EE122
Introduction to NS-2

Marghoob Mohiyuddin

Outline

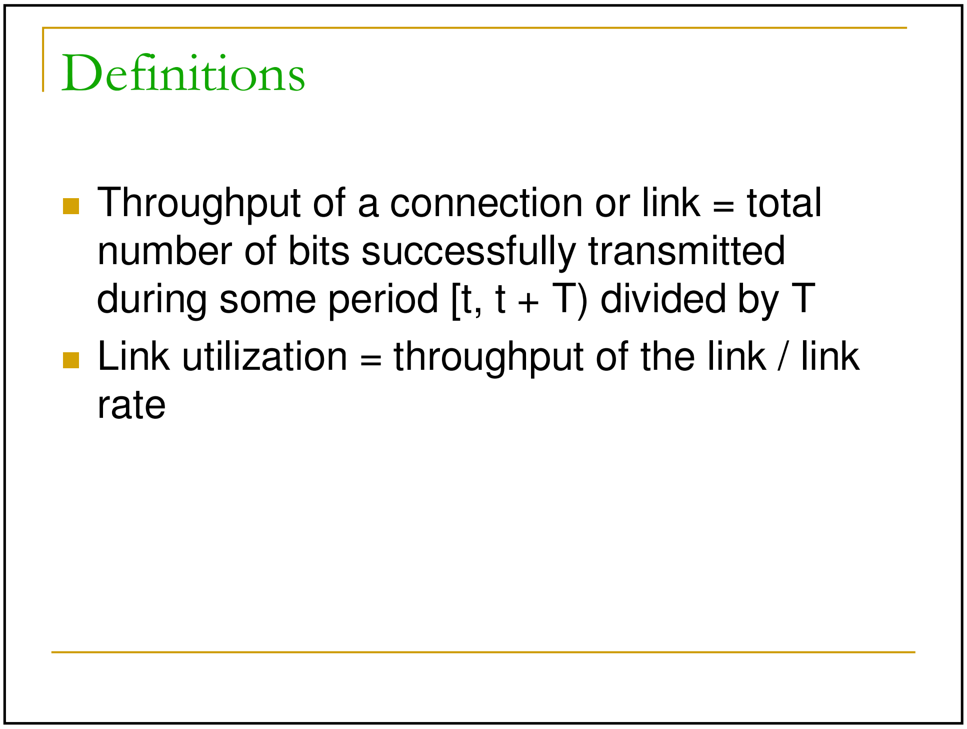
- **Review of network performance metrics**
- ns-2
- ns-2 demo

Measuring ‘network performance’ – Motivation

- Understanding network behavior
- Improving protocols
- Verifying correctness of implementation
- Detecting faults
- Monitor service level agreements
- Choosing provider
- Billing

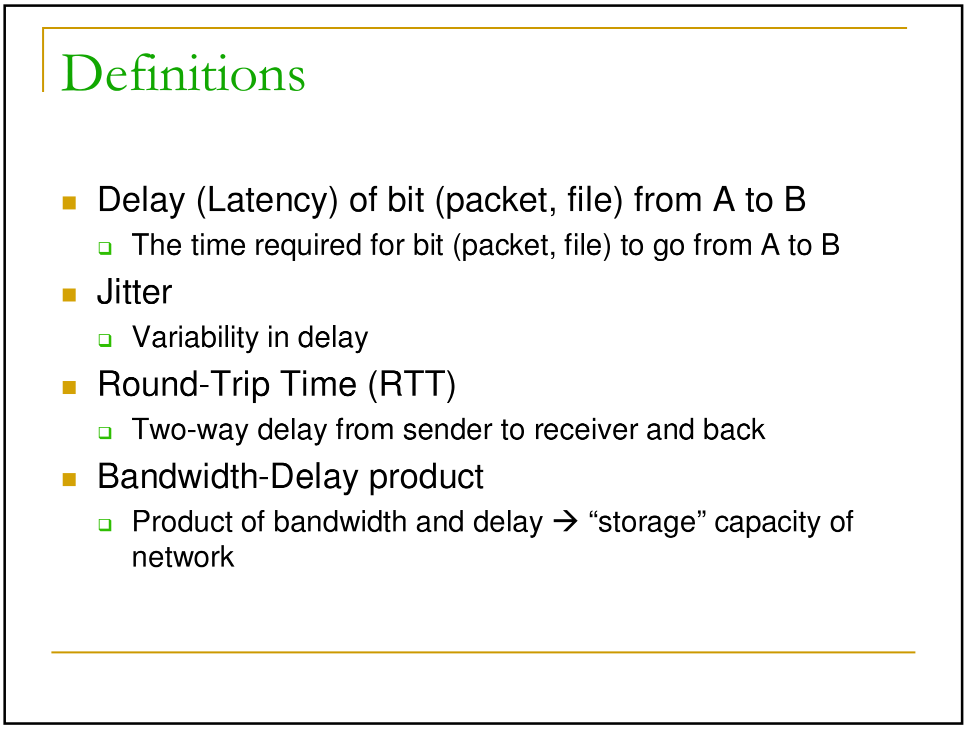
Definitions

- Link bandwidth (capacity): maximum rate (in bps) at which the sender can send data along the link
- Propagation delay: time it takes the signal to travel from source to destination
- Packet transmission time: time it takes the sender to transmit all bits of the packet
- Queuing delay: time the packet need to wait before being transmitted because the queue was not empty when it arrived
- Processing Time: time it takes a router/switch to process the packet header, manage memory, etc



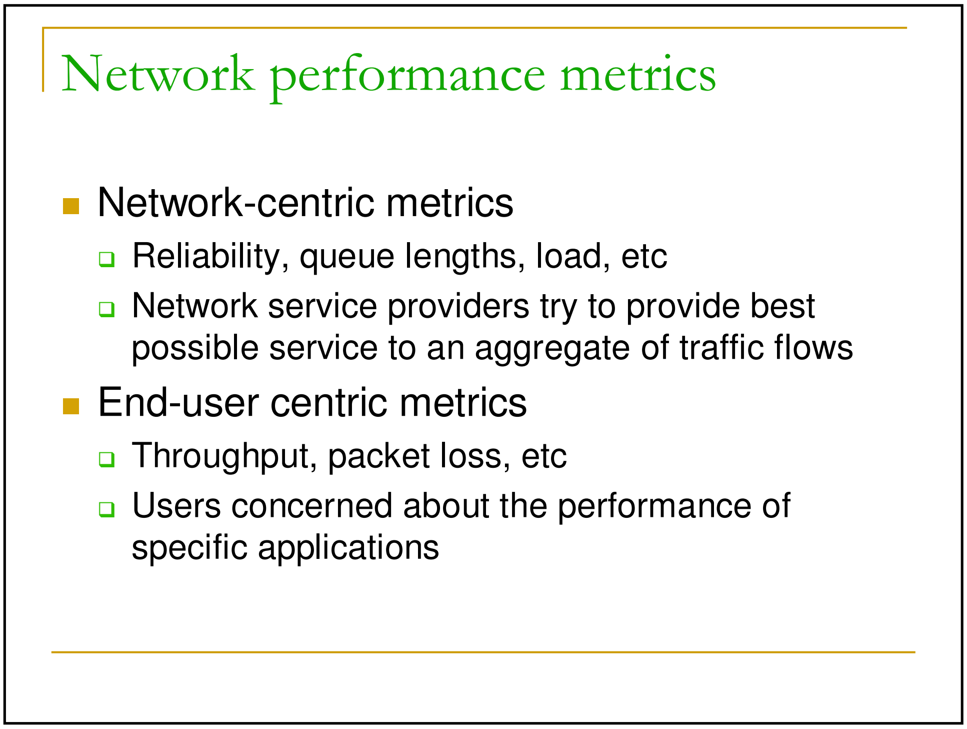
Definitions

- Throughput of a connection or link = total number of bits successfully transmitted during some period [t, t + T) divided by T
- Link utilization = throughput of the link / link rate



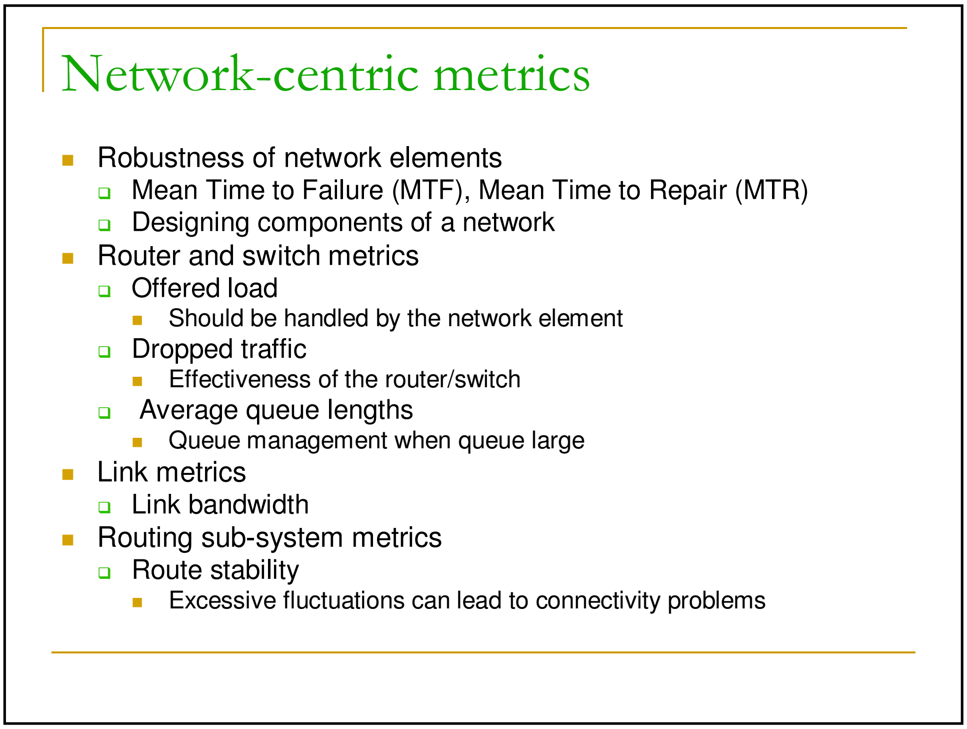
Definitions

- Delay (Latency) of bit (packet, file) from A to B
  - The time required for bit (packet, file) to go from A to B
- Jitter
  - Variability in delay
- Round-Trip Time (RTT)
  - Two-way delay from sender to receiver and back
- Bandwidth-Delay product
  - Product of bandwidth and delay → "storage" capacity of network



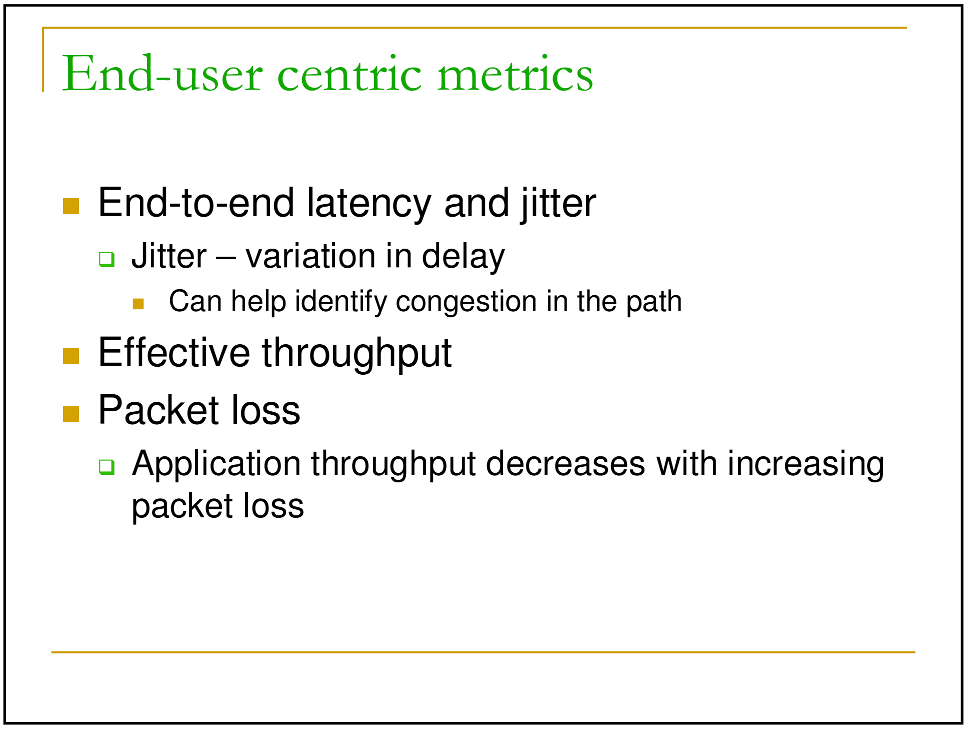
Network performance metrics

- Network-centric metrics
  - Reliability, queue lengths, load, etc
  - Network service providers try to provide best possible service to an aggregate of traffic flows
- End-user centric metrics
  - Throughput, packet loss, etc
  - Users concerned about the performance of specific applications



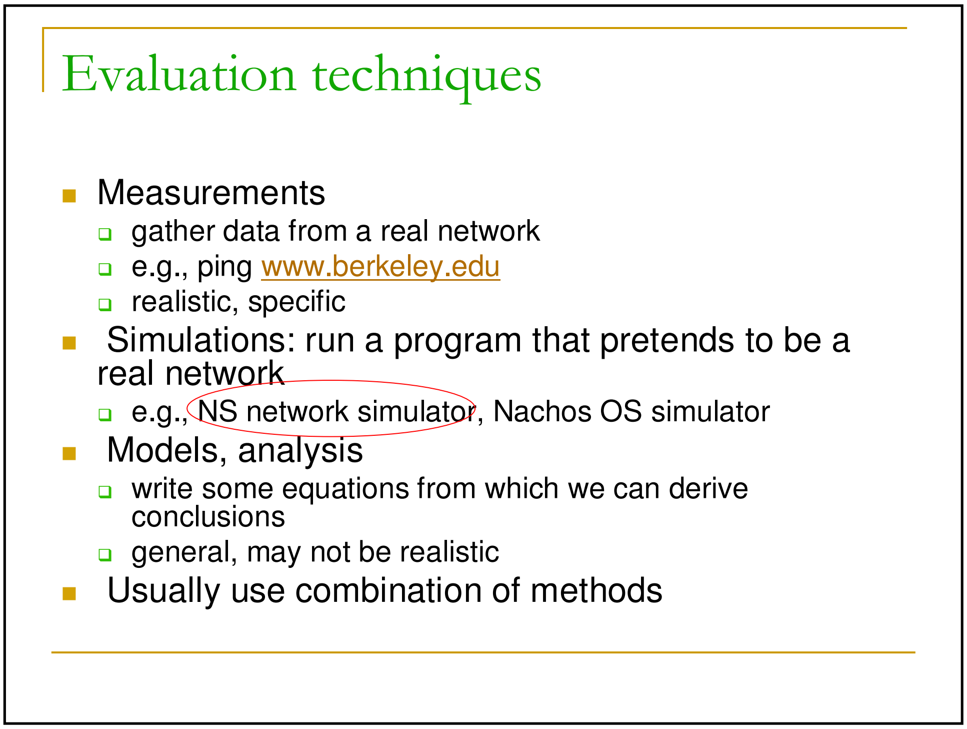
Network-centric metrics

- Robustness of network elements
  - Mean Time to Failure (MTF), Mean Time to Repair (MTR)
  - Designing components of a network
- Router and switch metrics
  - Offered load
    - Should be handled by the network element
  - Dropped traffic
    - Effectiveness of the router/switch
  - Average queue lengths
    - Queue management when queue large
- Link metrics
  - Link bandwidth
- Routing sub-system metrics
  - Route stability
    - Excessive fluctuations can lead to connectivity problems



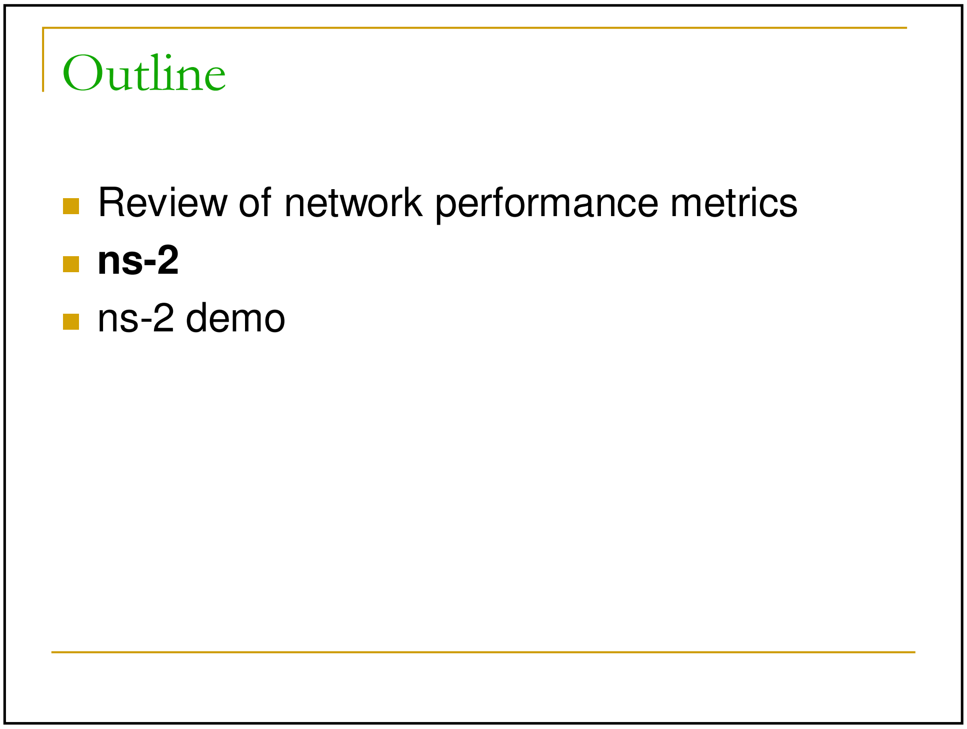
End-user centric metrics

- End-to-end latency and jitter
  - Jitter – variation in delay
    - Can help identify congestion in the path
- Effective throughput
- Packet loss
  - Application throughput decreases with increasing packet loss



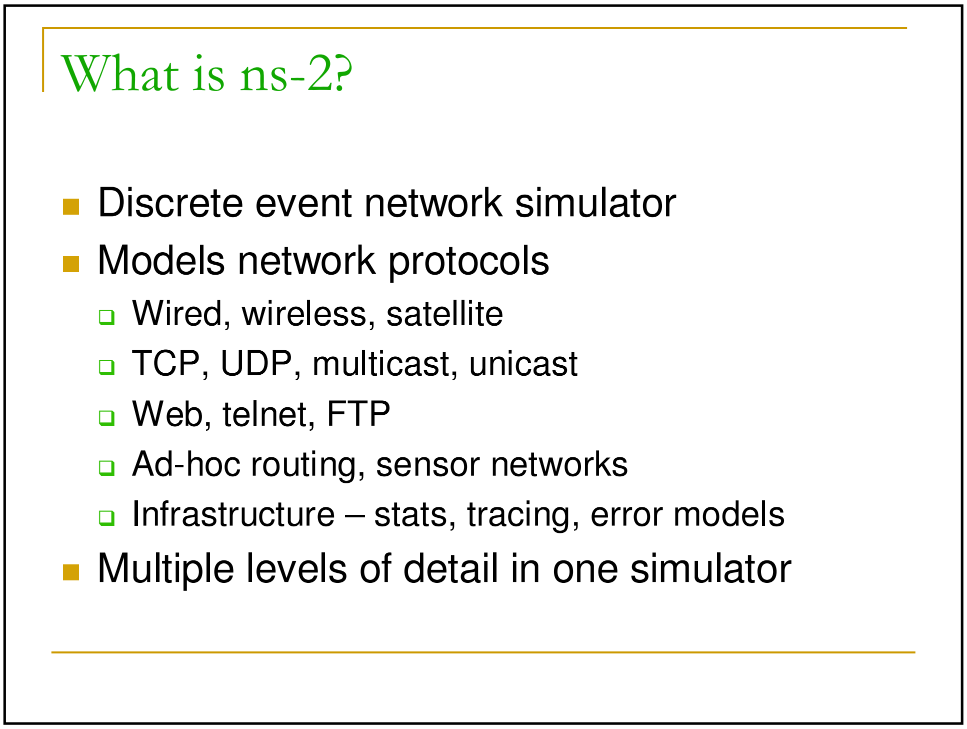
Evaluation techniques

- Measurements
  - gather data from a real network
  - e.g., ping www.berkeley.edu
  - realistic, specific
- Simulations: run a program that pretends to be a real network
  - e.g., NS network simulator, Nachos OS simulator
- Models, analysis
  - write some equations from which we can derive conclusions
  - general, may not be realistic
- Usually use combination of methods



Outline

- Review of network performance metrics
- **ns-2**
- ns-2 demo



What is ns-2?

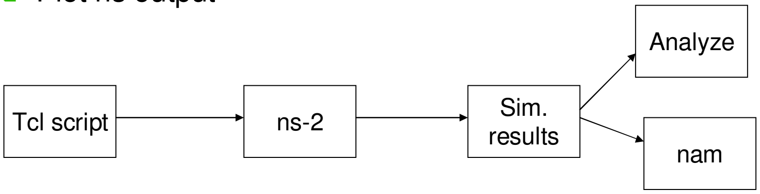
- Discrete event network simulator
- Models network protocols
  - Wired, wireless, satellite
  - TCP, UDP, multicast, unicast
  - Web, telnet, FTP
  - Ad-hoc routing, sensor networks
  - Infrastructure – stats, tracing, error models
- Multiple levels of detail in one simulator

Why simulate?

- To examine protocol in controlled environment
- Repeatable experiments
- Alternatives
  - Experimentation: operation details, but limited scale, limited flexibility
  - Analysis: can provide deeper understanding, but ignores implementation details

ns-2 components

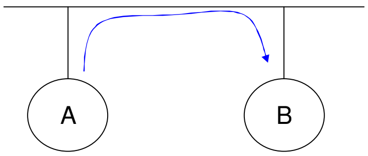
- ns – Network Simulator
  - Executes Tcl scripts containing simulation setup and events
- nam – Network AniMator
  - Visualize ns output
- xgraph – graph plotter
  - Plot ns output



Discrete event simulation

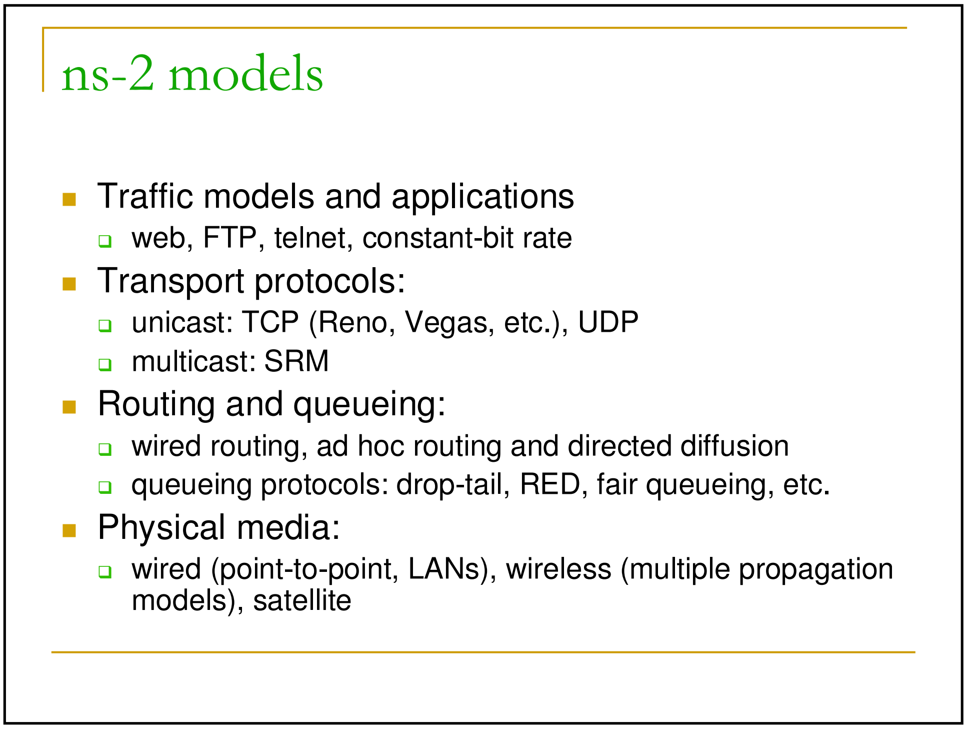
- Model world as events
  - Maintain queue of events, ordered by time
  - Main virtual (simulated) time
  - Repeat
    - Extract event at head, set virtual time to event's time
    - Process it
    - If processing generates another event, then add it to queue
  - Each event takes predefined amount of virtual time, arbitrary amount of real time
    - Slow CPU makes simulation run slower (in real time), but does not change result

Discrete event example



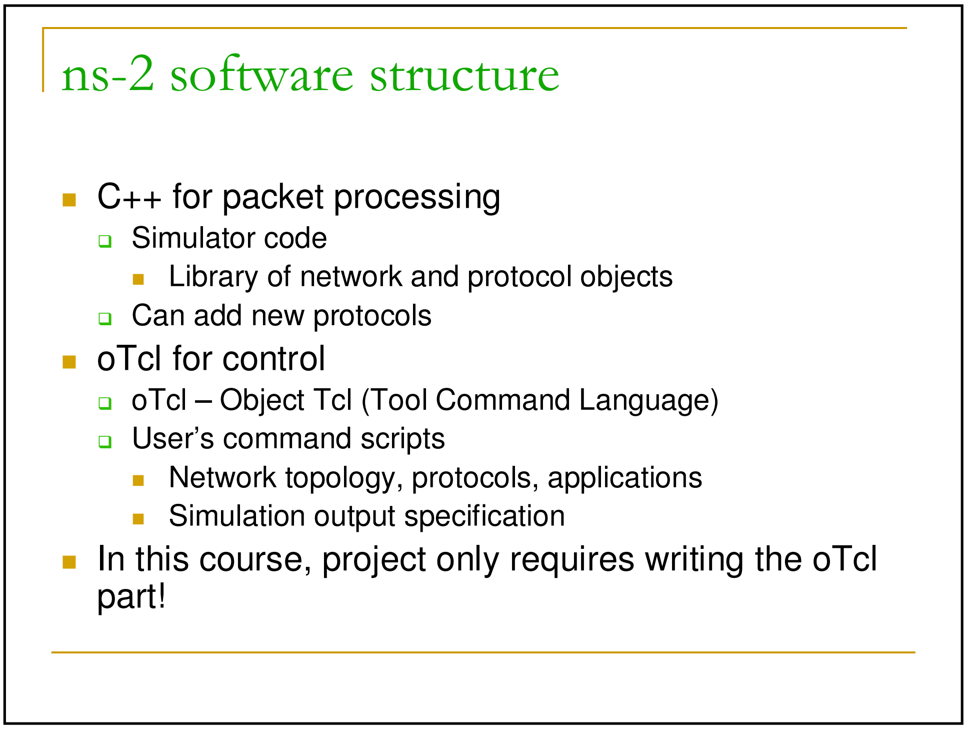
A and B two nodes on an ethernet

- Assuming a simple queue model
- Event at t=1
  - A enqueues packet on the LAN
  - Generates event at t=1.1
- Event at t=1.1
  - LAN dequeues packet and triggers B



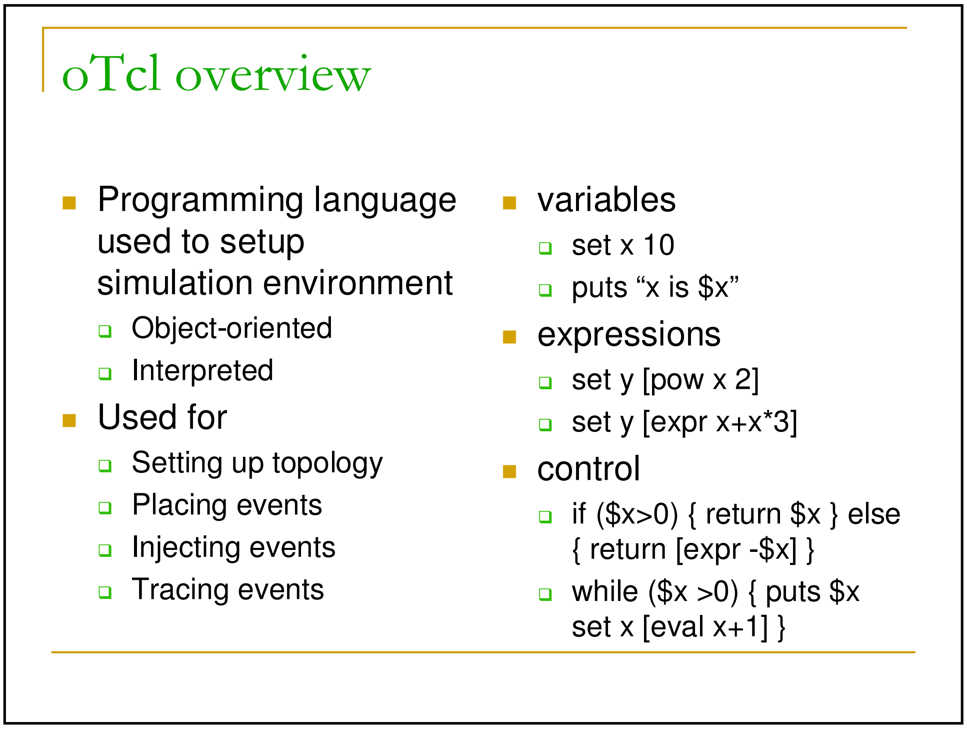
ns-2 models

- Traffic models and applications
  - web, FTP, telnet, constant-bit rate
- Transport protocols:
  - unicast: TCP (Reno, Vegas, etc.), UDP
  - multicast: SRM
- Routing and queueing:
  - wired routing, ad hoc routing and directed diffusion
  - queueing protocols: drop-tail, RED, fair queueing, etc.
- Physical media:
  - wired (point-to-point, LANs), wireless (multiple propagation models), satellite



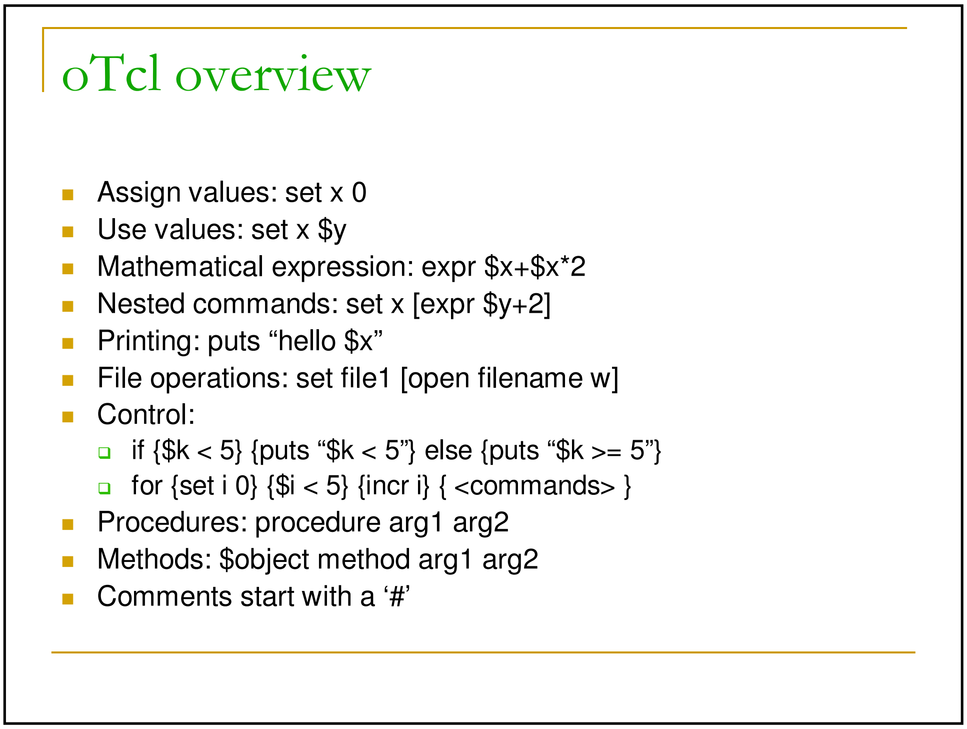
ns-2 software structure

- C++ for packet processing
  - Simulator code
    - Library of network and protocol objects
  - Can add new protocols
- oTcl for control
  - oTcl – Object Tcl (Tool Command Language)
  - User’s command scripts
    - Network topology, protocols, applications
    - Simulation output specification
- In this course, project only requires writing the oTcl part!



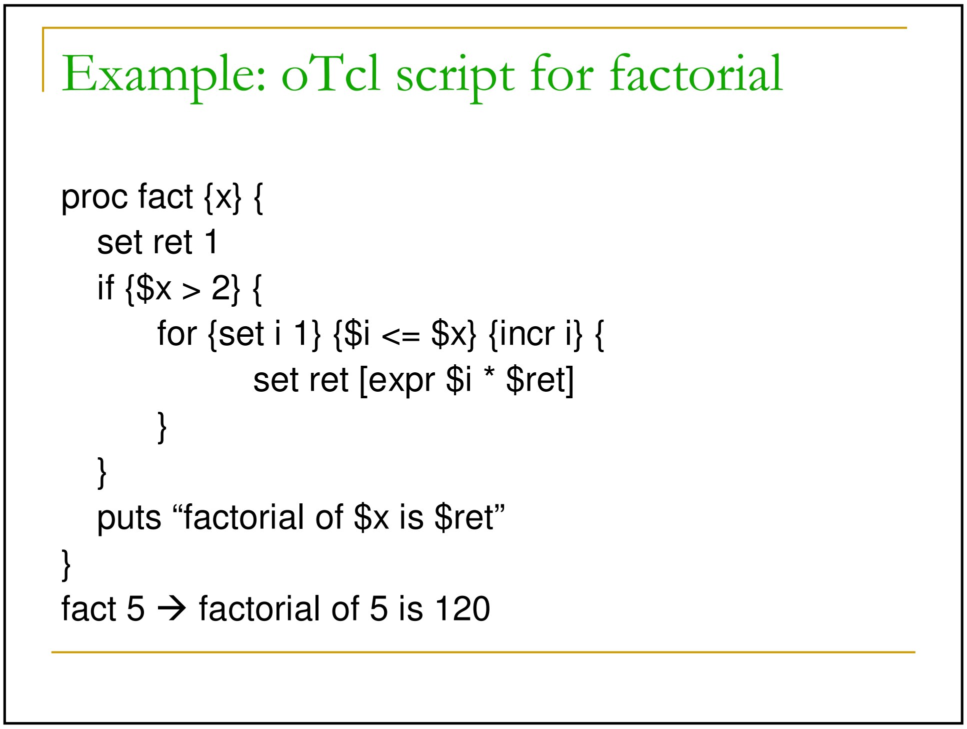
oTcl overview

- Programming language used to setup simulation environment
  - Object-oriented
  - Interpreted
- Used for
  - Setting up topology
  - Placing events
  - Injecting events
  - Tracing events
- variables
  - set x 10
  - puts “x is $x”
- expressions
  - set y [pow x 2]
  - set y [expr x+x*3]
- control
  - if ($x>0) { return $x } else { return [expr -$x] }
  - while ($x >0) { puts $x set x [eval x+1] }



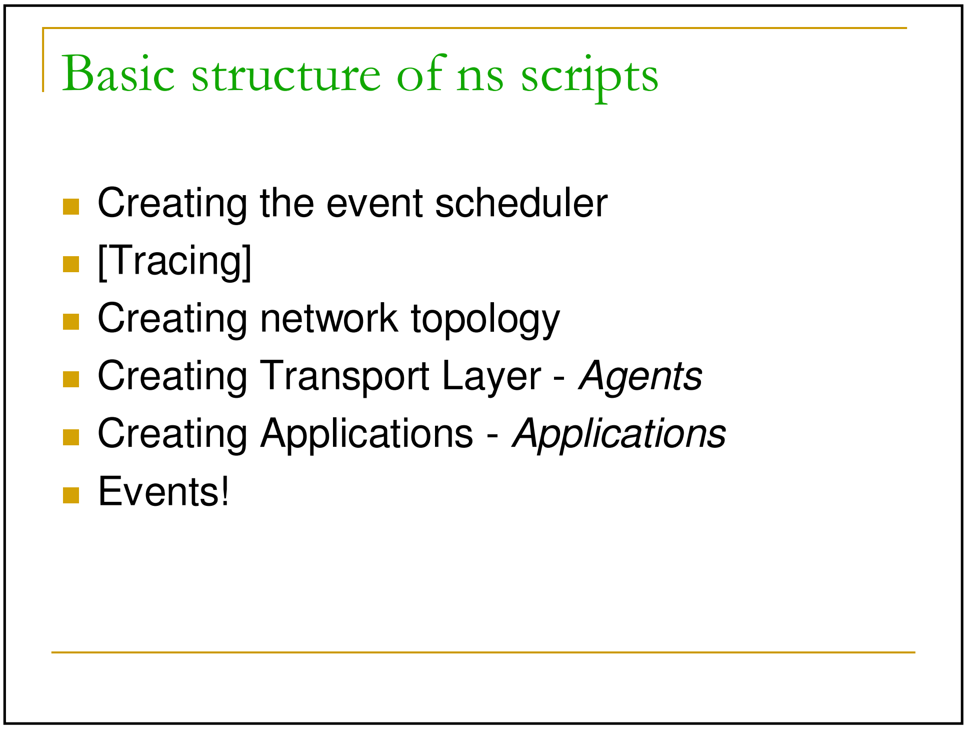
oTcl overview

- Assign values: set x 0
- Use values: set x $y
- Mathematical expression: expr $x+$x*2
- Nested commands: set x [expr $y+2]
- Printing: puts “hello $x”
- File operations: set file1 [open filename w]
- Control:
  - if {$k < 5} {puts “$k < 5”} else {puts “$k >= 5”}
  - for {set i 0} {$i < 5} {incr i} { <commands> }
- Procedures: procedure arg1 arg2
- Methods: $object method arg1 arg2
- Comments start with a ‘#’



Example: oTcl script for factorial

```
proc fact {x} {
   set ret 1
   if {$x > 2} {
        for {set i 1} {$i <= $x} {incr i} {
                set ret [expr $i * $ret]
        }
   }
   puts “factorial of $x is $ret”
}
fact 5 → factorial of 5 is 120
```



Basic structure of ns scripts

- Creating the event scheduler
- [Tracing]
- Creating network topology
- Creating Transport Layer - *Agents*
- Creating Applications - *Applications*
- Events!

Creating event scheduler

- Create scheduler
  - set ns [new Simulator]

Creates new simulator object
store this in the var. ns

- Schedule event
  - $ns at <time> <event>
  - <event>: any legitimate ns/tcl commands
- Start scheduler
  - $ns run

‘Hello World’ in ns-2

- helloworld.tcl:

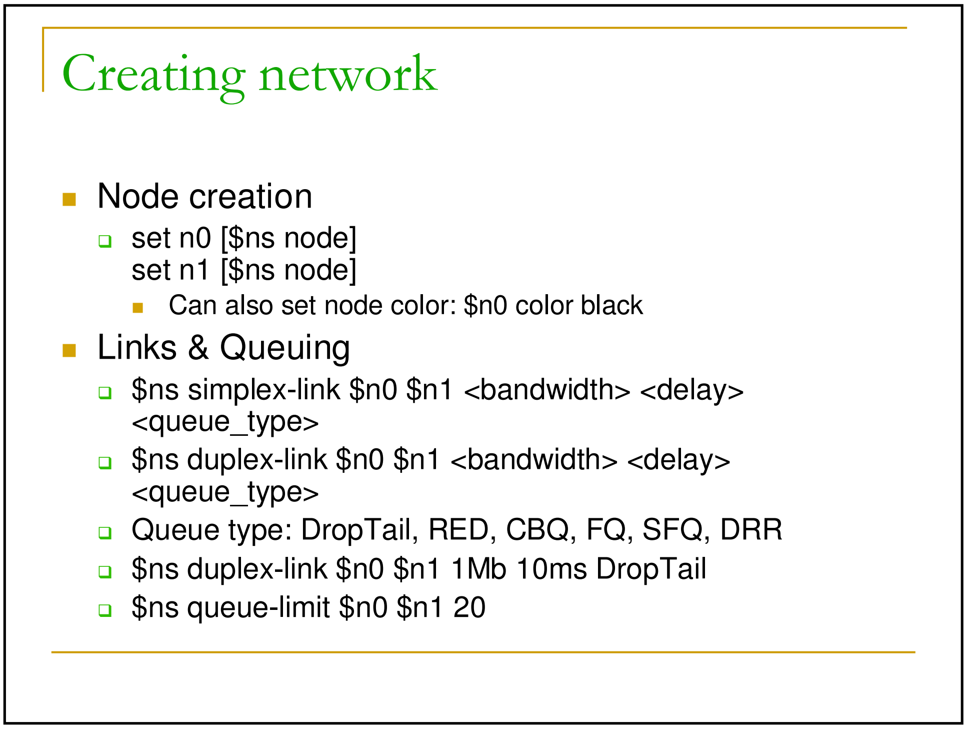
  -
    ```
    set ns [new Simulator]
    $ns at 1 “puts \“Hello World!\””
    $ns at 1.5 “exit”
    $ns run
    ```
- c199% ns helloworld.tcl
- c199%Hello World!

Create a simulator, put in var ns

Schedule event ‘print HelloWorld
at time t=1

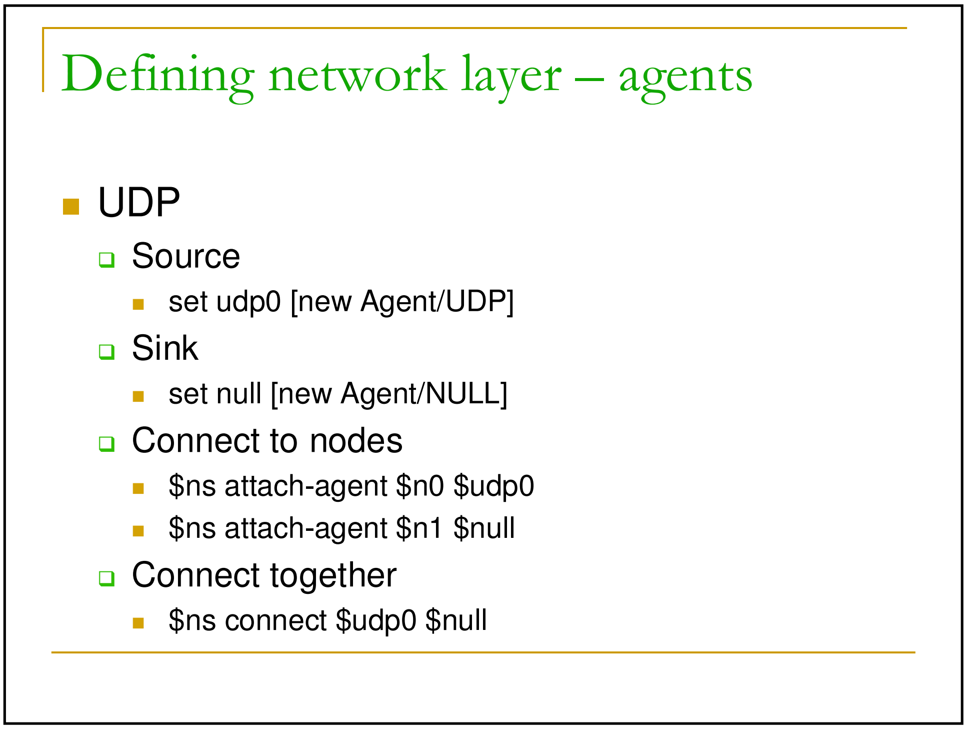
Run the simulator executing events

Execute the script



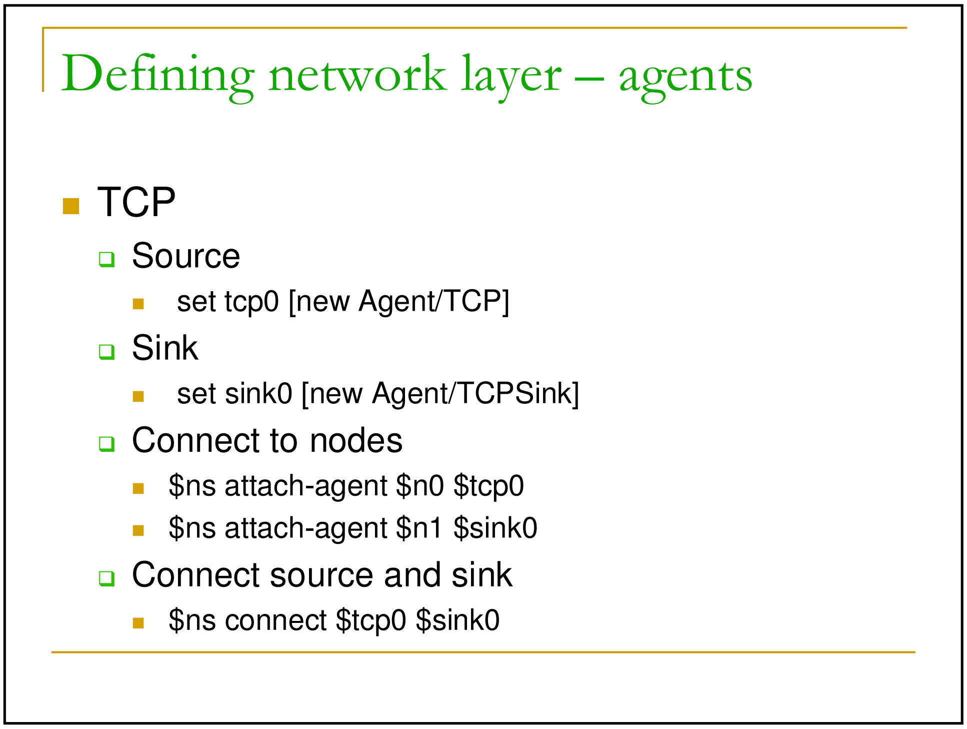
Creating network

- Node creation
  - set n0 [$ns node]
    set n1 [$ns node]
    - Can also set node color: $n0 color black
- Links & Queuing
  - $ns simplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
  - $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
  - Queue type: DropTail, RED, CBQ, FQ, SFQ, DRR
  - $ns duplex-link $n0 $n1 1Mb 10ms DropTail
  - $ns queue-limit $n0 $n1 20



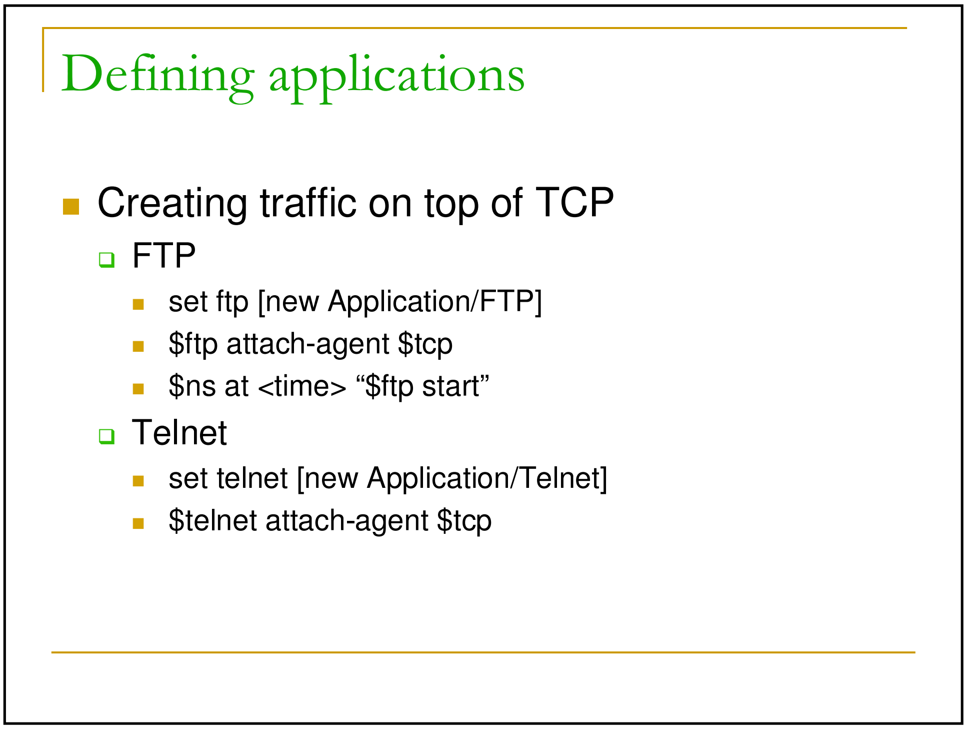
Defining network layer – agents

- UDP
  - Source
    - set udp0 [new Agent/UDP]
  - Sink
    - set null [new Agent/NULL]
  - Connect to nodes
    - $ns attach-agent $n0 $udp0
    - $ns attach-agent $n1 $null
  - Connect together
    - $ns connect $udp0 $null



Defining network layer – agents

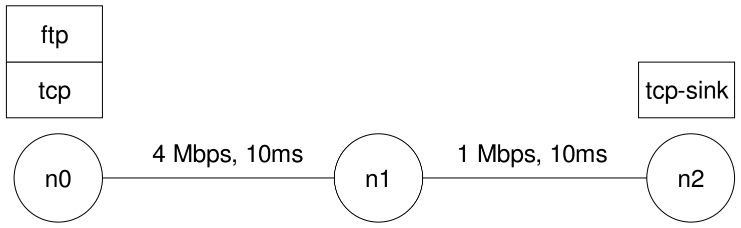
- TCP
  - Source
    - set tcp0 [new Agent/TCP]
  - Sink
    - set sink0 [new Agent/TCPSink]
  - Connect to nodes
    - $ns attach-agent $n0 $tcp0
    - $ns attach-agent $n1 $sink0
  - Connect source and sink
    - $ns connect $tcp0 $sink0



Defining applications

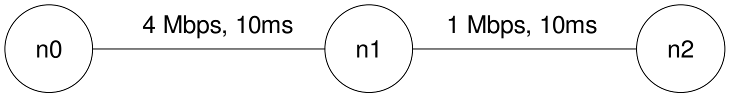
- Creating traffic on top of TCP
  - FTP
    - set ftp [new Application/FTP]
    - $ftp attach-agent $tcp
    - $ns at <time> “$ftp start”
  - Telnet
    - set telnet [new Application/Telnet]
    - $telnet attach-agent $tcp

Example



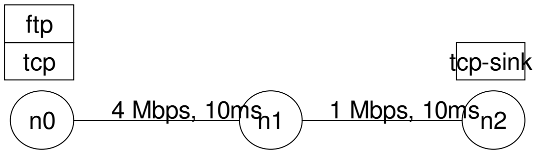
Example

1. Create the simulator
   - set ns [new simulator]
2. Set Up Network Topology
   - set n0 [$ns node]
   - set n1 [$ns node]
   - set n2 [$ns node]
3. Define Traffic Patterns
   - $ns duplex-link $n0 $n1 4Mb 10ms DropTail
   - $ns duplex-link $n2 $n1 1Mb 10ms DropTail
   - $ns queue-limit $n1 $n2 10



Example

4. Define Agents
   - #Create a TCP agent and attach it to node n0
   - set tcp0 [new Agent/TCP]
   - $ns attach-agent $n0 $tcp0
   - #Create a TCP sink agent and attach it to node n2
   - set sink [new Agent/TCPSink]
   - $ns attach-agent $n2 $sink
   - #Connect both agents
   - $ns connect $tcp0 $sink
   - #Create an FTP source
   - set ftp [new Application/FTP]
   - $ftp set maxpkts_ 1000
   - $ftp attach-agent $tcp0



Example

5. Schedule Simulation Events
   $ns at 0.0 "$ftp start“
   $ns at 10.0 "$ftp stop“
   $ns at 10.1 "finish“
6. Run the simulation
   $ns run

Example

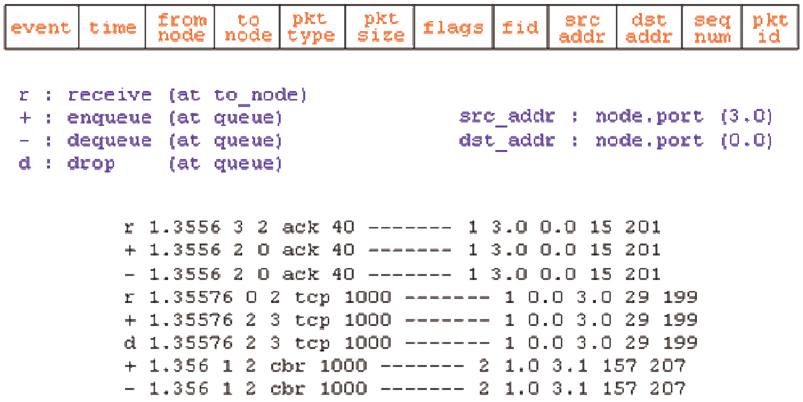
```
#Create a simulator object
set ns [new Simulator]
#Create three nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
#Create link between the nodes
$ns duplex-link $n0 $n1 4Mb 10ms DropTail
$ns duplex-link $n2 $n1 1Mb 10ms DropTail
$ns queue-limit $n1 $n2 10
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
#Create a TCP sink agent and attach it to
    node n2
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
```

```
#Connect both agents
$ns connect $tcp0 $sink
# create an FTP source
set ftp [new Application/FTP]
$ftp set maxpkts_ 1000
$ftp attach-agent $tcp0
#Inject starting events
$ns at 0.0 "$ftp start"
$ns at 10.0 "$ftp stop"
$ns at 10.1 "finish"
#Run the simulation
$ns run
```

Collecting traces

- Tracing all packets on all links
  - set trace_file [open out.tr w]
  - $ns trace-all $trace_file
  - $ns flush-trace
  - close $trace_file
- Tracing packets on a specific link
  - ns trace-queue $node0 $node1 $trace_file

Trace format – example



event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)              src_addr : node.port (3.0)
- : dequeue (at queue)              dst_addr : node.port (0.0)
d : drop     (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

fid is IPv6 flow identifier

Analyzing ns-2 output

- Unix tools
  - awk
    - Simple processing of data files – summing up a column, averaging, etc.
  - grep
    - Filter a file
  - perl
    - Processing and filtering
- Plotting tools like xgraph, gnuplot to plot the relevant statistics

nam to visualize ns output

- Collecting traces for nam
  - set nf [open out.nam w]
  - $ns namtrace-all $nf
- Visualizing the trace
  - nam out.nam

nam demo

```
#Create a simulator object
set ns [new Simulator]
# open the nam trace file
set nam_trace_fd [open tcp_tahoe.nam w]
$ns namtrace-all $nam_trace_fd
# define a 'finish' procedure
proc finish {} {
      global ns nam_trace_fd trace_fd
      # close the nam trace file
      $ns flush-trace
      close $nam_trace_fd
      # execute nam on the trace file
      exit 0
}
#Create three nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
#Create link between the nodes
$ns duplex-link $n0 $n1 4Mb 10ms DropTail
$ns duplex-link $n2 $n1 1Mb 10ms DropTail
$ns queue-limit $n1 $n2 10
```

```
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
#Create a TCP sink agent and attach it to
      node n2
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
#Connect both agents
$ns connect $tcp0 $sink
# create an FTP source
set ftp [new Application/FTP]
$ftp set maxpkts_ 1000
$ftp attach-agent $tcp0
#Inject starting events
$ns at 0.0 "$ftp start"
$ns at 10.0 "$ftp stop"
$ns at 10.1 "finish"
#Run the simulation
$ns run
```

Tips

- ns-2 man pages
  - Lot of details omitted in the presentation
- Working oTcl code as a template
- Verify topology!
  - nam might be helpful

References

- NS by example
  - http://nile.wpi.edu/NS/
- Marc Greis's tutorial
  - http://www.isi.edu/nsnam/ns/tutorial/index.html
- EE122, Fall 2005 slides on ns
  - http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/NS2005.pdf
- Official NS manual
  - http://www.isi.edu/nsnam/ns/ns-documentation.html

End of show!
Questions?