Network Protocol and Kernel Development in Virtual Environments

George V. Neville-Neil

gnn@freebsd.org

Disclaimer

- I do not now, nor have I ever worked for any of these companies
- I have not been paid or otherwise compensated to make these statements
- All the software and hardware was paid for by me

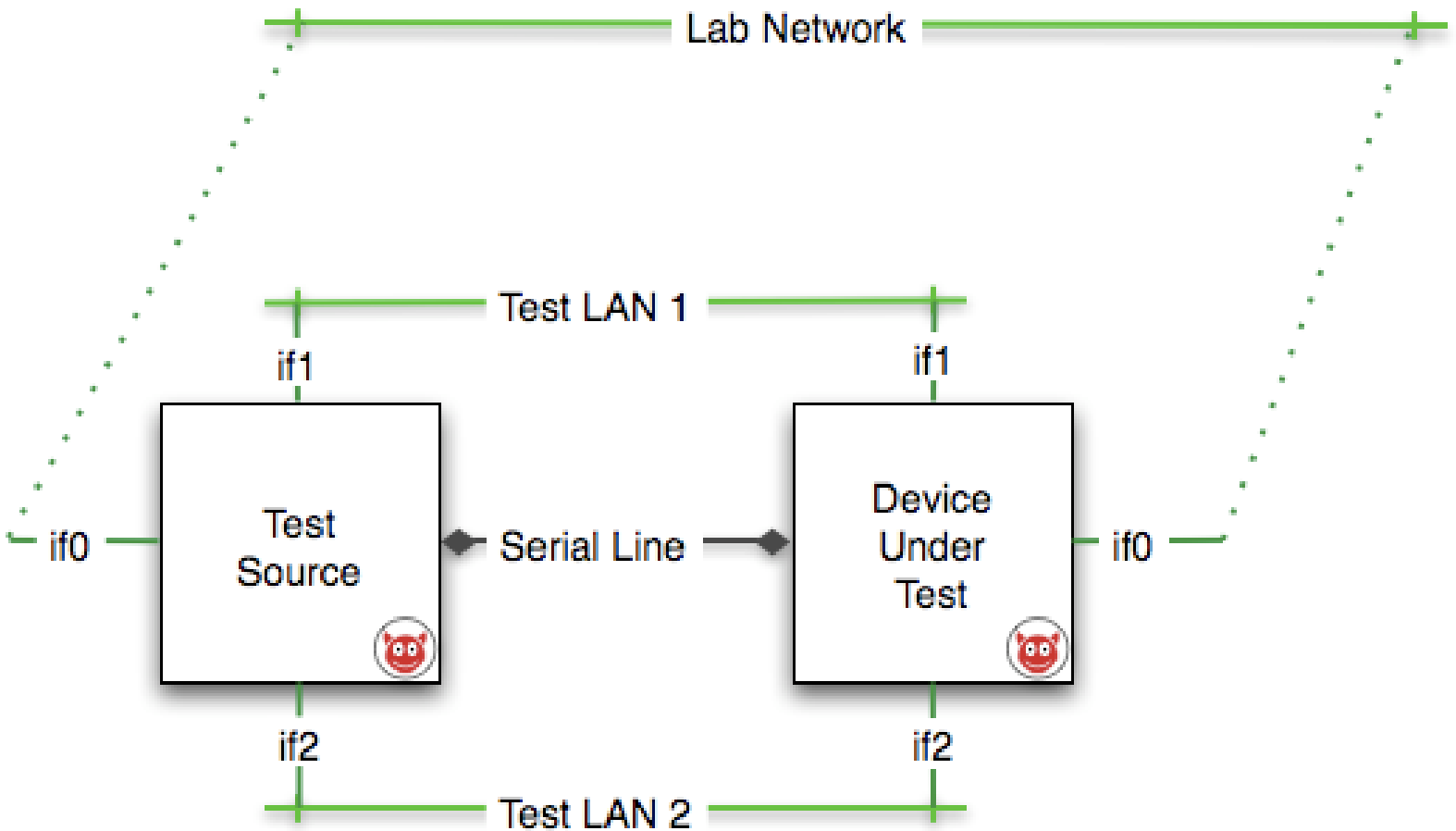
Motivation

- Kernel and Protocol development requires at least two machines
  - Client and server
  - Build machine and target
- Loopback testing is useful only in early stages
- Real protocol testing requires more than two machines
  - Two hosts and a router
  - Somewhat arbitrary networks
- Simulators do not fully exercise all the possible kernel interactions
  - Rarely show heisenbugs

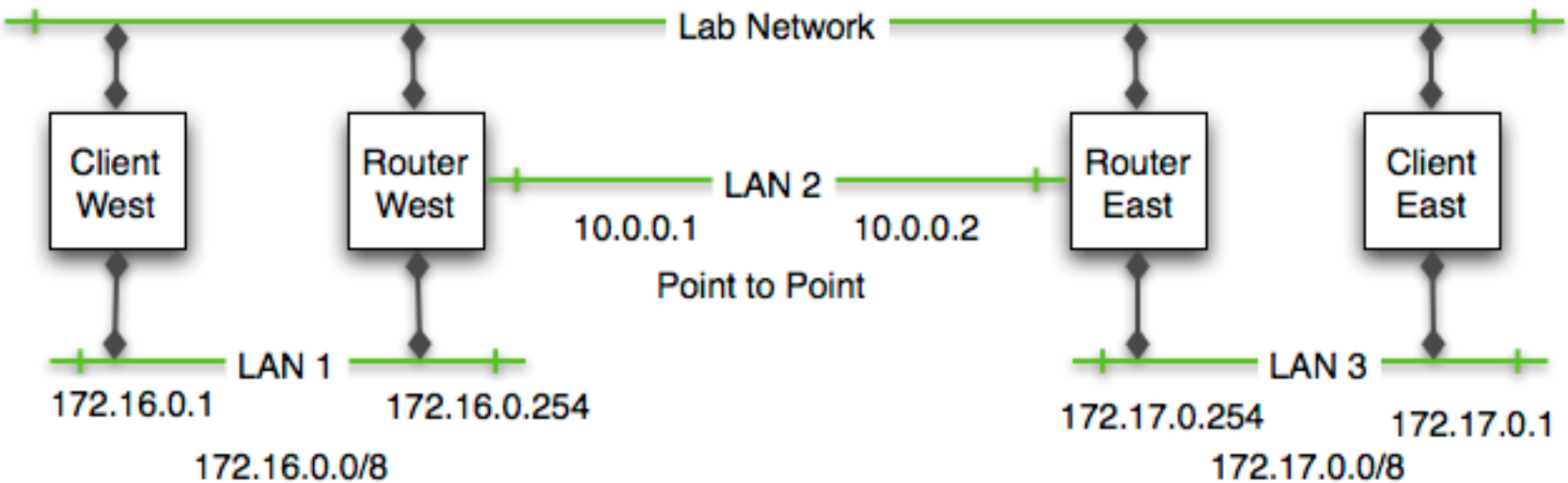
Motivation Con't

- Real machines take up space
  - generate heat
  - use power
  - are difficult to freely interconnect
  - make a mess of your workspace

A Simple Example



Complex Example



Lab Equipment (Physical)

- 2..N Machines
- KVM Switch or Terminal Server
- Hubs
- Network Cabling
- Electrical Cabling

Lab Equipment (Virtual)

- One very fast computer
- Lots of memory
- Lots of disk
- One power cable
- One network cable
- VNC

Some Terminology

- Host Machine
  - The real metal on which the virtualization software runs
- Host Operating System
- Guest
  - A virtual machine running within the virtualization software
- Guest OS
- Clone
  - Guest that is a copy of another guest
- Team
  - A collection of guests

Setting up your laboratory

- Machines
  - What kind of machines will you run?
  - Memory Sizing
  - Disk Sizing
  - What to install
- Connections
  - Networks
  - Console
  - Serial Lines

Machines in my lab

- Nightly
  - A machine on which nightly builds are done
  - NFS Server
  - Several source trees
  - cvsup and p4
  - Editors
  - Test tools
- Test Targets
  - Cloned machines that can be thrown away and trivially rebuilt
  - NFS Client
  - New kernels and code are installed from NFS mounted file systems
  - Minimal tools required
  - No need for source trees

Memory Sizing

- Console only FreeBSD runs fine in 128M of RAM
  - Building the world
  - Building kernels
  - Using as a router or network device
- With X-Windows probably 256M of RAM
- If you're running a more memory intensive workload then you may need more memory

What I install

- The Nightly box will be the root of your cloning tree
- Full sources, binaries, and doc but no games
- sudo
- cvsup, p4 (optional)
- Emacs
- Ethereal
- Python
- TAHI Test Tools (IPv6/IPSec)
- Your tool set will vary

Virtual Disks

- Two disks on the main machine
- Main Disk
  - /, swap and /usr for the regular install
  - Currently 6G but could be 4G
- CVS Disk
  - /cvs for source trees
  - 8G
- Target machines only get the main disk

Virtual Disk Types

- Use sparse disks if you can
- Pre-allocate your disks only if you need the speed
- SCSI disks are faster than IDE, even if they're virtual
- If you're going to back up your machines to CD or DVD then go with split disks
  - These are 2G files which make up a disk
- Disk space isn't really infinite
  - really
    - trust me

What does Nightly look like now?

```
nightly ? df -h
Filesystem     Size    Used   Avail Capacity  Mounted
on
/dev/da0s1a    989M    192M    719M    21%    /
/dev/da0s1d    4.4G    2.3G    1.7G    58%    /usr
/dev/da1s1d    7.7G    3.2G    4.0G    45%    /cvs

nightly ? du -sh *
1.8G    FreeBSD-CVS  /* Full CVS Mirror */
439M    FreeBSD.5    /* Checked out trees */
457M    FreeBSD.nightly
453M    FreeBSD.stable
 55M    FreeBSD.gnn  /* doc and www only */
```

Networks

- Nightly only has one network interface
- All targets have at least two but normally three interfaces
- Main (lnc0)
  - Connected to the lab network
  - Best to use DHCP
- Test Lan 1 (lnc1)
  - Connected to a network with little or no other traffic
  - Statically assigned address
- Test Lan 2 (lnc2)
  - In testing router configurations you need a second test network
  - Statically assigned address

Serial Lines

- Debugging a kernel panic with printf()s is tedious
- Virtual serial lines work just like the real thing
  - Only they don't require you to remember how to make a NULL modem cable
- I use two lines
  - Console
  - Debug

Other Devices

- CD/DVD ROM
  - Boot and install from an ISO file on disk
- USB
- Sound

Building Your Lab

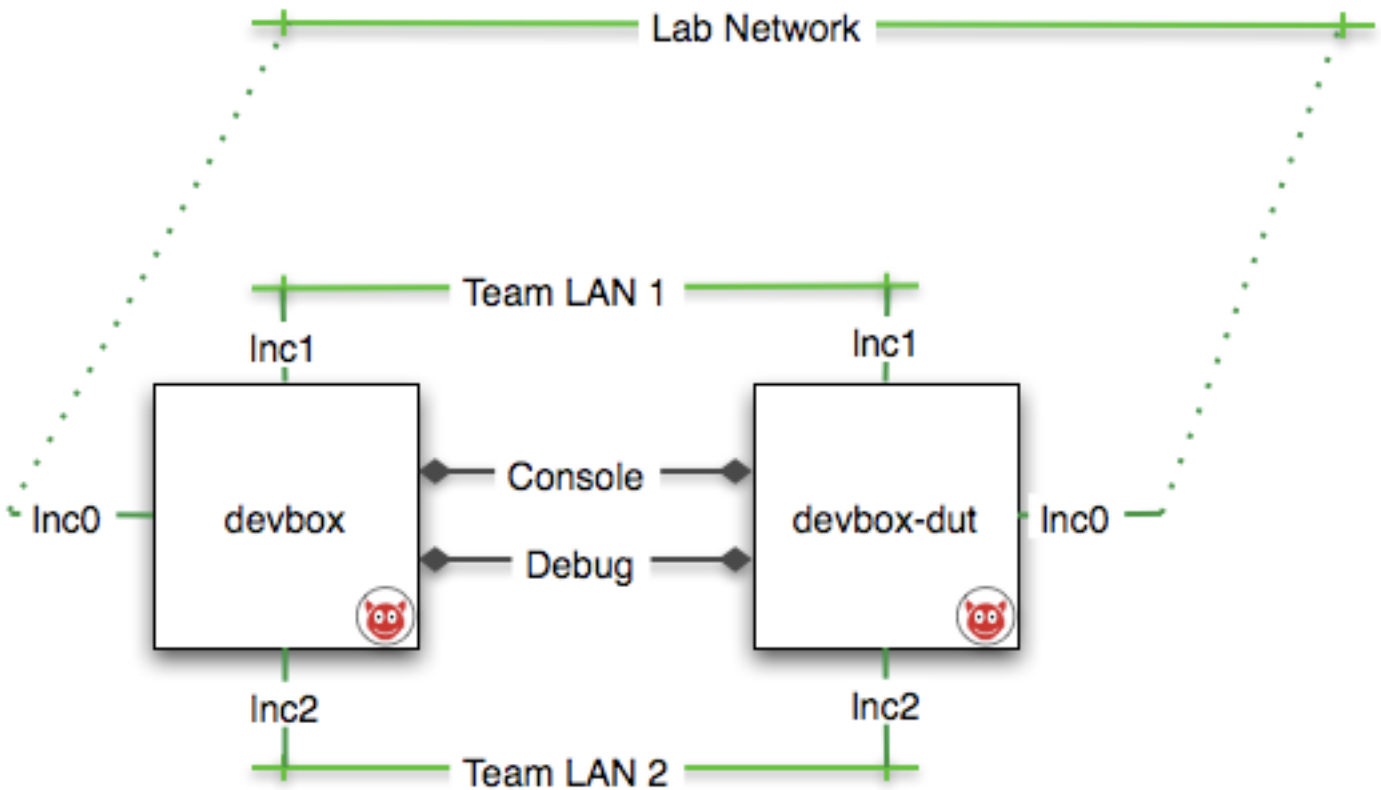
- Install Software
- Create a single guest to use as Nightly
- Install the requisite software into Nightly
- Get your nightly build scripts running
- Leave Nightly running if you have the CPU horsepower

Clones

- Take a guest and make a copy of it
- Linked clones only store new data when it is created
  - A version of copy on write
- Linked clones depend on the root machine not going away
  - Never ever delete Nightly if it's the root of your tree
- Full clones are independent copies
  - Take up more space than linked clones
  - Run faster than linked clones, but by how much I don't know

Teams

- A team is 2 or more guests treated as a group
- Teams can have their own, private, networks



Suspension

- Never turn off machines again
- Suspending a guest to disk releases memory and other resources but does not require a reboot
- Does not depend on the OS to be able to suspend
- Some things are not allowed with suspended machines
  - Updating configuration
  - Cloning

Development Process

- Clone Nightly as many times as necessary
  - Need to shut down Nightly first
- Make one machine the master
- Check out code and modify on the master
  - I usually call this devbox
- DO NOT INSTALL THE KERNEL ON THE MASTER
- Mount build directory on the device under test (DUT)
- Install the kernel on the DUT
- Test, crash, burn
- Use ddb, gdb, what have you

Development Process Con't

- Suspend and resume teams and guests as you need
- Eventually check in code
- Destroy guests you don't need
- Archive guests you consider important

VMWare

- VMWare Workstation
  - Officially runs on Linux and Windows
  - Teams
  - Team networks
  - Linked and independent clones
- Virtual Hardware
  - IDE and/or SCSI Hard Disks
  - IDE CD/DVD ROM
  - Lance Ethernet (lnc driver)
  - Serial and Parallel cables
  - Sound
  - USB
- VMWare Server Beta
  - Does not have teams or team networks

Parallels

- Intel Mac OS/X as well as Linux and Windows
- No teams
- No linked clones
- Currently in Beta (Beta 5)
- Very occasionally panics my machine
- Devices
  - IDE Hard Disks and CD/DVD Drives
  - Floppies
  - Reatek 8209 Network Interface (ed0)
  - Serial and Parallel Cables
  - Sound
  - USB

Open Source Alternatives

- Xen
  - Requires the guest OS to be "ported" to the platform
- Qemu
  - Doesn't support acceleration on the Mac yet
  - Supports several different architectures
    - ARM
    - PowerPC
    - SPARC
    - MIPS
  - Does not require kernel customization
  - No teams or cloning
  - Interesting possibility for the future

Hardware Choices

- How effective your virtual machines depends on how fast your host is
- Disks
  - Use the highest bandwidth and lowest seek time you can afford
- Memory
  - The more the better
  - 2 G is a good number for a network of 4 machines
- CPU
  - Clock rate is not the most important factor
  - On chip L1 and L2 Cache is the most important factor
  - If the chip support Virtualization that's even better

Chuo

- HP DL-360 G4 1U Rack Mount
- 4 Gigs of RAM
- 2x Xeon 3.4GHz
- 2x 73G U320 Fast/Wide SCSI Disks (RAID 0)
  - Single biggest boost to performance
- Host OS is Linux RH EL 4
- Easily runs 4 guests
- Builds a kernel in

Minion

- Intel Based Mac Book Pro
- Mac OS 10.4.6
- Dual Core 2.0 GHz CPU
- 2 G of RAM
  - You really really don't want to swap
- 120G SATA Hard Disk
- Parallels Beta 5
- Builds a kernel in about 15 minutes

Other uses and benefits

- Never install Windows on a real machine again
- Virus and security problem testing
- Keep old versions of systems just as they are
- Share systems with other people
  - Keep them on a server
  - Move them on a fast network
  - Mail a DVD with a machine on it

A Quick Demo

Questions?