Network Simulator :Ns-2:

Ns Functionality

- Discrete event simulator
- Traffic models and applications
  - Web, FTP, telnet, audio, sensor nets
- Transport protocols
  - TCP (Reno, SACK, etc), UDP, multicast
- Routing and queueing
  - static routing, DV routing, multicast, ad-hoc routing
  - queueing disciplines: drop-tail, RED, FQ
- Link layer
  - wired, wireless, satellite
- Infrastructure
  - tracing, visualization, error models, etc
  - modify or create your own modules

Ns Software Structure: C++ and OTCL

- Uses two languages
- C++ for packet-processing
  - per packet processing
  - fast to run, detailed, complete control
- OTCL for control [our focus]
  - simulation setup, configuration, occasional actions
  - fast to write and change

Creating a Basic Ns Model

- Create the event scheduler
- Create nodes and links
- Create connection
- Create traffic sources/sinks
- Enable tracing

Creating Event Scheduler

- Create scheduler
  - set ns [new Simulator]
- Schedule event
  - $ns at <time> <event>
  - <event>: any legitimate Ns/TCL commands
- Start scheduler
  - $ns run

Creating Network (Nodes + Links)

- Nodes
  - **set n0 [$ns node]**
  - **set n1 [$ns node]**
- Links: connect together two nodes
  - **$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>**
  - <delay> determines propagation delay
  - <queue_type> determines queueing policy
    - **DropTail, RED, CBQ, FQ, SFQ, DRR**

Transport and Traffic Models

- Two layer approach
- Transports:
  - TCP, UDP, multicast, etc.
  - transport protocol instances attach to nodes
- Traffic (applications): (*known as agents*)
  - Web, ftp, telnet, audio, etc.
  - application instances attach to transport protocol instances
  - generates traffic into transport protocol

Creating Transport Channels: UDP

- source and sink
  - set u_src [new Agent/UDP]
  - set u_dst [new Agent/NULL]

- attach them to nodes, then connect to each other
  - $ns attach-agent $n0 $u_src
  - $ns attach-agent $n1 $u_dst
  - $ns connect $u_src $u_dst

Creating Transport Channels: TCP

- source and sink
  - set t_src [new Agent/TCP/Newreno]
  - set t_dst [new Agent/TCPSink]

- attach to nodes and each other
  - $ns attach-agent $n0 $t_src
  - $ns attach-agent $n1 $t_dst
  - $ns connect $t_src $t_dst

Creating Traffic over TCP Channels

- FTP
- create traffic model
  - set ftp [new Application/FTP]
  - default is "infinite" file size
- attach to TCP channel
  - $ftp attach-agent $t_src
- schedule start time
  - $ns at <time> "$ftp start"

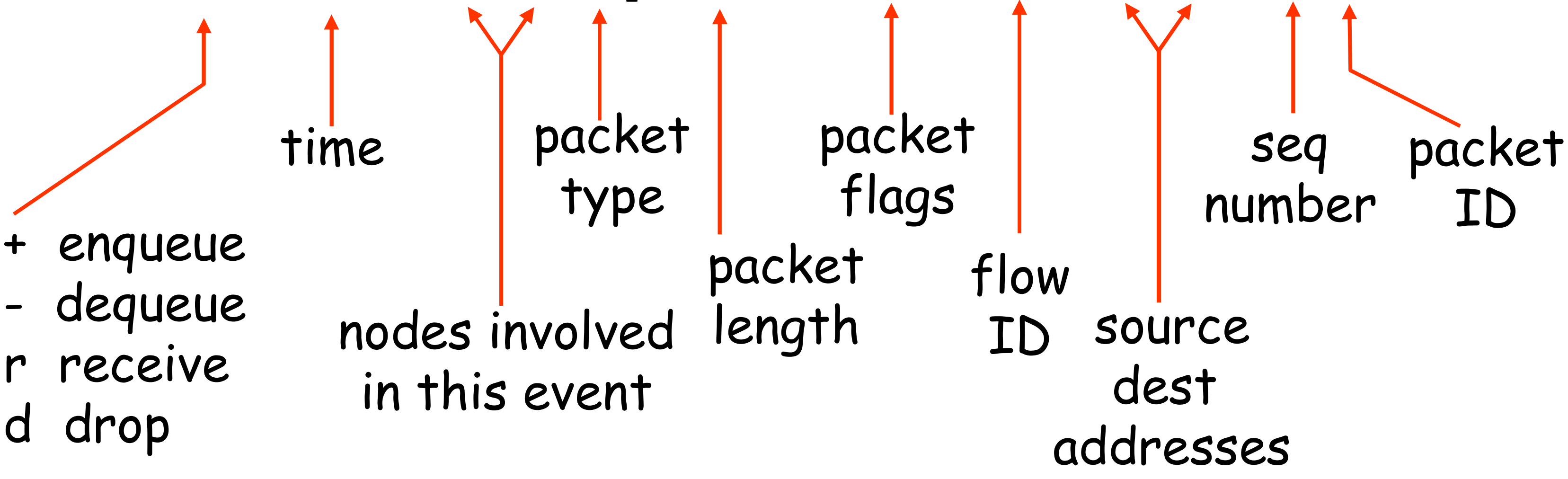
Creating Traffic over UDP Channels

- CBR
  - set cbr [new Application/Traffic/CBR]
  - $cbr set packetSize_ 512
  - $cbr set interval_ 0.250
  - $cbr attach-agent $u_src
  - $ns at <time> “$cbr start”

Tracing

- Trace packets on individual links
- Tracefile format:

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src>
   <dst> <seqno> <aseqno>
    + 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
    - 1       0 2 tcp 900 ------- 1 0.0 3.1 7 15
    r 1.00234 0 2 tcp 900 ------- 1 0.0 3.1 7 15
```



+ enqueue
- dequeue
r receive
d drop

time

nodes involved in this event

packet type

packet length

packet flags

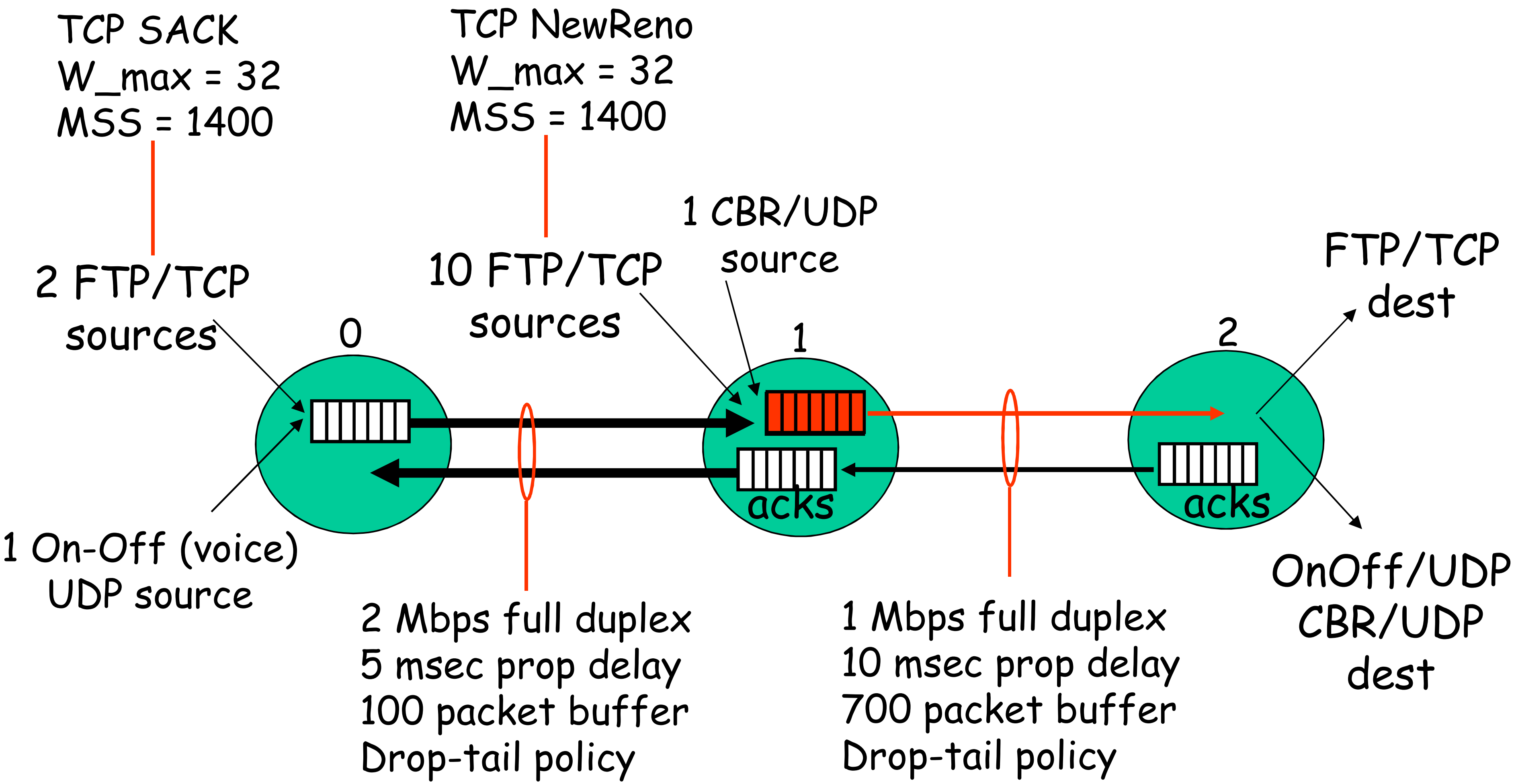
flow ID

source dest addresses

seq number

packet ID

Walk-through example



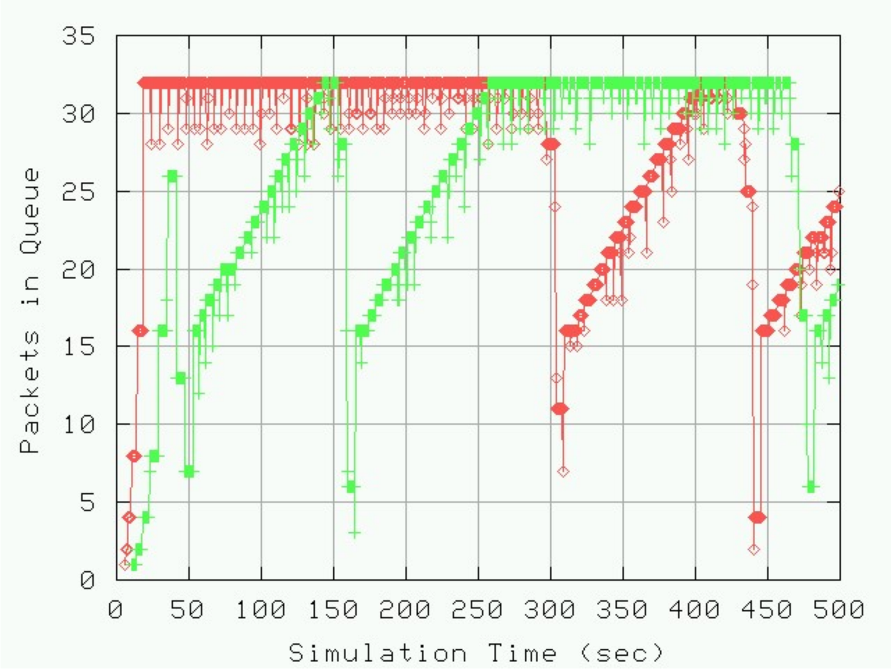
Ns Trace file : NS-trace.txt

```
...
+ 11.533441 1 2 tcp 1440 ------- 12 1.2 2.4 96 2092
r 11.535694 1 2 tcp 1440 ------- 12 1.2 2.4 65 1527
- 11.537214 1 2 exp 180 ------- 100 0.2 2.13 284 1528
- 11.538654 1 2 cbr 1440 ------- 101 1.11 2.14 155 1530
r 11.547214 1 2 tcp 1440 ------- 12 1.2 2.4 66 1529
+ 11.54728 1 2 tcp 1440 ------- 12 1.2 2.4 97 2095
r 11.548654 1 2 exp 180 ------- 100 0.2 2.13 284 1528
+ 11.55 1 2 cbr 1440 ------- 101 1.11 2.14 211 2096
- 11.550174 1 2 tcp 1440 ------- 12 1.2 2.4 67 1534
r 11.560174 1 2 cbr 1440 ------- 101 1.11 2.14 155 1530
- 11.561694 1 2 exp 180 ------- 100 0.2 2.13 285 1532
+ 11.56222 1 2 tcp 1440 ------- 12 1.2 2.4 98 2097
- 11.563134 1 2 tcp 1440 ------- 12 1.2 2.4 68 1537
r 11.571694 1 2 tcp 1440 ------- 12 1.2 2.4 67 1534
r 11.573134 1 2 exp 180 ------- 100 0.2 2.13 285 1532
- 11.574654 1 2 exp 180 ------- 100 0.2 2.13 286 1536
...
```

Graphic

✓Xgraph

✓ Gnuplot



- ❒ Red - TCP flow 1
- ❒ Green - TCP flow 2

Wireless extensions

Simulation of two wireless nodes

- A simple simulation for two nodes and no infrastructure we need to configure the nodes first (in ns2.1b7).

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

Simple wireless simulation

- Array opt( )

```
set opt(chan)   Channel/WirelessChannel  ;# channel type
set opt(prop)   Propagation/TwoRayGround ;# radio-propagation
set opt(netif)  Phy/WirelessPhy          ;# network interface
set opt(mac)    Mac/802_11               ;# MAC type
set opt(ifq)    Queue/DropTail/PriQueue  ;# interface queue type
set opt(ll)     LL                       ;# link layer type
set opt(ant)    Antenna/OmniAntenna      ;# antenna model
set opt(ifqlen)         50               ;# max packet in ifq
set opt(rp)             DSDV             ;# routing protocol
```

Configuring the wireless/phy

- Phy/WirelessPhy set CPThresh_ 10.0
- Phy/WirelessPhy set CSThresh_ 1.559e-11
- Phy/WirelessPhy set RXThresh_ 3.652e-10
- Phy/WirelessPhy set Rb_ 2*1e6
- Phy/WirelessPhy set Pt_ 0.2818
- Phy/WirelessPhy set freq_ 914e+6

Important initialization variables

- `set topo    [new Topography]`
- `$topo load_flatgrid $opt(x) $opt(y) ;`
  - opt(x) and opt(y) are the dimensions of the simulation area
- `set chan    [new $opt(chan)]`
- `Create-god  Number_of_entities`

Creating the mobile nodes

- Set mobile node
  - `set node [$ns_ node]`
- Attach it to the topography
  - `$node topography $topo`
- Determine the motion pattern and initial position
- `$node random-motion 0    #; no random`
- `$node set Z_ 0.000000`
- `$node set Y_ 100.0000`
- `$node set X_ 2.000000`

Adding movement to the nodes

- Random set to 0
  - `$ns at <time> “$node setdest <x> <y> <speed>”`
- Random set to 1
  - `$ns at <time> “$node start”`