Overview

- Introduction
- Design
  - Process domain
  - Node domain
  - Network domain
  - Communication mechanism
- Simulation
  - Statistics
  - Probe
  - Analysis
- IEEE 802.11 MAC and PHY
OPNET Basics

- **Three-tiered OPNET hierarchy**
  - Network, Node and Process
  - Node model specifies objects in network domain
  - Process model specifies object in node domain
Object Hierarchy
OPNET Modeler -1

- **Project Editor**
  - Specify network topology and configure nodes and links. Choose results, run simulations and view results

- **Node Editor**
  - Create models of nodes by specifying internal structure and capabilities
  - Eg: - [ wireless station ]

- **Process Editor**
  - Develop models of decision-making processes representing protocols, algorithms, resource management, operating systems, etc.
  - Eg: - [ wireless_mac ]
OPNET Modeler -2

- **Link Model Editor**
  - Create, edit and view link models.
  - Eg:- [ 10Base_T_adv ]

- **Path Editor**
  - Create new path objects that define a traffic route

- **Packet Format Editor**
  - Specify packet format, defining the order, data type and size of fields contained within the packet
  - Eg:- [ 802.11packet ]

- **Antenna Pattern Editor**
  - Model the direction-dependent gain properties of antennas.
OPNET Modeler -3

- ICI Editor (radio only)
  - Define the internal structure of Interface Control Informations (ICIs)
- Modulation Curve Editor (radio only)
  - Create modulation functions to characterize the vulnerability of an information coding and modulation scheme to noise. (plots BER vs SNR)
- PDF Editor
- Probe Editor
  - Specify the statistics to be collected.
OPNET Modeler -4

- **Simulation Tool**
  - Define additional simulation constraints. Simulation sequences
- **Analysis Tool**
  - Create graphs, apply statistical data.
- **Filter Editor**
  - Create additional filters
OPNET Basics

- **Process Domain**
  - OPNET process models consist of
    - State transition diagrams / Finite State Machines
    - Blocks of C code
    - State/Temporary variables
  - A process is an instance of a process model
  - Processes can create additional child processes dynamically
  - Processes can respond to interrupts
  - Forced(Green) & Unforced States(Red) differ significantly in execution timing.
OPNET Basics

- Forced State (Green)
  - Invokes enter exec
  - Invokes exit exec
  - Evaluates all conditions
  - If exactly 1 condition evaluates to true, the transition is traversed to next state
OPNET Basics

- Unforced State (Red)
  - Invokes enter exec
  - 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
OPNET Process Model

- OPNET allows you to attach fragments of C/C++ code to each part of an FSM.
- This code augmented by OPNET-specific functions, is called **Proto-C**.

- The three primary places to use Proto-C are:
  - **Enter Executives**: Code executed when the module moves into a state.
  - **Exit Executives**: Code executed when the module leaves a state.
  - **Transition Executives**: Code executed in response to a given event.
Node Models

- Processor Modules
- Queue Modules
- Transmitter Modules
  - Point to point transmitter
  - Bus transmitter
  - Radio transmitter
- Receiver Modules
  - Point to point receiver
  - Bus receiver
  - Radio receiver
- Connections
  - Packet Stream
  - Statistic Wires
OPNET Basics

- **Node Domain**
  - Basic building blocks include processors, queues and transceivers/receivers.
    - Processors are fully programmable via their process model.
    - Queues also buffer and manage data packets.
    - Transceivers/Receivers are a node's outbound/inbound interfaces.
  - Interfaces between blocks.
    - **Packet streams**: carry data packets from a source to a destination module.
    - **Statistic wires**: carry a single data value from a source to a destination module.
Node Domain
Model Components

- Queue model requires
  - a means of generating packets (source module)
  - queuing packets (queue module)
  - servicing packets (queue module)
  - destroying packets (sink module)
Network Model

- Subnetworks
  - Fixed
  - Mobile
  - Satellite

- Communication Nodes
  - Fixed
  - Mobile
  - Satellite
OPNET Basics

- **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, etc.
    - LAN nodes, IP clouds, etc.
  - Links represent point-to-point and bus links
  - OPNET also has many vendor support network models from Cisco, 3Com, Lucent, HP, Xylan, etc.
Simulation Design

- Simulation output
- Output vectors
- Output scalars
- Animation
- Proprietary reports and files
- Web reporting
Simulation time advances when an event occurs.

- New event reaches head of event list, which causes Simulation Kernel to deliver an interrupt to the appropriate module.
- Process within the module gains control and processes interrupt.
- Simulation Kernel regains control from module.
- Simulation Kernel deletes event from event list, allowing new event to reach head of list.

**OPNET’s Event Methodology**
Probes

- Statistic
  - Processors
  - Queues and Subqueues
    - Queue size, interarrival times, sizes of packets
  - Generators
    - Output levels, interarrival times, sizes of packets
  - Transmitter channels
    - Throughput utilization, queue size
  - Receiver channels
    - Throughput, utilization, collisions, error rates, radio statistics.
  - Links
    - Throughput, utilization, error rates, collisions

- Attribute
- Animation
STEP 3
Statistics Collection
Wireless LAN

- Access Mechanism
  - DCF, PCF
- RTS/CTS
- Deference and Backoff
- Data Rate: 1-2-5.5-11 Mbps
- Recovery Mechanisms
  - ACK, Short and Long retry counters
- Fragmentation and Reassembly
Wireless LAN 2

- Duplicate packet detection
- Access Point Functionality
  - Each subnet is considered to be one BSS
- Buffer Size in MAC
- Radio IP Auto-addressing
- Physical Layer
  - Does not simulate the actual PHY but parameters
WLAN model applications

- Infrastructure BSS
- Independent BSS
- Wireless LAN Object Palette
  - Wireless Workstation (fixed and mobile)
  - Wireless Server (fixed and server)
  - Wireless terminal station (WLAN MAC without IP)
  - Wireless Router (access point)
WLAN Parameters

- RTS Threshold
- Fragmentation Threshold
- Data Rate
- PHY
  - PHY parameters needed by the MAC only
  - SIFS, DIFS, Min and Max CW
- Short and Long Retry Limit
- Access Point Functionality
  - Only one AP per BSS is allowed
- Channel Settings
- Buffer Size
- Max Receive Lifetime
- Simulation Attribute
- Wireless LAN range max 300m (1 microsecond air propagation time)
- WLAN statistics
WLAN Statistics

- Load
- Media Access Delay
- Throughput
- Backoff Slots
- Channel Reservation (NAV counter)
- Control Traffic Sent/Received (Ack, Rts, Cts)
- Data Traffic Sent/Received
- Retransmission attempts
- Dropped data packets
Wireless Node
Radio Transceiver Pipeline

- Stage 0: receiver group: invoked only once: rxgroup model
- Stage 1: transmission delay: invoked per tx: txdel model
- Stage 2: closure: invoked per tx: closure model
- Stage 3: channel match
- Stage 4: tx antenna gain
- Stage 5: propagation delay
- Stage 6: rx antenna gain
- Stage 7: received power
- Stage 8: background noise
- Stage 9: interference noise
- Stage 10: signal to noise ratio
- Stage 11: bit error rate
- Stage 12: error allocation
- Stage 13: error correction
- Stage 14: receivers
Simulation Time

- 1 Access Point, 9 Nodes:
  - InterArrvl: exp(1) / Pcktsize: exp(1024)
  - Events: Total (263552),
  - Average Speed (30567 events/sec.),
  - Time: Elapsed (9 sec.), Simulated (1 hr. 0 min. 0 sec.) $3600/9=400$
## Simulation Time

<table>
<thead>
<tr>
<th>#node</th>
<th>Traffic</th>
<th>Events</th>
<th>Average Speed events/sec</th>
<th>Elapsed time</th>
<th>Simulated Time</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1AP 16STA</td>
<td>Exp(10)</td>
<td>108101</td>
<td>29090</td>
<td>4s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 16STA</td>
<td>Exp(5)</td>
<td>231815</td>
<td>29602</td>
<td>8s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 16STA</td>
<td>Exp(1)</td>
<td>964168</td>
<td>29187</td>
<td>33s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 16STA</td>
<td>Exp(0.5)</td>
<td>1920307</td>
<td>29940</td>
<td>1m4s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 16STA</td>
<td>Exp(0.1)</td>
<td>9614683</td>
<td>29794</td>
<td>5m22s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 16STA</td>
<td>Exp(0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Simulation Time

<table>
<thead>
<tr>
<th>#node</th>
<th>Traffic</th>
<th>Events</th>
<th>Average Speed events/sec</th>
<th>Elapsed time</th>
<th>Simulated Time</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1AP 32 STA</td>
<td>Exp(10)</td>
<td>273995</td>
<td>36047</td>
<td>8s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 32 STA</td>
<td>Exp(5)</td>
<td>506717</td>
<td>36718</td>
<td>14s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 32 STA</td>
<td>Exp(1)</td>
<td>2524616</td>
<td>37386</td>
<td>1m7s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 32 STA</td>
<td>Exp(0.5)</td>
<td>5105918</td>
<td>37575</td>
<td>2m15s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 32 STA</td>
<td>Exp(0.1)</td>
<td>22981552</td>
<td>38585</td>
<td>10m28s</td>
<td>1h</td>
<td></td>
</tr>
<tr>
<td>1AP 32 STA</td>
<td>Exp(0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
100 STA
Packet size 1000 byte
Interarrival time 0.005
Load 1.6 Mbps
Retry counts 255

Saturation throughput
No RTS CTS

Number of nodes
Conclusion

1. Define packet format
2. Define link model
3. Create peripheral node model
4. Create hub node model
5. Build network model
Kernel Procedures (KP)s

- Op_ev_cancel(evhandle)
- Op_intrpt_code()
- Op_intrpt_stat()
- Op_intrpt_schedule_self(time,code)
- Op_intrpt_strm()
- Op_intrpt_type()
- Op_pk_copy(pkptr)
- Op_pk_create_fmt(format_name)
- Op_pk_destroy(pkptr)
- Op_pk_get(instrm_index)
- Op_pk_send(pkptr,outstrm_index)
- Op_pk_send_forced(pkptr,outstrm_index)
- Op_pk_nfd_set(pkptr,fd_name,value)
- Op_pk_nfd_get(pkptr,fd_name,value_ptr)
- Op_sim_time()
- Op_stat_reg(name,index,type)
- Op_stat_write(stathandle,value)
- Op_stat_local_read(instat_index)
- Op_subq_empty(subg_index)
- Op_subq_pk_access(subq_index,pos_index)
- Op_subq_pk_insert(subq_index,pkptr,pos_index)
- Op_subq_pk_remove(subq_index,pos_index)
- Op_subq_stat(subq_index,stat_index)
- Op_pk_stamp(pkptr)
- Op_pk_stamp_time_get(pkptr)
Physical Layer

- Radio receiver
  - Transmitting station does not receive its transmitted packets.
  - wlan_rxgroup.ps.c
- Channel match
  - Data rate should match in both parties
  - Wlan_propdel.ps.c
- Propagation delay
  - Wlan_propdel.ps.c
- Error Correction
- Corrupted if has errors more than the error threshold.
  - Wlan_ecc.ps.c
Encapsulation/De-capsulation

- Lower_layer_pkptr=op_pk_create_fmt(“my format”)
- Op_pk_nfd_set(lower_layer_pkptr,”data_field”, upper_layer_pkptr);
- Op_pk_nfd_get(lower_layer_pkptr,”data_field”,&upper_layer_pkptr);
dra_xxx

- Receiver Group: dra_rxgroup
  - Invoked once
  - Dra_rxgroup(objid tx, objid rx)
- Transmission Delay Model: dra_txdel
  - Invoked immediately upon the start of a packet
  - Pklen/tx_drate
Dra_xxx -2

- **Closure Model: dra_closure**
  - Invoked immediately after the tx_delay
  - Check connectivity one more time.
  - Line of sight algorithm
- **Channel match: dra_chanmatch**
  - Invoked after clouser
  - Check compatibility of tx and rx.
  - Identify pkts: Valid, interference, ignored