6.829 $ns-2$ tutorial, part 1

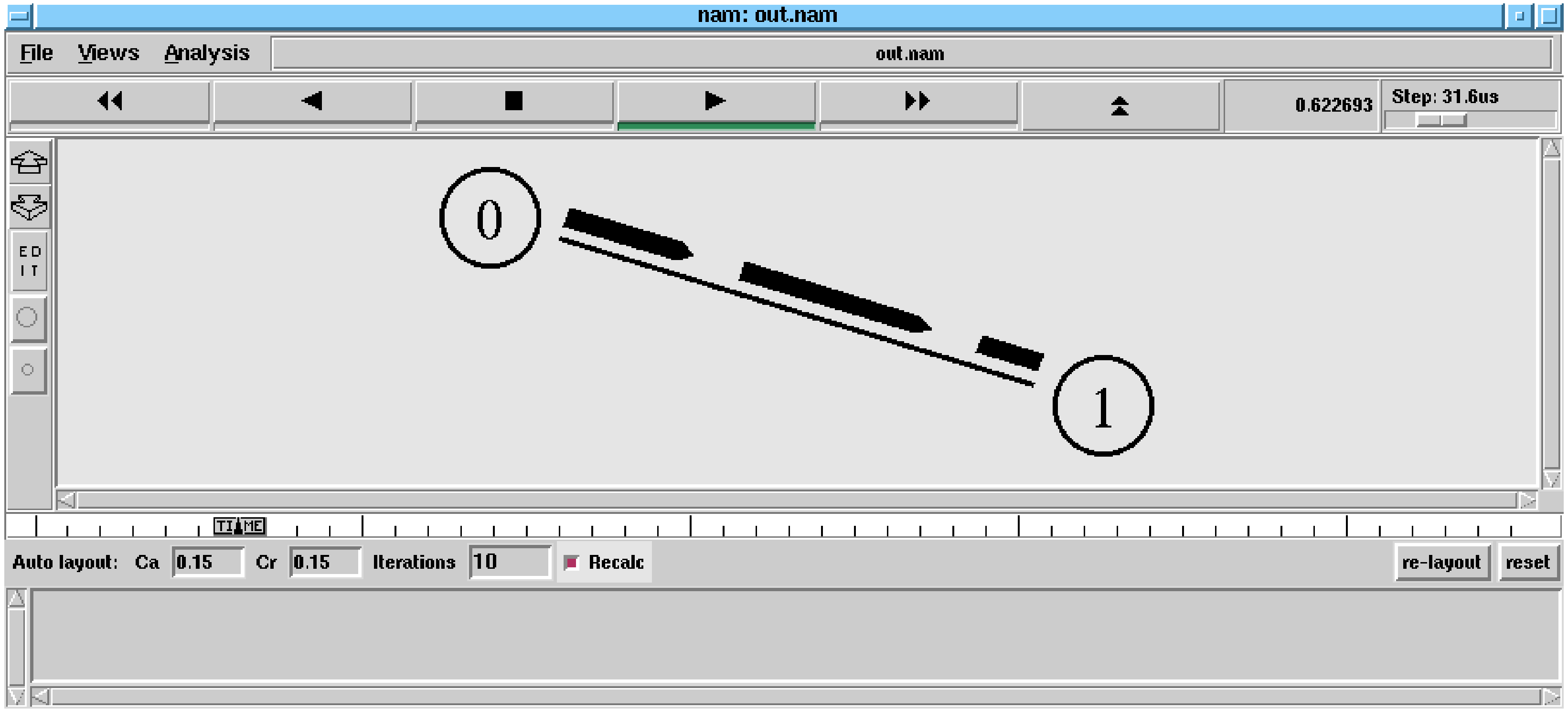
Dave Andersen, Xiaowei Yang

MIT

May 2001

Welcome to $ns - 2$

The de-facto standard for network simulation



ns **features**

- Discrete event simulator
- Understands TCP, UDP, FTP, HTTP, and more
- Wired and wireless networks
- Topology generation
- Visualization tool (*nam*)
- Public source that you can hack up

ns the language

What's it written in?

- Object-oriented TCL (otcl) and extensions
- tclcl: Use C++ and otcl with TCL

How's it structured?

- C++ core: event simulator, basic protocols
- oTCL and TCL: Scripts, some protocols

Scripts

- Instantiate nodes, links, senders, receivers
- Control timing of events
- Trace important events for later analysis

Scripts are in TCL

The Tool Command Language is a simple scripting language commonly used to drive other programs.

```
set a 14
```
a = 14... set using name

```
set b $a
```
b = 14... dereference with $

```
set c [expr $a + $b]
```
evaluate with [

```
set colors {red green blue}
```
lists...

More TCL: Procedures

```
proc foo {arg1 arg2list} {
  set tmp 0
  foreach arg $arg2list {
      set tmp [expr $tmp + $arg]
  }
  return [expr $arg1 * $tmp ]
}

foo 2 { 1 1 1 }            2 * 3 = 6
```

oTCL: Objects

```
Class Bagel                  Define Bagels..
set food [new Bagel]         Instantiate one
$food info class             Returns "Bagel"
```

Initializers:

```
Bagel instproc init {args} {
 $self instvar toasted_
 set toasted_ ''nope''
 eval $self next $args
}
```

oTCL: Object methods

```
Bagel instproc status {} {
  $self instvar toasted_
  return $toasted_
}
```

Access:

```
set food [new Bagel]
$food status
```

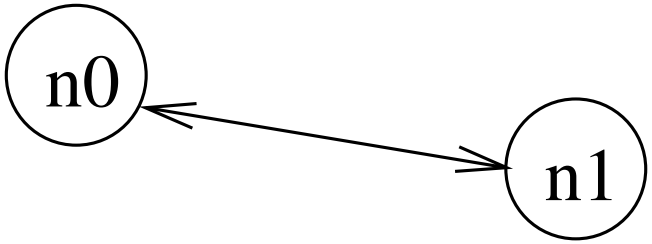
Go shopping

Returns “nope”

Starting up

`set ns [new Simulator]`	Create *one*!
`$ns at 5.0 ''finish''`	Run for 5 sec
`$ns run`	Get going!

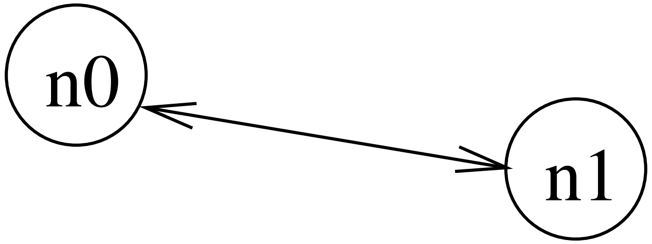
Create a topology




```
set ns [new Simulator]
$ns at 5.0 ``finish''
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms Droptail
$ns run
```

Of course, this still doesn't *do* anything

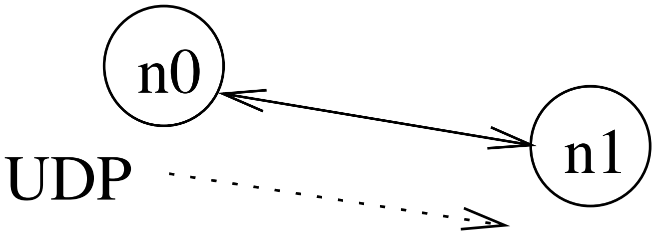
About links



`$ns link src dst bw delay queue`

- Link: `simplex-link, duplex-link`
- bw: Bandwidth, e.g. 1Mb, 10Kb, etc.
- delay: One-way delay, e.g. 10ms
- queue: Queue discipline, e.g. DropTail, RED, CBQ, FQ, SFQ, ...

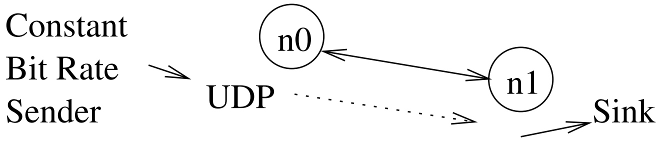
Create a transport layer




```
set UDPsrc [new Agent/UDP]
set null0 [new Agent/Null]
$ns attach-agent $n0 $udp0
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
```

UDP sender
UDP sink
n0 is sender
n1 is sink
Hook together

Send data!




```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

$ns at 0.5 ``$cbr0 start''
$ns at 4.5 ``$cbr0 stop''
```

Tracing

You want output? We've got output.

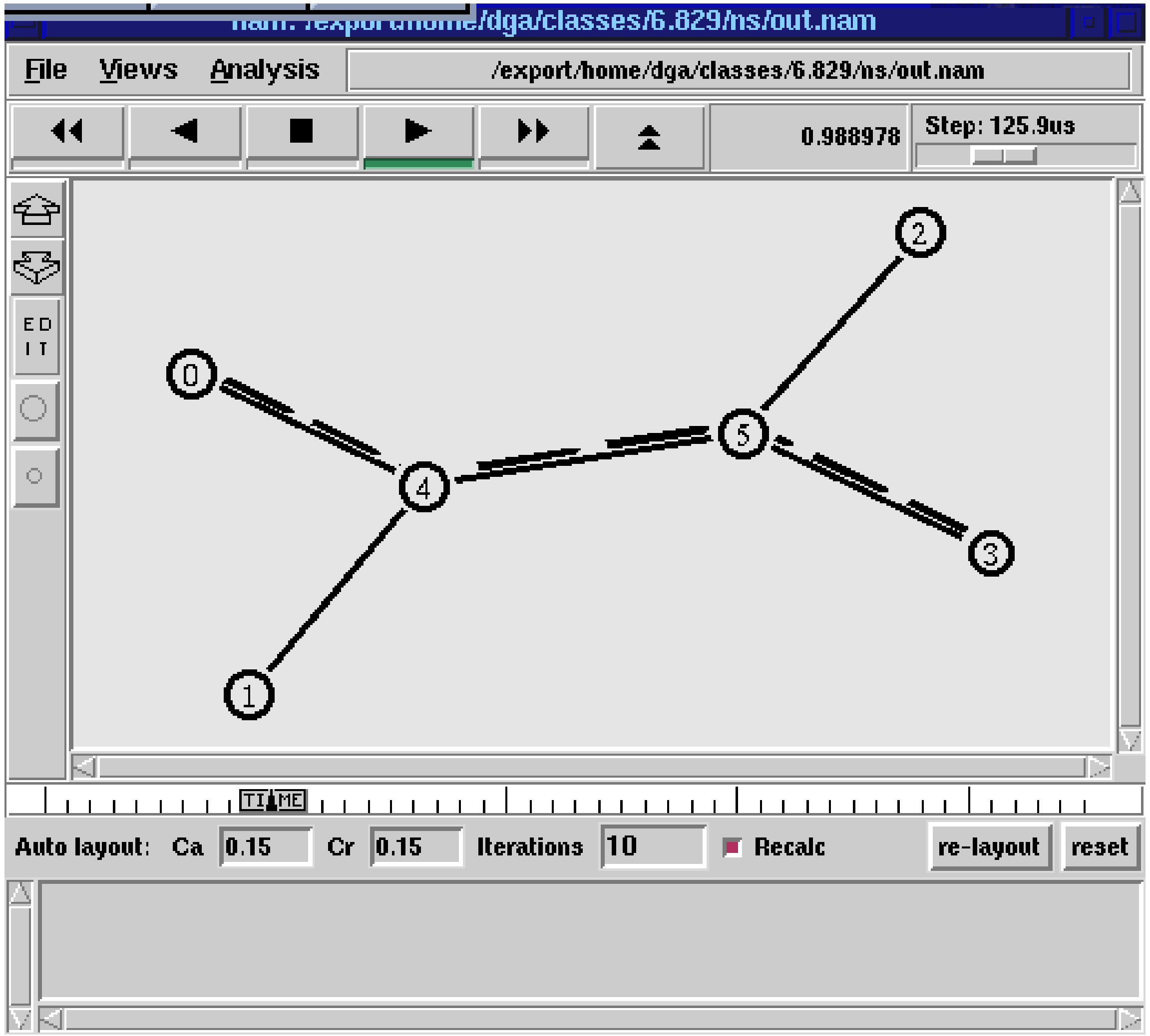
- All packets:

  ```
  $ns trace-all [open out.packets w]
  ```

- All packets, *nam* format:

  ```
  $ns namtrace-all [open out.nam w]
  ```

NAM: Visualizing simulations



NAM: Visualizing simulations

- Create a NAM trace file:

  ```
  $ns namtrace-all $namfile
  ```

- Invoke nam manually:

  ```
  shell% nam out.nam &
  ```

- ... or invoke it automatically...

Invoke *nam* automatically

```
set namfile [open out.nam w]
proc finish {} {
        global ns nf
        $ns flush-trace
        close $namfile
        exec nam out.nam &
        exit 0
}
$ns at 5.0 "finish"
```

Finding *ns*

- Athena (Linux x86): `add 6.829`
- Linux: Grab and compile http://www.isi.edu/nsnam/dist/ns-allinone-2.1b8a.tar.gz
- FreeBSD: ports/net/ns, ports/net/nam
- Windows: Good luck. http://www.isi.edu/nsnam/ns/ns-win32-build.html