

# Data Exchange between Real Network Component and OPNET Modeler Simulation Environment

MILAN BARTL, JIRI HOSEK, TOMAS MATOCHA, KAROL MOLNAR, LUKAS RUCKA

Department of Telecommunications

Faculty of Electrical Engineering and Communication, Brno University of Technology

Purkynova 118, 612 00 Brno

CZECH REPUBLIC

xbartl02@stud.feec.vutbr.cz, hosek@feec.vutbr.cz, xmatoc00@stud.feec.vutbr.cz, molnar@feec.vutbr.cz,  
rucka.lukas@phd.feec.vutbr.cz

*Abstract:* - This paper is focused on the simulation tool OPNET Modeler and its possibility to communicate with external systems and applications. Due to this we are able to interconnect simulation environment with real network system and gain the simulation results more accurate and in accordance with the reality. We created communication model composed from network scenario created in the OPNET Modeler, C/C++ external application and Cisco router. This model enables data exchange between external network component and work station in OPNET Modeler scenario. This model is very useful especially in the situation when you need to include a real network device into simulation process.

*Key-Words:* BER, Esys interface, MIB, OPNET Modeler, SNMP.

## 1 Introduction

The complexity of today's networks necessitates the need for network management to optimize network performance. There are several methods to manage network equipment. The first possibility is to manage network devices locally. This method requires direct access to network devices and is time demanding. This type of management is very complicated, especially if the network consists of many devices or the network is a geographically vast. Generally, local management is not able to ensure optimal network performance and fault detection.

Remote network management represents another solution. This method does not require direct access to the network devices because it uses a remote application for the network management. The prerequisite for this type of management is that this functionality must be implemented in each network device to be managed.

An important part of the network management is network monitoring. Network monitoring allows surveying of the current state and behaviour of the network equipment. Due to network monitoring it is possible to respond to different events in the network more quickly and solve non-standard behaviour of the network devices.

The development of a monitoring and management system for network equipment is quite a complex task. Often different simulation tools can be used to improve the development process by modelling specific events rarely appearing in real networks. The simulation environment allows the developers to evaluate several

alternative solutions before their implementation into real systems. Recently there are many simulation tools available to cover different needs from the field of network and data communications. The essential difference between these tools lies in the complexity and degree of abstraction.

One of the leaders between simulation environments specialized for complex modelling and simulation of communications networks, devices and protocols is the OPNET Modeler (OM) simulation tool [1]. OM is based on an object oriented modelling approach and uses a discrete-event simulation system. OM provides a hierarchical and graphical user friendly interface with many specific editors (e.g. network editor, node editor, process editor etc.). All components in OM are modelled in the object oriented approach which provides intuitive mapping to real systems. Particular components in OM are described by C/C++ source codes and are user accessible. It allows the users to modify the source code and to add new custom functionalities if required. A unique feature of the OM is the possibility of an interaction between the OM and external systems. This possibility is ensured by a complex set of functions called External System Domain/Definition (ESD) [1], [2]. The external system might represent almost anything from a general algorithm to a specific model of a hardware entity (e.g. user designed hardware, real network devices). It gives a flexible platform to test new ideas and solutions with low cost and brings more accurate results to the entire research process.

## 2 Implementation of SNMP Protocol into OPNET Modeler

### 2.1 SNMP Protocol

The Simple Network Management Protocol (SNMP) is one of the most often used solutions for remote network management. SNMP was introduced in 1988 [3] to meet the growing need for a standard for network devices management. SNMP was developed as a temporary tool and was intended to be replaced by a solution based on the Common Management Information Service/Protocol (CMIS/CMIP) architecture. Today SNMP is still the most popular method of network management because it can be easily implemented and disposes with great interoperability [4].

SNMP is a communication platform between the SNMP agent and the SNMP manager. The agent is located at the managed device and makes accessible the configuration information and statistics of this devices. The SNMP manager is placed on the device that manages the network nodes. The manager sends requests, encapsulated into the SNMP operations, to the agent. As a response the agent usually sends back the requested data. In the case of critical events the agents can inform the manager without any previous polling using a special message called trap [4]. The communication model of SNMP is shown in Fig. 1.



Fig. 1. SNMP communication model

#### 2.1.1 SNMP Operations

There are three versions of the SNMP protocol – SNMPv1, SNMPv2 and SNMPv3. Currently the most used version is the SNMPv2. For gathering and changing management information in network devices following operations can be used [4]:

- Get,
- Get-Next,
- Get-Bulk,
- Set,
- Get-Response,
- Trap,
- Notification,
- Inform,
- Report.

Each of these operations has a standardized Protocol Data Unit (PDU) format that is used by managers and agents to send and receive information. The structure

and detailed description of the above mentioned SNMP operations can be found in [4].

### 2.2 Management Information Base

The Management Information Base (MIB) is a structured collection of configuration and measurement information of a given network device. The MIB entries are organized into a tree-like hierarchy that enables to divide the MIB database into several independent branches. The existing branches can be extended by other branches or objects corresponding to the supported functions and requirements of network component manufacturers [4], [5].

#### 2.2.1 MIB Management

The statistics and configuration values stored in the MIB are accessible through Object Identifiers (OIDs) [5]. The value corresponding to the OID represents the current state of the object. The above mentioned SNMP operations Get, Get-Next, Get-Response are used to gather and modify management information in the MIB of the SNMP agent.

When the manager sends an SNMP request to the agent it has to know the OID of the MIB's entry that wants to discover/modify. In the case when the manager needs just one entry from the agent it uses the Get-Request SNMP operation. After the agent receives this SNMP request, the value bound to the corresponding OID is obtained from the MIB. Then the retrieved information is encapsulated into the Get-Response SNMP message and sent back to the manager.

Another way to obtain the required information from the MIB is the Get-Next SNMP operation that enables to retrieve a block of values from the MIB. The Get-Next message contains only the starting OID from which the SNMP agent starts to look-up the MIB. When the manager receives the Get-Next Response from the agent it generates another Get-Next Request command repeatedly until the agent returns an error, signaling that the end of the MIB has been reached and there are no more objects left to retrieve [4].

The above described SNMP operations are used for reading from the MIB database. When the manager needs to write a specific value into the MIB it has to use the Set SNMP operation. This operation contains the specific OID and the corresponding value that should be entered.

### 2.3 SNMP Model in OPNET Modeler

The majority of application protocols available in OPNET Modeler are represented by a parametric traffic generator producing a mathematically described pattern

of network traffic. It means that the communication on the application level is simulated by dummy data units with application specific traffic parameters. On top of that, the SNMP protocol is not implicitly implemented in the current version of OPNET Modeler, but OM enables the functional modelling of custom network technologies and protocols. We used this feature to implement the SNMP communication model into OPNET Modeler in C programming language.

As a result of this work a simulation model with agent – manager architecture had been created where the components communicated with each other using SNMPv2 messages. The SNMP agent and manager were modelled by finite-state machines realized by C/C++ functions. These functions support receiving and sending of SNMP messages. The process model of the SNMP manager created in OM is shown in Fig. 2.

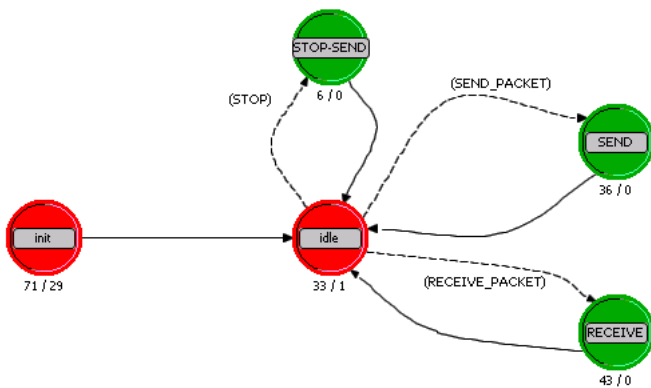


Fig. 2. Process model of SNMP manager

According to [4] the SNMPv2 message has the following standardized structure:

```

/* Variable Bindings*/
typedef struct variablebin{
    char *ObjectID;
    char *Value;
} VARIABLEBIN;

/* SNMP PDU */
typedef struct snmppdu{
    int PDUtype;
    int RequestId;
    int ErrorStatus;
    int ErrorIndex;
    VARIABLEBIN *VariableBin;
} SNMPPDU;

/* SNMP Packet */
typedef struct snmppaket{
    int Version;
    char *Community;
    SNMPPDU *SNMPPdu;
} SNMPPACKET;
  
```

Each SNMP message is created dynamically. First, the SNMP operation type (PDUtype) is defined and then the message is filled up with the OID (ObjectID) or specific MIB value (Value) according to the operation type.

In order to transmit the SNMP message in the way described in RFC 1067 [3], it was necessary to encode and decode every message using Basic Encoding Rules (BER). BER is one of the encoding formats defined as the part of ASN.1 (Abstract Syntax Notation One) and specified by the ITU (International Telecommunication Union) in X.690 recommendation [6]. A single instance of an SNMP message has to be encoded into a string of octets. BER defines how the objects are encoded and decoded so that they can be transmitted over a transport medium such as Ethernet [7].

Since BER is not available in current version of OPNET Modeler we have to implement it in C language. The C functions programmed perform encoding and decoding of every field of SNMP message as a data type identifier, the length description in bytes and the actual data. By this way each SNMP message is encoded into an octet sequence. Subsequently, this sequence is transmitted over the communication link to the destination node where it is decoded back into the original SNMP message.

The implementation of the SNMP protocol and BER encoding was the first step to develop a functional communication system between real network components and OM which is described in next chapter.

### 3 Communication with Real Network Component

#### 3.1 Possibilities of Communication between OPNET Modeler and Real Systems

There are two possibilities how to interconnect OM with real network equipment. The first possibility is the already mentioned ESD system. The second is called System In The Loop (SITL). Naturally, both of them have their advantages and disadvantages.

SITL is a separately distributed library for OM which provides an interface to link real network hardware or software applications to the OPNET discrete event simulation. External devices are connected to the simulation loop over SITL gateways operated as a bridge interface between the simulation environment and the network interface of the host computer. Packets transmitted between the simulated and real networks are converted between real and simulation formats. The SITL module is mainly focused on real-time communication with devices based on Ethernet

technology while the use of ESD system is much more versatile [1].

As mentioned before, the ESD system allows an interconnection between the OM and external system. This interconnection is called co-simulation [1]. The ESD system consists of several components. These components are Simulation description, External System Definitions model, External System Interface, co-simulation code and code implemented in external system or application. A detailed description of the ESD system can be found for example in [8].

The co-simulation is established by using a special interface of OM called External System Interface (esys). The esys interface is represented in OM by a process module. Thus it can be implemented anywhere in the model structure. The esys interface ensures the control and data exchange between OM and external systems [1]. The whole ESD system is more complex and more universal than SITL but on the other hand it does not provide such a high speed of the communication and simulation. It is because the control of simulation is shared between OM and an external system which can lead to slower event processing.

### 3.2 External Application

In order to implement a communication between OPNET Modeler and a real network node we created an external application that is used as middleware between the OM model and the network interface of the local workstation. The application reads data from OPNET Modeler, translates them into SNMP requests and sends them to a network node. Then it waits for the SNMP response from an agent implemented in the destination node. When SNMP response arrives, it is translated back into a data structure compatible with OPNET and it is entered into the simulation model.

We used the functions of OPNET API described in previous chapter for sending/reading data to/from OM. These functions are declared in the *esa.h* header file, which also contains the functions to initialize the co-simulation and register callback functions.

The external application developed is compiled as a dynamically loaded library. All the functions that should be available to the co-simulation coordinator have to be exported with code extern "C" DLLEXPORT. There are two functions of this type *esa\_main* and *callback*.

The *esa\_main* function completes the basic initialization procedures of the co-simulation and includes the Winsock library initialization (for communication through the network interface) and registration of the callback function to the ESYS interface.

The *callback* function is called, whenever data are written on this interface [9]. The data are read from the interface, encapsulated to the SNMP request and sent to the SNMP agent. After obtaining the response, the corresponding data is converted and forwarded to the simulation model through the ESYS interface.

For sending the SNMP request there is an internal function (without an export) called *snmp*. This function realises the SNMP communication using the functions and structures from the *snmp.h* and *Mgmtapi.h* windows header files [10]. It can create an SNMP Get-Request and Get-Next-Request to obtain values from the MIB database of the SNMP agent. When the SNMP structure is filled with data, the request can be sent via the *SnmpMgrRequest* function. If this call succeeds, the *SnmpVarBind* structure is filled with the value of the requested object by the SNMP agent.

The data returned by real equipment is stored in a data type, which is incompatible with OPNET Modeler, so a conversion is necessary. With respect to the type of the returned value it is converted and saved in a special structure. This structure contains a constant used to specify the value type and the ID of the returned object (saved as string). The conversion of complex data types to standard string is executed by functions *readAsnOid*, *readAsnAddress* and *readAsnString*.

After processing the data obtained, the corresponding structures are deallocated by calling the *SnmpUtilMemFree* function. Finally, the *SnmpMgrClose* function destroys the SNMP manager instance and the pointer on the structure for the converted values.

### 3.3 Simulation Scenario

In real network conditions the SNMP manager creates the SNMP message and sends it toward to the SNMP agent. By default, the SNMP message is delivered through the network to the SNMP agent using the User Datagram Protocol (UDP) transport protocol.

Our evaluation test-bed consists of a real network represented by a Cisco router C1841 and a simulation model in OPNET Modeler environment. Because of this combination of real and simulated systems, this solution has more complex architecture than the majority of real infrastructures, but at the same time, it offers several interesting research opportunities. The whole evaluation test-bed is composed from the SNMP manager created in OM, esys interface, external application and the SNMP agent implemented in a real router with SNMP support. The architecture of the test-bed is shown in Fig. 3.

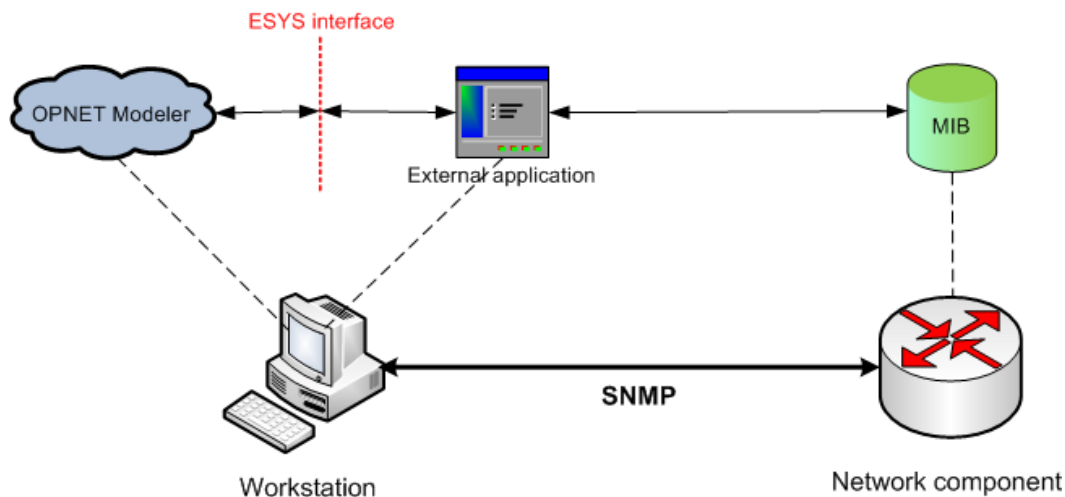


Fig. 3. Architecture of the evaluation test-bed

The communication process of our simulation scenario is described in following text. The SNMP manager creates the SNMP message in OM and sends it to the esys interface. Through the esys interface the SNMP message is passed to the external application running on local workstation. During the transmission through the esys interface data conversion must be provided. This conversion is necessary because OM and the external application use incompatible data types. The external application places the converted data into an SNMP message and sends it to the SNMP agent in UDP datagrams. The SNMP agent is located in a hardware device. After processing, the SNMP agent sends the answer in as an SNMP message back to the external application. The external application extracts data from the response and forwards them to the esys interface. In the esys interface, data is converted and sent into OM. In OPNET Modeler, the data received is processed by the model of the SNMP manager.

## 4 Conclusion

We have created a communication system able mutually exchange information between real network components and the OPNET Modeler simulation environment. This system can create an SNMP message inside the simulation environment of OM, encode it using the BER algorithm and transmit to a real network device. The destination network device can search for and read the required value from the MIB database and send it back to the OM. The interconnection of real and simulating environments opens the way towards complex simulation scenarios that can be use e. g. within the development of complex communication systems for network managing or quality of service assurance.

### Acknowledgement:

This paper has been supported by the Grant Agency of the Czech Republic (Grant No. GA102/09/1130) and the Ministry of Education of the Czech Republic (Project No. MSM0021630513).

### References:

- [1] Opnet Technologies, *OPNET Modeler Product Documentation Release 15.0*, 2009.
- [2] Hosek, J., Rucka, L., Molnar, K., DiffServ Extension Allowing User Applications to Effect QoS Control. *Proceedings of the 13th WSEAS International Conference on Communications*, 2009, pp. 39-43.
- [3] Case, J., Fedor, M., Davin, J., *Simple Network Management Protocol*, RFC1067, 1988.
- [4] Mauro, D., Schmidt, K., *Essentials SNMP, Second Edition*, O'Reilly Media, 2005.
- [5] Barker F., Chan, K., Smith, A., *Management Information Base for the Differentiated Services Architecture*, RFC3289, 2002.
- [6] *ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, ITU-T Standard X.690, 2002.
- [7] Larmouth, J., *ASN.1 Complete*, Academic Press, 1990.
- [8] Hosek, J., Rucka, L., Molnar, K., Mutual Cooperation of external application and OPNET Modeler simulation environment. *Proceedings of the International Workshop RTT 2009 Research in Telecommunication Technology*, 2009, pp. 1-5.
- [9] Opnet Technologies, Using Modeler's Co-simulation Capabilities to Integrate with External Systems, *Proceedings of the OPNETWORK 2007*, 2007.
- [10] Microsoft Corporation (2009). Simple Network Management Protocol. *Microsoft Development Library* [Online]. Available: <http://msdn.microsoft.com/en-us/library/aa377993>