Introduction to OPNET Modeler

Karann Chew
CCSR

Contents

Session #1 (week#6, Friday, 9-10am)
- Overview
  - OPNET Environment – various editors

Session #2 (week#7, Friday, 9-10am)
- Process Modelling
- Event and event list concept
- Process Editor

Session #3 (week#8, Friday, 9-10am)
- Simulation Execution & result collection
- Others topics – model library, tools
- Assignment
Prologue

- Ask questions as they come up – do not hesitate
- Materials are based on OPNET version 7
- Prerequisites for OPNET
  - Ability to program in C or C++ or at least be comfortable in reading/understanding C or C++ code
  - Basic understanding of networks
- Disclaimer:
  - Some materials are adopted from OPNETWORK 2000

Introduction

- Simulation is an abstract concept
  - Definition - *a process that represents another process*
- Goals
  - Study behavior of systems that are not well understood
  - Allow measurement of difficult quantities
  - Analyse measurement
  - Allow modification of a system to see how it performs when changed
A Simulation Study: Flow Diagram

Start

Understanding the system

Understanding your goals for the simulation

Choosing aspects to be modeled

Defining input and output

Specifying the system model

Choosing input and running simulations

System results accurate?

Yes

No

Results sufficiently detailed?

No

End

Results statistically useful?

No

Yes

Session I

OPNET Overview
Agenda

- OPNET Overview
- OPNET Hierarchy
- Navigating Various Editors: Project, Node, Process Editor
- Other Editors: Packet format, ICI format, Antenna, Modulation, PDF etc.

OPNET OVERVIEW

- Originally developed at MIT, introduced in 1987 as the first commercial network simulator.

- An Integrated Network Modelling Environment
  - Tools to build network model - LAN, WAN, Wired, Wireless, Mobile, & Satellite
  - Tools to construct, execute & debug simulation
  - Tools to collect, analyse & present results
  - Interactive debugging engine
  - Built-in & contributed model library
  - Tutorial & online documentation
**OPNET Features**

- Hierarchical network models
- Finite state machine modeling
- Integrated analysis tools
- Comprehensive library of detailed protocol and application models and network devices
- Object-oriented modeling
- Wireless, point-to-point, and multipoint links
- Geographical and mobility modeling
- Animation
- Integrated debugger
- Financial cost attribute for devices

---

**Modelling Methodology & Hierarchy**

- OPNET utilizes different model format for topology, data flow, and control flow
- All fit together in a hierarchical fashion
Three-Tiered Hierarchy

- **Network Model** - define the position and interconnection of communicating entities, or nodes. Each node is described by a block structured data flow diagram.

- **Node Model** - depicts the interrelation of processes, protocols, and subsystems. Each programmable block in a Node Model has its functionality defined by an process model.

- **Process Model** - combines the graphical power of a state-transition diagram with the flexibility of a standard programming language and a broad library of pre-defined modeling functions.

Three-Tiered Hierarchy (cont.)

- Node model specifies object in network domain
- Process model specifies object in node domain
Main Tools

- Project Editor (Network Editor)
  - Drag Nodes from the Object Palette & Drop It to the Desired Place
  - Connects nodes with links

- Node Editor
  - Select modules & connection them by streams
  - Specific process model for the modules

- Process Editor
  - Add states & connect them by transitions
  - Add in C/C++ codes & kernel functions

- Simulation Tool
  - Specify the simulation parameter & run simulation

Other Tools

- Link Editor
- Packet Editor
- ICI Editor
- Probe Editor
- Antenna Pattern Editor
- Modulation Curve Editor
- PDF Editor
- Filter Editor
- Icon Database Editor
Project/Scenario Workflow

- Create project
- Create baseline scenario
  - Import or create topology
  - Import or create traffic
  - Choose results and reports to be collected
  - Run simulation
  - View results
- Duplicate scenario
  - Make changes
  - Re-run simulation
  - Compare results

Project Editor
Network Domain

- Network models consist of nodes, links and subnets deployed in a geographical context.
- Nodes represent network devices and groups of devices
  - servers, workstations, routers, LAN nodes, IP clouds, etc.
- Links represent point-to-point and bus links.
- Icons assist the user in quickly locating the correct nodes and links.
- Vendor models are distinguished by a specific color and logo for each company.

Generic Devices

Vendor Devices
**Node Domain**

- Basic building blocks include processors, queues, and transceivers
  - Processors are fully programmable via their process model
  - Queues also buffer and manage data packets
  - Transceivers are a node’s interface
- Interfaces between blocks
  - Packet streams
  - Statistic wires

**Node Editor**
OPNET process models consist of:

- State transition diagrams
- Blocks of C codes
- The library of OPNET Kernel Procedures (KPs)
- State variables (private to each process)
- Temporary variables

Processes can create additional child processes dynamically.
Processes respond to interrupts.
Simulation Tool

The graph below shows the output summary of the congestion window feature of the link comparison between the TCP Reno and the TCP Tahoe.

Here the TCP Reno window faster due to its implementation.

Probe Editor

The Probe Editor - to configure where information is collected and what statistics are measured.

Can save collections of probes to a file for use in different scenarios.

1. Global Statistic Probe
2. Node Statistic Probe
3. Link Statistic Probe
4. Coupled Statistic Probe
5. Attribute Probe
6. Automatic Animation Probe
7. Statistic Animation Probe
8. Custom Animation Probe
**Links**

Link objects model physical layer effects between nodes, such as delays, noise, etc.

A point-to-point link transfers data between two fixed nodes.

A bus link transfers data among many nodes and is a shared media.

A radio link, established during a simulation, can be created between any radio transmitter-receiver channel pair. Satellite and mobile nodes must use radio links. Fixed nodes may use radio links. A radio link is not drawn but is established if nodes contain radio transceivers.

**Link Editor**

The Link Editor is used to configure and manage the links between nodes. It provides a graphical interface to select and modify various link properties, such as type, supported protocols, and attributes.
Packets

- Carry information between objects in node and network domains
- Can represent messages, packets, cells
- Either formatted or unformatted
- Data areas
  - User-defined
  - OPNET defined

Packet Format Editor
Session II
Process Modeling

Agenda
- Event-driven simulation
- Event and Event List concept
- Simulation Kernel
- Process Editor
- Process Modeling and Execution
- Kernel Procedures
Event-Driven Simulation

- An event is a request for a particular activity to occur at a certain time
- OPNET simulations are event-driven
- Simulation time advances when an event occurs
- A different method might be to sample at regular intervals...
- ...But disadvantages are as follows:
  - Accuracy of results is limited by the sampling resolution
  - Simulation is inefficient if nothing happens for long periods

Event-List Concepts

- Single global event list
- Shared simulation time clock
- Events scheduled in time order
- Event removed from event list when it completes

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Type</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>Initialize</td>
<td>src.gen</td>
</tr>
<tr>
<td>0.0</td>
<td>Initialize</td>
<td>src.rte</td>
</tr>
<tr>
<td>4.3</td>
<td>Timer expires</td>
<td>src.gen</td>
</tr>
<tr>
<td>4.3</td>
<td>Packet arrives</td>
<td>src.rte</td>
</tr>
</tbody>
</table>
The Simulation Kernel

- The Simulation Kernel (SK) manages the event list
- The SK delivers each event, in sequence, to the appropriate module
- The SK receives requests from modules and inserts new events in the event list

How Event List Work

1. New event reaches head of event list, causes Simulation Kernel to deliver an interrupt to the appropriate module
2. Process within the module gains control and processes interrupt
3. Simulation Kernel deletes event from event list, allowing new event to reach head of list
4. Simulation Kernel regains control from module
**Interrupts**

- First event in the event list becomes an interrupt
- Delivered by the Simulation Kernel to the designated module
- Data associated with the event can be obtained by the module
- When an interrupt is delivered to a module, control passes from the Simulation Kernel to the module
- Processors and queues can have `BEGSIM` interrupts

---

**Event-List Concepts**

- Some events must be entered in the event list at the start of a simulation
  - A processor or queue module could have the `begsim interrupt` attribute enabled
- An event list typically has a few events – each event spawns another event or two that is placed on the list as the spawning event is deleted
- The event list is always growing and shrinking
- An event is pending until executed
- A pending event can be cancelled
Process Models

- Process models to represent both hardware and software – Finite State Machine
- OPNET process models describe the logic of real-world processes, such as:
  - Communications protocols and algorithms
  - Shared-resource managers
  - Queuing disciplines
  - Specialized traffic generators
  - Statistic-collection mechanisms
  - Operating systems
- FSM + C/C++ programming code + library functions
- FSMs are dynamic and can be spawned (by other FSMs) during simulation in response to specific events
- The Process Editor provides the necessary features for specifying process models

Process Editor

Toolbar

1. Create state
2. Create transition
3. Set initial state
4. Edit state variable
5. Edit temporary variable
6. Edit header block
7. Edit function block
8. Edit diagnostic block
9. Edit termination block
10. Compile process model
Process Models

Consists of:
- State transition diagrams
- The complete C programming language
- The library of OPNET Kernel Procedures
- State variables (private to each process)
- Temporary variables

Executive blocks

Each state has two executive blocks
- Enter executives are invoked on entering a state
- Exit executives are invoked before exiting a state
State Connections - Transitions

Transitions describe the possible movement of a process from state to state and the conditions allowing such a change.

Exactly one condition must evaluate to true.

If the condition statement \((x == y)\) is true, the transition executive \((\text{Reset}_\text{Timers})\) is invoked.

Forced States

Forced (green) and unforced (red) states differ significantly in execution timing.

In a forced state, the process:
- Invokes the enter executives
- Invokes the exit executives
- Evaluates all condition statements
- If exactly one condition statement evaluates to true, the transition is traversed to the next state.
**Unforced States**

- In an unforced state, the process:
  - Invokes the enter executives
  - Places a marker at the middle of the state
  - Releases control to the Simulation Kernel and becomes idle
  - Resumes at the marker and processes the exit execs when next invoked

**Transitions Between States**

- After completing the exit executives, the process evaluates the condition statements of all departing transitions from the state
- One and only one condition statement must evaluate to true
- The process traverses the transition associated with this condition statement
- A transition with condition = “default” is true if and only if no other conditions are true
- A transition with no condition set is termed *unconditional* and is always true
**Process Model - Example**

Model with three forced states and one unforced state:

1. Initial interrupt delivered and the enter execs invoked.
2. Exit execs invoked immediately. Transition condition \( pk\_count == 0 \) evaluates to true.
3. Transition occurs.
4. Enter execs invoked.
5. Exit execs invoked immediately.
6. Transition occurs.
7. Enter execs invoked.
8. Marker is placed and process stops here.

**Event-List - Example**

Node model: src

Network model
Event-List Example (cont.)

- The network model has three nodes (`src`, `dest1`, `dest2`) relying on two node models (both dest nodes use the same node model).
- In the `src` node model, packets are generated at `gen` and sent by `queue` to either transmitter (`tx0` / `tx1`).
- Three modules (`gen`, `queue`, and `sink`) have process models associated with them.

The BEGSIM Interrupt

- BEGSIM is a special type of interrupt that occurs at simulation time 0.0, before any other type of interrupt.
- A BEGSIM interrupt usually initializes a module and schedules future events.
- Any processor or queue can have its `begsim intrpt` attribute enabled, resulting in an event being placed on the event list for time 0.
Processing the First Interrupt

Consider the process model specified by the \texttt{src.gen} module

Process model: gen

Starting the Simulation

1. To start the simulation, the head of the event list is processed and a BEGSIM interrupt is delivered to the process in module \texttt{src.gen}
2. Because this is the first interrupt, the process begins execution at the initial state, marked with a black arrow
3. Because this state is being entered, the enter execs are executed
Processing the First Interrupt in Process gen

4. Process invokes and completes the enter execs of Init.

5. Because Init is a forced (green) state, process immediately invokes and completes the exit execs.

6. Process evaluates all condition statements. This state has only one departing transition which evaluates to true.

8. Process invokes and completes the enter execs of Wait.

9. One action in the enter execs (line 9) is to schedule a self interrupt by means of a KP. This adds an event to the event list.
10. Process places a marker at the middle of Wait.
12. Simulation Kernel removes the first event and advances the next event to the head of the event list. The simulation time remains 0.0.

13. Simulation Kernel delivers a BEG SIM interrupt to src.queue.

14. Process in src.queue module gains control. It executes until it reaches an unforced (red) state, places a marker, and then becomes idle. (This model is not shown.)

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Type</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>BEG SIM</td>
<td>src.queue</td>
</tr>
<tr>
<td>4.3</td>
<td>SELF</td>
<td>src.queue</td>
</tr>
</tbody>
</table>

Processing the Next Interrupt at module src.gen

15. The Simulation Kernel removes the previous event and advances the next event to the head of the event list. The simulation time becomes 4.3 seconds.

16. The event at the head of the list causes a SELF interrupt to be delivered to the gen process. The process resumes at the marker in the middle of Wait.

17. Process invokes and completes the exit execs of Wait.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Type</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>SELF</td>
<td>src.gen</td>
</tr>
</tbody>
</table>
Continuing the Process at gen

18. To leave Wait, the process evaluates all condition statements. This state has one outgoing transition and its condition statement (shown) evaluates to true.

19. The process transitions to Send.
20. The process invokes the enter execs of Send and calls `op_pk_send()` to send a packet. This results in an event of type STRM being placed on the event list.
Continuing the Process at gen (cont.)

21. Because send is a forced (green) state, the process immediately invokes and completes the exit execs.
22. Process evaluates all possible transitions. One evaluates to true.
24. Process invokes enter execs of Wait, schedules another SELF interrupt in the event list and becomes idle.
25. Simulation Kernel takes control and processes the next event in the list.

Simulation Termination

Simulations terminate in one of four ways:
- The event list is emptied
- Simulation attribute duration expires
- A process calls for termination, using the KP op_sim_end()
- A fatal error occurs
How Does Time Advance?

- Simulation time advances only when an event with a later time is taken from the event list.
- No simulation time occurs during an invocation of a process model.
- No time elapses during transitions between states.
- A process model must always end in a red state so time can advance.
- Avoid endless looping between forced (green) states.

Kernel Procedures

- OPNET’s library functions that abstract difficult, tedious, or common operations.
- KPs free users from addressing memory management, data structure, handling event processing, etc.
- Can be called within a process model.
- KPs focus on communication modeling.
- Categorized based on type of object to manipulate.
- Categorized by primary function into 22 packages:
Kernel Procedures

Naming convention for Kernel Procedures:
\( \text{op}_{\text{<package name>}}_{\text{<action>}} \)

Example:
- \( \text{op}_p\text{k}_\text{get()} \)
- \( \text{op}_\text{intrpt}_\text{type()} \)

Commonly used KPs:

Packet Package:
- \( \text{op}_p\text{k}_\text{create()} \)
- \( \text{op}_p\text{k}_\text{create}_\text{fmt()} \)
- \( \text{op}_p\text{k}_\text{copy()} \)
- \( \text{op}_p\text{k}_\text{get()} \)
- \( \text{op}_p\text{k}_\text{total}_\text{size}_\text{get()} \)
- \( \text{op}_p\text{k}_\text{nfd}_\text{get()} \)
- \( \text{op}_p\text{k}_\text{send()} \)
- \( \text{op}_p\text{k}_\text{send}_\text{delayed()} \)
- \( \text{op}_p\text{k}_\text{destroy()} \)

Subq Package:
- \( \text{op}_p\text{subq}_\text{pk}_\text{insert()} \)
- \( \text{op}_p\text{subq}_\text{pk}_\text{remove()} \)

Distribution Package:
- \( \text{op}_\text{dist}_\text{load()} \)
- \( \text{op}_\text{dist}_\text{outcome()} \)

ID, Topo and Internal Model Access Packages:
- \( \text{op}_\text{id}_\text{self()} \)
- \( \text{op}_\text{topo}_\text{parent()} \)
- \( \text{op}_\text{topo}_\text{child()} \)
- \( \text{op}_\text{ima}_\text{obj}_\text{attr}_\text{get()} \)

Interrupt Package:
- \( \text{op}_\text{intrpt}_\text{schedule}_\text{self()} \)
- \( \text{op}_\text{intrpt}_\text{type()} \)
- \( \text{op}_\text{intrpt}_\text{strm()} \)
- \( \text{op}_\text{intrpt}_\text{code()} \)

Simulation and Event Packages:
- \( \text{op}_\text{ev}_\text{cancel()} \)
- \( \text{op}_\text{sim}_\text{time()} \)

Summary

Process models - Finite state machines
Forced and unforced states differ in when they execute their instructions
Any processor or queue can have the attribute \( \text{begsim intrpt enabled} \), scheduling an event for time 0.0
Control passes dynamically between the Simulation Kernel and individual processes
Kernel Procedures
Session II
The End
Q&A ?

Session III
Simulation Design & Execution
**Agenda**

- Running Simulation
  - Simulation Outputs
  - Statistics
  - Simulation Execution
  - Result Viewing & Analysis
- Model Libraries
- Others Tools + Tips
- Assignment

---

**Simulation Output**

- **Three kinds of Simulation Outputs:**
  i) Vector Output
     - A series of values and associated time
     - Monitor changes with time in one simulation
  ii) Scalar Output
     - A single value in one simulation
     - Monitor average value during one simulation
     - Multiple simulation often needed to generate a series of values with different input
  iii) Animation
     - Visualize the activity of model, e.g. Packets flow over links, Node movement
- Vector & Scalar output file can be viewed and manipulated in built-in analysis tool
Simulation Output

- Example:

Vector Output
  - x axis = simulation time

Scalar Output
  - x axis = traffic rate

Type of Statistic

- Local Statistics
  - Used to collect the statistics generated by an object, eg. a node, link, module etc.
  - Access by placing a probe to the specified object

- Global Statistics
  - Used to collect data generated by different objects in the model
  - Used to collect the statistics generated by the overall system
  - Access by their names
Declaring Statistics

Statistics - declared in the process model
- Local statistics gather data from an individual process
- Global statistics gather data from the system as a whole

Enter the Statistic name in Stat Name.

Mode is either Single or Dimensioned.

Count is the number of dimensions in a Dimensioned Statistic.

Description is a text field to supply a brief description of the statistic.

Statistics Collection

Probes - passive data collection devices
Probe Editor used to create probes

Type of Probes
i Statistic – collect output
ii Attribute – collect input
iii Animation – collect output
Statistic Capture Modes

- Normal mode: Every data point is collected from a statistic
- Sample mode: The data is collected according to a user-defined time interval or sample count
- Bucket mode: All the data points in a bucket are collected and processed according to a user-defined parameter
  - Max
  - Min
  - Sum
  - Count
  - Sample average
  - Time average

Running Simulation

- From the Project Editor...
  - Choosing results/probes
  - Configuring & run simulation
  - Viewing collected results
Configuring Simulations

Set simulation attributes by choosing “Configure Simulation” from the Simulation menu

- Duration – duration of simulation, in simulation time
- Values per Statistic – number of values to be collected for each statistic
- Seed - Random number generation seed
- Simulation Attribute – Values for attributes relevant to the network model being simulated
- Generate Web Report – If checked, the simulation will produce a web report for the results

Simulation Dialog Box

Running a Simulation

The simulation window shows the progress of simulation and error messages, if any.
During simulations, the elapsed time bar monitors the status of the simulation.
Appears after 1,000,000 events

- Progress: Simulation time elapsed and number of events processed
- Time: Real time elapsed and remaining
- Simulation Log: Number of entries present in the simulation log
Viewing Results

Results can be displayed by:

- Selecting “View Results” from the Results menu
- Right-clicking the project workspace and selecting from the pop-up menu
- Selecting the “View Results” button on the tool bar

The “View Results” dialog box allows the user to select the results to display.

Note: Only the statistics you chose for collection will be available.

The “Show” button in the “View Results” dialog box displays a graph of the selected statistics.

View Result Dialog Box

Viewing Results (cont.)

Multiple panels can be open at the same time.

Each panel can contain one or more traces that graphically depict data either Overlaid or Stacked.

Graphs overlaid

Graphs stacked
**Simulation Sequence Editor**

- By promoting an object’s attribute to the simulation level, multiple values for that attribute can be set.
- By left-clicking in a blank field under “Attribute”, the user can add a currently promoted attribute.
- A sequence of Simulations will be run with each value.

---

**Exporting Data**

- Modeler allows the user to export collected statistics to a text file or to a user-specified spreadsheet application.
- Users can specify the path of the desired spreadsheet application in the user preferences.
- If no spreadsheet application is specified the data is written to an ASCII file.
Exporting Data - Spreadsheets

User right-clicks inside graph on desired panel
Selects “Export Graph Data to Spreadsheet”
Modeler launches the appropriate program with data

Generating Web Reports

Web Reports give the Modeler user a common interface with non-Modeler users
Reports are automatically generated and present data in a manner that is easy to use, navigate, and interpret
Web Reports containing data and statistic graphics can be created after a simulation has run
The reports will be HTML-formatted pages that can be read by any Web browser
Two methods for generating web reports

Before running a simulation, check the option in the “Configure Simulation” dialog box

After running a simulation, select “Generate Web Reports” from the Results menu

Web Reports can take a while to generate due to the many graphic files that must be created

How to Generate Web Reports

A suite of models of network protocols, technologies, and applications available.

OPNET Standard Models

• Support majority of users
• General Models
• Protocol Models

Specialised Models

• Support user with particular interest in emerging or vendor-specific technologies

Contributed Models

• Free from OPNET user communities

Model Library
Standard Model - General

- Application Models - Traffic Generator
  - Email, File Transfer, Web browsing, …
- Process Models – Queuing
- Pipeline Stages
  - Bus, Point-to-point, Radio
- Modulation Functions
  - BPSK, DPSK, MSK, …
- Jammer Model
- Maritime Satellite Model
  - GEO Satellite, ALOHA Multiple Access
- SWP - Sliding Window Protocol Model
- ISDN / xDSL Models
- Vendor Models
  3Com, Ascend, Bay Networks, Cabletron, Cisco System, Fore Systems, Hewlett Packard, Newbridge, Novell, Xylan

Standard Model – Protocol Specific

- Ethernet
- FDDI
- LANE, LAPB
- Token Ring
- X.25
- Frame Relay
- ATM
- IP, IPX
- Multicast
- BGP, IGRP, RIP, OSPF
- RSVP
- TCP, UDP, NCP
Specialised Model

- MPLS
- IP Multicasting
- UMTS
- Circuit Switch

Contributed by OPNET user community
Freely Downloadable from OPNET website

Examples:
- Army Force: both voice and data packets in a military mobile scenario.
- Cellular Sample Models: use of the AMPS cellphone standard models
- GPRS
- SS7
- Mobile IP
Common Pitfalls

- OPNET is a powerful tool
- Common pitfall: “I will model everything”
  - Developing model without clear direction
  - No idea of questions to answer
  - Perform more computation and use more memory than necessary
- Model only what impacts results
  - May be complex and time consuming to model everything
  - Some parts of the system don’t have a major impact on overall performance

Questions Should Drive Model Design

- Decide which questions to answer
- Design model to answer those questions
  - Don’t try to answer everything
- Design multiple models
  - Answer different questions
  - Study different aspects of your system
- Limit your model
  - Less fidelity pays off in higher performance
- Flexibility still important
  - Questions may change
The Pluses of OPNET

- An Integrated modeling, simulation & analysis environment
- Object Orientation
  - Objects & Attributes
- Hierarchical Models
  - Network, sub-network, nodes, module, process
- Graphical Specification
  - Visualising models, good presentation, animation
- Support of Node Mobility and Satellite orbit
  - Predefined or adaptive trajectories, orbits radio link models
- Vast set of model library

The Minuses of OPNET

- Long learning period
- Programming experience necessary
- Longer debugging time – if using library model
- Some library model are extremely complex
Summary

- Statistic declaration in Process model
- Statistic collection Probe Editor
- Configure & run simulation, view result in project editor
- Simulation Sequence Editor
- Exporting topology and data to web
- Model libraries

More Information

- OPNET WWW page
- OPNET Documentation
- OPNET Support
  - Contact Points in the University
Session III
The End
Q&A ?