12th 2008 IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications

HLA-OMNET++: an HLA compliant network simulator

Emanuele Galli, Gaetano Cavarretta and Salvatore Tucci
Università of Roma "Tor Vergata"
Via del Politecnico 1, 00133, Rome, Italy
galli@disp.uniroma2.it

Abstract

An approach to the modeling of Critical Infrastructure can be the integration of already implemented and heterogeneous simulators. In this way the model builder can concentrate more on the modeling of interdependencies between heterogeneus infrastructure than on the developing of an huge and unique simulator. The HLA standard is probably one of the more common technique to reach such goal. We present the architecture and implementation of the HLA-OMNet++: an HLA-1516 network simulator used to simulate the communication network infrastructure.

1 Introduction

In the last years different projects sponsored by the government agencies (e.g. NISAC in USA and EISAC in EU) address the problem of modeling and simulation of complex critical systems. From an homeland security point of view, complex critical systems (e.g. communication networks, power grid, transportation systems, financial services, etc...) can not be considered as standalone systems, but, due to their interdependencies, have to be considered as a complex system composed of interdependent systems. One of the main approaches used to simulate complex critical systems is the federation of existing simulation models [2, 1, 3, 4]. Unfortunately, sector-specific simulation model, developed during the past, typically were designed not having in mind distributed simulation features. Then, one of the main steps toward the federated simulation of critical complex systems is the development of HLA compliant network simulation framework.

This work, part of the CRESCO project, enhances the OMNeT++ network simulation framework (http://www.omnetpp.org), implementing an HLA scheduler and an IEEE 1516 compliant Federate Ambassador (implemented and tested using the poRTIco HLA implementation - http://www.porticoproject.org). The proposed scheduler allows to use existing OMNeT++ network simulation model, to design new network model and use them in a distributed simulation. Right now the HLA-OMNeT++ publishes interactions to send and receive messages between network nodes as well as to modify the network topologies and network nodes properties. However, the proposed HLA compliant scheduler supports the publishing of any type of interaction and object.

The paper is organized as it follows. Section 2 briefly describes how OMNeT++ works. Section 3 describes the architecture of the OMNeT++ federate (IEEE 1516 compliant) and of the HLA compliant scheduler. Section 4 concludes the paper.

2 OMNeT++ overview

OMNeT++ is an event driven simulation framework to model and to simulate communication networks and, more in general, computer network systems. The OMNeT++ architecture is based on components and it is completely modular.

An OMNeT++ model is obtained composing one or more, hierarchically nested, modules. The lower level modules of the hierarchy, named *Simple Modules*, are programmed in C++ using the simulation library and extending the `cSimpleModule` class. Simple Modules generate and react to events and they implement the behavior of a modeled object (e.g. a network node or a link). Each module is characterized by a set of parameters that can be used to parametrize the module behavior. Simple module can be combined to compose more complex modules, named *compound modules*. Simple and/or compound modules communicate by exchanging messages. Messages can be used to model events, network messages, packets, bits, signals or other. A message in OMNeT++ is an instance of the class `cMessage` and it can be used by any module. Generally it's necessary to define a module which is the traffic generator of the simulation that defines the distribution of traffic, the dimension of packages and so on. All the OMNET++ events are queued in the Future Events Set (FES) queue. The scheduler extracts messages from the FES in

1550-6525/08 $25.00 © 2008 IEEE
DOI 10.1109/DS-RT.2008.44

arrival time and priority order.

OMNeT++ offers two methods to send messages: the `sendDelayed()` method to send messages with a specified delay; and the `send()` method to send messages with zero delay.

3 The HLA-OMNeT++

To make OMNeT++ IEEE 1516 compliant we need: 1) to design a new HLA compliant scheduler for OMNeT++, named `rtiScheduler` and 2) to design a Federate Ambassador that allows the interaction between poRTIco, written in Java, and the `rtiScheduler` other than the OMNeT++ modules, written in C++.

3.1 The OMNeT++ Federate architecture

The HLA-OMNeT++ is composed by three blocks (see figure 1). The first one (bottom of the figure) is the interface between the OMNeT++ scheduler and the RTI. It is a Java module that implements the OMNeT++ federate, that communicates directly with the RTI-Ambassador, and the OMNeT++ Federate Ambassador (OFA) that interacts directly with the RTI. The HLA compliant scheduler is implemented by a C++ module, the `rtiScheduler` (middle of the figure). The rtiScheduler guarantees the synchronization with other federates. The OMNeT++ Federate and the rtiScheduler communicate using the client/server paradigm implemented by a socket based interprocess communication and an XML based communication protocol. The third block is composed by the OMNeT++ modules that implements the communication network simulation model. The network model designer will work only at this level without care about how events are scheduled and she/he has to define the *Entry Point Module* (the traffic generator module) that will "communicate" with other federates and initializes the simulation of every packet. Moreover the designer has to define the network topology and the simulation model.

```
message NetPkt
{
    fields:
        string srcAddress;
        string destAddress;
        string payload;
        string destStatus;
        string id;
        string protocol;
        unsigned int pointerTopology;
};
```

Listing 1. The NetPkt, defined using the NED language

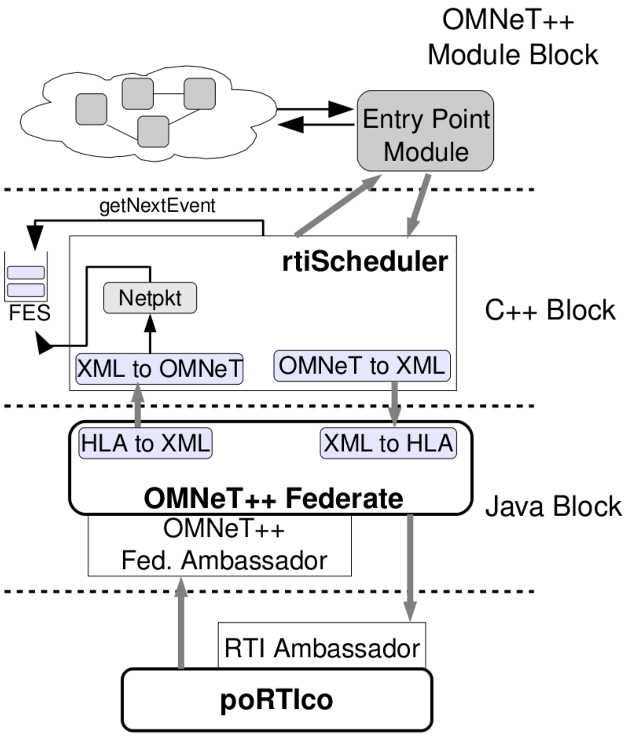

Figure 1. OMNeT++ Federate Architecture

When the OFA receives a message from RTI, the message is translated in Xml language and sent to the rtiScheduler by the open socket. As soon as the xml-message is received by the rtiScheduler, it's used to create a new OMNeT++'s message, named Netpkt, as shown in listing 1.

The Netpkt-message contains the source and destination address node of the communication network in addition to the payload, the transportation protocol, the status of the destination node (if active or inactive due to some fault) and other information. The new message is put in the FES queue to be managed by the rtiScheduler. As soon as the message is gotten by the scheduler it is delivered to a specified OMNeT++'s module, responsible to manage all messages received by other federates. When the message arrives to the destination node, it is delivered to the entry point module which calls the `sendToRTI` method of the scheduler. This method is responsible to translate the OMNeT++ message in a well formatted xml-message and sends it to the Java block. Subsequently the message is translated in HLA language to be received by the RTI Ambassador.

3.2 The rtiScheduler

OMNeT++ offers the abstract class `cScheduler` to customize the scheduler. The `cScheduler` class defines four virtual method that are: `startRun()`, to start a simulation; `endRun()`, to terminate the simulation;

`executionResumed()`, to restart the simulation from a pause; `getNextEvent()`, to extract the next event to process from the the FES (NULL is returned if the FES is empty).

Moreover we have defined other two methods that make the `rtiScheduler` independent from the modeled network topology and the Local RTI Component. Such virtual methods are: `setInterfModule(cModule *entryPointModule`, used to set the OMNeT++ module that will receive the arriving request from the federation; `sendResponseToRTI(cMessage *msg)`, used to reply requests from the federation. In the following we describe the main virtual methods implemented.

`startRun().` This method is used to initialize the simulation and to activate the `msgServer` (see Fig. 1), a thread responsible to manage the XML messages received/sent from/to the Federate Ambassador.

`setInterfModule(cModule *entryPointModule).` This method defines which is the OMNET++ module that receives messages from the federation. All messages from federation are always received from the `entryPointModule` module (see Fig. 1).

`sendResponseToRTI(cMessage *msg).` This method can be used by any OMNET++ module that wants to send a message to the federation. Usually the `entryPointModule` is delegated to manage and send the reply to the federation. The OHLA module receives the `msg` message (a NetPkt message) from the entryPointModule. Then it parses the `NetPkt` and it creates the `commRespToRTI` message that is sent to the Federate Ambassador.

`*getNextEvent().` This method implement the selection of the next event from the FES (see Figure 2). If there is an event in the queue, and it is time to deliver the message, the message is returned, otherwise the method return NULL. In a distributed simulation, an empty FES does not necessarily means that the simulation is terminated. Then we have modified the scheduler to generate, in case of an empty FES, a `Time Advance Request` to the RTI and to wait the arrival of a `Time Advance Grant` signal from the RTI. The `Time Advance Grant` signal wakes up the scheduler that check again the FES.

4 Concluding remarks

This paper presents the implementation of an HLA compliant version of the OMNeT++ network simulation framework. The proposed extension is under submission, for approval, to the OMNeT++ community. The HLA-OMNeT++ has been used with success in the implementation of a simulation framework of complex system developed in the context of the CRESCO project and which results are published with success in [1]. HLA-OMNeT++ has been validated comparing the results obtained running monolithic and distributed simulations. HLA-OMNeT++ can be obtained sending an email to the authors.

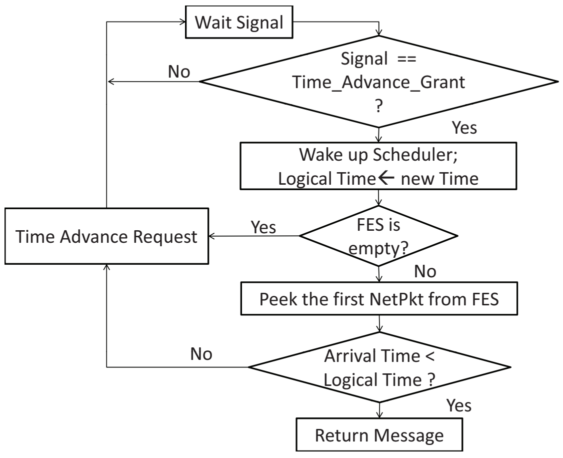
Wait Signal
Signal == Time_Advance_Grant ?
No
Yes
Wake up Scheduler; Logical Time ← new Time
Time Advance Request
Yes
FES is empty?
No
Peek the first NetPkt from FES
No
Arrival Time < Logical Time ?
Yes
Return Message

Figure 2. Flow chart of getNextEvent method

References

[1] E.Casalicchio, E.Galli. ”Metrics and Statistical Measures to Quantify Critical Infrastructure Interdependencies”. Second IFIP WG 11.10 International Conference on Critical Infrastructure Protection, George Mason University, Arlington, VA, USA, March, 2008

[2] E.Casalicchio, E.Galli, S.Tucci. ”Federated Agent-based Modeling and Simulation Approach to Study Interdependencies in IT Critical Infrastructures”. Proc. of 11th IEEE Int’l Symposium on Distributed Simulation and Real Time applications (DS-RT07), Crete, Greece, Oct 2007.

[3] D. Dudenhoeffer, M. Permann, and M. Manic. Cims: A framework for infrastructure interdependency modeling and analysis. *Winter Simulation Conference, 2006. WSC 06. Proceedings of the*, pages 478–485, 3-6 Dec. 2006.

[4] The IRRISS Project. http://www.irriis.org