

How to Use the Netbed (Emulab++) Network Testbeds

Jay Lepreau Rob Ricci Mac Newbold
University of Utah

SIGCOMM Tutorial

August 19, 2002

1

**So, you've built the next great
{distributed system, network
protocol, P2P app, etc.}**

**But, you need to test and
evaluate it**

2

Netbed Can Help

- At its base: machines with accounts (even root)
- We configure networks, but control is yours
 - Do whatever you want on/to nodes
 - Even install a new OS!
- All the amenities of home
 - Console access
 - Power control
- Incorporates other experimental environments
 - Wide-area nodes, simulated nodes
 - Use what makes the most sense for your experiment
- Simple stuff is simple; hard stuff (anything) is possible

3

So, Show Me!

Let's set up an experiment:

<http://www.netbed.org/>

4

Why?

- "We evaluated our system on five nodes."
 - job talk from university with 300-node cluster
- "We evaluated our Web proxy design with 10 clients on 100Mbit ethernet."
- "Simulation results indicate ..."
- "Memory and CPU demands on the individual nodes were not measured, but we believe will be modest."
- "You have to know the right people to get access to the cluster."
- "The cluster is hard to use."
- "We obtained guest accounts through 13 friends around the world to carry out our Internet measurements."

5

Common Misconceptions

- Unfamiliar environment
 - No, you typically get standard hardware and software
- Like a simulation, it must "run on its own"
 - No, you ask for just the features you want
- Lots of NS expertise required
 - No, there's a Java GUI for experiment configuration
 - No, configuration can be done with a subset of NS and cut-and-paste
- "Just a cluster"
 - No, configures network to emulate custom topologies
- "Just emulation"
 - No, support for real wide-area nodes & simulated nodes

6

What's a Node? What's a Router? (misconceptions)

- Physical hardware:
 - PC (local or remote)
 - (StrongARM box: in past)
 - (IXP1200, a specialized network processor: soon)
 - (Wireless: future)
- Virtual node:
 - Router (network emulation)
 - “Middlebox” (distributed system)
 - End host
 - A piece of a distributed node

7

What is Netbed / Emulab?

- A time- and space-shared platform for research, development, and education in distributed systems and networks
- A large software system
- Machines with configurable connectivity
- Emulab is the primary *emulation* portion of Netbed
 - www.emulab.net (Utah, 168 nodes, public)
 - uky.emulab.net (Kentucky, 48 nodes)
 - Georgia Tech (~50 nodes, soon)
 -

8

What is it (cont'd): Emulation Portion

- A configurable and controllable network emulator in a room
 - Utah Emulab today: 168 nodes, 1646 cables, 4 big switches
 - virtualizable topology, links, node software
- Bare hardware with lots of tools
- A controllable virtual world for distributed systems and networks

9

What is it? (cont'd)

- ... a base for physically distributed network testbeds and virtual (overlay) networks
- A way to get access to nodes all over the world
- An instrument for experimental CS research
- Universally available to any remote experimenter
- Simple to use

10

Utah Netbed Site



Kentucky Netbed Site

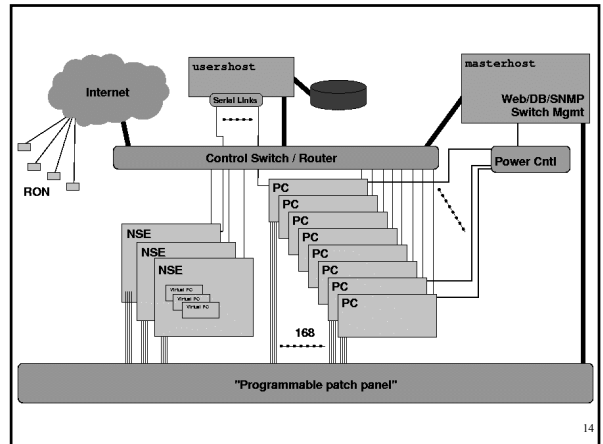


12

Distributed (Wide-Area) Nodes



13



14

Node Types In Utah Emulab Today

- pc600 (40)
 - 600MHz processor
 - 256 MB RAM
 - 13 GB IDE disks
- pc850 (128)
 - 850MHz processor
 - 512 MB RAM
 - 40 GB IDE disks

15

On With How to Use It

16

Getting Started

- **Visit the website at www.netbed.org**
- **Apply to *start or join a project***
 - Creates a new user account
- **Create an *experiment***
 - Topology/configuration specified with
 - a Java GUI, or
 - an *ns* file
- **Start using your experiment!**

17

www.netbed.org (emulab.net)

- **Most work can be done through our web interface**
 - Beginning/ending experiments
 - Applying for/approving access
 - Controlling nodes
- **Searchable documentation**
- **Secure access using https**

18

A "Project"

- **Central administrative entity**
- **Started by a faculty member or senior student**
 - Submitted through web interface
 - User account gets created for experiment leader
- **Approval of project users delegated to leader**
 - Saves on administrative overhead
 - Project leader responsible for users' behaviour
- **Project gets its own disk space**

19

An "Experiment"

- **Central operational entity**
- **Represents network configuration, including**
 - Network links
 - Node configuration
 - May include traffic generations, event stream
 - May simply be some allocated machines!
- **Created with an *ns* file or a simple GUI**
- **Started through web interface**
- **Mail sent when setup is complete**

20

The Netbed Documentation

- At <http://www.netbed.org/doc.php3>
- **Searchable with WebGlimpse**
- **Also useful**
 - NS-2 documentation
 - www.isi.edu/nsnam/ns/ns-documentation.html
 - TCL books, manuals, etc.

21

Experiment Creation Mail

- **Virtual Node Information**
- **Physical Node Mapping**
- **LAN/Link Info**
- **Delay Node Info**
- **Log of experiment creation**

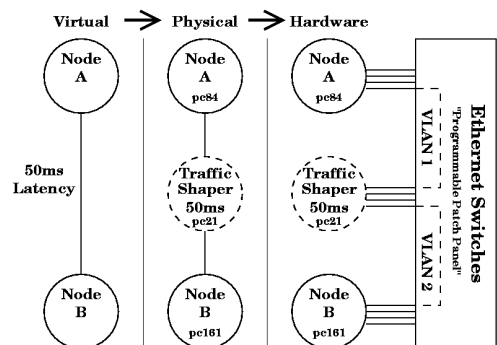
22

VLANs and Delay Nodes

- Isolation done with Virtual LANs (VLANs) on our switches
- Traffic shaping done with transparent bridges
 - Invisible to nodes
 - Regular nodes running FreeBSD
 - dummynet used for traffic shaping
 - Listens for events related to its links

23

VLANs and Delay Nodes - Diagram



24

Introduction to Using Your Experiment

25

Nodes

- **Logging into nodes**
 - ssh access
 - Add public keys via our web interface
 - Fully-qualified names
- **Shared NFS home directory**
- **Root access via sudo**
- **Testbed-specific configuration in /etc/testbed**
- **You're free to do whatever you want to them – disks get reloaded afterwards**

26

Web

- **Web control of running experiments**
 - View experiment report
 - Swap in/out
 - View NS file and visualization
- **Node control**
 - Set OS
 - Add RPMs, tarballs, startup scripts, etc.
 - Reboot node
 - Access to node serial console

27

users.emulab.net

- **Commands available on users.emulab.net**
 - node_reboot -reboot/power cycle
 - os_load - recover scrogged disks
 - portstats - see switch port counters
- **'console' - serial console access**
- **Disk space:**
 - /users – small stuff
 - /proj – bigger stuff (shared among members of the project)

28

Serial Consoles

- Link on node page
- Requires some setup
 - Download `tiptunnel` (Windows, Linux, FreeBSD binaries available)
 - Install wherever convenient
 - Associate file type with downloaded binary
- All output logged on **users.emulab.net**
 - `/var/log/tiplogs/<physid>.run`

29

NS Specifics

30

Audience Familiarity With NS

- Use it all the time?
- Use it a little?
- Have used TCL, but not NS?
 - NS scripts are written in TCL
- Never used either?

31

Boilerplate

- Statements required in every Netbed NS file
- `set $ns [new Simulator]`
 - Creates a new NS “simulator object”
- `source tb_compat.tcl`
 - Load testbed-specific commands
 - Stub version provided for running in NS
- `$ns run`
 - In NS, runs the simulation

32

Nodes – Netbed-Specific Commands

- `tb-set-node-os nodeA FBSD-STD`
 - Set OS. Currently supported:
 - `FBSD-STD`
 - `RHL-STD`
 - `<your own>`
- `tb-set-hardware nodeA pc600`
 - Pick specific PC type: `pc600/pc850`
 - `pcvron/pcvwa`

33

Links

- `$ns duplex-link $nodeA $nodeB 100Mb 0ms DropTail`
 - Set bandwidth and/or latency
 - Queuing types: `DropTail`, `RED`, `GRED`
- **Naming links:**
 - `set link0 [$ns duplex-link ...]`
 - Always name your links
- `tb-set-link-loss $link0 0.05`
 - Ratio of lost packets: 1.0 means drop all packets

34

LANs

- `$ns make-lan "$nodeA $nodeB $nodeC" 100Mb 0ms`
- **Naming works the same as with links**
- **Setting packet loss on a LAN**
 - `tb-set-lan-loss $lan0 0.01`
- **Setting different characteristics for a single node:**
 - `tb-set-node-lan-delay $lan0 $nodeA 40ms`
 - `tb-set-node-lan-bandwidth $lan0 $nodeA 20Mb`
- **Unlike links, no queuing discipline**

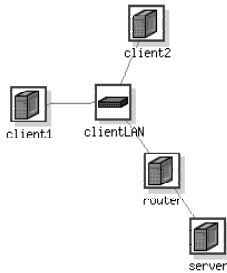
35

Routing

- **Types of routing**
 - Manual - You specify
 - Static - Computed by testbed software
 - Session – Dynamic (OSPF), using gated
- `$ns rtproto Static`
 - Set routing type
- `$client add-route $server $router`
 - Adds routes when using Manual routing

36

LAN Example



37

LAN Example NS File

```
set ns [new Simulator]
source tb_compat.tcl

$ns rtproto Static

set server [$ns node]
set router [$ns node]
set client1 [$ns node]
set client2 [$ns node]

set serverLink [$ns duplex-link $router $server 1.5Mb 30ms DropTail]
tb-set-link-loss $serverLink 0.01

set clientLAN [$ns make-lan "$client1 $client2 $router" 100Mb 0ms]

$ns run
```

38

Traffic Generation

- Standard NS
- 3 Parts
 - Agent: TCP/UDP socket
 - Gets attached to a node
 - Application
 - Generates traffic, attached to an agent
 - Sink
 - Connected to the agent, just discards traffic
- Has to be started with an event

39

Traffic Generation (cont'd)

```
set tcp0 [new Agent/TCP]
$ns attach-agent $nodeA $tcp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1200
$cbr0 set rate_ 100Mb
$cbr0 attach-agent $tcp0

set null0 [new Agent/Null]
$ns attach-agent $nodeB $null0

$ns connect $tcp0 $null0

$ns at 1 "$cbr0 start"
```

40

Program Objects

```
set prog0 [new Program $ns]
$prog0 set node $nodeA
$prog0 set command "/users/ricci/dostuff args"

$ns at 10 "$prog0 start"
$ns at 20 "$prog0 stop"
$ns at 30 "$prog0 start"
```

41

Constants

- **Makes it easy to change operating systems**
 - set OS FreeBSD45-STD
 - tb-set-node-os nodeA \$OS
 - tb-set-node-os nodeB \$OS
- **Makes it easy to set node types**
- ... to set bandwidth
- ... to set latency
- ... etc.

42

Loops

```
set num_nodes 20

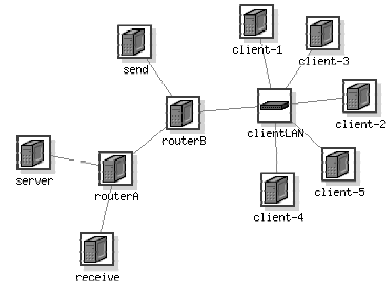
for {set i 1} {i <= $num_pcs} {incr i} {
  set pc($i) [$ns node]
  tb-set-node-os $pc($i) FBSD-STD
  append lan_string "$pc(${i}) "
}

set lan0 [$ns make-lan "$lan_string" 100Mb]

$pc($i) gets converted to "pc-$i" in node names
```

43

Large Example



44

Large Example NS File

```
set ns [new Simulator]
source tb_compat.tcl
$ns rtproto Static

set num_clients 5
set server_os FBSD-STD
set client_os RHL-STD

set server [$ns node]
set routerA [$ns node]
set routerB [$ns node]
set send [$ns node]
set receive [$ns node]
for {set i 1} {i <= $num_clients} {incr i} {
  set client($i) [$ns node]
  tb-set-node-os $client($i) $client_os
  append lan_string "$client(${i}) "
}
```

45

Large Example NS File (cont'd)

```
tb-set-node-os $server $server_os

set routerLink [$ns duplex-link $routerA $routerB 100Mb 0ms DropTail]
set serverLink [$ns duplex-link $routerA $server 100Mb 0ms DropTail]
set sendLink [$ns duplex-link $routerB $send 100Mb 0ms DropTail]
set receiveLink [$ns duplex-link $routerA $receive 100Mb 0ms DropTail]
set clientLAN [$ns make-lan "$lan_string $routerB" 100Mb 0ms]

set tcp0 [new Agent/TCP]
$ns attach-agent $send $tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1200
$cbr0 set rate_ 50Mb
$cbr0 attach-agent $tcp0

set null0 [new Agent/Null]
$ns attach-agent $receive $null0
$ns connect $tcp0 $null0

$ns at 1 "$cbr0 start"
```

46

Large Example NS File (cont'd)

```
set server_prog [new Program $ns]
$server_prog set node $server
$server_prog set command "/proj/testbed/bin/serverprogram"
$ns at 1 "$server_prog start"

$ns run
```

47

RPMs and Tarfiles

- `tb-set-node-rpms $node a.rpm`
 - Convenient way to install Linux packages
 - Installation is forced
 - Can specify multiple RPMs on one line
- `tb-set-node-tarfiles $node`
 - Arguments: alternating directory and tarball paths
 - Changes to directory before untarring
 - Untars as root (owner in tarfile still applies)

48

Startup Commands

- `tb-set-node-startup $node "command"`
 - Script should be in home or project directory
 - Command is run as experiment creator
- Differences from Program Objects
 - Executed every time node boots
 - No synchronization
- Uses
 - Tweak node configuration (routing, etc.)
 - Run services

49

Setting Node IP Addresses

- Assigned for you automatically if omitted
 - Recommended
 - Uses a deterministic algorithm
- `tb-set-ip $node IP`
 - Use only for single-interface nodes
- `tb-set-ip-link $node $link IP`
- `tb-set-ip-lan $node $lan IP`

50

Existing Tools

- **Can use existing topology generators**
 - Tiers
 - GT-ITM
 - BRITE
- **Anything that exports NS**

51

More Netbed Control

52

Swapping an Experiment

- **Release hardware resources without ending experiment - OS analogy**
- **Experiment information is maintained in DB**
- **Can easily swap back in - a few minutes**
- **We typically have more experiments swapped out than in, at any point in time.**
- **Role of node state in determining & specifying swappability**

53

Swapping an Experiment – Soft State

- Soft state is the part not saved on swapout
- It includes
 - Contents of nodes' local disks
 - Effects of dynamic events (next slides)
- Hard state includes
 - Things in your home directory
 - Anything given in the NS file
- Disk contents can be saved in disk images

54

Event System - Overview

- **Used for distributed control**
 - Starting/stopping programs
 - Controlling traffic
 - Changing link characteristics
- **Underlying publish/subscribe system**
- **Static events can be injected by NS scripts**
- **Dynamic events can be injected by hand**
- **Users can write their own programs that hook into the event system**

55

Event System – Static Events from NS Scripts

- **Link control**
 - `$ns at 10 "$link0 down"`
 - `$ns at 20 "$link0 delay 5.5ms"`
- **Traffic control**
 - `$ns at 5.5 "$cbr0 start"`
- **Program control**
 - `$ns at 1 "$prog0 start"`
- **Loops, of course...**

56

Event System – Dynamic Events

- **tevc**
 - Available on nodes or `users.emulab.net`
 - Arguments
 - `"-e pid/eid"` (Only required if used on users)
 - Time (`now`, `+seconds`, or `[[[yy]mm]dd]HH]MMss`)
 - Object
 - Event
- **Examples**
 - `tevc now cbr0 start`
 - `tevc -e testbed/foo +30 link0 set delay=50`

57

Virtual Types

- Allow you to specify that a set of nodes should be of the same type, chosen from a set of possible types
- Make an equivalence class (virtual type)
- Set nodes to be that virtual type
 - Instead of a physical type
- Two kinds of virtual types
 - Soft – Will allow exceptions if resources are scarce
 - Hard – Swapin will fail if class cannot be satisfied

58

Virtual Types – In Your NS File

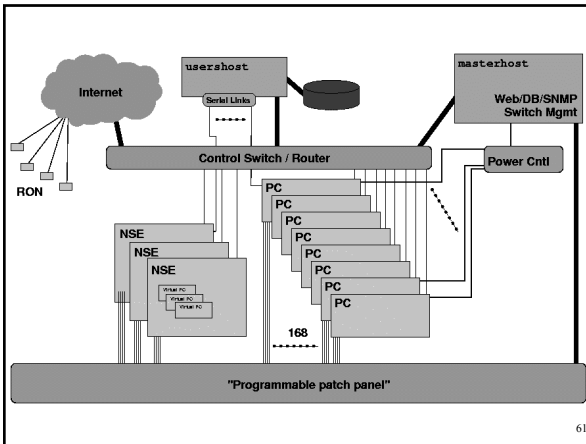
- `tb-make-soft-vtype vtype {types}`
- `tb-make-hard-vtype vtype {types}`
- `tb-set-hardware $node vtype`
- **Currently, types can be**
 - `pc600`
 - `pc850`
 - Any widearea types

59

Physically Distributed Nodes

- **Netbed provides access to distributed nodes**
 - Machines from MIT's "RON testbed" (32 as of this writing)
 - Includes Internet2, DSL, and international sites
 - Access policy is more restricted
 - PlanetLab machines
 - Support is evolving
- **Supported features**
 - Account management, ssh key management
 - Optional tunnelling (virtual links)
 - Traffic generation
 - SFS secure distributed filesystems

60



- ## Wide Area Resources
- An experimenter can request
 - N random nodes
 - N specific nodes
 - N, M, nodes of certain "last-mile" types: `pcinet2`, `pcint1`, `pcdsl`, `pcinet`
 - As above, but just a piece of a physical node: a "virtual node"
 - N nodes, and M links between them with particular characteristics (can specify any of latency, bw, loss rate).
 - In all these cases, Netbed finds the best matching nodes/links from its DB, updated frequently from MIT's realtime data.
- 62

- ## Requesting Physically Distributed Nodes
- **Specifying specific nodes**
 - `tb-fix-node nodeA ron0`
 - **Specifying general classes**
 - `tb-set-hardware nodeA pcrinet2`
 - **Specifying link characteristics**
 - `$ns duplex-link $nodeA $nodeB 1.5Mb 10ms`
- 63

- ## Widearea Demos
- Simple matching, without tunneling
 - More complex matching, with tunneling
- 64

- ## Using Purely Simulated Nodes
- **NSE – The NS emulation facility**
 - **Allows NS to interact with real network**
 - **Packets inside NSE can be converted into real packets and sent on the network**
 - **Packets on the network can be converted into NSE packets, travel through the simulated network, and then return to the real network**
- 65

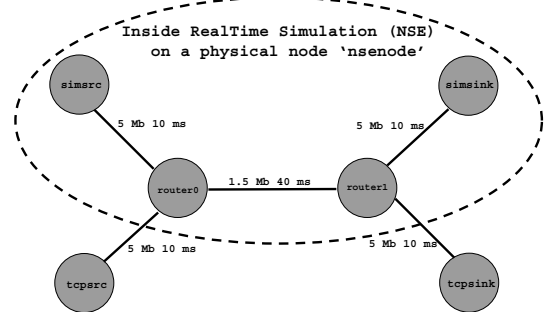
- ## Using Simulated Nodes (contd.)
- **How to specify simulated nodes in your NS file**
 - **Create an NSE node (physical machine running NSE):**
 - `set nsenode [$ns nsenode]`
 - **Make objects in the simulated world:**
 - `$nsnode make-simulated { # Simulated node`
 - `set simnode [$ns node]`
 - }
 - **Anything inside `make-simulated` is processed by NSE on `$nsnode`**
- 66

Using Simulated Nodes (contd.)

- **Connections between live/simulated networks are configured automatically (needs to be specified outside make-simulated block)**

67

Simulation Integration Demo



68

Simulation Integration Demo – NS File

```
set ns [new Simulator]
source tb_compat.tcl
$ns rtproto Static

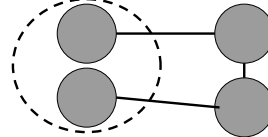
# Hybrid dumbbell topology
set tcpsrc [$ns node]
set tcpsink [$ns node]
set nnode [$ns nnode]
$nsnode make-simulated {
    set router0 [$ns node]
    set router1 [$ns node]
    $ns duplex-link $router0 $router1 1.5Mb 40ms DropTail

    set simsrc [$ns node]
    $ns duplex-link $simsrc $router0 5Mb 10ms DropTail
    set simsink [$ns node]
    $ns duplex-link $simsink $router1 5Mb 10ms DropTail
}
$ns duplex-link $tcpsrc $router0 5Mb 10ms DropTail
$ns duplex-link $tcpsink $router1 5Mb 10ms DropTail
$ns run
```

69

NSE Caveats

- Our support is still young
- Can have trouble keeping up with too much traffic or too many simulated nodes
- Multiple paths between NSE node and real nodes can be problematic



70

Simulation Integration Demo

71

Batch Experiments

- **Batch queue**
- **Runs whenever enough nodes become available**
- **When startup command finishes, experiment is automatically terminated**
- **Great for:**
 - Fitting in large experiments
 - Exploring many topologies/parameters
 - Having work done for you while you sleep!

72

Creating Batch Experiment From the Command Line

- Often easier than submitting the same web form many times
- `batchexp` on `users`
- Main arguments:
 - “-p project”
 - “-e experiment”
 - nsfile

73

Custom Disk Images

- **When to use a custom disk image**
 - Custom kernels
 - Extensive OS changes
 - Your own custom OS
- **Loading time**
 - 88 seconds for a single partition - 150MB compressed

74

Using a Custom Disk Image

- **Creating - web form**
 - Small web form to fill out (‘OSIDs and ImageIDs’) link
 - Image gets created automatically
 - [Demo]
- **Specifying in NS file**
 - Automatically loaded for you
 - `tb-set-node-os nodeA FBSD45-MINE`

75

Debugging Experiments

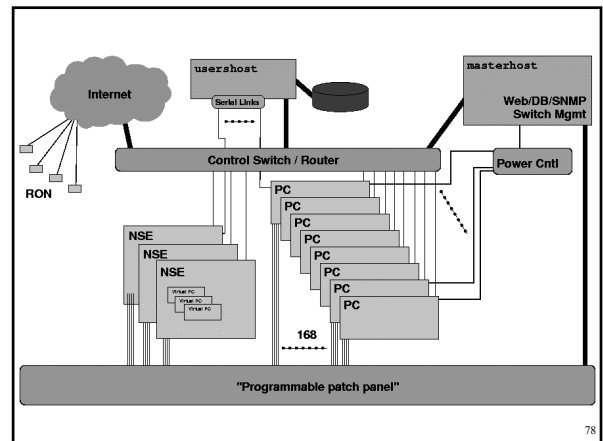
- **Some common error messages**
 - “Failed to map to reality”
 - Typically: not enough free nodes
 - Recommended approach:
 - Verify against “# of free PCs”
 - Make request less specific (`pcxxx -> pc`)
 - Try again later
 - Use batch system
 - “pcXXX appears to be dead”
- **Where to find log files**
 - `/proj/<proj>/exp/<expt>/log/...`

76

Recovering From Disasters

- Can always do a good old reboot
 - First, we try a graceful reboot
 - Then, we try our custom ‘ping of death’ (ipod)
 - If all else fails, power cycle
- If the network is down
 - Get in on the serial console
- If all else fails
 - Reload the disk (`os_load` on `users`)

77



78

Control vs. Experimental Nets – Differing Purposes

- Control
 - NFS (homedir), DNS, node monitoring
 - Routable to outside world (you log in via it)
 - Not completely isolated today
- Experimental
 - Isolated – no interference from other experiments
 - Configured in the topology you requested
 - ‘Clean’ – no stray traffic

79

Control vs. Experimental Nets – How To Tell Them Apart

- IP addresses
 - Control net has ‘real’ IPs
 - Experimental net has 192.168.*.* or 10.*.*
- /etc/testbed/control_interface
 - Prints name (ie. ‘eth0’ or ‘fxp4’) to stdout
- If you were expecting delays, bandwidth limits, etc., but don’t get them, you may be using the control net by accident

80

Control vs. Experimental Nets - Naming

- Outside of the nodes
 - Only control net is nameable/reachable
- On the nodes
 - Unqualified names (eg. `nodeA`) refer to directly-connected experimental interfaces
 - Can refer to any experimental interface as ‘<node>-<link>’ (`nodeA-link0`, `nodeB-clientLAN`)
 - Qualified names (eg. `nodeA.myexp.myproj`) refer to control net

81

Barrier-like Synchronization

- Simple barrier synchronization provided by `tmcd`: the “ready count”
- Nodes can report ready
- Poll for how many other nodes, out of the total number, are ready
 - Make sure to delay a few seconds
- Simple text-based protocol; simple scripting interface

82

Under the Hood

83

Netbed Servers

- **Hardware: Netbed Servers**
- **boss.emulab.net**
 - Secure server, no direct access for users
 - Hosts the web server and database
 - Controls everything
- **{users,fs,ops}.emulab.net**
 - Accounts and home dirs for everyone
 - NFS server for boss, nodes
 - Access to node consoles

84

Software and Experiments

- **Software base:**
 - Web site is PHP, Database is MySQL, NS parser is TCL, back end is mostly perl and C
- **Four main steps to running an experiment**
 - Pre-run: parse NS file, store in DB
 - Swap-in: map expt. to phys. nodes, set up state in DB, reboot nodes, configure nodes
 - Swap-out: Clean up nodes, release them
 - End: Clean out data for experiment
- **Experiment may swap in/out many times**

85

Selected Hard Problems

- **Resource mapping**
 - NP-hard problem (simulated annealing)
 - Minimize inter-switch bandwidth
 - Make efficient use of node features
- **Experiment swap-in**
 - Automate many system administration tasks
 - Must deal with hardware failures at any time
 - Many automatic conveniences for ease-of-use
- **Disk reloading**
 - Multicast disk loader: Frisbee (think "flying disks")
 - Loads 50 nodes simultaneously in 100 seconds

86

Node Boot Process

- **Obtains IP through DHCP**
- **NIC boots custom PXE program**
- **Queries boss for which OS to boot**
 - Can boot from disk or network
- **Boots into selected OS**
- **Contacts tmcld for configuration**
 - Accounts, IPs, software to install, delay configuration, traffic generation, etc.

87

How Has Netbed Been Used?

- **Armada (Dartmouth)**
 - Parameter-space exploration
 - Hundreds of batch experiments
- **WanSpread (Johns Hopkins)**
 - Emulated the CAIRN testbed
 - Tried variations with delays doubled and halved
- **SANDS (TASC)**
 - Large topologies, custom disk images
- **Spinglass (Cornell)**
 - Fault tolerant group communication

88

What Is It Not Good For?

- Packet-level expts. across many nodes
 - Clock synchronization good, but not perfect
 - Non-determinism in the real world
- Experiments that require real routers
 - All nodes are PCs
 - But, we can use a few different queuing strategies
 - And, you can reprogram them all you want
- Experiments that require gigabit links
 - None yet, but we hope to add some
- Experiments that need 1000s of links/nodes
 - ModelNet, coming soon, will help

89

Netbed In Education

- **Has been used by classes at remote institutions**
 - MIT (Balakrishnan, Andersen)
 - Kentucky (Griffioen)
 - Harvey Mudd (Kuenning)
- **Group model, to give TAs control over student experiments**
- **Safe to give students root access**
- **In OS classes, students can replace kernels, etc.**
- **For networking classes, students can run on an emulated network**

90

Guest Segment: Experiences with Emulab in Education

Jim Griffioen
University of Kentucky

91

OS/Network Projects

- Possible Approaches
 - Simulation/Software Emulation
 - ns, cnet, jns, jnetsim, netsim, opnet, nachos, csim, ...
 - Overlay Techniques
 - Xbone, multicast-based emulation, ...
 - Dedicated Facilities (networks and machines)
 - Requires significant \$, space, tolerant sys-admins, scheduled used/reconfig
- Other Issues
 - Applications and realistic traffic generators
 - Policies/mechanisms for sharing/access
 - Monitoring/Tracing/Debugging
 - Learning curve and long-term utility of acquired training
 - Assistance/Grading/Documentation

92

Why Emulab?

- shared resource – don't have to have your own dedicated facility (\$\$\$)
- sharing policies/mechanism already developed
- no sys admin (or wars with sys admins)
- arbitrary topologies
- reasonable learning curve
- well-known environments, real traffic, real applications
- real protocols
- good supplemental texts exist (i.e., good documentation)
- students will directly use the experience gained
- instructor access
- Standard debugging, tracing, traffic analyzer tools
- Language independence
- OS independence

93

Types of Projects

- What layers can students work at?
 - User-level applications and services (easy)
 - OS modifications
 - Module-based approach (relatively easy)
 - Modifying built-in components (can probably find a better way)
- Types of projects
 - Routing (ok but can mess up access to the machine)
 - Distributed systems/services (work well)
 - Dynamic network characteristic (doable but take effort)
 - Apps that require special I/O like audio, cameras, etc (have done but suggest avoiding these)
 - Apps that run over X (worked fine for us – YMMV)

94

Suggestions

- Simplify the learning curve
 - Provide preconfigured scripts, routing, etc as much as possible – students rarely have sys admin experience
 - Time spent teaching the Unix administration steps required by the project will be well spent (e.g., modifying the routing table)
 - Students are easily confused about things like home directory vs /proj directory, what is lost when swapping an experiment, node names and their scope, programs to run on users/ops, reboot vs power cycle, use of sudo, the importance of the control net interface, group access and sharing
 - TCL vs GUI (which is best depends on the student's background and ability)
- Emphasize responsible usage
 - Students forget they are tying up real (\$\$) machines
 - Comparing topologies is nice, but limit number and size of topologies
- Demonstrate debugging/tracing tools
 - Today's students are clueless
- Think about grading up front
 - Interactive grading sessions
 - Tarball with batch experiments
 - Students code for a well-defined emulab grading environment
- Don't forget the local environment
 - Necessary for code development and initial testing
 - Show students how to sync local environment with emulab

95

Questions and Feedback

- Audience questions
- What features would make Netbed more useful?
 - Most of our features are driven by user demand

96

Contributing to the Distributed Netbed

- **What we provide**
 - CD-ROM, maybe a disk sometimes
 - Working OS installation
 - Database state
- **What you provide**
 - Machine
 - Switch port
 - IP address
- **Caveats**
 - Security may be a concern
 - May consume bandwidth occasionally

97

Building Your Own

- **Our software is portable to other sites**
 - Kentucky has built their own
 - Georgia Tech is working on another
- **Lots of tradeoffs between price and usability**
 - Degree of nodes
 - Level of control (serial consoles, power control)
 - Big switches vs. stacks of small switches
 - Rack mount vs. desktop cases
- **Hardware recommendations on our website**

98

Ongoing and Future Work

- **Integrating Duke's "ModelNet"**
- Wide area, PlanetLab
- Federation
 - heterogeneous sites
 - resource allocation
- Wireless nodes, mobile nodes
- Hierarchical nodes (multiplex, VM)
- Pre-emptive swapout, rollback, "single-step"
- IXP1200 nodes, tools, code fragments
 - Routers, high-capacity shapers
- Scheduling system
- Packet capture, logging, visualization tools
- Microsoft OSs, high speed links, more nodes...

99

Conclusions

- Easy to use, while giving experimenters lots of control
- Suitable for distributed systems, network, and OS research and education
- Powerful NS/Tcl input language
- Integrates emulation, simulation, and wide-area experimentation
- Sign up for a project at www.netbed.org!

100

Afternoon Tutorial

- **Get a laptop with wireless support (alone or pair up)**
- **It will need to provide:**
 - Internet access
 - Web browser (Netscape/IE/Opera are tested)
 - SSH client
 - An editor (preferred but optional)
- **We provide pre-built accounts on Utah Netbed**

101

Available for universities, labs,
and companies, for research
and teaching, at:

www.netbed.org
www.emulab.net

102

Afternoon: The Lab Session

103

Using Your Guest Account

- Log in at www.emulab.net
- Optional: "Update User Information"
 - Change password
 - cracklib in use, good passwords only
 - Add ssh public key (link at bottom of page)
- Receive mail on users.emulab.net
 - Read mail directly
 - (or) Make a .forward file to send to another account

104

Using Your Guest Account (cont'd)

- Log into users.emulab.net via ssh
 - Hostname reported as 'ops'
 - Keep at least one shell on this machine open
- Make sure you can read mail
 - There should be one message already in your inbox
- Make sure you have an editor you're comfortable with
 - Either on `users`, or on your laptop

105

Experiments Overview

- Three experiments
 - First, get something simple going with our GUI
 - Next, make something a little more complex by editing NS files directly
 - Finally, use some advanced features to make a moderately complex experiment
 - Each one will build on the last
- We have a few example/template files on `users` in `/proj/tutorial/ns/`

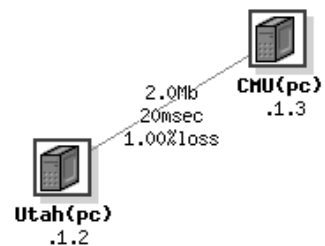
106

Starting an Experiment – NS Files

- Edit on your local machine
 - Use file upload box on experiment creation form
- Or, edit on `users`
 - Place file in your home dir or `/proj/tutorial/`
 - Your home directory is `/users/<username>/`
 - Put **full** path to NS file in form's textbox
- To get NS file from netbuild
 - Choose "Create Experiment"
 - Click "View NS File"

107

Experiment 1 Topology



108

Experiment 1

- Make two nodes (**Utah** and **CMU**)
 - Use NetBuild if your browser supports Java
- Link them together – name the link **link0**
 - Bandwidth 2Mb
 - 20ms one-way latency
 - 1% packet loss

109

Experiment 1 (cont'd)

- “Begin Experiment” when ready
 - Two things to enter:
 - Name, description
 - Pick any name, just make sure it's one no one else is likely to pick
 - Wait for experiment creation mail
 - Watch realtime experiment creation log
- Explore experiment page on web interface
 - Use “More Detail” link in visualization to verify parameters

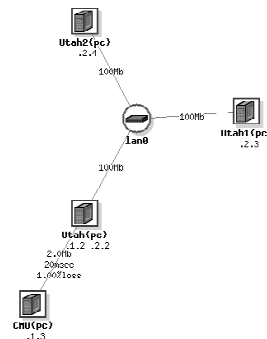
110

Experiment 1 (cont'd)

- Log into **Utah**
 - Ping on control and experimental interfaces
 - **CMU** (test network)
 - **CMU.<expt>.tutorial** (control network)
- Swap experiment out
- Swap experiment back in
- Terminate experiment

111

Experiment 2 Topology



112

Experiment 2

- Start with NS file from Experiment 1
- Add two new nodes (**Utah1** and **Utah2**)
- Make a LAN called **lan0** containing **Utah**, **Utah1**, and **Utah2**
 - 100 Mb, no latency or packet loss
- Install some software on **Utah**
 - `/proj/tutorial/rpms/trafshow.rpm`
- Set startup command for **Utah1**
 - `/proj/tutorial/bin/simplescript`
- Enable static routing

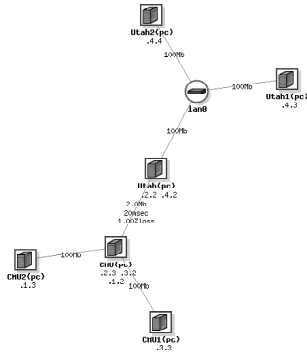
113

Experiment 2 (cont'd)

- Begin experiment
- Log into **Utah** and run **trafshow**
- Log into **CMU** and ping **Utah1-lan0** to confirm routing setup
- Log into **users.emulab.net**
 - Use ‘**console**’ to view a node’s serial console
 - Use ‘**node_reboot**’ or webpage to reboot it
- Terminate experiment

114

Experiment 3 Topology



115

Experiment 3

- Start with the NS file from Experiment 2
- Add two more nodes (**CMU1** and **CMU2**)
 - Connect them directly to **CMU** (full bandwidth, no delay)
- Set up two constants to set OSEs
 - **RHL-STD** for routers
 - **FBSD-STD** for end nodes
 - Set **Utah** and **CMU** to the router OS, and the other to the end node OS

116

Experiment 3 (cont'd)

- Create two traffic generators
 - One, sending TCP at 100Mb from **CMU1** to **Utah1** (call the application **cbr0**)
 - The other, sending UDP at 100Mb from **CMU2** to **Utah2** (call the application **cbr1**)
- Turn the first traffic on and off at 10 second intervals
- Leave the second traffic off

117

Experiment 3 (cont'd)

- Have **Utah1** prepare to run a program with a program object called **prog0**
 - `/proj/tutorial/bin/simpledaemon`
- Begin the experiment
- Log into an end node and check the OS
- Log into **Utah**
 - Find its interface to **CMU**
 - (Hint: Use `ifconfig` and experiment creation mail)
 - Run `trafshow` on that interface to watch TCP traffic go on and off

118

Experiment 3 (cont'd)

- Log into **users**
 - Start UDP cross traffic
 - `tevc -e tutorial/<expt> now cbr1 start`
 - Watch the TCP stream get clobbered with `trafshow`
 - Start and stop the program object
 - `tevc -e tutorial/<expt> now prog0 start`
 - Logs these events to `/tmp/simpledaemon.log`
- Terminate Experiment

119

Expt Creation Scaling

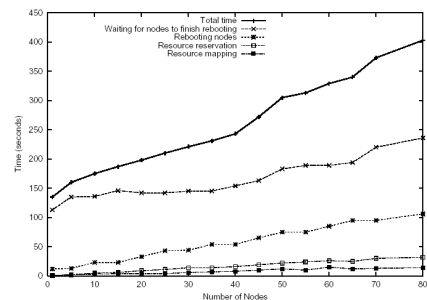


Figure 4: Time to create an experiment without disk loading. Stages shown

120

**Join the federation!
Or just use it.**

**Where network fantasies become reality:
www.netbed.org**

121

Bonus Slides

122

Who Uses Netbed?

- **Researchers**
 - Distributed systems
 - Networking (traditional and “active”)
 - Operating systems
- **Educators**
 - Advanced networking class at MIT
 - Basic networking class at Univ. Kentucky
 - OS class at Harvey Mudd College
 - Student projects
- **Advanced developers**
- [Browse project list on www.netbed.org]

123

Other Experimental Environments

- Simulation
 - Fast prototyping, easy to use, easy to control, but less realistic
- Live networks
 - Realistic, but hard to control, measure, or reproduce results
- Small static testbeds *emulating* a network
 - Real hardware and software, but hard to configure and maintain, lack scale

All 3 live on, implying both the continued importance and inadequacies of each

Key Points

- Netbed seamlessly *integrates* all three: simulation, emulation, and live networks
- Netbed’s primary goals: *ease of use, control, and realism*. Unlike the constituent approaches, meets all 3 goals simultaneously
 - Can mix and match in same experiment
- Netbed brings orders of magnitude improvements to the *emulation approach*: our focus today
- This all works *today*, and most is in full production mode for external users

125

Other Experimental Environments

- Simulation
 - Fast prototyping, easy to use, easy to control, but less realistic
- Live networks
 - Realistic, but hard to control, measure, or reproduce results
- Small static testbeds *emulating* a network
 - Real hardware and software, but hard to configure and maintain, lack scale

All 3 live on, implying both the continued importance and inadequacies of each

Key Points

- Netbed seamlessly *integrates* all three: simulation, emulation, and live networks
- Netbed's primary goals: *ease of use, control, and realism*. Unlike the constituent approaches, meets all 3 goals simultaneously
 - Can mix and match in same experiment
- Netbed brings orders of magnitude improvements to the *emulation approach*: our focus today
- This all works *today*, and most is in full production mode for external users

127

Primary Design Principles

- Transparency
 - Common specification language: *ns*
 - Common namespaces for nodes, links, agents...
- Virtualization
 - of all IP addr, hosts, hostnames, links, ...
 - Level of indirection allows
 - Control and configuration
 - Efficient time sharing (swapping to different physical resources)
 - Scalability via seamless multiplexing

128

Design Principles (cont'd)

- Automation
 - Replaces hundreds of steps of manual configuration
 - Arbitrary programmatic control through integrated event system and general-purpose PL for spec (Tcl)
- Efficiency
 - Of use of physical resources (space and time-shared)
 - Of experimenters' time: interactive style of use
- Policy today: conservative resource allocation

129

Simple NS file

```
set $ns [new Simulator]
source tb-compat.tcl

set nodeA [$ns node]
set nodeB [$ns node]

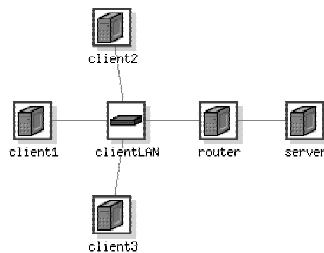
$ns duplex-link $nodeA $nodeB 100Mb 0ms DropTail

$ns run

# Comments look like this
```

130

Example Experiment Creation Mail – Topology



131

Example Experiment Creation Mail - Overview

```
User:      Robert P Ricci
EID:       example
PID:       testbed
GID:       testbed
Name:      An example experiment
Created:   2002-07-31 16:14:05
Expires:   2002-11-28 00:00:00
Started:   2002-07-31 16:19:18
Directory: /proj/testbed/exp/example
```

132

Example Experiment Creation Mail – Node Info

```

Virtual Node Info:
ID          Type      OS          Qualified Name
-----
server      pc          RHL71-STD  server.example.testbed.emulab.net
client2     pc          RHL71-STD  client2.example.testbed.emulab.net
client3     pc          RHL71-STD  client3.example.testbed.emulab.net
client1     pc          RHL71-STD  client1.example.testbed.emulab.net
router      pc          RHL71-STD  router.example.testbed.emulab.net

Physical Node Mapping:
ID          Type      OS          Physical
-----
client1     pc850     RHL71-STD  pc154
tbsdelay0  pc850     FBSD45-STD pc158
router      pc850     RHL71-STD  pc90
client2     pc850     RHL71-STD  pc113
client3     pc850     RHL71-STD  pc161
server      pc850     RHL71-STD  pc152

```

133

Example Experiment Creation Mail – LAN/link info

```

Lan/Link Info:
ID          Member      IP          Delay      BW (Kbs)  Loss
-----
clientLAN   client2:0    192.168.1.3  0.00       100000    0.000
clientLAN   client1:0    192.168.1.2  0.00       100000    0.000
clientLAN   router:1     192.168.1.5  0.00       100000    0.000
clientLAN   client3:0    192.168.1.4  0.00       100000    0.000
link0       router:0     192.168.2.2  30.00      1500      0.010
link0       server:0     192.168.2.3  30.00      1500      0.010

Delay Node Info:
ID          Virtual      Physical      Pipe Numbers
-----
link0       tbsdelay0   pc158         100,110

```

134