COS 597E: Software Defined Networking

Jennifer Rexford
Princeton University

MW 11:00am-12:20pm

The State of Networking

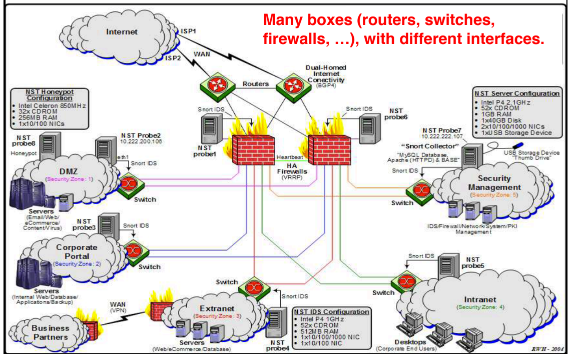
The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Programmable hosts: arbitrary applications
- Enables innovation
  - Apps: Web, P2P, VoIP, social networks, …
  - Links: Ethernet, fiber optics, WiFi, cellular, …

Inside the 'Net: A Different Story…

- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features

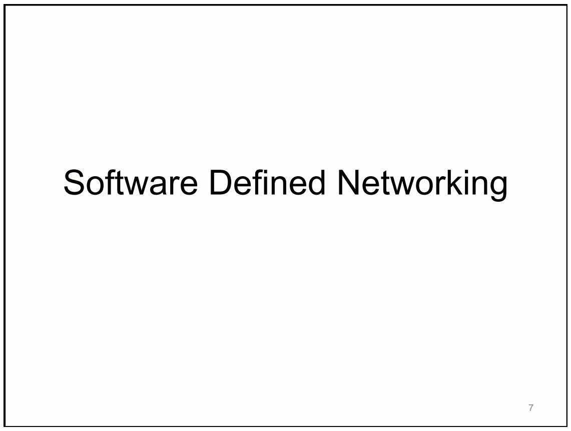
Do We Need Innovation Inside?



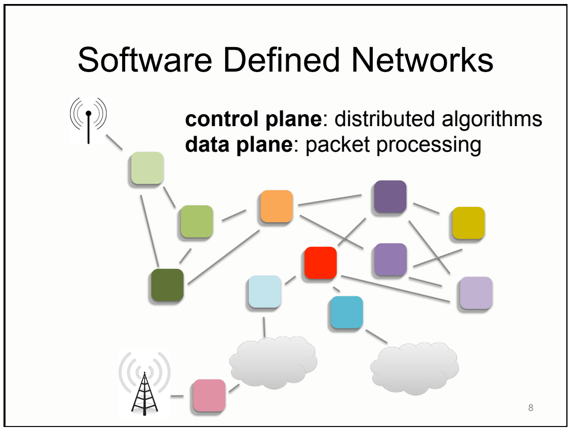
Many boxes (routers, switches, firewalls, …), with different interfaces.

Do We Need Intellectual Progress?

- Lots of domain details
  - Plethora of protocols
  - Heaps of header formats
  - Big bunch of boxes
  - Tons of tools
- Teaching networking
  - Practitioners: certification courses, on the job
  - Undergraduates: how the Internet works

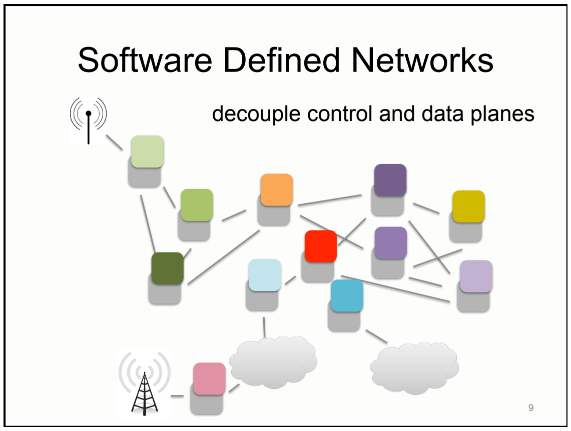


Software Defined Networking



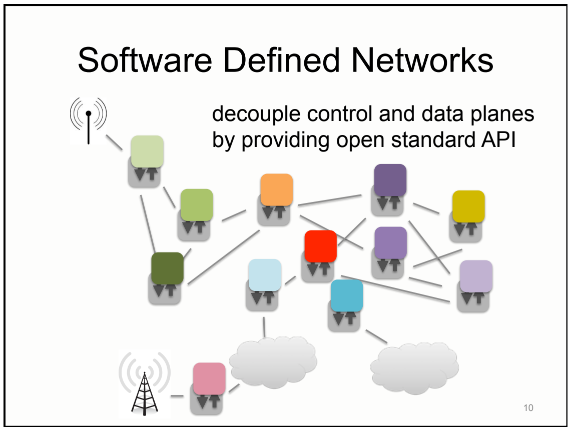
Software Defined Networks

control plane: distributed algorithms
data plane: packet processing



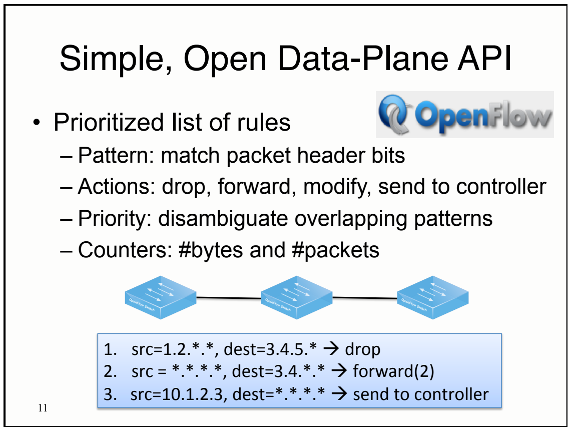
Software Defined Networks

decouple control and data planes



Software Defined Networks

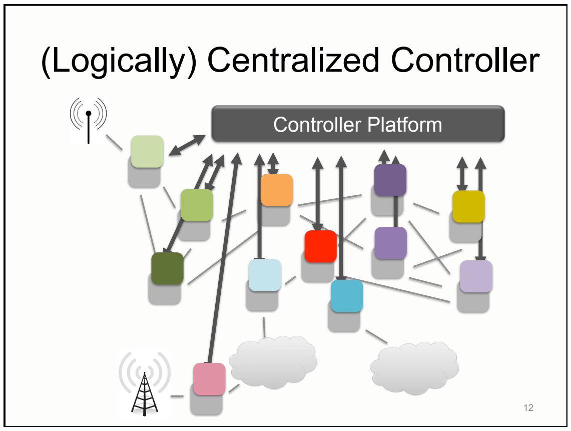
decouple control and data planes
by providing open standard API



Simple, Open Data-Plane API

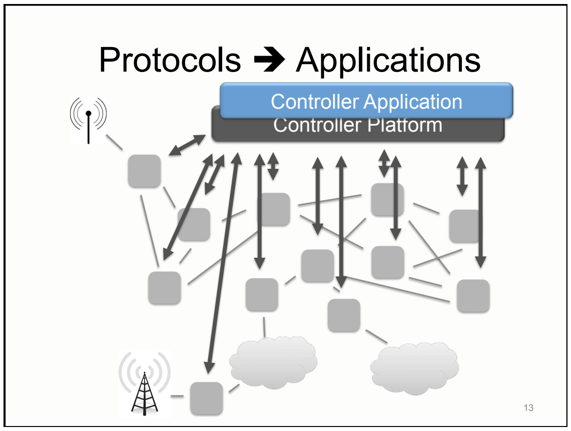
- Prioritized list of rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller



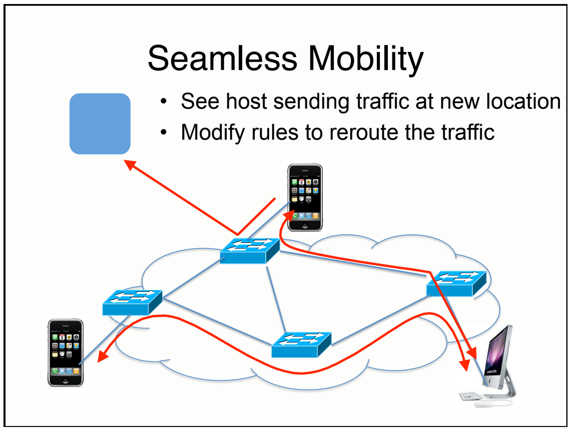
(Logically) Centralized Controller

Controller Platform



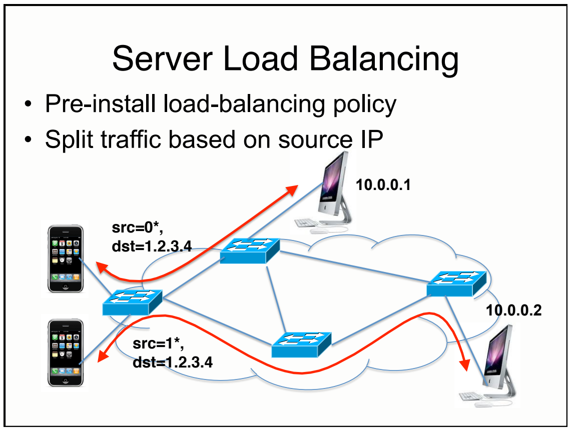
Protocols ➔ Applications

13



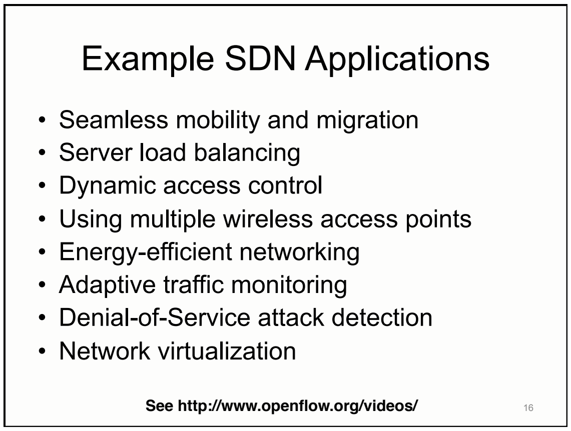
Seamless Mobility

- See host sending traffic at new location
- Modify rules to reroute the traffic



Server Load Balancing

- Pre-install load-balancing policy
- Split traffic based on source IP

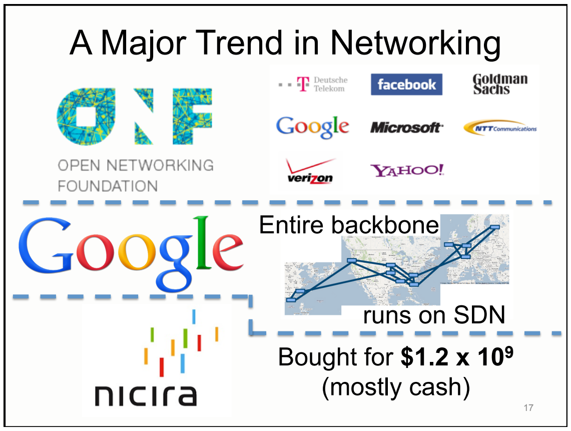


Example SDN Applications

- Seamless mobility and migration
- Server load balancing
- Dynamic access control
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection
- Network virtualization

See http://www.openflow.org/videos/

16



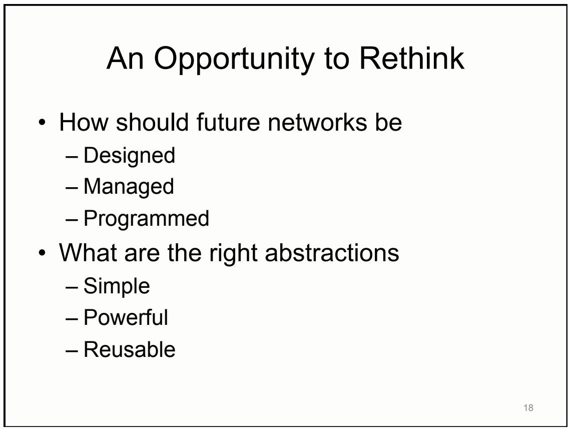
A Major Trend in Networking

ONF
OPEN NETWORKING
FOUNDATION
Deutsche Telekom
facebook
Goldman Sachs
Google
Microsoft
NTT Communications
verizon
YAHOO!
Google
nicira

Entire backbone runs on SDN

Bought for **$1.2 x 10^9** (mostly cash)

17



An Opportunity to Rethink

- How should future networks be
  - Designed
  - Managed
  - Programmed
- What are the right abstractions
  - Simple
  - Powerful
  - Reusable

18

Structure of the Course

Syllabus

- Introduction (3)
- SDN abstractions (6)
- SDN applications (4)
- SDN systems challenges (4)
- Enhancing the data plane (5)
- Course wrap-up (2)

Paper Reading

- Read ~2 papers for each class
  - Recent research papers on SDN
  - Basis for discussions in class
- Write reviews (1 page each)
  - Summary (problem, solution)
  - What you like
  - What could use improvement
  - What you would do next
- Upload reviews to CS Dropbox before class
- See “How to Read” on today’s syllabus

Lightweight Assignments

- Programming assignments
  - MiniNet platform (due 5pm Mon Sep 16)
  - Ryu controller (due 5pm Tue Oct 1)
  - Pyretic language (due 5pm Fri Oct 11)
- Assignments are *not* graded
- Collaboration policy
  - Can freely collaborate with others
  - Each person should understand all material
- Will help with your course project

Course Project

- Final research project
  - Work alone or in teams of 2-3
  - Your own topic, or from a list we suggest
- Schedule
  - Talk to me (and others) about project ideas
  - 5pm Mon Oct 21: short proposal due
  - 5pm Tue Jan 14: written report due
  - Later that week: short oral presentation

Grading

- 0% programming assignments
- 30% class participation
- 30% paper reviews
- 40% course project (paper, talk)

To Do

- Next steps
  - Join the Piazza site: https://piazza.com/princeton/fall2013/cos597e/home
  - Complete assignment 1 (due Tuesday)
  - Read and review 4D and Ethane papers
- Brush up on basic Python programming
  - http://docs.python.org/2/tutorial/
  - http://www.greenteapress.com/thinkpython/html/index.html

25

Review of "How the Internet Works"

26

Why Review?

- SDN interacts with "legacy" networks
  - Unmodified end-host computers
  - Hybrid deployments of SDN
  - Connecting to non-SDN domains
- SDN is a reaction to legacy networks
  - Challenges of managing and changing them
- General lessons on abstractions
  - Practice talking about abstractions
  - Some abstractions should be retained

27

Main Ideas

- Best-effort packet delivery
- Protocol layering for modularity
  - Internet hourglass design
- Relationships between layers
  - Naming and addressing
  - Directories and routing
- Scalability
  - Through hierarchy and indirection

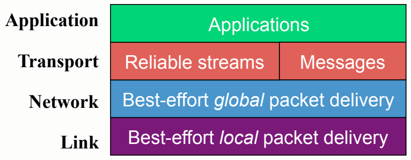
28

Best-Effort Packet Switching

- Packet switching
  - Divide data into packets
  - Packets travel separately
  - Enables statistical multiplexing
- Best-effort delivery
  - Packets may be lost, delayed, out-of-order
  - Simplify network design and failure handling
  - Build timely, ordered, reliability delivery on top

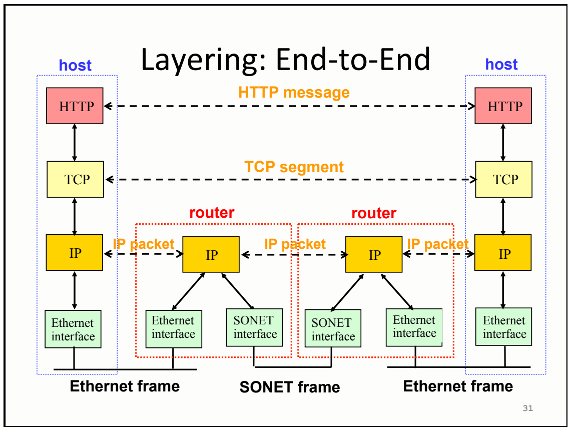
29

Layering: Internet Protocol Stack



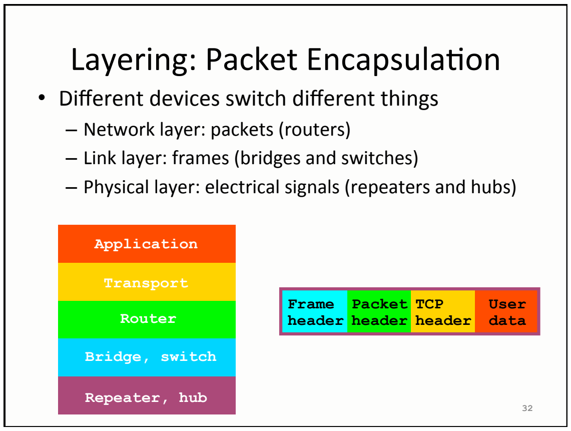
Application	Applications	
Transport	Reliable streams	Messages
Network	Best-effort *global* packet delivery	
Link	Best-effort *local* packet delivery	

30



Layering: End-to-End

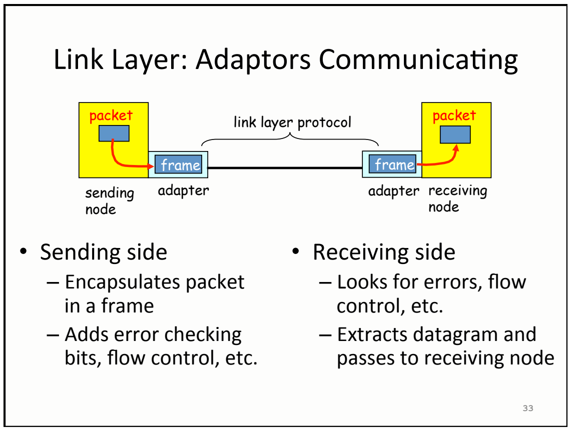
31



Layering: Packet Encapsulation

- Different devices switch different things
  - Network layer: packets (routers)
  - Link layer: frames (bridges and switches)
  - Physical layer: electrical signals (repeaters and hubs)

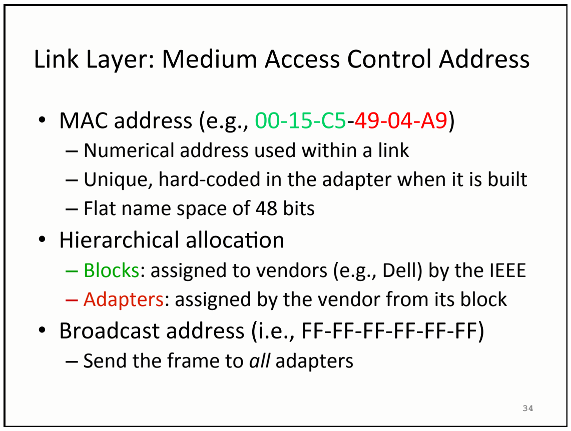
32



Link Layer: Adaptors Communicating

- Sending side
  - Encapsulates packet in a frame
  - Adds error checking bits, flow control, etc.
- Receiving side
  - Looks for errors, flow control, etc.
  - Extracts datagram and passes to receiving node

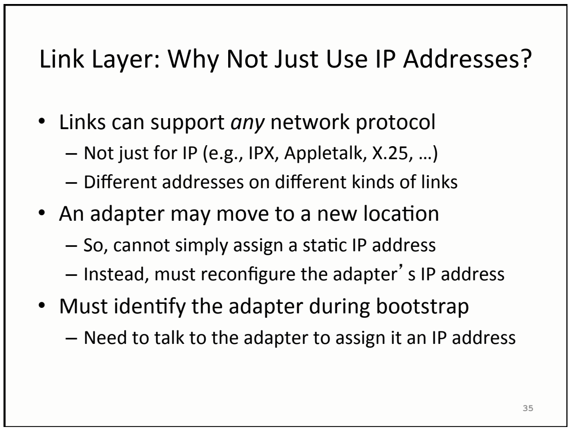
33



Link Layer: Medium Access Control Address

- MAC address (e.g., 00-15-C5-49-04-A9)
  - Numerical address used within a link
  - Unique, hard-coded in the adapter when it is built
  - Flat name space of 48 bits
- Hierarchical allocation
  - Blocks: assigned to vendors (e.g., Dell) by the IEEE
  - Adapters: assigned by the vendor from its block
- Broadcast address (i.e., FF-FF-FF-FF-FF-FF)
  - Send the frame to *all* adapters

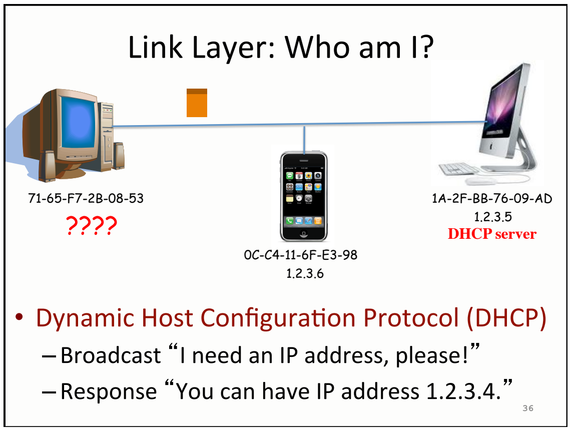
34



Link Layer: Why Not Just Use IP Addresses?

- Links can support *any* network protocol
  - Not just for IP (e.g., IPX, Appletalk, X.25, …)
  - Different addresses on different kinds of links
- An adapter may move to a new location
  - So, cannot simply assign a static IP address
  - Instead, must reconfigure the adapter’ s IP address
- Must identify the adapter during bootstrap
  - Need to talk to the adapter to assign it an IP address

35

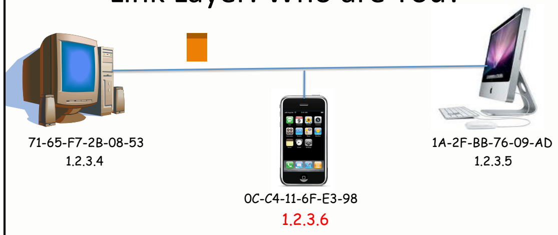


Link Layer: Who am I?

- Dynamic Host Configuration Protocol (DHCP)
  - Broadcast “I need an IP address, please!”
  - Response “You can have IP address 1.2.3.4.”

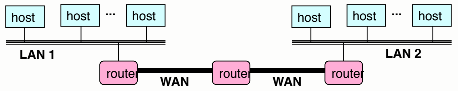
36

Link Layer: Who are You?



- Address Resolution Protocol (ARP)
  - Broadcast "who has IP address 1.2.3.6?"
  - Response "0C-C4-11-6F-E3-98 has 1.2.3.6!"

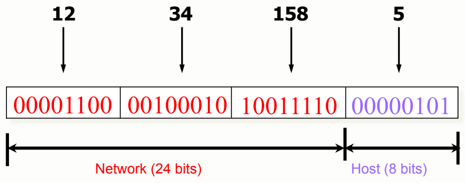
Network Layer: Connect Local Networks



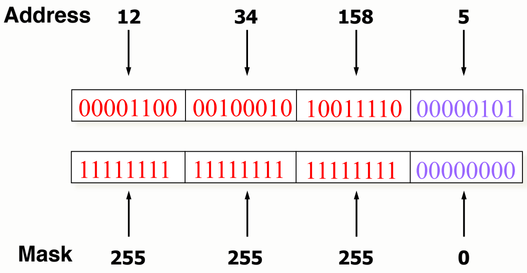
- Main challenges
  - Scalability
  - Autonomy

Network Layer: Hierarchical Addressing

- Network and host portions (left and right)
- 12.34.158.0/24 is a 24-bit **prefix** with 2^8 addresses

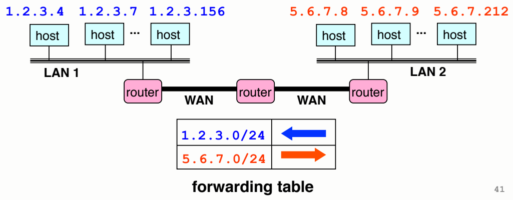


Network Layer: Address and Mask



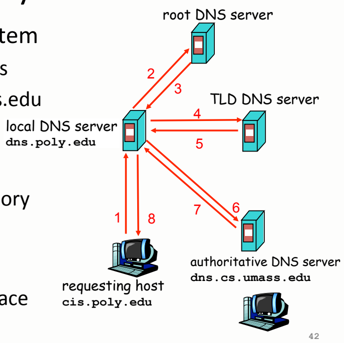
Network Layer: Scalability

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



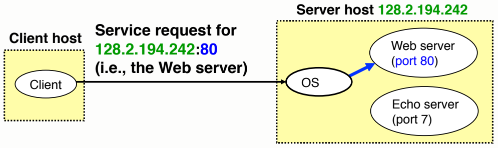
Network Layer: Who are You?

- Domain Name System
  - Hierarchical names
  - E.g., gaia.cs.umass.edu
- Scalability
  - Hierarchical directory
  - Caching of results
- Autonomy
  - Separate name space
  - Separate servers

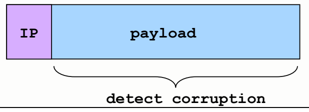


Transport Layer: Two Main Ideas

- **Demultiplexing:** port numbers



- **Error detection:** checksums



43

Transport Layer: User Datagram Protocol (UDP)

- Datagram messaging service
  - Demultiplexing: port numbers
  - Detecting corruption: checksum
- Lightweight communication between processes
  - Send and receive messages
  - Avoid overhead of ordered, reliable delivery

SRC port	DST port
checksum	length
DATA	

44

Transport Layer: Transmission Control Protocol (TCP)

- Stream-of-bytes service
  - Sends and receives a stream of bytes
- Reliable, in-order delivery
  - Corruption: checksums
  - Detect loss/reordering: sequence numbers
  - Reliable delivery: acknowledgments and retransmissions
- Connection oriented
  - Explicit set-up and tear-down of TCP connection
- Flow control
  - Prevent overflow of the receiver's buffer space
- Congestion control
  - Adapt to network congestion for the greater good

45

Application Layer: HyperText Transfer Protocol

Request

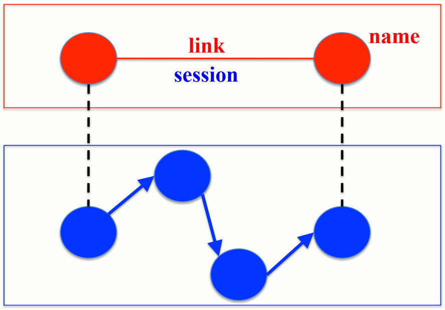
```
GET /courses/archive/spr12/cos461/ HTTP/1.1
Host: www.cs.princeton.edu
User-Agent: Mozilla/4.03
CRLF
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 6 Feb 2012 13:09:03 GMT
Server: Netscape-Enterprise/3.5.1
Last-Modified: Mon, 7 Feb  2011 11:12:23 GMT
Content-Length: 21
CRLF
Site under construction
```

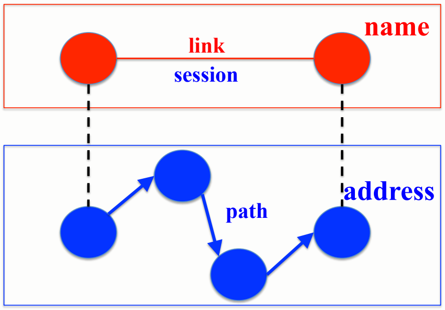
46

Relationship Between Layers



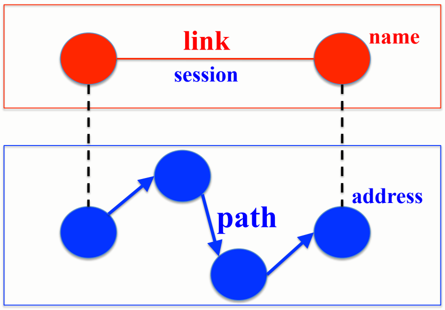
47

Discovery: Mapping Name to Address



48

Routing: Mapping Link to Path



Names: Different Kinds

- **Host name** (e.g., www.cs.princeton.edu)
  - Mnemonic, variable-length, appreciated *by humans*
  - Hierarchical, based on organizations
- **IP address** (e.g., 128.112.7.156)
  - Numerical 32-bit address appreciated *by routers*
  - Hierarchical, based on organizations and topology
- **MAC address** (e.g., 00-15-C5-49-04-A9)
  - Numerical 48-bit address appreciated *by adapters*
  - Non-hierarchical, unrelated to network topology

Names: Hierarchical Assignment

- **Host name:** www.cs.princeton.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- **IP addresses:** 128.112.7.156
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP
- **MAC addresses:** 00-15-C5-49-04-A9
  - Blocks: assigned to vendors by the IEEE
  - Adapters: assigned by the vendor from its block

Directories

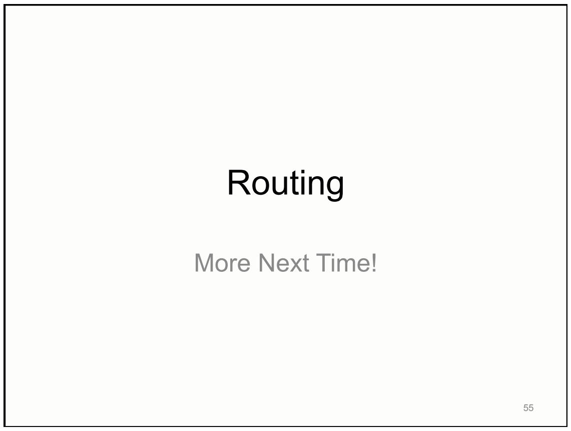
- A key-value store
  - Key: name, value: address(es)
  - Answer queries: given name, return address(es)
- Caching the response
  - Reuse the response, for a period of time
  - Better performance and lower overhead
- Allow entries to change
  - Updating the address(es) associated with a name
  - Invalidating or expiring cached responses

Directory Design: Three Extremes

- Flood the query (e.g., ARP)
  - The named node responds with its address
  - But, high overhead in large networks
- Push data to all clients (/etc/hosts)
  - All nodes store a full copy of the directory
  - But, high overhead for many names and updates
- Central directory server
  - All data and queries handled by one machine
  - But, poor performance, scalability, and reliability

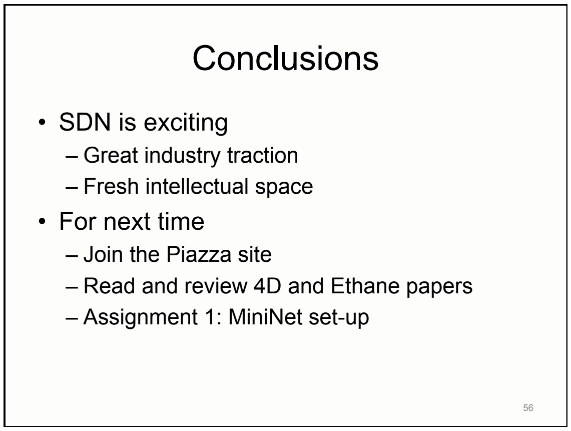
Directory Design: Distributed Solutions

- Hierarchical directory (e.g., DNS)
  - Follow the hierarchy in the name space
  - Distribute the directory, distribute the queries
  - Enable decentralized updates to the directory
- Distributed Hash Table (e.g. P2P applications)
  - Directory as a hash table with flat names
  - Each node handles range of hash outputs
  - Use hash to direct query to the directory node



Routing

More Next Time!



Conclusions

- SDN is exciting
  - – Great industry traction
  - – Fresh intellectual space
- For next time
  - – Join the Piazza site
  - – Read and review 4D and Ethane papers
  - – Assignment 1: MiniNet set-up