Modeling and Simulation of Mobile Gateways Interacting with Wireless Sensor Networks

F. Fummi[§], D. Quaglia[§], F. Ricciato[‡], M. Turolla[‡]
[§]Dipartimento di Informatica, Università di Verona, strada le Grazie 15, I-37134, Verona, Italy
[‡]Telecom Italia Lab, Via G. Reiss Romoli 274, I-10148, Torino, Italy
[franco.fummi|davide.quaglia]@univr.it, [fabio.ricciato|maura.turolla]@tilab.com

Abstract

Sensor networks are emerging wireless technologies; their integration with the existing 2.5G, 3G mobile networks is a key issue to provide advanced services, e.g., health control. However this integration poses new challenges in the design and simulation of the involved embedded systems since it requires the cooperation of simulation tools that model hardware, software, and network aspects and their interactions. We present the modeling and simulation of a network scenario, core of a telecom provider's future portfolio, in which an ARM-based mobile handset is used as the gateway between a wireless sensor network (WSN) and remote users through a wide area network (WAN). Initially, the gateway and the WSN are modeled at system level with SystemC while the wide area network is modeled with NS-2. Then, HW/SW partitioning is performed on the gateway and an instruction set simulator of the ARM processor is used for the cycle-accurate execution of the RTOS and the application software.

1 Introduction

Wireless sensor networks (WSN) appear a promising tool to build ambient intelligence systems. Past research effort has been primarily focused on internal issues like routing, self-organization, MAC layer design, and collaborative data processing [7, 12, 13]. Now telecom providers are ever more interested in the interconnection of WSN's with traditional communication networks to provide value-added services through the interaction with a large number of remote users and service centers. The potentialities of this approach are not yet investigated completely but a possible application is a monitoring system which sends alarms to remote users when a dangerous event happens.

Fig. 1 shows the scenario modeled in our work. The wireless sensor network consists of heterogeneous nodes connected in ad-hoc manner through short-range radio links; some nodes are equipped with environmental sensors while others simply aggregate and relay data from sources to destinations. The gateway is a mobile phone equipped with two radio interfaces; from one side it is connected to the WSN to inject queries and collect data; from the other side it exchanges voice and data through the traditional mobile infrastructure which is part of the wide area network. The wide area network consists of a number of links of different capacity and delay managed by one or more telecom providers. It is responsible for connecting the gateway to geographically distributed users and Internet hosts to deliver events, log monitored data and retrieve information from servers. Therefore, the gateway is the center of a so-called *heterogeneous network*.

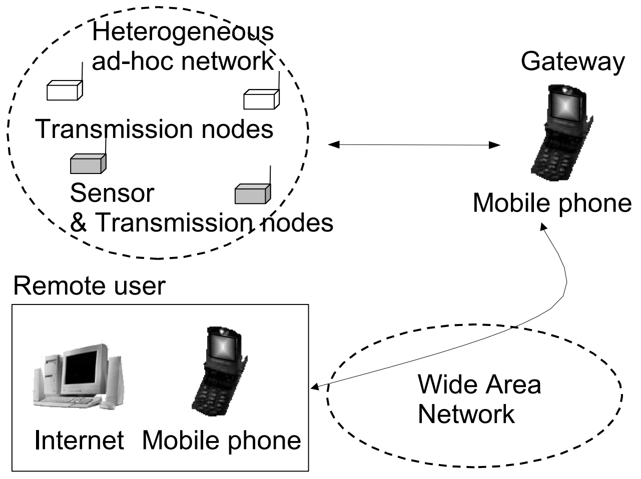

Figure 1. Network of heterogeneous nodes.

The design flow of the gateway requires the co-design and co-simulation of hardware, software, and network aspects in order to achieve energy efficiency, quality-of-service support, reliability, and network scalability. The traditional simulation approaches [10, 9, 11] often emphasize only one of these three aspects because of the difficulty to merge different knowledge domains. To fill this gap, this work applies an alternative HW/SW/network co-design and co-simulation methodology, based on the integration of domain-specific simulation tools, i.e., NS-2 [10] for network simulation, SystemC [6] for system-level simulation and hardware description, and an instruction set simulator [5] to run the operating system and the application software.

The paper is organized as follows. Section 2 describes the different aspects to be modeled in the above scenario and the available tools. Section 3 shows a recent methodology to combine different domain-specific simulation tools and thus to provide a complete simulation framework. In Section 4 the proposed modeling approach for the above scenario is presented in detail. Finally, conclusions are drawn in Section 5.

2 Background

Efficient modeling and simulation of networked systems require that tools exhibit a good level of scalability, completeness, fidelity, and reusability. The simulator should be able to handle large networks of thousands of nodes in a wide range of configurations (scalability). It should be able to cover as many system interactions as possible, accurately capturing behavior at a wide range of levels (completeness) and revealing unanticipated interactions, not just those a developer suspects (fidelity). Finally, the simulator should bridge the gap between algorithm and implementation, allowing developers to test and verify the code that will run on actual hardware (reusability).

Different aspects should be addressed during the modeling and simulation of networked systems. They can be classified according to three domains, i.e., software, hardware, and network.

2.1 Software domain

The characteristics to simulate in the software domain are: the functional and timing behavior of the software and its interaction with external events through interrupts (e.g., the presence of concurrency issues). While the functional behavior of a system can be easily simulated through general-purpose languages such as C or C++, other characteristics can be reproduced only by a cycle-accurate emulation of the CPU through an instruction set simulator [5] and the support of debug facilities.

The instruction set simulator (ISS) is an application which runs on a host workstation and executes programs written and compiled for a different processor (target platform) [5]. ISS simulates the behavior of a program and the associated operating system at the instruction-set level; simulation is cycle-accurate, i.e., the number of simulated instruction cycles to perform a given operation is the same as on actual hardware. This tool can be used to verify the interactions between the application and the operating system and, if a power model of the CPU is available, to evaluate power consumption. Using this tool, developers can test and verify the *same object code* that will run on actual hardware.

Simulations performed by ISS lack realistic timing information since instruction cycles, not seconds, are the basic time unit. For this reason, this tool cannot be used to model asynchronous events triggered by hardware components or by the network.

2.2 Hardware domain

Also in this domain, the functional behavior of the system should be reproduced at the first design stage. Then, the tool should allow to refine the description to represent the architecture as a set of interconnected blocks (structural view). In this flow, non-functional information should be managed, e.g., timing behavior, area utilization and power consumption. A desired feature for a HW simulation tool is its support for the synthesis of the architecture.

A traditional language for hardware description is VHDL while SystemC [6] is gaining increasing attention for its great flexibility in describing devices at different abstraction levels, from system level down to RTL and gate levels. SystemC is a C++ class library that provides the constructs required to model system architectures including hardware timing, concurrency and reactive behavior that are missing in standard C++. In literature SystemC was already used to describe network-on-chip architectures [2] and to simulate the lowest network layers of the Bluetooth communication standard [1].

2.3 Network domain

Network can be modeled at different levels of detail, from packet level down to the electromagnetic propagation. Simulated values can be either generated by an analytic model or taken from experimental data sets; the first approach is more general but it strongly depends on the model validity and may be computational intensive.

Network Simulator, NS-2 [10], is the most widely used discrete event simulator for computer networks. It is written in C++ and provides modules for the simulation of well-known protocols both wired and wireless. NS-2 simulates networks at the packet level and provides facilities to collect statistics at different detail levels. Some extensions have been developed to simulate sensor networks for environmental monitoring applications [12, 13, 11]. The main weakness of NS-2 for the simulation of ad-hoc wireless networks is that it does not model concurrent processes within the network node. With NS-2, simulation scenarios are created by connecting together different kind of objects, i.e., nodes, agents and applications describing different layers of the ISO/OSI model. Since a cross-layer approach is preferred in the design of wireless sensor networks, NS-2 should be deeply modified to exploit the interaction between protocols and applications. Besides, implementing a new protocol requires the update of a lot of NS-2 configuration files.

Some specific tools were developed in the past for the simulation of wireless sensor networks (e.g., TOSSIM [9], AVRORA, EMSTAR, ATEMU, SQUALNET) even if most of them are targeted to a specific architecture (e.g., Berkeley's motes). A common drawback of these approaches is that they do not offer a direct path to the implementation, e.g., hardware synthesis. Using different tools for modeling and implementation limits the reuse of code and testbenches. Although this issue can be tolerated in today's wireless systems often designed using off-the-shelf hardware components, the high integration of next-generation networked embedded systems will require that hardware design and network simulation will be applied on the same models.

3 Co-simulation

Table 1 summarizes the main features of the tools described above. It can be seen that domain-specific tools do not provide all the capabilities required for a comprehensive simulation of the heterogeneous network depicted in Fig. 1. Specific simulators for WSN (e.g., TOSSIM) and NS-2 do not provide mechanisms for HW description; ISS models software with high fidelity but should be combined with other tools. SystemC is the most versatile tool but it does not provide cycle-accurate emulation of the CPU and lacks models for Internet protocols. This context suggests an alternative approach in which different domain-specific tools are combined to perform a synchronized simulation of the different aspects of the problem as in [4].

3.1 SystemC/ISS co-simulation

Two new types of ports `iss_in` and `iss_out` have been added to the SystemC kernel. From the point of view of the software running on the ISS, these ports behave as memory-mapped registers. Each port has a 32 bit address; when the software performs a write operation in that location, the value is delivered to the corresponding port and the associated sensitive method is triggered as in traditional write operations. Conversely, values written to an `iss_out` port can be read by the software at the corresponding location. Since memory-mapped I/O is a common approach in embedded system programming, software running on the ISS can be the same code that will be executed on the actual platform and this fact contributes to shorten the time-to-market. The SystemC kernel was modified to handle the presence of these special ports and processes and to communicate with the ISS through a network socket. The ISS was modified to re-direct read and write operations at a given set of addresses to the SystemC kernel.

Tool	HW	SW	Network
WSN simulators	no	yes (note 1)	yes (note 2)
NS-2	no	yes (note 1)	yes
ISS	no	yes	no
SystemC	yes	yes (note 1)	yes (note 2)
VHDL	yes	no	no

Table 1. Comparison of the described tools as a function of the various aspects to be simulated in heterogeneous networks. NOTE: (1) no cycle-accurate emulation of the CPU, (2) Internet protocols not supported.

3.2 SystemC/NS-2 co-simulation

Both simulators have conventional event-driven kernels; they schedule the execution of events in non-decreasing order of timestamp. In SystemC, events are associated to read and write operations on ports, while in NS-2 events are associated to packet transmission and reception. In our work we follow the approach from [3] in which special types of SystemC ports and a custom NS-2 agent allow the interaction between the corresponding models. A packet can be moved from a SystemC module to an instance of the special NS-2 agent by writing it on a special output port. Conversely, a packet arrived to the agent from the network is delivered to a special SystemC input port and triggers the associated sensitive method as traditional write operations. Both kernels have been modified to interact each other through a network socket in order to reach a global synchronization of events and to exchange data.

3.3 Simulation of wireless sensor networks

In this work we also evaluated the potentiality of SystemC in the modeling and simulation of wireless sensor networks. In the proposed simulator, devices are modeled in SystemC and their instances are connected to a module that reproduces the behavior of the wireless channel; propagation delay, radio interference, collisions and path loss are taken into account by considering the spatial position of nodes and their on-going transmissions. Even if these physical aspects are considered, simulation is performed at packet-level to speedup computation. Support for carrier sense medium access control (MAC) is also provided by the simulator. Nodes have a set of attributes that can be changed during the simulation; position can be changed to model mobile topologies; each node can switch its status from RUN to OFF to model failures; the use of the SLEEP status and the variation of the transmission power and rate can be exploited to specify and evaluate power-saving strategies. The design of the node can be dealt at different abstraction levels: from system/behavioral level (transaction-level model) down to register transfer level (RTL) and gate level. After each refinement step, nodes can be tested in their network environment to verify that design constraints are met. Synthesis can be directly performed on those models provided that they are described using a suitable subset of the SystemC syntax. Different kinds of node can be mixed in the network and many instances of the network module can be created with different values for channel parameters to build complex systems consisting of many sub-networks of heterogeneous devices.

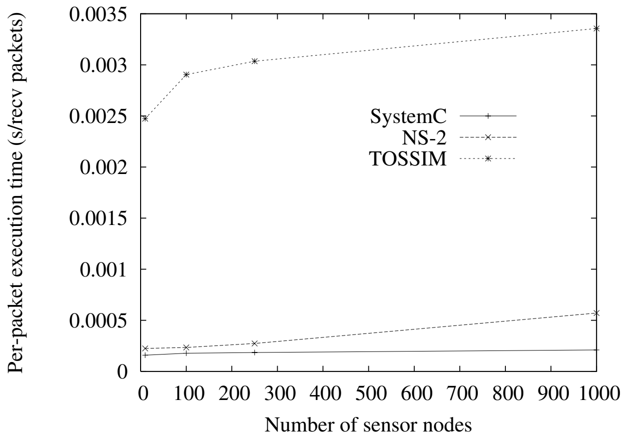

Figure 2. Per-packet execution time as a function of the number of nodes for different simulation tools.

4 Test case

To show the proposed modeling flow we designed a simple test application for the scenario depicted in Fig. 1. WSN nodes receive stimuli from the environment in the form of integer values. When a stimulus is received, the corresponding node sends a packet reporting its value, the location of the node and the timestamp of the reception. Stimuli are generated every 1 s. Since WSN's do not provide direct node addressing, a routing algorithm and multi-hop transfers are needed for data delivery from sources of stimuli to the gateway; for simplicity's sake received packets are re-sent in broadcast by all nodes except the gateway; more complex routing algorithms can be used [7].

The gateway collects data from the received packets and creates a table in which, for each stimulus location, the average of the sampled values is reported. Every 10 s the table is delivered over the Internet to a remote host.

4.1 Model of the WSN

The WSN is completely modeled in SystemC using the simulator described in Section 3.3. This tool enables the modeling of a set of heterogeneous nodes and favorites cross-layer design and synthesis. Furthermore, during preliminary tests, it provided a higher simulation speed with respect to TOSSIM and NS-2 as shown in Fig. 2.

A sub-class of the `Node` module is created to implement the specific relaying functionality. In this class, the reception of a stimulus or of a packet triggers the execution of appropriate `SC_THREAD` methods which put a new packet into a queue. Another method performs carrier sense and transmits enqueued packets when the channel is free. An instance of this class has been created for each node of the WSN. Sources of stimuli are represented by instances of a sub-class of the `Stimulus` module in which a clocked `SC_THREAD` generates a random value every 1 s.

4.2 Model of the wide area network

The wide area network is modeled with NS-2 to exploit its full support of standard protocols and traffic models. Fig. 3 graphically represents the NS-2 model. A classical bottleneck topology has been created to represent a backbone with access links. Node 0 represents the gateway. The 10 kb/s links between node 0 and node 1 and between node 2 and node 3 represent mobile connections. The link between node 1 and node 2 models a geographical backbone with 100 kb/s of available capacity and 50 ms delay. Node 3 represents another mobile user talking with the gateway. Node 4 represents a host connected to Internet through a 10 Mb/s access link. Two agents reproduce an UDP connection carrying sampled data from node 0 (the gateway) to node 4 (the Internet host). The `sc_ns_agent` on node 0 connects NS-2 with the network interface of the gateway modeled in SystemC; the technique described in Section 3.2 is used. Other two agents model a UDP flow carrying voice from the gateway to node 3 (the listener); an 8 kb/s constant bit-rate (CBR) traffic model is applied to the agent on node 0 to reproduce a mobile voice connection. Statistics on delay and packet drops are generated by NS-2 and used to evaluate the quality of service. The simulation of the lowest layers of the mobile infrastructure is possible through third-party modules included in the NS-2 distribution.

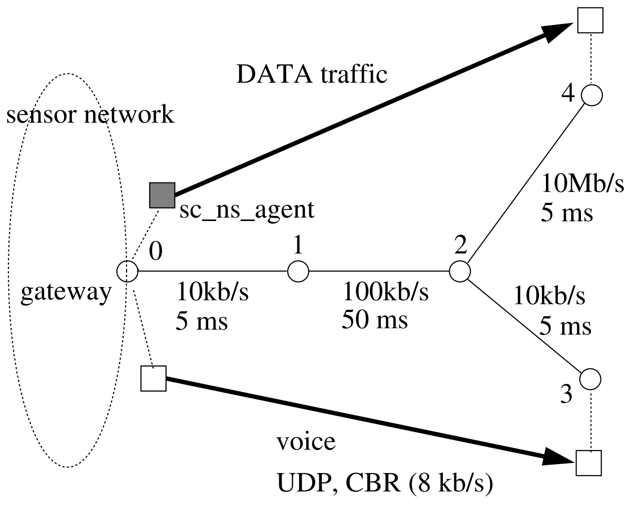

Figure 3. NS-2 model of the WAN.

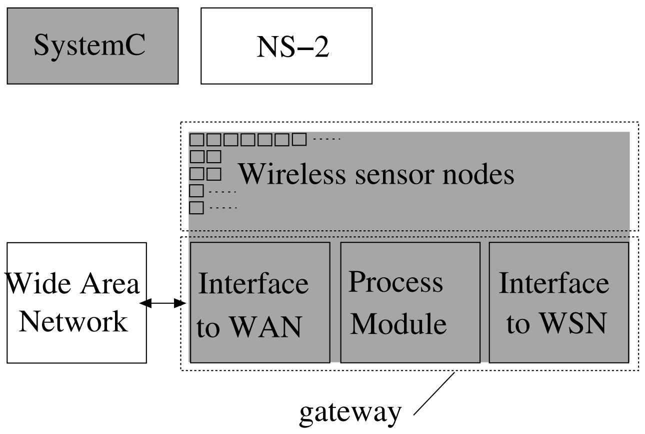

Figure 4. Co-simulation framework with a system-level description of the gateway.

4.3 Model of the gateway

The gateway has been initially modeled at system-level in SystemC. Three modules have been created: the interface to the WSN, the interface to the wide area network, and the processing module. The interface to the WSN is a sub-class of the `Node` module of the WSN and interacts with other nodes through the SystemC simulator. The interface to the wide area network contains special ports to interact with the NS-2 simulation kernel as described in Section 3.2. The processing module is connected to both interfaces through signals and queues; it implements the creation of the table containing the average of the sampled values.

Then, HW/SW partitioning is applied to the model. Since in the actual system the creation of the table will be performed by an application software running over a real-

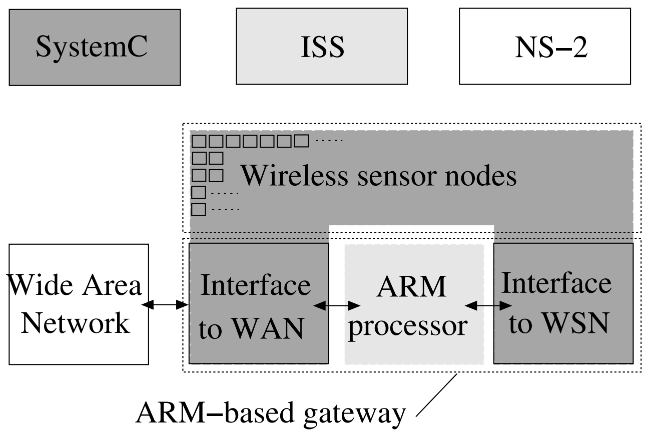

Figure 5. Assignment of the different tools after the HW/SW partitioning.

time operating system, we decided to replace the SystemC processing module by the instruction set simulator (ISS) of the ARM CPU. ISS interacts with SystemC as described in Section 3.1. Fig. 5 shows the assignment of the different tools in the final co-simulation framework. The application software is a C program running over the eCos operating system [8]; packet transmission and reception is performed by reading from and writing to memory-mapped registers of both network interfaces. A debugger can be used to find and fix bugs. It is worth to note that the same code will run on the actual platform thus shortening its time-to-market.

4.4 Experimental results

To assess the performance of the proposed co-simulation methodology we simulated the application described in Section 4. In the experimental setup, nodes are placed in a line and equally spaced. The node at one end of the line receives environment data every 1 s and broadcasts a packet containing the sampled value over the radio interface. Antennas are omni-directional and the distance between nodes is such that each node can reliably communicate with the adjacent nodes only. This scenario lead the simulator to handle the problem of hidden terminals since it can happen that nodes that are not able to hear each other when they are sending, disturb each others transmissions in a receiving node. When a node receives a packet, it re-broadcasts the message contributing to deliver it to the other end of the network. This routing protocol, a kind of *unselective flooding*, is not the best choice for a real sensor network since maximizes the number of transmitted packets, but has the advantage to test the performance of the simulator under heavy load conditions. The simulation length is 40 s and total execution time is reported as a function of the number

Nodes	SystemC+NS
10	30.9 s
100	504.3 s
250	534.4 s
1000	1717.9 s

Table 2. CPU time of the system-level simulation as a function of the number of nodes in the WSN.

of nodes to test the scalability of the technique. Tests have been performed on a Linux workstation.

Table 2 reports the total execution time of the system-level simulation in which the gateway and the WSN are modeled using SystemC and the wide area network is modeled with NS-2. The total wall-clock time has been considered since the simulator consists of different concurrent processes and it is difficult to obtain the actual *aggregated* CPU time for them. Results show that the total elapsed time strongly depends on the size of the WSN. With ten nodes the simulation is faster than real-time.

5 Conclusions

In this paper we have presented a methodology for HW/SW/network co-design and co-simulation of an heterogeneous set of networked embedded systems. One of these systems, a mobile phone, acts as the gateway between a wireless sensor network and the traditional communication network; the gateway exchanges data with the WSN and data/voice with remote hosts through the WAN. The WSN has been completely modeled in SystemC reproducing network interactions; the wide area network has been modeled with a well-known network simulator, NS-2, thus exploiting its full support of standard Internet protocols for data and traffic models for voice. The gateway has been initially modeled at system level with SystemC and then HW/SW partitioning has been applied on it; an instruction set simulator of the ARM processor has been used for the cycle-accurate execution of the operating system and the application software. Simulation of the system-level scenario shows that the total elapsed time strongly depends on the size of the WSN. With ten nodes the simulation is faster than real-time.

References

[1] M. Conti and D. Moretti. System level analysis of the bluetooth standard. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, volume 3, pages 118–123, March 2005.

[2] D. Bertozzi et al. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):113–129, Feb. 2005.

[3] F. Fummi, P. Gallo, S. Martini, G. Perbellini, M. Poncino, and F. Ricciato. A timing-accurate modeling and simulation environment for networked embedded systems. In *Proc. ACM Design and Automation Conf. (DAC)*, pages 42–47, Jun. 2003.

[4] F. Fummi, S. Martini, G. Perbellini, M. Poncino, F. Ricciato, and M. Turolla. Heterogeneous co-simulation of networked embedded systems. In *Proc. IEEE Design Automation and Test in Europe Conference (DATE)*, Feb. 2004.

[5] G. Braun et al. A universal technique for fast and flexible instruction-set architecture simulation. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 23(12):1625–1639, Dec. 2004.

[6] Grotker, Liao, Martin, and Swan. *SystemC*. Kluwer, 2002.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of ACM MOBICOM*, Boston, MA, Aug. 2000.

[8] J. Dallaway et al. Embedded Configurable Operating System – eCos. *URL: http://ecos.sourceware.org*.

[9] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. of the First ACM Conference on Embedded Networked Sensor Systems*, 2003.

[10] S. McCanne and S. Floyd. NS Network Simulator – version 2. *URL: http://www.isi.edu/nsnam/ns*.

[11] S. Park, A. Savvides, and M. Srivastava. Sensorsim: a simulation framework for sensor networks. In *Proc. of 3rd ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 104–111, 2000.

[12] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage. In *Proc. of the 1st ACM Int. Workshop on Wireless Sensor Networks and Applications*, 2002.

[13] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proc. of the 1st ACM Int. Workshop on Wireless Sensor Networks and Applications*, 2002.