Bulletin of the *Transilvania* University of Braşov
Series I: Engineering Sciences • Vol. 4 (53) No. 1 - 2011

ONTOLOGY BASED RESOURCE MANAGEMENT OF REAL AND EMULATED TELECOM SYSTEMS

T. BĂLAN[1] F. SANDU[2] V. CAZACU[1]

Abstract: *Because recent heterogeneous telecom networks have developed into complex distributed systems, the management of these systems is more elaborated and resource consuming. Self-organization of network resource colonies based on semantic associations with logical relations can provide a solution. This paper presents a method for resource management in telecom distributed testing environments with real and emulated elements. Generic Path is an object oriented methodology for modelling and engineering of network functions and services. We propose a method for integration of real and emulated testing environments and a network model for resource sharing using Generic Path and a resource management ontology.*

Key words: *ontology, resource management, Generic Path, emulation.*

1. Introduction

The dimensions and complexity of latest network architectures makes end-to-end testing based on real equipment almost impossible. Emulation/simulation of network elements is a common practice for development and testing. Interconnecting real and emulated elements in complex testing environments, along with soft-switching technology proves that the border between real and emulated is very thin.

From solution implementation point of view we first focus on the integration of real network interfaces at simulator/ emulator level, based on sockets, using OMNeT++ [10]. Using the same simulator and the Formux Generic Path Architecture Prototype [3-4] implementation we describe the method of creating the Generic Patch (GP) entities at simulator level.

The paper proposes a solution for real and emulated interfaces integration and management of resource colonies. Network Resource Ontology with formalized semantics allows automatic composition and sharing of GPs between different “layers” as well as between different domains via mediation points. Resource grouping, organization and selection are based on capabilities, QoS, availability and probing.

2. Network of Information

Ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationship between those concepts.

[1] Siemens Program and System Engineering Braşov.
[2] Dept. of Electrical Engineering and Computer Science, *Transilvania* University of Braşov.

It is used for structuring and organizing of data/information and as means for information and mechanisms sharing between different systems, by using shared vocabulary. Ontology extracts essential data from information, recognizes conflicts and allows completing missing knowledge with the help of available information.

One of the domains where ontology impact is more visible is Semantic Web, enabling machines to understand the semantics, or meaning, of Internet available information. But networks of information are largely used also for artificial intelligence, systems engineering, software engineering, biomedical informatics, library and indexing science.

Describing the network resource non-functional features with network resource ontology and integrating it into the resource profiles enables quality-aware network resource selection and composition. This is a method for fulfillment of application service QoS requirements and objectives. The use of network resource ontology facilitates networking interoperability [9].

3. The Generic Path Concept

The Generic Path (GP) [11-12] is an abstraction that provides object-oriented constructs in the context of the “network of information” paradigm, enabling connections to information objects rather than just too specific hosts. Generic Path architecture was proposed in the European FP7 project 4WARD - “Architecture and Design for the Future Internet“. GP’s objective is to overcome the rigidness of the OSI layer model that, together with the TCP/IP protocol limitations, is considered not cross-technological adaptable. By adapting transport procedures to the capabilities of the underlying network, the GP is intended to provide an easy-to-use and efficient operation for both user and network.

The GP model is based on contextualized communication, meaning the logic behind elements association and communication is decided using algorithms that take into consideration the relationship between elements, described as ontology.

The GP is a general concept, but can be better applied in telecom networks. With the increase use of Cloud Computing, Generic Path can be a solution for organizing of the distributed resources and computational information in the cloud.

We will describe the concepts for the GP abstraction, as mentioned in the 4WARD project [11]. “Entities” are the basic building blocks of the GP architecture. An Entity is the container for any kind of programming logic, but are also responsible for routing, access control, name resolution and management of records of the resources used by the GPs. Additionally, there is one special type of Entity, unique for each physical node called Core. All Entities and the Core together on one single physical node form a Node Compartment (the vertical grouping of entities in our diagram). The Core Entity is responsible for the management of Entities and the communication between them.

Furthermore, Entities are grouped in Compartments when they support the same Protocol (the horizontal grouping of entities in our diagram). The decision to become part of one Compartment is made by the Entity itself - see Figure 1.

The Compartment concept is very important as it “contextualize communication and service infrastructures of different scales or functional complexity; from a single link, a domestic network, a campus, an enterprise, to large federated structures” [12]. In many cases, Compartments can use, but are not limited, to a specific protocol, so somehow resemble to OSI layers. But, as example, Compartments can represent, in case of mobility, grouping

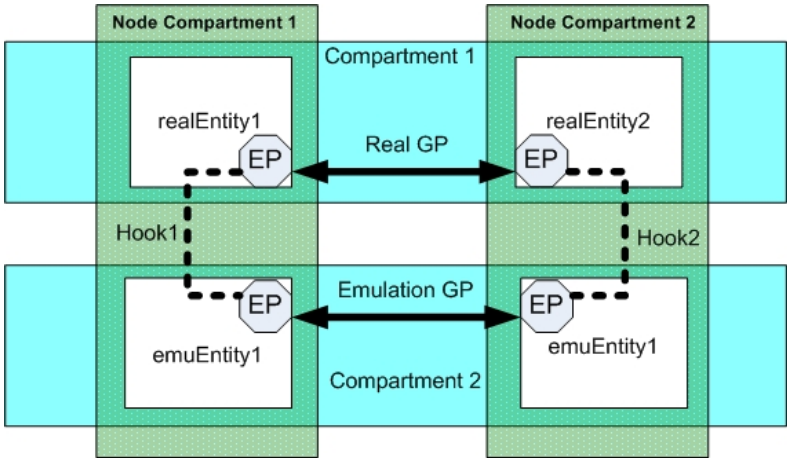

Fig. 1. *Basic building blocks of the GP architecture*

based on different wireless technologies (GSM, WCDMA, LTE) or for voice services the separation between VoIP elements and GSM elements [2]. In the case presented in this paper, Compartments are grouping resources into real and emulated resources domains in case of complex testing environments.

The relation established between Entities is called Generic Path. A Generic Path is, from the conceptual point of view, the instantiation of a protocol and optionally the state of the protocol, too. The GP is a result of ontology logic processing for establishing a direct relationship between communication involved elements.

Entities interact with each other over "Hooks" and "Ports" or Generic Paths and Endpoints. Endpoints have an interfacing role but also take care of error control, flow control, header processing and data manipulation (encryption, trans-coding).

The communication is restricted to those Entities that are part of the same Compartment. But Node Compartments enable Entities - supporting different protocols - to communicate.

When entities in the same Node Compartment but in different Compartment are connected, this link is called a Hook, while the interfaces (Endpoints) of the Hook are named Ports.

4. Real/Emulated Resource Sharing Architecture

After presenting the concept of the Generic Path the model that we elaborated for resource sharing of real and emulated testing environment will be introduced.

Resource Management is the process of allocating and de-allocating available resources to requesting entities, in order to avoid colliding access to objects. Resource management comprises traffic management, resource allocation strategies and performance management, thus ensuring that QoS demands.

When it comes to differentiate between real and emulated/simulated resources, there are two main parameters that should be weighted: processing performance and costs. In many cases, depending on the simulation machine, emulators/simulators are not capable of handling the processing amount like dedicated equipment that is replaced.

But in many cases emulators can act several network roles, replacing more types of dedicated elements (this is one advantage of emulation).

One possible scenario for traffic separation based on capability can be the usage of emulators for low bandwidth traffic, for example for signaling, while the user-data traffic is handled by the real equipment. Another possible scenario can be using in parallel more elements from the resource colony based on cost, as long as performance is not affected. Depending of the protocol/resources used, it is possible that real and emulated interfaces do not switch from one to another excluding each other, but can work in parallel, sharing resources with help from mediation points for aggregation and multiplexing.

A Mediation Point is an Entity that belongs to many Compartments and has the capability to "mediate" (switch), between the respective Endpoints. This mediation is performed according to the defined rules of the ontology, based on the network resource parameters.

Our proposed real/emulated resource sharing GP architecture has three distinct horizontal compartments, one Real, one Emulated and one Common.

The last compartment is a dedicated shared medium, where the end-to-end GPs are aggregated and multiplexed. The aggregation/multiplexing function and its inverse operation are handled via mediation points, interrupting and inter-connecting but not terminating the different GPs - Figure 2.

Through mediation points, the network flow can be routed or aggregated, including new network elements or replacing real elements in the network flow with emulated ones, or reverse.

The GPs are "interrupted" by the mediation points MPrx, MPex in the node compartment Node-Compartment x. In order to use the common GP-C, the mediation points of GP real and GP emu have to be connected within the node compartment by a Hook.

In the endpoints of GP common, EPc1 and EPc2, the processing for execution of the mux/demux function takes place.

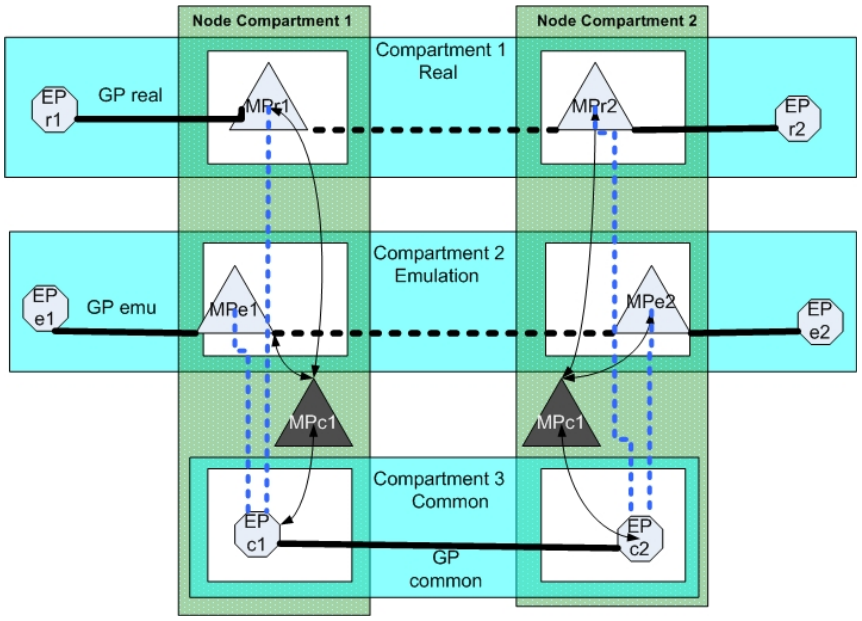

Fig. 2. *Our proposed real/emulated resource sharing GP architecture*

5. Integrating Real Interfaces in OMNeT++

Communication between simulation models and real-time applications is problematic, especially from the point of view of event synchronization, but also from a message coordination perspective. For this reason, an adaptation layer is introduced.

The PC that runs the discrete events simulator software will act as a listening server for a number of applications that also run on the real equipment that we try to emulate: TCP and UDP sockets are the most used interfaces, but also raw sockets can be used. The emulator should be prepared any time to accept messages, the same way as a real element. The simulation engine used is OMNeT++, a well-know open source discrete event simulator. The simulation engine used is OMNeT++, a well-know open source discrete event simulator. OMNeT++ is an object-oriented modular network simulator, which can be used for: traffic modelling of telecommunication networks, protocol modelling and other network-related simulations [8].

There are two time domains that need to be coordinated: the simulation time of OMNeT++ with the real time used by the equipment that generates the traffic on the real interface [6].

For that purpose, a scheduler class was implemented, based on the model of cSocketRTScheduler that OMNeT++ offers. This class has the role of receiving real messages and buffering them in a byte array variable. Also, the time of receiving packets is registered.

For reading from and writing to the buffer, an external interface class is implemented [1]. At simulator level, the external interface is visible as a separate module, which acts as client for another module that is the server - see the diagram of Figure 3.

In order to start the real listening socket, the external client has to run an initialization method that also triggers the listening state.

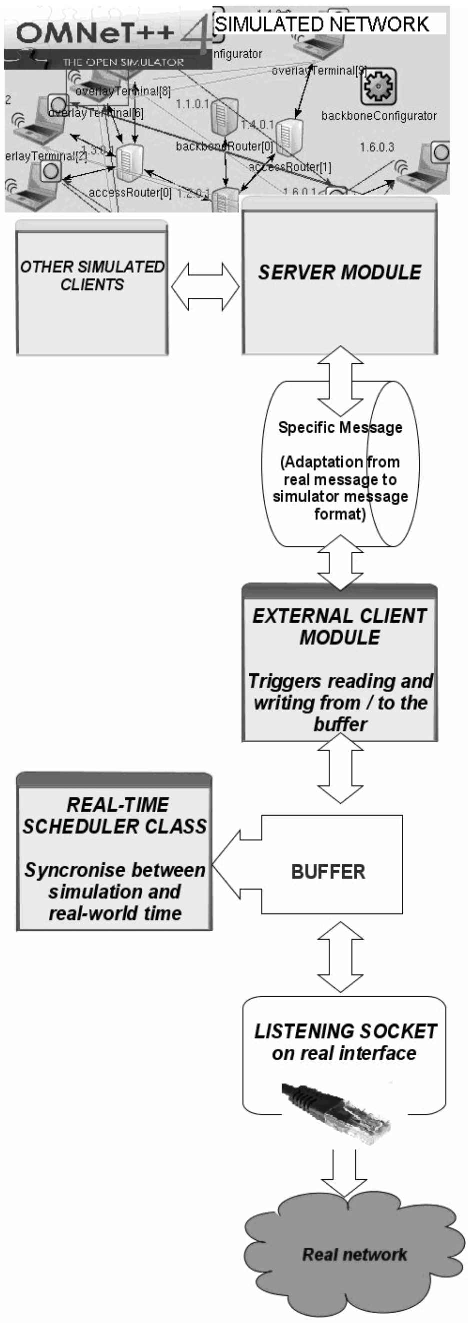
SIMULATED NETWORK
OMNeT++ 4
THE OPEN SIMULATOR
OTHER SIMULATED CLIENTS
SERVER MODULE
Specific Message
(Adaptation from real message to simulator message format)
EXTERNAL CLIENT MODULE
Triggers reading and writing from / to the buffer
REAL-TIME SCHEDULER CLASS
Syncronise between simulation and real-world time
BUFFER
LISTENING SOCKET on real interface
Real network

Fig. 3. *Architecture of the "mixed reality" test environment*

For reading incoming packages from the buffer, the external interface class is using self-destined packages. This is a common method used by OMNeT++ as trigger for events execution. The incoming packets are replayed, one by one, in a serialized way into the simulator, in close coordination with the scheduler class for time synchronization. Before each event execution there will be some small delays.

The real messages are mapped into the message structure of the simulator. The OMNeT++ INET framework has support for many types of protocols, so the message mapping can be done directly to one of the INET classes.

For sending packages, the process is reversed: the message is converted into a byte array, and passed to the buffer.

A method of the scheduler class is invoked, so outgoing packages are played on the real interface using the socket.

This adaptation layer may prove disadvantageous when the number of real messages sent is greater than the scheduler class' ability to process, like the case of flooding one interface with traffic.

The effect will be a break of the socket connection.

In order that the simulation keeps up with real time, the OMNeT++ simulation should be run in fast or express mode. For didactic purpose, normal animation time can be used in message analysis. When simulation is running with animation and the external interface is waiting for a reply, timeout that might occur represent quite a big risk, because the simulator time is delayed.

6. Real and Emulated Interfaces Integration with Generic Path

Extending the Formux Generic Path OMNeT modules [3] with real interfaces is performed using the socket based method described in the previous paragraph. The module acting as Server for the socket-based real interface connection can be included in any other OMNeT++ module. So in this case the Server functionality is embedded in the GP Entity module. At OMNeT level, the Simulated Entities modules can include also the logic of the emulated equipment node (protocol implementations and their functionality).

The OMNeT simulator is using the information provided in specific configuration files. The topology description with mentioning of the network elements is performed through the usage of.ned files or in a "visual programming" graphical way.

Another mandatory file for the simulation is the.ini file where certain parameters for the simulation can be defined. In order to initialize the "GP enabled" network nodes, the initString should be configured: every node from the Formux implementation is basically a Compartment Node and the Entities and Compartment affiliation for each Entity can be described, like in the example below:

```
**.node1.os.initString=
"EmuEntity:emuEntity1,
Emulation:emuEntityA;realEntity:
realEntity1, Real:realEntityA"
```

In the above example the Node Compartment consists of two Entities, each being part of "Emulation" and "Real", that are the names of the two Compartments. The names of the Entities for each Compartments are emuEntityA and realEntityA.

For instantiating a Generic Path the function createGP() should be called, with suitable parameters:

```
# include <ForMux/Core/ForMux.h>
# using namespace ForMux ;
public MyEntity : public Entity {
 void someMethod () {
AbstractEntity :: createGP (&
ourCompartment, & myName,&
destinationName, boost :: bind (&
MyEntity :: _createGPCallback,
this, _1));
}
```

One important element in the OMNeT implementation is the way Capabilities are defined, because they define the network resource ontology components: Network Resource Parameter, Metric, Unit, Type, Relationship, Impact, Aggregated (ontology parameters are similar to the QoS ontology described in [5], [7]).

Capabilities are intended to provide a way of expressing the needs for services when establishing a communication path between more communication partners via GP. These needs will be expressed via sets of Capabilities, whereas each Capability is represented by a name and a set of Properties. A Property essentially is a key-value pair with the addition of a type. The Capability identifier is a simple string, called CapabilityList, representing a point within the hierarchy to insert this capability:

```
Capability :: pointer cap =
Capability :: create (
<unique capability identifier >,
<property name 1>, Property ::
value_t :: < TYPE >, param
a1,..., param an,
```

The OMNeT nodes are Node Compartments. Each Node consists of several Entities that represent the same type of resource. The resource election is related to the ontology defined by Capability parameters.

Using the real and emulated interfaces described in chapter 5 we were able to integrate an emulated element in a real logical network chain. A Generic Path was established between one node that represented the simulated element and the node that was containing the listening socket for the real interface.

So far only GP establishment test and Hook creation tests were performed but the goal is to test also relocation scenarios based on capabilities (accessed via ontology logic).

7. Conclusions

Semantic organization of network resources represents a solution for the autonomous management of distributed telecom networks. By the means of Generic Path concept, introduced by the 4WARD FP7 project, the object-oriented network is organized as a network of information. GP offers possibility to preserve service modularity, enable abstraction, reuse algorithms, and compose networks and network services in a structured way. By means of cloud computing and virtualization, resources location and type (real or emulated) are more diverse. The Generic Path ontology based framework can be used for the abstraction and management of federation of resources. Using a socket-based method for connecting real interfaces to simulation environment and grouping the network elements in logical compartments using the OMNeT Formux implementation.

References

1. Bălan, T., Sandu, F., Cserey, S., Cazacu, V.: *LTE eNodeB Demonstrator with Real and Simulated Interfac*e. In: 10th Int. Conference on Development and Application Systems, Suceava, Romania, May 27-29, 2010, p. 179-184.
2. Curpen, D.M., Szekely, I.: *Elements of a Modern Telecommunication Infrastructure*. In: Bulletin of the *Transilvania* University of Braşov (2005) Vol. 12 (47), New Series, Series A1, p. 365-371.
3. Draxler, M., Droge, S., et al.: *ForMux/ GP Prototype Design & Implementation*. In: Project Group Augmented Internet

II, University of Paderborn, Germany, June, 2010. Available at: *genericpath. googlecode.com/files/gp_doc.pdf*. Accessed: 28-06-2011.

4. Draxler, M., Droge, S., et al.: *Generic Path Architecture Prototype*. Available at: http://code.google.com/p/genericpath/. Accessed: 25-04-2011.
5. Guo, G.: *A Method for Semantic Web Service Selection Based on QoS Ontology*. In: Hunan Institute of Humanities, Science and Technology, Journal of Computers **6** (2011) No. 2, p. 377-386.
6. Mayer, C., Gamer, T.: *Integrating Real World Applications Into OMNeT++*. In:Institute of Telematics, Universität Karlsruhe (TH), Technischer Bericht, Nr. TM-2008-2, Feb. 2008, p. 80-89.
7. Papaioannou, I., Tsesmetzis, D., et al.: *A QoS Ontology Language for Web-Services*. In: Proceedings of the IEEE 20th International Conference on Advanced Information Networking and Applications, 2006, p. 101-106.
8. Sandu, F., Cserey, S., et al.: *Simulation-Based UMTS e-Learning Software*. In: Proceedings of the 1st International Conference on PErvasive Technologies Related to Assistive Environments, July 2008, p. 326-331.
9. Serrano, M., Foghlú, M., Donnelly, W.: *Enabling Information Interoperability in the Future Internet: beyond of a SOA Design Requirement*. In: FIA Session Linked Data in the Future Internet, FIA Assembly, Ghent, 15-17 December 2010, online proceedings. Available at: http://linkeddata.future-internet.eu. Accessed: 20-05-2011.
10. Varga, A.: *OMNeT++ Discrete Event Simulation System Version 3.2*. In: User Manual, 2005, Available at: www.OMNeTpp.org. Accessed: 28-06-2011.
11. *** 4WARD: *Architecture and Design for the Future Internet*. In: Deliverables, D-5.2 Mechanisms for Generic Paths FP7-ICT-2007-1-216041-4WARD/D-5.2. Available at: http://www.4ward-project.eu. Accessed: 20-06-2011.
12. *** 4WARD: *Architecture and Design for the Future Internet*. In: Deliverables, D-5.3 Evaluation of Generic Path Architecture and Mechanisms, FP7-ICT-2007-1-216041-4WARD/D-5.3. Available at: http://www.4ward-project.eu. Accessed: 20-06-2011.