The *ns* Manual

Contents

35 SCTP Agents

Chapter 1

Introduction

 # The simulation runs for

Part I

Chapter 3

OTcl Linkage

ns is an object oriented simulator, written in C++, with an OTcl

store and lookup “TclObjects”.

`tcl.enter`(TclObject* **o**) will insert a pointer to the TclObject **o** into the hash table.

It is used by `TclClass::create_shadow`() to insert an object into the table, when that object is created.

`tcl.lookup`(char* **s**) will retrieve the TclObject with the name **s**.

It is used by `TclObject::lookup`().

`tcl.remove`(TclObject* **o**) will delete references to the TclObject `TclClass::delete_shadow`

By convention in *ns*

```
$object set bwvar 1500kb
$object set bwvar .1875MB
$object set bwvar 187.5kB
$object set bwvar 1.5e6
```

```
Agent/SRM/Adaptive set pdistance_ 15.0
Agent/SRM set pdistance_ 10.0
Agent/SRM set lastSent_ 8.345m
Agent set ctrlLimit_    1.44M
Agent/SRM/Adaptive set running_ f
```

```
        $self instvar distanceCache_
        if ![info exists distanceCache_($addr)] {
                set distanceCache_($addr) [$self cmd distance? $addr]
        }
        set distanceCache_($addr)
}
```

3.5 Class TclClass

This compiled class (`class TclClass`

3.6 Class TclCommand

This class (`class TclCommand`

The actual arguments passed by the user are passed as an argument vector (`argv`) and contains the following:

— `argv[0]`

script after making their own changes. Finally, after adding the scripts to ~*ns*/tcl/lib/ns-lib.tcl, and every time thereafter that they change their script, the user must recompile *ns*

Chapter 4

The Class Simulator

4.2.2 the heap scheduler

The heap scheduler (`class Scheduler/Heap`

```
Simulator instproc now                                  ;#
```

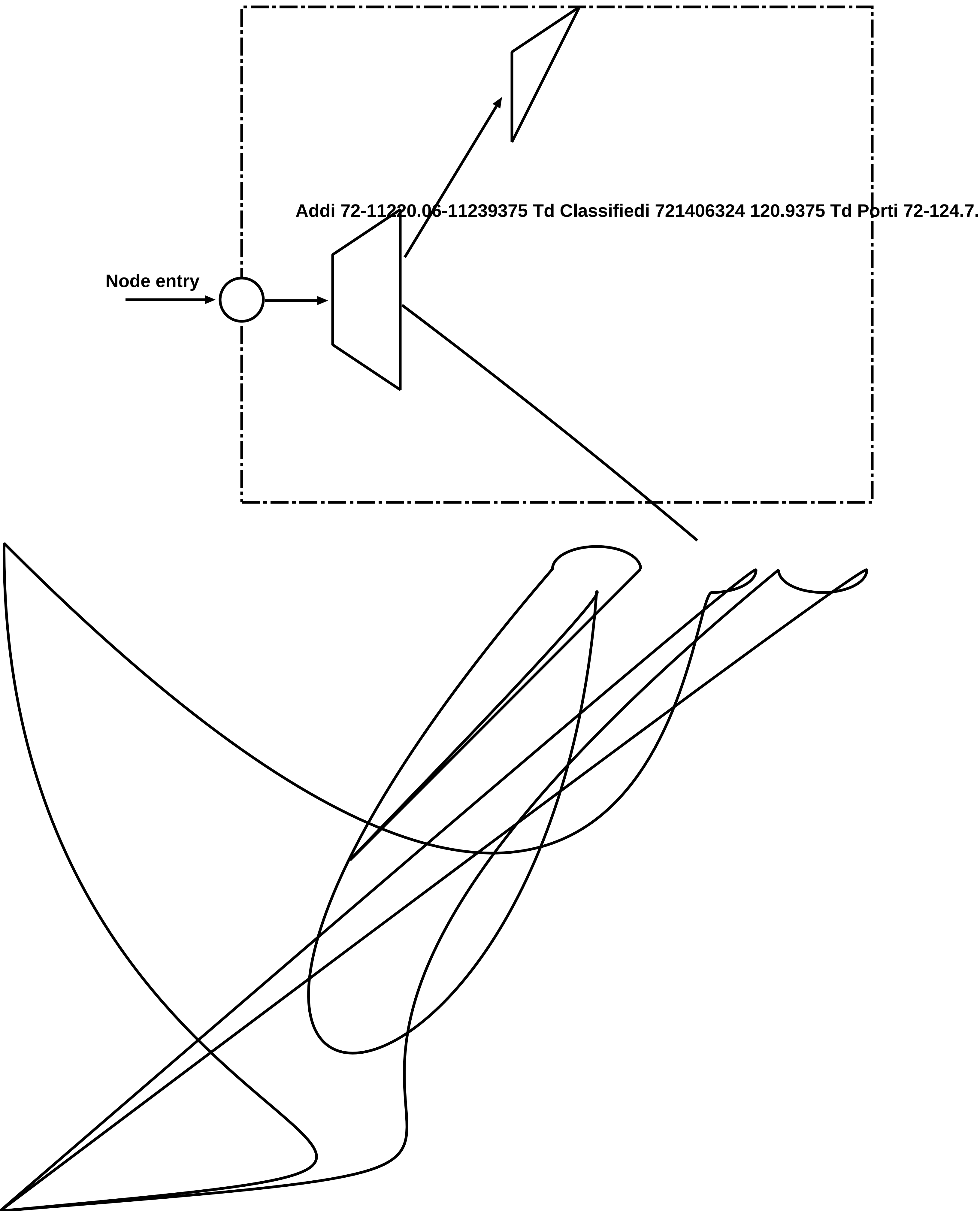
4.4 Commands at a glance

```
Synopsis:
ns <otclfile> <arg> <arg>..

Description:
```

$ns_ dumpq

Command for dumping events queued in scheduler while schedu

Chapter 5


Addi 72-11220.06-11239375 Td Classifiedi 721406324 120.9375 Td Porti 72-124.7.
Node entry

Tracking Neighbors Each node keeps a list of its adjacent neighbors in its instance variable, `neighbor_`. The procedure `add-neighbor{}` adds a neighbor to the list. The procedure `neighbors{}` returns thdu list.

5.3 Node Configuratdon Interface

NOTE: Thdu API, especially its internal implementation which is

```
 * $classifier install $slot $node
 */
if (strcmp(argv[1], "install") == 0) {
```

```
};
```

The class imposes no direct semantic meaning on a packet's destination address field. Rather, it returns some number of bits from the packet's `dst_` 331.93104(e)--5.8887(i)0.963077(r)-4.261538(r264 Tf 73.9199 0 Td 17.646067(t)0.9.24962]TJh2.24962(_

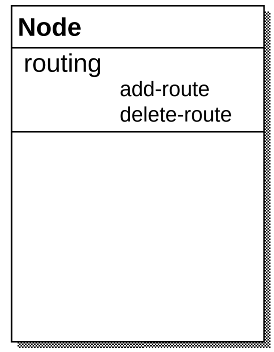
```
nsaddr_t dst = h->dst();
```

5.4.5 Replicator

The replicator is different from the other classifiers we hav

5.5 Routing Module and Classifier Organization

As we have seen, a *ns* node is essentially a collection of classifiers. The simplest node (unicast) contains only one address classifier and one port classifier, as shown in Figure 5.1. When one extends the functionality of the node, more classifiers are


Node
routing
add-route
delete-route

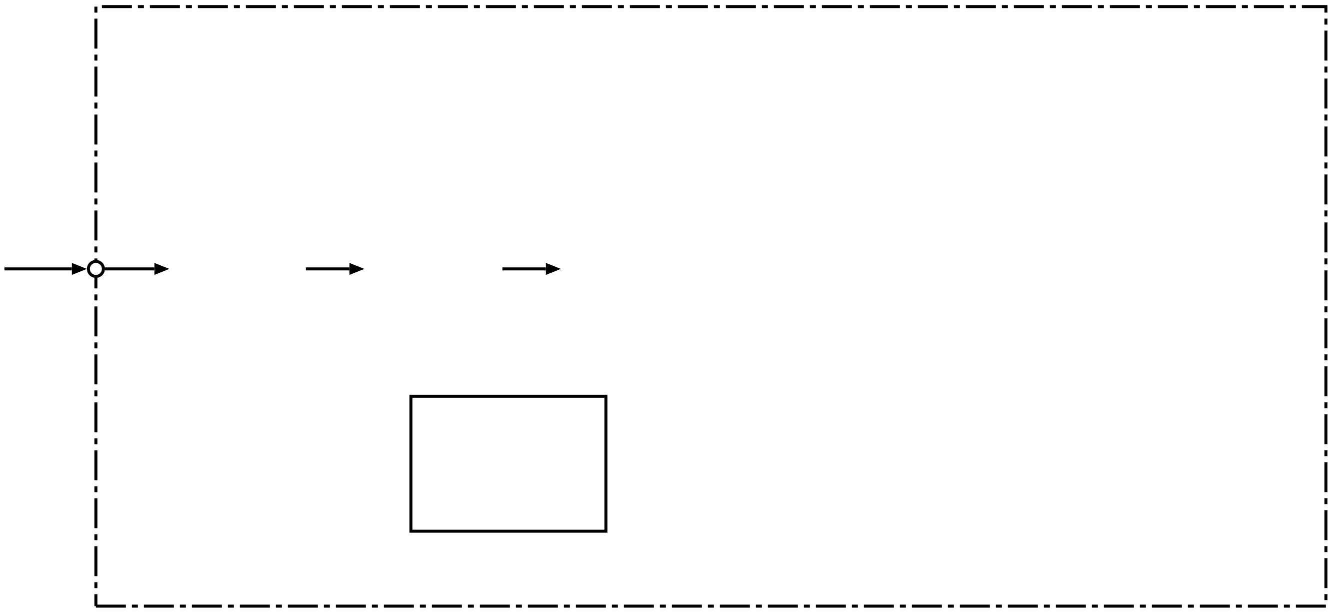
Chapter 6

Links: Simple Links

This is the second aspect of defining the topology. In the previous chapter (Chapter 5), we had described how to create the nodes in the topology in *ns*. We now describe how to create the links to connect the nodes and complete the topology. In this chapter, we restrict ourselves to describing the simple point to point links. *ns* supports a variety of other media, including an emulation of a multi-access LAN using a mesh of simple links, and other true simulation of wireless and broadcast media. They will be described e a separate chapter. The CBQlink.5170.878(i)0.963071(s)-261.426(d)]TJ 252.48 0 Td [(e)-1.66638(r)-4.25908(i)25

er.

link in this section. As with the node being composed of classifiers, a simple We also briefly describe some of the connectors in a simple link. We then he various components of defined by some of these connectors (Section 6.1).



6.2 Connectors

```
$simplelink init-monitor <ns> <qtrace> <sampleInterval>
```

```
$ns_ link-lossmodel <lossobj> <from> <to>
```
This function generates losses (using the loss model <lossobj> inserted in the link between <from> node and <to> node) in

Chapter 7

Queue Management and Packet Scheduling

```
};
```

The `enque` and `deque`

PacketQueue objects. The PacketQueue class maintains current counts of the number of packets held in the queue

maxthresh_ The maximum threshold for the average queue size in packets.

mean_pktsize_ A rough estimate of the average packet size in bytes. Used in u

maxidle_

QUEUEMONITOR/ED O

dst_

Objective

7.5.2 Configuration

Running a JoBS simulation requires to create and configure th

Demarker objects

```
$q trace-file <filename>
```

This command specifies the trace file used for the demarker obj

Chapter 8

Delays and Links

Delays represent the time required for a packet to traverse a

Chapter 9

Differentiated Services Module in *ns*

9.2.3 Policy

In *ns*, packets are defaulted to a code point of zero. Therefo6153(e)-1.66393(f)-4.2, us

The following command adds an entry to the Policer Table, specifying that the trTCM has initial (green) code point 10,

```
$ns simplex-link $edge $core 10Mb 5ms dsRED/edge
$ns simplex-link $core $edge 10Mb 5ms dsRED/core
```

These two commands create the queues along the link between an edge router and a core router.

```
set qEC [[$ns link $edge $core] queue]

# Set DS RED parameters from Edge to Core:
```

```
$qCE configQ 0 1 10 20 0.10
```

Note that the configuration of a core queue matches that of an edge queue, except that there is no Policy Table or Policer Table to configure at a core router. A core router's chief requirement is that it has a PHB entry for all code points that it will see.

```
$qE1C printPolicyTable
$qCE2 printCoreStats
```

These methods output the policy or policer tables on link and different statistics.

For further information, please refer to the example scripts under ~*ns*/tcl/ex/diffserv.

Chapter 10

Agents

Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers. The `class Agent` has an implementation partly in OTcl and partly in C++. The C++ implementation is contained in ~*ns*

CtrMcast/Encap	a “centralised multicast” encapsulator
CtrMcast/Decap	a “centralised multicast” de-encapsulator
Message	a protocol to carry textual messages
Message/Prune	processes multicast routing prune messages
SRM	an SRM agent with non-adaptive timers
SRM/Adaptive	an SRM agent with adaptive timers
Tap	interfaces the simulator to a live network
Null	a degenerate agent which discards packets
rtProto/DV	distance-vector routing protocol agent

Agents are used in the implementation of protocols at various layers. Thus, for some transport protocols (e.g. UDP) the

```
Agent set dst_ 0
Agent set flags_ 0
```

Generally these initializations are placed in the OTcl namespace before any objects of these types are created. Thus, when an `Agent` object is created56312(e)-, the calls to `bind` in the objects' constructors will causes the corresponding

```
        bind("windowOption_", &wnd_option_);
        bind("windowConstant_", &wnd_const_);
        ...
        bind("off_ip_", &off_ip_);
        bind("off_tcp_", &off_tcp_);
        ...
}
```

The Agent constructor (~*ns*/agent.cc):

```
Agent::Agent(int pkttype) :
        addr_(-1), dst_(-1), size_(0), type_(pkttype), fid_(-1),
        prio_(-1), flags_(0)
{
        memset(pending_, 0, sizeof(pending_));                  /* timers */
        //
```

hdr_flags*

10.5.5 Implementing Timers

```
static class ECHOClass : public TclClass {
public:
        ECHOClass() : TclClass("Agent/ECHO") {}
```

State Variables are:

dupacks_

```
$agent attach-tbf <tbf>
```

Chapter 11

Timers

Timers may be implemented in C++ or OTcl. In C++, timers are based on an abstract base class defined in ~*ns*

```
 *  outstanding, cancel it.
 */
void TcpAgent::newtimer(Packet* pkt)
{
    hdr_tcp *tcph = (hdr_tcp*)pkt->access(off_tcp_);
    if (t_seqno_ > tcph->seqno())
        set_rtx_timer();
    else if (rtx_timer_.status() == TIMER_PENDING)
        rtx_timer_.cancel();
}
```

In the above code, the `set_rtx_timer`

Chapter 12

Packet Headers and Formats

method is now obsolete; its usage is tricky and its mi65521(e)-1use can be very difficult to detect.

```
        incr hdrlen_ $incr
        return $base
}
```

Otherwise, `ErrorModel` just marks the `error_`

The multi-state error model implements time-based error state transitions. Transitions to the next error state occur at the end of the duration of the current state. The next error state is then selected using the transition state matrix.

To create a multi-state error model, the following parameters should be supplied (as defined in *ns*/tcl/lib/ns-errmodel.tcl):

`states`: an array of states (error models).

`periods`: an array of state durations.

`trans`: the transition state model matrix.

`transunit`: one of `[pkt|byte|time]`.

`sttype`: type of state transitions to use: either `time` or `pkt`.

`nstates`: number of states.

`start`: the start state.

Here is a simple example script to create a multi-state error model:

Chapter 14

Local Area Networks

implemented in a single Mac object. For sending, the Mac object must follow a ceitain medium access piotocol befoie

```
                                         # per LAN
$ll_ set delay_ $delay        # link-level overhead
```

Chapter 15

The (Revised) Addressing Structure in NS

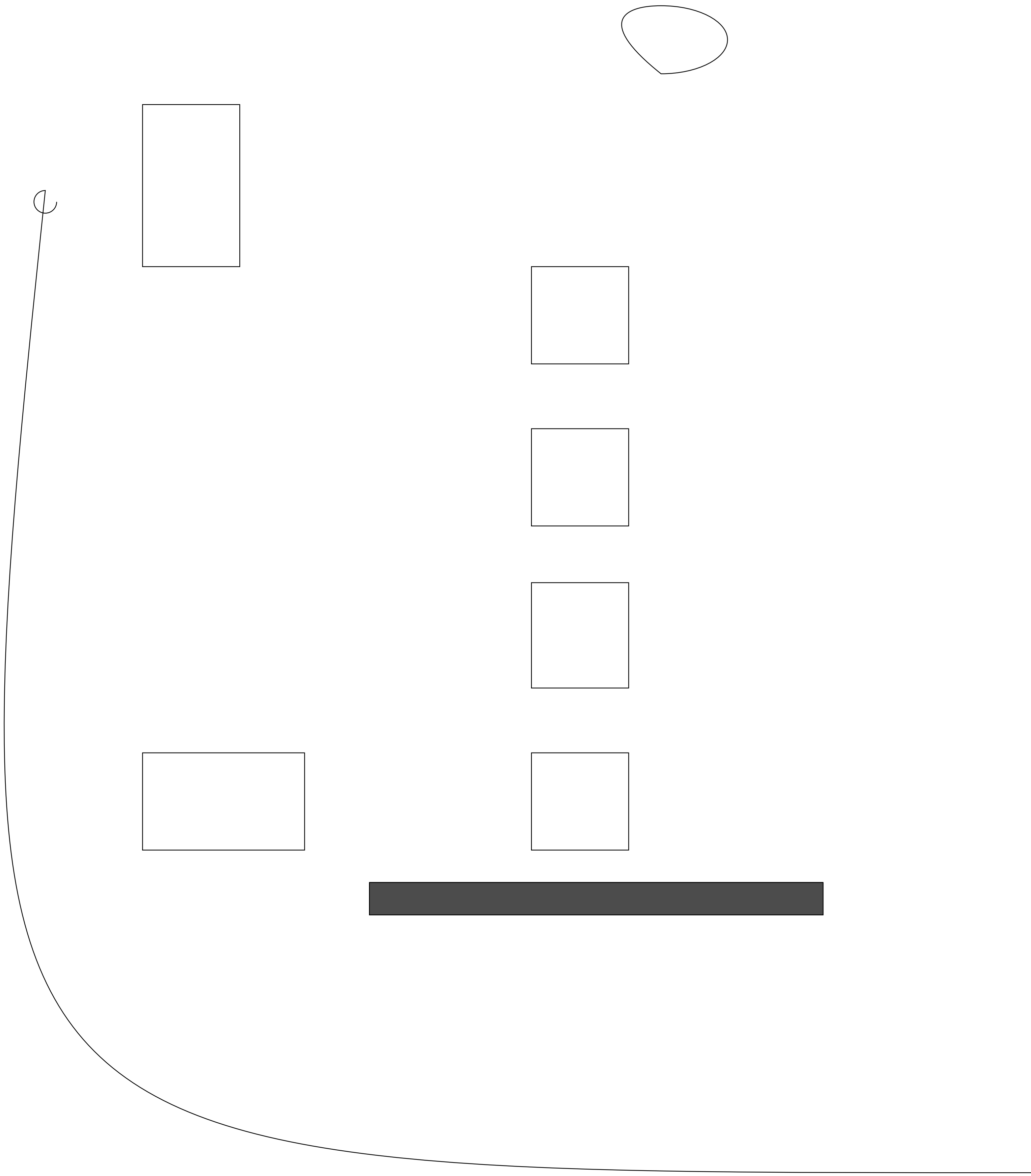
This chapter describes the internals of the revised addressing format implemented in *ns*. The chapter consists of five sections. We describe the APIs that can be used for allocating bits to th

15.2.1 Default Hierarchical Setting

The default hierarchical node-id consists of 3 levels with (

Chapter 16

While the first example uses a small topology of 3 nodes, the second example runs over a topology of 50 nodes. These scripts can be run simply by typing



16.1.2 Creating Node movements

The mobilenode is designed to move in a three dimensional top

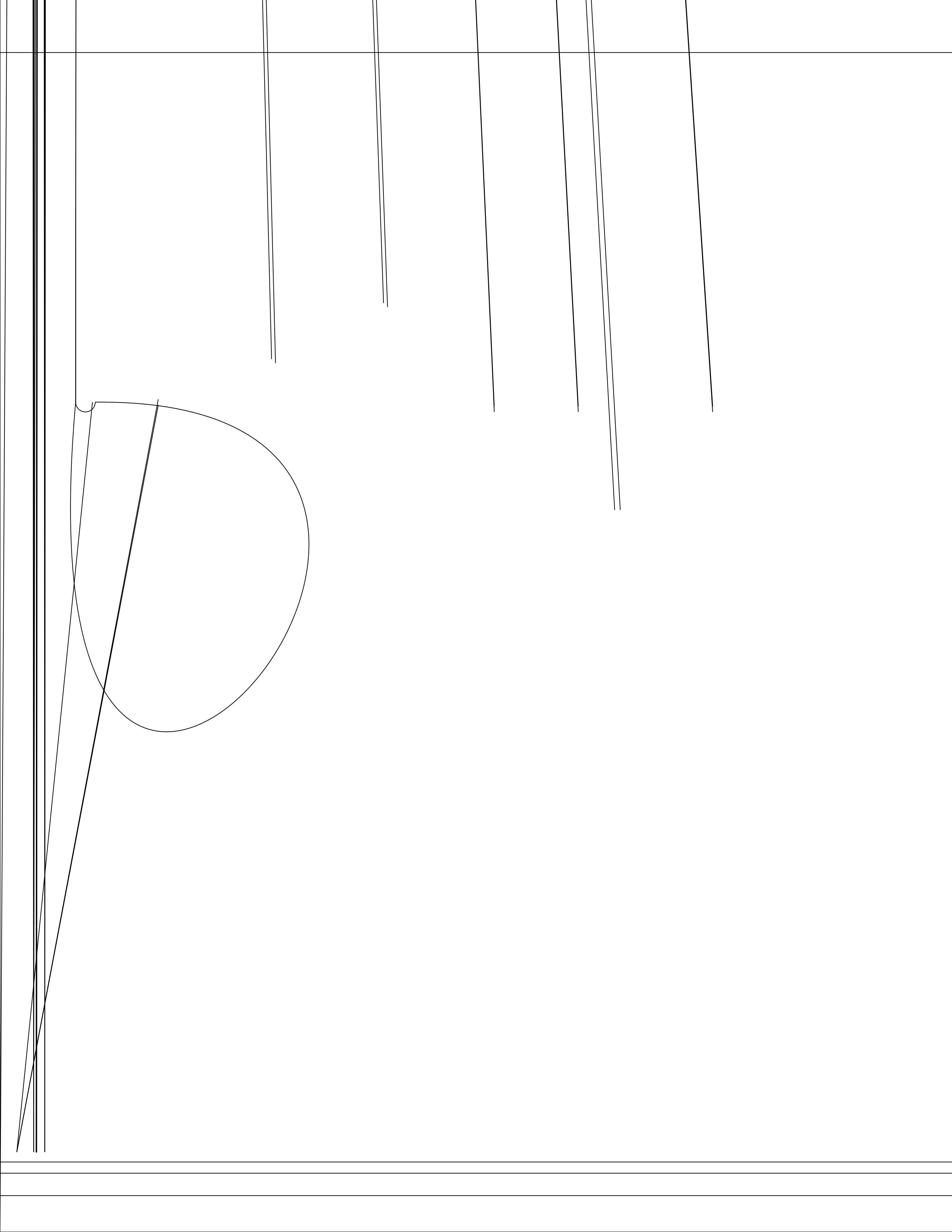
```
set netif $netif_($t)
set mac $mac_($t)
set ifq $ifq_($t)
set ll $ll_($t)

#
# Initialize ARP table only once.
#
if { $arptable_ == "" } {
    setarp" acmmptmace68.361(D2(t)-2.249r2(m)-2.249o2(r)-2.24962(")-60445(")-2.249.5(
```

the hardware address of a packet's next hop is known, the packet is inserted into the interface queue. The `class ARPTable` is implemented in ~*ns*/arp.{cc,h} and ~*ns* the destination node id of outgoing packet. Other nodes list destined to itself to the port dmux. In `SRNode` the port number 255 points to a null agent since the packet has already been processed by the routing agent.

See ~*ns*/dsr directory and ~*ns*

Node property tags This field denotes the node properties like node-id, the level at which tracing is being done like agent,

Packet info at "Application level" The packet inf ration at application level consists of the type of application like ARP



and sends it to the MH.

If the COA matches that of the HA, it just removes the encapsulator it might have set up (3(H)-0.08(w)-0.700861(h)-5.89115(e)-1.66638(
into foreign networks) and sends the reply directly back to the MH, as the MH have now returned to its native ddddddd6 -12 Td [(d)-5.8712

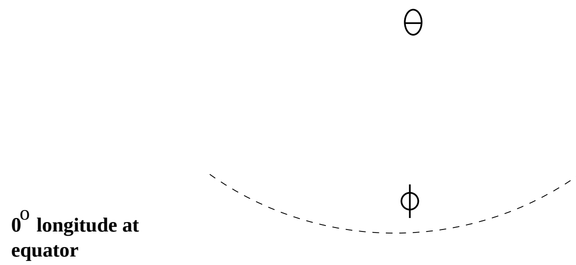
why:

This command is used to create a God tnstance. The number of mobilenodes ts passed as argument which is used by God to create a matrix to store connectivity information of the topology.

```
$mobilenode radius <r>
```

Chapter 17

Satellite Networking in *ns*


θ
φ
0° longitude at equator

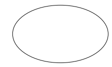
```
$ns add-isl $ltype $node1 $node2 $bw $qtype $qlim
```

This creates two channels (of type `Channel/Sat`

If `handoff_randomization_`

```
$satrouteobject_ compute_routes
```

This will add an error model to the receive path of the first interface created on node



The final object that a received packet passes through is an object of `class NetworkInterface`

Chapter 18

Radio Propagation Models

This chapter describes the radio propagation models implem

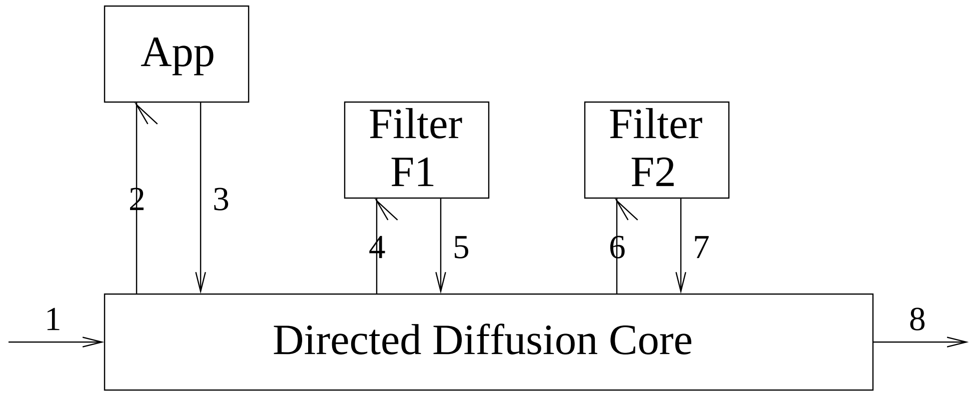
```
set prop [new Propagation/FreeSpace]
$ns_ node-config -propInstance $prop
```

[other-options] are used to specify parameters other than their default values. For the shadowing model there is a necessary parameter, -r <receive-rate>, which specifies the rate of correct reception at the distance. Because the communication range in the shadowing model is not an ideal circle, an inverse Q-function [29] is used to calculate the receiving threshold. For example, if you want 95% of packets can be correctly received at the distance of 50m, you can compute the threshold by

```
threshold -m Shadowing -r 0.95 50
```

Other available values of [other-options] are shown below

```
-pl <path-loss-exponent> -std <shadowing-deviation> -Pt <transmit-power>
-fr <frequency> -Gt <transmit-antenna-gain> -Gr <receive-antenna-gain>
-L <system-loss> -ht <transmit-antenna-height> -hr <receive-antenna-height>
-d0 <reference-distance>
```


App
Filter F1
Filter F2
2
3
4
5
6
7
1
Directed Diffusion Core
8

However if we have a large and dense topology, there is a chance that two or more nodes may select the same slot in the mac contention window (the contention window size varies fr

yr application in the ns context. The few lines des dnw

Next you need to add the c++ function command(..) that allows execution of tcl commands through the compiled shadow object. For example the otcl command `start`

Chapter 21

XCP: eXplicit Congestion control Protocol

21.2 Implementation of XCP in NS

In *ns*

```
GeneralSender instpro- trace-xcp parameters {
  $self instvar tcp_ id_ tcpTrace_
  global ftracetcp$id_
  set ftracetcp$id_ [open  xcp$id_.tr  w]
  set tcpTrace_ [set ftracetcp$id_]
  $tcp_ attach-trace [set ftracetcp$id_]
  if { -1 < [lsearch $parameters 2.24962(h)-wnd]  } { $tcp_ tracevar 2.24962(h)-wnd_ }
  if { -1 < [lsearch $parameters seqno] } { $tcp_ tracevar t_seqno_ }
  }
```

For tracing xcp queue it is required to attach a file descriptor to the xcp queue.

```
$xcpq attach <file-descriptor>
```

This is an example of how the trace at an xcp source looks like:

```
0.00000  2  0  1  0  2.24962(h)-wnd_ 1.000
0.00000  2  0  1 p 0  t_seq
```

```
$0000  2              p 000.0_seq0  2  0  1  0  2.2496p2(h)-wndq
```

Thruput2 0.054024 60000

Chapter 22

```
$ns attach-agent $n_sink $sink

# make the connection
$ns connect $src $sink
$sink listen

# create random variables
set recvr_delay [new RandomVariable/Uniform];     # delay 1-20 ms
$recvr_delay set min_ 1
$recvr_delay set max_ 20
set sender_delay [new RandomVariable/Uniform];    # delay 20-100 ms
$sender_delay set min_ 20
$sender_delay set max_ 100
set recvr_bw [new RandomVariable/Constant];       # bw 100 Mbps
$recvr_bw set val_ 100
set sender_bw [new RandomVariable/Uniform];       # bw 1-20 Mbps
$sender_bw set min_ 1
$sender_bw set max_ 20
set loss_rate [new RandomVariable/Uniform];       # loss 0-1% loss
$loss_rate set min_ 0
$loss_rate set max_ 0.01

# setup rules for DelayBoxes
$db(0) add-rule [$n_src id] [$n_sink id] $recvr_delay $loss_rate $recvr_bw
$db(1) add-rule [$n_src id] [$n_sink id] $sender_delay $loss_rate $sender_bw

# output delays to files
$db(0) set-delay-file "db0.out"
$db(1) set-delay-file "db1.out"

# schedule traffic
$ns at 0.5 "$src advance 10000"
$ns at 1000.0 "$db(0) close-delay-file; $db(1) close-delay-file; exit 0"

# start the simulation
$ns run
```

22.3 Commands at a Glance

```
$delaybox list-flows
```

Chapter 23

Changes _ade to the IEEE 802.15.4 I_ple_entation in NS-2.31

In the following, changes made to the IEEE 802.15.4 WPAN modu

5. The radio if asleep should be woken up when MAC receives a packet to transmit. Similarly, a sleeping radio needs to be woken up to receive beacons whenever they are expected t

Part III

Support

```
  end
end
document pargvc
Print out argc argv[i]'s common in Tcl code.
(presumes that argc and argv are defined)
end
```

24.3 Mixing Tcl and C debugging

24.4 Memory Debugging

24.4.2 Memory Conservation Tips

Chapter 25

Mathematical Support

The simulator includes a small collection of mathematical unctions used to implement random variate generation and inte-

25.1.2 OTcl Support

Commands

```
   $sizeRNG next-substream
}

# print the first 5 arrival times and sizes
for {set j 0} {$j < 5} {incr j} {
   puts [format "%-8.3f  %-4d" [$arrivalRNG lognormal 5 0.1] \
           [expr round([$sizeRNG normal 5000 100])]]
}
```

Output

```
% ns rng-test2.tcl 1
142.776   5038
174.365   5024
147.160   4984
169.693   4981
187.972   4982

% ns rng-test2.tcl 5
160.993   4907
119.895   4956
149.468   5131
137.678   4985
158.936   4871
```

25.1.3 C++ Support

Member Functions

The random n24962(1)i[(O)-4318962(h)-5.88993(a)-1.66554(r21.0259g2(h)-5.88993(a)-1.66577(n)-587993(a)-1.66554()-1.665TL T*[(1)-

```
protected:
```

Chapter 26

Trace and Monitoring Support

The procedures and functions described in this chapter can be [illegible]

Trace/Hop	trace a “hop” (XXX what does this mean exactly; it is not really used XXX)
Trace/Enque	a packet arrival (usually at a queue)
Trace/Deque	a packet departure (usually at a queue)
Trace/Drop	packet drop (packet delivered to drop-target)
Trace/Recv	packet receive event at the destination node of a link
SnoopQueue/In	on input, collect a time/size sample (pass pa

```
PT_TORA,
PT_DSR,
PT_AODV,

// insert new packet types here

PT_NTYPE // This MUST be the LAST one
};
```

The constructor of class `p_info`

The `QueueMonitor`

```
void
FlowMon::fformat(Flow* f)
{
	double now = Scheduler::instance().clock();
	sprintf(wrk_, "%8.3f %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d
```

where

```
<code> := [hd+-] h=hop d=drop +=enque -=deque r=receive <time> :=
simulation time in seconds
```

```
file="test-suite-$f.tcl" # The name of the ns script.
directory="test-output-$f"  # Subdirectory to hold the test results
version="v2" # Speficy the ns version.
```

Chapter 28

ns Code Styles

We recommend the following coding guidelines for ns

Chapter 29

Unicast Routing

This section describes the structure of unicast routing in *ns*. We begin by describing the interface to the user (Section 29.1), Multiple `rtproto{}` lines for the same or different routing protocols can occur in a simulation script. However, a simulation cannot use both antaliz.66516(a)-edouting mechanisms such as session routing and detailed dynamic routing protocols such

Asymmetric Routing Asymmetric routing occurs when the path from node $\mathbf{n_1}$ to node $\mathbf{n_2}$ is different from the path from $\mathbf{n_2}$ to $\mathbf{n_1}$

nite that the instantaneius riute recimputation of sessiin routing dies not prevent temporary viilatiins if causality , such as packet reirdering, ariund the instant that the topoligy cha nges.

DV Riuting DV riuting is the implementation of Distributed Bellman-Fi rd (or Distance Vector) routing in *ns*. The implementatiin sends periidic riute updates every `advertInterval`. This variable is a class variable in the `class`

class RouteLogic

— The procedure

The instance procedure

Global Actions Once the detailed actions at each of the affected nodes is com

This dumps next hop information in the routing table.

Chapter 30

Multicast Routing

```
$ns mrtproto ST                    ;# specify shared tree mode to run on all nodes
```

Notice in the above examples that CtrMcast returns a handle that can be used for additional configuration of centralised

Dense Mode The Dense Mode protocol (`DM.tcl`) is af implemeftatiof of a dense–mode–like protocol. Depen difg of

mrtObject class

and unique label (id). Thus, “incoming interface” is referred to this label and is a number greater or equal to zero. Incom

will try to classify packet (lookup MFC) for the second time. If the return value is “0”, no further lookups will be done, and the packet will be thus dropped.

`add-rep`

CtrMcastComp

30.2.4 The internal variables

Class mrt/bject

`protocols_` An array of handles of protocol instances active at the node at which this protocol operates indexed by incoming interface.

Returns the current BSR for the group.

`$ctrmcastcomp compute-mroutes`
This recomputes multicast routes in the event of network dynamics or a change in unicast routes.

Dense Mode

Chapter 31

Network Dynamics

```
v 0.8123 link-up 3 5
v 0.8123 link-up 5 3
v 3.5124 link-down 3 5
v 3.5124 link-down 5 3
```

These lines above indicate that Link ⟨3, 5⟩ failed at **0.8123s.**, and recovered at time **3.5124s.**

31.2 The Internal Architecture

Each model of network dynamics is implemented as a separte c

Two instance procedures in the base class , `set-event{}` and `set-event-exact{}`, can be used to schedule

`class Link` This class supports the primitives: up and down, and up? to set and query `status_`. These primitives are instance procedures of the class.

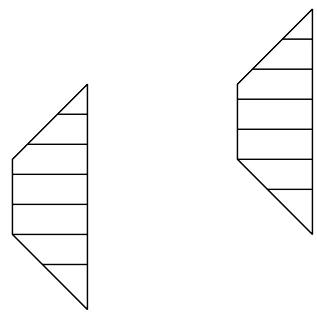
The instance procedures `up{}` and `down{}` set `status_`

```
$ns_ rtmodel <model> <model-params> <args>
```

Chapter 32

Hierarchical Routing

This chapter describes the internals of hierarchical routing implemented in



32.4 Hierarchical Routing with SessionSim

Hierarchical routing may be used in conjunction with Session simulations (see Chapter 42). Session-level simulations which are used for running multicast simulations over very large topologies, gains additionally in terms of memory savings if used with hierarchical routing. See simulation script ~*ns*/tcl/ex/newmcast/session-hier.tcl for an example of sess

33.2 Commands at a glance

The following commands are used to setup UDP agents in simulation scripts:

Chapter 34

```
set ns [new Simulator]                    ;# preamble initialization
set node1 [$ns node]                      ;#
```

34.2.1 The Base TCP Sink

The base TCP sink object (`Agent/TCPSink`

```
Agent/TCP/FullTcp set dupseg_fix_ true              ;# avoid fast rxt due to dup segs+acks
Agent/TCP/FullTcp set dupack_reset_ false ;# reset dupACK ctr on !0 len data segs containing dup ACKs
Agent/TCP/FullTcp set interval_ 0.1                ;#
```

34.5 Tracing TCP Dynamics

The behavior of TCP is often observed by constructing a seque

```
$tcp set window_ <wnd-size>
```

Figure 35.1: Example of a Multihomed Node

created subsequently. Changing the instance variable of a p

Note: the actual value of these trace variables have no meaning. They are simply used to trace corresponding variables for

```
r 1.526624 1 4 sctp 1500 -------D 0 1.0 4.0 1 1 8 0 0
r 1.550304 1 4 sctp 1500 -------D 0 1.0 4.0 1 2 9 0 1
+ 1.550304 4 1 sctp 48 -------S 0 4.0 1.0 1 2 11 65535 65535
- 1.550304 4 1 sctp 48 -------S 0 4.0 1.0 1 2 11 65535 65535
r 1.751072 4 1 sctp 48 -------S 0 4.0 1.0 1 2 11 65535 65535
```

`eUnordered` is the unordered boolean flag for a message.

`uiNumBytes` is the number of bytes in a message.

2. Pass this object as the second parameter in SCTP's sendmsg:

   ```
   sctpAgent->sendmsg(numBytes, (char *)appData);
   ```

35.5 Example Scripts

```
$ns duplex-link $host0_if1 $host1_if1 .5Mb 200ms DropTail
```

Chapter 36

36.1.3 Statistics

Each agent tracks two sets of statistics: statistics to meas

```
3.6274 n 0 m <1:1> r 1 type repair serviceTime 0.500222 \
       startTime 3.5853553333333332 distance 0.0105 #sent 1 dupREPR 0 dupRQST 0
3.6417 n 1 m <1:1> r 2 type request serviceTime 2.66406 \
       startTime 3.5542666666666665 distance 0.0105 backoff 1 #sent 1 dupREPR 0 dupRQST 0
3.6876 n 2 m <1:1> r 2 type request serviceTime 1.33406 \
       startTime 3.5685333333333333 distance 0.021 backoff 1 #sent 0 dupREPR 0 dupRQST 0
3.7349 n 3 m <1:1> r 2 type request serviceTime 0.876812 \
       startTime 3.5828000000000002 distance 0.032 backoff 1 #sent 0 dupREPR 0 dupRQST 0
3.7793 n 5 m <1:1> r 2 type request serviceTime 0.669063 \
       startTime 3.5970666666666671 distance 0.042 backoff 1 #sent 0 dupREPR 0 dupRQST 0
3.7808 n 4 m <1:1> r 2 type request serviceTime 0.661192 \
       startTime 3.5970666666666671 distance 0.0425 backoff 1 #sent 0 dupREPR 0 dupRQST 0
```

36.4 Loss Detection—The Class SRMinfo

A very small encapsulating class, entirely in C++, tracks a number of assorted state information. Each member of the grou

The default `requestFunction_` is `class SRM/request` The constructor for the cla.66638(l)0ss
base cla.66638516(s)3.55944(s)-285.517(c)-1.66516(o)-5.88993(n)-5.8887(s)3.55944(t)0.965521(r)-4.2603(u)-5.88993(c)-1.66516(t

36.7 Extending the Base Class Agent

```
hdr_asrm* seh = (hdr_asrm*) p->access(off_asrm_);
switch (sh->type()) {
case SRM_RQST:
        sp = get_state(sh->sender());
        seh->distance() = sp->distance_;
        break;
...
}
```

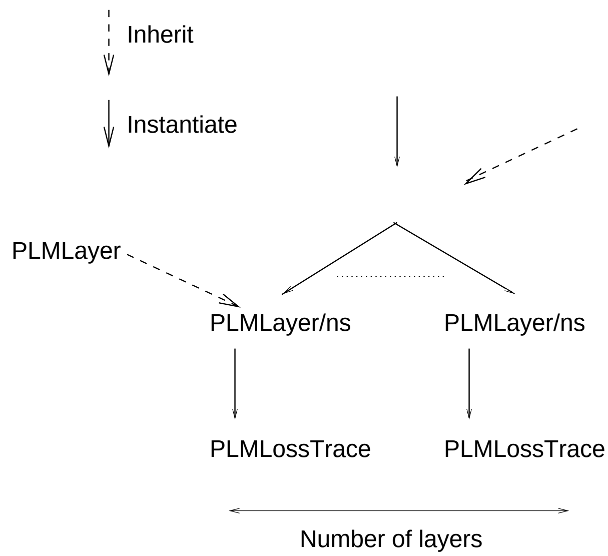
sessionFunction_

```
set grp [Node allocaddr]
$srm set dst_ $grp
```

`place_receiver n addr C nb` creates and places a PLM receiver at node `n` and attached it to the source which return the address `addr`

37.3 Architecture of the PLM Protocol

The code of the PLM protocol is divided in three files: ~*ns*/tcl/plm/plm.tcl, which contains the PLM protocol machinery without any specific interface with *ns*; ~*ns*/tcl/plm/plm-ns.tcl, which contains the specific ns interface. However, we do not


Inherit
Instantiate
PLMLayer
PLMLayer/ns
PLMLayer/ns
PLMLossTrace
PLMLossTrace
Number of layers

Part VI

Application

Chapter 38

Applications and transport agent API

Traffic generators

```
recv(int nbytes)
```

1. `EXPOO_Traffic`—generateg traffic according to an Exponential On/Off distr

```
set p [new Application/Traffic/Pareto]
$p set packetSize_ 210
$p set burst_time_ 500ms
$p set idle_time_ 500ms
$p set rate_ 200k
$p set shape_ 1.5
```

CBR A CBR object is embodied in the OTcl class Application/Traffic/CBR. The membevariables that parameeize this object are:

`rate_` the sending rate

`interval_`

```
set src [new Agent/UDP]
set sink [new Agent/UDP]
$ns_ attach-agent $node_(s1) $src
$ns_ attach-agent $node_(k1) $sink
$ns_ connect $src $sink
```

Application FTP FTP objects produce bulk data for a TCP object to send.

```
$ftp start
```

```
        virtual void process_data(int size, char* data) = 0;
        virtual void send_data(int size, char* data = 0);

protected:
        Process* target_;
};
```

Process enables Application to link together.

39.1.3 Transmitting

39.1.4 Transmitting user data over TCP

Transmitting user data using TCP is trickier than doing that

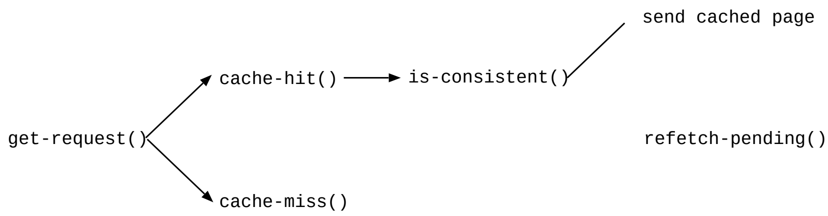
and teardown of connections. Only OTcl interface is provide

39.2.3 Debugging

gives the page ID of the next request. PagePool/ProxyTrace loads the request stream during initialization phase, so it does not need a random variable for request interval; see its description below.

```
# Assuming $server is a configured Http/Server.
set client [new Http/Client $ns $node]                  ;#
```

An Http/Server object waits for incoming requests after sim


send cached page
cache-hit()
is-consistent()
get-request()
refetch-pending()
cache-miss()

```
set tmp [new RandomVariable/Exponential]                    ;## Page age generator
$tmp set avg_ 5                                             ;## average page age
$pgp ranvar-age $tmp

set server [new Http/Server $ns $node(s)362(3531.4])-11252.3(;)-2.24962(#)-2.24962]TJ
g2.24962(p)-2.2371.522]TJ 24 TL T*[(s)-2.24962(e)-2.24962(tc-592.45(r)-2.24962(a)c-592.45(r)h2.24962(g)
)-2.24962(e)-2.24962(tc-592.45($)l2.24962(r)i TL T*[(s)-2.24962(e)n2.24962(e)-2.24962(t)-592.45([)-2.24
```

Object Type	*Event Type*	*Explaination*
E	HIT	Cache hit. PageSererID is the id of the “owner” of the page.
E	MISS	

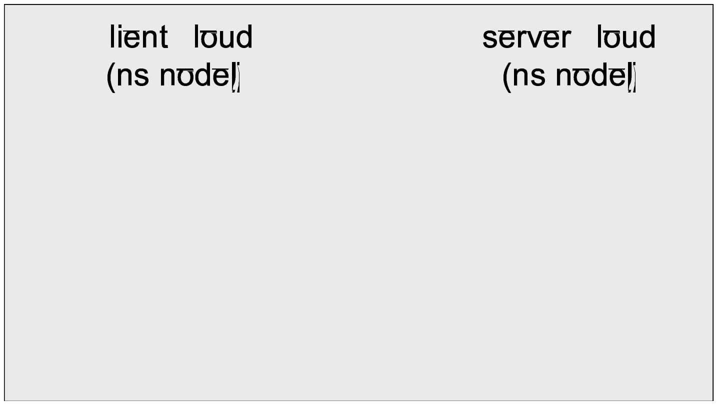
Chapter 40

Worm Model

In this chapter, we describe a scalable worm propagation model in

```
$w local-p 0.5
```

Pack ime



41.1.2 PackMimeHTTP Server Application

Each web server controls the response sizes that are transfe

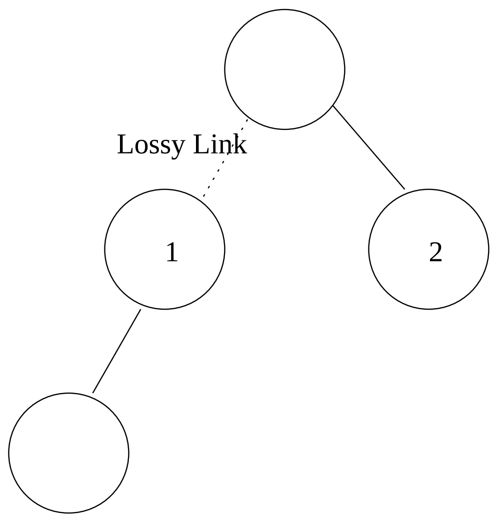
```
remove-all-packet-headers;              # removes all packet headers
add-packet-header IP TCP;               # adds TCP/IP headers
```

```
$packmime no-pm-persistent-rspsz
```

HTTP response size (bytes)

Part VII

Scale


Lossy Link
1
2

42.1.2 Inserting a Loss Module

When studying a protocol (*e.g.*

Delay and Loss Modules Each receiver in a group requires a delay module that reflects its delay with respect to the particular source. When the receiver joins a group, `join-group{}` identifies all session helpers o `session_`. If the

Chapter 43

By defalt, this fla is se to 0

Part VIII

Emulation

Chapter 44

Emulation

This chapter describes the *emulation* facility of *ns*. Emulation refers to the ability to introduce the simulator

When using the emulation mode, a special version of the system scheduler is used: the

```
set intf [$pf1 open readonly]

sett [$pf1 fitrfit]

puts "ro [$pf1ro],nt [$pf1t]"]
```

44.5 An Example

The following code illustrates a small but complete simulat

```
puts "install nets into taps..."
$a0 network $bpf0
$a1 network $bpf1
$a2 network $ipnet

set node0 [$ns node]
set node1 [$ns node]
set node2 [$ns node]

$ns simplex-link $node0 $node2 10Mb 10ms DropTail
$ns simplex-link $node1 $node2 10Mb 10ms DropTail

$ns attach-agent $node0 $a0
$ns attach-agent $node1 $a1
$ns attach-agent $node2 $a2

$ns connect $a0 $a2
$ns connect $a1 $a2

puts "okey"
$ns run
```

44.6 Commands at a glance

Following is a list of emulation related commands:

`$ns_ use-scheduler RealTime`

This command sets up the real-time scheduler. Note that a real-time scheduler should be used with any emulation

Chapter 46

```
dst_portaddr,
seqno,flags,sname);
```

up

down

right

left

up-right

down-right

up-left

down-left

46.1.7 Agent Tracing

This event is very generic, in that it may execute several different procedures at a given time, as long as it is in one line (no more than 256 characters). There may be white spaces in the string which are passed as arguments to the tcl procedure.

l :	link

-t	<time>	time

h : hop

-t	<time>	time
-s	<int>	source id
-d	<int>	destination id
-e	<int>	extent
-a	<int>	attribute
-i	<int>	id
-l	<int>	energy
-c	<strin8993(y)-5.8>	conversation
-x	<comment>	comment
-p	<strin8993(y)-5.8>	p993(y)-5.8acket ytype
-k	<strin8993(y)-5.8>	p993(y)-5.8acket ytype
-R	<double>	wireless broadcast r6393(s)3.56dius
-D	<double>	wireless broadcast duration
-d		
-d		
-i	<inmeve	
-i	<int>aond	
-c	<strin8993(y)-5.8>	tnversamenttr8993(y)-5.8> >yp

ouble> wi-D6(l)0.965521(e)-1.66516(s)3.55944(s)-249.384(b)-5.8887(r)-4.26153(o)-5.8887(a)-1.66638(d)-5.8887(c)-1.66638(a)-1.66393(s)3

f : feature

-t	<time>	time

X : layout lan

-t	<time>	time
-n	<string>	name
-r	<double>	rate
-D	<double>	delay

```
$ns duplex-link-op orient right      ;# orientation is set as right. The order
                                     ;# in which links are created in nam
                                     ;# depends on calling order of this function.
$ns duplex-link-op color "green"
$ns duplex-link-op queuePos 0.5
$ns duplex-link-op label "A"
```

46.2.3 Agent and Features

Agents are used to separate protocol states from nodes. They are always associated with nodes. An agent has a name, which is a unique identifier of the agent. It is shownsn.014(a)-1.66393(s)-309.609(a)-314.833(s)3.56067(q)-5.89115(u)-5.8887(a)-1.66638(r)-4.25

Part X

Other

Chapter 47

Educational use of NS and NAM