applications. Simulations have been averaged over 50 runs of 15 minutes each, with standard deviation values within the range of 1 to 3 percent for PDR, and 0.02 to 0.08 seconds for RTT. Figures 5a and 5b are related to the small-scale scenario; Figures 5c and 5d are related to the medium-scale scenario composed of partially overlapping subnetworks; and Figures 5e and 5f are related to the large-scale scenario composed of non-overlapping subnetworks. In all the reported simulations, we considered subnetworks comprising 50 SOs deployed into a squared area of 10.000 m^2, which simplifies the results analysis.

With respect to the small-scale scenario, Figures 5a and 5b show that the increase in the SO population adversely affects RTT, especially in the case of reliable transmission protocol. Such phenomena are particularly noted when SOs exceed 50 units, while the increase in SOs reduces PDR only in the case of unreliable protocols. With respect to the medium-scale scenario, Figures 5c and 5d highlight the fact that the overlap in an increasing number of subnetworks in a certain area adversely affects both RTT and PDR. It's particularly evident in the case of 100 SOs deployed in two subnetworks (medium scale) as the related PDR values are notably improved with respect to the same

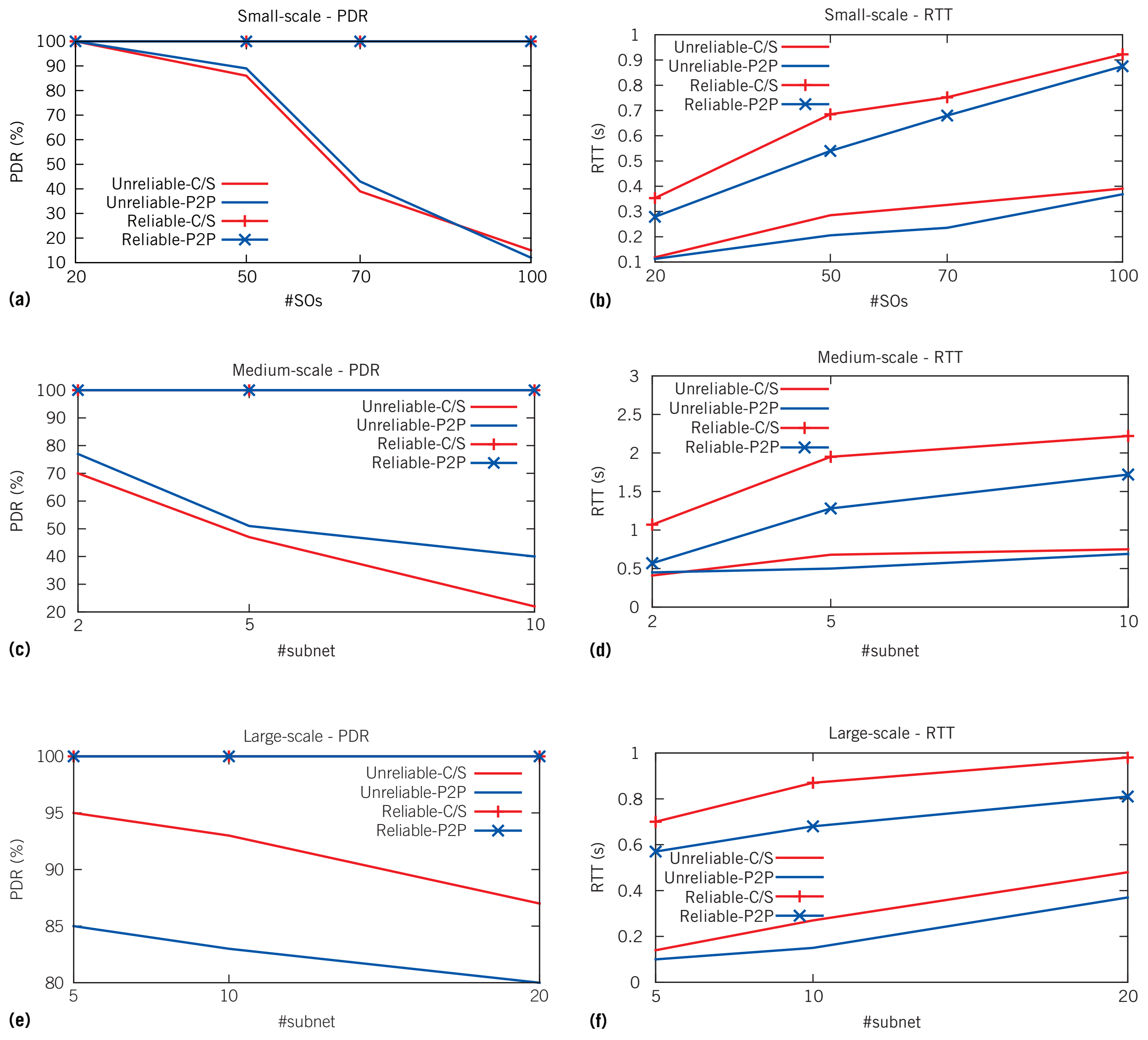

Figure 5. Simulation results: (a) small-scale PDR, (b) small-scale RTT, (c) medium-scale PDR, (d) medium-scale RTT, (e) large-scale PDR, and (f) large-scale RTT.

number of SOs deployed in a single network (small scale). Accordingly, other (not shown in this article) simulations highlighted that traffic is better balanced and performance increases by fixing #SOs and increasing #subnet. In the large-scale scenario, because subnetworks aren't overlapped, the absence of mutual interferences stabilizes the PDR better than in the medium-scale scenario and with values similar to the small-scale scenario if considering the 50 SOs case. Again, the reliable protocol implies greater RTT, whose values are comparable to the small-scale single network scenario's values. Conversely, performance is significantly different with respect to the medium-scale scenario, with notably higher PDR and lower RTT.

In all the simulated scenarios, including the ones reported in the plots, the P2P paradigm outperforms the C/S by avoiding the bottleneck effect on the nodes; the choice of a deterministic data generation model implies no substantial worsening in terms of PDR and RTT with respect to the stochastic normal one; and the choice of a higher packet generation rate (10 pk/s) contributes to network congestion with slightly higher RTT values but unchanged overall trends. The simulations highlight some indications that are valid across all the considered scenarios, regardless of specific scale:

- *SO design*. An application-driven approach should be followed to choose the most suitable communication protocols and settings. Any SO configuration setting can be accurately evaluated only in relation to the application and its specific context, physical deployment environment, functional and nonfunctional requirements, resource availability, and so on. As general guidelines, reliable protocols should be avoided in time-sensitive application contexts because they guarantee best results in terms of PDR but imply higher values of RTT and should be adopted only when full reliability is a mandatory requirement. When applicable, P2P is preferred: indeed, performance is adversely affected by an unbalanced traffic load, which could be due to the adoption of centralized communication paradigms like C/S. Finally, packet generation rate should be set as low as possible to reduce network traffic.
- *Network design*. It should be preferable to design IoT systems by minimizing the number of involved SOs. Indeed, when the SO number increases, performance worsens: the concentration of a high number of SOs in the same area causes wireless interference and network congestion. In the case of overlapping subnetworks, control packets can likely flow to the wrong subnetworks, thereby causing mutual congestion, hence, an even more careful network design and deployment is recommended to avoid performance degradation. If there are no overlaps among the subnetworks (as in the large-scale case), communications are scalable and provided performance is improved. Because scalability is a crucial issue in IoT, it's particularly important to take such network design indications into account.

Although pursuing different goals, the ACOSO framework and OMNeT++ simulator complementary supported IoT systems development at different phases and degrees of granularity; the definition of well-formalized and automatic translation rules is an ongoing work. The insights provided by our simulations underline that an application-driven approach is required to perform the best design choices both at the SO and network levels. Due to the wide range of potential configurations related to IoT system design (communication pattern, protocol and parameter settings, augmentation devices, and so on), as well as those concerning IoT system deployment, the proposed hybrid approach based on jointly exploiting agent-oriented modeling and network-based simulation results is particularly suitable to tackling intrinsic IoT system complexity. ■

Acknowledgments

This work has been carried out under the framework of INTER-IoT, Research and Innovation action - Horizon 2020 European Project, Grant Agreement n.687283, financed by the EU.

References

1. G. Kortuem et al., "Smart Objects as Building Blocks for the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 1, 2010, pp. 44–51.
2. J. Gubbi et al., "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, 2013, pp. 1645–1660.
3. C. MacGillivray, V. Turner, and D. Lund, "Worldwide Internet of Things (IoT) 2013–2020 Forecast: Billions of Things, Trillions of Dollars," *IDC*, vol. 243661, no. 3, 2013, pp. 1–22.
4. G. Fortino, W. Russo, and C. Savaglio, "Agent-Oriented Modeling and Simulation of IoT Networks," *Proc. Federated Conf. Computer Science and Information Systems*, 2016, pp. 1449–1452.
5. G. Fortino, C. Savaglio, and W. Russo, "Simulation of Agent-Oriented Internet of Things Systems," *Proc. 17th Workshop Objects to Agents*, 2016, pp. 8–13.
6. M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2009.
7. G. Fortino et al., "An Agent-Based Middleware for Cooperating Smart Objects," *Proc. Int'l Conf. Practical Applications of Agents and Multi-Agent Systems*, Springer, 2013, pp. 387–398.
8. A. Varga et al., "The OMNET++ Discrete Event Simulation System," *Proc. European Simulation Multiconference*, vol. 9, no. S 185, 2001, p. 65.
9. L. Jarvenpaa et al., "Mobile Agents for the Internet of Things," *Proc. 17th Int'l Conf. System Theory, Control and Computing*, 2013, pp. 763–767.
10. M. Thoma et al., "On IoT-Services: Survey, Classification and Enterprise Integration," *Proc. IEEE*

Int'l Conf. Green Computing and Communications, 2012, pp. 257–260.

11. P. Vlacheas et al., "Enabling Smart Cities through a Cognitive Management Framework for the Internet of Things," *IEEE Comm.*, vol. 51, no. 6, 2013, pp. 102–111.
12. P. Derler, E.A. Lee, and A.S. Vincentelli, "Modeling Cyber–Physical Systems," *Proc. IEEE*, vol. 100, no. 1, 2012, pp. 13–28.
13. S. Karnouskos and T.N. De Holanda, "Simulation of a Smart Grid City with Software Agents," *Proc. 3rd UKSim European Symp. Computer Modeling and Simulation*, 2009, pp. 424–429.
14. L. Costantino et al., "Performance Analysis of an LTE Gateway for the IoT," *Proc. IEEE Int'l Symp. World of Wireless, Mobile and Multimedia Networks*, 2012, pp. 1–6.
15. G. D'Angelo, S. Ferretti, and V. Ghini, "Multi-Level Simulation of Internet of Things on Smart Territories," *Simulation Modelling Practice and Theory*, 2016, pp. 3–21.

Giancarlo Fortino is a professor of computer engineering at the University of Calabria, Italy, and also adjunct professor of computer engineering at Wuhan University of Technology, China. His research interests include distributed computing, wireless sensor networks, software agents, cloud computing, and the Internet of Things. Fortino received a PhD in computer engineering from the University of Calabria. He's a senior member of IEEE. Contact him at g.fortino@unical.it.

Raffaele Gravina is an assistant professor of computer engineering at the University of Calabria, Italy. His research interests are focused on body sensor networks, physiological signal processing, and the Internet of Things. Gravina received a PhD in computer engineering from the University of Calabria. He's a member of IEEE. Contact him at r.gravina@dimes.unical.it.

Wilma Russo is a full professor of computer engineering at the University of Calabria, Italy. Her research interests include distributed computing and systems, software agents, multimedia networks, and the Internet of Things. Contact her at w.russo@unical.it.

Claudio Savaglio is a PhD student in computer engineering at the University of Calabria, Italy. His research interests include the Internet of Things, network simulation, and agent-oriented middleware. Savaglio received an MSc in computer engineering from the University of Calabria. Contact him at csavaglio@dimes.unical.it.