Testbeds
CS 239
Experimental Methodologies for System Software
Peter Reiher
May 15, 2007

Outline

- What is a testbed?
- Important shared testbeds
- Setting up your own testbed

What is a Testbed?

- A facility specifically devoted to running experiments
- Almost always with dedicated hardware
- Often with special software support

Components of a Testbed

- Computers
- A network
  - In most cases
  - Usually wired
  - Unless a specifically wireless testbed
- Supporting software

Purpose of a Testbed

- To set aside dedicated machines for testing
- Over a long period of time
- Often set up for a particular company or lab
- Recently, shared testbeds have become popular
  - Allows much larger testbeds
  - By sharing costs

Desirable Properties of a Testbed

- Sufficiently large
- Sufficiently modern hardware
- Flexibility in its use and control
- Ease of use in experiments
- Evolvable
- Sharable, at least at some level

Important Testbeds

- Emulab
- Planetlab
- Deter
- GENI

Emulab

- Large testbed located at University of Utah
- Funded initially by NSF and DARPA
- Designed to support experiments by researchers worldwide
- Probably the first really successful Internet-wide testbed
- http://www.emulab.net

Basic Philosophy of Emulab

- Provide large pool of machines to entire Internet community
- Almost all testing will be done remotely
- Almost all testing must be done without intervention by testbed admins
- Handle the widest possible kinds of experiments and testing situations

Basic Emulab Approach

- Emulab indeed provides large numbers of machines
  - Around 450 total nodes
- But also provides a rich, powerful testing environment
- Completely configurable remotely
- Designed for simultaneous sharing by many users

Core Emulab Characteristics

- Highly configurable
  - System software
  - Application software
  - Network topology and characteristics
- Controllable, predictable, repeatable
- Good guarantees of isolation from other experiments

Emulab Use Policy

- Public resource open to most researchers
- Including commercial researchers
- And those in other countries
- Rules about abuse of system
- And priorities when overloaded
- But otherwise, anyone can run any experiment they want

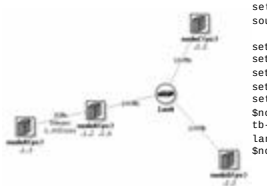
Using Emulab

- Start an Emulab project
  - Using on-line web form
  - Requires some description of what you'll be doing
  - Can also join existing project
- Log in to Emulab
- Set up and run an experiment

Running Emulab Experiments

- Must specify a network topology
  - Using NS-2 syntax
  - Includes specification of how many nodes you want, software used, etc.
- Use interface to start experiment
- Emulab automatically configures nodes as specified
- Experiment starts running
  - You can poke into your nodes during run
- When done, terminate the experiment

Simple Emulab Example

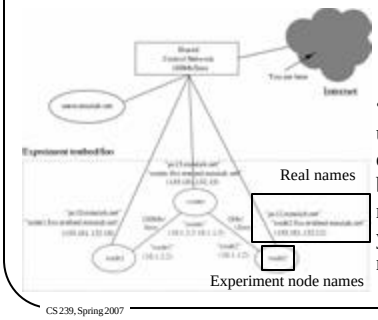


```
set ns [new Simulator]
source tb_compat.tcl

set nodeA [$ns node]
set nodeB [$ns node]
set nodeC [$ns node]
set nodeD [$ns node]
set link0 [$ns duplex-link $nodeB
$nodeA 30Mb 50ms DropTail]
tb-set-link-loss $link0 0.01 set
lan0 [$ns make-lan "$nodeD $nodeC
$nodeB " 100Mb 0ms]
```

- Nodes B, C, and D are connected via a 100 Mb LAN
- Node A connects to node B by a 30 Mb link

What's Really Going On



- Important to understand difference between real node names and your experiment node names

Emulab and Operating Systems

- Emulab configures each node with the OS you choose
  - Supported choices are FreeBSD, multiple Linux variants, Windows XP
  - Can also run OSKit kernels on some Emulab nodes
- Fresh instantiation of OS on each machine
- Don't get root password, but full sudo root access
- Allowed to customize pretty much any way you like
- Specified in the NS-2 config file

Emulab and Network Issues

- You can set up whatever network bandwidth you want
  - Up to 100 Mbps
- Can specify the network delays you want
- Can specify network queueing disciplines
- Multiple routing options for the network

Wireless in Emulab

- Some Emulab nodes have 802.11 cards
  - Some also have GNU Software Radio hardware
- More wireless nodes are set to be added soon
- Also a dense array of wireless nodes
  - For studying interference effects

Emulab and Internet Experiments

- Emulab has ability to link to Planetlab
  - Which is on the real Internet
- Use from within Emulab somewhat different than normal Planetlab use

Other Emulab Facilities

- Small sensor network (25 Mica nodes)
- Robotic-based testbed for mobile wireless experiments
  - Small number of robots with attached wireless
  - Controllable movement in a small space
  - Predecessor of larger testbed of this type
- Hybrid simulation capabilities

Other Emulabs

- The basic hardware and software has been adapted to build other testbeds
  - Mostly much smaller
- Some are for public use, others for private use
- Some are for specialized purposes

Emulab and Network Topologies

- Can specify any topology expressible in NS-2 language
- But where do they come from?
- Generally, network topology generation programs
  - Emulab recommends BRITE
- We'll discuss topology generation in more detail later

Sample Uses of Emulab

- Testing RON
- D-Ward, DefCOM, and other DDoS defenses
- Benchmarking CORBA tools
- Testing collaborative cache consistency
- Testing Internet game systems
- Active network testing

Planetlab

- A testbed designed to test Internet services
- Using nodes deployed widely around the Internet
- And software to support safe and controlled sharing of the nodes
- Run primarily by Princeton, Berkeley, and Washington
- Funding seeded by NSF and DARPA
- Strong Intel participation
  - Other industry involvement, as well
- http://www.planet-lab.org
  - www.planetlab.org was taken

Basic Planetlab Concept

- Deploy testbed nodes at many locations throughout Internet
  - Standardized hardware and software
- Allow those who deploy nodes to use the testbed facility
- Provide virtual machines to each tester using a node

Planetlab Nodes

- Hardware deploying the Planetlab software package
- Which support cheap virtual machines
- Otherwise, provides a typical Linux environment
- Pretty complete control of virtual machine
- But node-based mechanisms to ensure fair and safe sharing of hardware

Planetlab Locations

Usually two machines per location
788 nodes at 382 sites (as of 5/12/07)

Planetlab Experiments

- Usually run on many Planetlab nodes
- By one controlling researcher
- The collection of resources across all nodes supporting the experiment is called a Planetlab *slice*
  - A multimachine environment for the experiment
  - Also an organization for cooperating researchers to use
- *Services* run in slices

More On Slices

- Slices have computing resources associated with them
  - Processing, memory, storage, network bandwidth
  - On each participating node
- Networks of virtual machines
- In this sense, Planetlab is an overlay testbed

Planetlab Virtual Machines

- Multiple slices can co-exist on the same virtual machine
- Uses Linux VServers to create virtual kernels
  - Virtualization at system call level
  - Harder to ensure real separation than true virtual machines, like VMWare
  - But cheaper to run
- Semi-copy-on-write techniques used to limit disk storage required for each kernel

Privileged Access in Planetlab

- Slice owner has root-like privileges on his node
- But not all root services
  - E.g., not raw device control or rebooting
- Root services only applied to the virtual machine in his slice
  - Can modify root file system, e.g.
- Separate sets of password files per slice

Networking in Planetlab

- Basically uses variant of sockets
- Can only get get sockets bound to particular UDP or TCP ports
- Incoming packets delivered only to service that created the socket
- Outgoing packets filtered for "well-formedness"
  - E.g., no IP spoofing allowed
- Internode communication uses standard Internet
- Planetlab has no special network or privileges

Planetlab and Deployment

- Planetlab was designed to allow eventually deployment of real services
- By running them in a slice
- Reasonable to run experiments for a long time over Planetlab, today

A Key Issue in Planetlab

- Traffic between nodes crosses the Internet
- No guarantees about state of that communications medium
- Makes reproducibility of results and control of experiments challenging
- But experiment experiences realities of Internet communications

Planetlab Administration

- Overall administration handled by testbed leaders and steering committee
  - Software releases, overall policies, handling requests to join
- Distributed administration at each site
- Site's Planetlab PI approves users and slice requests
- Less centralized than Emulab
  - But also less open

Some Sample Uses of Planetlab

- Testing DHT concepts
- Anycast and multicast projects
- Measurement of Internet behavior and topology
- Video streaming research
- Protocol resiliency and survivability
- Lots of P2P work

Core Differences Between Emulab and Planetlab

- Emulab is centralized
- Planetlab is distributed
- Emulab is highly controllable
- Planetlab has highly uncontrollable elements
- Emulab gives exclusive access to nodes for short periods
- Planetlab gives shared access to nodes for long periods

More Differences

- Emulab gives total control of a node
- Planetlab gives limited control of a virtual node
- Emulab is a totally artificial environment
- Planetlab is a partially natural environment

So, What Do I Test Where?

- Anything requiring really controlled and reproducable testing should go on Emulab
- Anything that requires realistic Internet traffic/topology should go on Planetlab
- Most things involving security issues should go on Emulab
- Anything about observing long-term behaviors is better for Planetlab
- Anything requiring control of topologies should go on Emulab
- Anything to be opened to real users should go on Planetlab

Deter

- Some experiments are risky
  - In their potential to do unintentional harm
- Worm experiments are a classic example
  - Worms try to spread as far as possible
  - How sure are you that your testbed really constrains them?
  - Even one major Internet worm incident from an escaped experiment would be a disaster

Confining Risky Experiments

- That's the point of the Deter testbed
- Builds on functionality from Emulab
- But adds extra precautions to keep bad stuff from escaping the testbed
- Also includes set of tools speficially useful for these kinds of experiments

Why Do We Need More Isolation?

- DDoS experiments have been run on Emulab
  - With no known problems
- Why not just be careful?
- Question is, how careful?
- Especially if you're running real malicious code
  - Do you really understand it as well as you think?

What Is Deter For?

- Security testing, especially of risky code
  - Worms
  - DDoS attacks
  - Botnets
  - Attacks on routing protocols
- Other important element is network scale
  - Meant for problems of Internet scale
  - Or at least really big networks

Status of Deter

- Working testbed
- Similar model to Emulab
- Two clusters of nodes
  - At ISI and UC Berkeley
- Connected via high speed link
- Has over 300 nodes
- http://www.isi.deterlab.net
  - http://www.isi.edu/deter gets you to a lot of information about the testbed
- Funded by NSF and HSARPA

Security Issues for Deter

- Containment
  - Of both code and bad side effects
- Intrusion prevention
  - Bad guys might want to mess with it
- Confidentiality
  - Results of sensitive experiments shouldn't be leaked
- Isolation
  - Both during experiments
  - And from effects of previous experiments

Administration of Deter

- Run jointly by ISI and Berkeley
- Not as open as Emulab or Planetlab
- Must submit a project proposal to use the testbed
- Administration reviews it and approves or disapproves
  - Only approved users get access

What's Been Done on Deter?

- Lots of worm testing
- DDoS defense (including DefCOM)
- Analysis of malware
- Intrusion prevention research
- Attack traceback tools
- Network security model validation

GENI

- A new testbed for networks
  - Not yet built
- Specifically to support highly innovative network research
- Using ideas of virtualization to allow easy sharing of testbed resources
- Funded primarily by NSF
- http://www.geni.net

The Idea Behind GENI

- Owes a lot to Planetlab
  - Ideas of overlay and shared infrastructure
- Collection of physical resources (links, routers, etc.) will make up the *GENI substrate*
- Software management framework will overlay experiments on substrate

Key Ideas of GENI Architecture

1. Substrate components will be programmable
2. Substrate components will be virtualizable
3. Seamless opt-in mechanisms to allow users to access services
4. Modular, to allow addition of new network components in future

The GENI Slice

- Similar idea to Planetlab slice
- A set of resources across the testbed devoted to single use
- Virtualized into its own network
- Unlike Planetlab slices, need better ability for slices to interact
  - Not entirely separate in all cases

Challenges in Building GENI

- Security and robustness
  - Especially in times of crisis
- Embracing unforeseen technologies
  - Networking, end system, applications
- Network management must be improved
- Need a design that conforms to economic rules and realities
  - Ultimately, it can only work if someone is eager to pay for it

Status of GENI

- First discussed in detail at workshop in 2005
- Current development guided by planning and working groups
  - Composed of well-known networking researchers
  - Larry Peterson (Princeton) is key figure
    - As he was for Planetlab
- Various GENI Design Documents (GDDs) have come out

Some Current Elements of GENI Plans

- Many link and node technologies will be incorporated
  - Including wireless and sensor networks
  - Support of mobility very important
  - Optical networking seen as huge opportunity
- Possible to connect arbitrary networks to the edges
- Goal is to get actual useful stuff running over GENI
  - To attract users and validate ideas
- Heavily instrumented
  - To allow better testing
  - But also because we've learned from the first Internet

Some Other Testbeds

- Multi-antenna wireless testbed at UCLA
- Chiba City – scalable cluster computing testbed at Argonne Nat'l Labs
- City Sense – wireless sensor network testbed being set up by Harvard
- Open Network Laboratory – for educational purposes related to networks, at Wisconsin
- Many others out there

Learning More About Testbeds

- Tridentcom is a relatively new conference devoted to testbeds
- Publishes papers about new testbeds
- And about technology that supports testbeds

Setting Up Your Own Testbed

- What if you want to set up your own testbed?
- Why would you do it?
- How would you go about it?
- What issues should you be aware of?

Why Build Your Own?

- Shared testbeds have limited resources
  - Particularly close to due dates of major conferences
- If secrecy is important . . .
  - Particularly the case for companies
- Complete control and full customizability

How To Build Your Own Testbed

- Depends on exactly what kind of testbed you want
- Wireless mobile testbeds are a lot different than Internet protocol testbeds
- Assume a simple case:
  - Testbed to support typical OS/networks/distributed systems research

First Steps

- Space and money
  - Where will you put it?
  - How much can you afford to spend?
- Will probably be around for a long time
- If it's not tiny, it will take up significant space
- Options dwindle as dollars available become less

Space Issues

- Rack-mounted or desktop machines
- You often get more for your money with desktop machines
- But rack-mounted machines are much more compact
- Remember to consider issues of heat dissipation
  - If you are talking about more than a few machines
- Probably need a space where you can somewhat control access

Hardware Issues

- Generally want homogeneous hardware
  - Buy a bunch of identical machines
  - Eases administration and testing
- Probably important to consider size and power
  - Especially if you're buying a lot of them
- If you have sufficient expertise, might consider buying components
  - And assembling them yourself
  - Fewer dollars spent on hardware
  - But does the people time cost use up all those dollars?

What Goes in the Boxes?

- Usually want machines that are light on peripherals
  - Don't buy bunches of monitors and keyboards
  - Buy one or two of each, and a switch
  - Consider tradeoff between machine power and cost
  - Testbeds are built to last, so buy as close to top-of-the-line in performance as possible
- Consider devoting some hardware to mass data storage
- Consider issues of backups
  - If you really need a testbed, you'll generate a lot of data

Networks for Testbeds

- Generally want it to be fairly isolated
- But might be useful to allow remote access
  - At least from within your facility
- Again, build for the future
  - Go with best bandwidth possible
- Switched solutions for testbed are generally best
  - Consider issues of degree of control of network your research requires
- Do you need real routers?

Wireless Testbeds

- Considerably harder
- If you want any control, need a clean environment
  - Not a lot of other wireless networks around
  - If not clean, you'll learn about interference
  - But not about full range of possible network conditions
  - Clean environments generally mean isolated areas
  - Hard to find in CS departments or typical modern companies

Software Issues

- Highly variable
- At least need a bootable system to start with
- What else you need depends on what you're doing
- Might be sensible to set up multiple partitions on disks
- Important to make it easy for experimenters to install new SW on testbed

Running Testbeds

- If you needed one, you've got a lot going on
- Think about issues of testbed sharing and scheduling
- Small group testbeds maybe need only a signup sheet
- Larger systems with more users might need scheduling and reservation software
- If it's informal, users need to be careful not to stomp on other people's experiments

Testbeds and Aging

- Hardware gets old quickly
  - Testbed machines will start to die
  - And what you bought will no longer be available
- Might need to rejuvenate your testbed
  - Probably best to consider that at design time
- Generally a good idea to replace machines in bulk
  - Rather than piecemeal

Testbed Administration

- If it's complex, you need someone in charge
- Also need someone to deal with all the little hardware/software problems
- An ongoing cost of running a testbed
- Consider if you really need/can afford to pay that cost

Summary

- Setting up a testbed is a fair amount of work
- It's a big expense now and an ongoing expense for its lifetime
- Will Emulab/Planetlab/other public testbeds be good enough?
- Be sure they won't before you decide to build your own