NS2
Wireless Network
Simulation

Chang-Gun Lee (cglee@snu.ac.kr)

Assistant Professor

The School of Computer Science and Engineering

Seoul National University

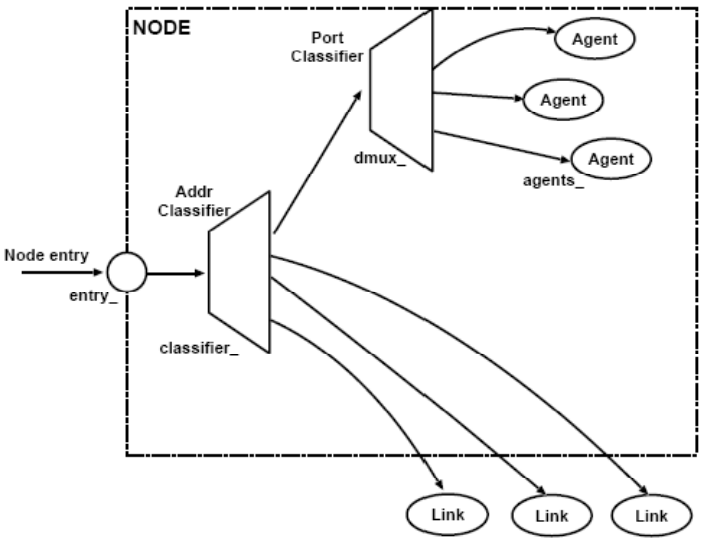
NS Wireless Extension

- Wireless environment is completely different from wired environment
  - for example, nodes are not connected through a dedicated channel
  - Instead, wireless medium is shared by multiple nodes
- New network objects are required
  - Radio interface
  - MAC protocol
  - Routing protocol

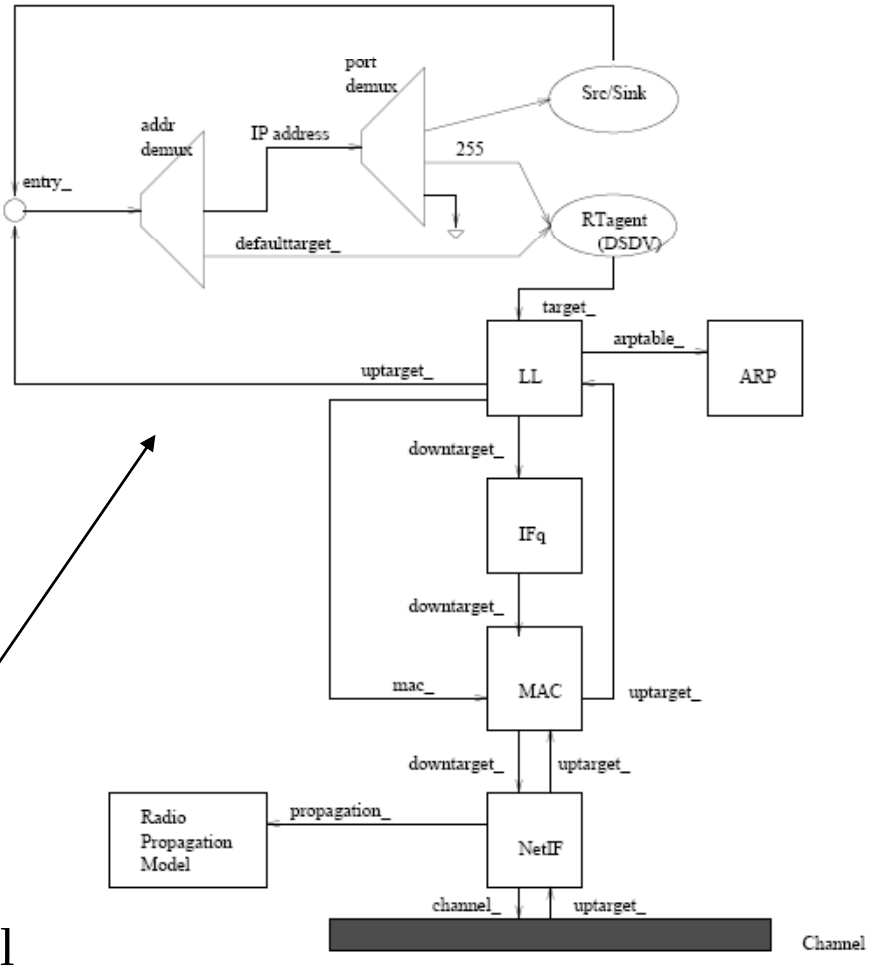
CMU’s Monarch Project

- CMU’s Monarch Projects implemented wireless-related objects to be added into NS-2
- Our version (ns-2.30) already has it!
  - C++ sources
    - ns-2.30/common/mobilenode.cc and mobilenode.h
    - ns-2.30/dsdv, aodv, dsr
    - ns-2.30/mac (802.11, smac, etc)
    - ns-2.30/mobile (radio propagation model, antenna, etc)
  - OTcl sources
    - ns-2.30/tcl/lib/ns-mobilenode.tcl
    - ns-2.30/tcl/mobility/dsdv.tcl, dsr.tcl, etc.

Node vs. Mobile Node



Node



defined in ns-mobilenode.tcl

Node/MobileNode

Wireless Mobile Network Simulation
- simple-wireless.tcl -

```
# Define options
# ======================================
set val(chan)          Channel/WirelessChannel
set val(prop)          Propagation/TwoRayGround
set val(ant)           Antenna/OmniAntenna
set val(netif)         Phy/WirelessPhy
set val(mac)           Mac/802_11
set val(ifq)           Queue/DropTail/PriQueue
set val(ifqlen)        50                ;# max packet in ifq
set val(ll)            LL
set val(rp)            DSDV              ;# routing protocol

set val(nn)            2                 ;# number of mobilenodes
set val(x)         500               ;# X dimension
set val(y)         500               ;# Y dimension
```

Define options that will be used in this tcl file

```
# ==================================
# Main Program
# ==========

#
# Initialize Glo
#
set ns_                          [new Simulator]
set tracefd     [ope             e-wireless.tr w]
$ns_ trace-all $
set nf                           en simple-wireless.nam w]
$ns_ nam                -all-wireless $nf $val(x) $val(y)

# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)


#
# Create God
#
create-god $val(nn)
```

create a topology object that keeps track of movements of mobile nodes

God object store global information about environment. Always needed!

```
#  Create the specified number of mobilenodes [$val(nn)]
#  and "attach" them  to the channel.
#  Here two nodes are created : node(0) and node(1)

# configure node
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace ON

for {set i 0} {$i < $val(nn) } {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 0 ;# disable random motion
}

# Provide initial (X,Y, for now Z=0) co-ordinates for
mobilenodes
#
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0
```

Node configuration:
which type of protocols will be used

Create TWO mobile nodes

Initial coordinates of two nodes

```
#enable node trace in nam
for {set i 0} {$i < $val(nn)} {incr i} {
                $node_($i) namattach $nf
                $ns_ initial_node_pos $node_($i) 20
}

#
# Now produce some simple node movements
# Node_(1) starts to move towards node_(0)
#
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

# Node_(1) then starts to move away from node_(0)
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"

# Setup traffic flow between nodes
# TCP connections between node_(0) and node_(1)

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"
```

Enable node trace in nam

node(1) starts move at 50.0 sec toward (25.0m, 20.0m) at the speed of 15.0m/sec

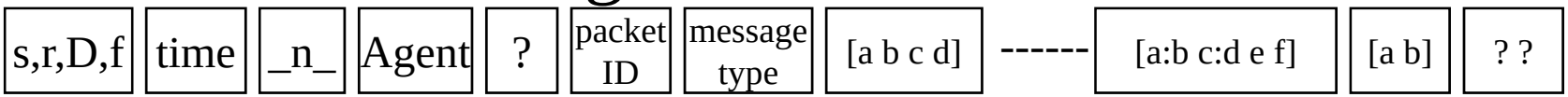
```
#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0 "stop"
$ns_ at 150.01 "puts \"NS EXITING...\" ;
$ns_ halt"
proc stop {} {
    global ns_ tracefd nf
    $ns_ flush-trace
    close $tracefd
    close $nf
}

puts "Starting Simulation..."
$ns_ run
```

tell mobile nodes to reset whuch actually reset their internal network components

Demo
- simple-wireless.tcl -

- ns simple-wireless.tcl
  - output files
    - simple-wireless.nam
    - simple-wireless.tr
- nam simple-wireless.nam (visualization)
- understanding the trace file



s: send
r: receive
D: drop
f: forward

n: node id

RTR: routing agent
IFQ: Interface Que
AGT: Agent

MAC info
a: expected wireless challen access time
b: macid of receiving node
c: macid of sending node
d: ethernet type IP

IP info
a: srcIP
b: srcPort
c: dstIP
d: dstPort
e: ttl
f: nexthopIP

TCP info
a: TCP seqNo
b: actNo

Larger Network
- wireless1.tcl -

- Node movement - from file (scen-3-test)
  - "setdest" tool can generate the random movement file
- Traffic flow - from file (cbr-3-test)
  - "ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]" can generate the random traffic-pattern file

```
set god_ [God instance]
$ns_ at 50.000000000000 "$node_(2) setdest 369.463244915743
170.519203111152 3.371785899154"
$ns_ at 51.000000000000 "$node_(1) setdest 221.826585497093
80.855495003839 14.909259208114"
$ns_ at 33.000000000000 "$node_(0) setdest 89.663708107313
283.494644426442 19.153832288917"
$god_ set-dist 1 2 2
$god_ set-dist 0 2 3
$god_ set-dist 0 1 1
$node_(2) set Z_ 0.000000000000
$node_(2) set Y_ 199.373306816804
$node_(2) set X_ 591.256560093833
$node_(1) set Z_ 0.000000000000
$node_(1) set Y_ 345.357731779204
$node_(1) set X_ 257.046298323157
$node_(0) set Z_ 0.000000000000
$node_(0) set Y_ 239.438009831261
$node_(0) set X_ 83.364418416244
```

scen-3-test

```
#
# 0 connecting to 2 at time 127.93667922166023
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 4.0
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 127.93667922166023 "$cbr_(0) start"

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp
$ns_ attach-agent $node_(2) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 150.00000000000000 "$ftp start"
```

cbr-3-test

Larger Network
- wireless1.tcl -

```
# ======================================
# Define options
# ======================================
set val(chan)       Channel/WirelessChannel
set val(prop)       Propagation/TwoRayGround
set val(netif)      Phy/WirelessPhy
set val(mac)        Mac/802_11
set val(ifq)        Queue/DropTail/PriQueue
set val(ll)         LL
set val(ant)        Antenna/OmniAntenna
set val(x)            670   ;# X dimension of the topography
set val(y)            670   ;# Y dimension of the topography
set val(ifqlen)        50           ;# max packet in ifq
set val(seed)          0.0
set val(adhocRouting)   DSR
set val(nn)            3            ;# how many nodes are simulated
set val(cp)           "../mobility/scene/cbr-3-test"
set val(sc)           "../mobility/scene/scen-3-test"
set val(stop)          400.0          ;# simulation time
```

…. similar as before …..

```
#
# Define node movement model
#
puts "Loading connection pattern..."
source $val(cp)

#
# Define traffic model
#
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

   # 20 defines the node size in nam, must adjust it according to
your scenario
   # The function must be called after mobility model is defined

   $ns_ initial_node_pos $node_($i) 20
}
```

… similar as before ….

Demo
- wireless1.tcl -

- ns wireless1.tcl
  - output files
    - wireless1.nam
    - wireless1.tr
- nam wireless1.nam (visualization)
- understanding the trace file: wireless1.tr