

Network Security

Is It Needed and Who Can You Trust?

Jim Griffioen

Laboratory for Advanced Networking

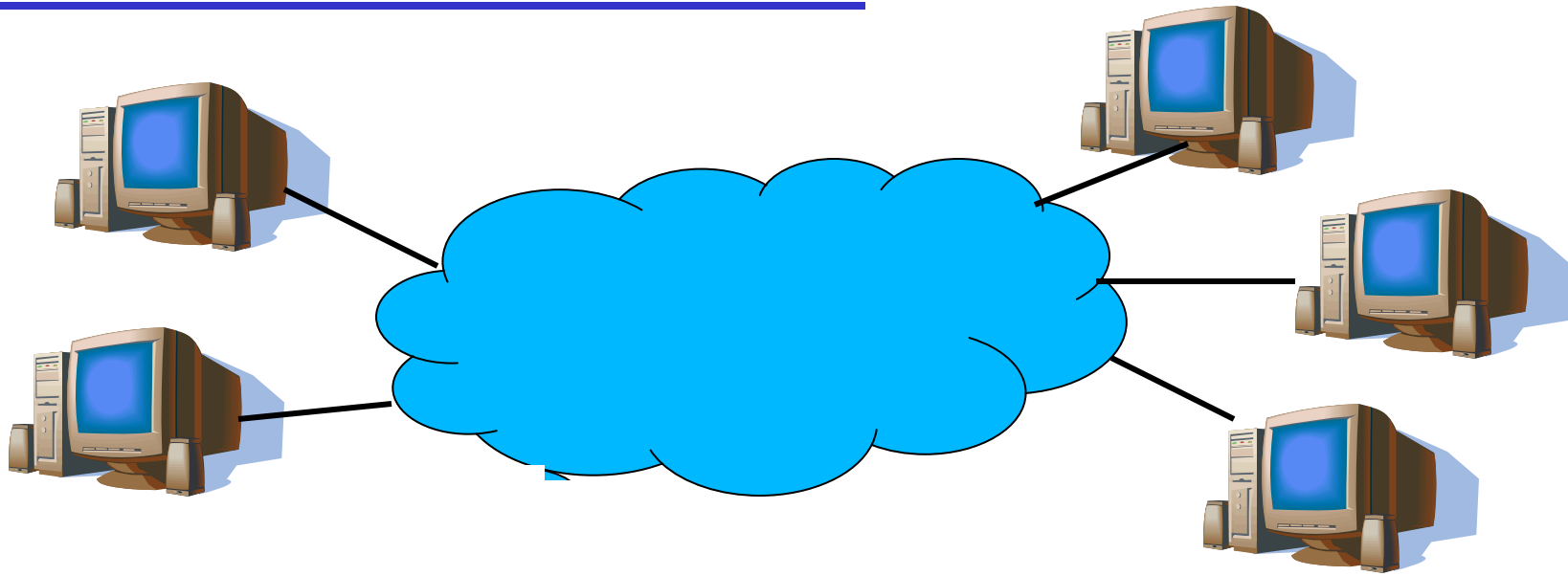
University of Kentucky

Introduction

- I'm not a security person - but I am learning
- Unlike previous talks, this talk will raise more questions than it answers.
- Goal is to look at securing networked systems, and to highlight recent advances that are making the problem harder
- Much of the inspiration and information for the talk come from "Firewalls and Internet Security: Repelling the Wily Hacker", by W. Cheswick and S. Bellovin.

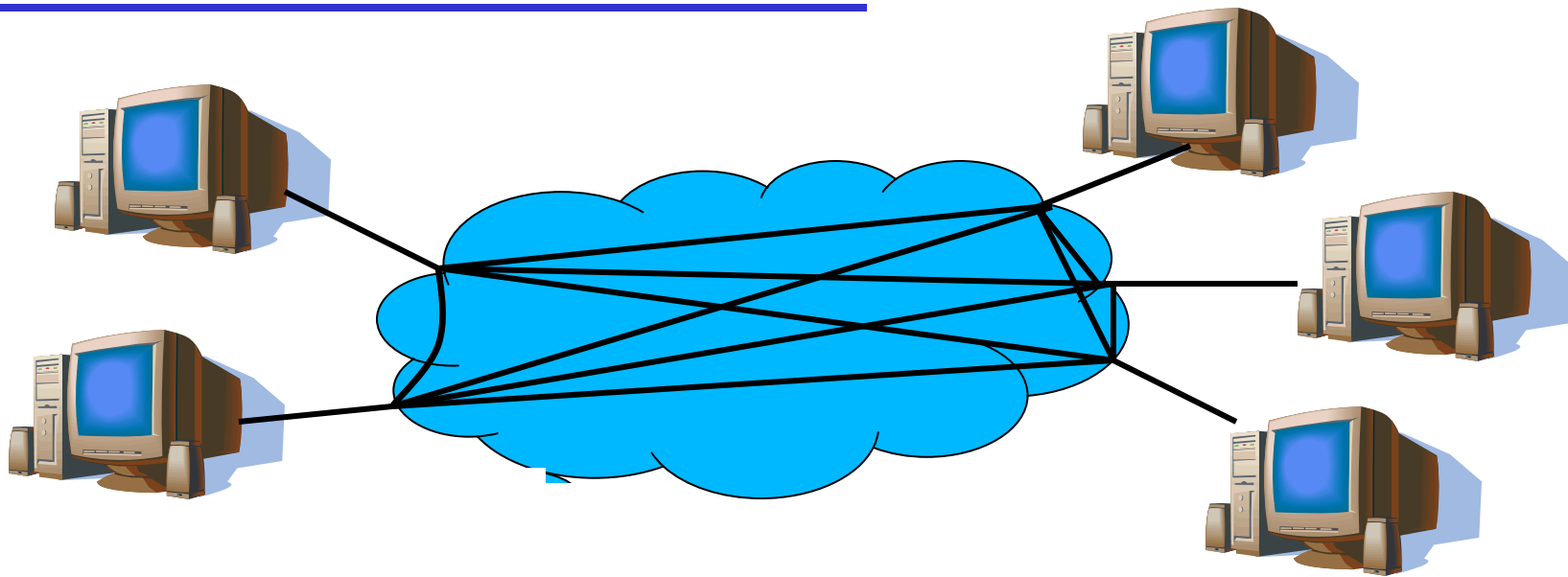


Conventional Wisdom



- Computers at the edge of the network house the **resources** and **information** that needs to be protected.
- The **Network** only provides connectivity. Because it is just a delivery mechanism, it does not need to be protected.

Conventional Wisdom



- Computers at the edge of the network house the **resources** and **information** that needs to be protected.
- The **Network** only provides connectivity. Because it is just a delivery mechanism, it does not need to be protected.

What's there to Protect?

○ Resources

- Processor
- Storage
- Peripherals
- Bandwidth

○ Information

- User data
- System (configuration) data
- User (or machine) identity



Protect from Whom?

- Intruders
- Other (local) users
- One's self
 - Accidental modification/removal of information
 - Accidental information leakage
 - Accidental infection
 - Unintended resource usage

Note: **One's self** may be the biggest risk



Keep Away From Each Other!

- Keep intruders out
 - Minimize and secure entry points into the system
- Keep local users in
 - Limit the ways users can get out
- Keep local users from hurting one another
 - Enforce barriers between users
- Keep a local user from hurting him/her-self
 - This is a difficult problem.
 - One approach is to ask the user multiple times before taking action.



Protecting the End Systems

○ Need to protect:

- The **Operating System**
- The **Applications** (normal apps or deamons).
- The **Users** from themselves.
 - ◆ Many protection programs that try to stop users from doing potentially harmful things.



Protecting the OS

- Need to write secure OS code
- Should not be able to obtain control of the machine via user-level application or external network attack.
- Internal attacks abound - have access to resources and local information to assist in the attack.
- Most common (external) attack is to attack entry points such as user accounts/passwords.
 - ❑ Picking good passwords is hard and getting harder
 - ❑ Passwords should change regularly
 - ❑ Single factor is not good enough
 - ❑ Challenge/response is even better



Protecting Applications

- Need to write secure applications/daemons
- Common problems include:
 - Buffer overflow attacks
 - Vulnerable authentication protocols used in
 - ◆ System daemons - e.g., TFTP, FTP, rlogin, rsh, HTTP(.htaccess), NFS, SMTP, SNMP, etc
 - ◆ Application daemons - e.g., collaborative tools, distributed gaming, VoIP, file sharing, etc.
 - Apps that open up outgoing connections to malicious servers.

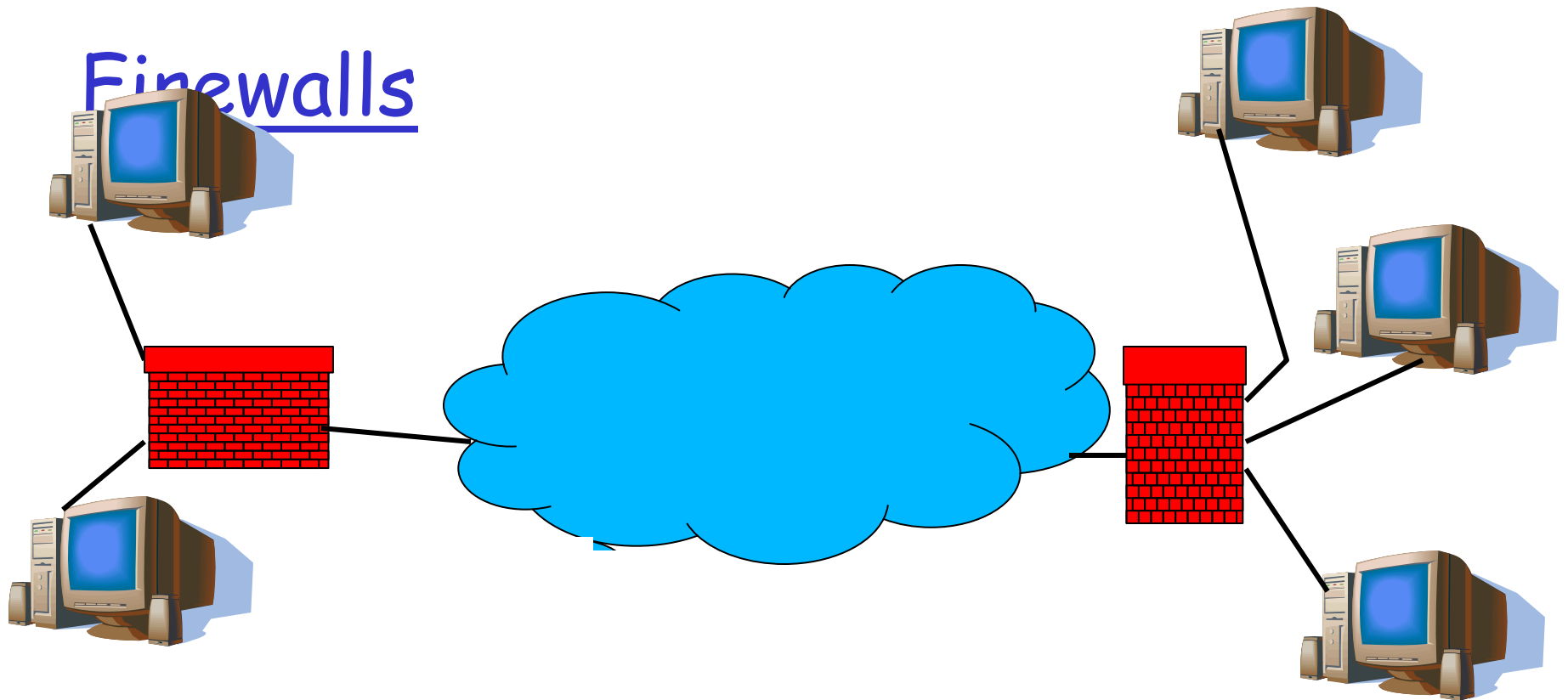


Securing the OS and Applications is Hopeless!

- Well maybe not hopeless, but it sure is not easy.
- Constant battle between user-friendly features and locked-down tight security
- The less features and entry points there are, the easier it is to secure.
- But even then, all programs will have bugs.
- Need to minimize the number of potential security holes.



Firewalls



- Firewalls limit the number of entry points and the potential for bugs.
- Firewalls do not run user programs, and have no users. They are often built as special-purpose hardware devices.

Firewalls

- Limit what can get in and what can get out.
- Successful because of the Internet's current client/server (request/response) architecture. Clients always initiate communication => firewalls don't let anything in unless it was requested.
- But, still must allow outbound connections - which can be very dangerous!



Two-way Firewalls

- Don't let anything in
- Don't let much out
 - Only let web requests out
 - Only let outgoing mail connections
 - Maybe a few other things (DNS, time server, etc) from a few select machines.
- Unfortunately, limiting communication just frustrates users
- *The Result:* Port 80 has become the new network layer service.
- *Observation 1:* Phishing encourages outgoing connections that can still go out and bring in viruses and other malware (via web and mail)
- *Observation 2:* Firewalls operate too low in the protocol stack and thus filter based on protocol type and port number. To be more effective at determining what is really going over a port, some firewalls now do *deep packet inspection*. It is an escalating arms race between users and network providers/administrators.



Defending end systems
against attacks is
really hard!



Sometimes the best defense
is a good offense.

Sometimes the best defense
is a good offense.

Thought: What if we secure the
network leading to the attacker?

Sometimes the best defense
is a good offense.

*Thought: What if we secure the
network leading to the attacker?*

But what (in the network) needs to
be secured?

Trusting Network Addresses

- IP Addresses can be easily spoofed
 - you can't trust the IP source address
- How do you prevent IP address spoofing?
 - You don't. Instead rely on PKI certificates.
- Other proposed solutions:
 - **Ingress Filtering**: Did the packet arrive on the interface the router expected it to arrive on?
 - **Packet traceback**: Unreasonable for address verification



Related Problem: Denial of Service

- Flaws in IP
 - Can't trust the source address
 - Routers do not record packets they forward
 - No cost to the sender to transmit a packet
- Result: (Distributed) Denial of Service Attacks
- DDoS attacks are almost impossible to stop.
- Packet Traceback Approaches:
 - Packet logging
 - Packet digests
- Unfortunately, ISPs aren't necessarily motivated to stop DDoS attacks!



Trusting Network Names

- Domain Name System is an integral part of the Internet.
- DNS attacks include:
 - Denial of Service attacks
 - Spoofing DNS servers
 - Compromised DNS servers
 - Altering DNS responses
- Proposed solution: DNSSEC



Trusting Auto-configuration

- The Dynamic Host Configuration Protocol (DHCP) has become fundamental to network configuration.
- Because requests are broadcasts, it is relatively easy to masquerade as the DHCP server. The bogus DHCP server can then give out incorrect information.
- Common problem is accidental masquerading (starting two DHCP servers)
- Partial solutions: hardcoded client MAC addresses, authentication protocols, IPSEC, DNSSEC

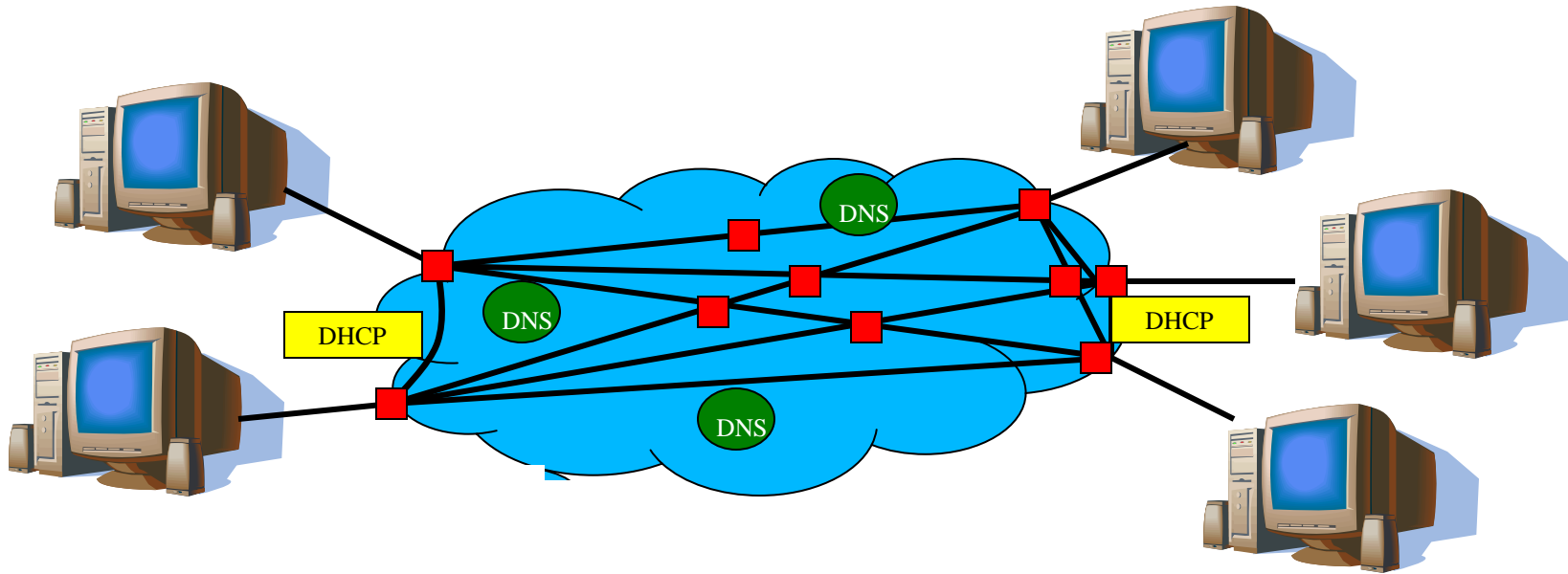


Trusting Routes

- Each router makes independent routing decisions based on routing information exchanged via routing protocols
- Implicit trust exists between routers in the current Internet.
- The routing protocols themselves can be compromised - e.g., bogus/incorrect BGP announcements can cause traffic to be hijacked or diverted/redirected
- Solutions: Secure BGP



Securing the Network is Hard!



- In summary, the network is not just a set of wires. There is a lot of stuff to protect/secure:
 - Addressing
 - Routing
 - Naming
 - Configuration
 - etc

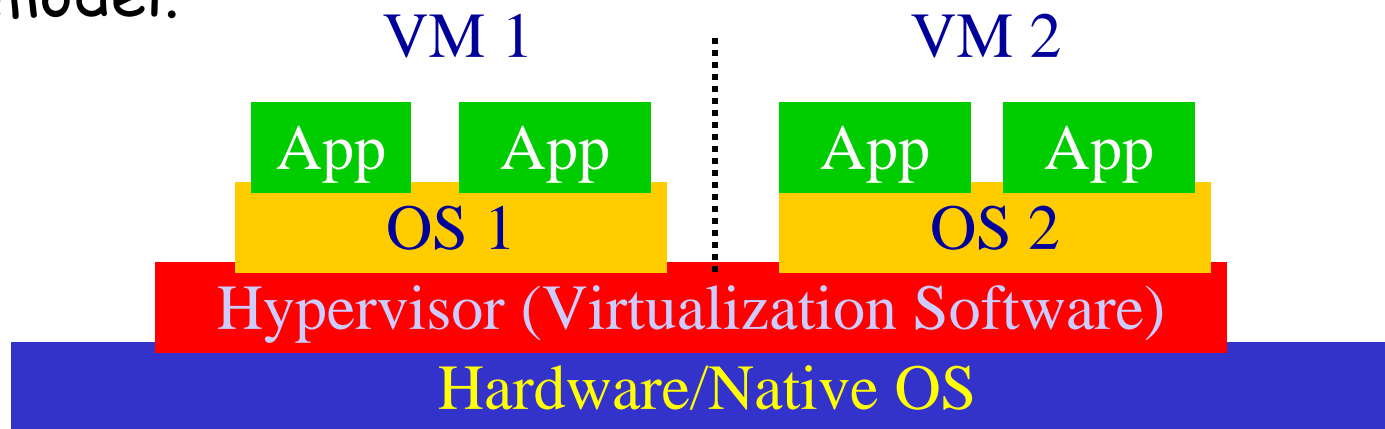
But wait, it can get worse!

- We are developing new “features” that can make the situation even worse.
- Although some thought has gone into thinking about security before developing these new enhancements, we have not yet integrated security as a first class requirement.



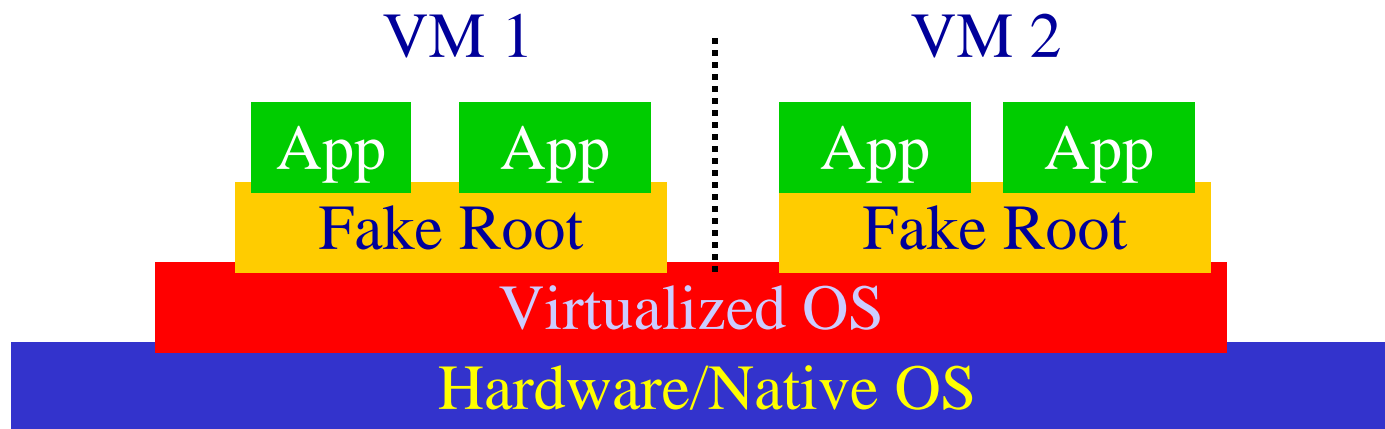
Computer Virtualization

- Virtualization has been around for a long time.
- Run many operating systems on the same machine at the same time - i.e., multiple virtual computers
- Datacenters everywhere are rushing to adopt this model.



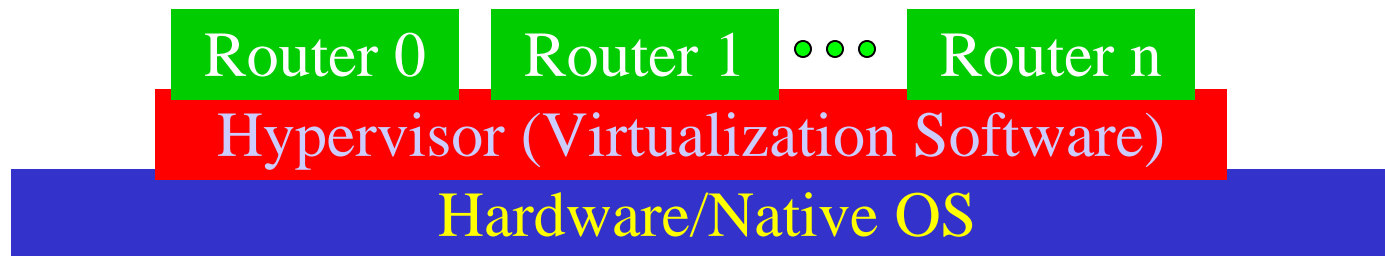
Computer Virtualization (2)

- Lightweight forms of virtualization are also becoming very popular - e.g., Jails in BSD



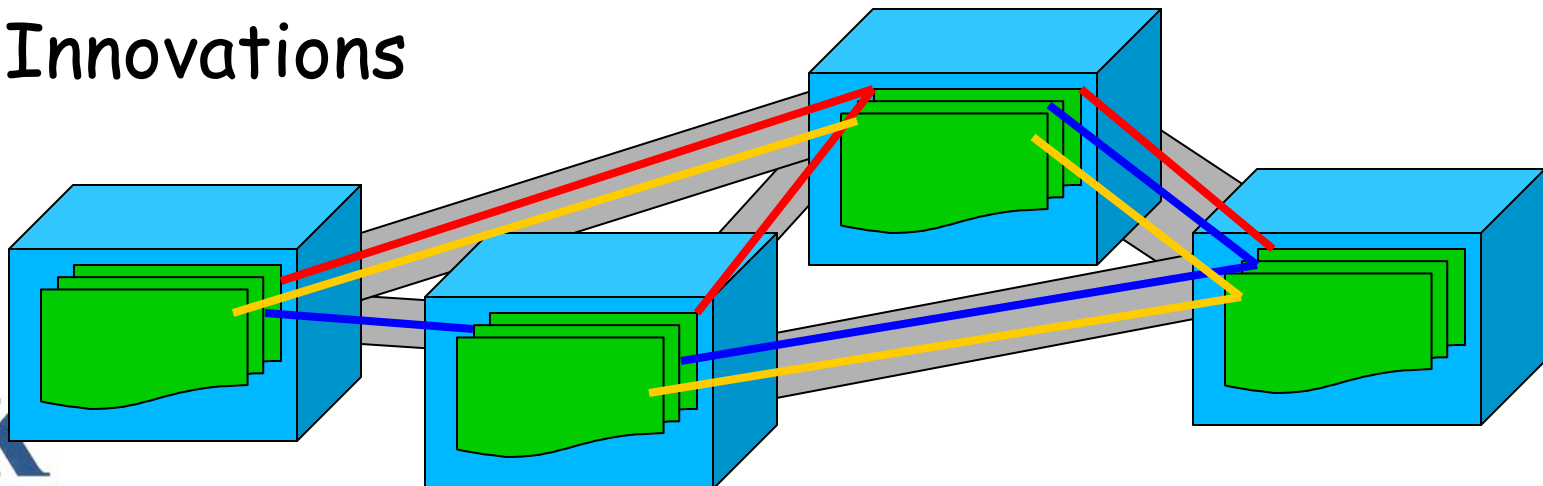
Network Virtualization

- Extending the virtualization concept to network routers.
- Router virtualization enables multiple “virtual routers” to use the same hardware.
- Can program each router differently.



Network Virtualization

- Single infrastructure supports multiple networks simultaneously.
- Each network is a "slice" of the each of the routers.
- Run net-specific code in each "sliver".
- (GENI) Global Environment for Network Innovations



Lots of new questions

- Who gets to start/stop/load VMs?
- How do you fire up a lot of VMs across many nodes all at once (i.e., to create a slice)?
- Who determines the sharing policies between VMs?
- What dangers arise if anyone can load anything they want into their VM?
- How are slices "connected".
- Is there any shared infrastructure between slices?
- Is every slice built independently from the ground up?
- Should slices support user-level programmability?
- Etc ...



Questions?

Comments?

