OPNET Workshop

Introduction to OPNET Modeler

Dr Steven Gordon
Sirindhorn International Institute of Technology
Thammasat University

June 2010

Supported by:
- Telecommunications Research and Industrial Development Institute (TRIDI), National Telecommunications Commission (NTC)
- Prince of Songkla University (PSU)
- Sirindhorn International Institute of Technology (SIIT), Thammasat University
Introduction to OPNET Modeler

Workshop Overview

OPNET Modeler is a software environment for modelling and analysing the performance of computer networks and protocols. This workshop aims to introduce you to OPNET Modeler, with focus on how it can be used for telecommunications research. The basic operations and features of OPNET Modeler will be explained, followed by several hands-on tasks, where you will use OPNET Modeler to setup a simple simulation, create your own traffic and application profiles, and modify an existing protocol. These are key tasks when using OPNET Modeler for simulation of new/modified protocols and algorithms in your research.

At the end of the workshop you should have the following knowledge:

1. Understand how OPNET Modeler is structured
2. Navigate the menus and operations of OPNET Modeler
3. Create models of networks using existing models
4. Configure, run and analyse results from simulations
5. Knowledge of existing device models and capabilities
6. Understand the implementation of Process Models
7. Create your own model of a protocol

This workshop should be of benefit to telecommunication researchers, especially graduate students and faculty interested in detailed analysis of computer and communication networks (e.g. wireless LAN, ad hoc networks, IPv6, routing algorithms, voice/video over IP, WiMax, 3G/LTE).

The “Hands On” sessions will allow each participant to use OPNET Modeler to complete short tasks. You will use the computers provided, each running OPNET Modeler.

OPNET Modeler is a large and complex software package. This workshop will only cover selected features which are important in getting started with OPNET Modeler for research. Most of the source code presented in the workshop will be in C: basic knowledge of C programming is assumed. Knowledge of computer networking is also assumed (e.g. layering concepts especially related to the Internet, protocol mechanisms, packets, performance metrics).

The workshop will be led by Dr Steven Gordon, Assistant Professor at Sirindhorn International Institute of Technology (SIIT), Thammasat University. He has used OPNET Modeler (as well as other simulation tools, e.g. NS2, Glomosim) for research over the past 10 years, and recently introduced OPNET software in a course taught at SIIT. Questions and comments on OPNET are welcome.

Email steve@siit.tu.ac.th
Web http://ict.siit.tu.ac.th/~steven/
Phone +66-2-501-3505 ext 2014
OPNET and OPNET Modeler

Who is OPNET? www.opnet.com
  ▶ US company creating/selling network design and management software
  ▶ Founded by MIT graduate that developed a network simulator OPNET Modeler

What is OPNET Modeler?
  ▶ Discrete event network simulation software
  ▶ Used for network/protocol design and analysis

Other OPNET Products
  ▶ ACE Analyst, IT Guru Systems Planner, ...
  ▶ IT Sentinel, nCompass, IT Guru Network Planner, ...

Why OPNET Modeler?
Alternatives: ns2/ns3, Qualnet/Glomosim, OMNeT++, ...

Advantages
  ▶ Extensive library of detailed models
  ▶ Long development history, experience
  ▶ Detailed documentation and technical support
  ▶ Built-in statistics collection and results presentation
  ▶ Logical, hierarchical structure of models

Disadvantages
  ▶ Expensive!
  ▶ Closed-source simulator
  ▶ Slower to include models of new technologies
Workshop Outcomes

Apply OPNET Modeler to network research
1. Understand how OPNET Modeler is structured
2. Navigate the menus and operations of OPNET Modeler
3. Create models of networks using existing models
4. Configure, run and analyse results from simulations
5. Knowledge of existing device models and capabilities
6. Understand the implementation of Process Models
7. Create your own model of a protocol

Assumed Knowledge

▶ Computer networking, e.g. layering, protocol mechanisms, performance metrics
▶ C programming
▶ Ability to explore menus, manuals and files on your own

Workshop Format

Topics
1. Introduction to OPNET
2. Analysing networks
3. Editors and features
4. Process Models

Hands On
1. OPNET examples
2. Ethernet/WLAN
3. OPNET examples
4. Sensor Network

Contents

OPNET and OPNET Modeler

Simulation Analysis Methodology with OPNET
Performance of Real Networks

A real network has:

- **Topology**: arrangement of devices and links
- **Configuration**: protocols, parameter values and options selected
- **Users**: generating and receiving traffic via applications
- **Mobility**: users/devices moving

Understanding performance of the network

Specify a set of statistics to measure during operation

- Application response time, link utilization, data throughput, error rates, ...

A Methodology for Simulation Performance Analysis

1. **Create a network topology**
   Select the area of the network; position hosts, switches, routers, antennas, servers in the area; connect devices via links

2. **Configure devices and protocols**
   Examples: set link data rate to 1Mb/s; router forwarding to 500,000pps; mobile host to use IEEE 802.11g

3. **Specify traffic from users**
   Select the types of applications users are using, e.g. web browsing, email, voice call; specify the characteristics of the applications, e.g. voice source generates 50pps at 128Bytes.
A Methodology for Simulation Performance Analysis

4. Specify the mobility of users/devices
Specify a trajectory of devices throughout network area; user-defined or modelled (e.g. random waypoint, grid-based)

5. Select statistics to measure
Per node or entire network; throughput, bytes sent/received, delay, jitter, server load, link utilisation, ...

6. Setup and run simulations
Duration of simulation; number of runs and random seeds; simulation parameters

7. Analyse the results
Raw data, plots and reports
OPNET Modelling Hierarchy

Network Model
Create network topology using existing node models

Node Model
Models of devices (e.g. switches, PCs, routers, links)
Created using existing process models

Process Model
Models of applications and protocols (e.g. HTTP, TCP, IEEE 802.11)
Created using state diagrams and C source code

Model Library
- Protocols
  - ATM, BGP, DHCP, DOCSIS, EIGRP, EthCoax, Ethernet,
  - FibreChannel, FDDI, FrameRelay, H323, IGRP, IKE, IP, ISIS,
  - L2TP, LANE, LAPB, OSPF, RIP, RSVP, RTP, SIP, TCP,
  - TDM, X25, xDSL
- Applications
  - HTTP, FTP, Email, Database, Voice, Video, Print, Rlogin,
  - Generic request/response
- Wireless
  - IEEE 802.11, MANET (AODV, DSR, OLSR, ...), ZigBee,
  - WiMax, UMTS
- Others
  - MPLS, PNNI, PSTN, Servers, community contributed models

Internal Structure of OPNET
- A network consists of multiple nodes; each node consists of multiple processes
- For a simulation, OPNET compiles source code (C/C++) of each Process model used
- Also included is Simulation Kernel:
  - Controls the execution of processes
  - Manages list of events to occur
- Resulting executable is then run, saving output in results files

States and Events
- Process models are state-based
  - In state \( x \) if event \( e \) occurs then action \( a \) is taken and enter state \( y \)
  - Implemented as graphical state-machines and C/C++ code
- Events are called interrupts
  - A process schedules interrupts, e.g. timeouts
  - Kernel issues interrupts, e.g. packet arrives, channel status changes

Kernel manages the Event List
- 0.004503 Timer \( t_1 \) expires at process \( X \)
- 0.004515 Packet \( p_1 \) arrives at process \( Y \)
- 0.004602 Packet \( p_2 \) arrives at process \( X \)
- 0.004603 Timer \( t_2 \) expires at process \( Z \)
- ...
Analysing Networks

Steven Gordon
Sirindhorn International Institute of Technology
Thammasat University

Lab Tasks

1. Creating a new project
2. Create network topology
3. Edit node attributes
4. Specify traffic from users
5. Select statistics to collect
6. Run the simulation
7. View the results

Create a New Project

Task 1

Use an empty scenario, create an office network of 100x100m and select the ethernet and wireless_lan model families

Projects and Scenarios

A project may have multiple scenarios.

Model Families

The Model Families selected gives you quick access to these models. You can still access all other models.

Create Network Topology

Task 2

1. Create a switched Ethernet LAN that has: two workstations, one server and a 16-port switch
2. Create a Wireless LAN that has: two wireless clients and an integrated access point/IP router
3. Connect the Ethernet hosts and router via the switch using 100 Base-T links

Node/link models: ethernet_wkstn, ethernet_server, ethernet16_switch, 100BaseT, wlan_wkstn, wlan_ethernet_router

Stations vs Workstations

- Workstations are full IP clients; stations are simple hosts without IP support
**Edit Attributes**

**Task 3**
1. Give the nodes meaningful names
2. Set the three wireless nodes to use IEEE 802.11g at 54Mb/s

**Addressing**
By default, IP addresses are *Auto Assigned*: OPNET assigns IP addresses to each node when the simulation starts.

**Specify Traffic From Users**

**Task 4**
1. Add the Profile Definition and Application Definition objects
2. Add two applications:
   2.1 WebBrowsing: Http is Image Browsing
   2.2 FileTransfer: Ftp is High Load
3. Add two profiles:
   3.1 Student: using WebBrowsing application
   3.2 Lecturer: using all three applications, where the profile starts within uniform(300,310) and operation mode is Simultaneous
4. Set the server to support all services
5. Set one wireless LAN client to have Student profile and other clients to have Lecturer profile

**Select Statistics to Collect**

**Task 5**
1. On nodes, *Choose Individual DES Statistics*:
   - Server: ServerFtp, ServerHttp
   - Switch: Switch
   - Laptops: ClientHttp, WirelessLAN
   - Switch—WirelessRouter: Utilization
2. From DES menu, *Choose Individual Statistics*:
   - Ethernet, WirelessLAN, Ftp

**Configure and Run the Simulations**

**Task 6**
1. From DES menu, *Configure/Run Discrete Event Simulation*
2. Set duration to 10 minutes
3. Run!

**What does OPNET do?**
- Processes are compiled (if changed since last simulation)
- Displays simulation progress, including number of events and simulated time
- DES Log: warnings and errors will be recorded (open via DES menu)
View the Results

Task 7

1. From DES menu, Results→View Results...
   1.1 HTTP statistics: Laptop1 page response time; Server load; Server task processing time
   1.2 Global wireless LAN throughput
   1.3 Wireless Router→Switch utilization

Hide plots leaves them available for viewing later.
Results can be shown As Is or with mathematical operators applied.
OPNET Editors and Features

Steven Gordon
Sirindhorn International Institute of Technology
Thammasat University

OPNET Editors

- Commonly used editors:
  1. Project editor: creating topology
  2. Node editor: modifying devices
  3. Process editor: modifying protocols
- Other useful editors:
  - Packet format: specify structure of packets, frames, messages
  - PDF: create your own probability density function
  - Probe: define your own statistics and collection mechanisms
  - Simulation Sequence: configure batch simulations
  - Antenna Patterns, Modulation Curve, Demands, ...

Examples

Packet Formats
TCP Segment: std/tcp/tcp_seg_v2.pk.m
TCP Segment Support (Header): std/include/tcp_seg_sup.h

Antenna Patterns
UMTS: std/umts

Modulation Curves
Generic wireless: std/wireless
IEEE 802.11: std/wireless_lan
Scenarios and Hierarchy in Projects

Scenarios

- A project may have multiple, independent scenarios
- Used for different topologies/configurations

Hierarchy

- Hierarchy in a network topology is achieved using subnets
- Example: IP project, cloud modelling scenario

Creating a Topology

Different methods:

1. Manual
2. Create bus, star, mesh, random topologies: Topology→Rapid Configuration
3. Deploy WLAN/WiMax nodes in common topologies: Topology→Deploy Wireless Network
4. Import/export from/to other software and file formats

Making it Look Good

Maps and Backgrounds

- Display world map with cities: View→Background→Set Properties...
- Display city maps or other images: View→Background→Add Image...

Annotations

- Comments/text, rectangles, circles: Topology→Open Annotation Palette
Generating Traffic

Explicit Traffic

▶ Each individual packet is simulated
▶ Detailed, but can make simulations slow
▶ Types: Packet generation based on distributions; Application traffic models (e.g. HTTP, FTP); Application demands

Background Traffic

▶ Analytical models used, individual packets not simulated
▶ Course-grained modelling, speed up simulations
▶ Types: Traffic flows between source/destination; Background utilization; Application demands

Packet Generation using Distributions

▶ Generic source process generates packets to selected destination(s)
  1. Time to start/stop generating packets
  2. Burst duration (on/off time)
  3. Interarrival time
  4. Packet size
▶ Operates directly above MAC (no IP or transport protocol)
▶ Distributions: constant, uniform, exponential, ...

Example: WLANSimple

Application Demands

Traffic Flows from Source to Destination

▶ Different layers: Application, IP, ATM, FibreChannel
▶ Specify traffic data rate and packet size characteristics
▶ Explicit: individual packets simulated
▶ Background: analytical model used to determine additional load/delay

Task

▶ Duplicate the SmallLAN scenario
▶ Add an IP traffic flow from PC1 to Laptop2 representing G711 voice
▶ Simulate and compare results

Application Traffic Models

Common Internet Applications

▶ Web Browsing, Email, Database, File Transfer, Printing, Remote Login, Video Conference, Voice, Custom
▶ Analytical model of user behaviour
  ▶ Web Browsing: HTTP, page request interarrival time, objects per page, object sizes

Task

▶ Duplicate the SmallLAN scenario
▶ Change WebBrowsing definition from Image Browsing to: mean interarrival 3s; page size 10000 Bytes; image size 10KB to 100KB
Editors and Features

Contents

OPNET Editors and Menus
Traffic
Nodes & Links
Channels
Statistics
Simulations

OPNET Editors and Menus
Traffic
Nodes and Links
Communication Channels
Statistics
Running Multiple Simulations

OPNET Workshop

Editors and Features
Editors
Traffic
Nodes & Links
Channels
Statistics
Simulations

Nodes
Example: ethernet_wkstn_adv

Red and blue lines: Packet streams—flow of data packets between processes

Links

▷ Specify link characteristics: packet formats, data rates, models of link behaviour (pipeline stages)
▷ Can specify background utilization (baseline load) for each direction; impacts on delay calculation for individual packets that traverse the link
▷ Examples: IP/PPP over dial-up, E1, E3, DSx, SONET-OCx; ADSL, ISDN; Ethernet; Frame Relay; EtherCoax; Frame Relay; DOCSIS; ATM over SONET; X.25

Node Structure
Physical Layer

▷ Transmitter: ethernet_v2 packet format; queue size of 1000 packets; data rate undefined
▷ Receiver: Packet must have 0 bit errors to be forwarded to MAC (ecc_threshold)
▷ Statistic wire between Tx and Rx: logical association for transceiver
▷ Statistic wires to MAC: tell the MAC status of transceiver (busy)
▷ Packet streams carry data packets and ICIs
  ▷ Packets are owned; treated like physical objects
  ▷ Interface Control Information (ICI): data structure for non-data communications
▷ Statistic wires carry single values, e.g. share state information between processes
Node Structure

Data Link Layer
- MAC: Ethernet MAC protocol
- MAC Parameters are promoted: values can be set in the Project editor
- ARP

Network Layer
- IP: includes IPv4, IPv6, ICMP, IGMP, MANET routing, Mobile IP, ...
- IP Encapsulation: manages multiplexing transport protocols over IP

Transport Layer
- TCP, UDP, RSVP
- Transport Protocol Adaptation Layer (tpal): provides applications generic interface to any transport protocol (TCP, UDP, ATM, X.25, Frame Relay)

Application Layer
- End-user applications (application): HTTP, FTP, Database, Custom, ...
- Management applications: DHCP, RIP
- CPU: Model performance of CPU, storage

Useful Menus
- Interfaces→Model Attributes: parameters for the node, not from a particular process
- Interfaces→Node Interfaces: parameters from individual processes are promoted to the project level
- Interfaces→Node Statistics: statistics from individual processes are promoted to the project level
**Pipeline Stages**

- Links are modelled by a set of pipeline stages
- Pipeline stage: C-based computations to model link behaviour
  - Transmitter object sends packets to pipeline
  - Packet processed in each stage
  - Result: packet received (or not) by Receiver object(s)
- Stages are different for point-to-point, bus and wireless links

**Point-to-Point Pipeline Stages**

- **Transmission delay** calculated from packet size and data rate
- **Propagation delay** calculated from link type and distance
- **Error allocation** estimates number of bit errors in packet
- **Error correction** accepts or rejects packet (e.g. depend on bit errors, error correction capability)

**Bus Pipeline Stages**

- **Closure** returns true for a receiver if it is capable to receive from the transmitter
- **Collision** determine if 2 packets overlap in time at receiver, record number of collisions for each packet

**Wireless Pipeline Stages**

- Executed once per transmission
- **Propagation delay** calculated from link type and distance
- **Error allocation** estimates number of bit errors in packet
- **Error correction** accepts or rejects packet (e.g. depend on bit errors, error correction capability)
IEEE 802.11 Pipeline Stages

**Receiver Group**

- **wlan_rxgroup**
  - Determines possible Tx/Rx pairs
  - Only at start of simulation

**Transmission Delay**

- **wlan_txdel**
  
  \[ \text{tx} \_ \text{delay} = \frac{\text{pkt} \_ \text{length}}{\text{data} \_ \text{rate}} \]

**Link Closure**

- **dra_closure_all**
  - Determines if packet can be transmitted between Tx/Rx pair
  - Null stage (always true)

**Rx Antenna Gain**

- **dra_rgain**
  - Default: 1 (isotropic)

**Received Power**

- **wlan_power**
  - Default: Free-space propagation
  
  \[ P_{\text{Rx}} = P_{\text{Tx}} \times G_{\text{Tx}} \times G_{\text{Rx}} \times \left( \frac{\lambda}{4\pi d} \right)^2 \]

**Interference Noise**

- **wlan_inoise**
  - Interfering packet: overlaps in time with packet \( p \)
  - Interference noise: sum of \( P_{\text{Rx}} \) for interfering packets

**Channel Match**

- **wlan_chanmatch**
  - **Ignore**—transmission outside of receiver bandwidth
  - **Noise**—Tx/Rx frequencies overlap
  - **Otherwise Valid**

**Tx Antenna Gain**

- **dra_tagain**
  - Calculate gain based on antenna properties
  - Default: 1 (isotropic)

**Propagation Delay**

- **wlan_propdel**
  
  \[ \text{prop} \_ \text{delay} = \frac{\text{distance}_{\text{tx}, \text{rx}}}{\text{speed}_{\text{light}}} \]

**Background Noise**

- **dra_bkgnoise**
  
  \[ bkg\_noise = \text{NoiseFigure} \times RxBW \times 290 \times \text{Boltzmann} + RxBW \times \text{Ambient} \]

\[ \text{Boltzmann} = 1.379 \times 10^{-23}; \text{Ambient} = 1.0 \times 10^{-26}; \]

**SNR**

- **dra_snr**
  
  \[ \text{SNR} = 10 \log_{10} \left( \frac{P_{\text{Rx}}}{\text{inoise} + bkg\_noise} \right) \]
IEEE 802.11 Pipeline Stages

Bit Error Rate

wlan_ber

Lookup modulation table to find BER for input Effective
SNR

- E.g. bpsk for 11a 6/9 Mb/s; qam64 for 11a 54Mb/s

Effective SNR = SNR + Processing Gain

Error Allocation

wlan_error

Calculate number of bits in error from BER and packet
length

Error Correction

wlan_ecc

If \( \frac{bit\_errors}{pkt\_length} > ecc\_threshold \) reject packet
Otherwise packet is received

Statistics

- All processes contain code to record performance
metrics, i.e. statistics
- Most process statistics are available for each node
- Global statistics are also available
- Two types of statistics:
  - Vector statistic versus time
  - Scalar single value across simulation
- Results are collected using probes

Probe Editor

- Probe can be created for any statistic
- Multiple probes on same statistic
- Collect both vector and scalar statistics over period of
time
  - All values
  - Samples: \( n \)-th or every \( T \) seconds
  - Bucket: collect \( n \) samples or for \( T \) seconds; record
    sum/time, min, max, mean, count
  - Scalar: last value, mean, min, max, variance, time
    average
Running Multiple Simulations

Varying Parameter Values
- Range of random seeds
- Range of global parameter values
- Node parameters must be promoted; range of values

Viewing Results
- Results for each simulation run versus time
- Parametric studies: results from multiple runs versus parameter

Example: Wireless LAN

Scenario: WLANSimple

Task: Configure Multiple Simulations

- Select all nodes and Edit Attributes
- Promote Interarrival Time to higher level
  - Apply to selected objects must be checked
- Configure simulation and select Object Attributes:
  - Add an attribute, select Wildcard for node name
  - Enter multiple values for interarrival time (exponential with mean 0.2, 0.1, 0.066, 0.05, 0.04)
- Set random seed to 128, 129, 130 and run

Task: View Parametric Results

- View results and select DES Parametric Studies tab
- Add to Y-axis: Global Statistics→Wireless LAN→Throughput→Sample Mean
- Add to X-axis: Scalar Statistics→Office Network→Interarrival Time
Custom Application for Sensor Network

Scenario

- Sensor network has sensors and 1 controller
- Sensors send updates to controller at regular frequency
- Controller may request a sensor to change update frequency

Application runs directly on IP over IEEE 802.11 wireless LAN (no transport)
Project, Nodes and Processes

Project
SensorNetwork: 3 scenarios

Nodes
wireless_sensor_adv: sensing node
wireless_controller_adv: controller node

Processes
sensor: application on sensing node
controller: application on controller node

State Transition Diagrams: States

Actions performed by code in enter executives and exit executives of states
Unforced states:
- Upon invocation, exit executives run in current state then enter executives in next state
- Blocked (waits for interrupt) between enter and exit executives
- Models true states
Forced states:
- No blocking: enter/exit executives run during invocation

State Transition Diagrams: Transitions

Transitions may contain: condition/executive
- If condition true, enter next state
- Run executive code if exists
Designer must ensure 1 and only 1 condition can be true at a time
Actions are combination of code in: exit executive, transition executive, enter executive
Variables

Variables can be scoped as:
- **Temporary** locally available in executives only
- **Function** locally available in functions only
- **State** global across executives and functions
- **Global** global across executives, functions and external code

Data types (other than C data types):
- Evhandle, Ici, Objid, Packet, Compcode, List, Stathandle, Distribution

Source Code

**Executives** state and transition executives
**Header Block** includes, defines and declarations for process model
**Function Block** functions used by process model
**Diagnostic Block** debugging code
**Termination Block** executed when simulation finishes
**External Code** headers/functions stored externally from process model

Contents

Example: Sensor Network

Process Modelling in OPNET

Controller Process Model

Simulation and Analysis
Controller Process Model

Four states:

- **INIT** performs initialization operations for process model
- **Idle** process resides here while waiting for interrupts
- **Receive** entered when the packet is received from a lower layer
- **Send** entered when a packet needs to be sent to the lower layer

Transitions:

- **PKT.RECEIVE** stream interrupt occurs due to packet arriving on input stream from `ip_encap`
- **PKT.SEND** self interrupt occurs due to timeout

Initialising the Process

**INIT state:**

1. Initialises state variables
2. Registers itself with IP
3. Schedules self interrupt to occur after a random time

```c
controller_sv_init()
```

After initialisation is complete, controller enters **Idle** state and waits for next event

**Task**

Add code to read the *Interarrival Time* attribute

**User-configurable Process Attributes**

- `Interfaces→Model Attributes`: attributes that can be set in the node model (or promoted to higher level)

Programmers Reference→Discrete Event Simulation→Internal Model Access Models→General Models→Model Support

`std/utilities/oms/oms_dist.support.ex.c`

Initialising the Process

**controller_register_self_with_ip()**

- `ip_encap` performs multiplexing: multiple higher layer protocols over IP
- Must register our higher layer protocol with IP

After initialisation is complete, controller enters **Idle** state and waits for next event
Events from Idle State

Receive a packet: stream interrupt from lower layer

```c
#define PKT_RECEIVE op_intrpt_type () == OPC_INTRPT_STRM
```

Send a packet: self interrupt (timeout)

```c
#define PKT_SEND op_intrpt_type () == OPC_INTRPT_SELF
```

Interrupts in general:

- Type: stream (packet), self (timeout), stat (status), ...
- Code: user-programmed, e.g. indicate which timer expired
- For stream interrupts, can determine which stream, e.g. from higher layer or lower layer
- Check the interrupt in exit executives of unforced states

Sending a Packet

Generate the packet:

1. Create random sized packet with random update frequency and destination
2. Send the packet (controller_packet_send_to_ip())
3. Update the statistics
4. Schedule self interrupt for next request for update frequency change

```c
controller_packet_send_to_ip()
```

1. ICI contains address of destination sensor
2. Packet is sent

Receiving a Packet

Process the packet:

1. Get the received packet from the stream
2. Get the ICI that accompanies the packet
3. Update the statistics
4. Destroy the packet and ICI

Return to Idle state

Sensor Process Model

Similar to controller process model: must regularly send packets, also send immediately after receiving packet

Task

Add code to retrieve the update period from the received packet

Programmers Reference→Discrete Event Simulation→Event Package

Programmers Reference→Discrete Event Simulation→Interface Control Information Package

Programmers Reference→Discrete Event Simulation→Packet Package
Contents

Example: Sensor Network

Process Modelling in OPNET

Controller Process Model

Simulation and Analysis

Simulation and Analysis

Tasks

1. Run the simulation
2. Show the existing result templates: DES → Panel Operations → Arrange Panels → Show All
3. Load the results into the templates: DES → Panel Operations → Panel Templates → Load with Latest Results

Results show the regular updates from sensors, and change in update interval triggered by controller

Statistics, Buckets and Sum/Time

- Buckets are used to reduce the statistic data stored during simulation, e.g. store 1 value for all packets sent in bucket duration of 10ms
- Sum/Time is often used for rate statistics, e.g. bits per second
- Last value in a bucket is proportionally split across current bucket and next bucket
- To avoid this with sum/time statistics, write value 0 after writing real statistic
- Default bucket size: sim duration / values per statistic