EEC 687/787 Mobile Computing

Ns-2 Laboratory

Prof. Chansu Yu

ns-2 Overview

- What is ns-2?
  - Abbreviation of Network Simulator
  - Discrete event simulator targeted at networking (wired and wireless) research
  - Basically, a TCL interpreter
- Where to get?
  - Free and open source
  - ns website http://www.isi.edu/nsnam/ns/
- Working platforms
  - Most UNIX or UNIX-like systems; e.g. Linux
  - Windows (using cygwin)

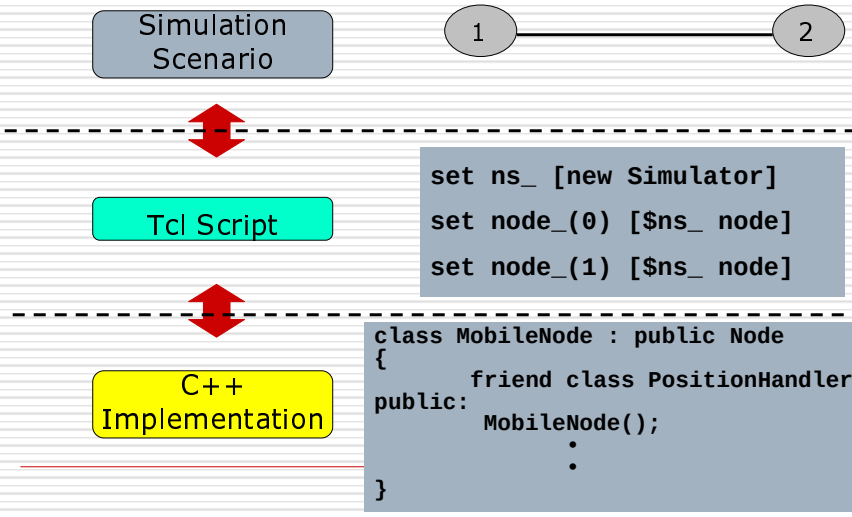
Download and Install ns-2

- Reading
  - Wireless and Mobility Extensions to ns-2 (http://www.isi.edu/nsnam/ns/tutorial/nsindex.html).
  - Ns2 manual "Mobile networking in ns," Ch. 16 (http://www.isi.edu/nsnam/ns/doc/index.html)
- Download ns-2 (version 2.34) from http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/ and put it to your desirable folder.
- Run the following commands at /home/student :
  - % gunzip ns-allinone-2.34.tar.gz
  - % tar –xvf ns-allinone-2.34.tar
- Run the script at /home/student/ns-allinone-2.34 : "% ./install"
  - This installation script will check your Linux environment, compile and install your ns-2 system.
- For detail information for install, see the appendix.

ns-2 uses two languages? (Tcl & C++)

- C++: Detailed protocol simulations require systems programming language
  - byte manipulation, packet processing, algorithm implementation
  - Run time speed is important
  - Turn around time (run simulation, find bug, fix bug, recompile, re-run) is slower
- Tcl: Simulation of slightly varying parameters or configurations
  - quickly exploring a number of scenarios
  - iteration time (change the model and re-run) is more important

ns-2 Environment



Basic TCL

(http://tmml.sourceforge.net/doc/tcl/index.html)

```
set a 43
set b 27
set c [expr $a + $b]
set d [expr [expr $a - $b] * $c]
for {set k 0} {$k < 10} {incr k} {
    if {$k < 5} {
        puts "k < 5, pow= [expr pow($d, $k)]"
    } else {
        puts "k >= 5, mod= [expr $d % $k]"
    }
}
```

Hello World - Interactive Mode

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

Hello World - Passive Mode

simple.tcl

```
set ns [new Simulator]
$ns at 1 “puts \“Hello World!\””
$ns at 1.5 “exit”
$ns run
```

% **ns simple.tcl**
Hello World!

%

Protocols or Controls Implemented in ns2

- Transport layer (traffic agent)
  TCP; UDP
- Network layer (routing agent)
  - Wired
    Distance vector; Link state (patch needed)
  - Wireless
    AODV; DSR; DSDV; TORA
- Interface queue
  FIFO queue; DropTail queue; Priority queue; etc.
- Logic link control layer
  IEEE 802.2; ARP

Protocols or Controls Implemented in ns2 (cont.)

- MAC layer
  - Wired
    - IEEE 802.3 (CSMA/CD)
  - Wireless
    - IEEE 802.11 (CSMA/CA)
      - DCF
      - PCF (partially implemented)
- Physical layer
  - Wired
    - IEEE 802.3
  - Wireless
    - IEEE 802.11
      - DSSS (Direct Sequence Spread Spectrum)
      - FHSS (Frequency-Hopping Spread Spectrum); not implemented
      - IR (Infrared); not implemented

Protocols or Controls Implemented in ns2 (cont.)

- Wireless channel
  - Friss-space model
  - Two-ray ground model
  - Shadowing model
  - Fading model (patch needed)
  - Omni directional antenna

How to Use ns-2?

- Design simulation

  Determine simulation scenario, parameters.
- Build ns-2 script using tcl

  If necessary implement algorithm using C++.
- Run simulation

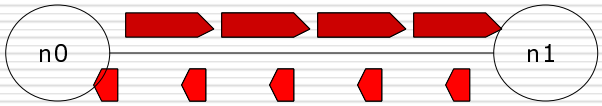
  For convenience use shell batch file.
- Analyze simulation results

  Use shell command or programming languages.

Simulation with ns-2

- Creating the event scheduler
- Creating network: nodes, links & queue
- Computing routes
- Creating connection
- Creating traffic
- Inserting errors
- Tracing
- Wireless Support

Example Script




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

$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail

set tcp [$ns create-connection TCP $n0 TCPSink $n1 0]
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 1.2 “exit”
$ns run
```

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: Node, Link & Queue

- Nodes
  - set n0 [$ns node]
  - set n1 [$ns node]
- Links & Queuing
  - $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>

where, <queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR

Creating Network: LAN

- LAN
  - $ns make-lan <node_list> <bandwidth> <delay> <ll_type> <ifq_type> <mac_type> <channel_type>

  where, <ll_type>: LL
  <ifq_type>: Queue/DropTail,
  <mac_type>: MAC/802_3
  <channel_type>: Channel

Computing routes

- Unicast
  - $ns rtproto <type>
  - <type>: Static, Session, DV, cost, multi-path
- Multicast
  - $ns multicast (right after [new Simulator])
  - $ns mrtproto <type>
  - <type>: CtrMcast, DM, ST, BST

Creating Connection: UDP

- 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

Creating Connection: TCP

- TCP
  - set tcp [new Agent/TCP]
  - set tcpsink [new Agent/TCPSink]
  - $ns attach-agent $n0 $tcp
  - $ns attach-agent $n1 $tcpsink
  - $ns connect $tcp $tcpsink

Creating Traffic: On Top of TCP

- FTP
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp
  - $ns at <time> “$ftp start”
- Telnet
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp

Creating Traffic: On Top of UDP

- CBR
  - set src [new Application/Traffic/CBR]
- Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]

Creating Traffic: Trace Driven

- Trace driven
  - set tfile [new Tracefile]
  - $tfile filename <file>
  - set src [new Application/Traffic/Trace]
  - $src attach-tracefile $tfile
- <file>:
  - Binary format
  - inter-packet time (msec) and packet size (byte)

Inserting Errors

- Creating Error Module
  - set loss_module [new ErrorModel]
  - $loss_module set rate_ 0.01
  - $loss_module unit pkt
  - $loss_module ranvar [new RandomVariable/Uniform]
  - $loss_module drop-target [new Agent/Null]
- Inserting Error Module
  - $ns lossmodel $loss_module $n0 $n1

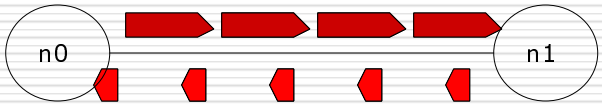
Tracing

- Trace packets on all links
  - $ns trace-all [open test.out w]

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>
     + 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
```

- Trace packets on all links in nam-1 format
  - $ns namtrace-all [open test.nam w]

Example Script (again)

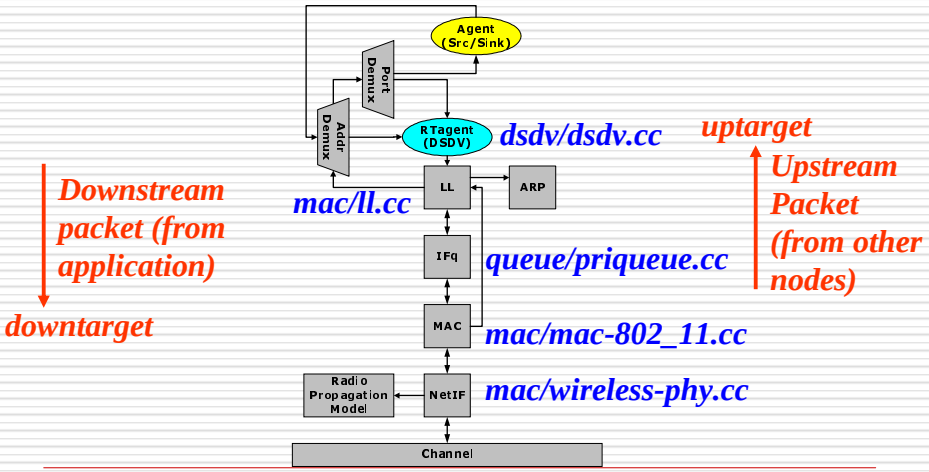



```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail

set tcp [$ns create-connection TCP $n0 TCPSink $n1 0]
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 1.2 “exit”
$ns run
```

Mobile Node Modules



Wireless Support: Setup

- set ns [new Simulator]
- set chan [new Channel/WirelessChannel]
- set prop [new Propagation/TwoRayGround]
- set topo [new Topography]
- $topo load_flatgrid <length> <width>
- $prop topology $topo

Wireless Support: MobileNode

- Creating mobile nodes
  - set mnode [<routing>-create-mobile-node <node_id>]

  where, <routing>: dsdv or dsr

- Node coordinates
  - $mnode set X_ <x>
  - $mnode set Y_ <y>
  - $mnode set Z_ 0

Wireless Support: Movement

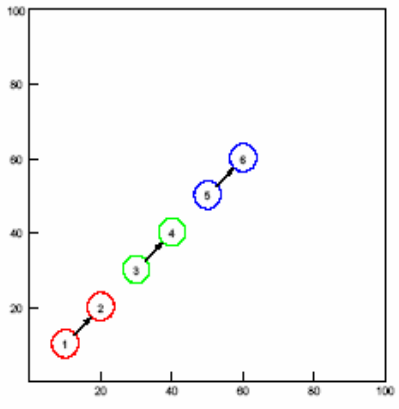
- Specified
  - $ns at 1.0 “$mnode setdest <x> <y> <speed>”

- Random
  - $ns at 1.0 “$mnode start”

Run the Simulation

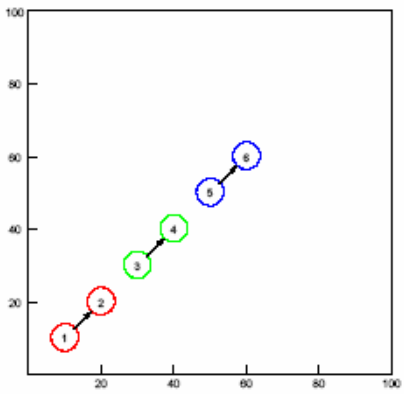
- Download ns-2 for the impatient (http://icapeople.epfl.ch/aad/teaching/ns/ns.html, expired)
  - ex6sta.tcl
  - fil.awk, fil2.awk, fil4.awk, fil6.awk

Mobile Network Simulation with ns-2



• 6 nodes
• Located (10,10), (20,20), …
• CBR (Constant bit rate) traffic based on UDP 1→2, 3→4, 5→6

Mobile Network Simulation with ns-2



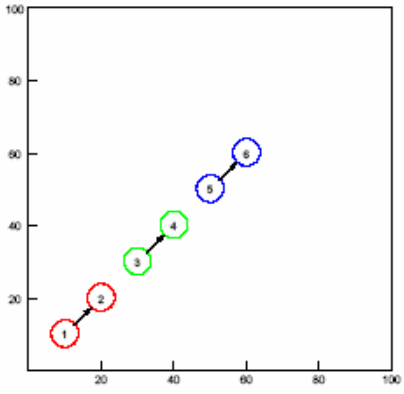
set ns_ [new Simulator]
set opt(nn) 6

• Defining 6 nodes

set WT1 [$ns_ node $1]
…
set WT6 [$ns_ node $6]
=>

for {set i 1} {$i<=$opt(nn)} {incr i} {
set WT($i) [$ns_ node $i]
}

Mobile Network Simulation with ns-2

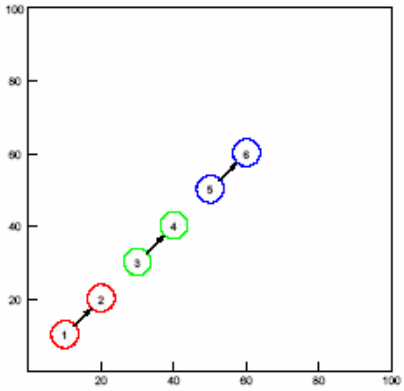


• Located (10,10), (20,20), …

$WT1 set X_ 10
$WT1 set Y_ 10
$WT1 set Z_ 0.0
…
$WT6 set X_ 60
$WT6 set Y_ 60
$WT6 set Z_ 0.0
=>

for {set i 1} {$i<=$opt(nn)} {incr i} {
$WT($i) set X_ [expr 10*$i]
$WT($i) set Y_ [expr 10*$i]
$WT($i) set Z_ 0.0
}

Mobile Network Simulation with ns-2



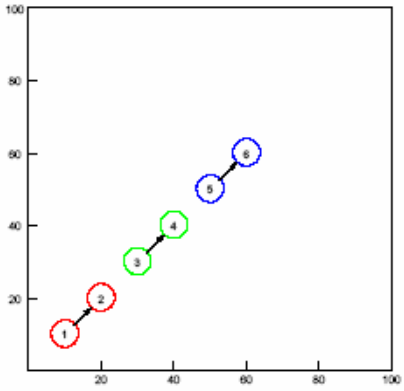
•CBR (Constant bit rate) traffic based on UDP 1→2, 3→4, 5→6

set udp1 [new Agent/UDP]
$ns_ attach-agent $WT1 $udp1

set sink1 [new Agent/Null]
$ns_ attach-agent $WT2 $sink1

$ns_ connect $udp1 $sink1
…

Mobile Network Simulation with ns-2

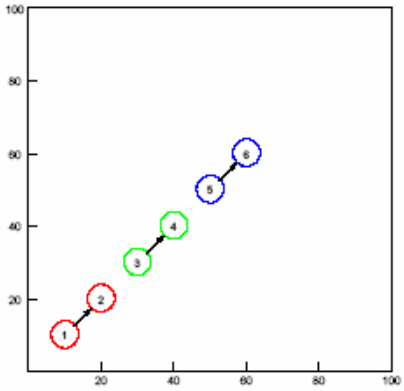


•CBR (Constant bit rate) traffic based on UDP 1→2, 3→4, 5→6

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 1000
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

$ns_ at 20.0 “$cbr1 start”
$ns_ at 150.0 “$cbr1 stop”
…

Mobile Network Simulation with ns-2



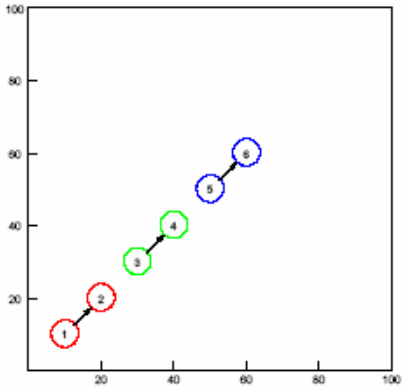
•CBR (Constant bit rate) traffic based on UDP 1→2, 3→4, 5→6

```
for {set i 1} {$i<=$opt(nn)} {incr i 2} {
  set udp($i) [new Agent/UDP]
  $ns_  attach-agent $WT($i) $udp($i)

  set sink($i) [new Agent/Null]
  $ns_  attach-agent $WT([expr $i+1]) $sink($i)

  $ns_  connect $udp($i) $sink($i)
```

Mobile Network Simulation with ns-2



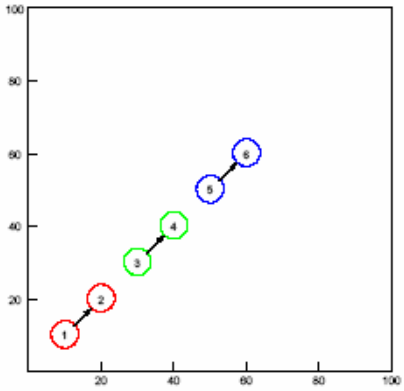
•CBR (Constant bit rate) traffic based on UDP 1→2, 3→4, 5→6

```
  set cbr($i) [new Application/Traffic/CBR]
  $cbr($i) set packetSize_ 1000
  $cbr($i) set interval_ 0.005
  $cbr($i) attach-agent $udp($i)

  $ns_  [expr 20.0*$i] "$cbr($i) start"
  $ns_  at $opt(stop) "$cbr($i) stop"
}
```

set opt(stop) 150

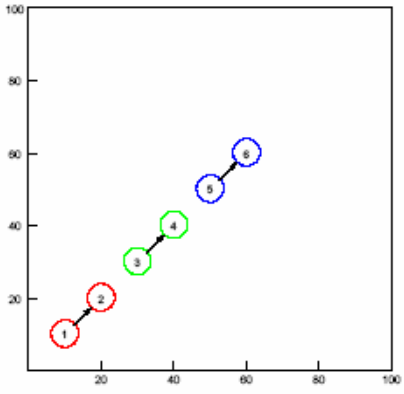
Mobile Network Simulation with ns-2



• Node configuration

```
$ns_   node-config
  -adhocRouting   DumbAgent \
  -llType         LL \
  -macType        Mac/802_11 \
  -ifqType        Queue/DropTail/PriQueue \
  -ifqLen         50 \
  -antType        Antenna/OmniAntenna \
  -propType       Propagation/FreeSpace \
  -phyType        Phy/WirelessPhy \
  -channelType    Channel/WirelessChannel \
  -topoInstance   $topo \
```

Mobile Network Simulation with ns-2



• Topology configuration

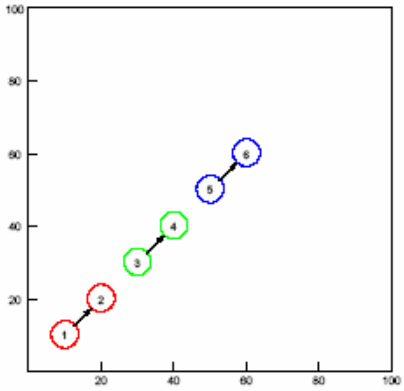
```
set topo   [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
```

set opt(x)	*100*
set opt(y)	*100*

• GOD (General operations director)

```
create-god  $opt(nn)
```

Mobile Network Simulation with ns-2



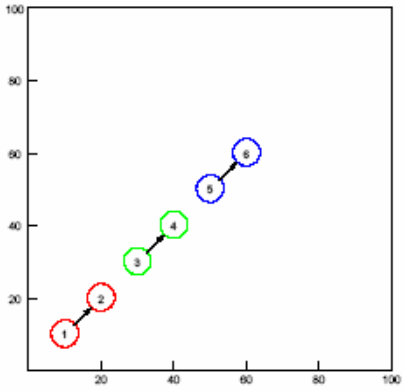
• Node configuration (cont'd)

```
$ns_ node-config
 …
  -agentTrace      ON \
  -routerTrace     OFF \
  -macTrace        OFF \
  -movementTrace OFF

set tracefd  [open $opt(tr) w]
$ns_  trace-all $tracefd
```

set opt(tr) out.tr

Mobile Network Simulation with ns-2



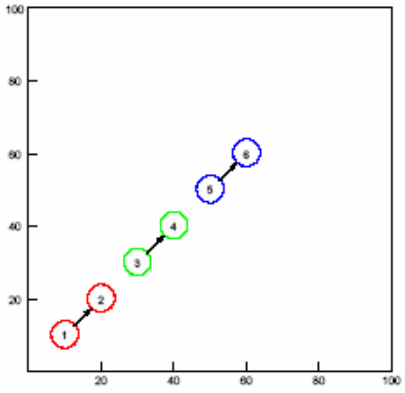
• End procedure & Start ns2 simulation

```
for {set i } {$i <= $opt(nn) } {incr i} {
   $ns_  at $opt(stop).0000010 "$WT($i) reset";
}

$ns_  at $opt(stop).2 "$ns_  halt"


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

Mobile Network Simulation with ns-2



- Adding finish procedure

```
$ns_ at $opt(stop).1 "finish"
$ns_ at $opt(stop).2 "$ns_  halt"

proc   finish {} {
   for {set i 2} {$i <=6  } {incr i 2} {
      exec rm -f out$i.xgr
      exec awk -f fil$i.awk out.tr > out$i.xgr
   }
   exec xgraph out2.xgr out4.xgr out6.xgr &
   puts "Finishing ns.."
   exit 0
}
```

Summarize the Results

- Ns2 manual, "Trace and Monitoring support," Ch. 26
- Linux commands to quickly get results
- Gets two files (if specified)
  - out-test.tr - use awk script to summarize the results
  - nam-out-test.nam - visualization

Visualize the Simulation Runs (nam file)

- nam-out-test.nam
  - Nam: network animator
- Try these
  - nam nam-1.14/edu/C2-sliding-color.nam
  - nam nam-1.14/tcl/test/test-wireless-2.nam

Trace File Format

Nodes are numbered as 0, 1, 2 , … inside ns2. I.e., node WT1 is node 0.

```
r 100.381997477 _1_ AGT  --- 82 cbr 1060 [13a 1 0 800] --
----- [0:0 1:0 32 1] [32 0] 1 0
```

r:receive event,
100.381997477:time stamps,
_1_:node 1,
AGT:trace generated by agent,
82:event(pkt) id,
cbr: cbr packet,
1060:packet size,
13a(hex):expected duration of pkt transmission (not working),
1:sender mac id,
0:transmitter mac id,
800:pkt type IP (806 for ARP),
0:0: sender address:port#
1:0: receiver address:port#,
32: TTL
1: next hop address,
[32 0]: TCP sequence #, ack #

Summarize Trace File

- Using simple Linux commands
  cat, grep, wc, |, >, >>, etc.
  eg. Calculate **packet delivery ratio** from a trace file (out.tr)

```
cat out.tr | grep AGT | grep cbr | grep ^s | wc -l
cat out.tr | grep AGT | grep cbr | grep ^r | wc -l

cat out.tr | grep AGT | grep cbr | grep ^s | grep _0_ | wc -l
cat out.tr | grep AGT | grep cbr | grep ^r | grep _1_ | wc -l
```

Base 0

- Simple programming
  shell, awk, etc.
- Advanced programming
  C/C++, Java, VB, etc.

Lab I

- Overall PDR (packet delivery ratio) or #packets received / #packets sent?
- PDR for WT1-WT2 connection?
- PDR for WT3-WT4 connection?
- PDR for WT5-WT6 connection?
- Are they very different? If yes, why?

awk

- Calling format
  - awk `/pattern-to-match/ {program to run}` trace-file
    eg. awk `$1 == "s" {print}` aaa.tr
  - awk –f awk-script trace-file
- Characteristics
  - Flexible (C style)
  - Simple (no pointers, no references)
  - Powerful
    - Float calculation
    - Automatic data type assignment and check
    - Branch/Loop control
    - Function call

awk (cont.)

- Script structure
  - Initialization
    BEGIN { … }
  - Body
    { … }
    Important: Every row in the trace file is scanned by the commands in the body part one time, just ONE time.
  - Summarization
    END { … }

An Example Of awk Script (fil2.awk)

```
BEGIN{ sim_end = 200;
       i=0;
       while (i<=sim_end) {sec[i]=0; i+=1;};
}

{      if ($1=="r" && $7=="cbr" && $3=="_1_") {
            sec[int($2)] += $8;
       };
}

END{ i=0;
       while (i<=sim_end) {
            print i " " sec[i]*8;
            i+=1;
       };
}
```

Every line in the trace file will be processed using the awk commands in the body.

What is $8?
What does it show?
What should be changed for fil4.awk & fil6.awk?

Working with awk Scripts

awk –f fil2.awk out.tr > out2.xgr

awk –f fil4.awk out.tr > out4.xgr

awk –f fil5.awk out.tr > out6.xgr

xgraph out2.xgr out4.xgr out6.xgr &

Another awk Script: **How to get Delay?**

```
BEGIN {
    nSentPackets = 0 ;
    nReceivedPackets = 0 ;
    rTotalDelay = 0.0 ;
}

{
    strEvent = $1 ;          rTime = $2 ;
    strAgt = $4 ;            idPacket = $6 ;
    strType = $7 ;
```

Another awk Script: **How to get Delay?**

```
    if ( strAgt == "AGT"   &&   strType == "cbr" ) {
        if ( strEvent == "s" ) {
                nSentPackets += 1 ;
                rSentTime[ idPacket ] = rTime ;
        }
        if ( strEvent == "r" ) {
                nReceivedPackets += 1 ;
                rReceivedTime[ idPacket ] = rTime ;
                rTotalDelay += rReceivedTime[ idPacket] - rSentTime[ idPacket ];
        }
    }
}

END {
    rPacketDeliveryRatio = nReceivedPackets / nSentPackets * 100 ;
    if ( nReceivedPackets != 0 ) rAverageDelay = rTotalDelay / nReceivedPackets ;
    printf( "AverageDelay: %15.5f PacketDeliveryRatio: %10.2f\n", rAverageDelay,
    rPacketDeliveryRatio ) ;
}
```

Lab I(ii)

- Create delay.awk
- Measure the average delay for the all traffic stream

- cp delay.awk delay2.awk
- Modify delay2.awk to measure the average delay for the first traffic stream (WT1->WT2)

- cp delay2.awk delay4.awk
- cp delay2.awk delay6.awk
- Modify delay4.awk and delay6.awk to measure the average delay for the second (WT3->WT4) and third (WT5->WT6) traffic stream

ns Tutorials

- NS website http://www.isi.edu/nsnam/ns/
- NS Manual http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf
- Marc Greis's Tutorial http://www.isi.edu/nsnam/ns/tutorial
- http://www.cs.virginia.edu/~cs757/slidespdf/cs757-ns2-tutorial-exercise1.pdf
- http://nile.wpi.edu/NS/
- http://nesl.ee.ucla.edu/courses/ee206a/2002s/guest_presentations/GP02_Park_ns2.ppt
- http://www.ece.ubc.ca/~elec565/ns2_tutorial.ppt

Appendix: Installing ns-2 (1)

- Before installing ns-2, you need to install required packages (With Ubuntu 10.04)
  - sudo apt-get install build-essential autoconf automake libxmu-dev libxt-dev libxt6 libsm-dev libsm6 libice-dev libice6 libxmu-dev libx11-dev sgb gcc-4.3
- Recent gcc version (4.4 or newer) could cause error, so use gcc-4.3 instead
  - CC=gcc-4.3 ./install

Appendix: Installing ns-2 (2)

- After installation, edit $HOME/.bashrc and append following in the file.

```
# NS2 Version & Path
NS2_VER=2.34
OTCL_VER=1.13
TCL_VER=8.4.18
TK_VER=$TCL_VER
NAM_VER=1.14
NS2_PATH=$HOME/ns-allinone-$NS2_VER
# LIBRARY
OTCL_LIB=$NS2_PATH/otcl-$OTCL_VER
NS2_LIB=$NS2_PATH/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:
$X11_LIB:$USR_LOCAL_LIB
TCL_LIB=$NS2_PATH/tcl$TCL_VER/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=$NS2_PATH/bin:$NS2_PATH/tcl$TCL_VER/unix:$NS2_PATH/tk
$TK_VER/unix
NS=$NS2_PATH/ns-$NS2_VER/
NAM=$NS2_PATH/nam-$NAM_VER/
PATH=$PATH:$XGRAPH:$NS:$NAM
```