Chapter 34
A SystemC/MATLAB Co-simulation Tool for Networked Control Systems

Davide Quaglia, Riccardo Muradore and Paolo Fiorini

34.1 Motivation and Problem Statement

Networked control systems (NCS) are feedback control systems in which the control loop is closed through a shared digital communication network rather than by an ideal point-to-point connection. In NCS, the communication channel can significantly affect the quality of the control due to communication delay and packet loss. Many solutions have been proposed to address this issue [3]. The simulation of NCS plays a crucial role in the verification, validation, and fine-tuning of these solutions. A simulator should capture and represent both the control and communication aspects. For instance, the control aspects include signal generation and analysis as well as plant/controller specification, whereas the communication aspects include channel and protocol specification as well as packet flow generation and routing.

This essay addresses the problem of *building an accurate simulator for NCSs*. For example, MATLAB/Simulink is one of the most used tools to design and simulate dynamic systems. Concerning the network, this tool provides low-level propagation models, which slow down simulation, and abstract queuing models which do not describe the network architecture in terms of nodes, links, and competing packet flows. There are well-known tools for network simulation [4, 7], but they do not address the simulation of dynamic systems in a native way. A possible solution to handle the heterogeneity of NCSs is the *co-simulation approach* to make different tools working together to simulate different parts of the overall system. The tools

D. Quaglia(✉) · R. Muradore · P. Fiorini
Department of Computer Science, University of Verona, Strada Le Grazie 15, Ca' Vignal 2, 37134 Verona, Italy
e-mail: davide.quaglia@univr.it

R. Muradore
e-mail: riccardo.muradore@univr.it

P. Fiorini
e-mail: paolo.fiorini@univr.it

J.H. van Schuppen and T. Villa (eds.), *Coordination Control of Distributed Systems*,
Lecture Notes in Control and Information Sciences 456,
DOI 10.1007/978-3-319-10407-2_34

have to run *simultaneously* (to reduce the simulation time on a multiprocessor system) and in a *synchronized way* (to provide correct results). Therefore, different problems must be solved to achieve this objective.

The first problem to be solved is the *interconnection of the simulation tools*. Assuming that each tool is executed by a specific process in the host operating system, simulation data should be exchanged by using inter-process communications, e.g., shared memory or network sockets. The transfer of simulation data between tools should be efficient and independent of the complexity of the simulated model.

Another problem consists in the introduction of *new modeling entities* in each tool to represent the connection of the standard entities provided by the tool with the other components modeled by the other tool. For instance, in MATLAB/Simulink workspace, new blocks are needed to represent the fact that the controller and the plant are connected together through a component modeled outside MATLAB to simulate the network.

The third issue is the creation of the *same time domain* for the global simulation. This implies that tools should perform simulation in a synchronized way and that cause–effect relationship between events belonging to different tools should be preserved.

To show how these issues can be solved, we refer to an actual co-simulation platform in which MATLAB/Simulink is connected to the SystemC Network Simulation Library (SCNSL) [1, 2]. However, the explained concepts are quite general and they can be adapted to other tools.

34.2 Framework Key Concepts

Regarding the *interconnection of the simulation tools*, network sockets are used to handle data transfer and synchronization between MATLAB and SystemC. As depicted in Fig. 34.1, socket management is separated from system/network modeling, thus making it independent of the complexity of the NCS model. In SystemC simulator, socket communications have been implemented in the simulation kernel, while in MATLAB, they are addressed by a special Simulink block named *MATLAB Wrapper*, developed as Level-2 m-file s-function. The use of sockets, instead of shared memory mechanisms, allows to distribute simulation not only among different CPUs but also among different hosts to enable load balancing or remote on-demand simulation services.

Concerning the introduction of *new modeling entities*, the MATLAB Wrapper plays this role in MATLAB/Simulink, while special objects named *registers* have been introduced in SystemC (Fig. 34.1). The MATLAB Wrapper can be connected to a user-defined number of scalar and vector input and output ports. Each port has a unique identification number and a given update frequency. MATLAB executes the Wrapper at the highest of these frequency values and, for each input port, creates a co-simulation message bearing data, the port identification number, and the simulation timestamp. Messages coming from SystemC with a given identification number

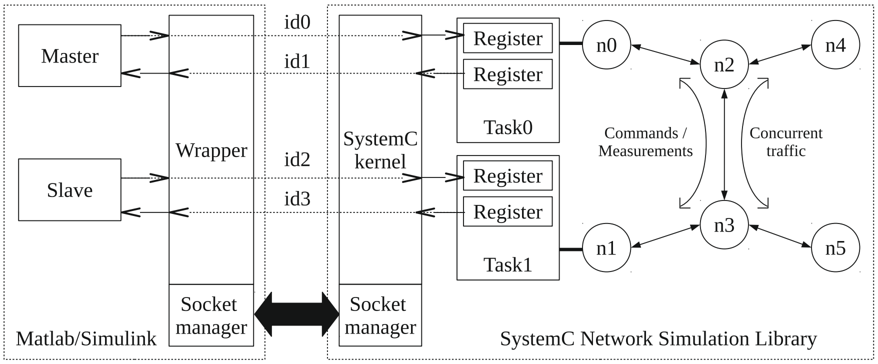

Fig. 34.1 Relationship between the new entities in MATLAB and SystemC and example of tele-operation system sharing the network with concurrent traffic

are decoded, and data are written on the corresponding output port. The SystemC simulator, born to model HW components, represents input/output ports as registers, which have been extended to let the model send/receive data to/from MATLAB. Each instance of these ports has a unique identification number which is used to associate it to the corresponding port in the MATLAB Wrapper.

Concerning the creation of the *same time domain*, a synchronization protocol has been created by using the blocking read primitive of the socket library [5]. MATLAB and SystemC exchange the time information about the next co-simulation event and then perform local simulation until this time is reached. Then, one of the peers waits for a co-simulation message from the other peer and the protocol is repeated. MATLAB simulation advances time according to a given sampling frequency, which allows to determine the time of the next co-simulation event. SystemC kernel is an event-driven simulator which manages a list of time-sorted simulation events; the next co-simulation event can be either the next event in the queue or a periodic synchronization point whose frequency is set by the designer to trade-off between time accuracy and simulation speed.

34.3 SystemC Network Simulation Library

SystemC Network Simulation Library (SCNSL) is an extension of SystemC to allow modeling packet-based networks such as wireless networks, ethernet, fieldbus, etc. As done by basic SystemC for signals on the bus, SCNSL provides primitives to model packet transmission, reception, propagation, and contention on the channel and wireless path loss. The use of SCNSL allows the easy and complete modeling of distributed applications of networked embedded systems such as wireless sensor networks, routers, and distributed plant controllers. The network scenario is described in an object-oriented way by instantiating tasks, nodes, and channels. *Tasks* are used to model node functionality in terms of packet transmission and reception; in the context of NCS, tasks are used to interface with the MATLAB Wrapper through the already mentioned registers; when data arrive from MATLAB, the corresponding task transmits them on the network and vice versa. Tasks are hosted by *Nodes*, which represent physical devices. Thus, tasks deployed on different nodes shall communicate through the channel. In case of wireless channel, some transmission properties are bound to the nodes, i.e., position in a 3D space, transmission power, and bitrate. Such properties are used by the simulator to reproduce mobility, propagation delay, loss of signal strength as a function of distance, and collisions. The *channel* represents the transmission medium; SCNSL provides models for both point-to-point (full-duplex, half-duplex, and unidirectional) and shared channels as described below. In this work, instances of point-to-point full-duplex channel are used to model a wired network, while the shared channel is used to model a wireless network.

Point-to-Point Channel Models

Point-to-point channels are used to connect node pairs. A point-to-point channel is characterized by its *capacity* and *delay*; the former represents the amount of bits that can be delivered in the time unit, while the latter represents the propagation delay; both are assumed constant during the simulation. Full-duplex channels allow transmission in both directions at the same time while half-duplex in different time intervals.

Shared Channel Models

Shared channels are used to connect more than two nodes. For each transmitting node, signal power and distance are considered with respect to all the other nodes to evaluate whether the transmitted packet can be reached and whether it generates collisions with other packets. A shared channel is characterized by its *attenuation exponent* which is applied to the distance to compute the power attenuation. Simple shared channels are also characterized by a constant *propagation delay*, while delayed shared channels are characterized by a *propagation speed* which allows to compute the propagation delay as a function of the distance between the transmitter and the receiver.

34.4 Applications

In this section, two NCS applications are presented and modeled by using the co-simulation tool to show its potentiality. For each application, the network model is detailed and some experimental results are presented.

Bilateral Teleoperation System

A particular example of NCS is the bilateral teleoperation system [5] shown in Fig. 34.1, which consists of the *master device* through which the operator controls the *remote slave robot* and a *packet-based network* which delivers all the signals (e.g., commands and measurements). `Task0` and `Task1` are the counterparts of master and slave devices in the network simulator. They are hosted by nodes `n0` and `n1`.

Nodes are connected by point-to-point full-duplex channels to create the so-called bottleneck topology; peripheral nodes are connected through dedicated links to a common backbone link. Nodes have queues to store packets exiting toward a busy link; since the backbone capacity is shared among the different traffic flows, queue level may vary during simulation and congestion may happen. Over this topology, two end-to-end application flows have been defined between applications endpoints; in Fig. 34.1, they are represented by curved arrows. The packet flow between master and slave sides in the teleoperation application (in general, between controller and plant in a NCS) has a constant bit rate since samples of commands and measurements are taken at constant rate and put in packets. A concurrent flow has been modeled between nodes `n4` and `n5`. It features an ON/OFF behavior with constant bit rate during ON periods. Teleoperation has been simulated in both uncongested and congested network conditions.

Figure 34.2a has been generated by MATLAB, and it shows the tracking error for one of the joints of the slave robot. The dashed black line refers to the uncongested scenario, whereas the red continuous line refers to the congested scenario in which control performance are affected by packet delays and losses. Figure 34.2b, c have been generated by the network simulator, and they show the packet loss rate and the communication delay, respectively, of the path from the master to the slave. In all the figures, the vertical lines separate the ON and OFF intervals of the concurrent traffic. During OFF periods, the delay is minimum and equal to the propagation delay. When the concurrent source is switched on, queues at the edges of the bottleneck link start to grow and the delay increases. When the queues are full, arriving packets are dropped. When the concurrent traffic is switched off, the number of enqueued packets decreases as well as the delay. These results show that the co-simulation tool works as expected. More complex network scenarios and concurrent traffic models (e.g., probabilistic, actual recorded traffic, etc.) can be easily implemented to model all possible kinds of working conditions.

Formation Control

Formation control is a traditional control problem in which autonomous vehicles should adapt their trajectory and speed to keep relative position with respect to each other [6]. In our scenario (Fig. 34.3a), each vehicle (except the formation leader) has only one leader and zero or more followers. We assume that each vehicle knows its position and speed and periodically broadcasts this information by using wireless messages so that followers can know it. Therefore, each vehicle receives position and speed of neighbors but considers only the information coming from its leader and changes its trajectory and speed accordingly. As depicted in Fig. 34.3b, each vehicle can be considered a NCS where the dynamic model of the vehicle represents the plant which receives directly the command u_i from the controller; the output y_i (position and speed of the vehicle) is sent back to the controller and to the neighbors by using wireless messages; the output of a given vehicle is the reference signal r_i for all its followers; each vehicle (i.e., the corresponding plant) is affected by a perturbation signal (e.g., due to wind, water flow, obstacles) which alters its position and speed.

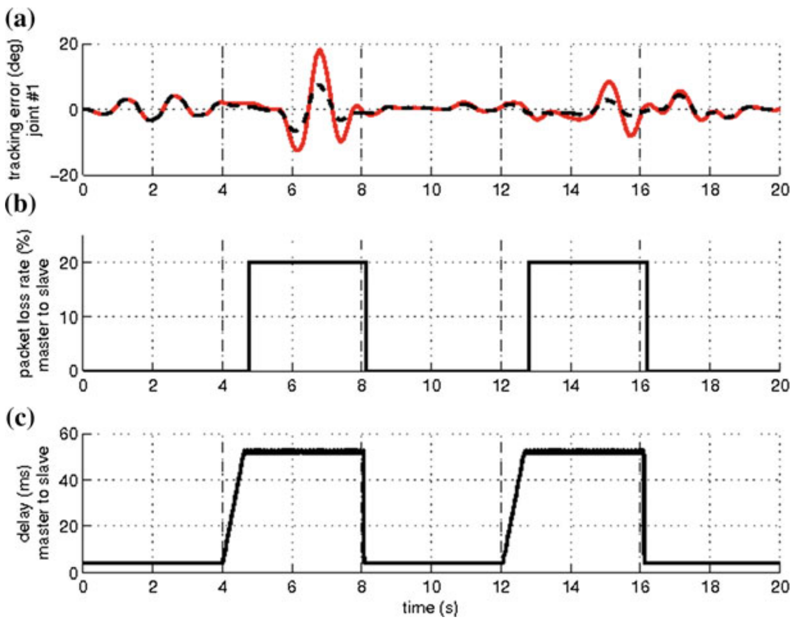

Fig. 34.2 Tracking error, packet loss rate, and delay from the teleoperation simulation

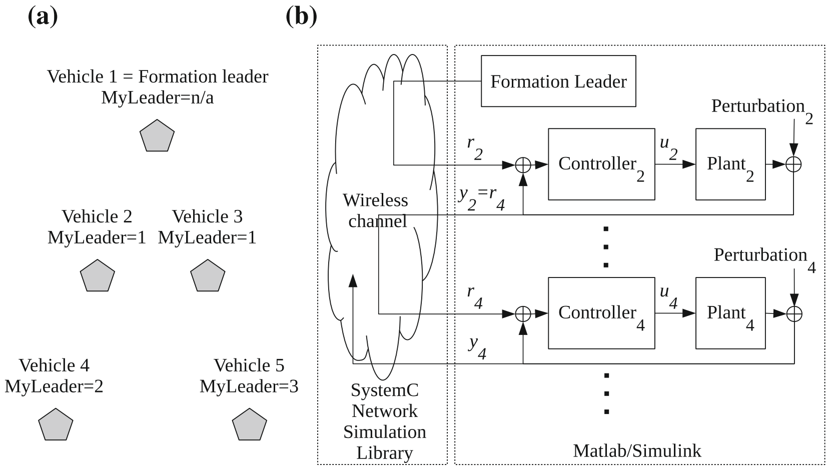

Fig. 34.3 Formation control scenario (**a**) and architecture of each NCS with tool mapping (**b**)

One of the most critical issues of this scenario is related to wireless transmission on the shared channel. Messages may not arrive to the followers because of packet collisions (i.e., overlapping of more transmitted signals at the receiver side) and out-of-range transmission. If the position of the leader is not heard, then the follower cannot react promptly to trajectory changes and perturbations so that colli-

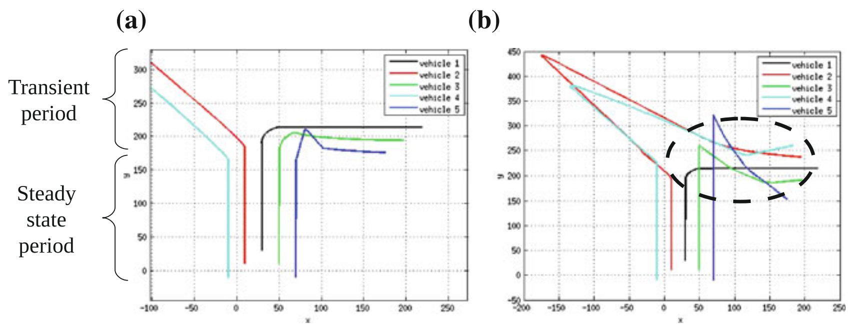

Fig. 34.4 Vehicle trajectories in two different communication scenarios: minimum (**a**) and maximum (**b**) transmission power

sions between vehicles and loss of vehicles may occur. Reaction delay depends on the timely reception of reference messages which depends on channel arbitration (to avoid or compensate collisions) and propagation delay as a function of distance. Analytical approaches to study channel behavior are not scalable with the number of vehicles, and therefore, simulation is crucial to identify problems and to validate solutions before the actual deployment. As depicted in Fig. 34.3b, we used the proposed co-simulation tool in which MATLAB simulates the different vehicles (i.e., controllers, plants, and perturbations), while SystemC reproduces the behavior of the wireless channel in between and the communication protocol.

Figure 34.4 shows the simulated behavior of the vehicles when the leader changes trajectory in the presence of perturbations as a function of two different transmission power settings. In the scenario shown in Figure 34.4.a, the transmission power of each vehicle is set to the minimum required to reach the followers when the distance requirements are satisfied. In the transient period, one vehicle (and its follower) gets lost since it cannot hear the reference signal of its leader. In the scenario shown in Figure 34.4.b, the transmission power of each vehicle is set to the maximum so that messages can be heard over a great distance. In the transient period, the perturbed vehicle increases the distance from the leader but, after a delay, rejoins the group since messages continue to be heard even if the distance is great. The drawback of this approach is the higher number of ripples in the trajectories (see dashed region) due to problems in the reception of reference messages caused by an increased number of collisions; in fact, nodes interfere with each other over a larger area due to the higher transmission power. The results in Table 34.1 confirm this conjecture; the higher variability of position error is related to the higher number of packet losses depending on message collisions. In the table, the position error is also compared to the one obtained with pure MATLAB simulation in which inter-node communications are modeled as constant delay blocks. It is worth noting that the purpose of this table is not to assess the performance of a particular control strategy, protocol, or simulation tool but to show that no verification is possible without the accurate modeling of the network provided by a suitable co-simulation tool.

Table 34.1 Relationship between control performance and network behavior as a function of the simulation strategy

Simulation strategy	Position error (m)		Packet loss rate
	Mean	Std. deviation	
Co-sim. with minimum TX power	2.0	5.5	29 %
Co-sim. with maximum TX power	2.3	9.4	72 %
MATLAB only	0.4	1.0	N/A

34.5 Further Research

In general, co-simulation has some computational overhead due to synchronization. To avoid this, a new trend in this context consists in representing both the discrete- and the continuous-time components of the system by a single hybrid model. The application of this approach to NCS should be still investigated in detail.

34.6 Further Reading

This co-simulation tool can be fruitfully used to fine-tune joint control/network design techniques as those proposed in Chap. 33.

References

1. Accellera. IEEE Std 1666–2005 IEEE Standard SystemC Language Reference Manual. IEEE Std 1666–2005, pages 1–423, 2006.
2. SystemC Network Simulation Library - version 2, 2012. URL: http://sourceforge.net/projects/scnsl
3. Hespanha JP, Naghshtabrizi P, Xu Y (2007) A survey of recent results in networked control systems. Proceedings of the IEEE 95(1):138–162
4. S. McCanne and S. Floyd. NS Network Simulator - version 2. URL: http://www.isi.edu/nsnam/ns
5. Quaglia D, Muradore R, Bragantini R, Fiorini P (2012) A SystemC/Matlab co-simulation tool for networked control systems. Simulation Modelling Practice and Theory 23:71–86
6. Wang PKC (1991) Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation. J. Robot. Syst. 8(2):177–195
7. Zhu C, Yang OWW, Aweya J, Ouellette M, Montuno DY (Jun. 2002) A comparison of active queue management algorithms using the OPNET Modeler. IEEE Communications Magazine 40(6):158–167