.- Chapter 2: SIMULATION INSTRUMENTS : OPNET Modeler

2.1.- About OPNET Modeler

OPNET Modeler was first demonstrated at MIT in 1987. It was designed for predictive network management software, which can boost R&D productivity, improve product quality and reduce time to marker. OPNET Modeler contains many models of existing hardware and allows also quick deployment of test's environment, networks and protocols. OPNET features include graphical specification of models; a dynamic, event scheduled simulation kernel; integrated data analysis tools and hierarchical object based modeling. The three primary editors included in the OPNET Modeler package and used to construct the simulation scenario are the Network Editor, the Node Editor and the Process Editor. Many other editors are available to guide the simulation steps (i.e. Probe Editor, Simulation Editor, Analysis Editor, etc.)

OPNET Modeler is classified as an “Event Driven” simulator, in which the execution of each process is activated by a specified event. Each modeled component in OPNET is specified using at least one FSM (Finite State Machine), formed by states and transitions. The programmer can associate C or C++ code at the beginning or at the end of the state and also at the transitions between states. In this mode, the clarity of the diagrams and the power of the language are joint, making OPNET a powerful tool.

2.2 Network Editor

The Network Editor graphically represents the topology of a communications network. Networks consist of node and link objects, configurable via dialog boxes. In the Network Editor we can drag and drop nodes and links from the editor's object palettes to build the network, or use import and rapid object deployment features. We can also use objects from OPNET's extensive Model Library, or customize palettes to contain your own node and link models. The Network Editor can provide geographical context, with physical characteristics reflected appropriately in simulation of both wireline and mobile/wireless networks.

This is a graphical example:

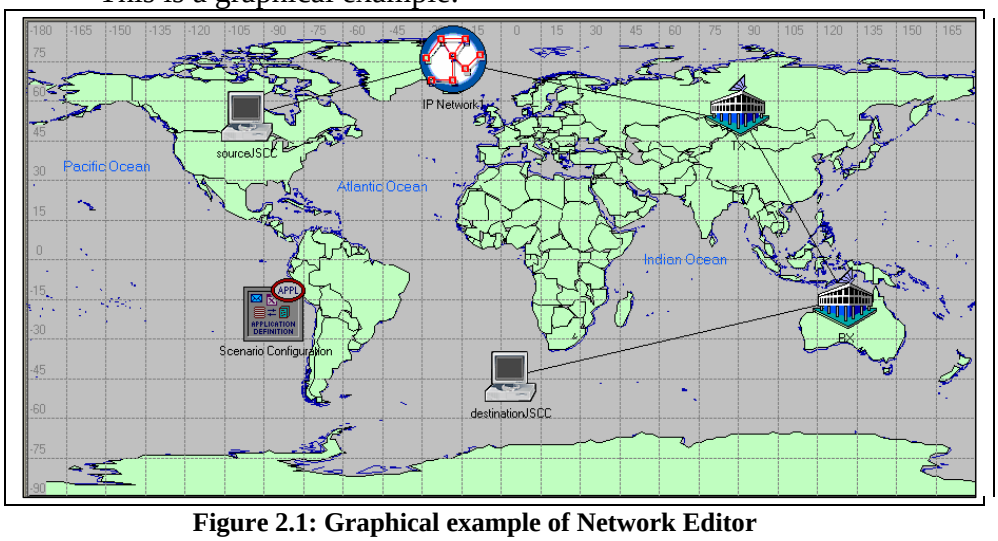

Figure 2.1: Graphical example of Network Editor

Nodes are selected from a palette and placed within the network to represent each of the communicating entities. Each node has a set of attributes, accessible via its dialog box. Reasonable default settings are already in place. For example, a router's

attributes may include its routing protocol, packet forwarding rates, interface addresses, buffer sizes, and routing table information.

Links are used to represent communication facilities between nodes. Link models can account for effects such as delays, transmission errors, and collisions. All versions of OPNET support point-to-point and bus links. OPNET Modeler/Radio adds mobile and satellite nodes and allows communication via radio channels. OPNET link models are highly customisable, if necessary.

SubNetworks are container objects that provide hierarchy to break down complexity in the network model. Within each subnetwork, you can deploy an arbitrarily complex collection of nodes, links, and other subnetworks as needed. Subnetworks can be nested to any depth.

Every object in the simulated scenario (Node or Process, see below) has model attributes useful to set its configuration. For example, considering the process model of the TCP protocol, an attribute can be set to choose the maximum congestion window size. Model attributes are called “parameters-mask”.

2.3.Node Editor

The Node Editor captures the architecture of a network device or system by depicting the flow of data between functional elements, called "modules". Each module can generate, send, and receive packets from other modules to perform its function within the node. Modules typically represent applications, protocol layers, and physical resources, such as buffers, ports, and buses. Modules are assigned process models (developed in the Process Editor) to achieve any required behaviour.

2.4.Process Editor

The Process Editor uses a powerful finite state machine (FSM) approach to support specification, at any level of detail, protocols, resources, applications, algorithms, and queuing policies.

“No forced States”, transitions and “Forced States” can be represented in the Process Editor. “Forced States” are not just a State for the process because remaining on them is not permitted. This type of state is useful for the separation of control flow actions or decisions. That means a higher modularity and a clearer definition of the process evolution.

States and transitions graphically define the progression of a process in response to events. Each state of a process model contains arbitrary C/C++ code, supported by an extensive library of functions designed for protocol programming. Each FSM can define private state variables and can make calls to code in user-provided libraries. FSM's are dynamic and can be spawned (by other FSM's) during simulation in response to specific events. Dynamic FSM's dramatically simplify specification of protocols that manage a scalable number of resources or sessions, such as TCP or ATM. We can use the Process Editor to develop entirely new process models, or use the models in OPNET's Model Library as a starting point.

The transitions between the different states are produced by established conditions and are according to the events that are sensible to.

2.5.Analyse results

The OPNET Analysis Engine provides a graphical environment that allows users to view and manipulate data collected during simulation runs. Standard and user-specified probes can be inserted at any point in the model to collect statistics.

Simulation output collected by probes can be displayed graphically, viewed numerically, or exported to other software packages. First and second order statistics on each trace as well as confidence intervals can be automatically calculated. OPNET supports the display of data traces as time-series plots, histograms, probability density and cumulative distribution functions, and scatter graphs. Graphs (as with models at any level in the OPNET modelling hierarchy) may be output to a printer or saved as bitmap files to be included in reports or proposals.