Presentation contents

Wireless multi-hop networks

• How is the radio signals propagation modeled during the simulation ?

- Node entities in NS2
- Emulating radio signals propagation
- Briefly mobility

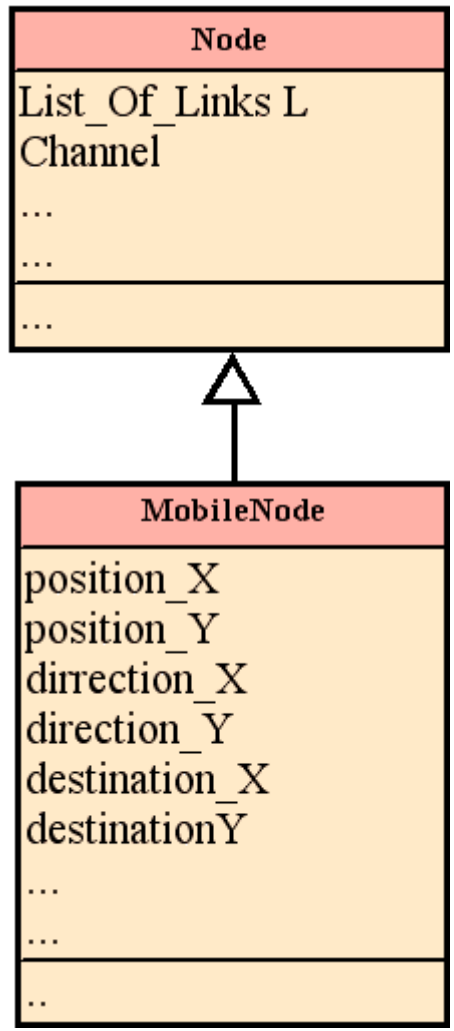
Mobile nodes in NS2

Mobile node (1)

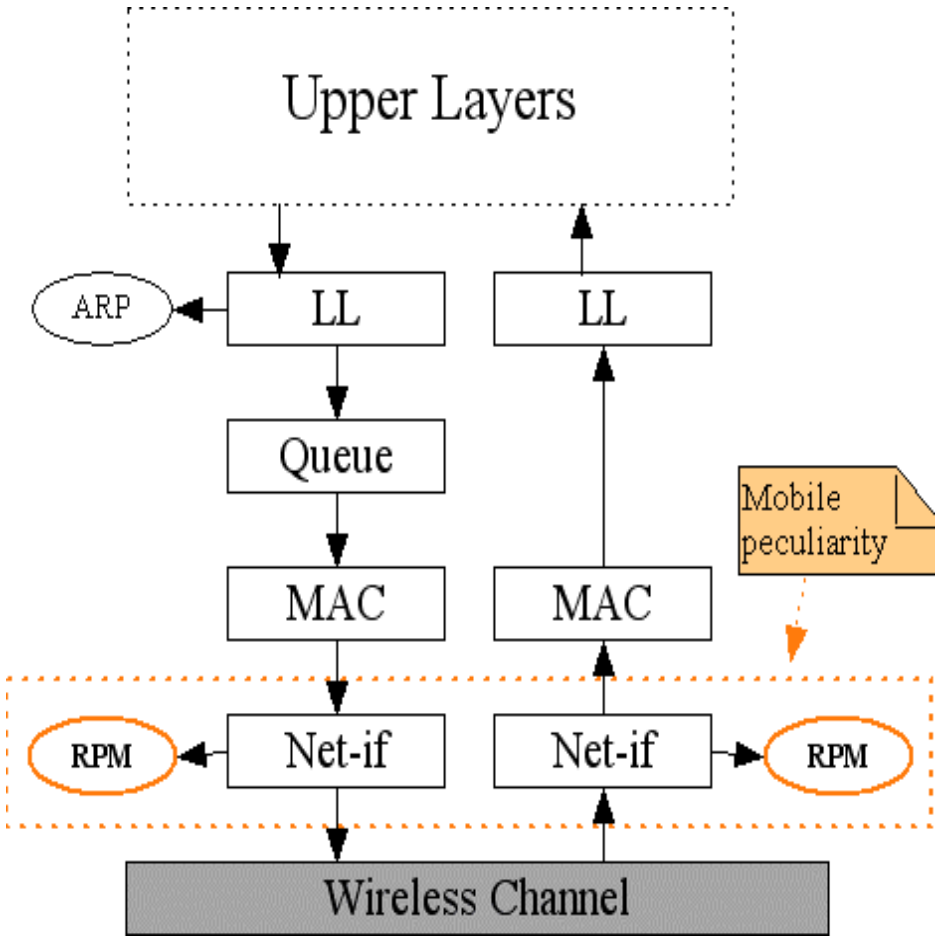
- A mobile node is a compound object:
  - the node object itself (C++)
    /ns/mobilenode.{cc,h} & node.{cc,h}
  - the network components within the mobile node (OTCL)
    /ns/tcl/lib/ns-mobilenode.tcl
- An additional set of modules implements, in C++, the routing agents (protocols) for mobile networking:
  /ns/tcl/mobiliy/dsdv.{cc,h}
  dsr, tora, and aodv are also implemented

Mobile node (2) - the node obj

- A mobile node is a basic node with added functionalities for:
  - moving
- Furthermore, it communicates via a wireless channel. Hence, it does not really have any list of links to which is connected to.
- 2 ways of creating nodes movement.
  (briefly explained later)



Mobile node (3)



- We are solely looking at the bottom part of a more complex stack. *Radio Signals Propagation* happens just over the channel...
- The depicted network stack is created, and plumbed for each mobile node at its born time with an OTCL routine.
- The routing agent mentioned earlier, resides on the top of the shown stack.

Mobile node (4) - tcl configuration

- Two possible API to create a mobile node with certain feautures:
  - Old - creates a default mobile node object. We are only allowed to specify the routing protocol: ($rp)

```
set mnode [$rp-create-mobile-node $id]
```

  - New - we can specify a more exhaustive set of features to be set:

```
$ns_ node-config -adhocRouting $val(adhocRouting)
        -llType $val(ll) \
        -macType $val(mac) \ ;# Either 802.11 or TDMA
        -ifqType $val(ifq) \ ;# priority
        -ifqLen $val(ifqlen) \ ;# length
        -antType $val(ant) \   ;# MultiDir/ Omni
        -propType $val(prop) \ ;# RPM
        -phyType $val(netif) \ ;# WirelessPhy
        -channeltype $val(chan) \
        -channel $chan \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace OFF \

set node_($j) [$ns_  node]   ;# Real creation in memory
```

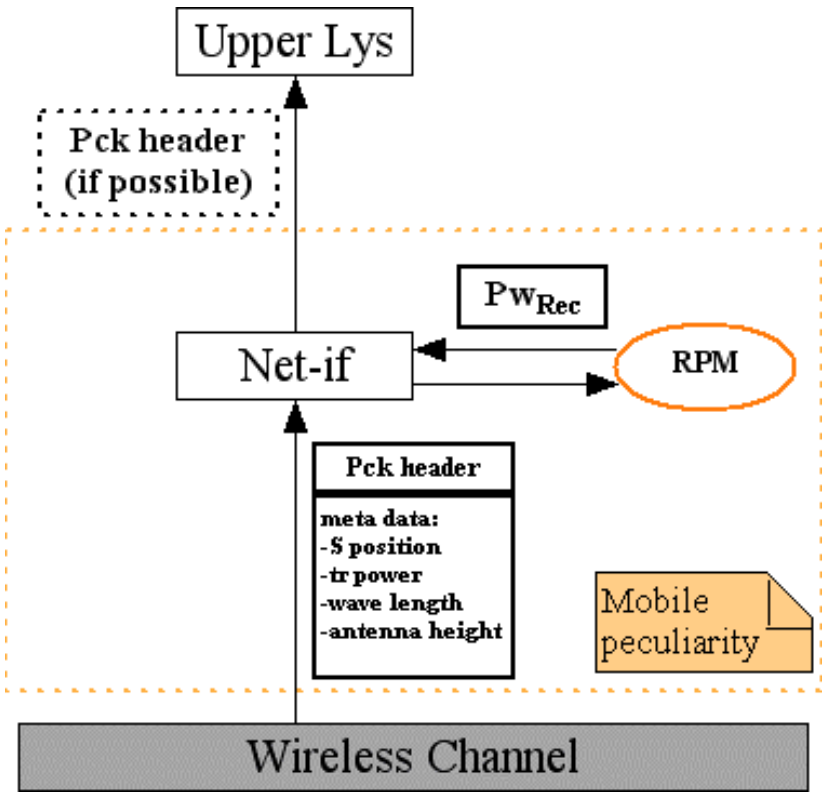
Radio Propagation Models

Radio signals propagation

- NS assumes that in wireless, multihop networks each packet arrives at the receiver side whenever the sender got the channel IDLE for the needed time $t=t_{tr}+t_{pr}$. This regardless the distance between the nodes.

- Hence, it is a receiver's duty to investigate whether or not the packet must climb the stack toward the application layer.

Receiver side stack



- When a pck is received, it is computed a prediction for the received signal power:
  $P_{Rec} = f_{pm}(meta\_data)$
- The destiny of the "received" pck can alternatively be:
  - going up through the stack
    $P_{Rec} > RX_{threshold}$
  - be dropped and traced as such
    $CS_{threshold} < P_{Rec} < RX_{threshold}$
  - be dropped without tracing it
    $P_{Rec} < CS_{threshold}$

Signals interference problem

- Interference is assumed when:
  - $P_{Rec} > CS_{threshold}$
- Multiple transmissions toward the same node:
  - NS assumes that if a packets PB arrives at a node N when it is correctly receiving another packet PA, their reception is ONLY compromised if $P_{rec}(PB) > CS_{threshold}$
- This is correct as long as we are sure that not more than two packets can arrive at the same node.

  In case that two pcks PB and PC arrive (with a weak power) while N is correctly receiving PA, ns2 thinks that PA reception is not compromised when:

  $-P_{rec}(PB) < CS_{threshold}$

  $-P_{rec}(PC) < CS_{threshold}$

  $-P_{rec}(PB) + P_{rec}(PC) > CS_{threshold}$
- It should compromise the reception of PA but it doesn't !! Other simulators as Opnet, Qualnet, Glomosim correctly address the situation

Setting $RX_{threshold}$

- An external application *(/ns/indep-utils/propagation/ threshold.cc)* is provided with the ns source files. It computes the value for the receiving threshold. It take as input:
  - *Pr <transmitted-power>*
  - *Gt <transmit-antenna-gain>*
  - *Gr <receive-antenna-gain>*
  - *L <system-loss-coefficient>*
  - *p <ratio-prop-model>*
  - *d <distance>*

- It gives in output the RX_thresh value to be set if we want to have correct reception from nodes transmitting at power *Pr* within a round area of ray *d*.

```
Phy/WirelessPhy set RXThresh_ <value>
```

- $RX_{threshold}$ is inverse proportional to *d*

Free space PM

- It predicts the radio signals received power on the direct path joining S/R considering:
  - P_t the power transmitted
  - G_t, G_r repectively the gain of the transmitter and receiver antennas. (their value is 1 by default)
  - λ the wave length
  - d the distance between S/R
  - $L >= 1$ a network pathloss coefficient (1 by default)
- The formula is:

$$P_{rec\text{-}FS}(d) = \frac{P_t G_t G_r \lambda^2}{L(4\pi d)^2}$$

- `$ns_ node-config -propType Propagation/FreeSpace`

Two ray ground PM

- In the reality, a receiver hears a received power which is the sum of the signal on the direct path and on the ground reflection path. TwoRay attempts to model this fact:

$$p_{r-trg}(d) = \begin{cases} p_{r-FS}(d) & if\ (d < d_{trhesh}) \\ \dfrac{P_t G_t G_R h_t h_r}{d^4 L} & Else \end{cases}$$

- d_{thresh} is the distance such that: $p_{r\text{-}FS}(d) = p_{r\text{-}trg}(d)$

$$d_{thresh} = \frac{4\pi h_t h_r}{\lambda}$$

- `$ns_ node-config -propType Propagation/TwoRayGround`

Shadowing PM

- It's a more sophisticated model. It represents the fact that the received power at a certain distance is a random variable due to multipath effect (fading effect):

$$p_{R-SW}(d) = p_{rec-FS}(d_0)\left(\frac{d}{d_0}\right)^{\beta} 10^{X}$$

Where:

$$X(x): \left\{ x \in [-\infty, +\infty] \mid P(x) = N(0, \sigma^2) \right\}$$

- `$ns_ node-config -propType Propagation/Shadowing`

Path loss β, Std dev σ, and d_0 must be previously set

Briefly mobility

Nodes movement (1)

- Mobiles move within a certain topology (width, and height must be specified)

```
set topo [new Topography]
$topo load_flatgrid $width $height
```

  a topography instance must be passed to the method which creates the mobile node: `-topoInstance $topo \`

- Nodes' initial location and movements directives are usually specified in a separated "scenario file"

```
$node set X_ <x0> ;# initial abscissa (0 by default)
$node set Y_ <y0> ;# initial ordinate (0 by default)
$ns at $time $node setdest <x1> <y1> <speed[m/s]>
```

- Nodes position is normally updated whenever it is needed to be known. Alternatively it can be regularly updated as :

```
proc update_position { dt } {
    $node log-movement
    $ns_ at [expr [$ns_ now]+dt] "update_position $n $dt"
}
```

Nodes movement (2)

- Scenario files, to obtain a motion that resembles certain mobility models, are usually generated by self-written scripts.
- The network simulator provides an external application (*/ns/indep-utils/cmu-scen-gen/setdest/setdest.cc*) to generate movement directives according to RWP:
  `./setdest -n <num_of_nodes> -p <max_pause_t> -s <maxspeed> -t <simtime> -x <maxx> -y <maxy> > <out-scenario-file>`
  it assumes the nodes to be named as *node_(index)*
- Instead of having a scenario file, we could just be happy to instruct the simulator to make the nodes move randomly without giving it any further instructions:

```
$node random-motion 1 ;# if we were using a scenario file we should
                       ;# have set it to 0 instead
```

Position updating

- Significant attributes at C++ level for a mobile node are:
  ```
  PosX, PosY, [PosZ]
  DestX, DestY, [DestZ]
  DirX, DirY, [DirZ]
  ArrivalTime
  PreviusUpdateTime
  ```
- the *position* attributes and the PreviousUpdateTime are assigned every time that an update event occures.
- The *destination,* and the *direction* attributes as well as the *ArrivalTime* are assigned whenever the *setdest* method is called

Thank you!

Any question?

Write to: Stefano Marinoni
marstefy@tcs.hut.fi
Office: T-B235 lab for theoretical CS