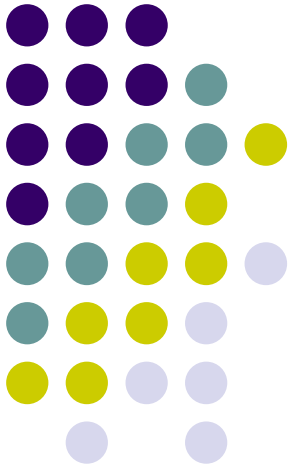
OPNET

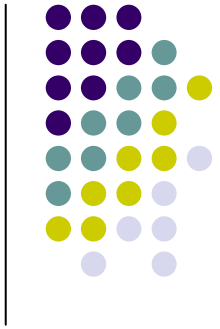
Mustafa Ergen

ergen@eecs.berkeley.edu

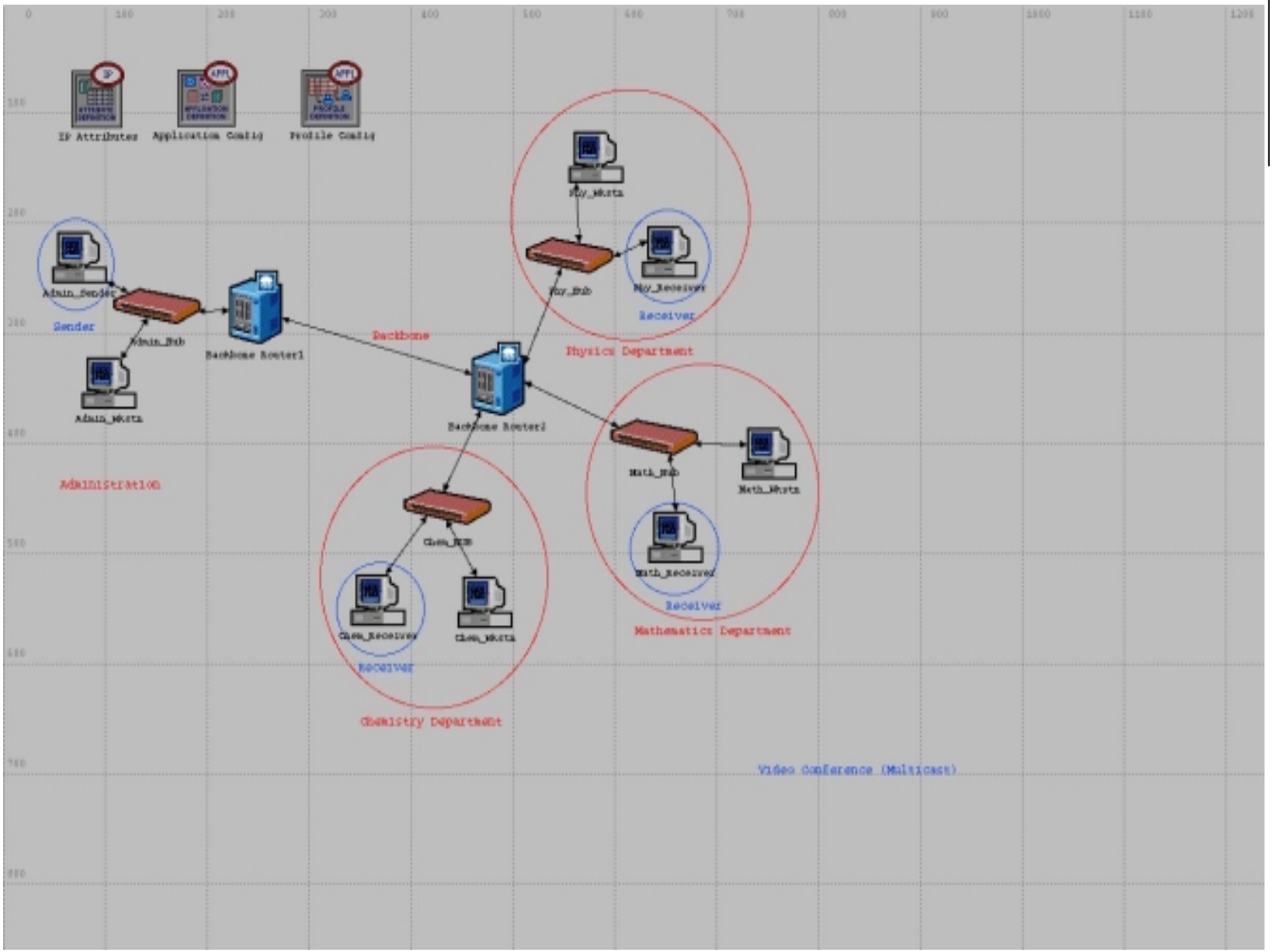
UC Berkeley



Overview

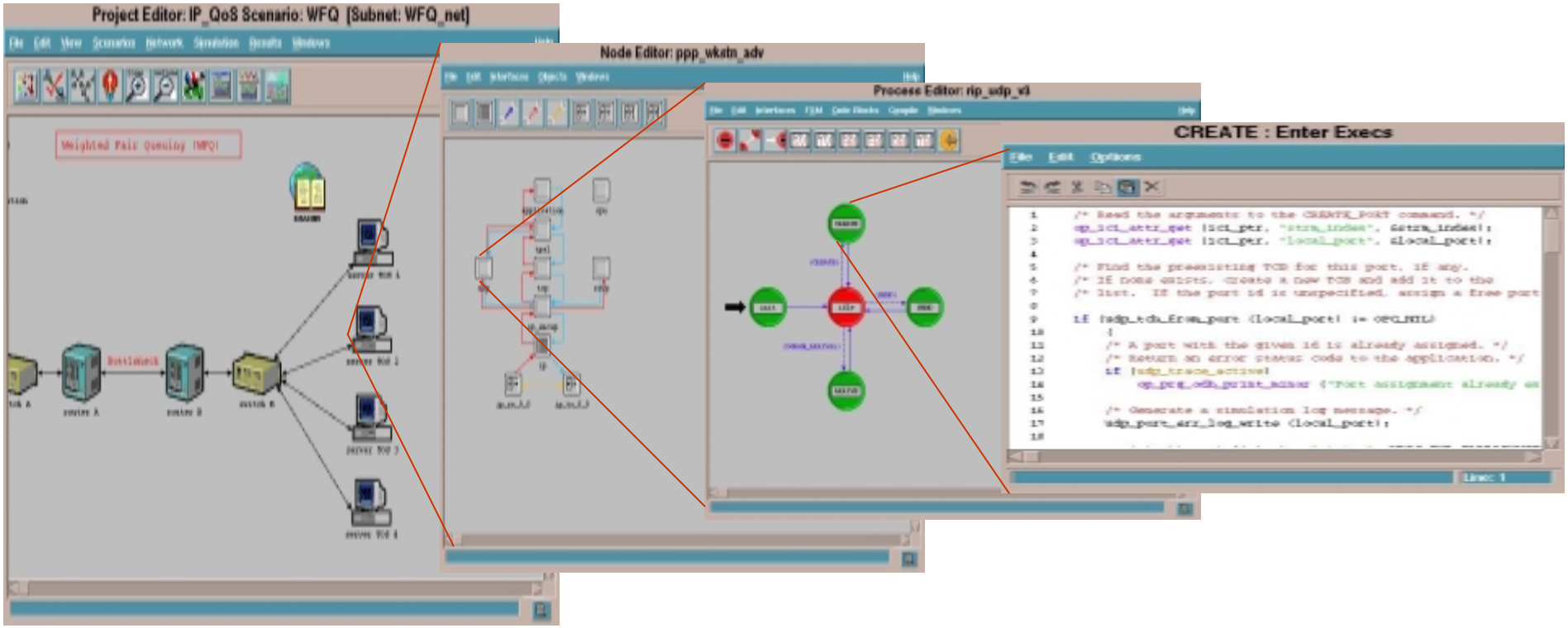


- Introduction
- Design
  - Process domain
  - Node domain
  - Network domain
  - Communication mechanism
- Simulation
  - Statistics
  - Probe
  - Analysis
- IEEE 802.11 MAC and PHY

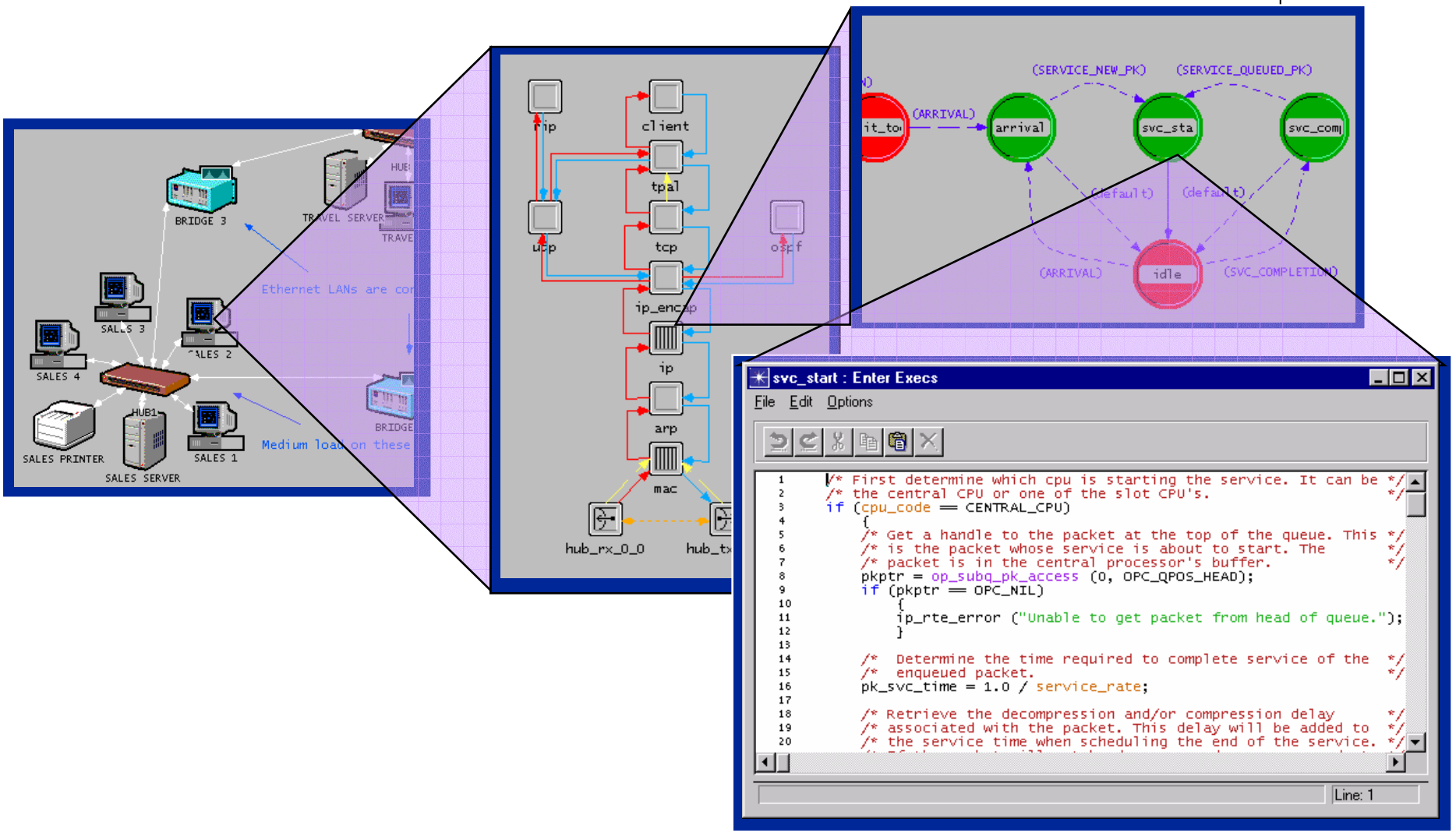

IP Attributes
Application Config
Profile Config
Admin_Sender
Sender
Admin_Hub
Admin_Hosts
Backbone Router1
Backbone
Backbone Router2
Administration
Phy_Hosts
Phy_Hub
Phy_Receiver
Receiver
Physics Department
Math_Hub
Math_Hosts
Math_Receiver
Receiver
Mathematics Department
Chem_Hub
Chem_Receiver
Chem_Hosts
Receiver
Chemistry Department
Video Conference (Multicast)

OPNET Basics

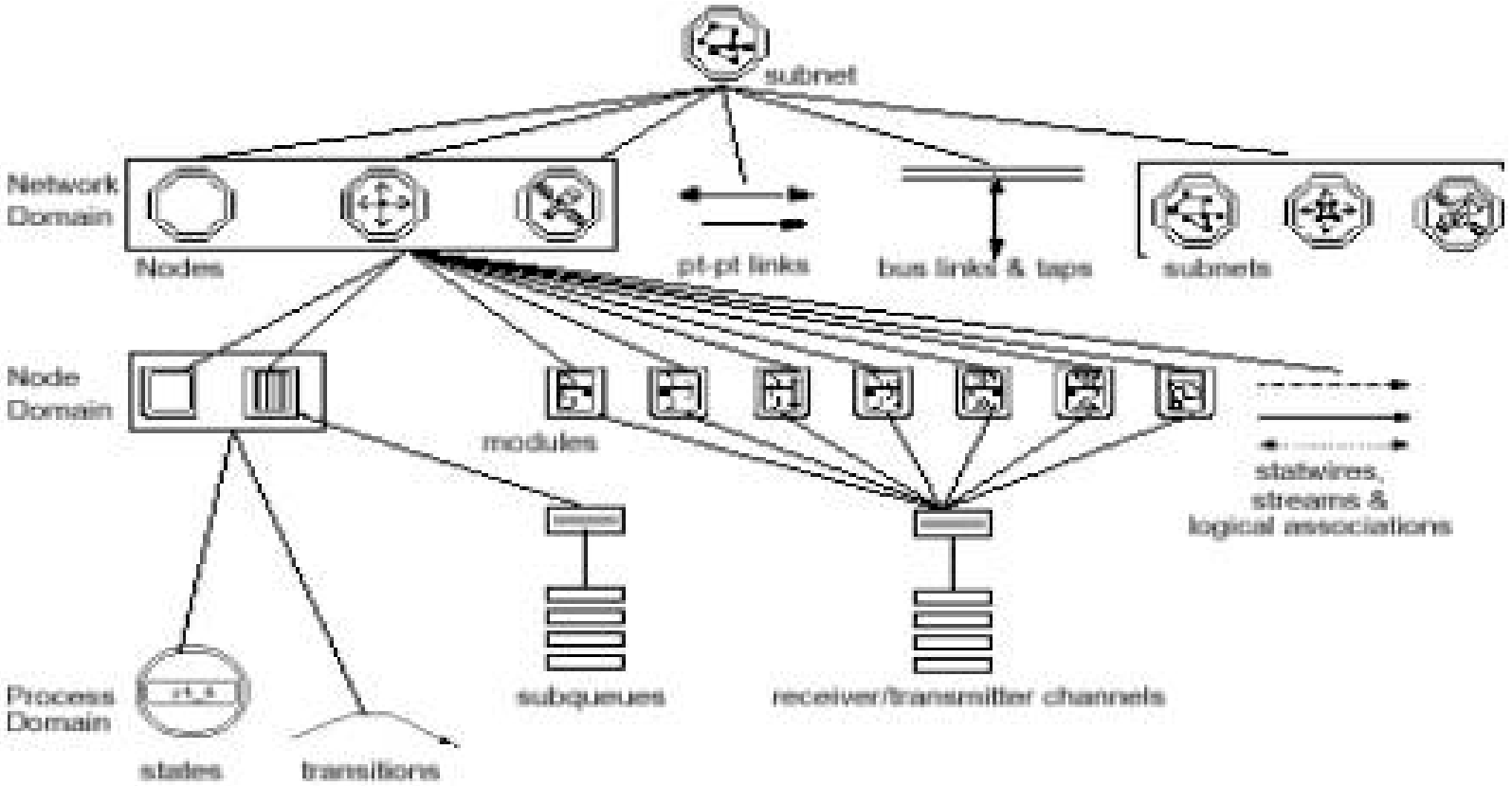
- **Three-tiered OPNET hierarchy**
  - Network, Node and Process
    - Node model specifies objects in network domain
    - Process model specifies object in node domain



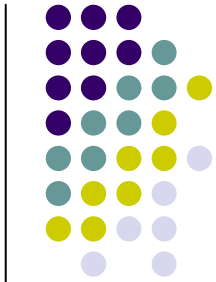
OPNET


BRIDGE 3
TRAVEL SERVER
Ethernet LANs are co
SALES 4
SALES 3
SALES 2
HUB1
SALES PRINTER
SALES SERVER
SALES 1
Medium load on these
BRIDGE
rip
client
tpal
udp
tcp
ospf
ip_encap
ip
arp
mac
hub_rx_0_0
(SERVICE_NEW_PK)
(SERVICE_QUEUED_PK)
(ARRIVAL)
arrival
svc_sta
svc_com
(default)
(default)
(ARRIVAL)
idle
(SVC_COMPLETION)
svc_start : Enter Execs
File Edit Options
/* First determine which cpu is starting the service. It can be */
/* the central CPU or one of the slot CPU's. */
if (cpu_code == CENTRAL_CPU)
{
/* Get a handle to the packet at the top of the queue. This */
/* is the packet whose service is about to start. The */
/* packet is in the central processor's buffer. */
pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);
if (pkptr == OPC_NIL)
{
ip_rte_error ("Unable to get packet from head of queue.");
}
/* Determine the time required to complete service of the */
/* enqueued packet. */
pk_svc_time = 1.0 / service_rate;
/* Retrieve the decompression and/or compression delay */
/* associated with the packet. This delay will be added to */
/* the service time when scheduling the end of the service. */
Line: 1

Object Hierarchy

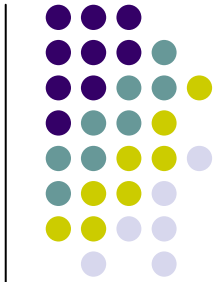


OPNET Modeler -1



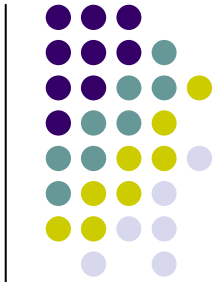
- Project Editor
  - Specify network topology and configure nodes and links. Choose results, run simulations and view results
- Node Editor
  - Create models of nodes by specifying internal structure and capabilities
  - Eg: - [wireless station]
- Process Editor
  - Develop models of decision-making processes representing protocols, algorithms, resource management, operating systems, etc.
  - Eg: - [wireless_mac]

OPNET Modeler -2



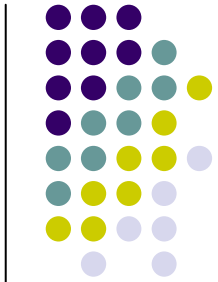
- Link Model Editor
  - Create, edit and view link models.
  - Eg:- [10Base_T_adv]
- Path Editor
  - Create new path objects that define a traffic route
- Packet Format Editor
  - Specify packet format, defining the order, data type and size of fields contained within the packet
  - Eg:- [802.11packet]
- Antenna Pattern Editor
  - Model the direction-dependent gain properties of antennas.

OPNET Modeler -3



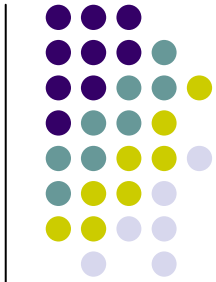
- ICI Editor(radio only)
  - Define the internal structure of Interface Control Informations (ICIs)
- Modulation Curve Editor (radio only)
  - Create modulation functions to characterize the vulnerability of an information coding and modulation scheme to noise. (plots BER vs SNR)
- PDF Editor
- Probe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

OPNET Modeler -4

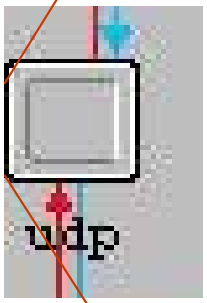


- Simulation Tool
  - Define additional simulation constraints. Simulation sequences
- Analysis Tool
  - Create graphs, apply statistical data.
- Filter Editor
  - Create additional filters

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.



Process Editor: rip_udp_v3

File Edit Interfaces FSM Code Blocks Compile Windows Help

SV TV HB FB DB TB

CREATE

(CREATE)

init

idle

(SEND)

SEND

(DGRAM_ARRIVAL)

ARRIVE

CREATE : Enter Execs

File Edit Options

```
/* Read the arguments to the CREATE_PORT command. */
op_ici_attr_get (ici_ptr, "strm_index", &strm_index);
op_ici_attr_get (ici_ptr, "local_port", &local_port);

/* Find the preexisting TCB for this port, if any.  */
/* If none exists, create a new TCB and add it to the */
/* list.  If the port id is unspecified, assign a free port */

if (udp_tcb_from_port (local_port) != OPC_NIL)
    {
    /* A port with the given id is already assigned. */
    /* Return an error status code to the application. */
    if (udp_trace_active)
        op_prg_odb_print_minor ("Port assignment already ex
```

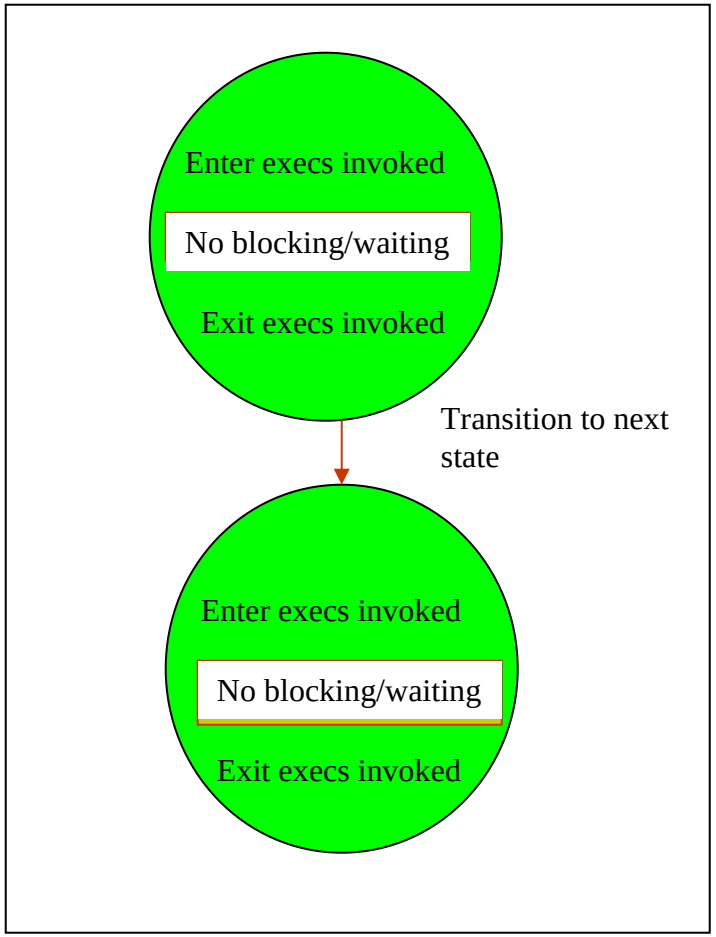
OPNET Basics

- Forced State (Green)
  - Invokes enter exec
  - Invokes exit exec
  - Evaluates all conditions
  - If exactly 1 condition

  evaluates to true,

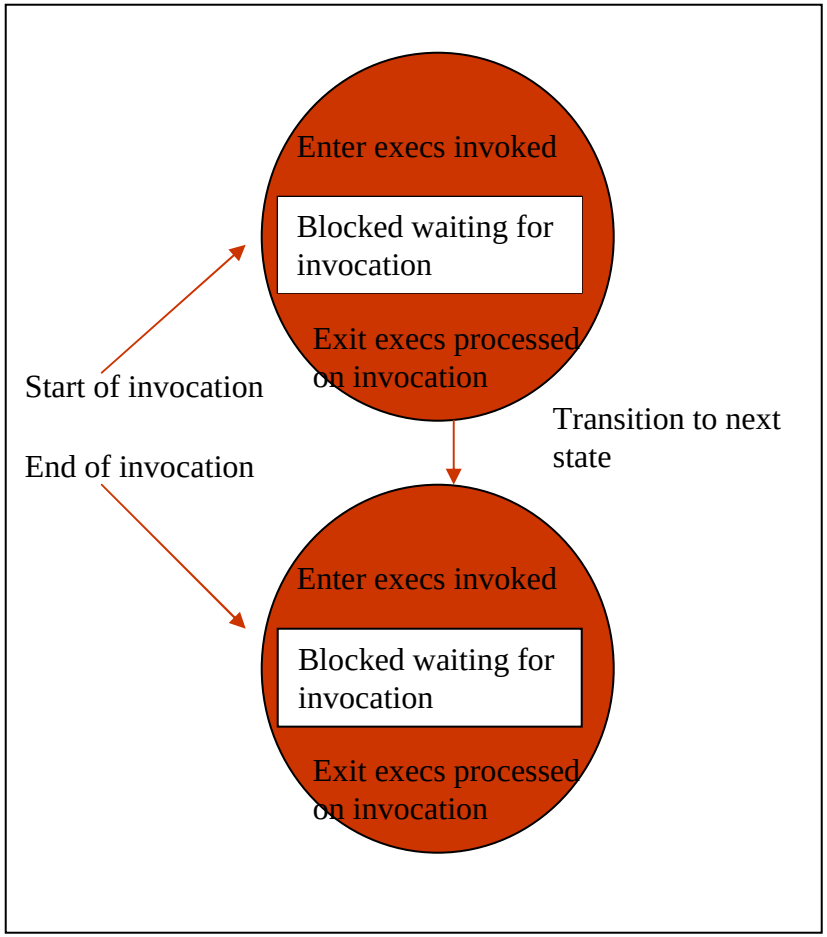
  the transition is

  traversed to next state



OPNET Basics

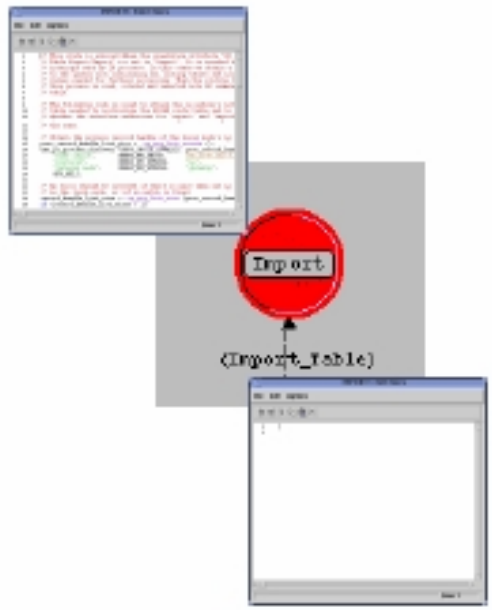
- Unforced State (Red)
  - Invokes enter exec
  - Places a marker at the middle of the state
  - Releases control to the simulation kernel and becomes idle
  - Resumes at the marker and processes the exit execs when next invoked



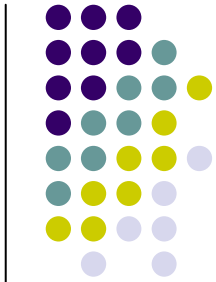
OPNET Process Model

- 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.

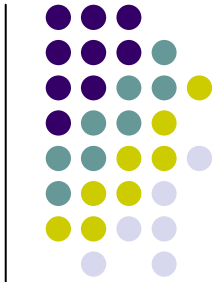


Node Models



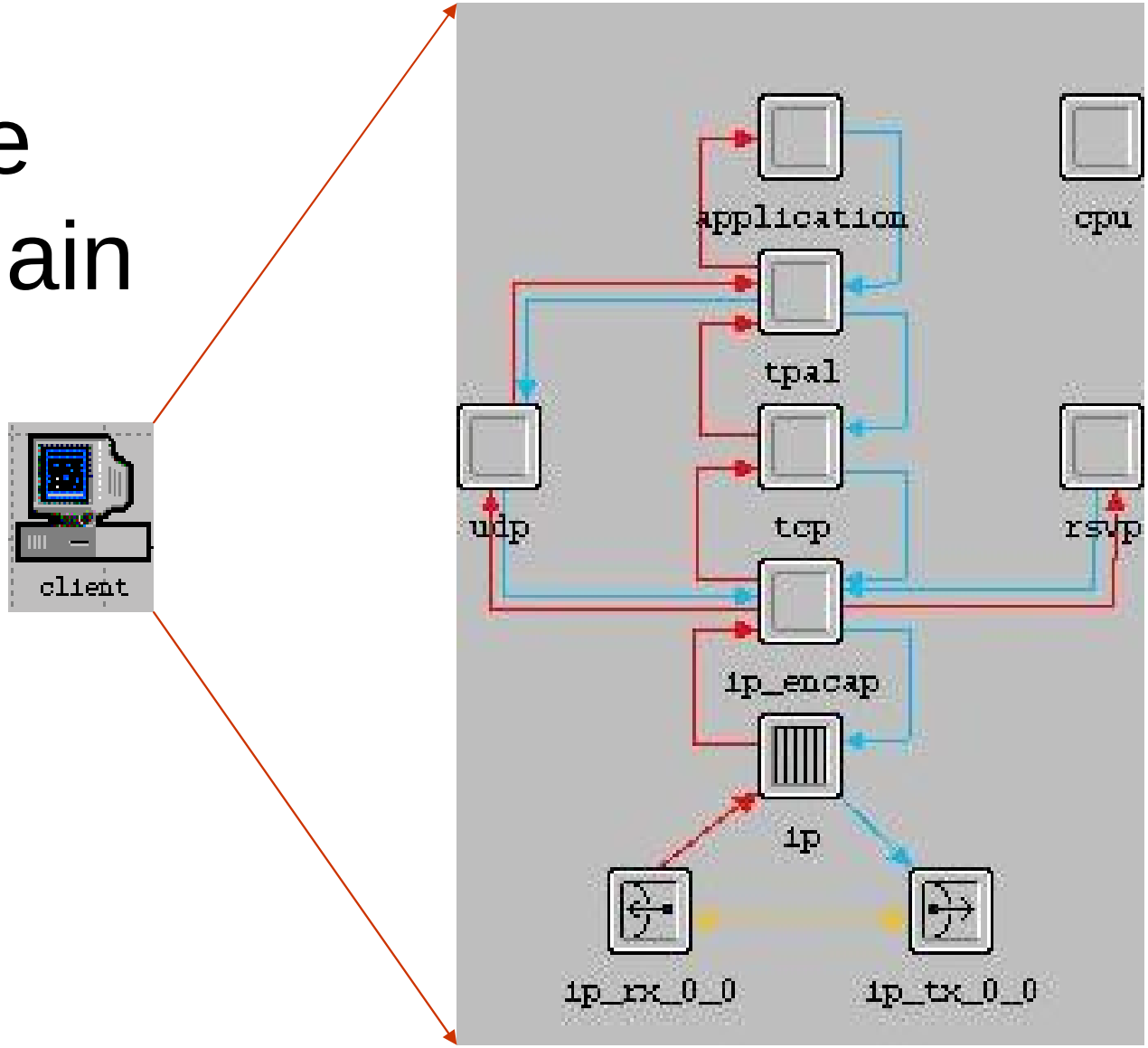
- Processor Modules
- Queue Modules
- Transmitter Modules
  - Point to point transmitter
  - Bus transmitter
  - Radio transmitter
- Receiver Modules
  - Point to point receiver
  - Bus receiver
  - Radio receiver
- Connections
  - Packet Stream
  - Statistic Wires

OPNET Basics



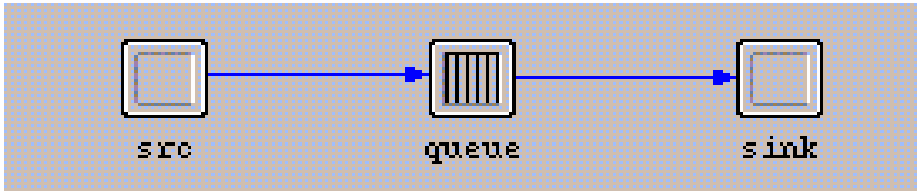
- **Node Domain**
  - Basic building blocks include processors, queues and transceivers/receivers.
    - Processors are fully programmable via their process model
    - Queues also buffer and manage data packets
    - Transceivers/Receivers are a node's outbound/inbound interfaces
  - Interfaces between 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


client
application
cpu
tpal
udp
tcp
rsvp
ip_encap
ip
ip_rx_0_0
ip_tx_0_0

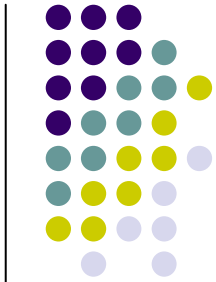
Model Components

- **Queue model requires**



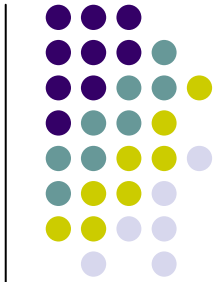
  - a means of generating packets (source module)
  - queuing packets (queue module)
  - servicing packets (queue module)
  - destroying packets (sink module)

Network Model



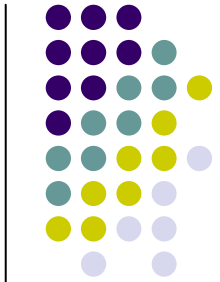
- Subnetworks
  - Fixed
  - Mobile
  - Satellite
- Communication Nodes
  - Fixed
  - Mobile
  - Satellite

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.

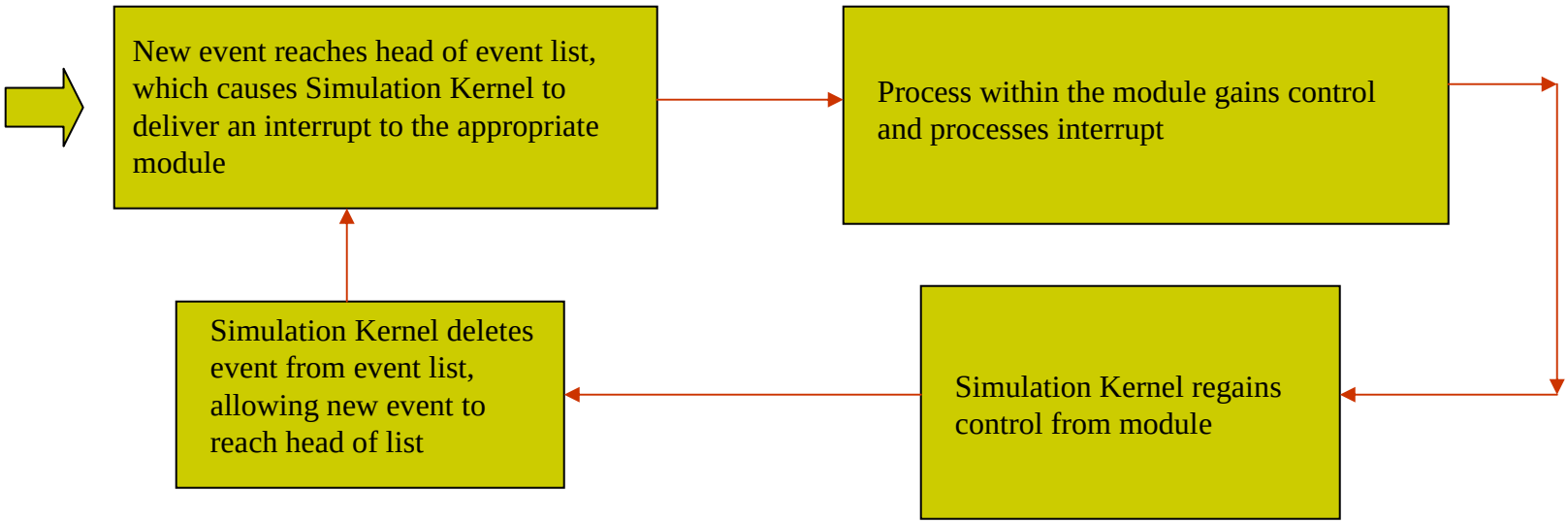
Simulation Design



- Simulation output
- Output vectors
- Output scalars
- Animation
- Proprietary reports and files
- Web reporting

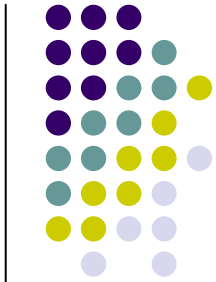
Event Driven Simulation

- Simulation time advances when an event occurs.



OPNET's Event Methodology

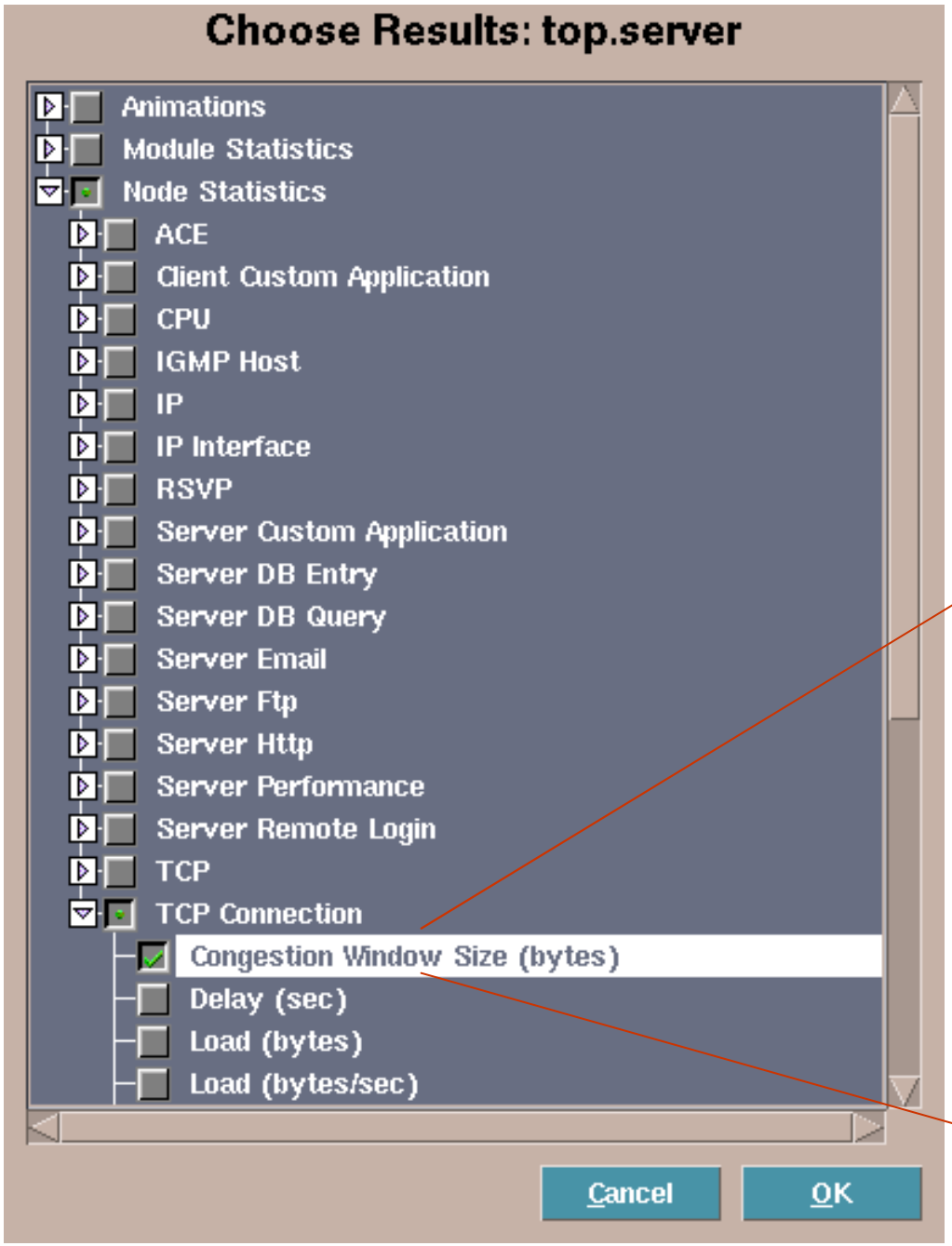
Probes



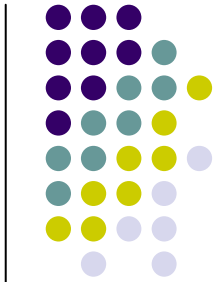
- Statistic
  - Processors
  - Queues and Subqueues
    - Queue size, interarrival times, sizes of packets
  - Generators
    - Output levels, interarrival times, sizes of packets
  - Transmitter channels
    - Throughput utilization, queue size
  - Receiver channels
    - Throughput, utilization, collisions,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

STEP 3

Statistics Collection

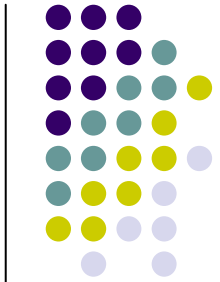


Wireless LAN



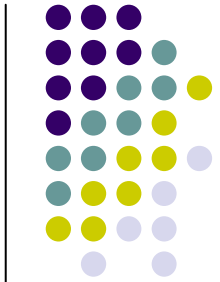
- Access Mechanism
  - DCF, PCF
- RTS/CTS
- Deference and Backoff
- Data Rate: 1-2-5.5-11 Mbps
- Recovery Mechanisms
  - ACK, Short and Long retry counters
- Fragmentation and Reassembly

Wireless LAN 2



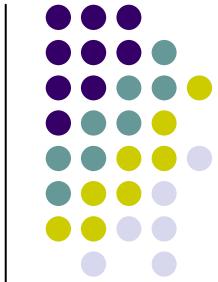
- Duplicate packet detection
- Access Point Functionality
  - Each subnet is considered to be one BSS
- Buffer Size in MAC
- Radio IP Auto-addressing
- Physical Layer
  - Does not simulate the actual PHY but parameters

WLAN model applications



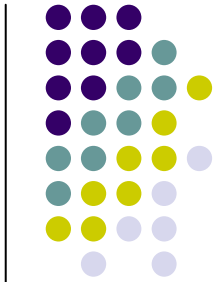
- Infrastructure BSS
- Independent BSS
- Wireless LAN Object Palette
  - Wireless Workstation (fixed and mobile)
  - Wireless Server (fixed and server)
  - Wireless terminal station (WLAN MAC without IP)
  - Wireless Router (access point)

WLAN Parameters



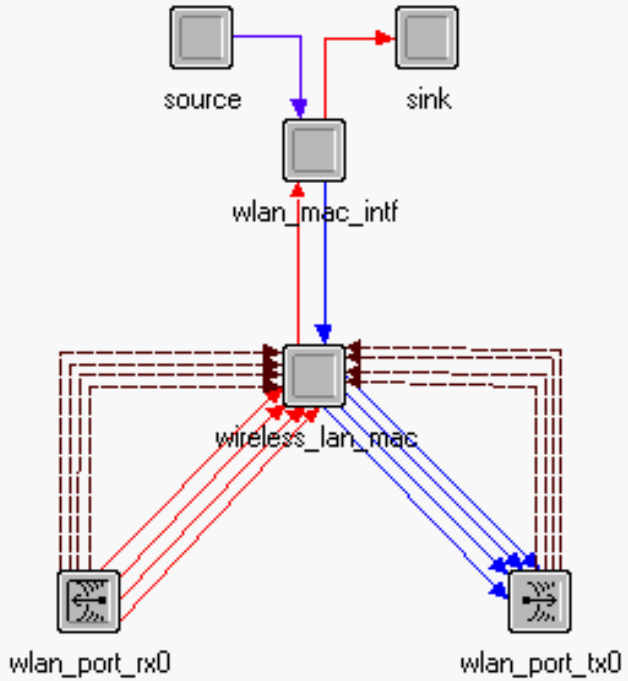
- RTS Threshold
- Fragmentation Threshold
- Data Rate
- PHY
  - PHY parameters needed by the MAC only
  - SIFS, DIFS, Min and Max CW
- Short and Long Retry Limit
- Access Point Functionality
  - Only one AP per BSS is allowed
- Channel Settings
- Buffer Size
- Max Receive Lifetime
- Simulation Attribute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

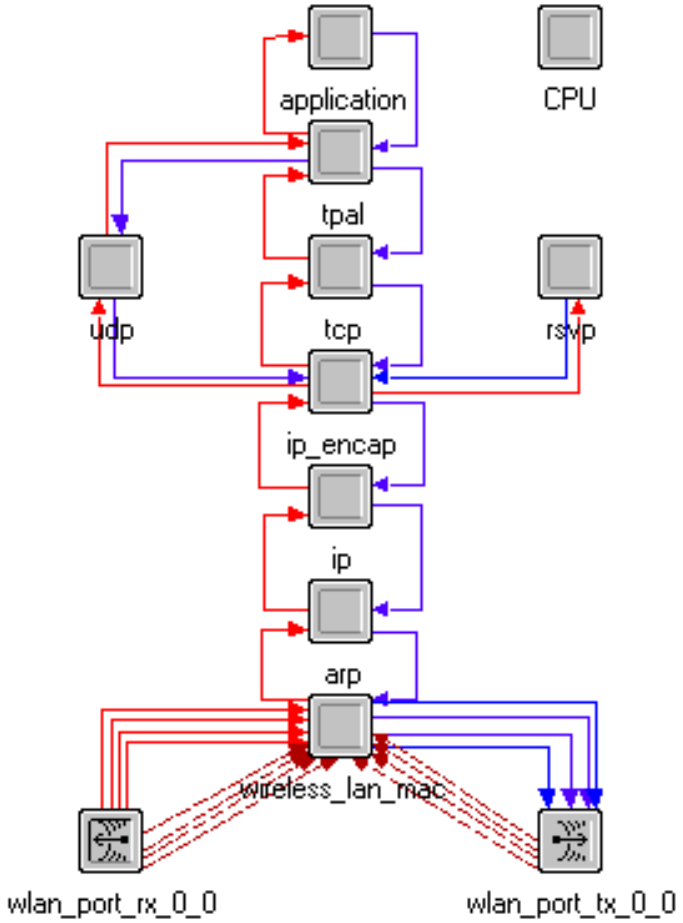
WLAN Statistics



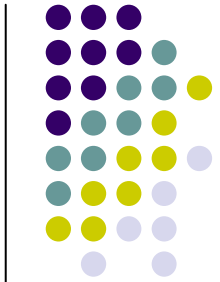
- Load
- Media Access Delay
- Throughput
- Backoff Slots
- Channel Reservation (NAV counter)
- Control Traffic Sent/Received (Ack, Rts, Cts)
- Data Traffic Sent/Received
- Retransmission attempts
- Dropped data packets

Wireless Node





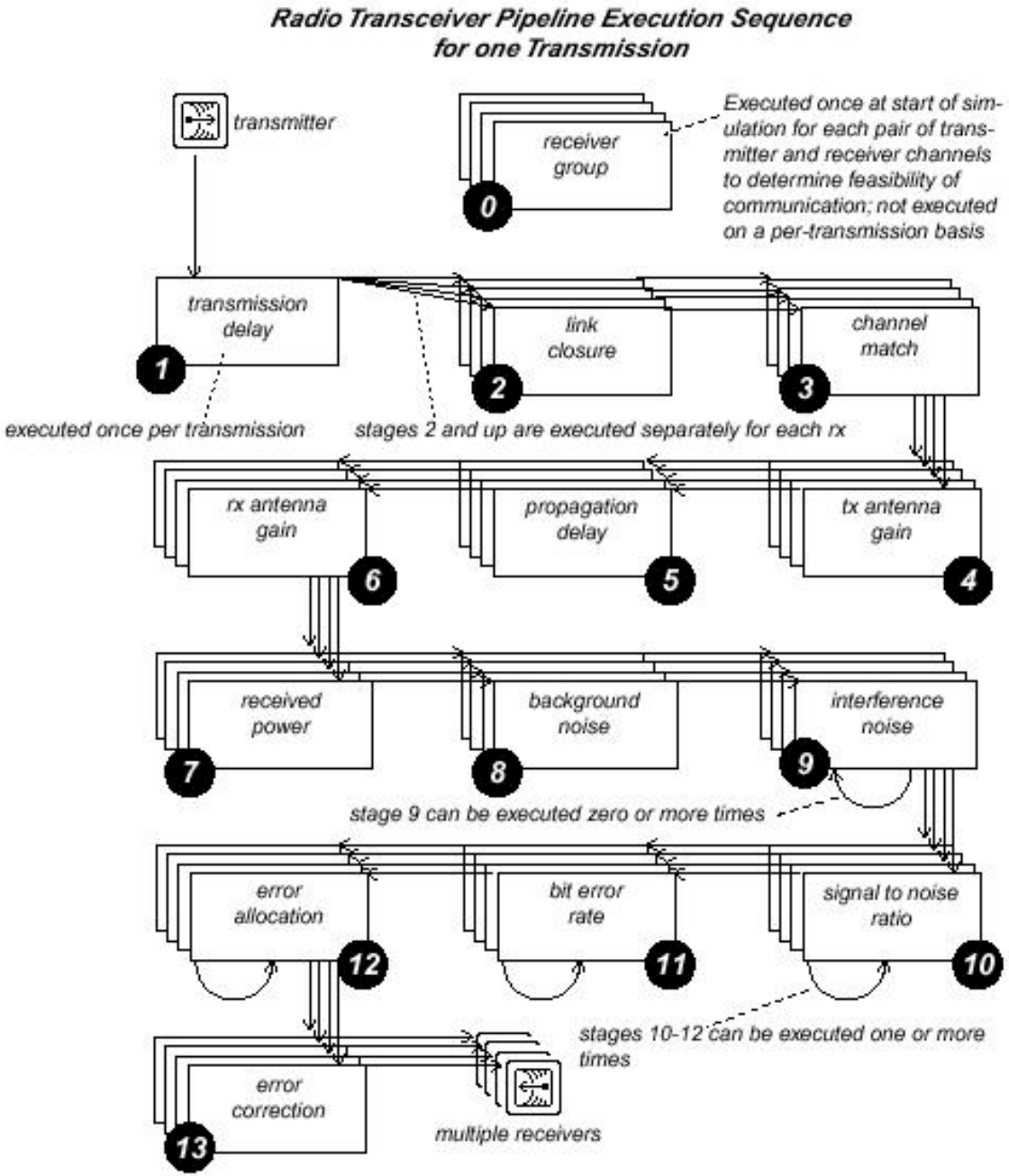
Radio Transceiver Pipeline

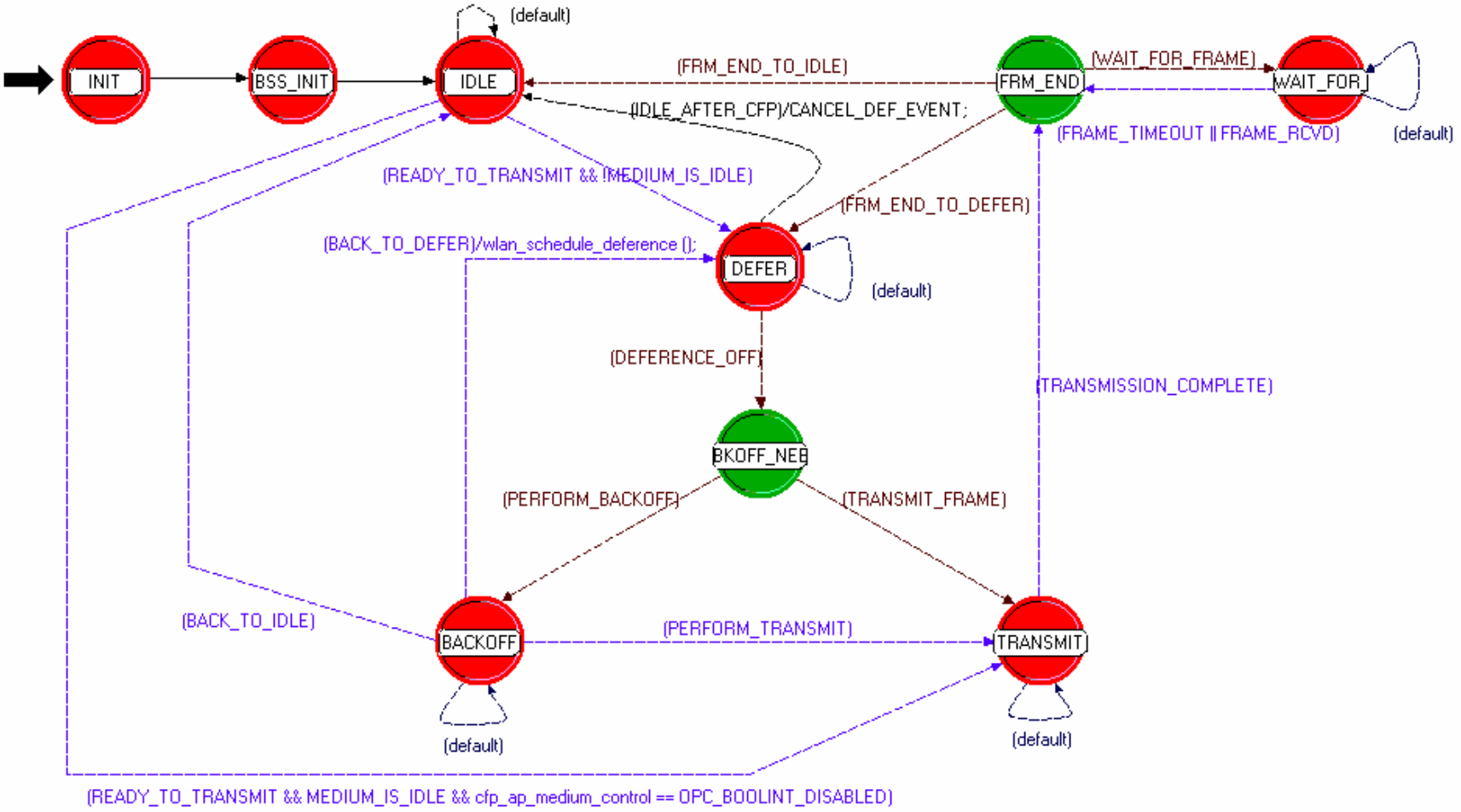


- Stage 0: receiver group: invoked only once : rxgroup model
- Stage 1:transmission delay: invked per tx: txdel model
- Stage 2:closure: invked per tx: closure model
- Stage 3: channel match
- Stage 4: tx antenna gain
- Stage 5: propagation delay
- Stage 6: rx antenna gain
- Stage 7: received power
- 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

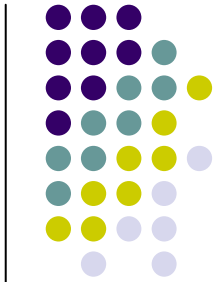
Radio Transmitter

Radio Receiver


Radio Transceiver Pipeline Execution Sequence
for one Transmission
transmitter
receiver group
0
Executed once at start of simulation for each pair of transmitter and receiver channels to determine feasibility of communication; not executed on a per-transmission basis
transmission delay
1
executed once per transmission
link closure
2
channel match
3
stages 2 and up are executed separately for each rx
rx antenna gain
6
propagation delay
5
tx antenna gain
4
received power
7
background noise
8
interference noise
9
stage 9 can be executed zero or more times
error allocation
12
bit error rate
11
signal to noise ratio
10
stages 10-12 can be executed one or more times
error correction
13
multiple receivers

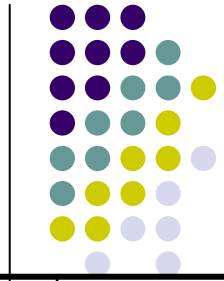

INIT
BSS_INIT
IDLE
(default)
(FRM_END_TO_IDLE)
(IDLE_AFTER_CFP)/CANCEL_DEF_EVENT;
FRM_END
(WAIT_FOR_FRAME)
WAIT_FOR
(FRAME_TIMEOUT || FRAME_RCVD)
(default)
(READY_TO_TRANSMIT && !MEDIUM_IS_IDLE)
(FRM_END_TO_DEFER)
(BACK_TO_DEFER)/wlan_schedule_deference ();
DEFER
(default)
(DEFERENCE_OFF)
(TRANSMISSION_COMPLETE)
BKOFF_NEB
(PERFORM_BACKOFF)
(TRANSMIT_FRAME)
(BACK_TO_IDLE)
BACKOFF
(PERFORM_TRANSMIT)
TRANSMIT
(default)
(default)
(READY_TO_TRANSMIT && MEDIUM_IS_IDLE && cfp_ap_medium_control == OPC_BOOLINT_DISABLED)

Simulation Time



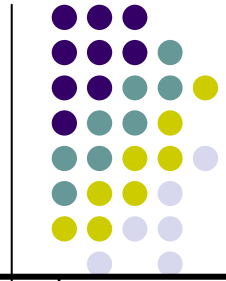
- 1 Access Point, 9 Nodes:
  - InterArrvl: exp(1)/ Pcktsize: exp(1024)
  - Events: Total (263552),
  - Average Speed (30567 events/sec.)
  - Time: Elapsed (9 sec.), Simulated (1 hr. 0 min. 0 sec.) 3600/9=400

Simulation Time

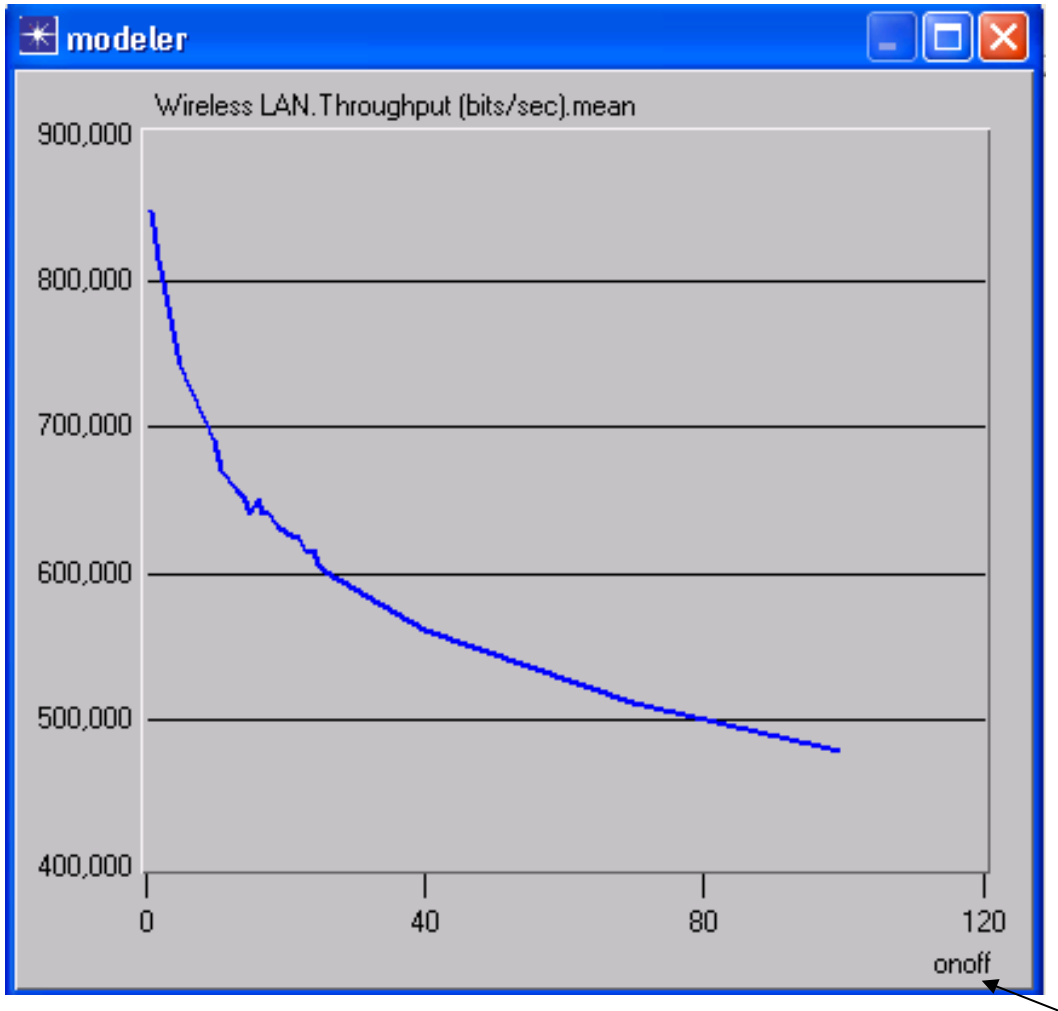


#node	Traffic	Events	Average Speed events/sec	Elapsed time	Simulated Time	Ratio
1AP 16STA	Exp(10)	108101	29090	4s	1h	
1AP 16STA	Exp(5)	231815	29602	8s	1h	
1AP 16STA	Exp(1)	964168	29187	33s	1h	
1AP 16STA	Exp(0.5)	1920307	29940	1m4s	1h	
1AP 16STA	Exp(0.1)	9614683	29794	5m22s	1h	
1AP 16STA	Exp(0.05)					

Simulation Time



#node	Traffic	Events	Average Speed events/sec	Elapsed time	Simulated Time	Ratio
1AP 32 STA	Exp(10)	273995	36047	8s	1h	
1AP 32 STA	Exp(5)	506717	36718	14s	1h	
1AP 32 STA	Exp(1)	2524616	37386	1m7s	1h	
1AP 32 STA	Exp(0.5)	5105918	37575	2m15s	1h	
1AP 32 STA	Exp(0.1)	22981552	38585	10m28s	1h	
1AP 32 STA	Exp(0.05)					



100 STA

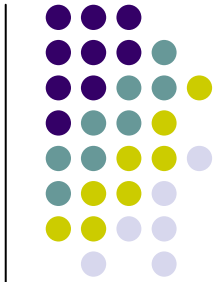
Packet size 1000byte
Interarrival time 0.005
Load 1.6Mbps
Retry counts 255

Saturation throughput

No RTS CTS

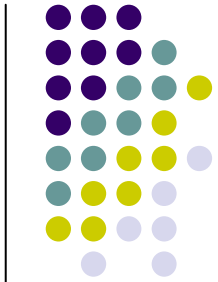
Number of nodes

Conclusion



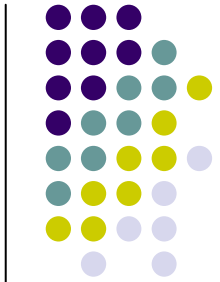
1. Define packet format
2. Define link model
3. Create peripheral node model
4. Create hub node model
5. Build network model

Kernel Procedures (KP)s



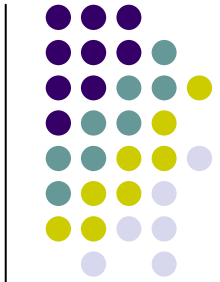
- Op_ev_cancel(evhandle)
- Op_intrpt_code()
- Op_intrpt_stat()
- Op_intrpt_schedule_self(time,code)
- Op_intrpt_strm()
- Op_intrpt_type()
- Op_pk_copy(pkptr)
- Op_pk_create_fmt(format_name)
- Op_pk_destroy(pkptr)
- Op_pk_get(instrm_index)
- Op_pk_send(pkptr,outstrm_index)
- Op_pk_send_forced(pkptr,outstrm_index)
- 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



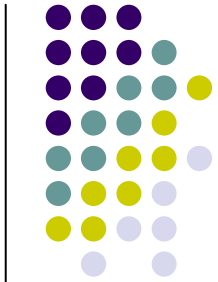
- Radio receiver
  - Transmitting station does not receive its transmitted packets.
  - wlan_rxgroup.ps.c
- Channel match
  - Data rate should match in both parties
  - Wlan_propdel.ps.c
- Propagation delay
  - Wlan_propdel.ps.c
- Error Correction
- Corrupted if has errors more than the error threshold.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

Encapsulation/De-capsulation



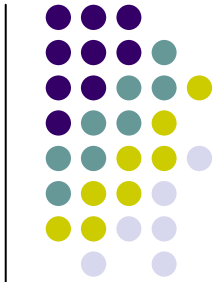
- Lower_layer_pkptr=op_pk_create_fmt(“my format”)
- Op_pk_nfd_set(lower_layer_pkptr,”data_field”, upper_layer_pkptr);
- Op_pk_nfd_get(lower_layer_pkptr,”data_field”,&upper_layer_pkptr);

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