Information Technology Engineering

Mohammad Hossein Manshaei

manshaei@gmail.com

1393

A brief Introduction to ns-2

NETWORK SIMULATION

Contents

ns-2, the Network Simulator

- A simulator for communication networks, developed at Laurence Berkely National Laboratory (LBNL) within VINT Project
- It fast became a property of the research community
- Everyone could add its own modules and contribute to its development
- Downloadable Freely at: http://www.isi.edu/nsnam/ns
- Can be installed on Linux and Windows

ns-2, the Network Simulator

- Academic project over 10 years old
  - freely distributed, open source
- Maintained by ISI (Information Science Institute - USC)
  - DARPA + NSF projects
- ~ 200K LoC, ~400 page manual
- Large user base
  - mostly academics
- *"de facto"* standard in networking research

ns-2 Functionality

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

Contents

ns-2 Components

1. Pre-processing:
   - traffic and topology generators
2. nam, the Network AniMator
   - visualize Ns (or other) output
   - GUI input simple Ns scenarios
3. Post-processing:
   - simple trace analysis, often in Awk, Perl, or Tcl

Tutorial: http://www.isi.edu/nsnam/ns/tutorial

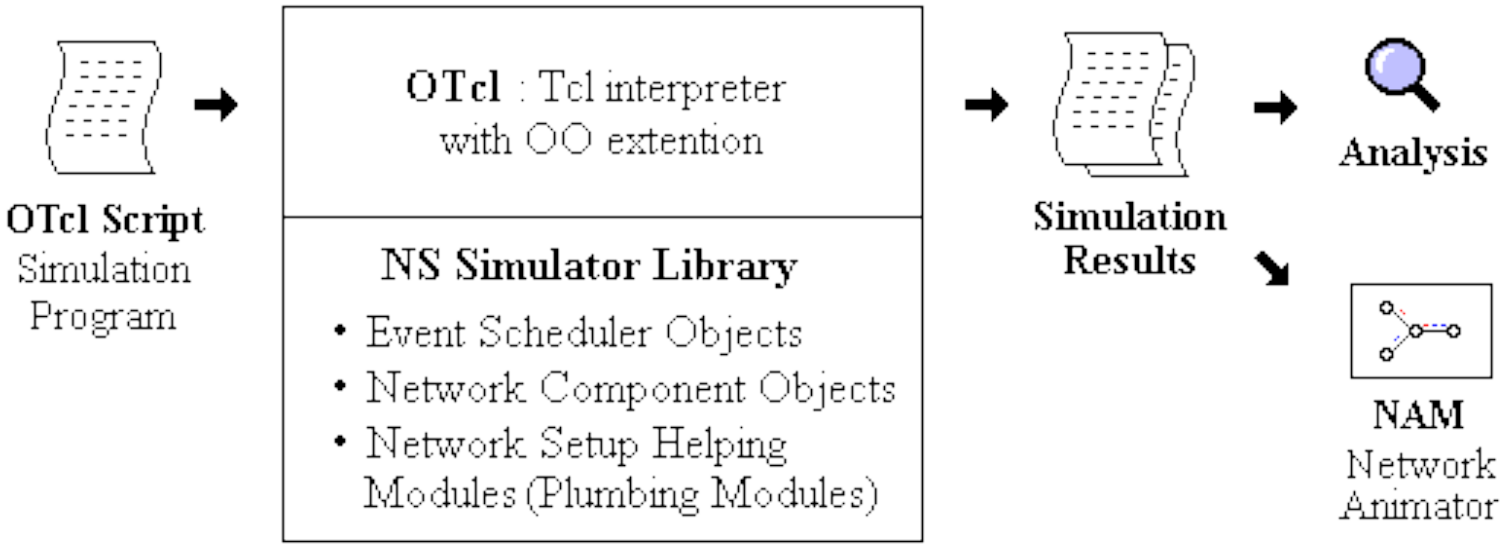
ns-2 Software Structure: C++ and OTCL

Uses two languages:

1. **C++ for packet-processing**
   1. per packet processing
   2. fast to run, detailed, complete control
2. **OTCL for control**
   1. simulation setup, configuration, occasional actions
   2. fast to write and change

ns-2 Software Structure: C++ and OTCL

- Object-oriented, discrete event-driven network simulator
- Written in C++ and OTcl



Steps when using ns-2

- **Create OTCL script with network model**
  - nodes, links, traffic sources, sinks, etc.
- **Parameterize simulation objects**
  - queue sizes, link speeds, TCP flavor and parameters (more than 30)
- **Collect statistics**
  - dump everything to trace, post process it
  - gather stats during simulation within OTCL script
  - Modify ns source code
- **Run ns multiple times**
  - confidence intervals

Contents

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**
  1. **Transports:**
     - TCP, UDP, multicast, etc.
     - transport protocol instances attach to nodes
  2. **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: TCP

- **source and sink**
  - set t_src [new Agent/TCP/Newreno]
  - set t_dst [new Agent/TCPSink]
  - "Newreno" flavor of TCP

- **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 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 Traffic over UDP Channels

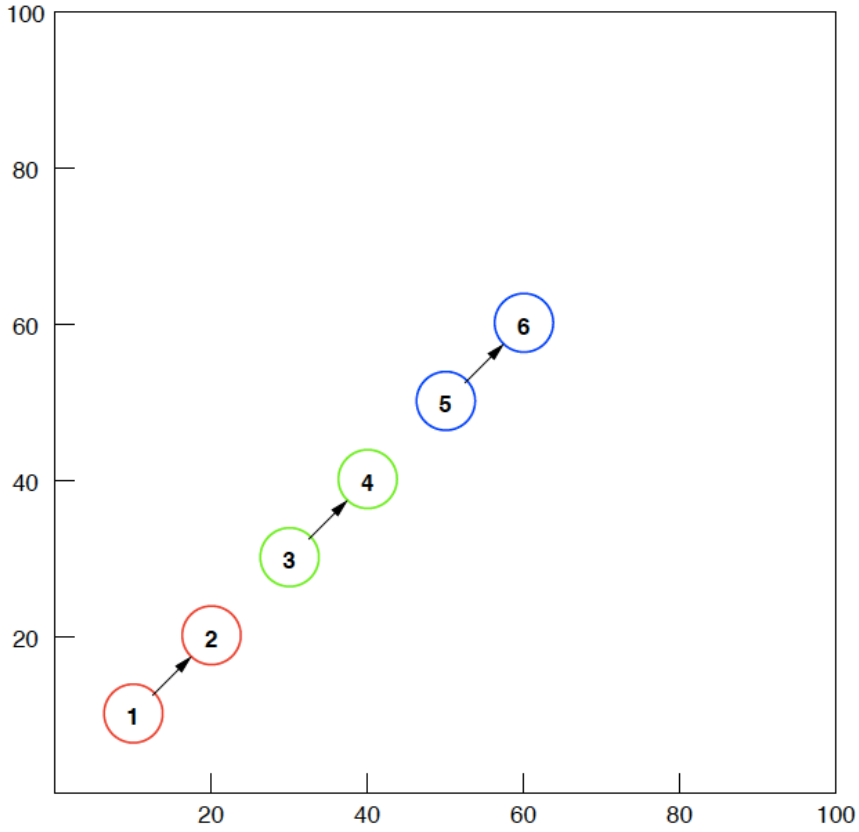
- **CBR**
  - set cbr [new Application/Traffic/CBR]
  - $cbr set packetSize_ 512
  - $cbr set interval_ 0.250
  - $cbr attach-agent $u_src
  - $ns at <time> “$cbr start”

Contents

1. Introduction to ns-2
2. ns-2 Components
3. Create a Basic ns-2 Model
4. Case Study: WiFi Simulation
5. Simulation Results Analysis (Traces and NAM)

Case Study: WiFi Simulation

- Ad Hoc Network
- 6 nodes and 3 connections
- CBR Traffic over UDP



Make Nodes and their Positions

```
for {set i 1} {$i <= $opt(nn)}
{incr i} {
set WT($i) [ $ns_ node $i ]
}
proc coord_proc {a} {
return [expr 10 * $a ]
}
for {set i 1} {$i <= $opt(nn)}
{incr i} {
$WT($i) set X_ [coord_proc $i]
$WT($i) set Y_ [coord_proc $i]
$WT($i) set Z_ 0.0
}
```

CBR sources over UDP Transport

```
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)

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

Mobile Network Parameters

```
$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 \
```

Trace Definition

```
-agentTrace ON \
-routerTrace OFF \
-macTrace OFF \
-movementTrace OFF
```

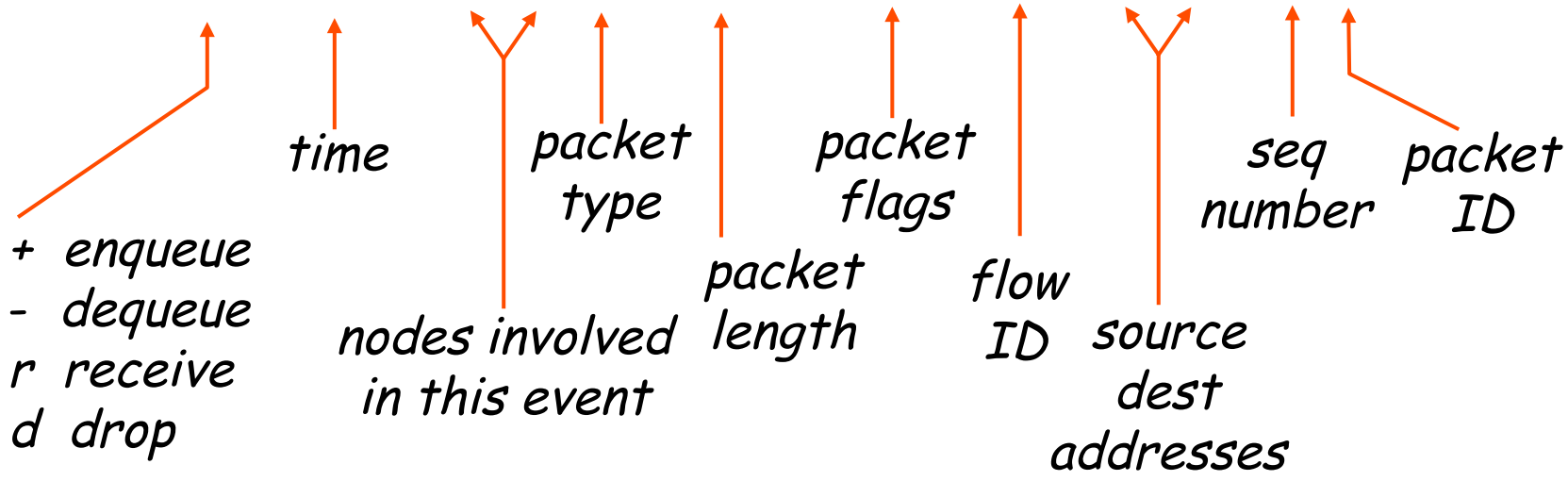
Contents

1. Introduction to ns-2
2. ns-2 Components
3. Create a Basic ns-2 Model
4. Case Study: WiFi Simulation
5. Simulation Results Analysis (Traces and NAM)

Tracing

- Trace packets on individual links
- Tracefile format:

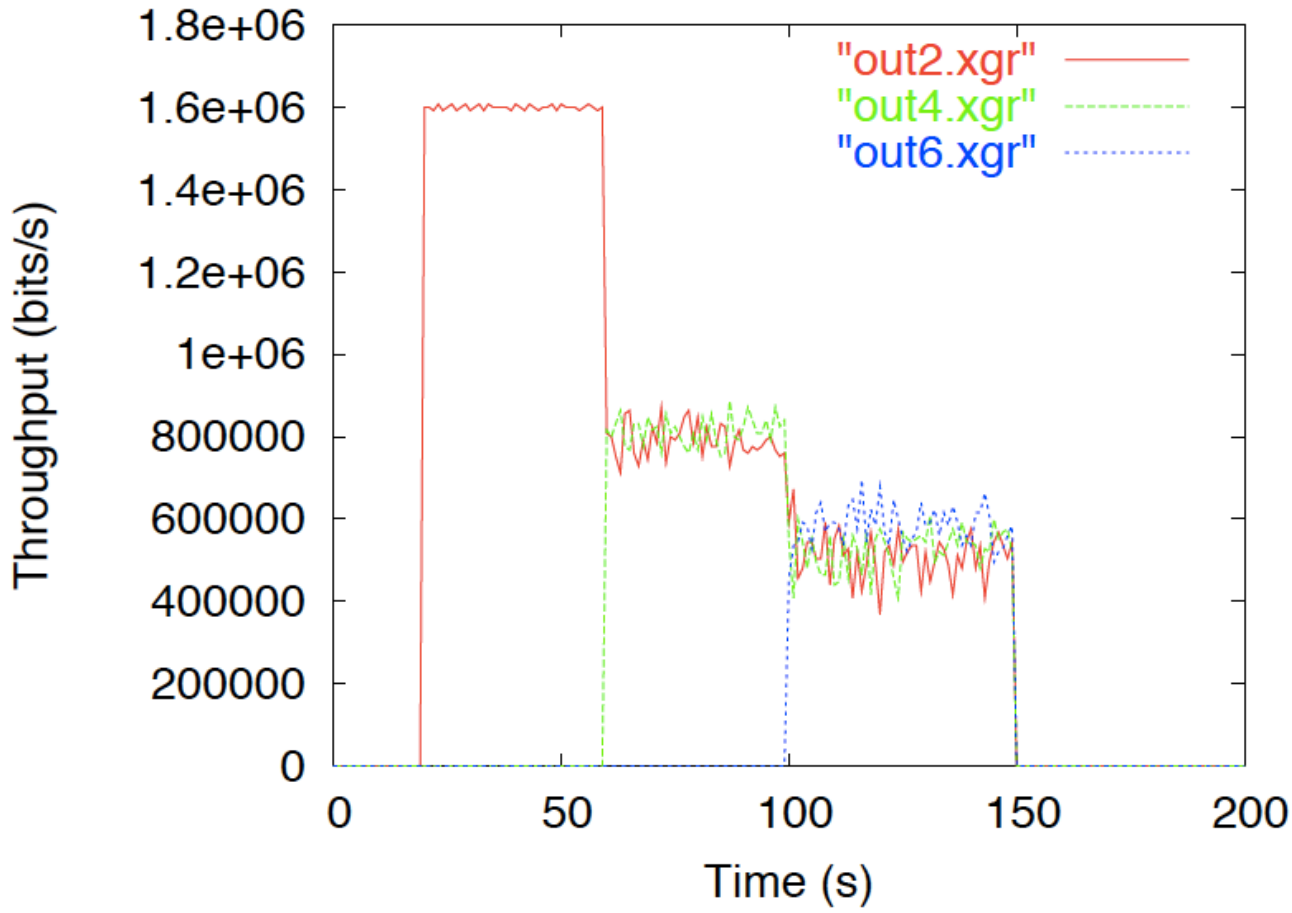
```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src>
   <dst> <seqno> <aseqno>
    + 1        0 2 tcp 900 ------- 1 0.0 3.1 7 15
    - 1        0 2 tcp 900 ------- 1 0.0 3.1 7 15
    r 1.00234  0 2 tcp 900 ------- 1 0.0 3.1 7 15
```



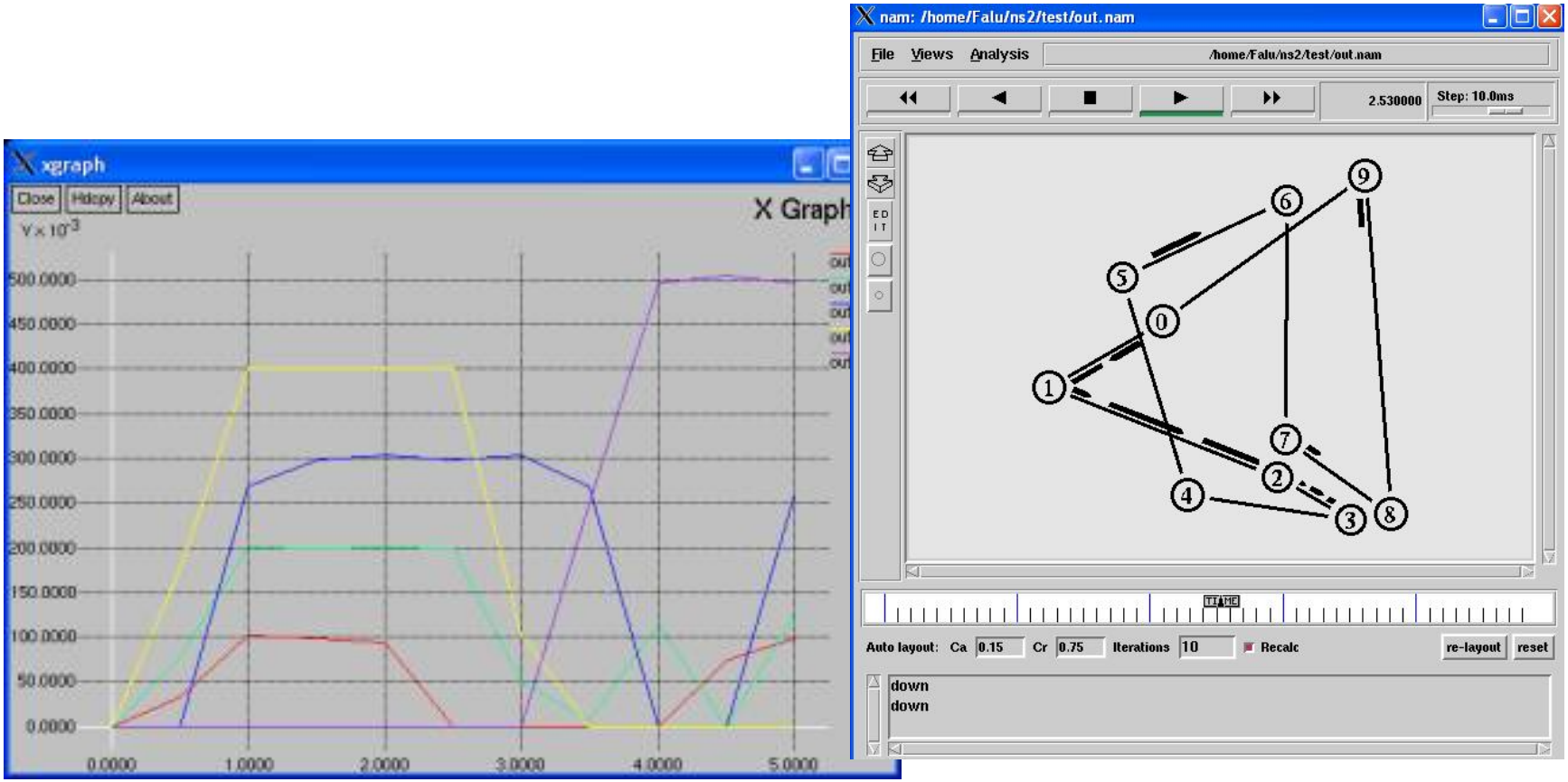
Ns Trace file : An Example

```
...
+ 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
...
```

Throughput Calculation: Our Case



Graphical Tools



NAM (Network Animator)