Network Research Tools

Ljiljana Trajković
ljilja@cs.sfu.ca

Communication Networks Laboratory
http://www.ensc.sfu.ca/cnl
School of Engineering Science
Simon Fraser University, Vancouver, Canada

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

BCNET 2006

Network Research Tools

This session will seek to acquaint participants with different tools that are used to conduct network research and explore researchers' experience with them. Two researchers – one from UBC and SFU – will present current tools they are using and experimenting with. At UBC, EmuLab is an experimental network environment that allows researchers access to simulated, emulated and wide-area network testbeds. This session will seek to build awareness and interest for EmuLab in the research community, exploring what EmuLab is and how it can be used. At SFU, research network simulation tools are being used to simulate and analyze protocols in high-performance networks. This session will provide an overview of network simulation tools and how they are being used in simulations projects at SFU.

Introductions by: **Dr. Alan Wagner**, Associate Professor, Dept of Computer Science, UBC

- **Dr. Charles Krasic**, Assistant Professor, Computer Science, UBC
- **Dr. Ljiljana Trajkovic**, Professor, School of Engineering Science, SFU

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

Communication Networks Laboratory

- Communication networks:
  - performance analysis of high-performance packet networks
  - simulation of protocols, connection admission, scheduling, and congestion control algorithms
  - traffic collection, characterization, and modeling
  - intelligent control of communication systems

Communication Networks Laboratory

- Projects
  - DAWN: data analysis of wireline/wireless networks
  - ICON: intelligent control of networks
  - SIMON: simulation of networks
  - Various projects
  - OPNET-specific projects

Communication Networks Laboratory

- SIMON: simulation of networks
  - Improving the Performance of the Gnutella Network
  - BGP Route Flap Damping Algorithms
  - BGP with an Adaptive Minimal Route Advertisement Interval
  - Implementation of BGP in a Network Simulator
  - Selective-TCP for wired/wireless networks
  - TCP over wireless networks
  - Modeling and performance evaluation of a General Packet Radio Services (GPRS) network using OPNET
  - Simulation of General Packet Radio Services (GPRS) network system using OPNET
  - Traffic engineering prioritized IP packets over Multi-Protocol Label Switching (MPLS) network
  - OPNET modeling and simulation of QoS aware medium access control (MAC) in wireless ad hoc networks
  - Implementation and performance simulation of VirtualClock scheduling algorithm in IP networks
  - Simulation of quality of service parameters in IP networks
  - OPNET modeling and simulation of CDPD MAC layer behavior
  - OPNET modeling and simulation of Deficit Round Robin scheduling algorithm for IP networks

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

OPNET

- OPNET: http://www.opnet.com
- University Program
  Academic research and teaching
- "Contributed Papers Library" is an open archive of OPNET studies and scientific papers submitted by OPNET users from educational, government, and industrial organizations worldwide
- "Contributed Models" depot is the official place for sharing OPNET models

SFU OPNET page: http://www.ensc.sfu.ca/~ljilja/opnet/

UNIVERSITY PROGRAMS

Sponsored by OPNET Technologies

Academic OPNET Research and Educational Projects

OPNET Optimum Network Performance

Simon Fraser University

- Enhanced General Packet Radio Service OPNET Model
- OPNET Implementation of the Megaco/H.248 Protocol: Multi-Call and Multi-Connection Scenarios
- Compressed Real-Time Transport Protocol (cRTP)
- Enhancements and performance evaluation of wireless local area networks
- Cellular Digital Packet Data (CDPD) MAC layer model
- OPNET implementation of the Mobile Application Part protocol
- OPNET implementation of endpoint admission control algorithm
- Performance evaluation of M-TCP over wireless links with periodic disconnections

SFU OPNET page: http://www.ensc.sfu.ca/~ljilja/opnet/

UNIVERSITY PROGRAMS

Sponsored by OPNET Technologies

Academic OPNET Research and Educational Projects

OPNET
Optimum Network Performance

Simon Fraser University

- OPNET implementation of the Megaco/H.248 protocol
- Simulation of General Packet Radio Service network
- Performance evaluation of TCP over WLAN 802.11 with the Snoop performance enhancing proxy
- Simulation of congestion control algorithms using OPNET
- OPNET implementation of IPv6 type of service over ATM network
- Differential service for Internet
- Analysis and simulation of wireless data network traffic

OPNET internals

- OPNET Modeler
- Settings
- Creating projects
- Creating links
- Node models
- Packet format
- ICI format
- Process model
- Kernel procedures
- Compiling and debugging
- Collecting results

OPNET modeler

- Editors:
  - Project editor
  - Node editor
  - Process editor
  - Link editor
  - Packet editor

Settings

- Model directories
- Edit -> Preferences:
  - bind_shobj_prog: bind_so_gcc
  - bind_static_prog: bind_gcc
  - comp_prog: comp_gcc
  - repositories: ()

Creating projects

- Network models: scenarios
- Choosing the size of the network:
  - world
  - campus
  - office
  - logical
- Nodes in the network
- Creating object palette
- Trajectories
- Managing scenarios

Creating links

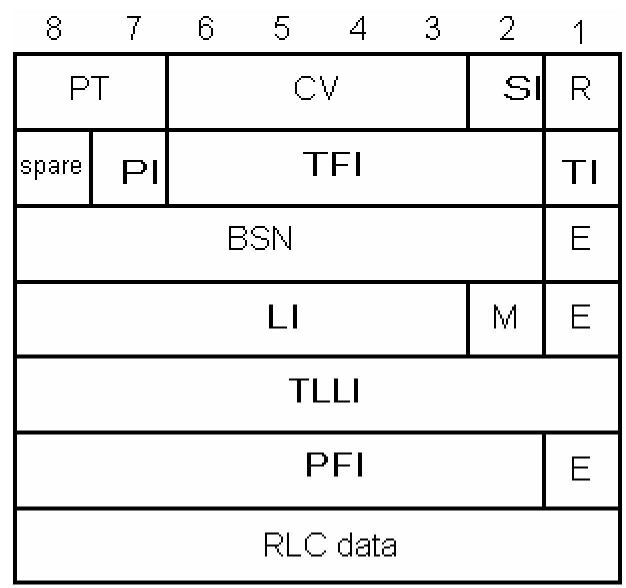
- Links
  - create links using link editor
  - example: gprs_llc_link
- Type of link:
  - point-to-point:
    - simplex: ptsimp
    - duplex: ptdup
  - bus
- Various packet formats are supported
- Transmission delay model (txdel):
  - point-to-point link: dpt_txdel
  - bus: dbu_txdel
- Propagation model
- Error model

Node models

- Create your own
- Modify an existing model
- Various modules:
  - processors
  - queues: active, passive
    - first-in-first-out
    - priority
    - last-in-first-out
  - transmitters, receivers, antenna
  - packet stream
  - statistic wires

Packet format

- Packet editor
- KP: op_pk_create_fmt()
- Fields: length could be zero
- Set and unset fields inside code



Uplink RLC data block

Process model

- States
- Forced and unforced states
- Transitions
- Enter and exit executives
- State variables
- Temporary variables
- Header block
- Function block
- Include files (.h)

Some kernel procedures (KPs)

- Packet processing:
  - Op_pk_get()
  - Op_pk_nfd_set()
  - Op_pk_nfd_get()
  - Op_pk_send()
- Interrupt processing:
  - Op_pk_intrpt_type()
  - Op_pk_intrpt_strm()
  - Op_pk_intrpt_schedule_self()
- Segmentation and reassembly:
  - Op_sar_segbuf_pk_insert()
- Queues:
  - Op_subq_pk_remove()

OPNET features

- Compiling and debugging
  - OPNET debugger
  - print statements
  - error file
- Collecting statistics:
  - global statistics
  - local statistics
- Animation
  - selecting animation

Simulations

- Running scenarios:
  - creating a simulation set
- Viewing results and animation:
  - comparing scenarios
  - playing animation
- Cleaning up
- Files that could be deleted to get more space:
  - .ah
  - .ov
  - temporary files
  - backup files
  - error files

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

Modeling and simulation of a GPRS protocol: case study

- Introduction to GPRS
- GPRS overview
- OPNET model:
  - previous work
  - radio link control/medium access control layer
  - base station subsystem GPRS protocol (BSSGP)
- Simulation results
- Conclusions and future work

Introduction to GPRS networks

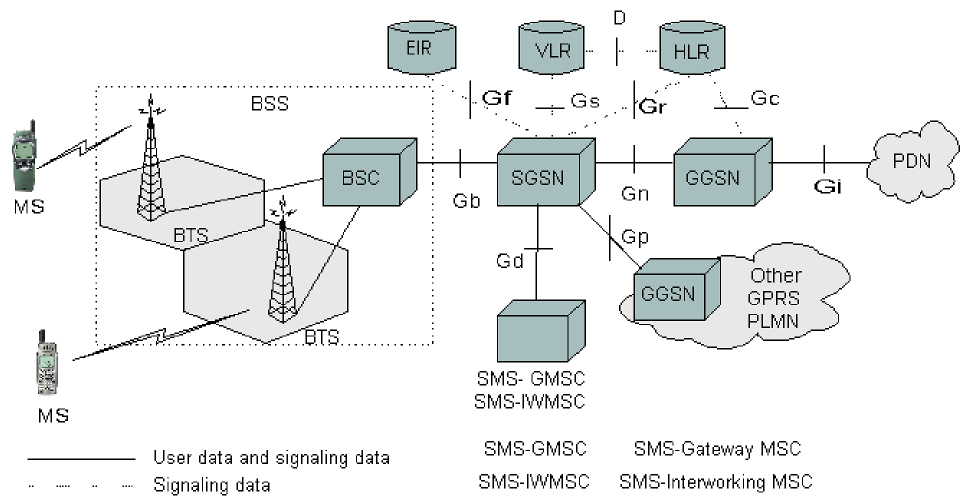
- General Packet Radio Service (GPRS) is a packet-switched wireless network technology
- Introduced as a bearer service for Global System for Mobile Communications (GSM):
  - circuit switched technology
  - bandwidth:
    - 900 MHz and 1,800 MHz (Europe and Asia)
    - 1,900 MHz (North America)
  - billing is based on a connection time
  - entire radio channel dedicated to a single user
  - slow data transmission: 9.6 kbps

GPRS overview

- Radio channels can be concurrently shared among several users
- Up to eight radio interface timeslots can be allocated per TDMA frame, supporting a speed up to 150 kbps
- Users may always be connected to the network
- Radio resources are allocated when users send or receive data
- GPRS employs same frequencies as GSM
- Average transmission speeds: 28.8 kbps to 40 kbps
- Billing may be based on traffic volume
- GPRS Mobile Classes
  - Class A: simultaneous GSM and GPRS communications
  - Class B: GSM and GPRS communications, but not simultaneously
  - Class C: manual selection of GSM or GPRS mode

TDMA: Time Division Multiple Access

GPRS network

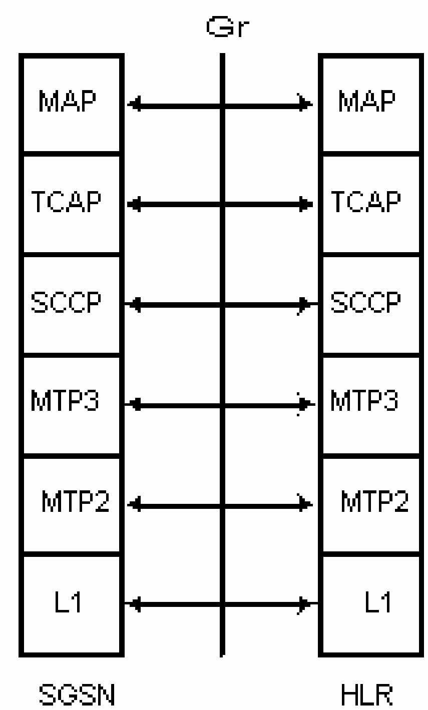


Main components of a GPRS network:

- Mobile Station (MS)
- Base Station Subsystem (BSS)
- Serving GPRS Support Node (SGSN)
- Gateway GPRS Support Node (GGSN)
- Packet Data Network (PDN)
- Equipment Identity Register (EIR)
- Visitors Location Register (VLR)
- Home Location Register (HLR)

Mobile Application Part (MAP) protocol

- Implementation of MAP protocol provides signaling between SGSN and Home Location Register (HLR)
- MAP protocol resides on top of the Signaling System 7 (SS7) protocol stack
- SS7 is an out-of-band signaling system for:
  - Public Switched Telephone Networks (PSTNs)
  - Public Land Mobile Networks (PLMNs)
- MAP provides procedures for:
  - location management
  - subscriber data management
  - authentication
  - call handling
  - subscriber tracing
  - short message service (SMS) management

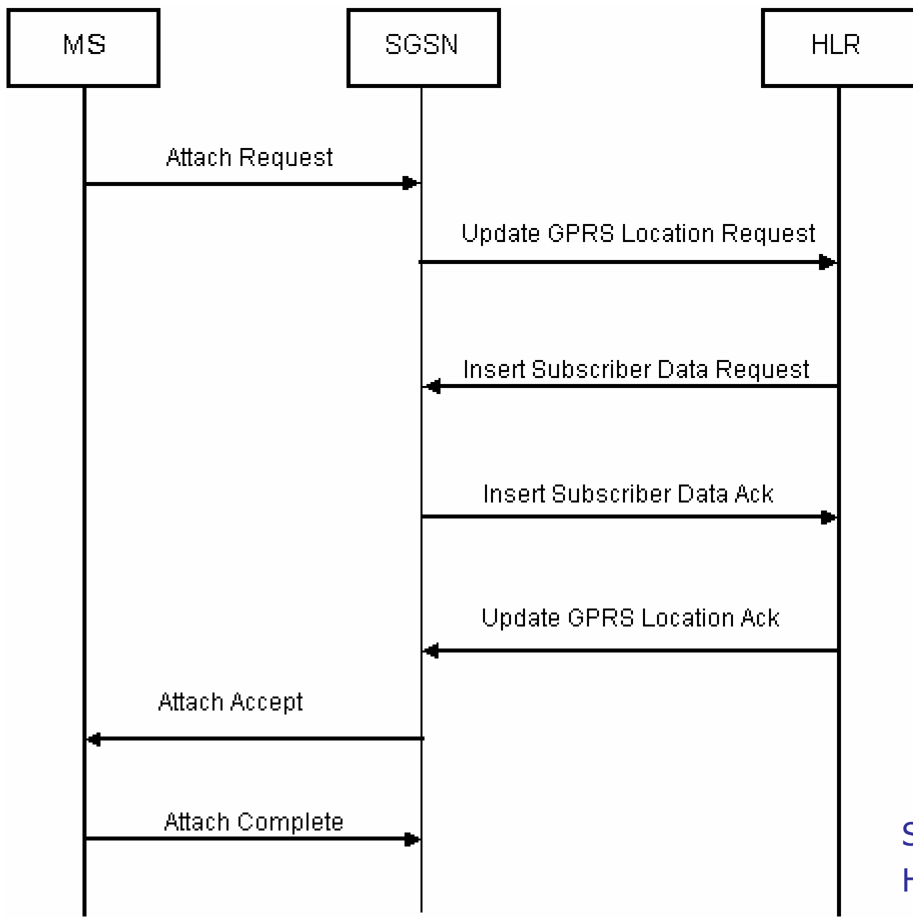


TCAP: Transaction Capabilities Application Part
SCCP: Signaling Connection Control Part
MTP: Message Transfer Part
L1: Level 1

Transmission plane functions

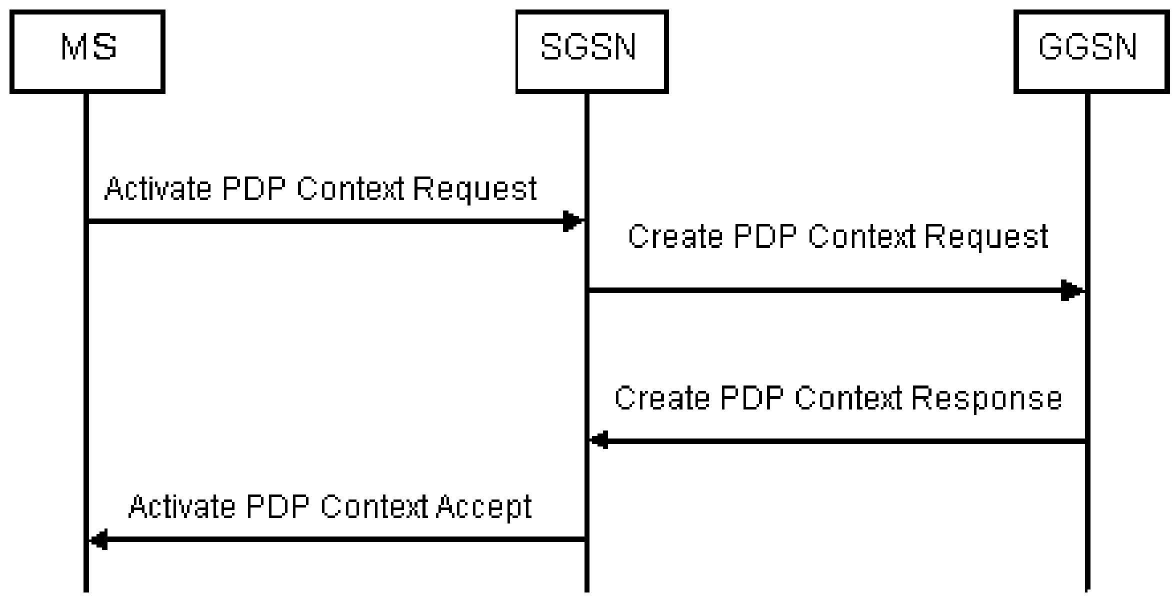
- SNDCP
- LLC
- BSSGP
- GTP

GPRS Attach procedure



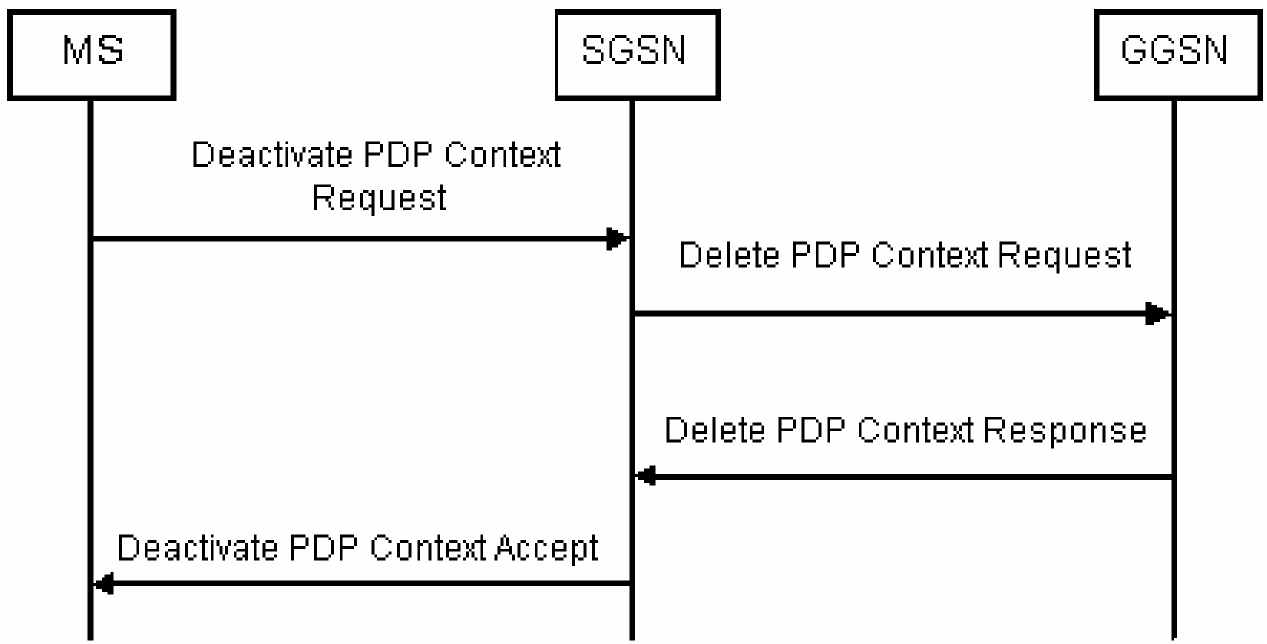
SGSN: Serving GPRS Support Node
HLR: Home Location Register

Activate PDP Context procedure



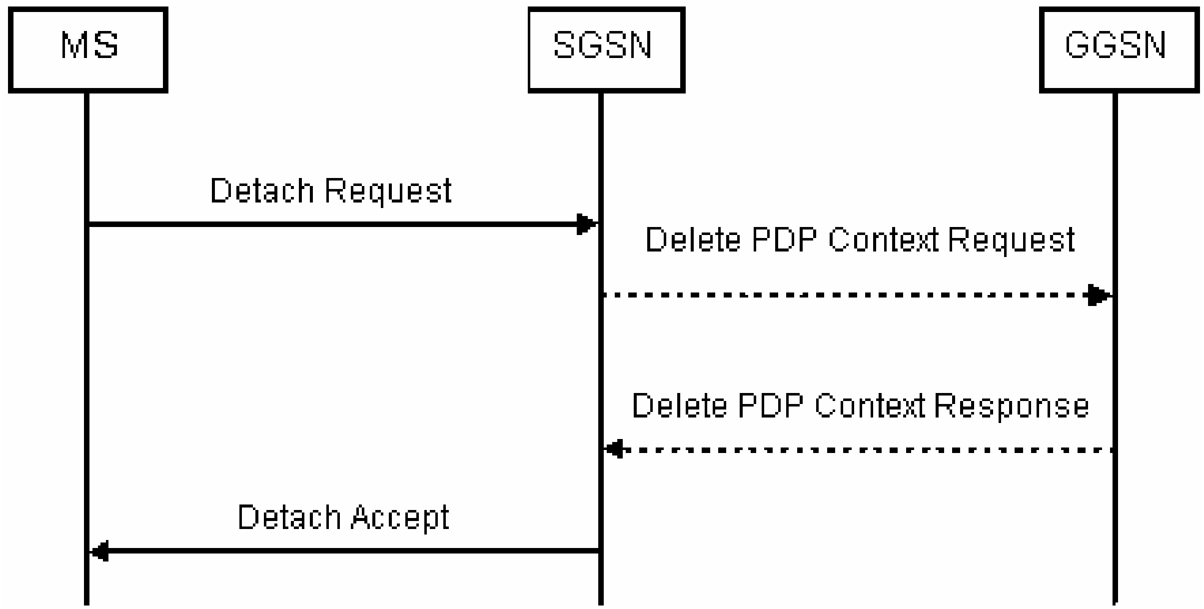
PDP: Packet Data Protocol
SGSN: Serving GPRS Support Node
GGSN: Gateway GPRS Support Node

Deactivation procedure



PDP: Packet Data Protocol

Detach procedure



GPRS Mobility Management (GMM) states

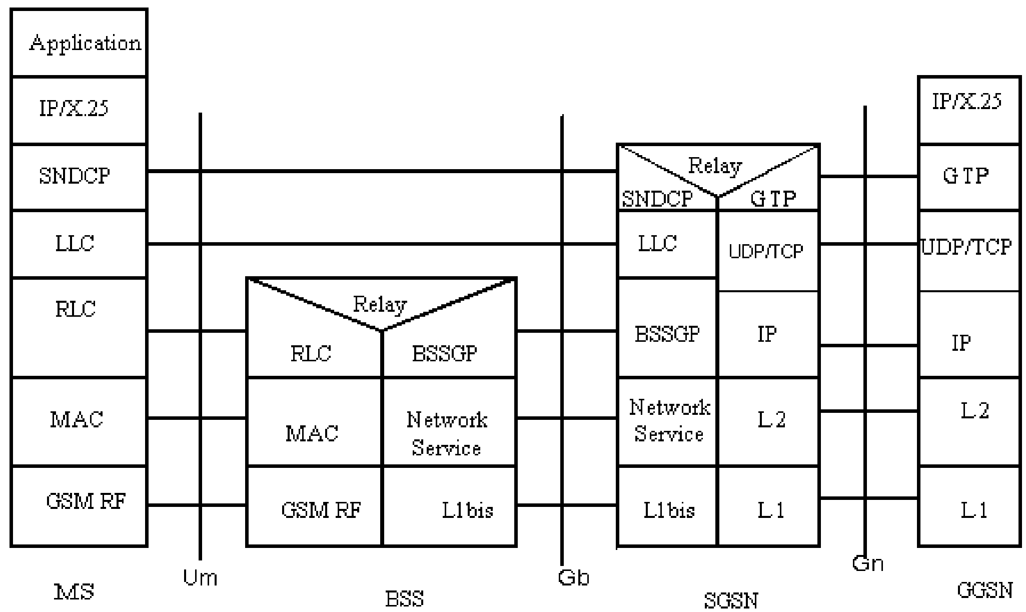
- Idle
- Standby
- Ready

Cell reselection

- Controlled by the mobile or the network
- Based on the received signal level measurements performed by the MS
- Three cell reselection modes:
  - NC0: MS performs autonomous cell reselection without sending measurement reports to the network
  - NC1: GPRS mobile controls the cell reselection process and sends the measurement reports to the network
  - NC2: Network controls the cell reselection procedure

NC: Network Control

GPRS transmission plane



SNDCP: Sub Network Dependent Convergence Protocol
LLC: Logical Link Control layer
RLC: Radio Link Control
MAC: Medium Access Control
BSSGP: Base Station Subsystem GPRS Protocol
GTP: GPRS Tunneling Protocol

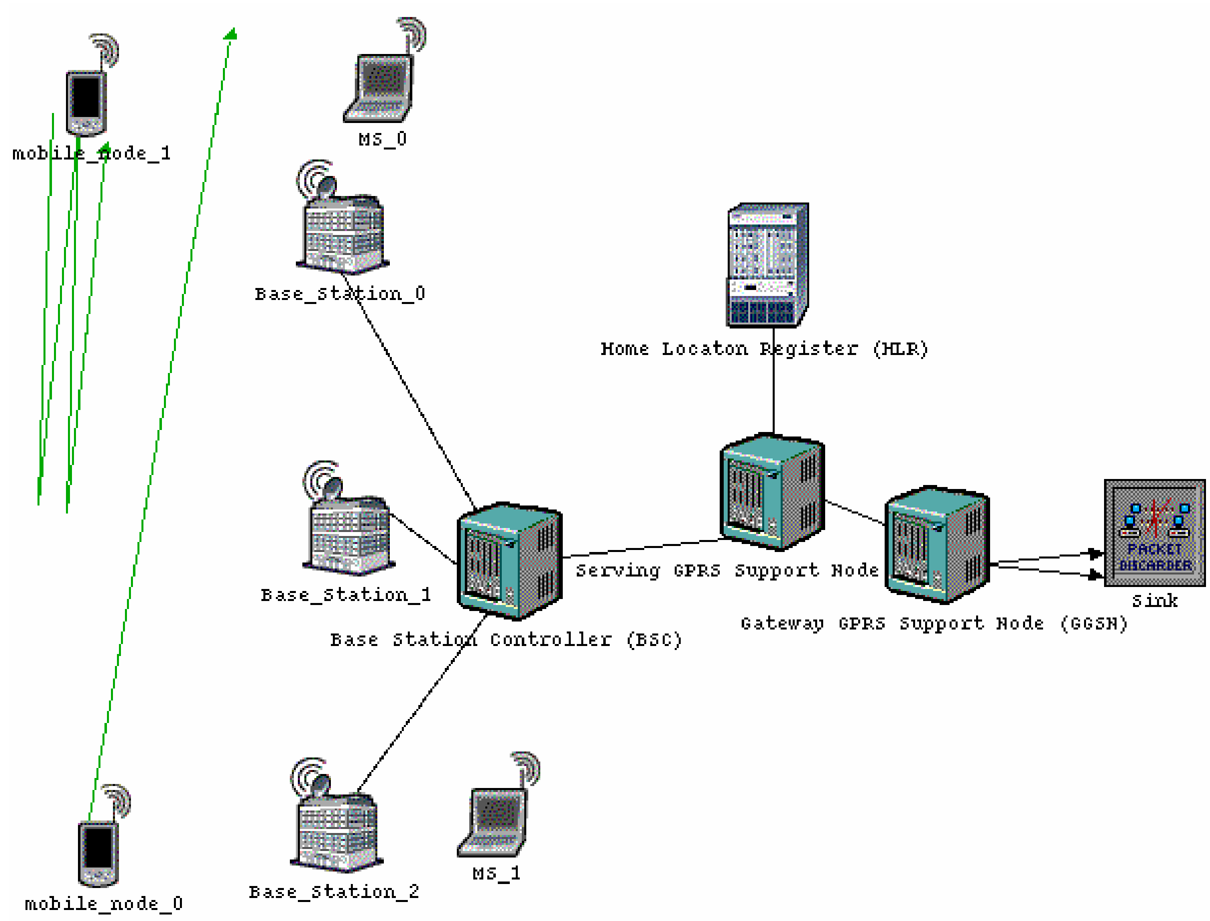
Contributions

- Implementation of:
  - Wireless links
  - Base station controller
  - Cell update
  - Radio link control/medium access control layer
  - Base station subsystem GPRS (BSSGP) protocol

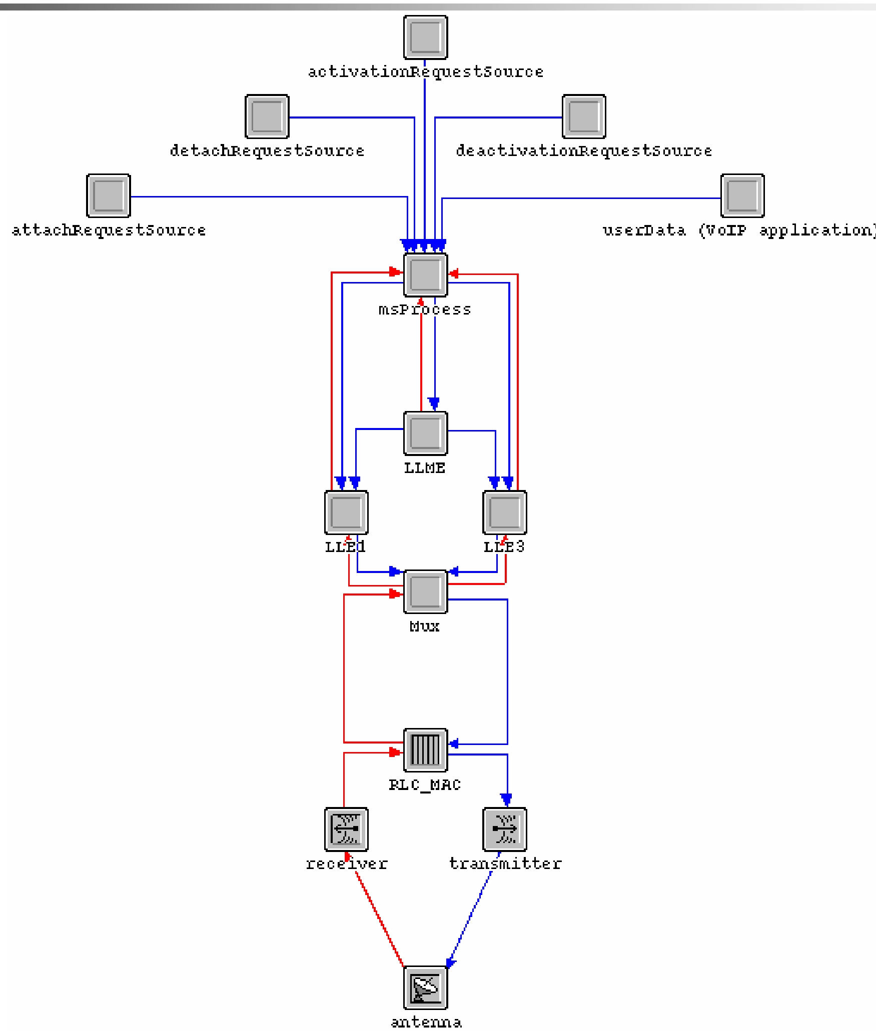
Network simulator: OPNET

- Discrete event simulator
- Hierarchical models paralleling the structure of real networks
- Network models
- Node models
- Process models
- Link models

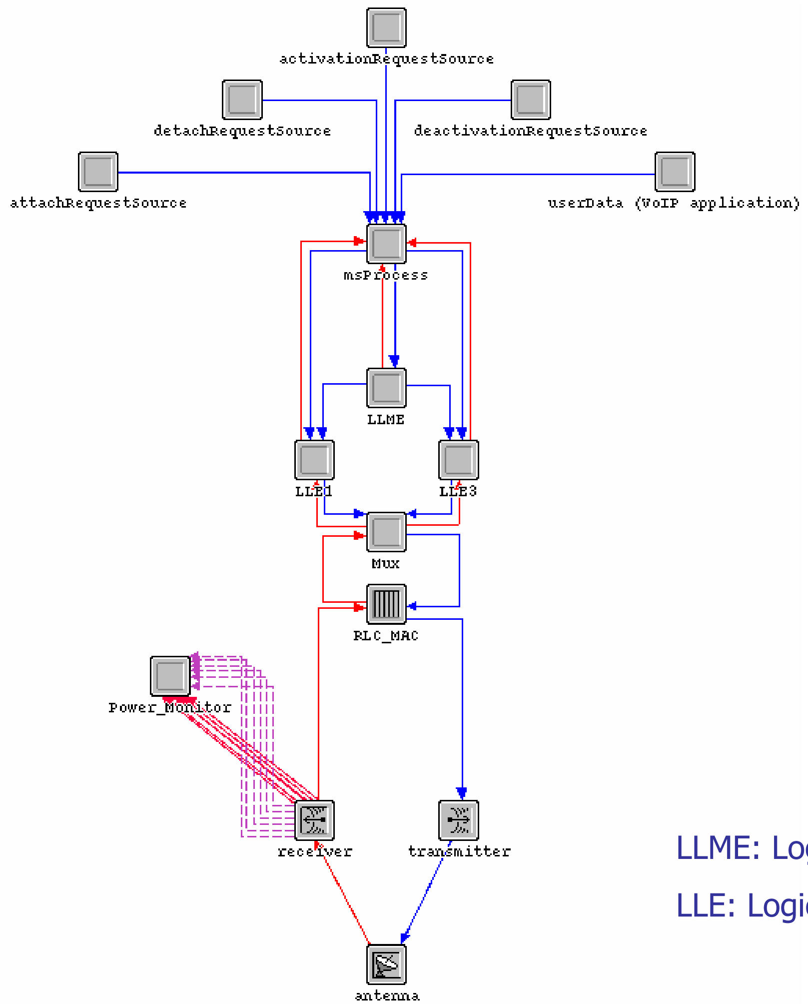
GPRS OPNET model



Node model: Mobile Station



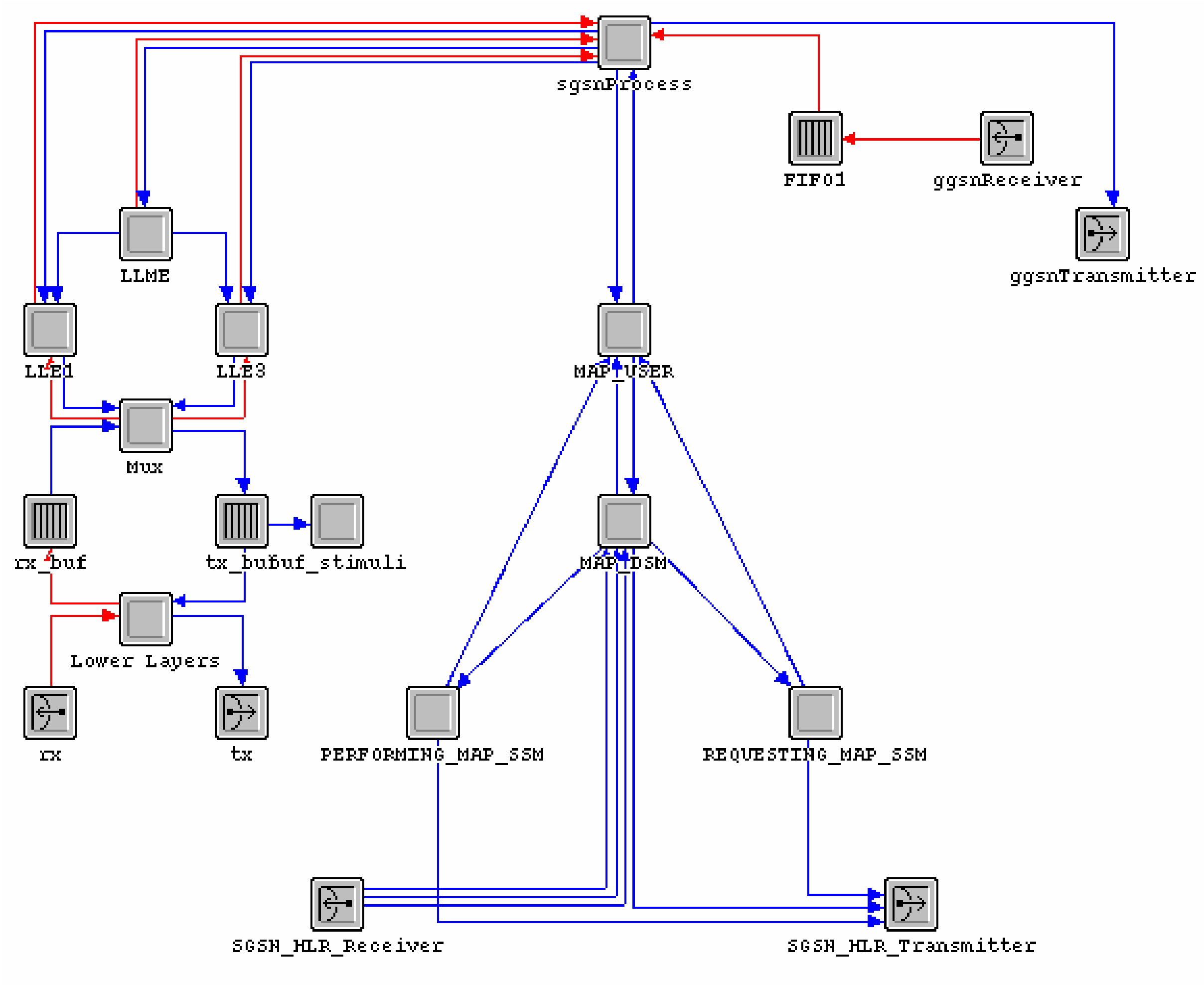
Node model: Mobile Station



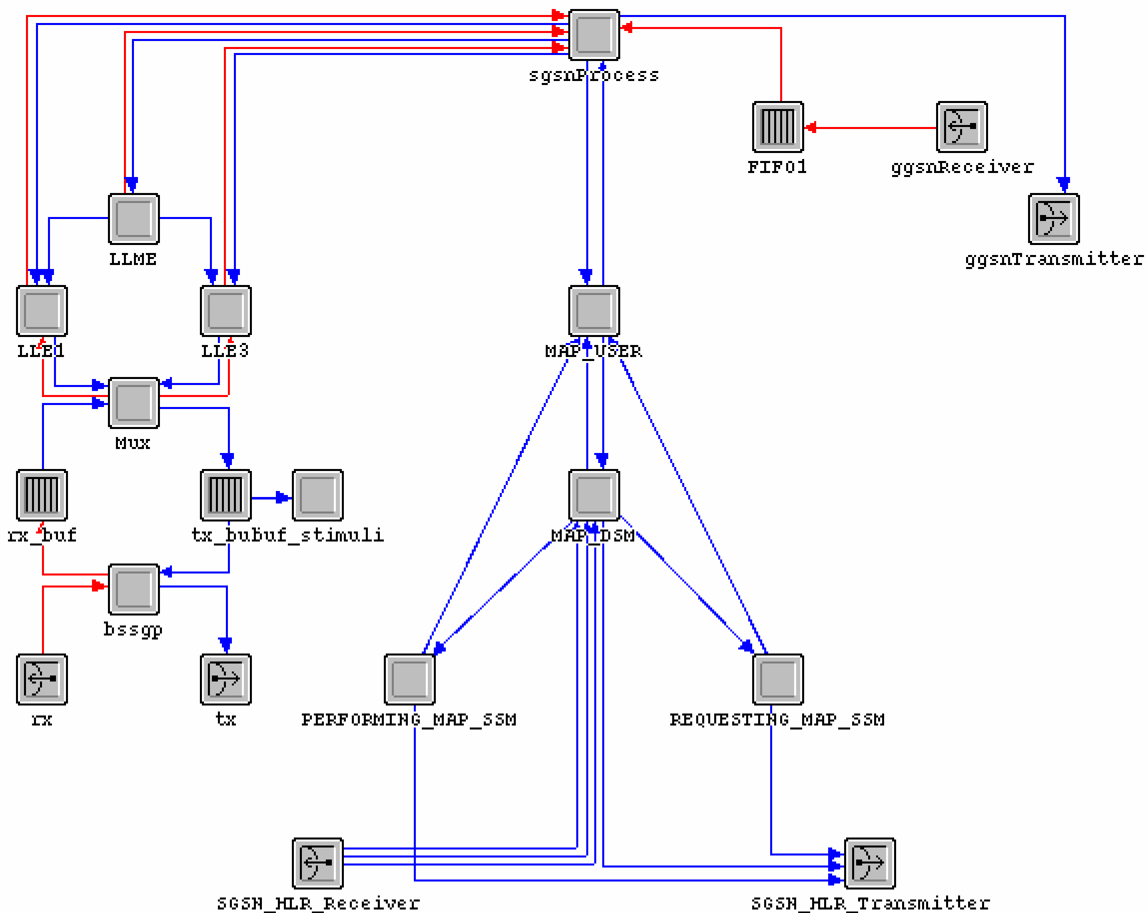
LLME: Logical Link Management Entity

LLE: Logical Link Entity

Node model: SGSN

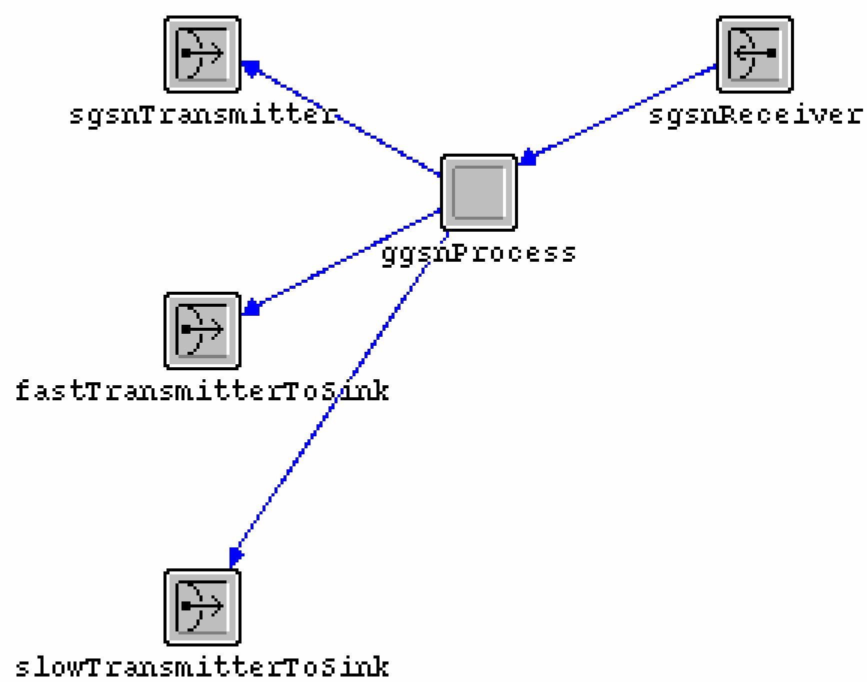


Node model: SGSN

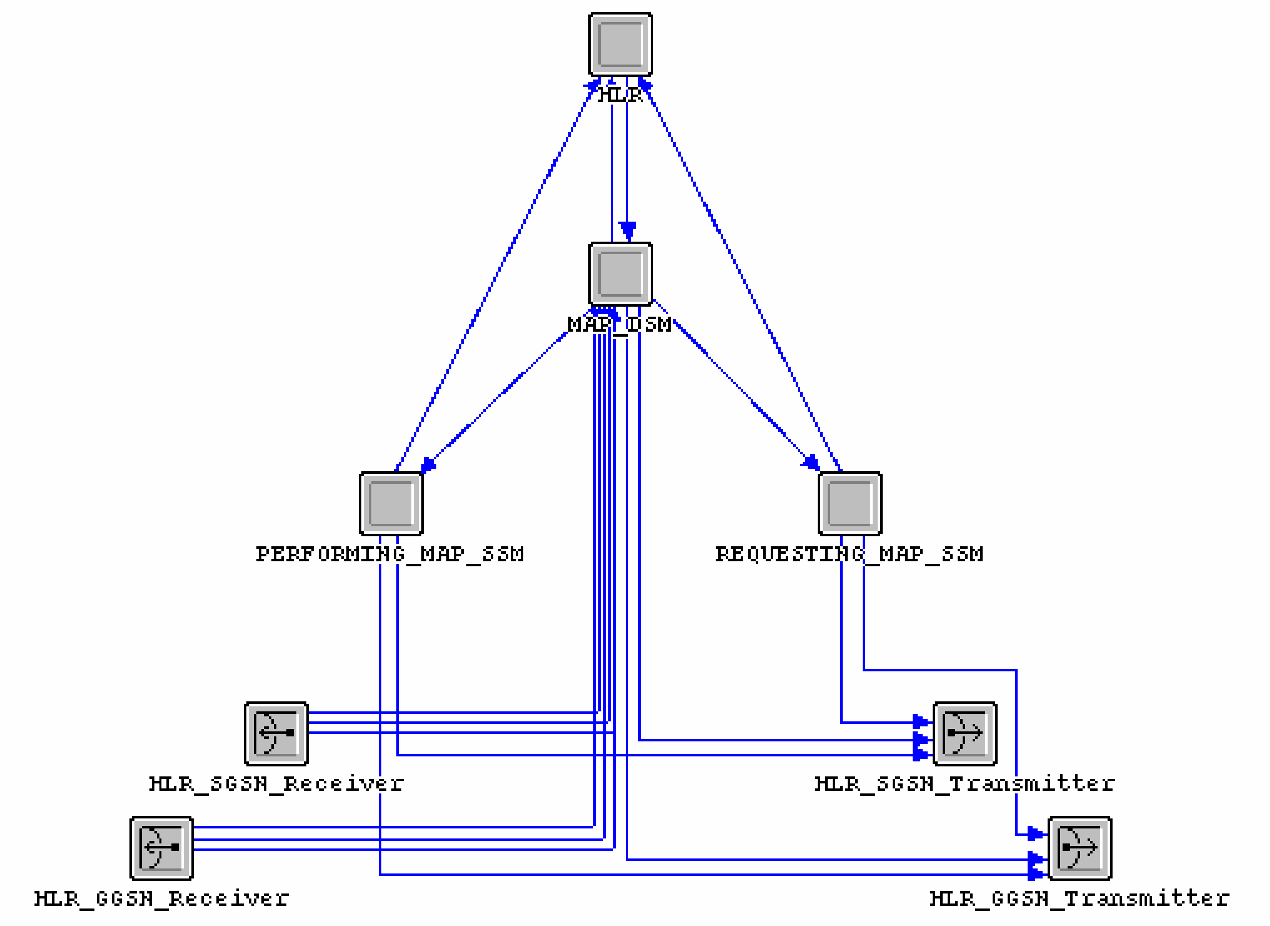


MAP: Mobile Application Part

Node model: GGSN



Node model: HLR

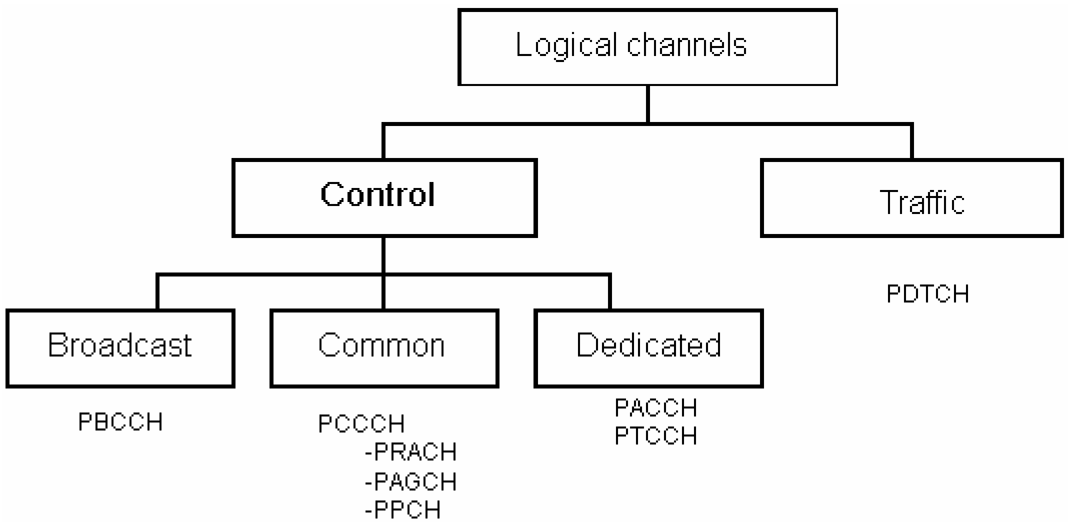


GPRS case study: roadmap

- Introduction to GPRS
- GPRS overview
- OPNET model:
  - previous work
  - radio link control/medium access control layer
  - base station subsystem GPRS protocol (BSSGP)
- Simulation results
- Conclusions and future work

Physical and logical channels

- Packet Data Channel (PDCH): physical channel used for packet logical channels



PBCCH: Packet Broadcast Control Channel
PCCCH: Packet Common Control Channel
PRACH: Packet Random Access Channel
PAGCH: Packet Access Grant Channel
PPCH: Packet Paging Channel
PACCH: Packet Associated Control Channel
PTCCH: Packet Timing Advance Control Channel
PDTCH: Packet Data Traffic Channel

RLC/MAC layer: functions

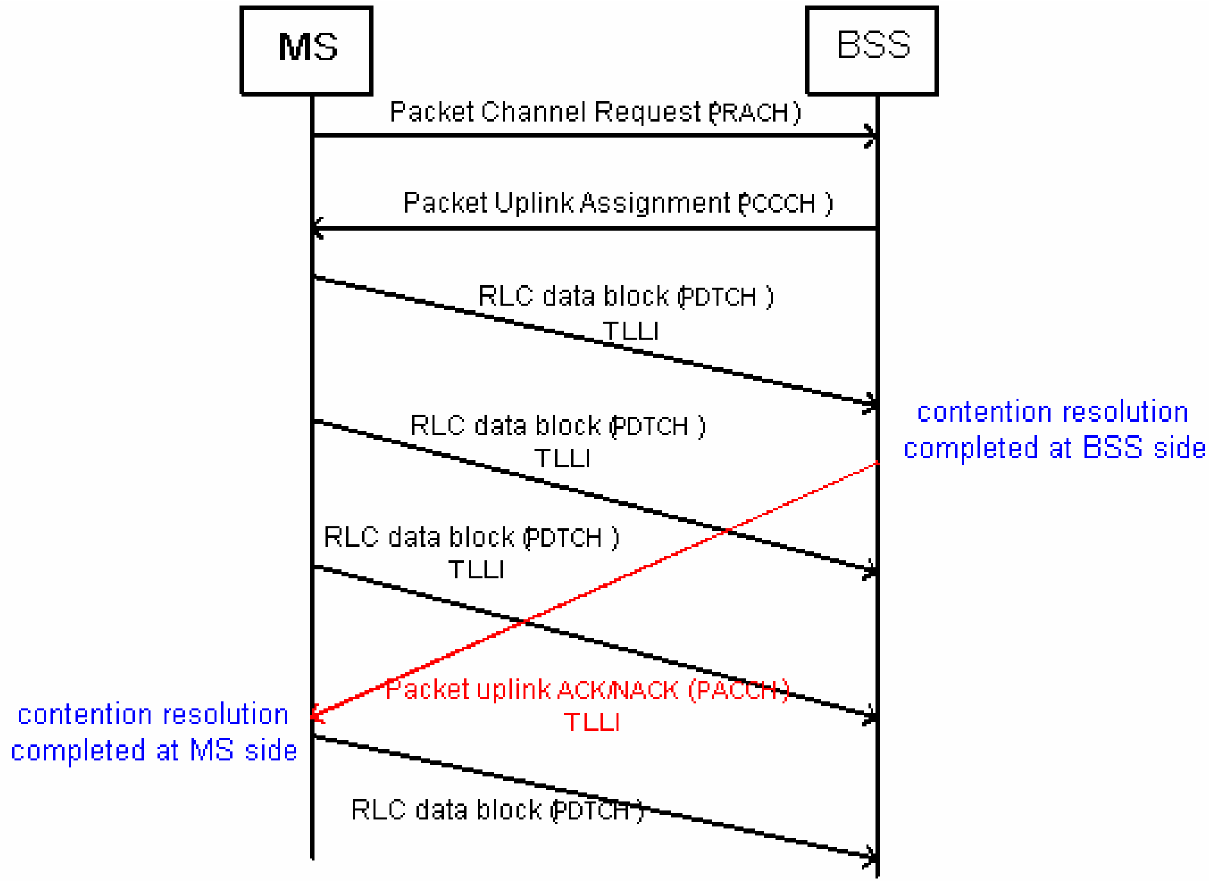
- RLC/MAC layer manages radio resources in a GPRS system
- Direction of data transfer:
  - Mobile Station to BSS: uplink
  - BSS to MS: downlink
- Radio Link Control layer:
  - segments and reassembles LLC PDUs into RLC/MAC blocks
  - acknowledged operation
  - unacknowledged operation
- Medium Access Control layer:
  - controls the allocation of channels and timeslots
  - multiplexes data and control signals
  - provides contention resolution

BSS: Base Station Subsystem
LLC: Logical Link Control
PDU: Packet Data Unit

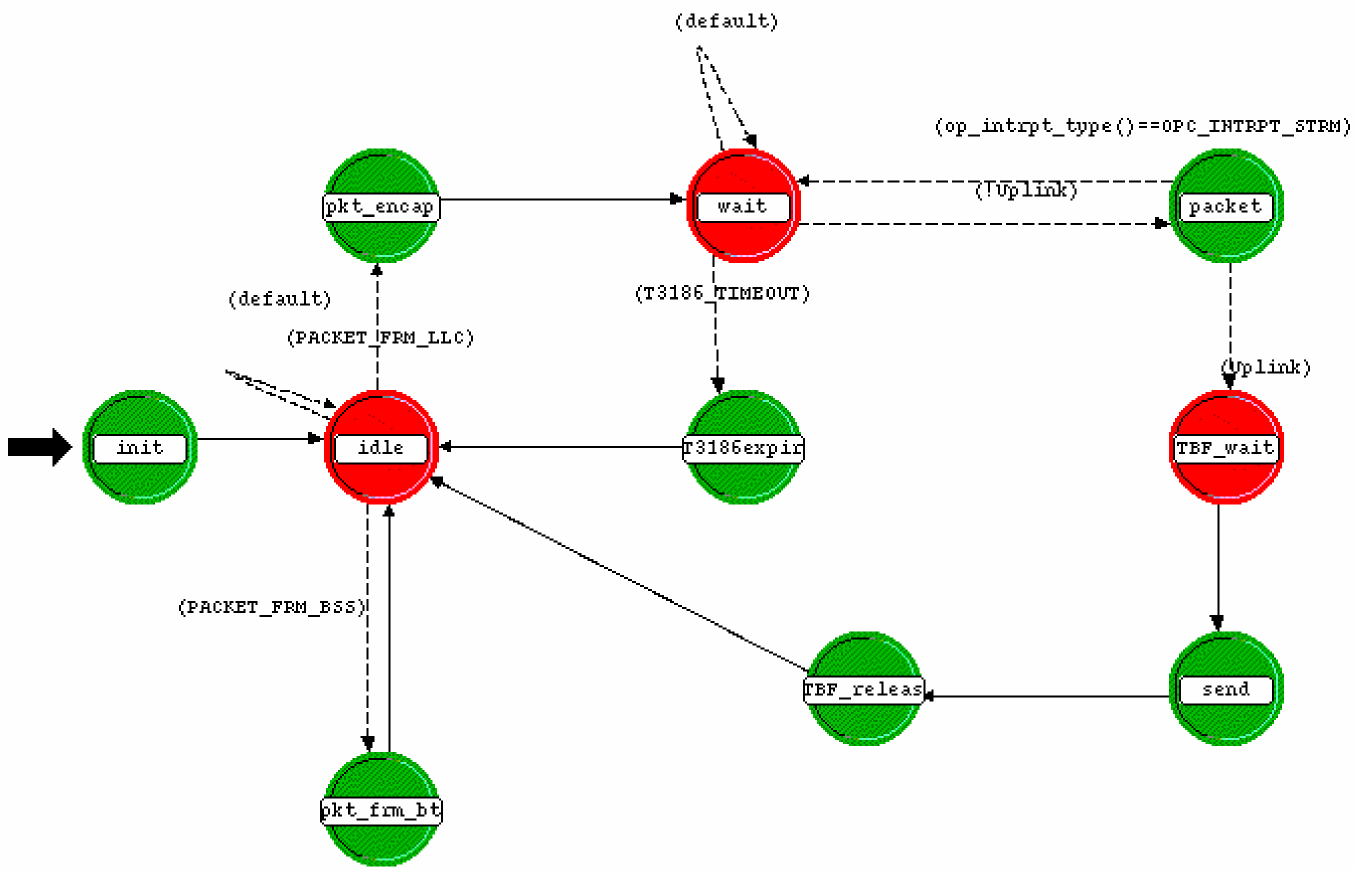
RLC/MAC parameters

- Temporary Block Flow (TBF): physical connection used by two radio resource entities to support unidirectional data transfer on physical channels
  - downlink and uplink TBF
  - temporary
  - maintained for the duration of data transfer only
- Network assigns a Temporary Flow Identity (TFI) to each TBF
  - TFI is unique among TBFs in the same direction

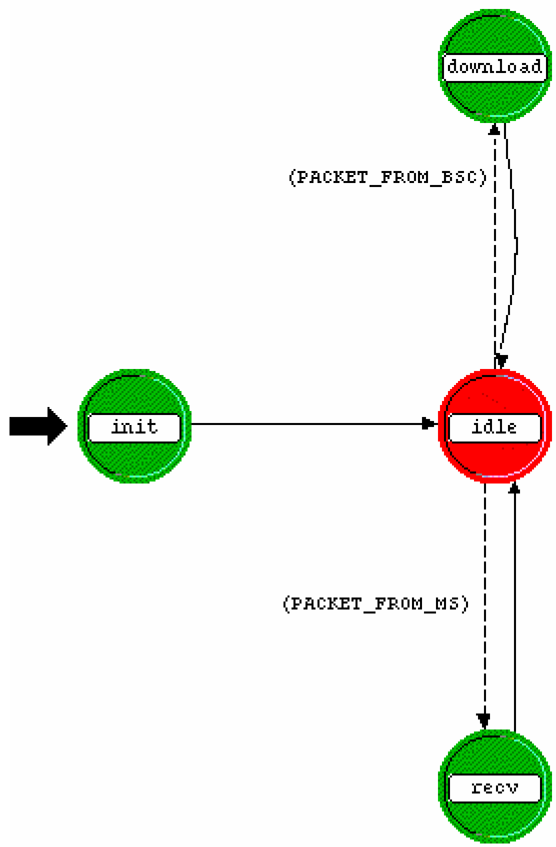
One phase access and contention resolution



Process model: RLC/MAC (MS)



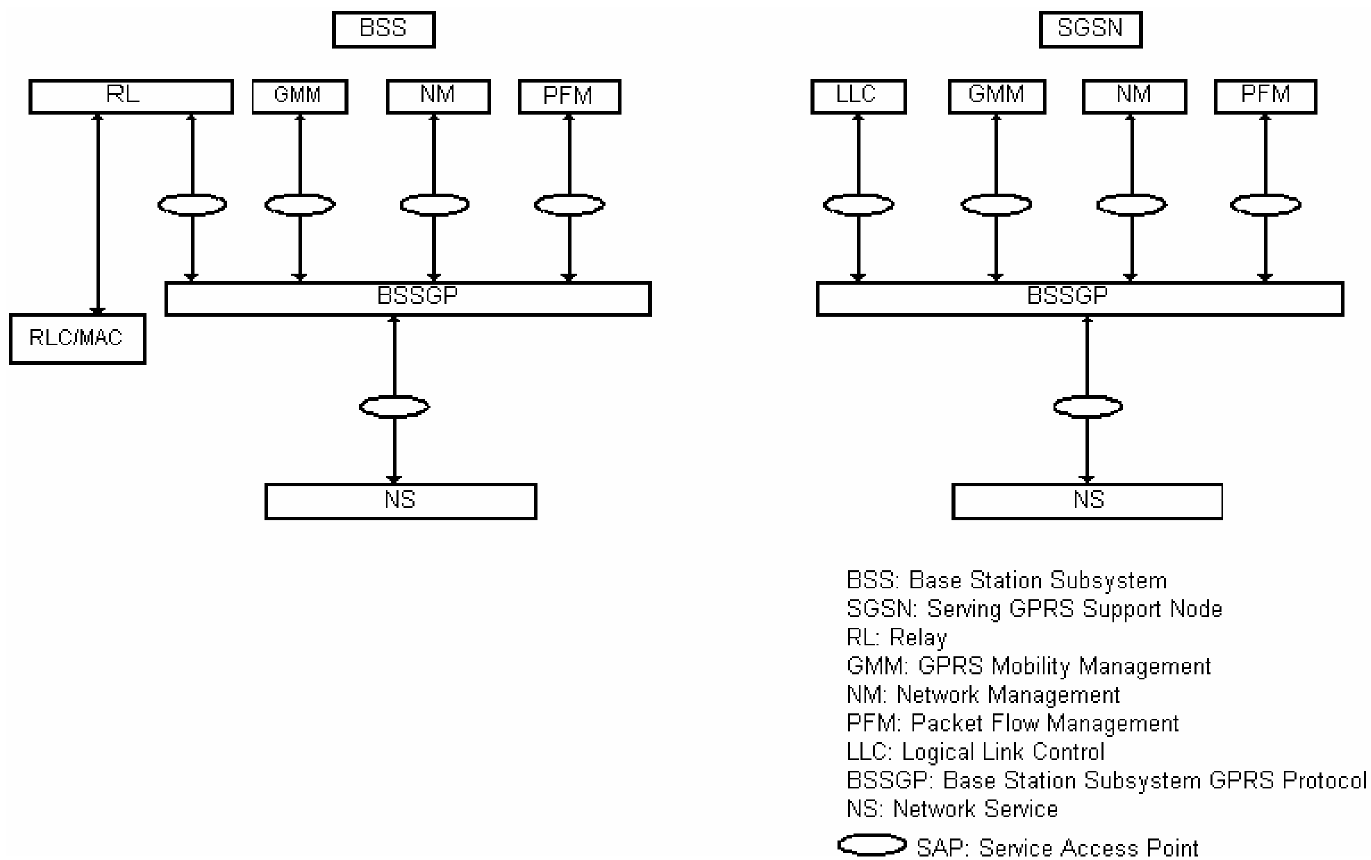
Process model: RLC/MAC (BS)



GPRS case study: roadmap

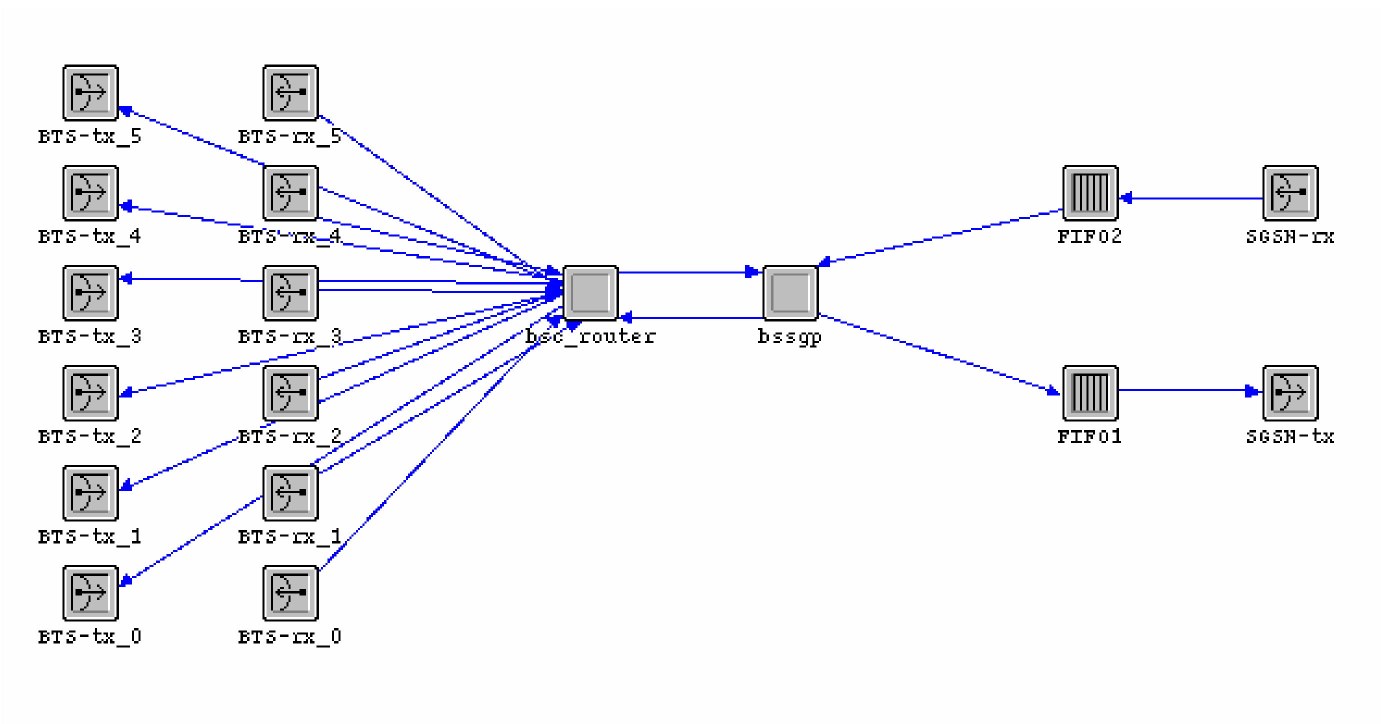
- Introduction to GPRS
- GPRS overview
- OPNET model:
  - previous work
  - radio link control/medium access control layer
  - base station subsystem GPRS protocol (BSSGP)
- Simulation results
- Conclusions and future work

Base Station Subsystem GPRS Protocol (BSSGP): service model

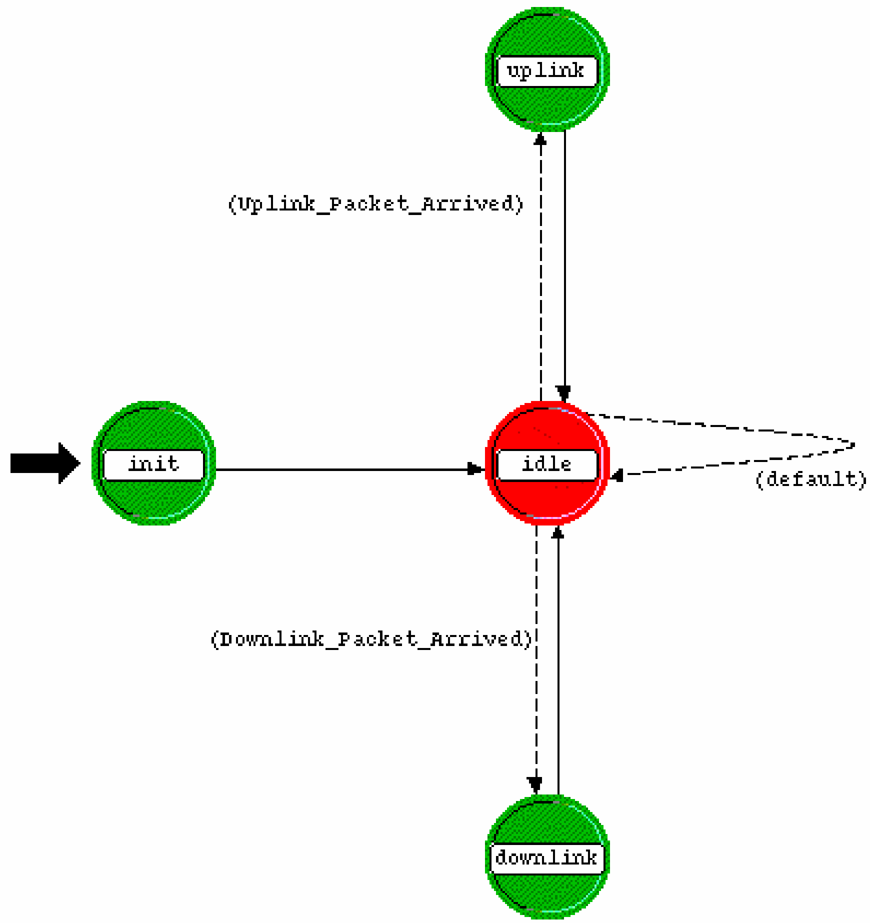


RLC/MAC: Radio Link Control/Medium Access Control

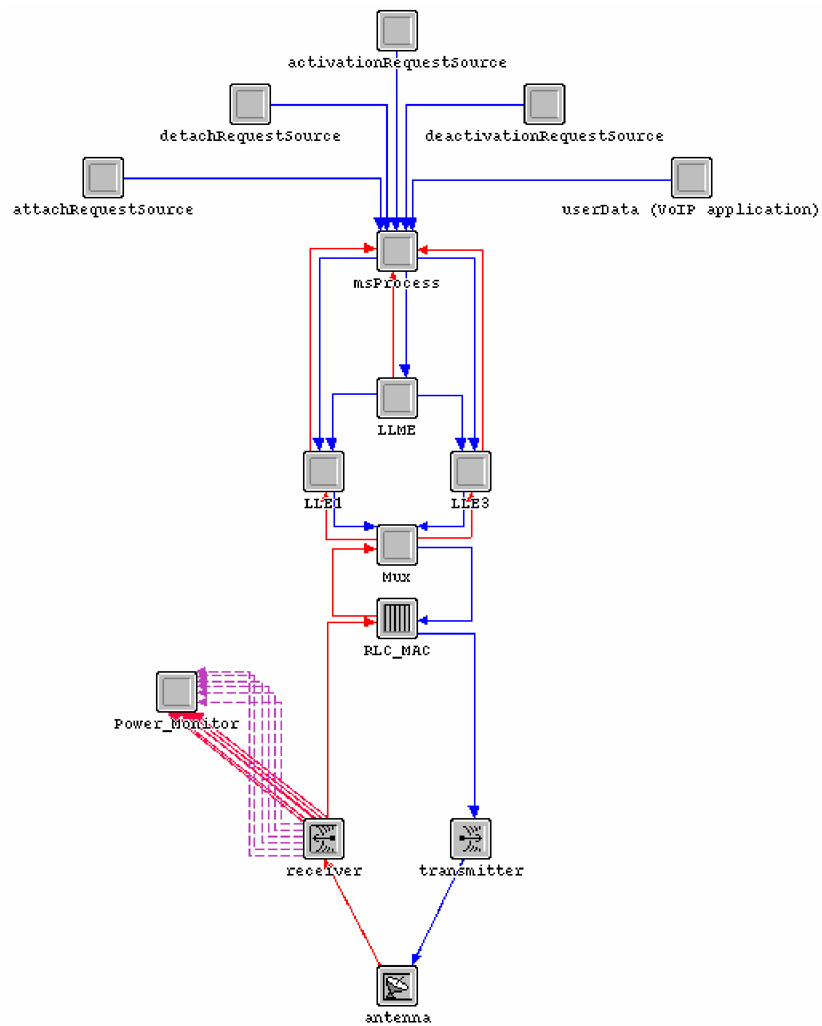
BSC node model



BSSGP process model

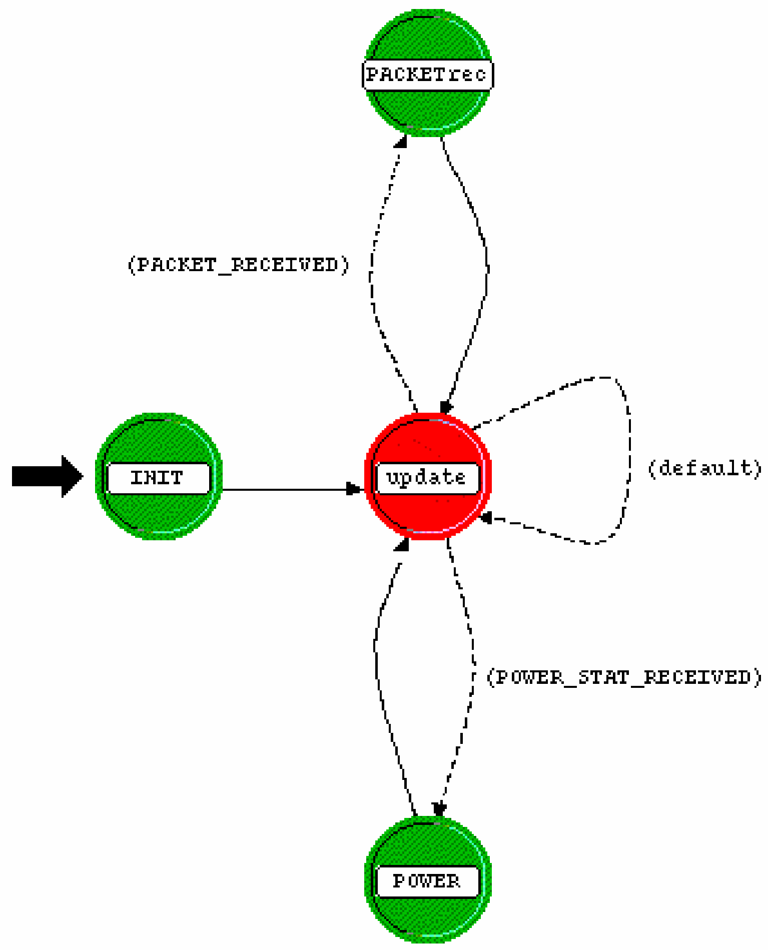


Cell update



activationRequestSource
detachRequestSource
deactivationRequestSource
attachRequestSource
userData (VoIP application)
msProcess
LLME
LLE1
LLE3
Mux
RLC_MAC
Power_Monitor
receiver
transmitter
antenna

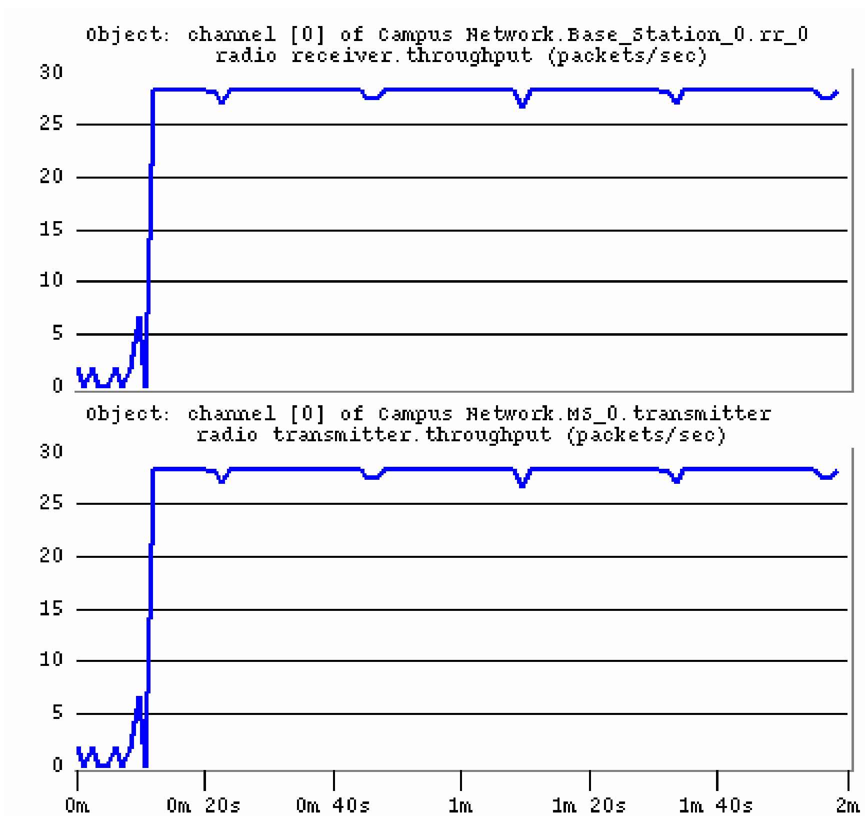
Cell update: Power Monitor process model

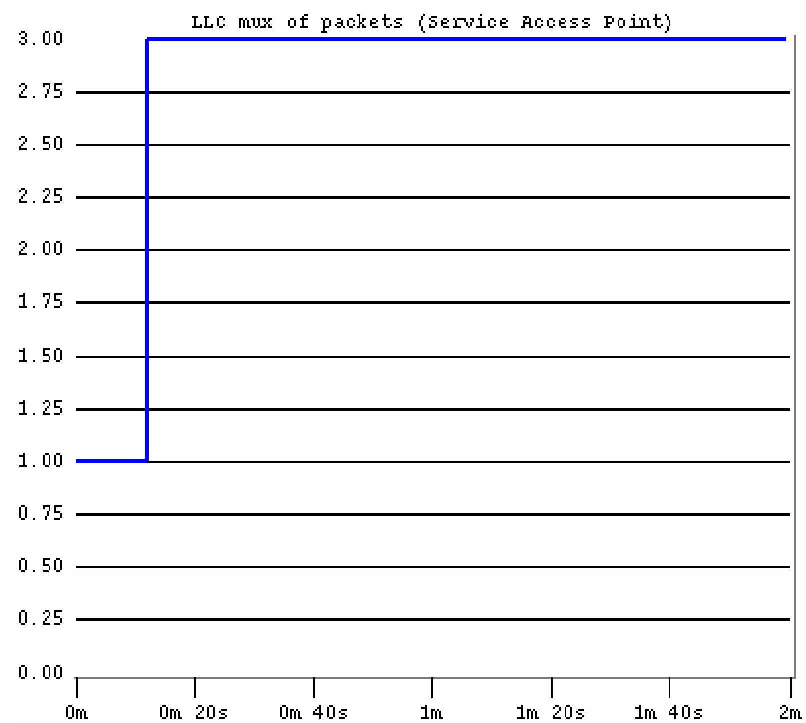


Simulation: configuration of attributes

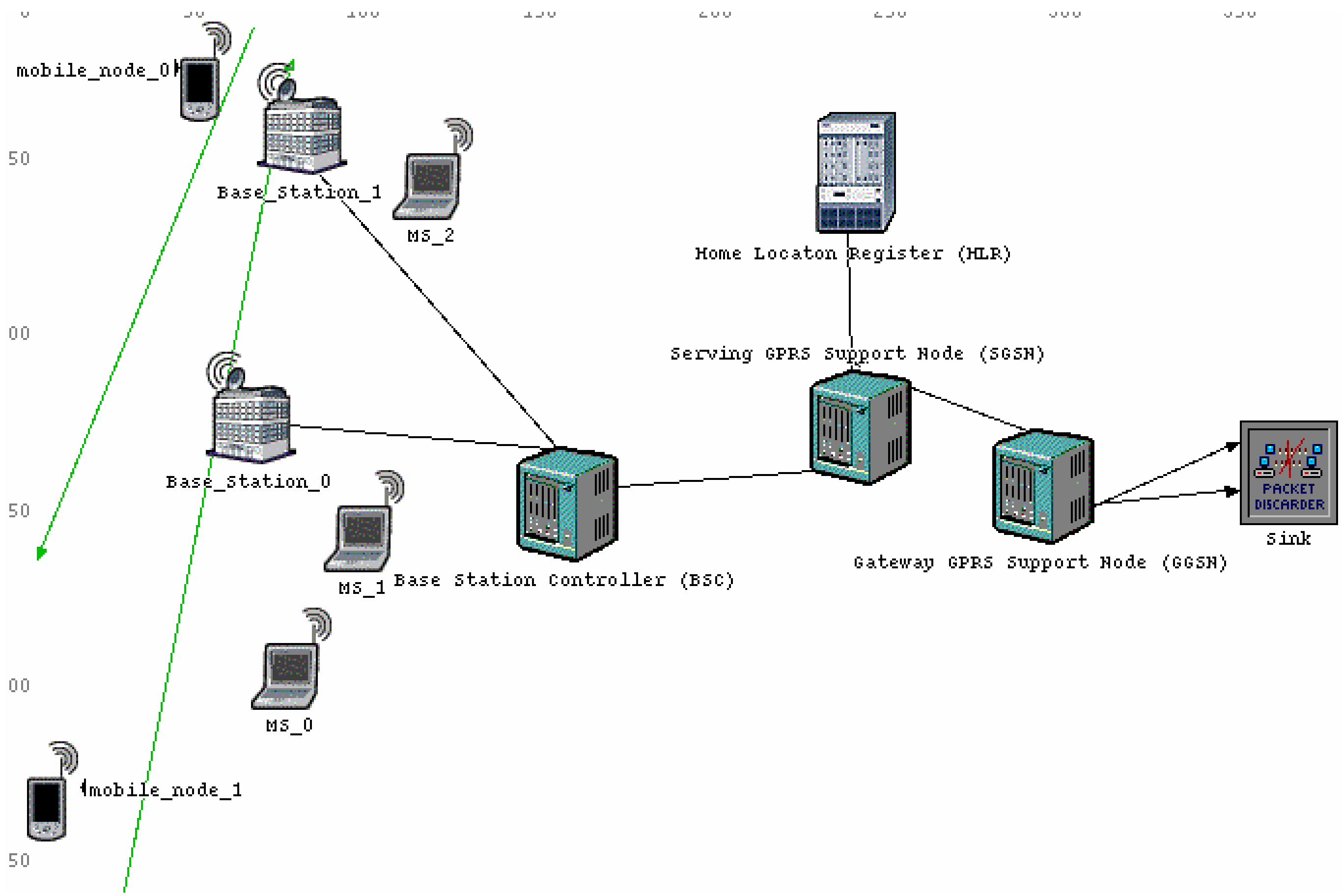
name	MS_0
model	MSProcess_MAC
IMSI	promoted
activationRequestSource.Packet I...	constant (3)
activationRequestSource.Start Ti...	3.0
activationRequestSource.Stop Time	Infinity
attachRequestSource.Packet Inte...	constant (3)
attachRequestSource.Start Time	0.0
attachRequestSource.Stop Time	40
deactivationRequestSource.Pack...	constant (6.0)
deactivationRequestSource.Start ...	Infinity
deactivationRequestSource.Stop ...	Infinity
detachRequestSource.Packet Inte...	constant (6.0)
detachRequestSource.Start Time	Infinity
detachRequestSource.Stop Time	Infinity
receiver.channel [0].min frequency	1,930.2

Simulation results: throughput

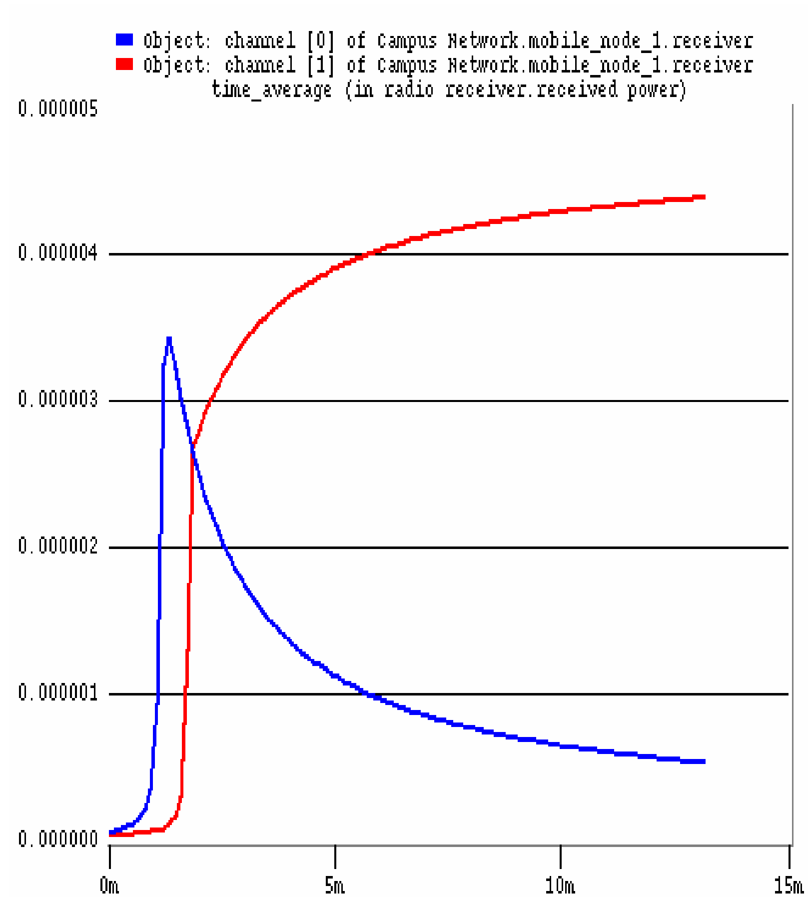


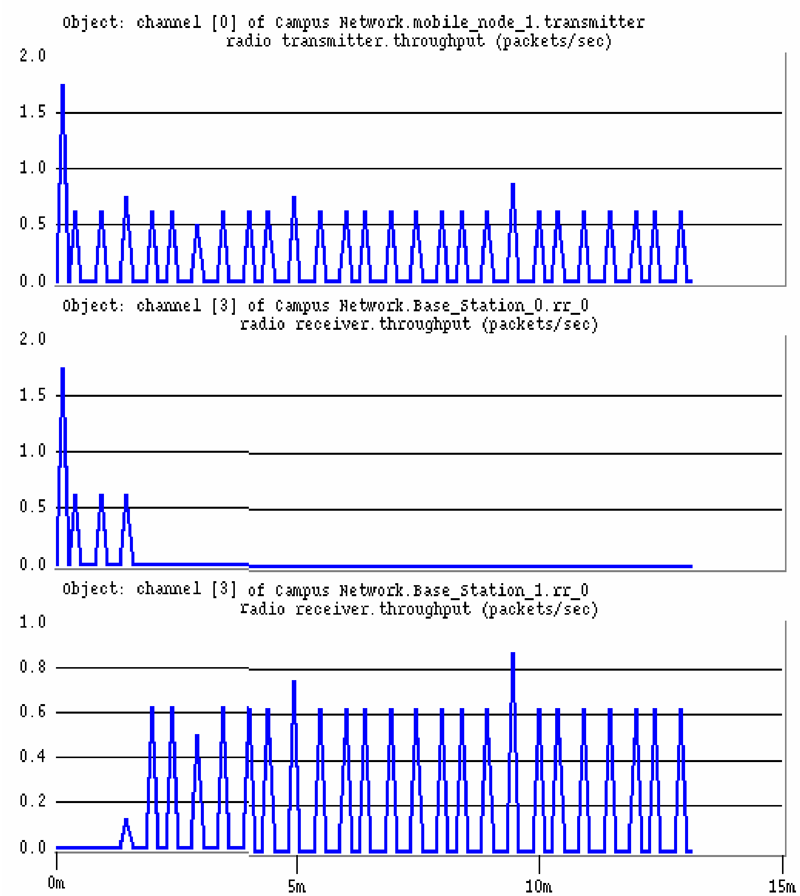


Simulation scenario



Simulation results: mobile_node_1: cell update



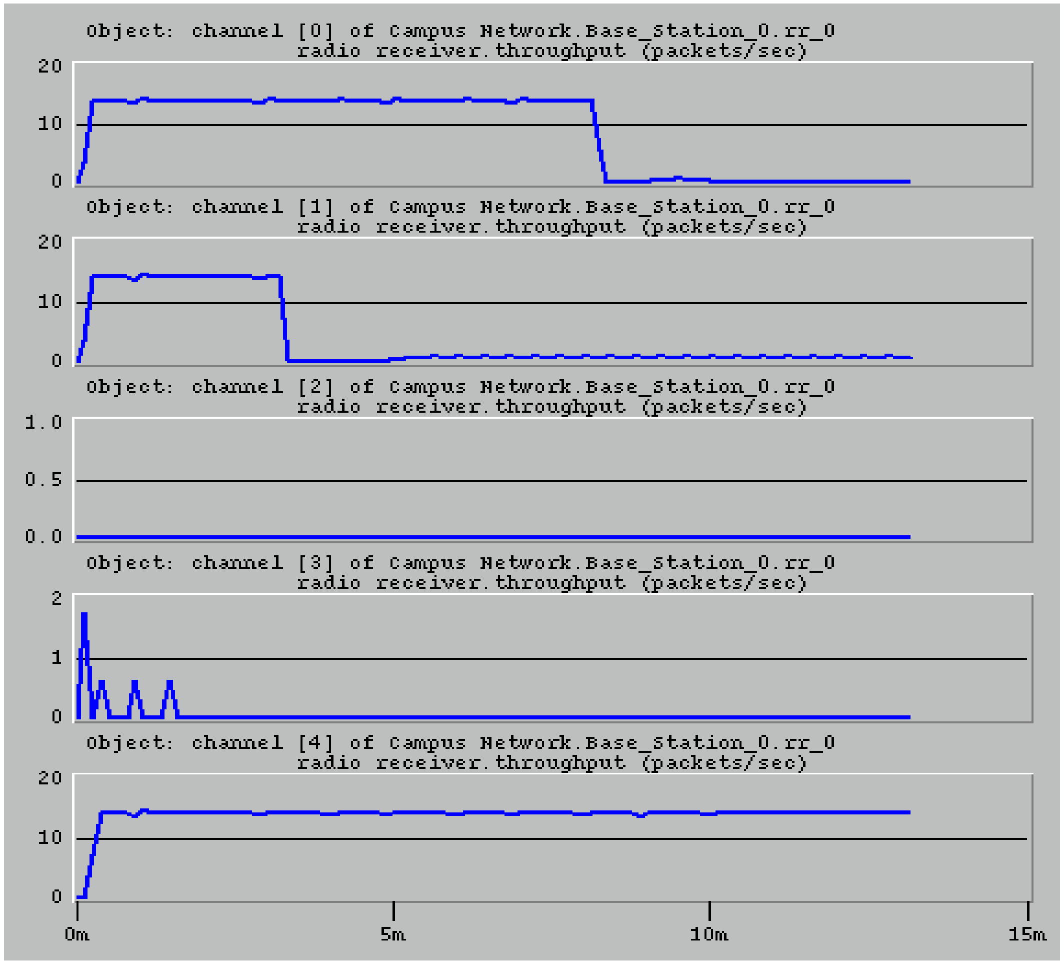


Simulation results: mobile states after simulation

```
SGSN MM and PDP Context after simulation
MM State 0 = detached, 1 = Attached
Attached + Is Active = Activated
==========
IMSI:   0       MM State: 0     Is Active: 0
IMSI:   1       MM State: 0     Is Active: 0
IMSI:   2       MM State: 0     Is Active: 0
IMSI:   3       MM State: 1     Is Active: 0
IMSI:   4       MM State: 1     Is Active: 1
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0
IMSI:   -1      MM State: 0     Is Active: 0

State information of MS after simulation, 0 = Detached, 1 = Attached, 2 = Activated
 Name: MS_0      IMSI:   0       MM State: 0
 Name: MS_1      IMSI:   1       MM State: 1
 Name: MS_2      IMSI:   2       MM State: 0
 Name: mobile_node_1     IMSI:   3       MM State: 1
 Name: mobile_node_0     IMSI:   4       MM State: 2
```

Results: Base_Station_0 throughput



GPRS case study: conclusions and future work

- Completed:
  - GPRS model was implemented in OPNET
  - Implementation includes
    - RLC/MAC protocol
    - BSSGP layer
- Future work:
  - implementation of RLC/MAC layer will be enhanced by adding a two-phase access procedure
  - additional simulations to demonstrate the contention resolution
  - performance evaluation

BSSGP: base station subsystem GPRS protocol

GPRS case study: references

- E. Seurre, P. Savelli, and P. Pietri, *GPRS for Mobile Internet*. Boston: Artech House, 2003.
- 3rd Generation Partnership Project, TS 03.60 version 7.9.0 General Packet Radio Service (GPRS) Service description.
- 3rd Generation Partnership Project, TS 04.64 version 8.7.0 General Packet Radio Service (GPRS) Logical Link Control (LLC) layer specification.
- 3rd Generation Partnership Project, TS 04.60 version 7.9.0 General Packet Radio Service (GPRS) Radio Link Control/Medium Access Control (RLC/MAC) layer specification.
- 3rd Generation Partnership Project, TS 08.18 version 8.10.0 General Packet Radio Service (GPRS) Base Station Subsystem GPRS Protocol specification.
- G. Jain and P. Shekhar, "GPRS model enhancements," *OPNETWORK 2003*, Washington, DC, Aug. 2003.
- Y. Sawant, K. Sastry, R. Krishnamoorthy, and S. Taparia, "GPRS model enhancements," *OPNETWORK 2004*, Washington, DC, Aug. 2004.
- R. Ng and Lj. Trajković, "Simulation of General Packet Radio Service network," *OPNETWORK 2002,* Washington, DC, Aug. 2002.
- V. Vukadinovic and Lj. Trajković, "OPNET implementation of the Mobile Application Part protocol," *OPNETWORK 2003*, Washington, DC, Aug. 2003.
- R. Narayanan, P. Chan, M. Johansson, F. Zimmermann, and Lj. Trajkovic, "Enhanced General Packet Radio Service OPNET model," *OPNETWORK 2004*, Washington, DC, Aug. 2004.
- OPNET documentation V.11.0.A, OPNET Technologies, Inc., Bethesda, MD, 2004.

GPRS OPNET Contributed Model

- Revised model (version 10.0.A, PL2) created by:
  Renju Narayan, Frank Zimmermann, Paulman Chan, and Vladimir Vukadinovic
- Revised model (version 9.1.A, PL1) created by:
  Vladimir Vukadinovic and Mikael Johansson
- Revised model (version 9.0.A, PL4) created by:
  James Song and Vladimir Vukadinovic
- Original model (version 7.0.B, PL6) created by:
  Ricky Ng

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

ns-2: network simulator

compiled from slides:

John Heidemann (USC/ISI)
Polly Huang (ETH Zurich)

UCLA/IPAM presentation, Mar. 2002

and

Padmaparna Haldar (USC/ISI)
Xuan Chen (USC/ISI)

ISI ns-2 Tutorial 2002, Nov. 2002

ns-2: roadmap

- Basic introduction
- ns-2 fundamentals
- ns-2 programming internal
- Extending ns-2 simulator

ns-2: history

- 1989: REAL network simulator
- 1995: DARPA VINT project at LBL, Xerox PARC, UCB, and USC/ISI
- Present: DARPA SAMAN project and NSF CONSER project
  - Collaboration with other researchers including ICIR (formerly ACIRI)

ns-2: status

- Periodical release (ns-2.29, Oct. 2005)
  - ~200k lines of code in C++ and OTcl,
  - ~100 test suites and 100+ examples
  - ns-2 manual (371 pages)
  - daily snapshot (with auto-validation)
- Stability validation
  - http://www.isi.edu/nsnam/ns/ns-tests.html

ns-2: status

- Platform support:
  - FreeBSD, Linux, Solaris, Windows, and Mac
- User base:
  - > 1k institutes (50 countries) and > 10k users
  - about 300 posts to ns-users@isi.edu monthly

ns-2: functionalities

- Wired
  - routing: distance vector (DV), link state (LS), multicast
  - transport protocols: TCP, UDP, RTP and SCTP
  - traffic sources: web, ftp, telnet, cbr, stochastic
  - queuing disciplines: drop-tail, RED, FQ, SFQ, DRR
  - QoS: IntServ and Diffserv
  - emulation
- Wireless
  - ad hoc routing (AODV, DSDV) and mobile IP
  - directed diffusion, sensor-MAC
- Tracing, visualization, various utilities

ns-2: components

- ns-2: the simulator
- nam: the network animator:
  - visualize ns (or other) outputs
  - nam editor: GUI interface to generate ns scripts
- Pre-processing:
  - traffic and topology generators
- Post-processing:
  - trace analysis with Unix or GNU/Linux tools like awk, Perl, or Tcl
  - graphical visualization with xgraph

ns-2: components

- Main components of ns-2
  - Tcl/TK 8.x (8.4.5 preferred):
    http://resource.tcl.tk/resource/software/tcltk/
  - OTcl and TclCL:
    http://otcl-tclcl.sourceforge.net
  - ns-2 and nam-1:
    http://www.isi.edu/nsnam/dist
- Other utilities
  - http://www.isi.edu/nsnam/ns/ns-build.html
  - Tcl-debug, GT-ITM, xgraph

ns-2: installation notes

- If the GNU/Linux distribution comes with Tcl/TK:
  - install each individual packages separately (ns, nam, xgraph, GT-ITM) to avoid conflicts and to save space
- If you are unfamiliar with the UNIX or GNU/Linux environment:
  - install ns-allinone
- ns-2 is available for Windows 9x/2000/XP under Cygwin:
  - not widely supported and problematic (avoid it)

ns-2: models

- Traffic models and applications:
  - Web, FTP, telnet, constant bit rate, real audio
- Transport protocols:
  - unicast: TCP (Reno, Vegas, etc.), UDP
  - multicast: SRM (scalable reliable multicast)
- Routing and queuing:
  - wired routing, ad hoc routing and directed diffusion
  - queuing protocols: RED, drop-tail, etc.
- Physical media:
  - wired (point-to-point, LANs), wireless (multiple propagation models), satellite

ns-2: roadmap

- Basic introduction
- ns-2 fundamentals
- ns-2 programming internal
- Extending ns-2 simulator

ns-2: the network simulator

- A discrete event simulator
- Focused on modeling network protocols:
  - wired, wireless, satellite
  - TCP, UDP, multicast, unicast
  - web, telnet, ftp
  - ad hoc routing, sensor networks
  - stats, tracing, error models, …

ns-2 architecture

- Object-oriented (C++, OTcl)
- Modular approach
  - fine-grained object composition
- \+ Reusability
- \+ Maintenance
- – Performance (speed and memory)
- – Careful planning of modularity

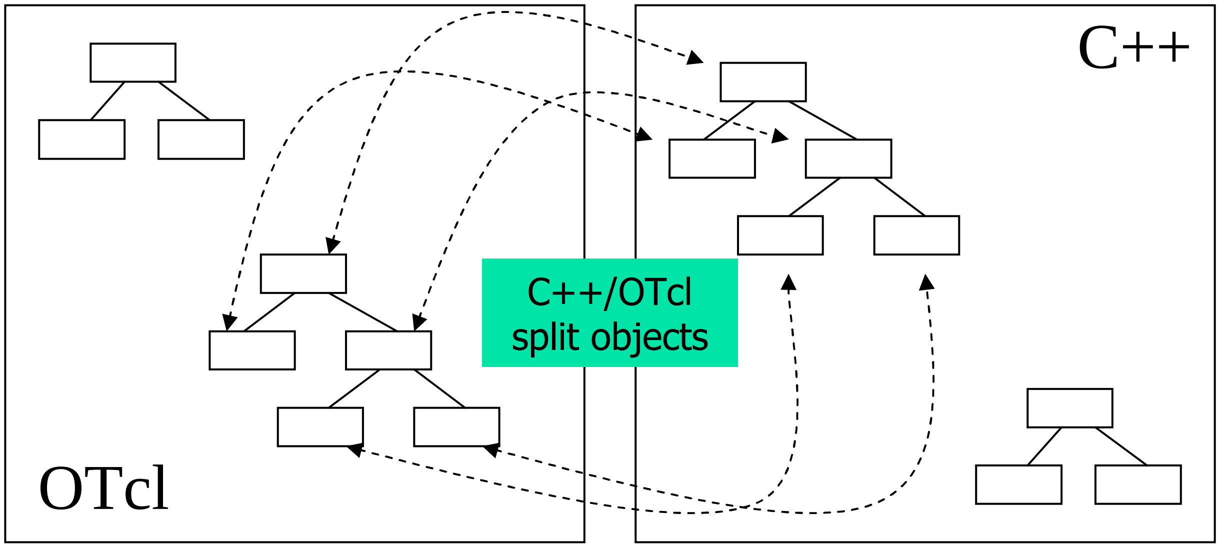
C++ and OTcl separation

- "data" / control separation
  - C++ for "data":
    - per packet processing, core of ns
    - fast to run, detailed, complete control
  - OTcl for control:
    - simulation scenario configurations
    - periodic or triggered action
    - manipulating existing C++ objects
    - fast to write and change

\+ Running vs. writing speed

– Learning and debugging (two languages)

OTcl and C++: the duality



- OTcl (object variant of Tcl) and C++ share class hierarchy
- Tclcl is glue library that makes it easy to share functions and variables

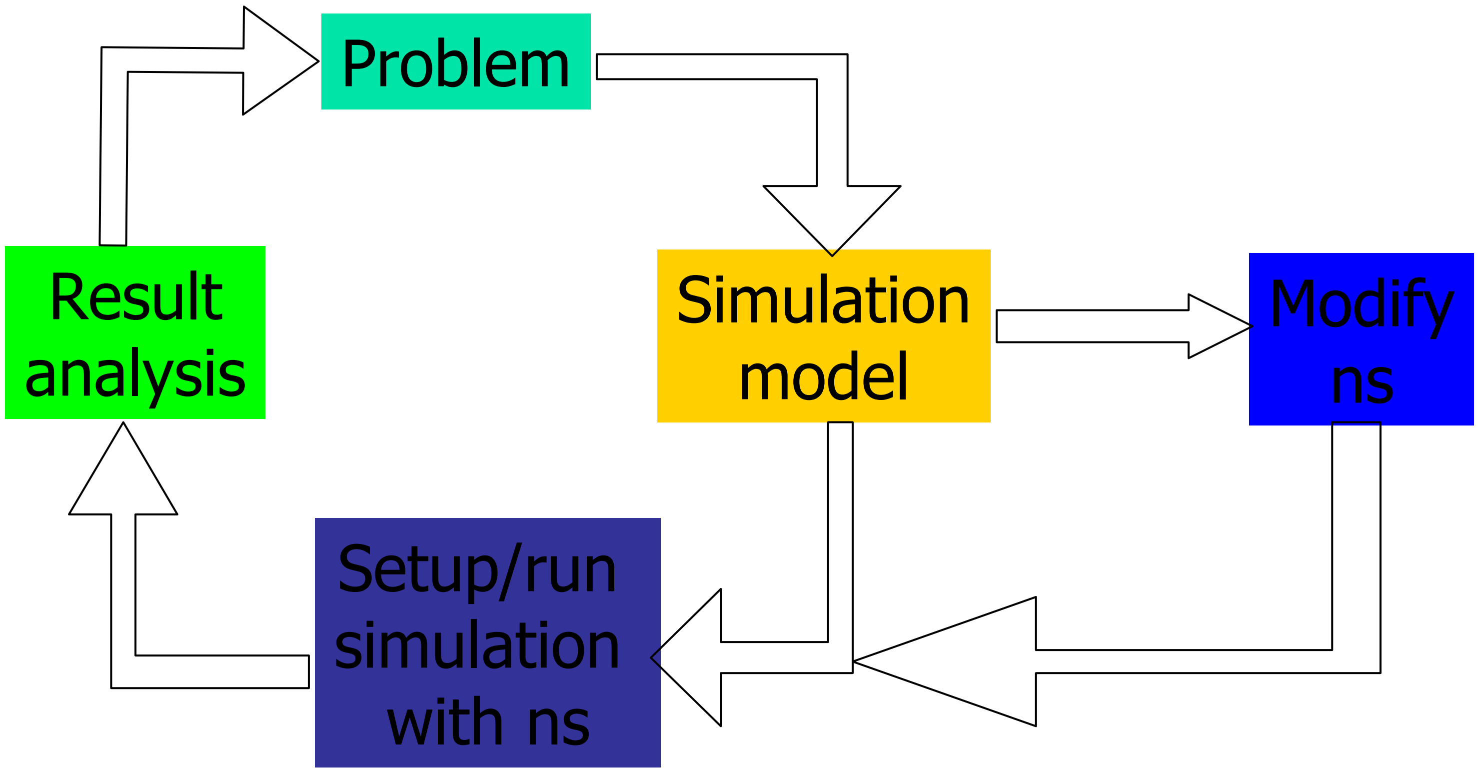
Basic OTcl

```
Class Person
# constructor:
Person instproc init {age} {
    $self instvar age_
    set age_ $age
}
# method:
Person instproc greet {} {
    $self instvar age_
    puts “$age_ years old: How
    are you doing?”
}
```

```
# subclass:
Class Kid -superclass Person
Kid instproc greet {} {
    $self instvar age_
    puts “$age_ years old kid:
    What’s up, dude?”
}

set a [new Person 45]
set b [new Kid 15]
$a greet
$b greet
```

Using ns-2


Problem
Result analysis
Simulation model
Modify ns
Setup/run simulation with ns

ns-2: roadmap

- Basic introduction
- ns-2 fundamentals
- ns-2 programming internal
- Extending ns-2 simulator

ns-2 programming internals

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Creating event scheduler

- Create event scheduler

```
set ns [new Simulator]
```

- Schedule events

```
$ns at <time> <event>
```

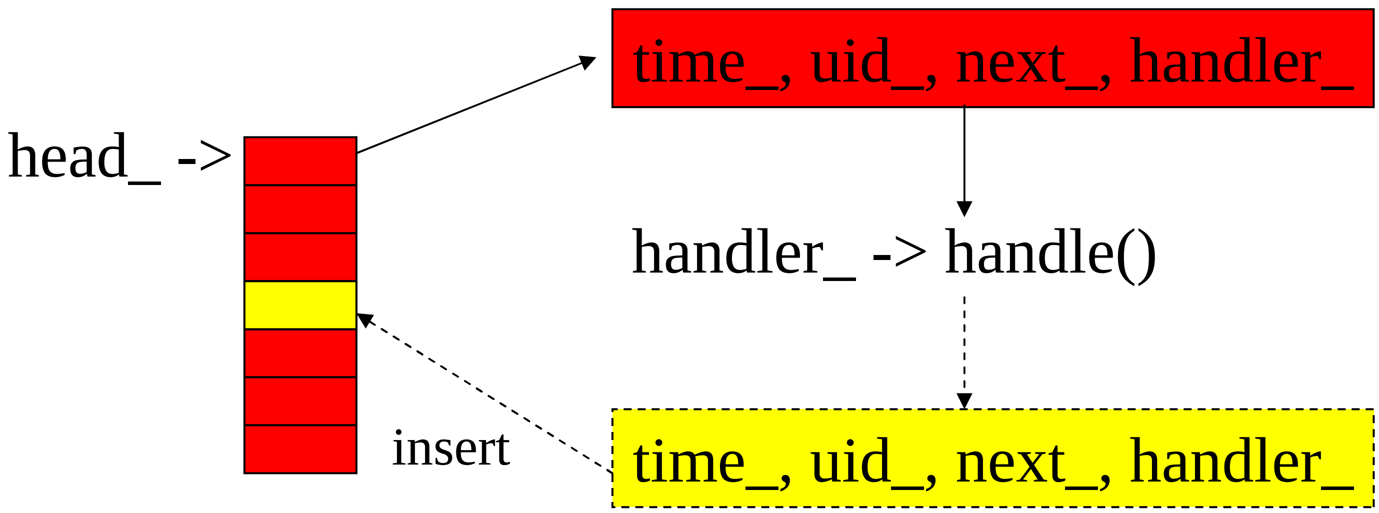
  - `<event>`: any legitimate ns/tcl commands

```
$ns at 5.0 “finish”
```

- Start scheduler

```
$ns run
```

Discrete event scheduler



Hello world: interactive mode

Interactive mode:

```
swallow 71% ns
% set ns [new Simulator]
_o3
% $ns at 1 “puts
  \“Hello World!\””
1
% $ns at 1.5 “exit”
2
% $ns run
Hello World!
swallow 72%
```

Batch mode:

```
simple.tcl
   set ns [new Simulator]
   $ns at 1 “puts
     \“Hello World!\””
   $ns at 1.5 “exit”
   $ns run
swallow 74% ns
  simple.tcl
Hello World!
swallow 75%
```

ns-2 programming internal

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Creating network

- Nodes

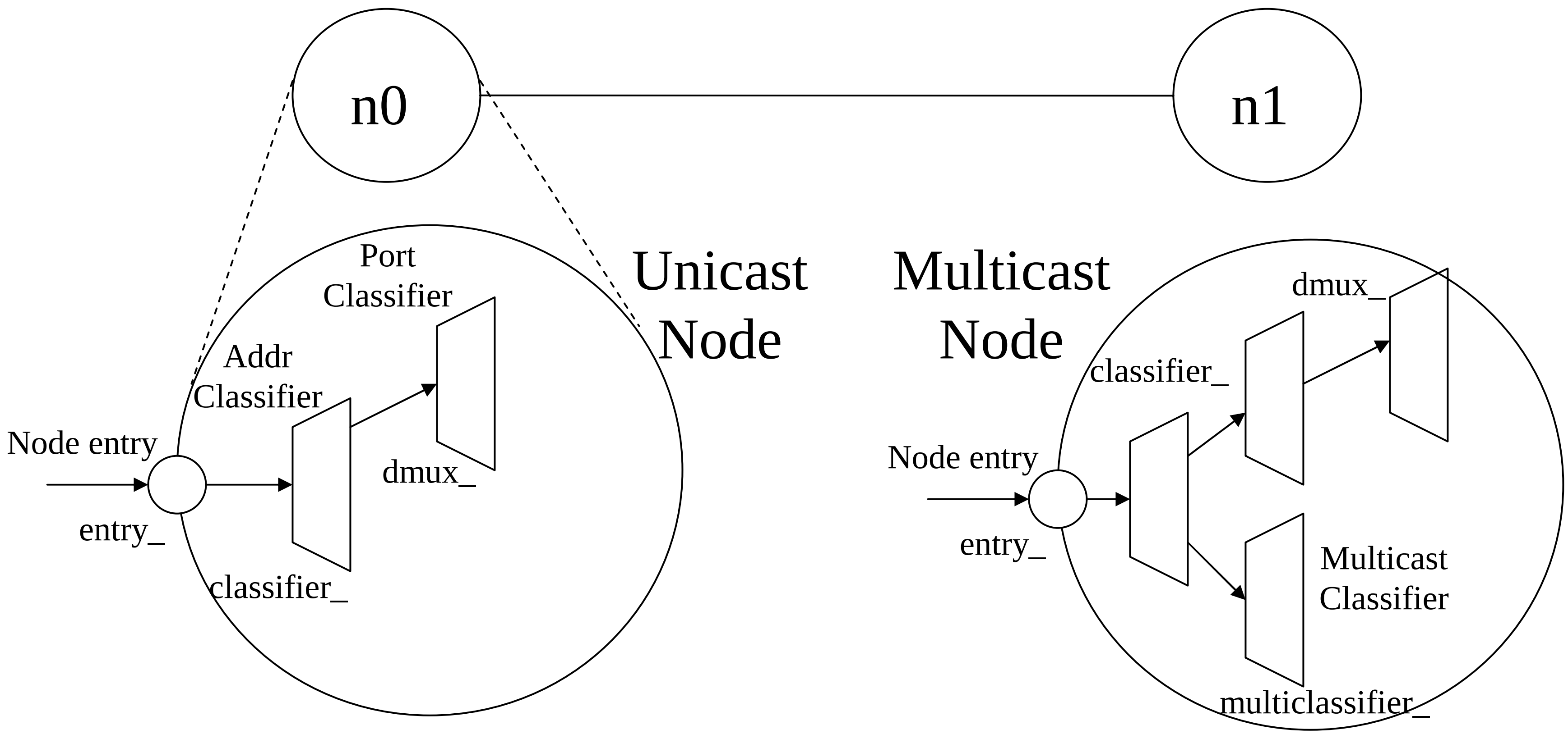
```
set n0 [$ns node]
set n1 [$ns node]
```

- Links and queuing

```
$ns <link_type> $n0 $n1 <bandwidth>
  <delay> <queue_type>
```

  - `<link_type>`: duplex-link, simplex-link
  - `<queue_type>`: DropTail, RED, CBQ, FQ, SFQ, DRR, diffserv RED queues

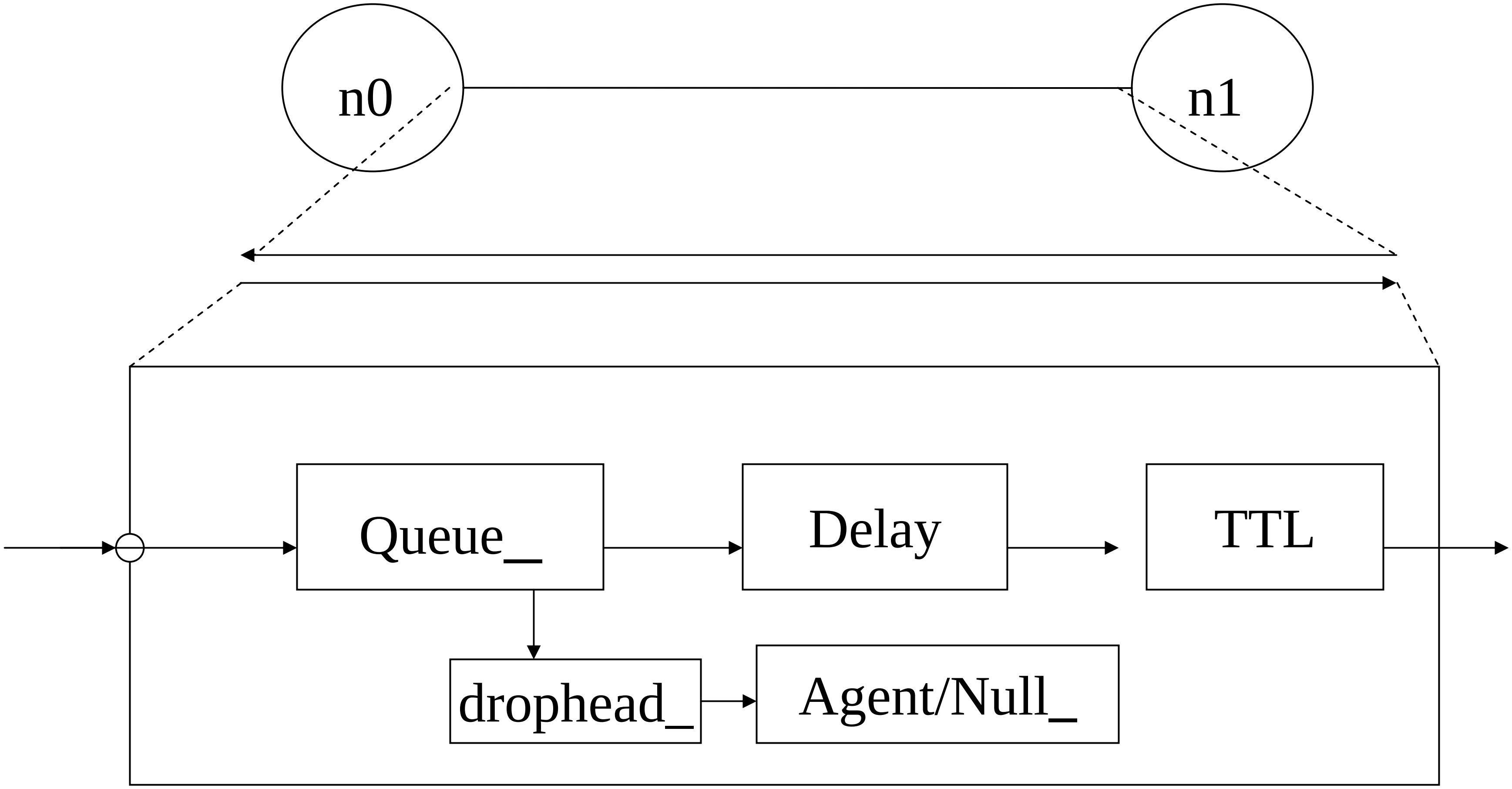
Creating network: node




```
set n0 [ns_ node]
```

```
Set ns [new Simulator –multicast on]
Set n1 [ns node]
```

Creating network: link



ns-2 programming internals

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Tracing and monitoring

- Packet tracing:
  - On all links: `$ns trace-all [open out.tr w]`
  - On one specific link: `$ns trace-queue $n0 $n1$tr`

```
<Event> <time> <from> <to> <pkt> <size> --
  <fid> <src> <dst> <seq> <attr>
```

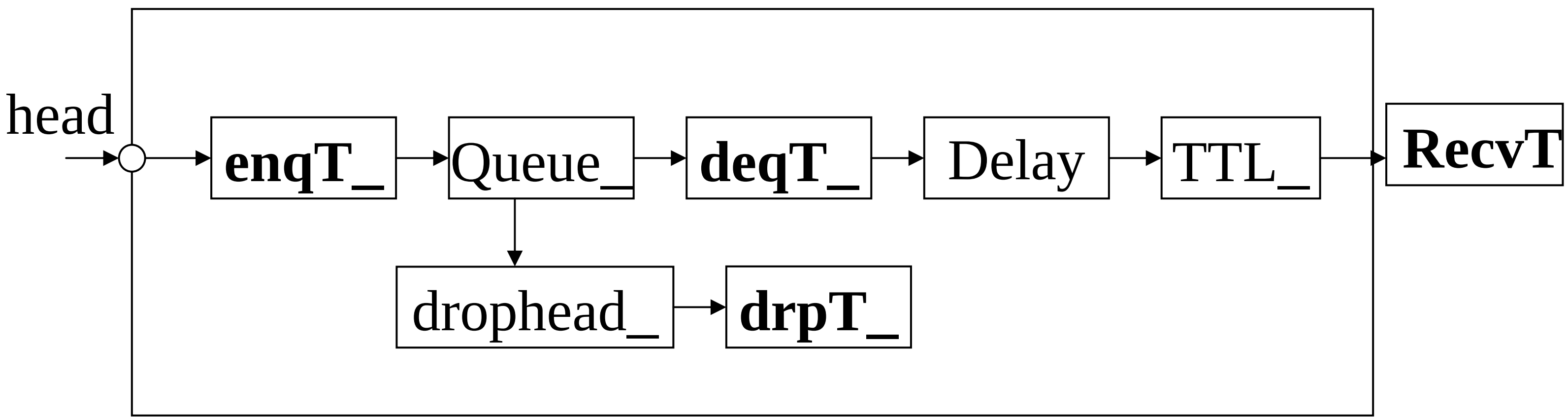
```
+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
- 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
```

- Event tracing (support TCP right now)
  - Record “event” in trace file: `$ns eventtrace-all`

```
E 2.267203 0 4 TCP slow_start 0 210 1
```

Tracing and monitoring

```
$ns trace-all filename
or
$ns namtrace-all filename
```



trace object

Tracing and monitoring

- Queue monitor

```
    set qmon [$ns monitor-queue $n0 $n1
$q_f               $sample_interval]
```

  - Get statistics for a queue

```
    $qmon set pdrops_
```

  - Record statistics to trace file as an option

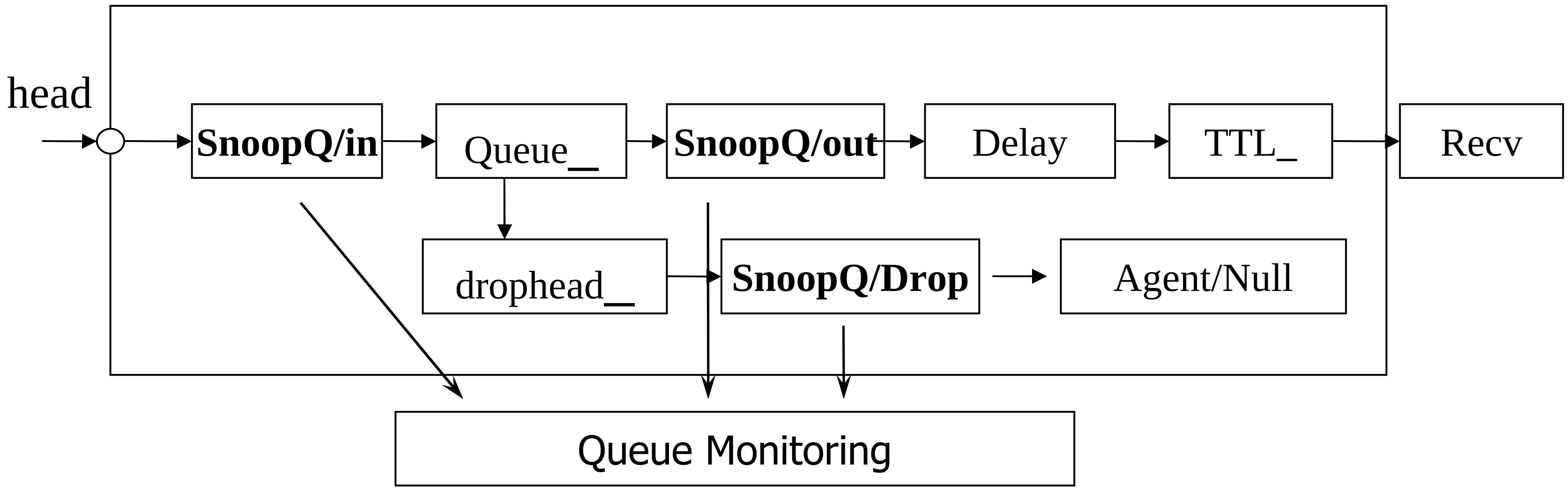
```
    29.000000000000142 0 1 0.0 0.0 4 4 0
1160 1160 0
```

- Flow monitor

```
    set fmon [$ns makeflowmon Fid]
    $ns attach-fmon $slink $fmon
    $fmon set pdrops_
```

Tracing and monitoring

```
$ns monitor-queue node1 node2
$ns at 0.0 qmon trace $filename
```



ns-2 programming internals

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Setup routing

- Unicast

  `$ns rtproto <type>`

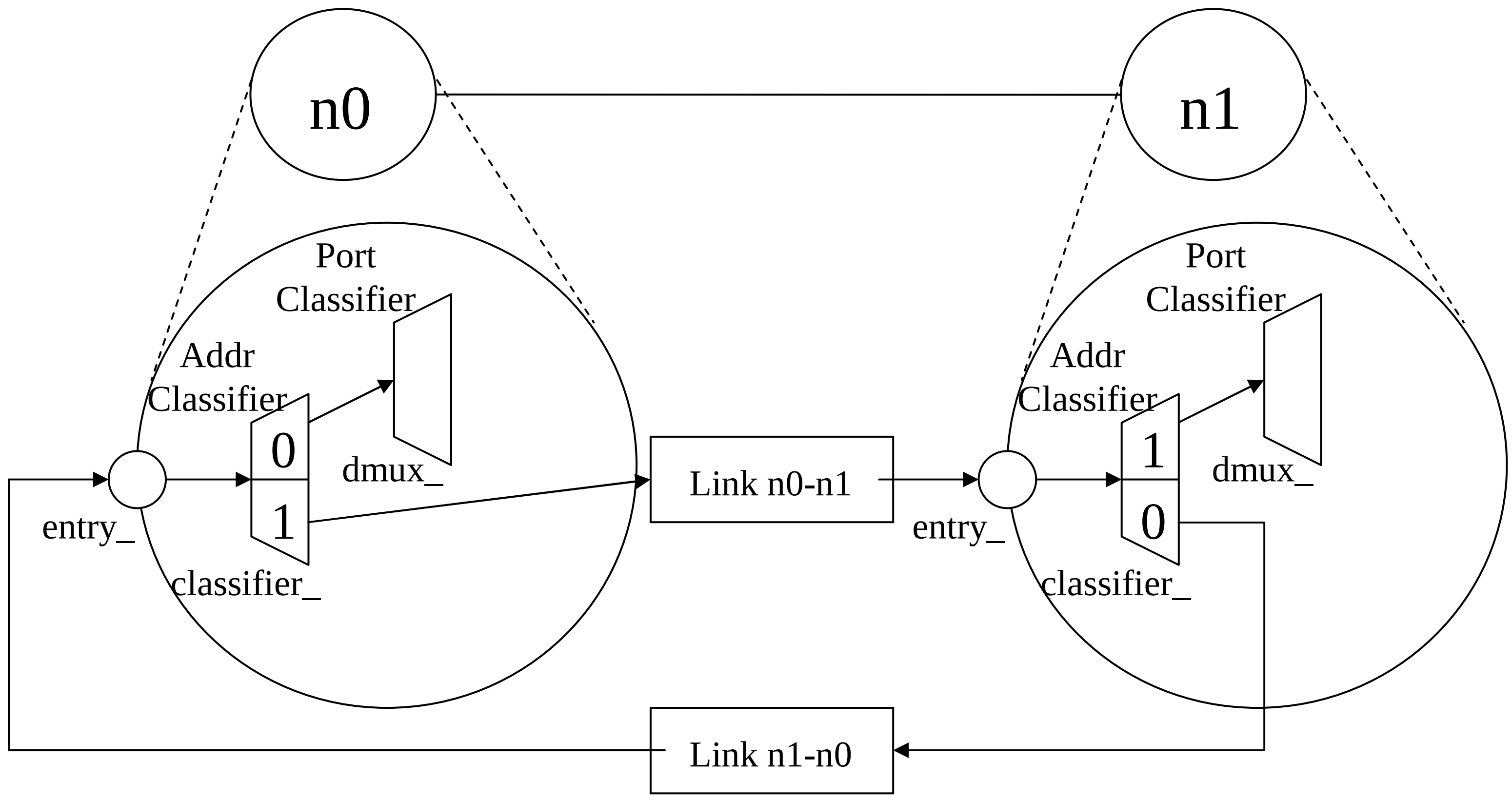
  `<type>`: Static, Session, DV, cost, multi-path
- Multicast

  `$ns multicast` (right after `[new Simulator]` call)

  `$ns mrtproto <type>`

  `<type>`: CtrMcast, DM, ST, BST
- Other types of routing supported: source routing, hierarchical routing

Setup routing



ns-2 programming internals

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Creating connection and traffic

- UDP

```
set udp [new Agent/UDP]
set null [new
  Agent/Null]
$ns attach-agent $n0
  $udp
$ns attach-agent $n1
  $null
$ns connect $udp $null
```

- CBR

```
set src [new
  Application/Traffic/CBR]
```

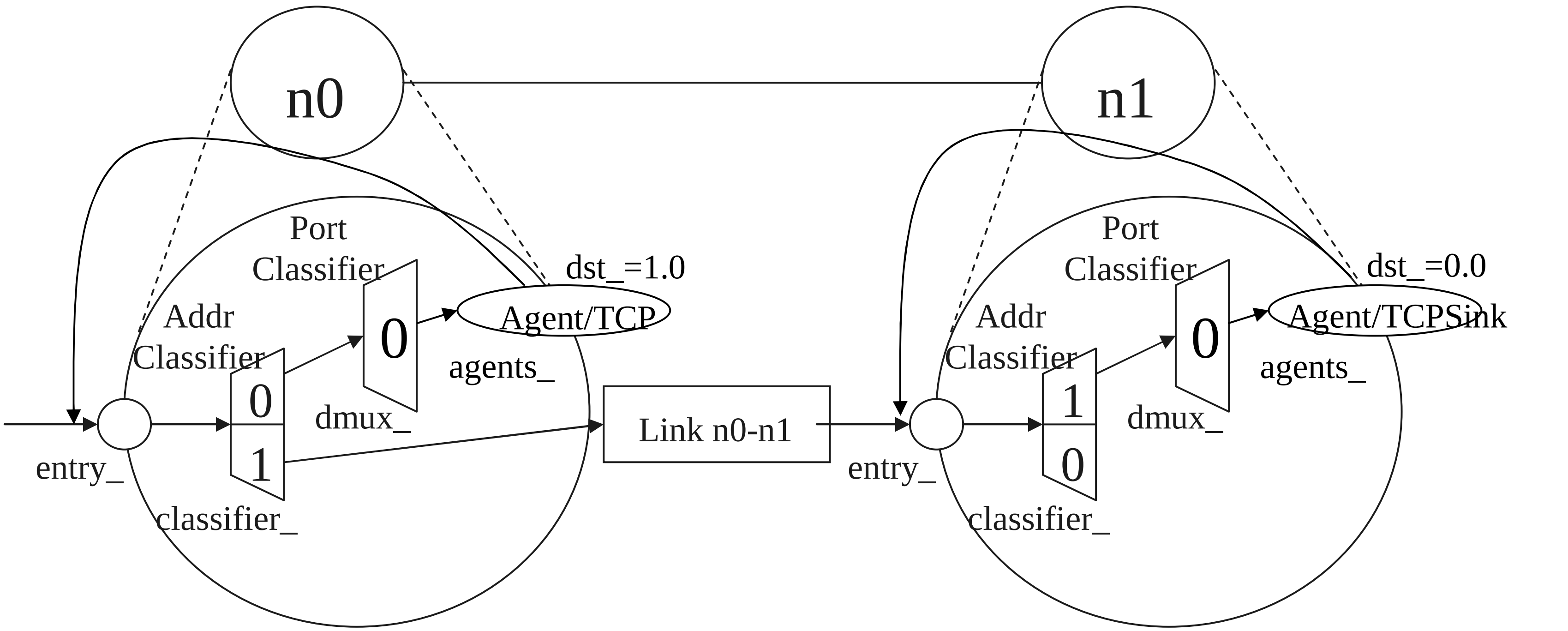
- Exponential

```
set src [new Application/
  Traffic/Exponential]
```

- Pareto on-off

```
set src [new Application/
  Traffic/Pareto]
```

Creating connection and traffic




```
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
```

```
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n1 $tcpsink
```

```
$ns connect $tcp $tcpsink
```

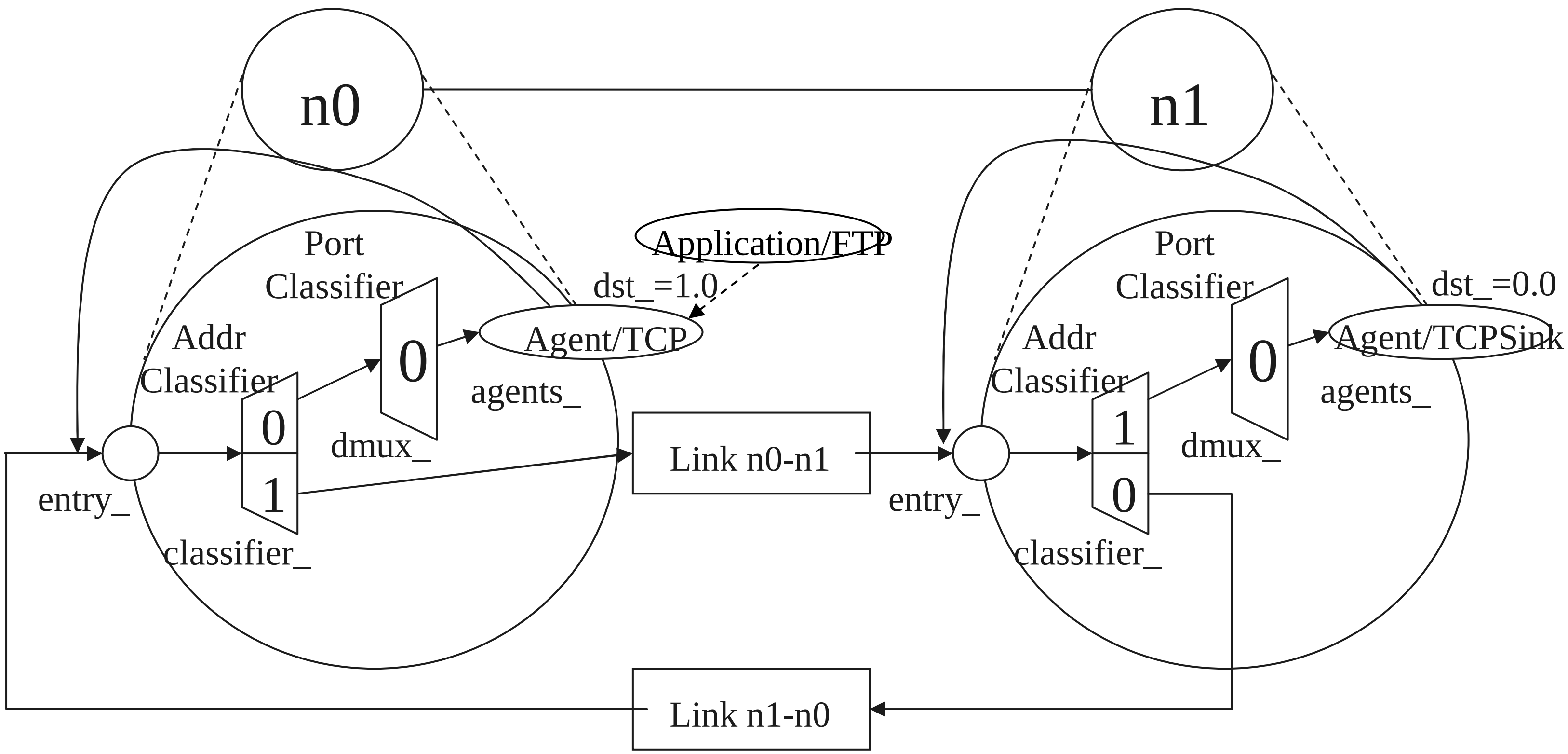
ns-2 programming internals

- Create the event scheduler
- Create network
- Turn on tracing
- Setup routing
- Create connection and traffic
- Transmit application-level data

Application-level simulation

- Features
  - Build on top of existing transport protocol
  - Transmit user data, e.g., HTTP header
- Two different solutions
  - TCP: `Application/TcpApp`
  - UDP: `Agent/Message`

Application-level simulation




```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 "$ftp start"
```

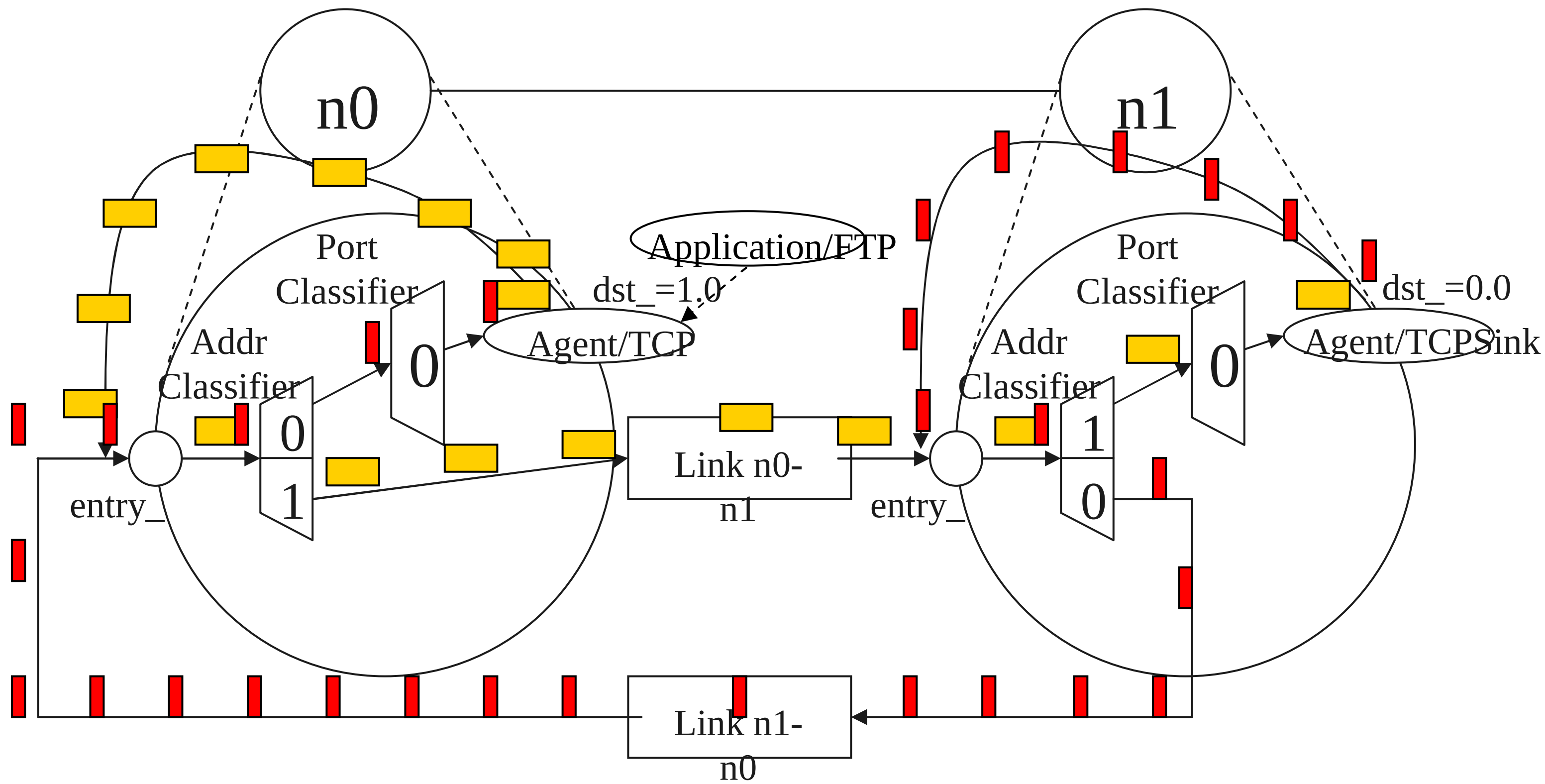
Creating traffic: trace driven simulations

- Trace driven:

```
set tfile [new Tracefile]
$tfile filename <file>
set src [new Application/Traffic/Trace]
$src attach-tracefile $tfile
```

  - <file>:
    - binary format (native)
    - inter-packet time (msec) and packet size (byte)

Packet flow



n0
n1
Application/FTP
dst_=1.0
dst_=0.0
Port
Classifier
Addr
Classifier
Agent/TCP
Agent/TCPSink
0
0
1
1
0
0
entry_
entry_
Link n0-
n1
Link n1-
n0

Compared to real world

- More abstract (much simpler):
  - no IP addresses used, global variables are used instead
  - nodes are connected directly rather than using name lookup/bind/listen/accept
- Easy to change implementation:

```
Set tsrc2 [new agent/TCP/Newreno]
Set tsrc3 [new agent/TCP/Vegas]
```

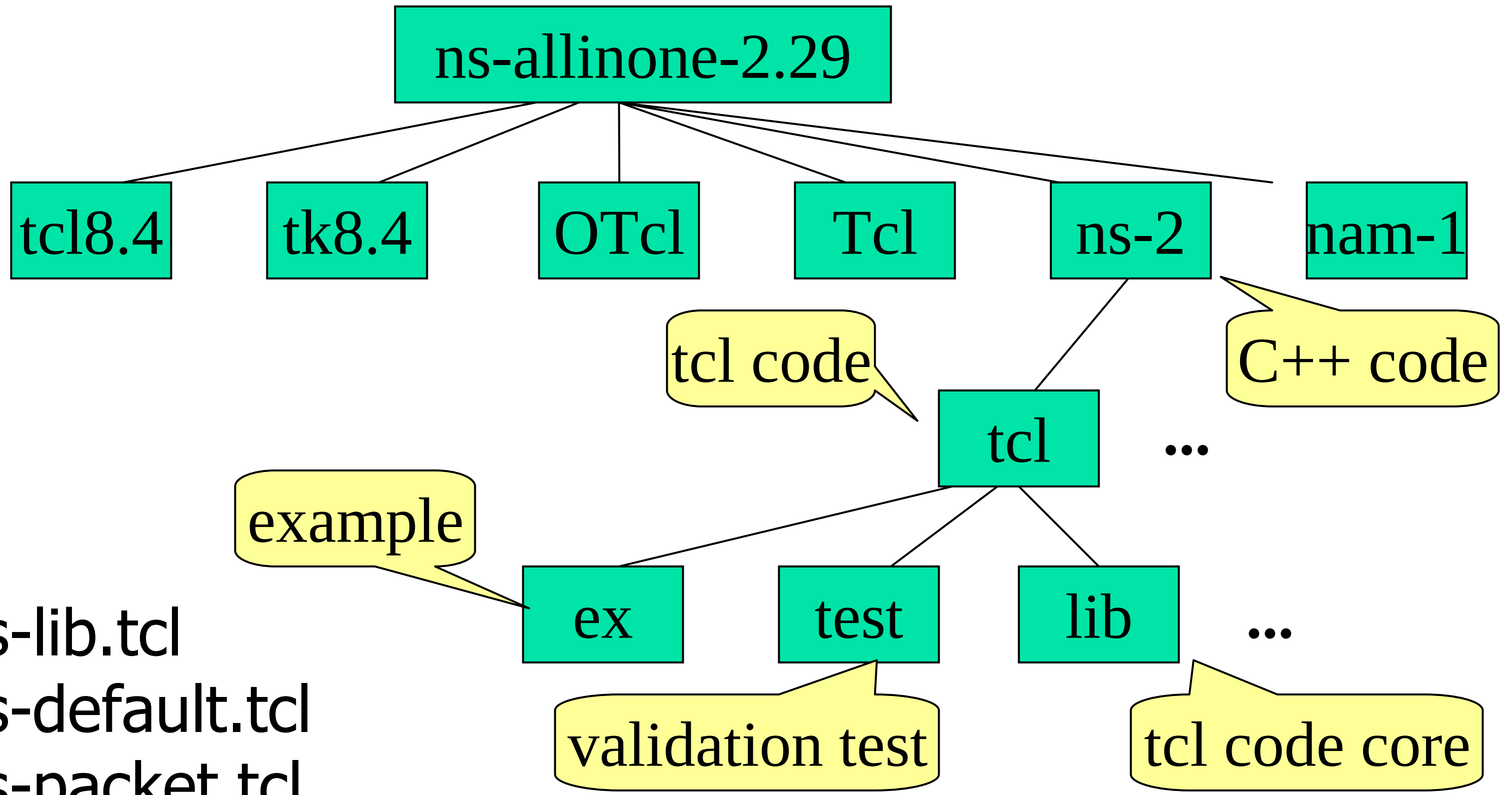
Summary: generic script structure

```
set ns [new Simulator]
# [Turn on tracing]
# Create topology
# Setup packet loss, link dynamics
# Create routing agents
# Create:
#   - multicast groups
#   - protocol agents
#   - application and/or setup traffic sources
# Post-processing procs
# Start simulation
```

Roadmap

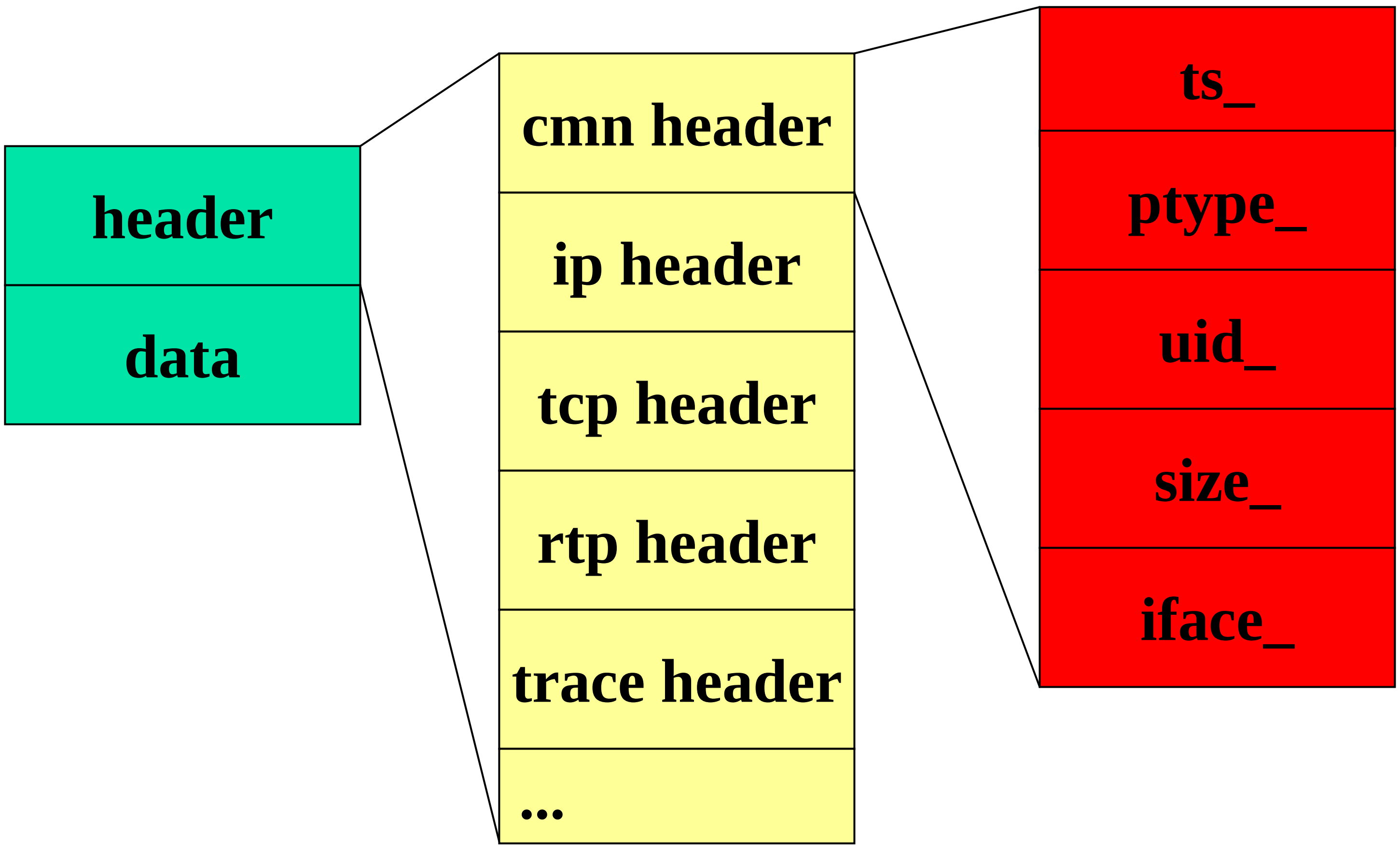
- Basic introduction
- ns fundamentals
- ns programming internal
- Extending ns-2 simulator

ns-2 directory structure

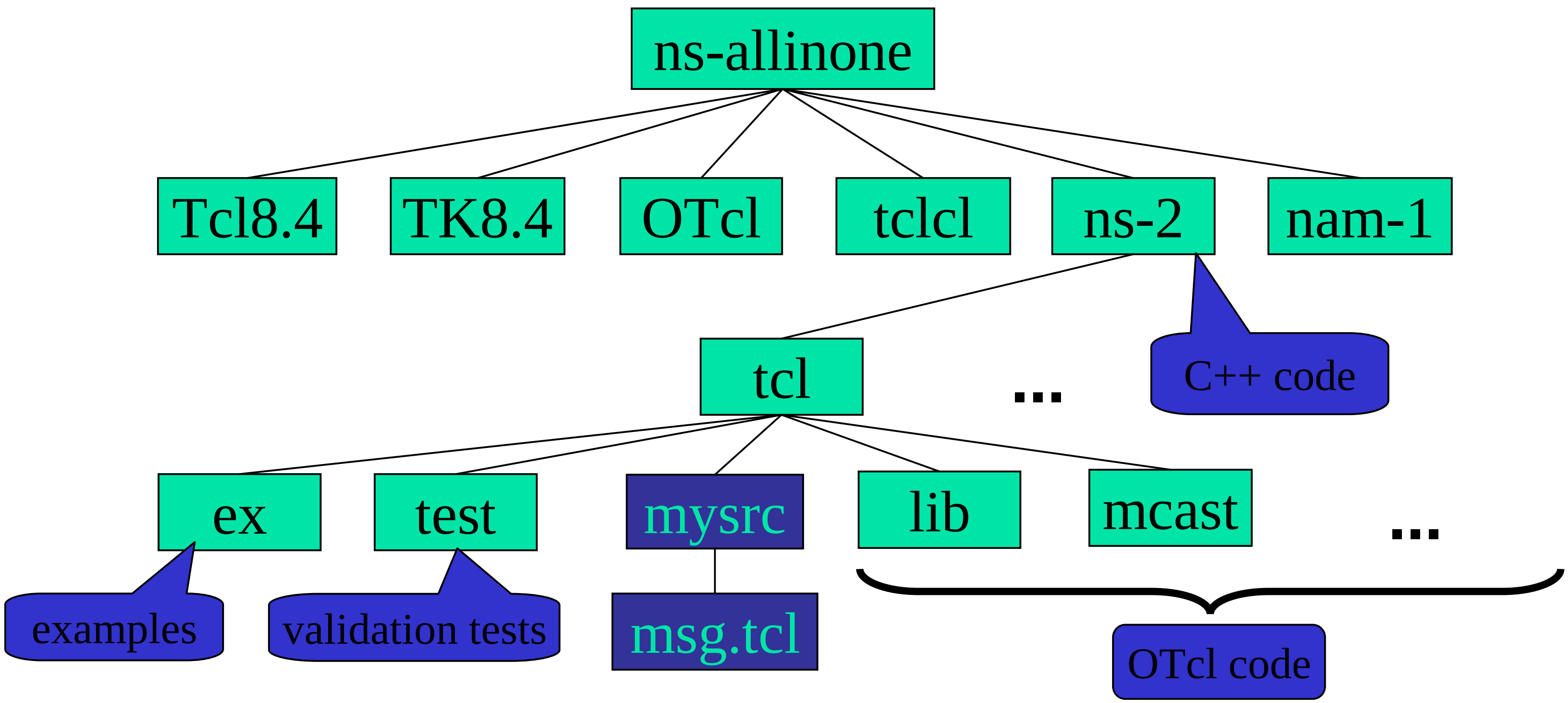


- ns-lib.tcl
- ns-default.tcl
- ns-packet.tcl

Packet format



Add your tcl changes into ns-2



Add your tcl changes into ns-2

- tcl/lib/ns-lib.tcl

```
Class Simulator
…
source ../mysrc/msg.tcl
```

- Makefile

```
 NS_TCL_LIB = \
   tcl/mysrc/msg.tcl \
   …
```

  - Or: change Makefile.in, `make distclean`, then `./configure --enable-debug`, `make depend && make`

Extending ns-2 in C++

- Modifying code
  - make depend
  - recompile
- Adding code in new files
  - change Makefile
  - make depend
  - recompile

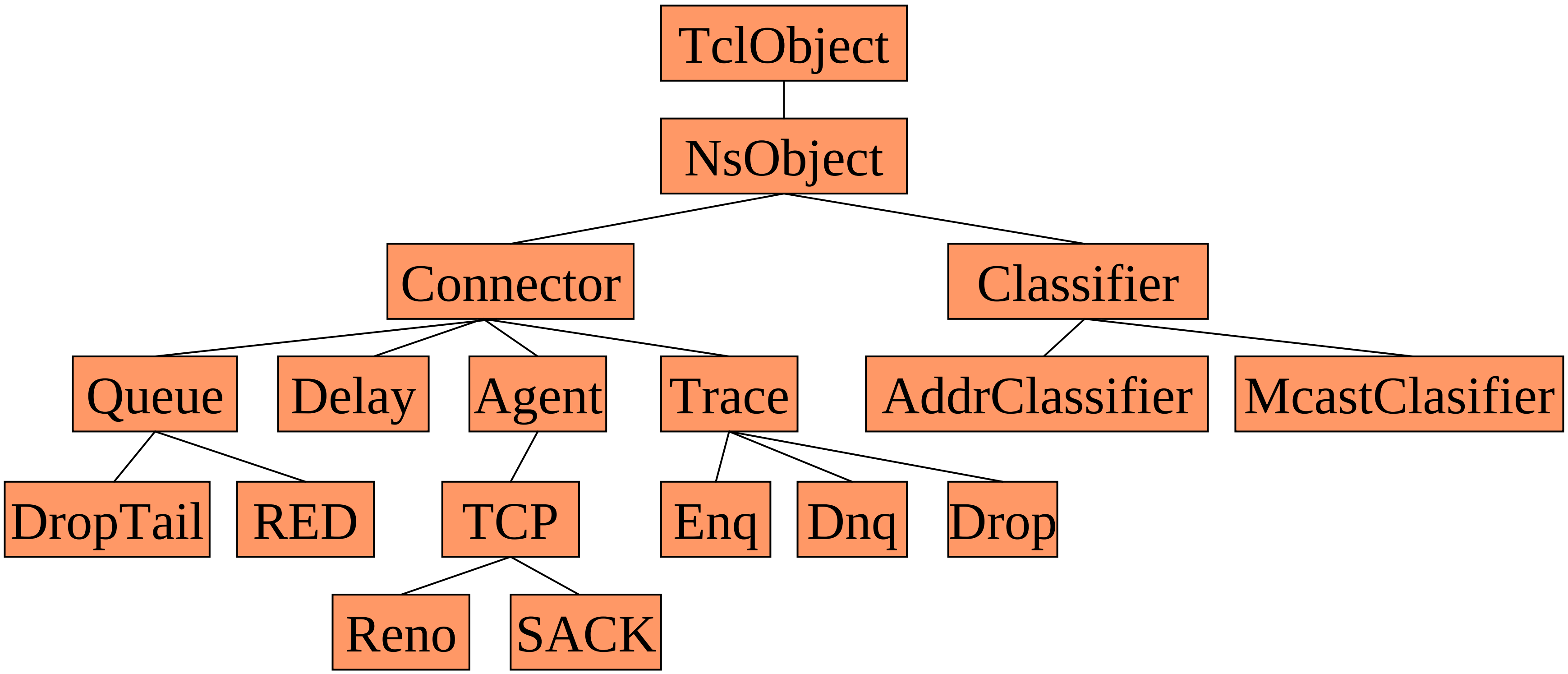
Creating new components

- Guidelines
- Inheritance Hierarchy
- C++ and OTcl Interface
- Debugging

Guidelines

- Decide its inheritance structure
- Create the class and fill in the virtual functions
- Define OTcl linkage functions
- Write the necessary OTcl code to access your agent

Class hierarchy (partial)



C++ and OTcl linkage

- TclClass
- TclObject: `bind()` method
- TclObject: `command()` method

Object granularity tips

- Functionality
  - per-packet processing → C++
  - hooks, frequently changing code → OTcl
- Data management
  - complex/large data structure → C++
  - runtime configuration variables → OTcl

Memory conservation tips

- Remove unused packet headers
- Avoid `trace-all`
- Use arrays for a sequence of variables:
  - instead of `n$i`, say `n($i)`
- Avoid OTcl temporary variables
- Use dynamic binding:
  - `delay_bind()` instead of `bind()`
- See tips for running large simulations in ns at: www.isi.edu/ns/nsnam/ns-largesim.html
- Not necessary until >100 nodes with complex models are used

Debugging

- `printf()` in C++ and `puts ""` in Tcl
- gdb
- tcl debugger
  - http://expect.nist.gov/tcl-debug/
  - place debug 1 at the appropriate location
  - trap to debugger from the script
  - single stepping through lines of codes
  - examine data and code using Tcl-like commands

Implementation tips

- ns-2 TCP model in ns-2 does not allow payload in packets:
  - number of bytes are specified
  - a workaround is available at "NS by example"
- ns-2 UDP model allows payload inside the packet header:
  - UDP model is a good starting point for most protocol improvement projects
- Look for similar projects' or modules' code and modify:
  - starting from scratch is difficult and prone to errors
  - be very careful with pointers and dynamic arrays:
    - faults are hard to debug due to C++/OTcl duality

ns-2 to nam interface

- Color
- Node manipulation
- Link manipulation
- Topology layout
- Protocol state
- Miscellaneous

nam interface: color

- Color mapping

```
$ns color 40 red
$ns color 41 blue
$ns color 42 chocolate
```

- Color ↔ flow id association

```
$tcp0 set fid_ 40  ;# red packets
$tcp1 set fid_ 41  ;# blue packets
```

nam interface: nodes

- Color

```
$node color red
```

- Shape (can't be changed after sim starts)

```
$node shape box    ;# circle, box, hexagon
```

- Marks (concentric "shapes")

```
$ns at 1.0 "$n0 add-mark m0 blue box"
$ns at 2.0 "$n0 delete-mark m0"
```

- Label (single string)

```
$ns at 1.1 "$n0 label \"web cache 0\""
```

nam interface: links

- Color

```
$ns duplex-link-op $n0 $n1 color "green"
```

- Label

```
$ns duplex-link-op $n0 $n1 label "abcde"
```

- Dynamics (automatically handled)

```
$ns rtmodel Deterministic {2.0 0.9 0.1} $n0
  $n1
```

- Asymmetric links not allowed

nam interface: topology

- "Manual" layout: specify everything

```
$ns duplex-link-op $n(0) $n(1) orient
  right
$ns duplex-link-op $n(1) $n(2) orient
  right
$ns duplex-link-op $n(2) $n(3) orient
  right
$ns duplex-link-op $n(3) $n(4) orient
  60deg
```

- If nodes are overlapped → use automatic layout

nam interface: miscellaneous

- Annotation:
  - add textual explanation to your simulation

```
$ns at 3.5 "$ns trace-annotate \“packet
   drop\"“
```

- Set animation rate

```
$ns at 0.0 "$ns set-animation-rate 0.1ms"
```

Help and resources

- ns and nam build questions
  - http://www.isi.edu/nsnam/ns/ns-build.html
- ns mailing list: ns-users@isi.edu
- ns manual and tutorial (in distribution)
- TCL: http://dev.scriptics.com/scripting
- OTcl tutorial (in distribution): ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/tutorial.html
- ns by example: http://nile.wpi.edu/NS
- ns simulator for beginners http://www-sop.inria.fr/maestro/personnel/Eltan.Altman/COURS-NS/n3.pdf

Roadmap

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Summary

BGP case study: roadmap

- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

BGP case study: roadmap

- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

Introduction

- Internet routing
  - Autonomous Systems
  - IGP: Interior Gateway Protocol (Intra-domain)
  - EGP: Exterior Gateway Protocol (Inter-domain)
- Border Gateway Protocol (BGP) weaknesses
  - routing instability
  - inefficient routing
  - scalability issues
- Employed approaches
  - empirical measurements
  - theoretical analysis
  - simulations

Internet routing

- Internet is organized as a collection of interconnected Autonomous Systems (AS)
- Routing in the Internet is performed on two levels
  - IGP: Interior Gateway Protocol (Intra-domain)
    - OSPF, IS-IS, EIGRP, RIP
  - EGP: Exterior Gateway Protocol (Inter-domain)
    - BGP

BGP weaknesses

- Poor integrity
  - vulnerable to malicious attacks and misconfiguration
- Slow convergence
  - up to tens of minutes
- Divergence
  - conflicts of routing policies can cause BGP to diverge, resulting in persistent route oscillations

Approaches

- Empirical measurements
  - expensive set-up
  - inflexible
- Theoretical analysis
  - highly simplified
  - inadequate in practical scenarios
- Simulations
  - full control over the system and flexible
  - cost effective
  - controlled experiments

BGSP case study: roadmap

- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

Background

- BGP version 4
- Network simulator ns-2
- BGP implementation in SSFNet
- Related work

BGP version 4

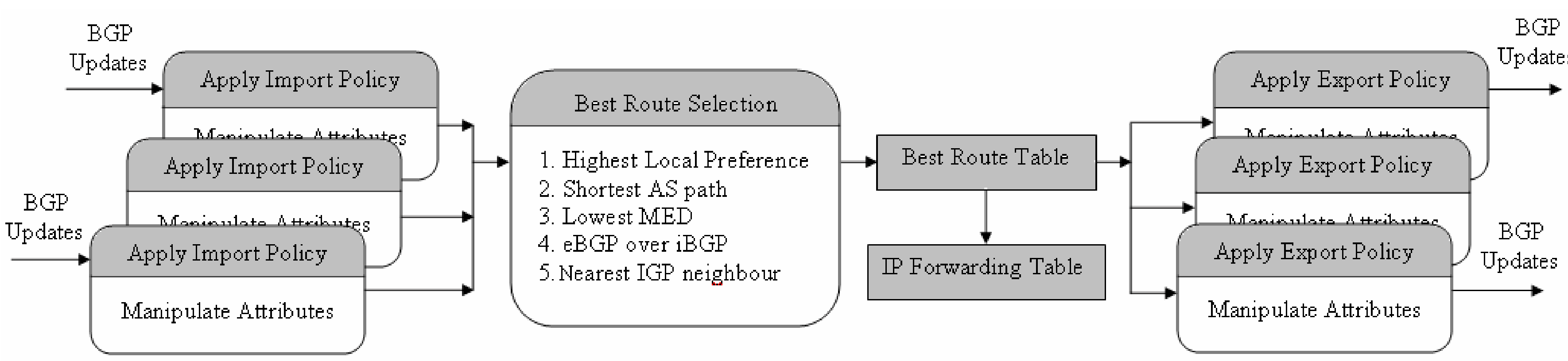
- RFC 1771, "A Border Gateway Protocol 4", March 1995
- The *de facto* inter-domain routing protocol of the Internet
- Path vector protocol
- Incremental
- Relies on TCP

Four types of BGP messages

- Open: establish a peering session
- Keep alive: handshake at regular intervals
- Notification: report errors, shut down a peer session
- Update: announce new routes or withdraw previously announced routes
  - advertisement
    - destination prefix
    - route attributes (local preference, AS path)

Route processing

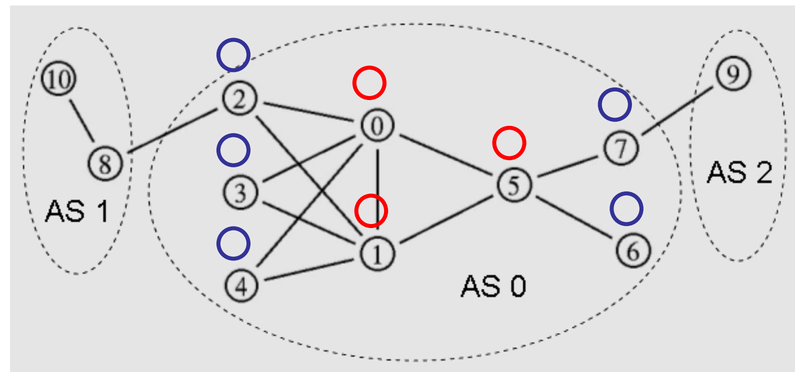
- Apply import policy
- Select a best route
- Install the best route
- Apply export policy and send out updates



MED: Multiple Exit Discriminator

BGP route reflection

- Two types of BGP peer connections:
  - external BGP (eBGP) connection
  - internal BGP (iBGP) connection
- BGP routers within an AS are required to be fully meshed with iBGP connections
- Route reflection provides one way to address the scalability issue of iBGP



- reflector ○
- client ○

Network simulator: ns-2

- One of the most popular network simulators
- Object oriented
  - written in C++ and OTcl
- Substantial support for TCP, routing, and multicast protocols
- Graphical animator: nam

SSF.OS.BGP4: BGP implementation in SSFNet

- Scalable Simulation Framework Network Models (SSFNet) is a Java-based simulator
- SSF.OS.BGP4 is developed and maintained by Brian J. Premore from Dartmouth College
- We implemented a BGP-4 model (ns-BGP) in ns-2 by porting the BGP implementation from SSFNet

Related work

- OPNET BGP model
  - the difference between OPNET and ns-2
- BGP daemon of GNU Zebra
  - object oriented paradigm
- J-Sim BGP model
  - also ported from SSFNet

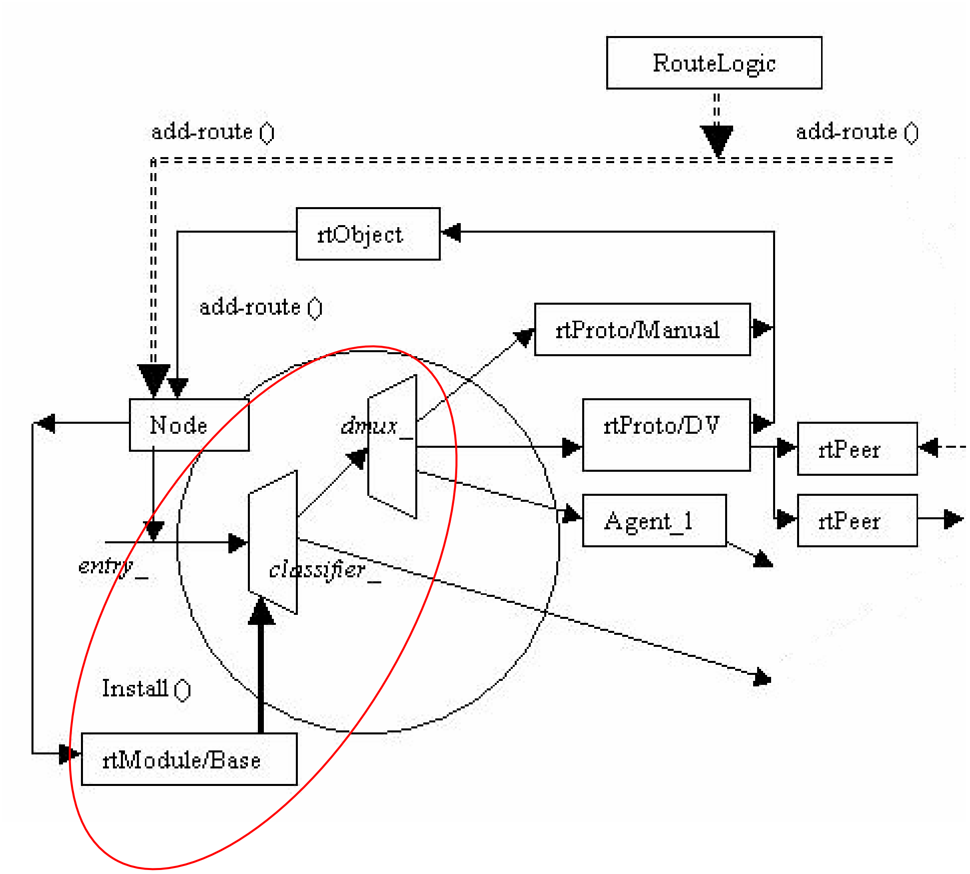
BGP case study: roadmap

- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

ns-2 unicast routing structure

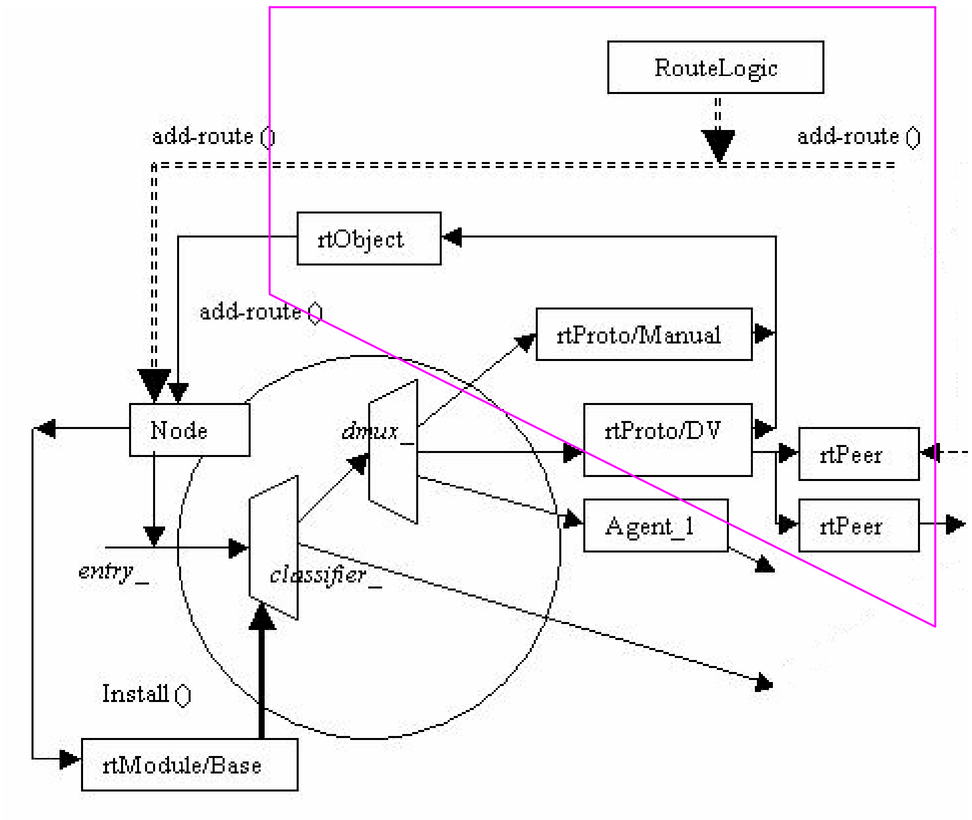
- Forwarding plane:
  - classify and forward packets
- Control plane:
  - routing info exchange, route computation, routing table creation and maintenance

Forwarding plane



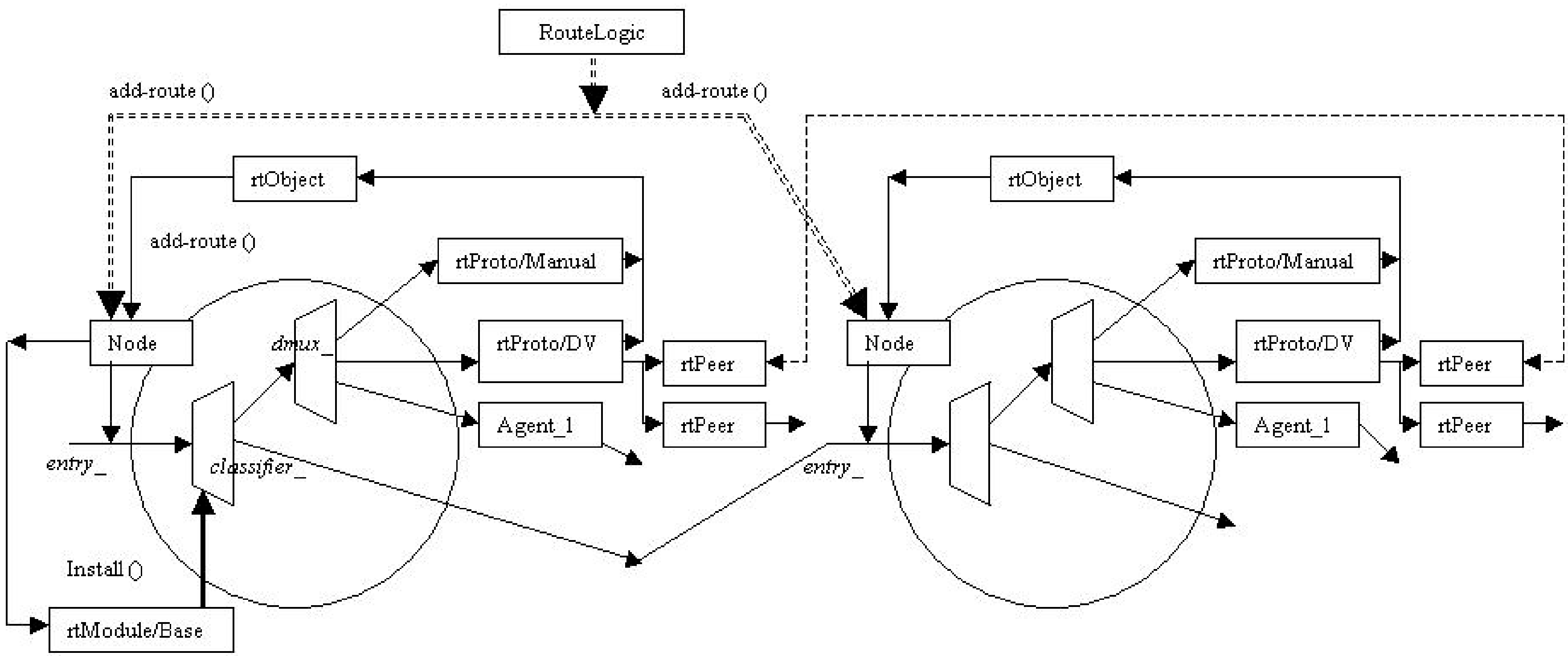
- Classifier (classifer_):
  - delivers the incoming packets either to the correct agent or to the outgoing link
- Routing Module (rtModule):
  - manages a node's classifier and provides an interface to the control plane

Control plane



- Route logic (RouteLogic):
  - the centrally created routing table
- Routing protocol (rtProto):
  - manual, DV, LS
  - implements specified routing algorithm
- Route peer (rtPeer):
  - stores the metric and preference for each route it advertised
- Route object (rtObject):
  - a coordinator for the node's routing instances

ns-2 routing structure diagram



Modifications to ns-2

- No socket layer in current ns-2:
  - Solution: we ported to ns-2 TcpSocket - the socket layer implementation of SSFNet
- Simplified packet transmission:
  - Solution: we modified FullTcpAgent, the TCP agent for TcpSocket to support data transmission
- No support for IPv4 addressing and packet forwarding schemes:
  - Solution: we created a new address classifier IPv4Classifier

No socket layer in current ns-2

- BGP is built on top of TCP layer
- Without a socket layer, BGP has to monitor the status of the TCP three-way handshake and connection termination process
- Solution: we ported to ns-2 TcpSocket, the socket layer implementation of SSFNet

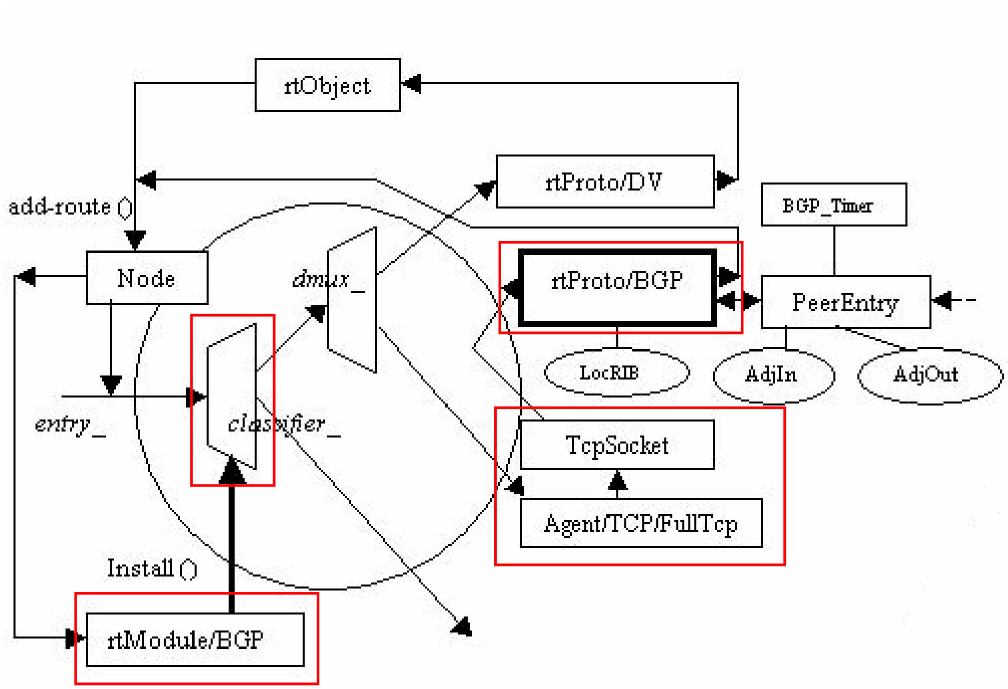
Simplified packet transmission

- Only packet headers (without data) are transmitted by the current TCP agent
- In order to exchange routing information, BGP need to transmit the whole packet
- Solution: we modified FullTcpAgent, the TCP agent for TcpSocket to support data transmission

No support for IPv4 addressing and packet forwarding schemes

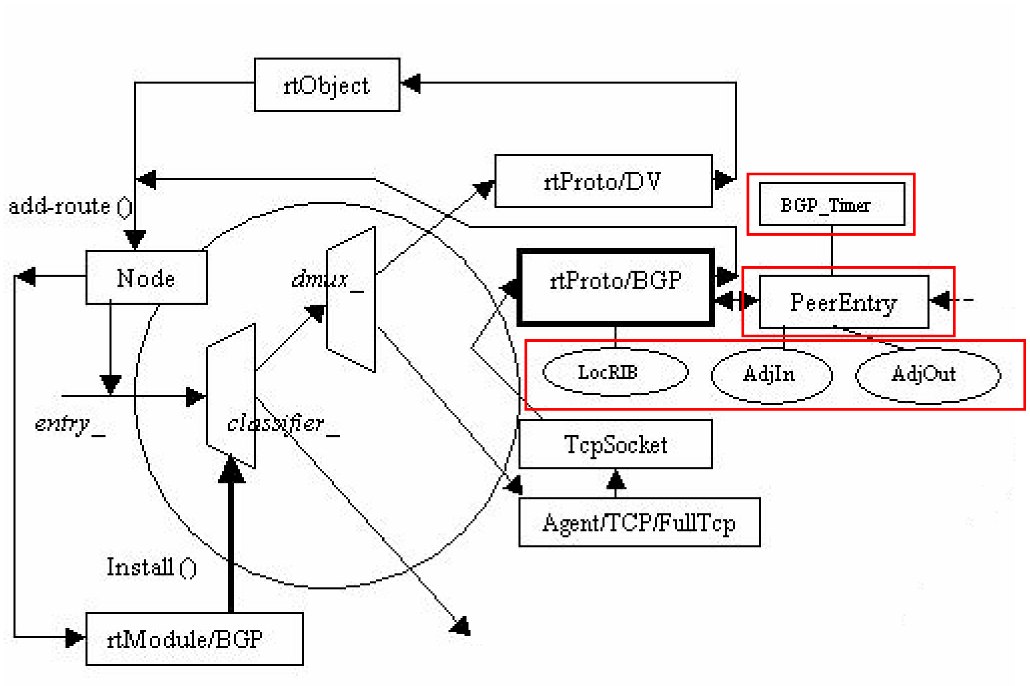
- BGP exchange routing information of IPv4 address blocks, called prefixes
- No support for IPv4 addressing and packet forwarding schemes in current ns-2.
- Solution: we created a new address classifier IPv4Classifier

ns-BGP unicast routing structure



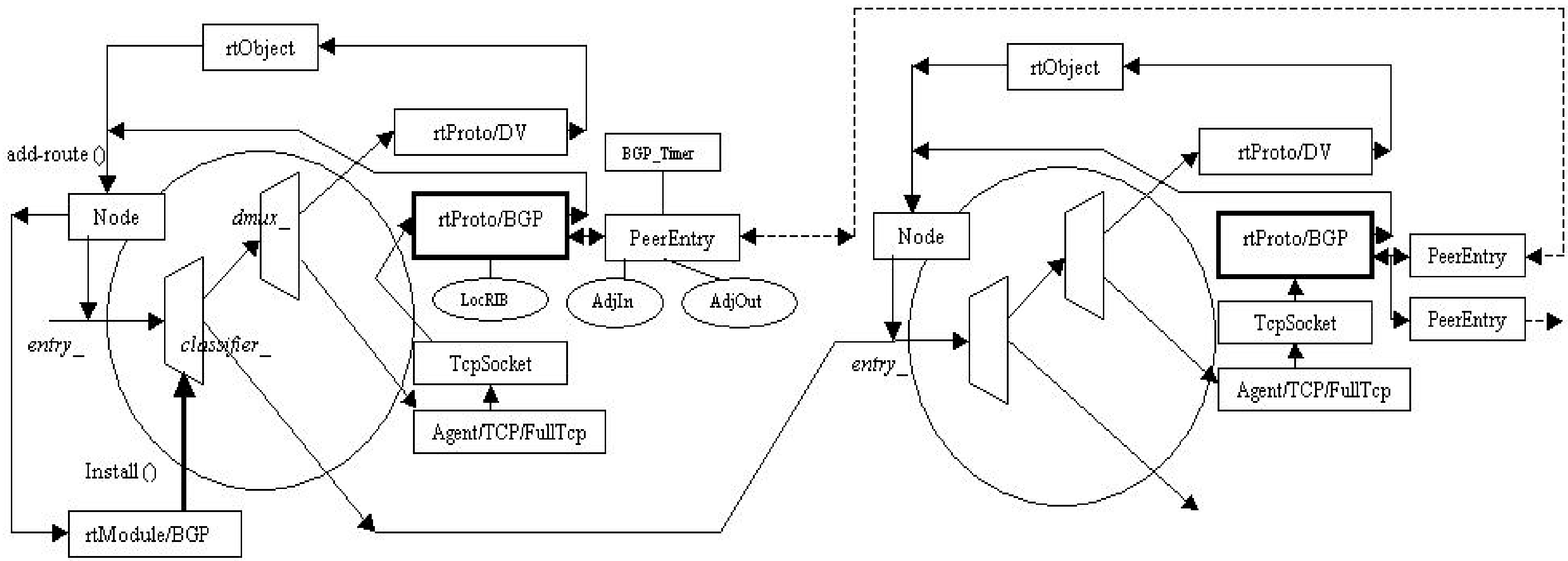
- IPv4Classifier (*classfier_*)
- BGP routing model (rtModule/BGP):
  - manages the IPv4Classifier
- TcpSocket:
  - encapsulating the TCP services into a socket interface
- BGP routing protocol (rtProto/BGP):
  - performs BGP operations

ns-BGP unicast routing structure



- BGP peer (PeerEntry):
  - establishes and closes a peer session, exchanges messages with a peer
- BGP routing tables (LocRIB, AdjIn, and AdjOut):
  - correspond to the BGP Routing Information Base (RIB): Loc-RIB, Adj-RIB-In, and Adj-RIB-Out
- BGP Timer (BGP_Timer):
  - provides supports for the BGP timing features (timers)

ns-BGP unicast routing structure



Supported features

- Implemented all required features in RFC 1771
- Experimental features:
  - sender-side loop detection
  - withdrawal rate limiting
  - per-peer and per-destination rate limiting
- Optional features:
  - Multiple Exit Discriminator (MED)
  - aggregator
  - community
  - originator ID
  - cluster list

BGSP case study: roadmap

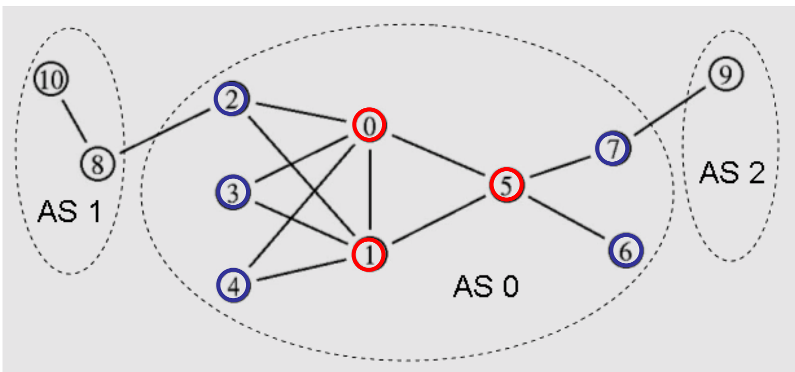
- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

Validation test

- Route reflection:
  - validates the behavior of multiple reflectors inside a BGP cluster

Route reflection validation test

- Network topology



- The network contains three ASs:
  - AS 0 has eight nodes (0 to 7), with IP addresses 10.0.0.0 - 10.0.7.0
  - AS 1 has two nodes (8 and 10), with IP addresses 10.1.8.0 and 10.1.10.0
  - AS 2 has a single node (9), with IP address 10.2.9.0
- Addressing scheme:

  10.(AS number).(node number).1

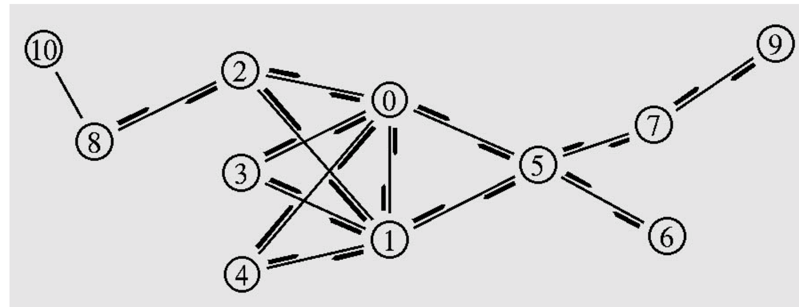
- BGP configuration:
  - AS 0 contains two clusters (0 and 1).
    - cluster 0 (nodes 0 – 4) contains 2 reflectors: nodes 0 and 1, with nodes 2, 3, and 4 as their clients
    - cluster 1 (nodes 5 -7) has one reflector (node 5), with nodes 6 and 7 as its clients
    - The three reflectors (nodes 0, 1, and 5) are fully connected via iBGP connections
  - eBGP connections:
    - nodes 2 and 8
    - nodes 7 and 9

Traffic source and event scheduling

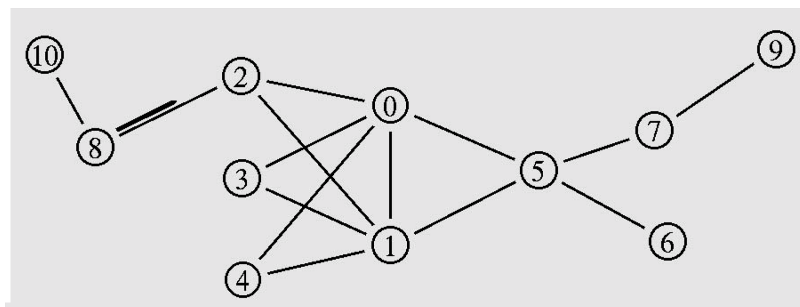
- Traffic source:
  - attached to node 4
  - constant bit rate (CBR)
  - transport protocol: UDP
  - sends segments of 20 bytes/ms to node 10 (10.1.10.1).
- Event scheduling:
  - traffic source begins sending at 0.23 s and stops at 20.0 s
  - 0.25 s: node 8 sends a route advertisement for network 10.1.10.0/24 that is within its AS (AS 1)
  - 0.35 s: node 9 sends a route advertisement for network 10.2.9.0/24
  - 39.0 s: displays all routing tables for BGP agents
  - 40.0 s: the simulation terminates

Simulation results: nam snapshots (1)

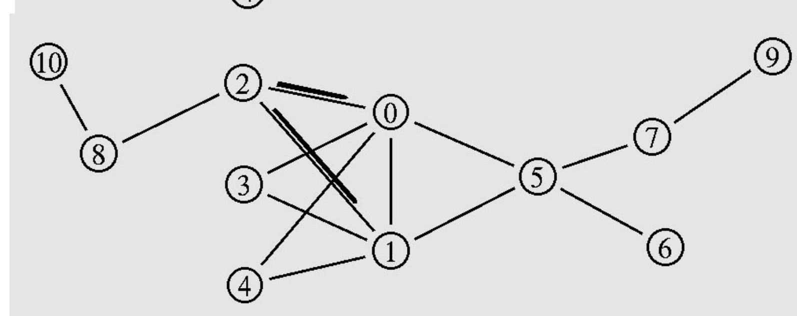
- 0.0503 s, TCP SYN segments are exchanged



- 0.2505 s, node 8 originates an update message for network 10.1.10.0/24

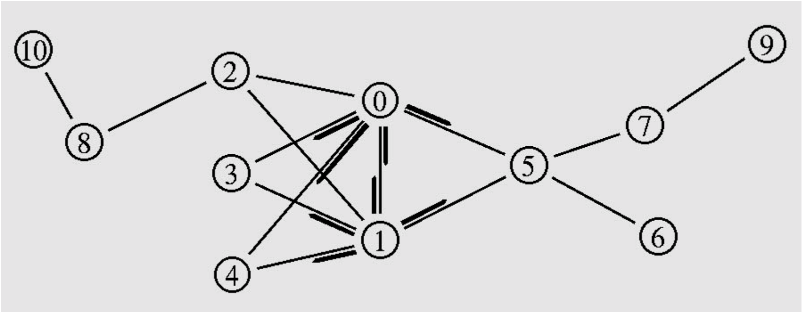


- 0.2525 s, node 2 propagates the route to nodes 0 and 1

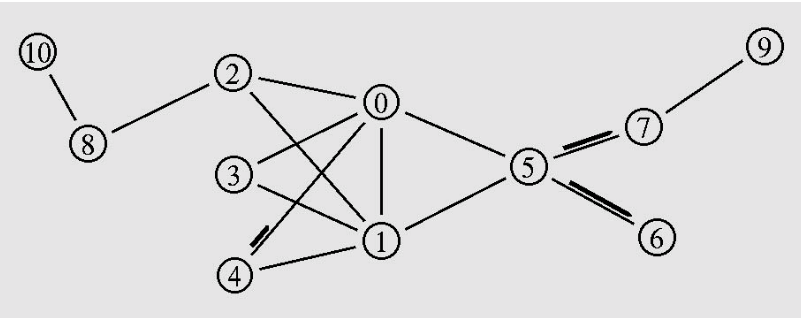


Simulation results: nam snapshots (2)

- 0.2561 s, nodes 0 and 1 reflect the route to nodes 3 and 4 and to their iBGP peers

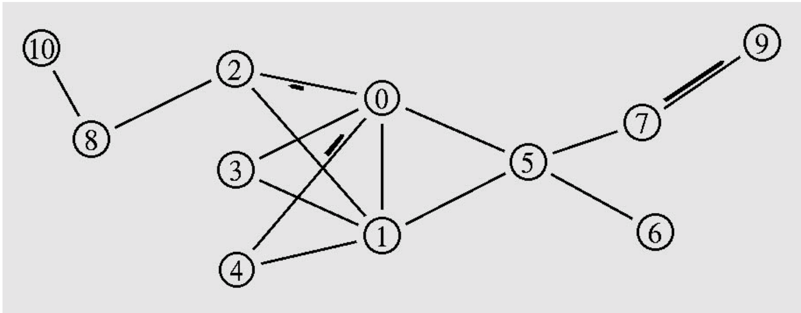


- 0.2568 s, node 5 reflects the route to nodes 6 and 7. Node 4 now knows the route to network 10.1.10.0/24, the UDP segment will be forwarded to node 10

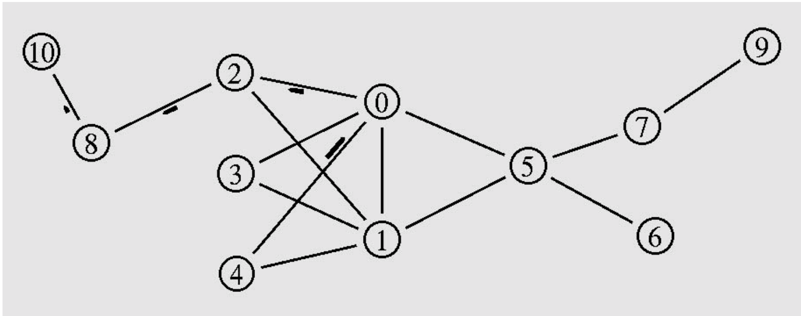


Simulation results: nam snapshots (3)

- 0.2578 s, the second UDP segment is sent to the node 10. Node 7 propagates the route to node 9



- 0.2580 s, UDP segments are delivered to node 10



Simulation results: routing tables

All ten nodes learned the routes to IP addresses 10.1.10.0/24 and 10.2.9.0/24

BGP routing table of node0
BGP table version is 2, local router ID is 10.0.0.1
Status codes: * valid, > best, i - internal.

	Network	Next Hop	Metric	LocPrf	Weight	Path	
*>	10.1.10.0/24	10.0.2.1	-	-	-	1	i
*>	10.2.9.0/24	10.0.7.1	-	-	-	2	i

.
.
.

BGP routing table of node9
BGP table version is 3, local router ID is 10.2.9.1
Status codes: * valid, > best, i - internal.

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.7.1	-	-	-	0 1
*>	10.2.9.0/24	self	-	-	-	

BGSP case study: roadmap

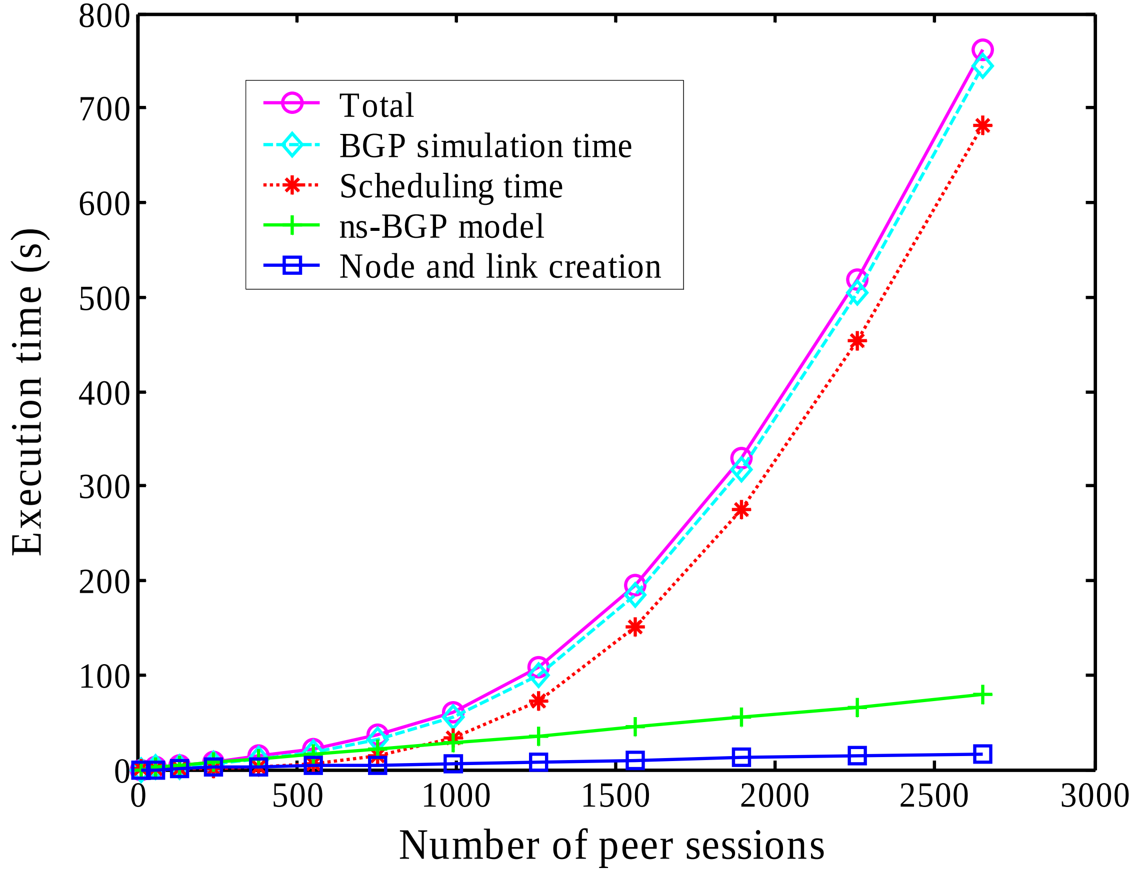
- Introduction
- Background
- Design and implementation of ns-BGP
- Validation test
- Scalability analysis
- Conclusions

Scalability analysis

- Scalability properties:
  - execution speed
  - memory requirements
- Scalability: number of peer sessions
- Scalability: size of routing tables
- Hardware platform:

  1.6 GHz Xeon host with 2 GBytes of memory

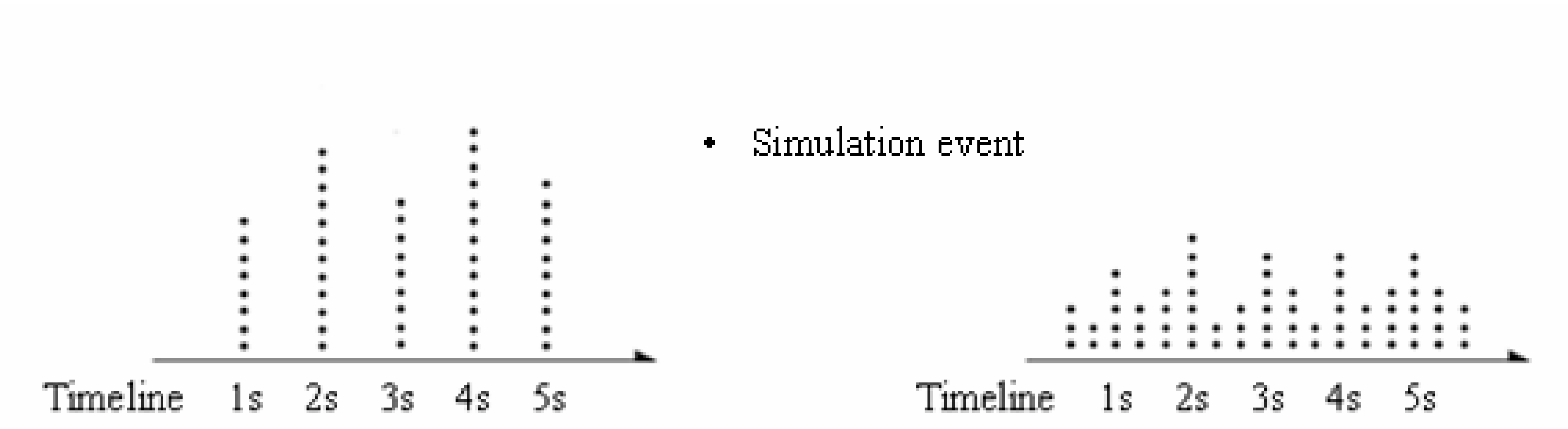
Scalability: number of peer sessions



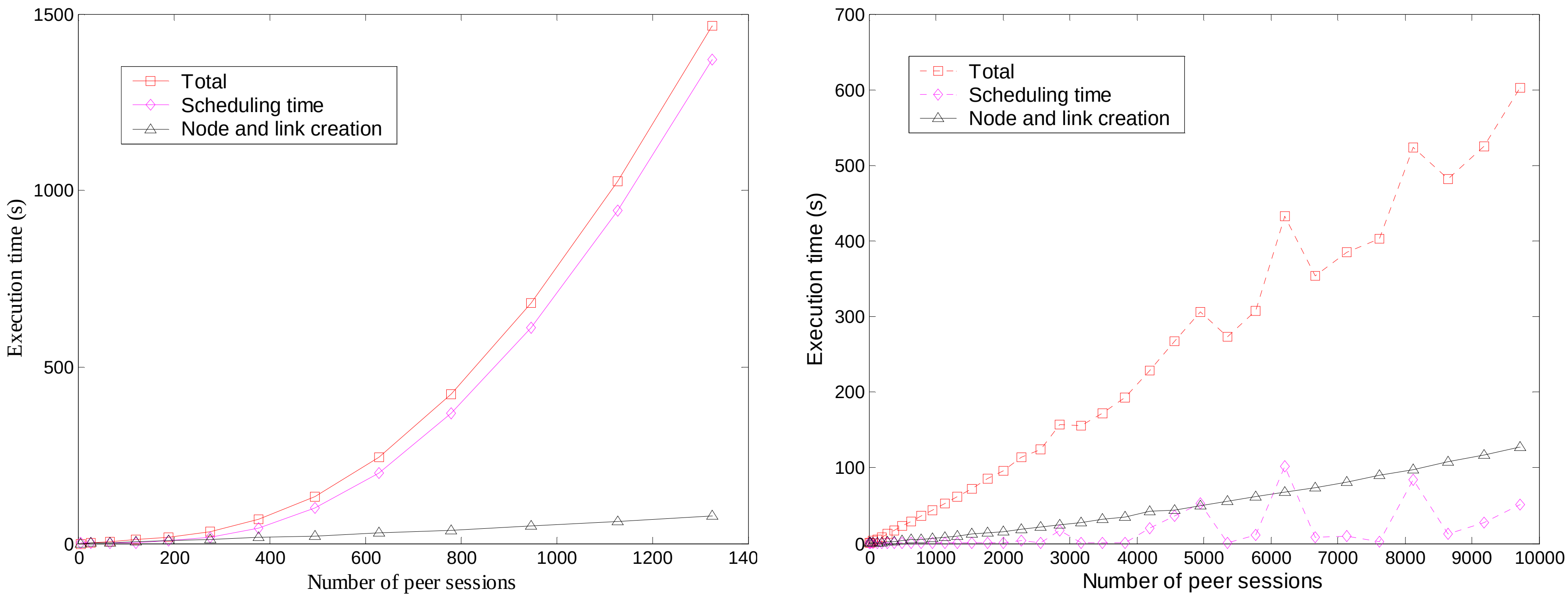
ns-2 calendar scheduler

- Performance is affected by the distribution of the event times
- Large number of events scheduled at the same time instance can cause the scheduling time to increase exponentially
- Solution: we jittered BGP timers (start-up, keep-alive) to scatter simulation events
- While the jittered scheduling times no longer increase exponentially, they are affected by the introduced jitter factors

Scattering events

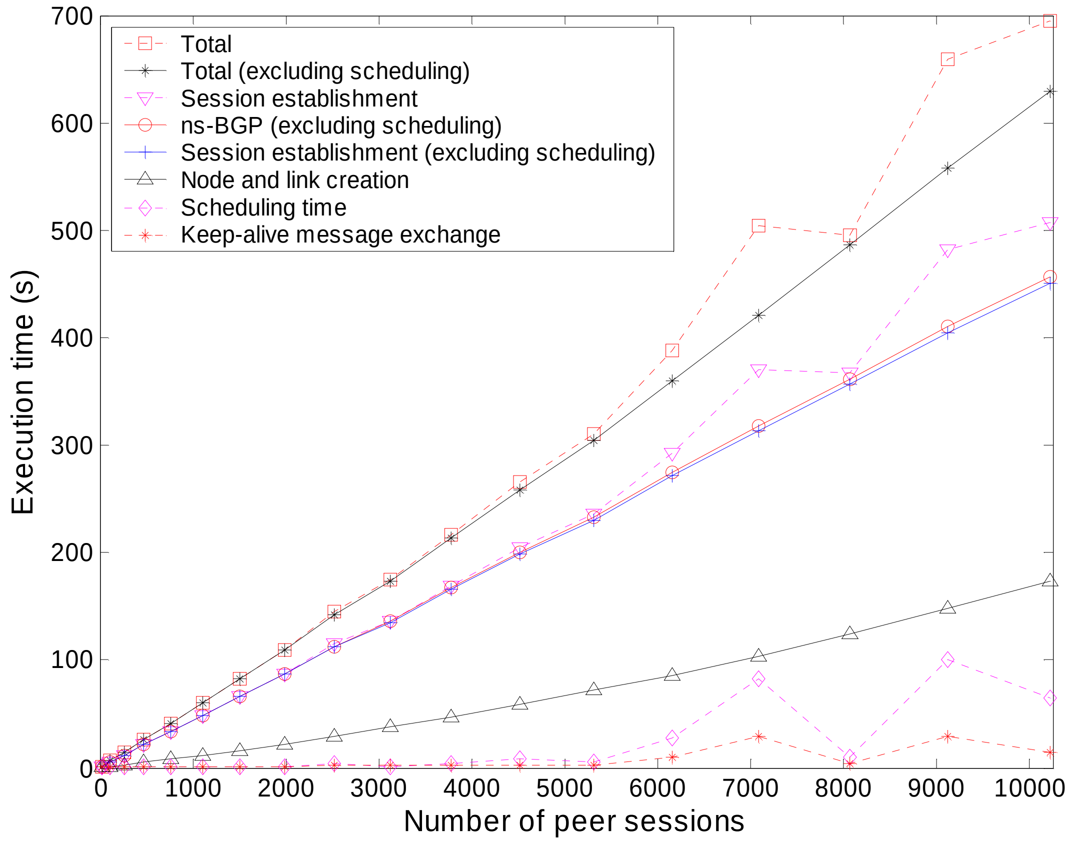


Scheduling times

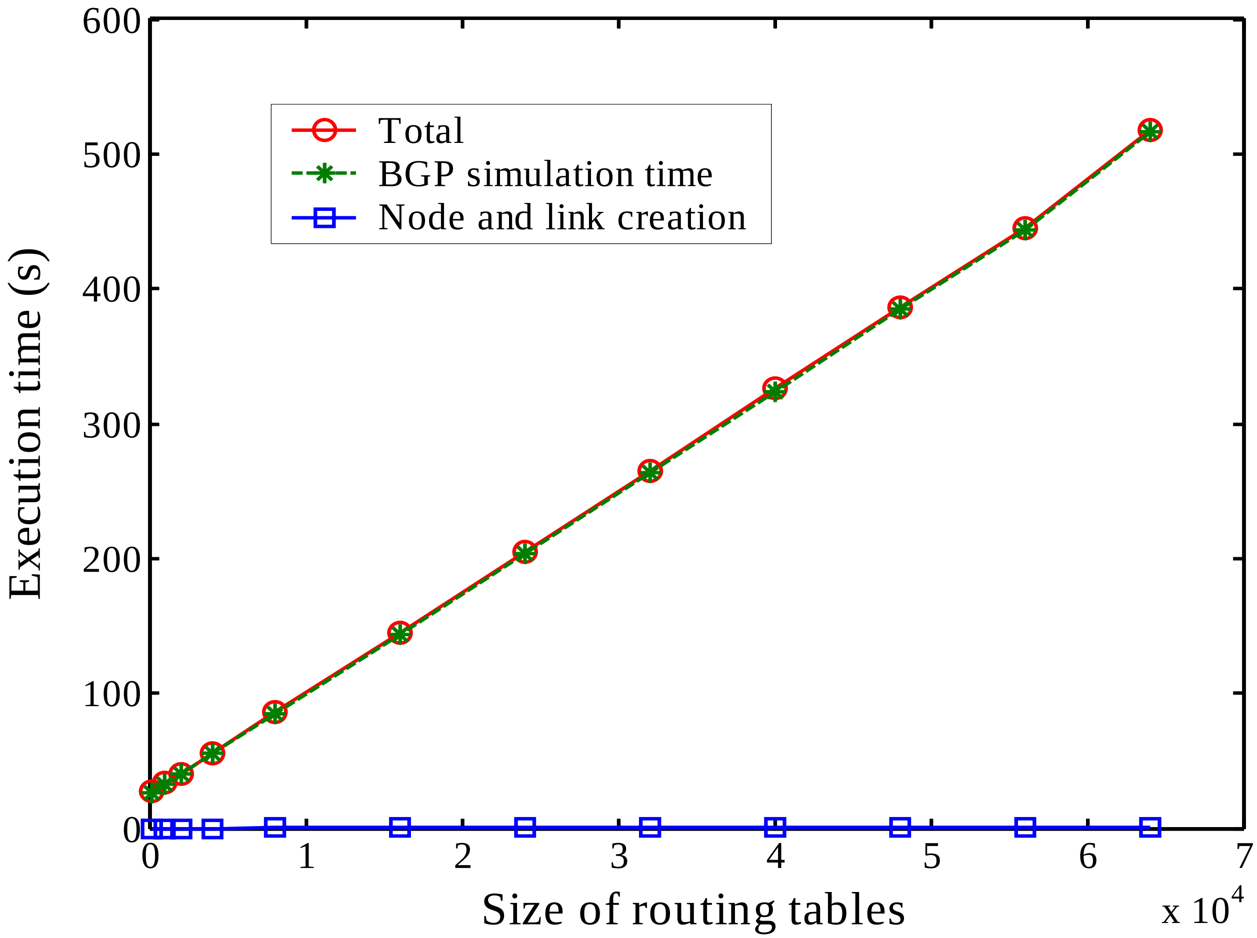


Execution time vs. number of peer sessions

- Line topology
  - total execution time
  - scheduling time
  - ns-BGP (excluding scheduling) execution time increases linearly
  - node and link creation time
- Ring, binary tree, grid, and clique topology
  - ns-BGP (excluding scheduling) execution times increase linearly



Scalability: size of routing tables



BGSP case study: conclusions

- We presented the architecture and implementation of ns-BGP, a BGP-4 model for the ns-2 network simulator
- ns-BGP enables simulation and evaluation of BGP protocol and its variants
- Validation tests illustrated the validity of the ns-BGP implementation
- Our scalability analysis showed that the internal data structures and employed algorithms are scalable with respect to the number of peer sessions and the size of routing tables
- New features: route flap damping

BGP case study: references

[1] T. Bates, R. Chandra, and E. Chen, "BGP route reflection – an alternative to full mesh IBGP," RFC 2796, April 2000.

[2] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, March 1995.

[3] T. Griffin and B. Premore, "An experimental analysis of BGP convergence time," in *Proc. ICNP*, Riverside, CA, November 2001, pp. 53-61.

[4] T. Griffin, F. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE Transactions on Networking*, vol. 10, no. 2, April 2002, pp. 232-243.

[5] S. Halabi and D. McPherson, *Internet Routing Architectures*. Indianapolis, IN: Cisco Press, 2000.

[6] D. Nicol, "Scalability of network simulators revisited," in *Proc of CNDS*, Orlando, FL, February 2003.

[7] B. Premore, *An Analysis of Convergence Properties of the Border Gateway Protocol Using Discrete Event Simulation*, PhD thesis, Dartmouth College, May 2003.

[8] J. Stewart III. BGP4: *Inter-Domain Routing in the Internet*, Addison-Wesley, 1998.

BGP case study: implementations

[9] T. D. Feng, R. Ballantyne, and Lj. Trajkovic, "Implementation of BGP in a network simulator," to be presented at the Applied Telecommunication Symposium, ATS '04, Arlington, Virginia, April 2004.

[10] BGP++: http://www.ece.gatech.edu/research/labs/MANIACS/BGP++. Accessed: April 10, 2004.

[11] GNU Zebra: http://www.zebra.org. Accessed: April 10, 2004.

[12] GNU Zebra BGP daemon: http://www.zebra.org/zebra/BGP.html#BGP. Accessed: April 10, 2004.

[13] ns manual: http://www.isi.edu/nsnam/ns/doc/index.html. Accessed: April 10, 2004.

[14] OPNET BGP: http://www.opnet.com/products/library/bgp.html. Accessed: April 10, 2004.

[15] B. Premore, SSFNet BGP User's Guide: http://www.ssfnet.org/bgp/user-guide-ps.zip. Accessed: April 10, 2004.

[16] SSFNet: http://www.ssfnet.org/homePage.html. Accessed: April 10, 2004.

Implementation of BGP in ns-2

- Rob Ballantyne
- Tony Dangliang Feng
- Wei (Steve) Shen
- Nenad Laskovic

Source code available at:

- ns-BGP 2.0: http://www.ensc.sfu.ca/~ljilja/cnl/projects/BGP
- Route Flap Damping and Adaptive Minimal Route Advertisement Interval in ns-BGP 2.0:

  http://www.ensc.sfu.ca/~ljilja/cnl/projects/RFD-AMRAI

Conclusions

- Introduction
- Network simulation tools
  - research: projects
  - teaching: graduate and undergraduate courses
- OPNET:
  - overview
  - simulation of GPRS: case study
- ns-2:
  - overview
  - BGP: case study
- Conclusions