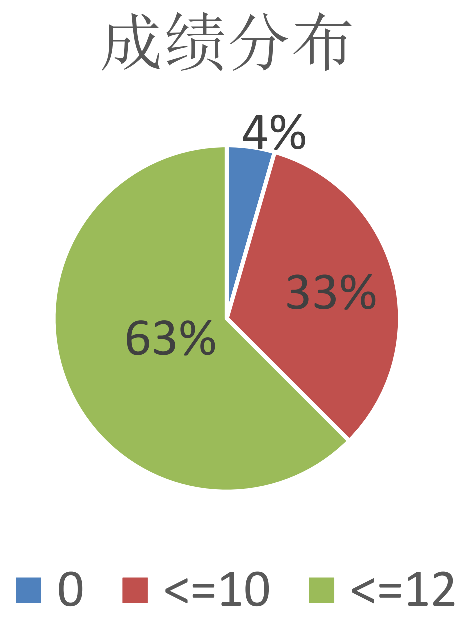
Z6110X0035: Introduction to Cloud Computing - Network

Lecturer: Prof. Zichen Xu

Recap from last Lecture

- Statistics about homework

- News on labs



Outline

- Data Center network overview
- Network system basics
- Data Center network efficiency

Data center networks

10's to 100's of thousands of hosts, often closely coupled, in close proximity:

- e-business (e.g. Amazon)
- content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
- search engines, data mining (e.g., Google)

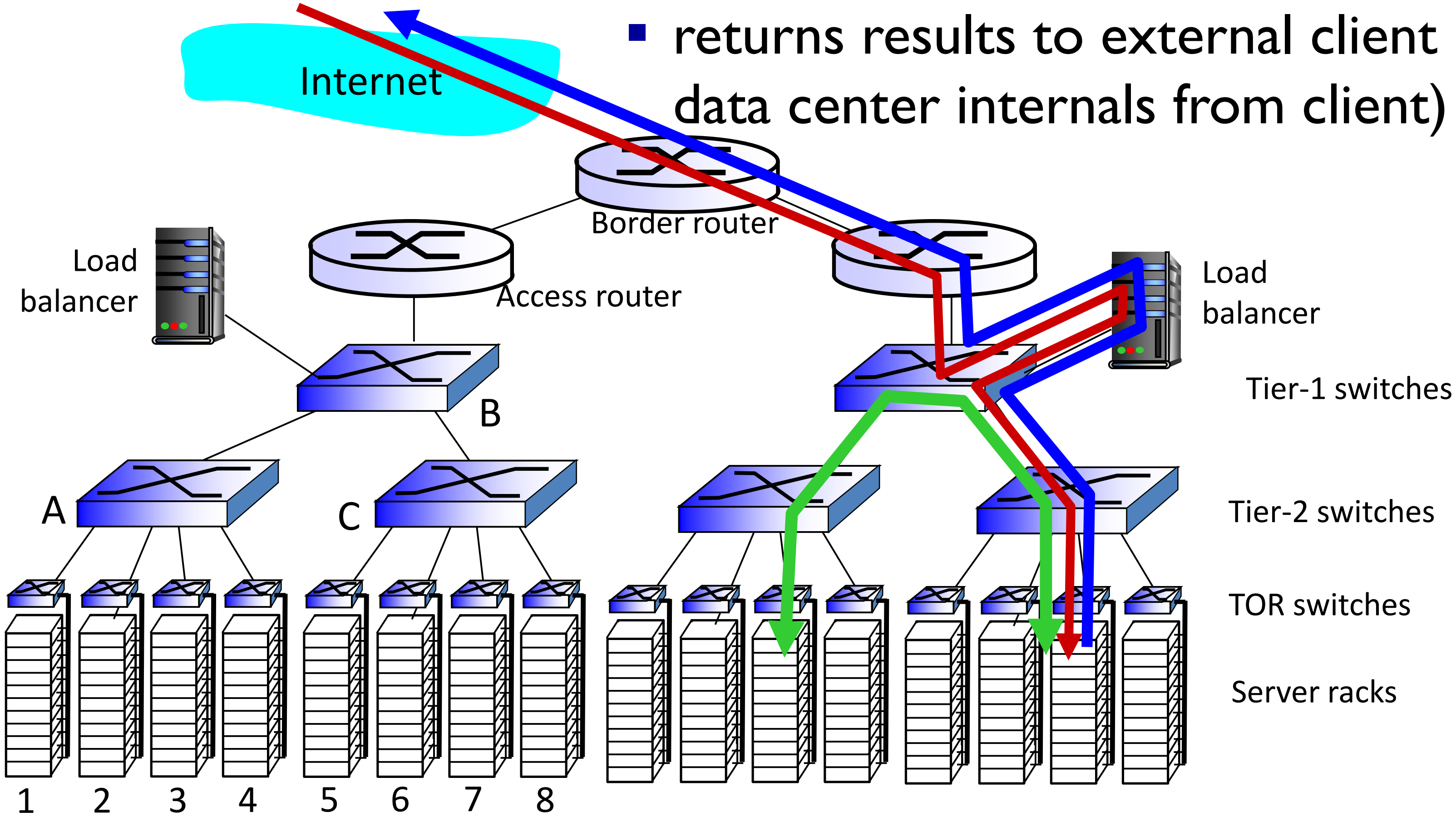
- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Data center networks

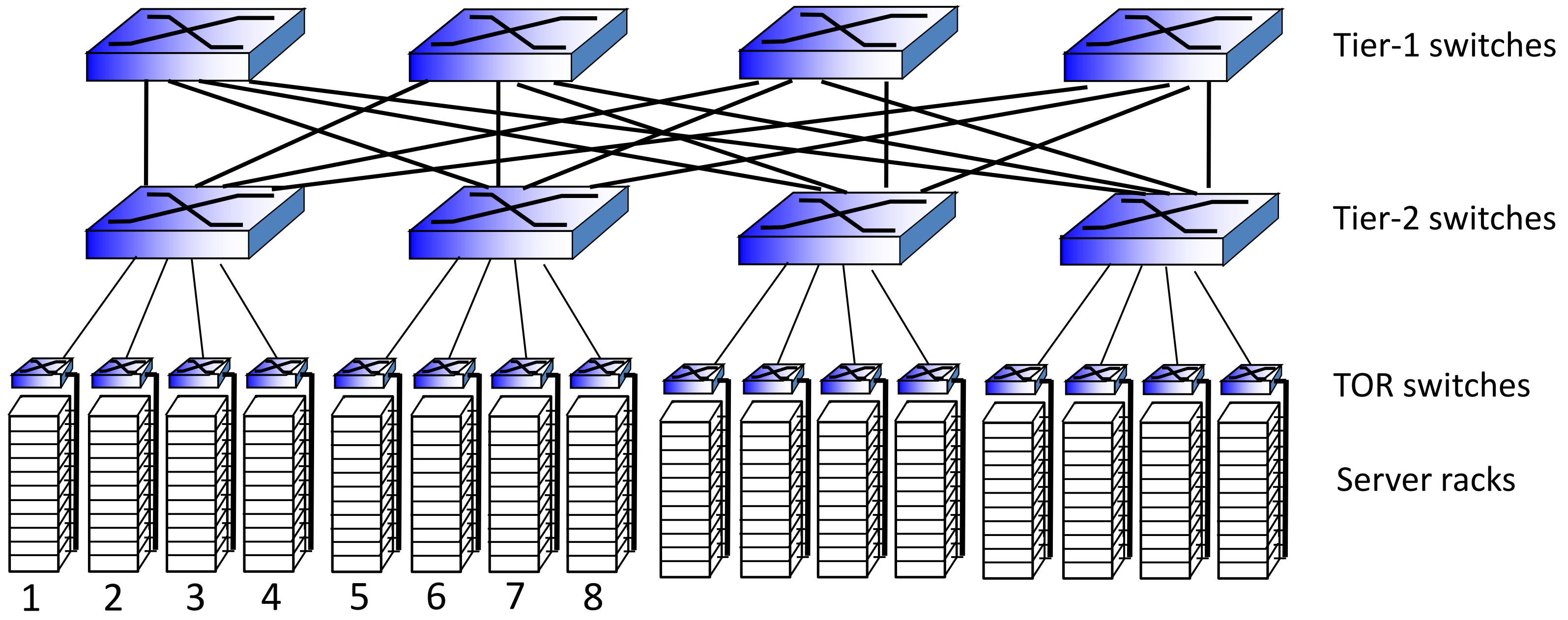
load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Data center networks

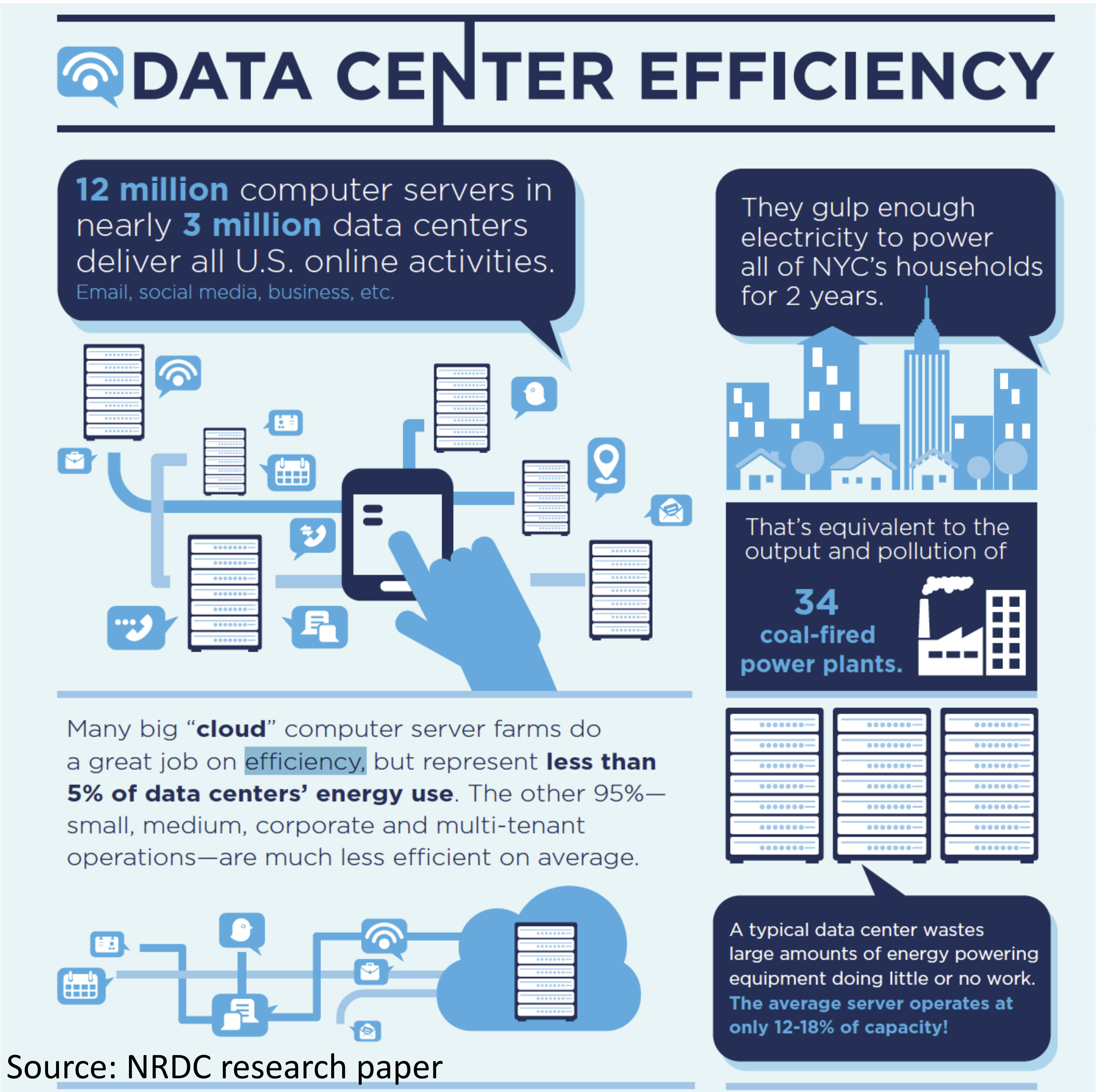
- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



Broad questions

- How are massive numbers of commodity machines networked inside a data center?
- Virtualization: How to effectively manage physical machine resources across client virtual machines?

- Operational costs:
  - Server equipment
  - **Power and cooling**



Source: NRDC research paper

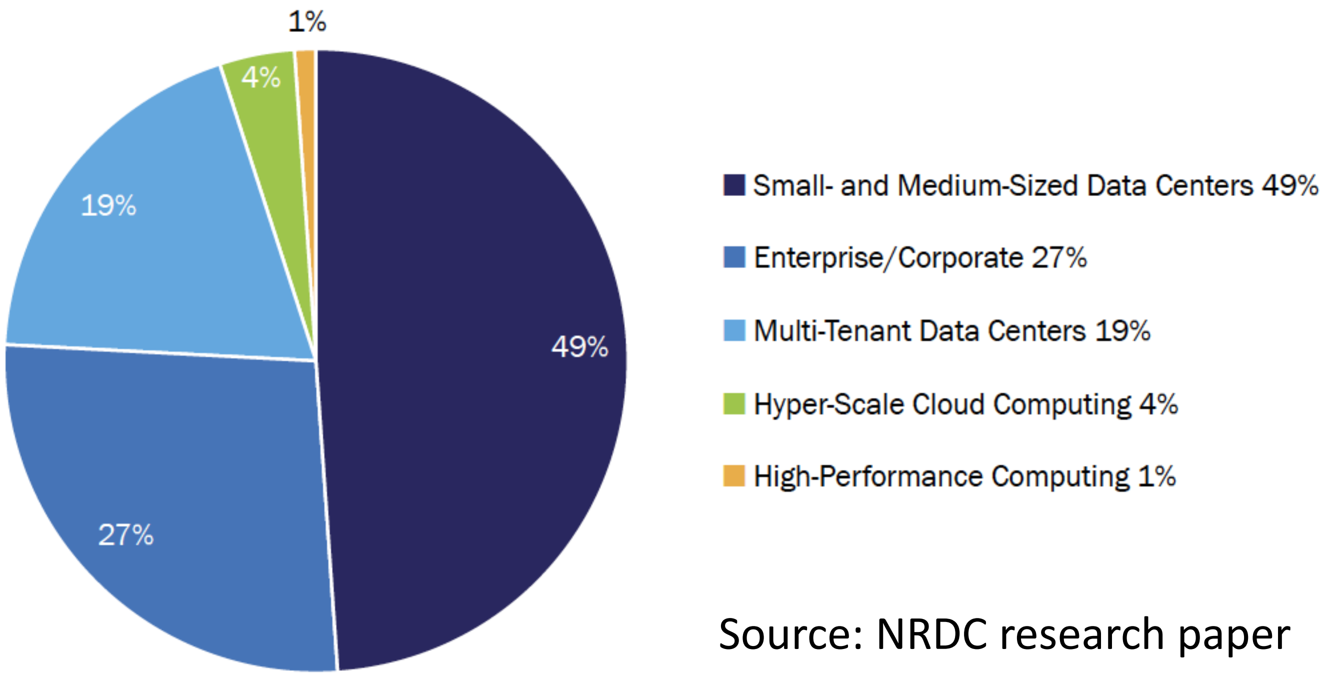
Breakc

Table 1: Estimated U.S. data center electricity consumption by market segment (2011)

Segment	Number of Servers (million)	Electricity Share	Total U.S. Data Center Electricity Use (billion kWh/y)
Small and Medium Server Rooms	4.9	49%	37.5
Enterprise/Corporate Data Centers	3.7	27%	20.5
Multi-Tenant Data Centers	2.7	19%	14.1
Hyper-Scale Cloud Computing	0.9	4%	3.3
High-Performance Computing	0.1	1%	1.0
Total (rounded)	**12.2**	**100%**	**76.4**

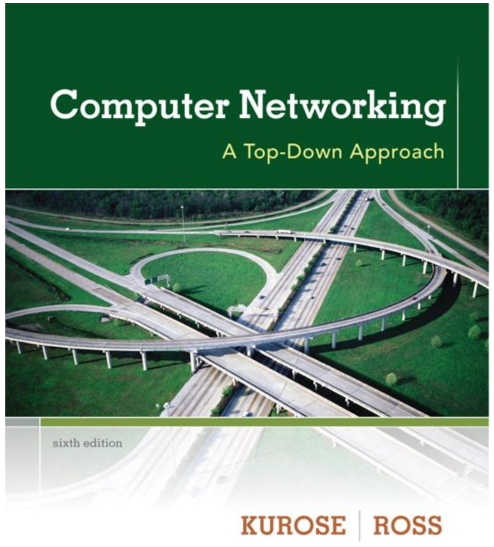
See Appendix 2 for source information

Figure 1: Estimated U.S. data center electricity consumption by market segment (2011)



Source: NRDC research paper

Computer Networking



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Link layer: introduction

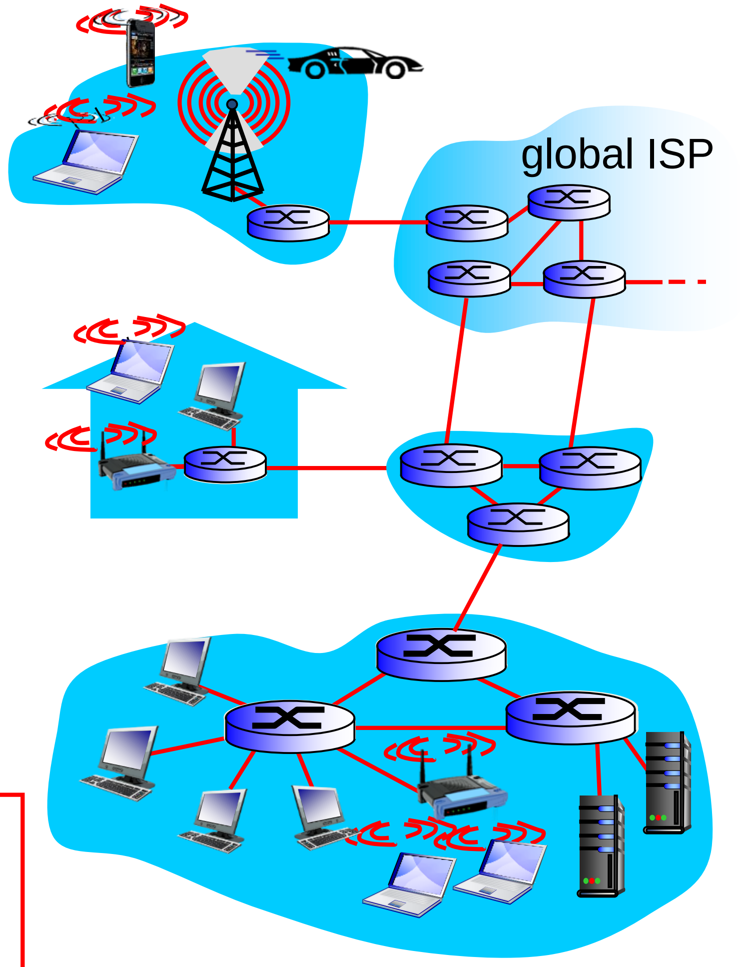
terminology:

hosts and routers: nodes

communication channels that connect adjacent nodes along communication path: links

- wired links
- wireless links
- LANs

layer-2 packet: frame, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Link layer: context

datagram transferred by different link protocols over different links:

- e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link

each link protocol provides different services

- e.g., may or may not provide rdt over link

transportation analogy:

trip from Amherst to Lausanne

- limo: Amherst to BOS
- plane: BOS to Geneva
- train: Geneva to Lausanne

tourist = datagram

transport segment = communication link

transportation mode = link layer protocol

travel agent = routing algorithm

An ideal multiple access protocol

given: broadcast channel of rate R bps

goal:

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

three broad classes:

channel partitioning

- divide channel into smaller "pieces" (time slots, frequency, code)
- allocate piece to node for exclusive use

random access

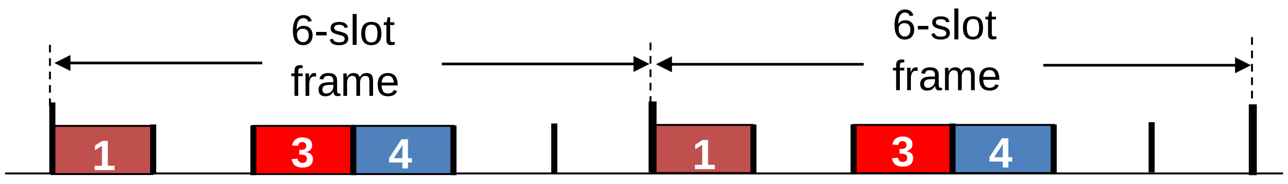
- channel not divided, allow collisions
- "recover" from collisions

"taking turns"

- nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access
access to channel in "rounds"
each station gets fixed length slot (length = pkt trans time) in each round
unused slots go idle
example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

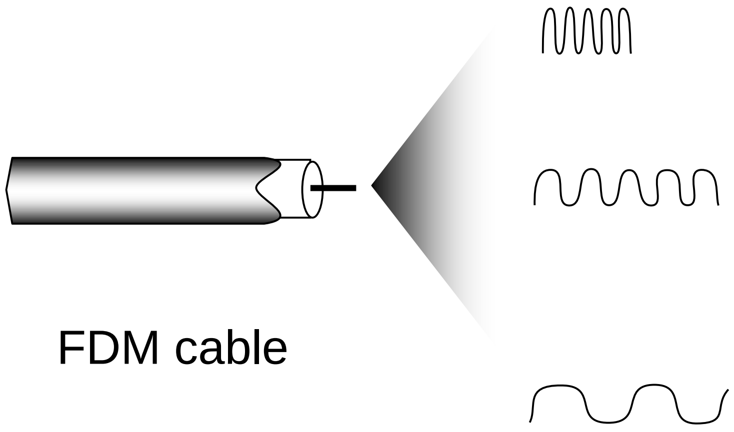
FDMA: frequency division multiple access

channel spectrum divided into frequency bands

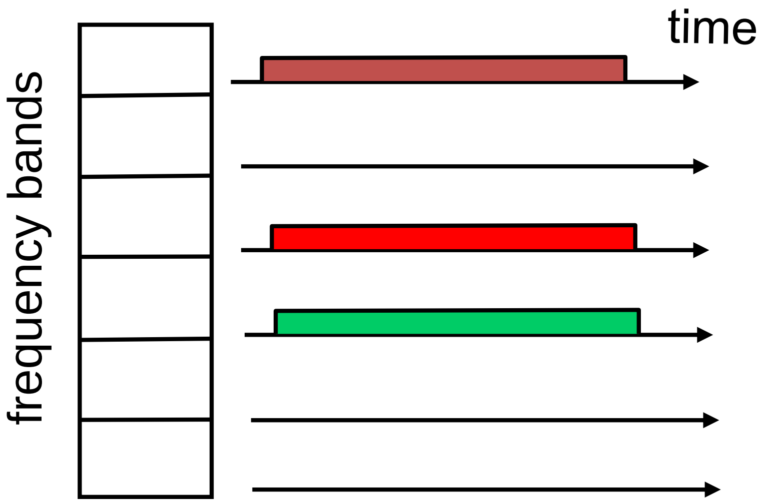
each station assigned fixed frequency band

unused transmission time in frequency bands go idle

example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



FDM cable



Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",

random access MAC protocol specifies:

- how to detect collisions
- how to recover from collisions (e.g., via delayed retransmissions)

- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

if channel sensed idle: transmit entire frame

if channel sensed busy, defer transmission

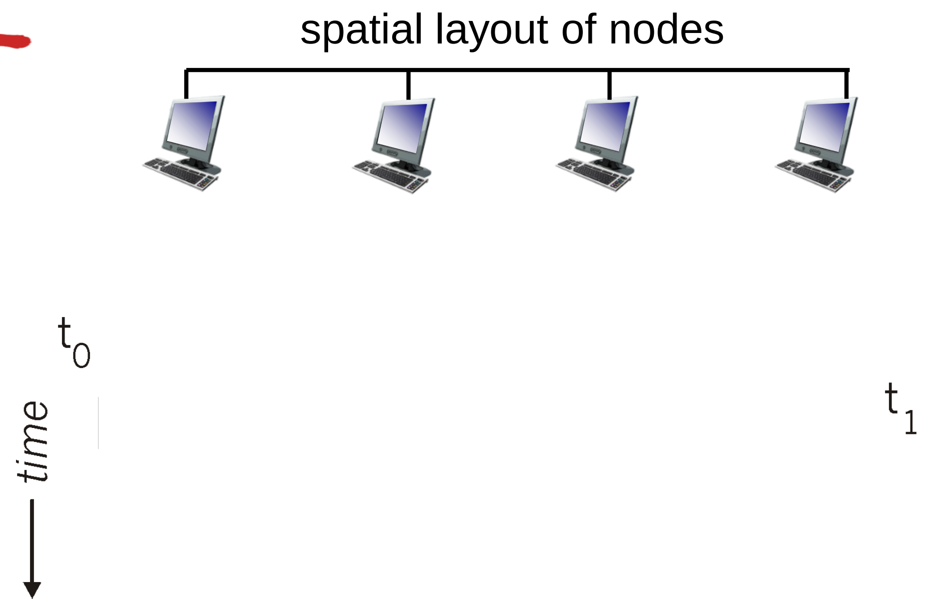
human analogy: don't interrupt others!

CSMA collisions

collisions *can* still occur: propagation delay means two nodes may not hear other's transmission

collision: entire packet transmission time wasted

- distance & propagation delay play role in in determining collision probability



CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

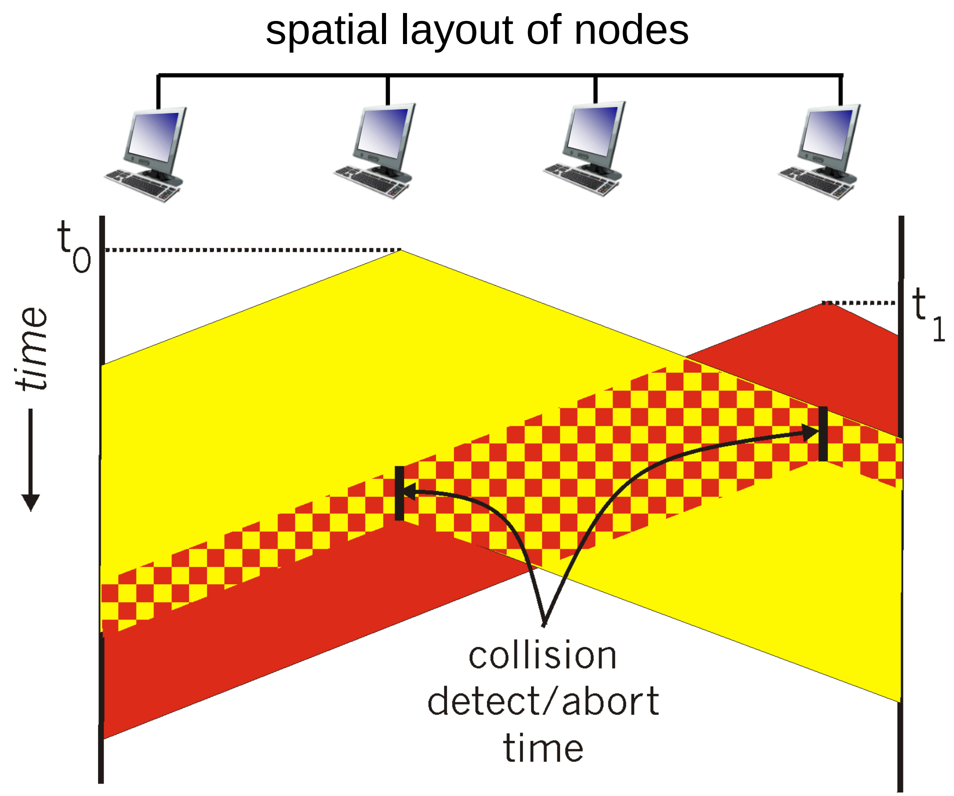
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

collision detection:

- easy in wired LANs: measure signal strengths, compare transmitted, received signals
- difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

human analogy: the polite conversationalist

CSMA/CD (collision detection)



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. Else if NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters *binary (exponential) backoff:*
   - after *m*th collision, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^m-1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
   - longer backoff interval with more collisions

CSMA/CD efficiency

t_{prop} = max prop delay between 2 nodes in LAN
t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

efficiency goes to 1

- as t_{prop} goes to 0
- as t_{trans} goes to infinity

better performance than ALOHA: and simple, cheap, decentralized!

"Taking turns" MAC protocols

- **channel partitioning MAC protocols:**
  - share channel *efficiently* and *fairly* at high load
  - inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!
- **random access MAC protocols**
  - efficient at low load: single node can fully utilize channel
  - high load: collision overhead
- **"taking turns" protocols**
  - look for best of both worlds!

"Taking turns" MAC protocols

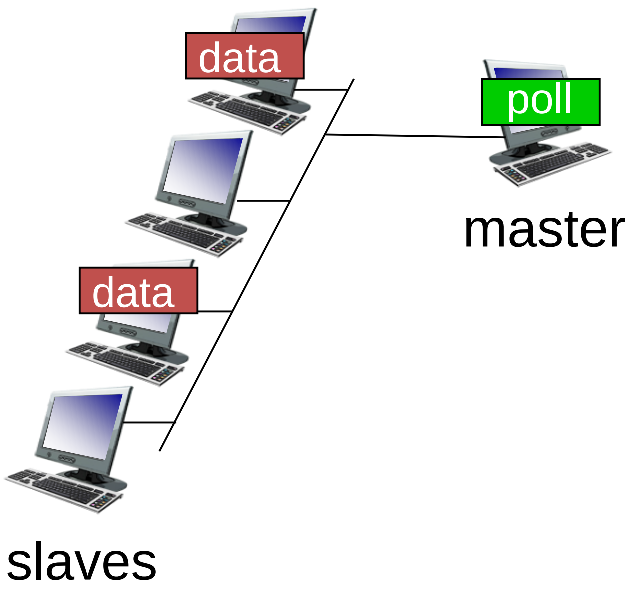
polling:

master node "invites" slave nodes to transmit in turn

typically used with "dumb" slave devices

concerns:

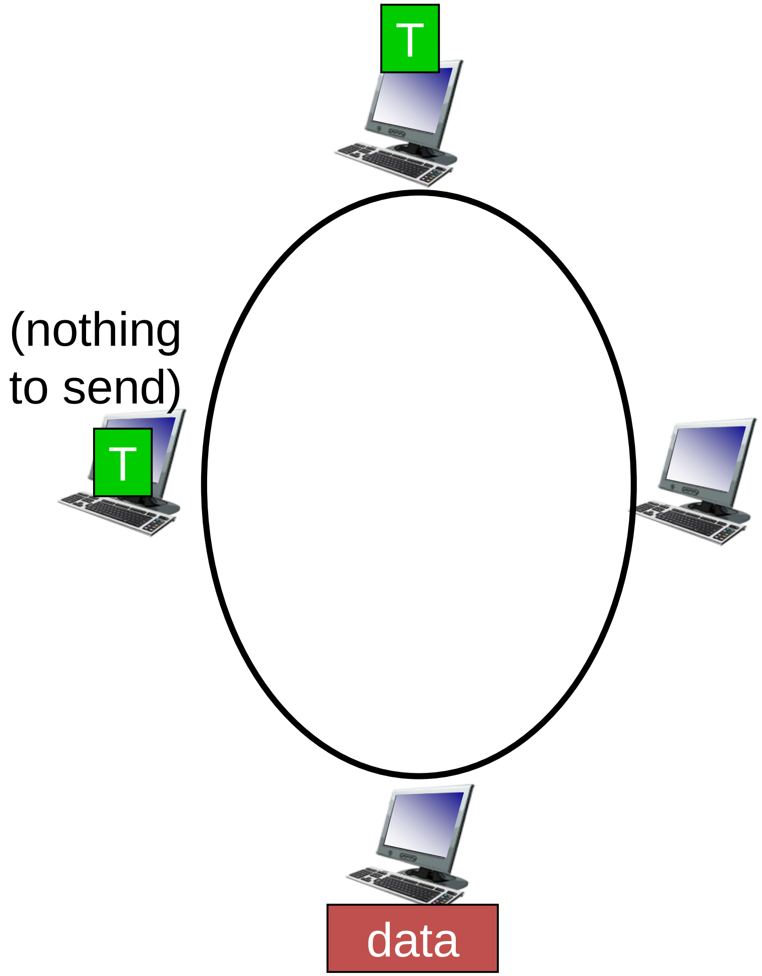
- polling overhead
- latency
- single point of failure (master)



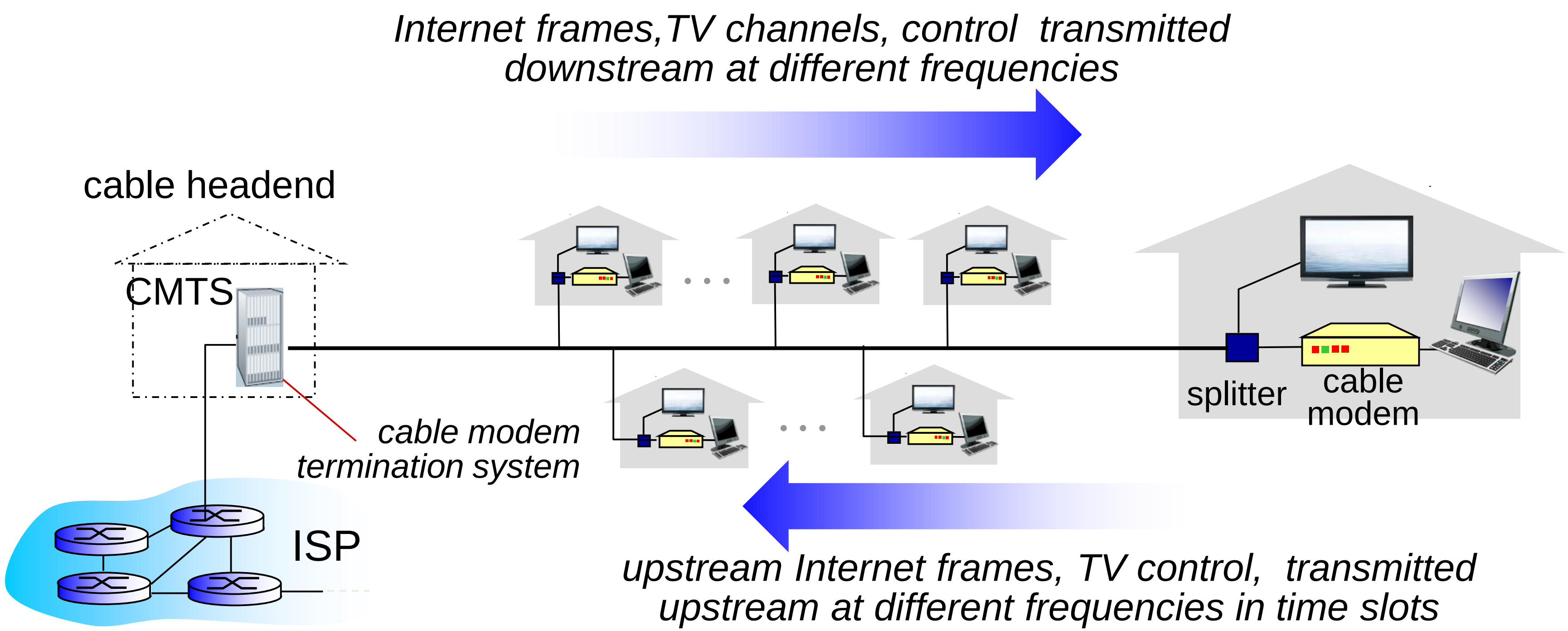
"Taking turns" MAC protocols

token passing:

- control *token* passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)

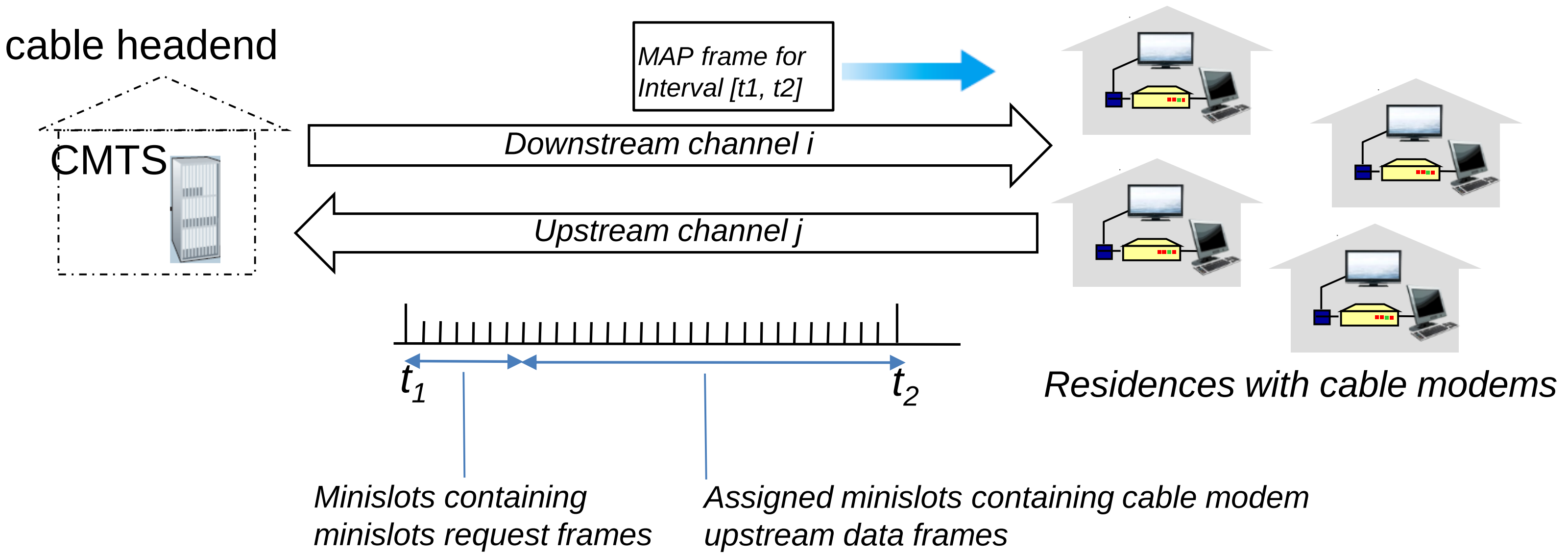


Cable access network



- *multiple* 40Mbps downstream (broadcast) channels
  - single CMTS transmits into channels
- *multiple* 30 Mbps upstream channels
  - *multiple access: all* users contend for certain upstream channel time slots (others assigned)

Cable access network



DOCSIS: data over cable service interface spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - bluetooth, FDDI, token ring

Ethernet switch

link-layer device: takes an *active* role

- store, forward Ethernet frames
- examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment

transparent

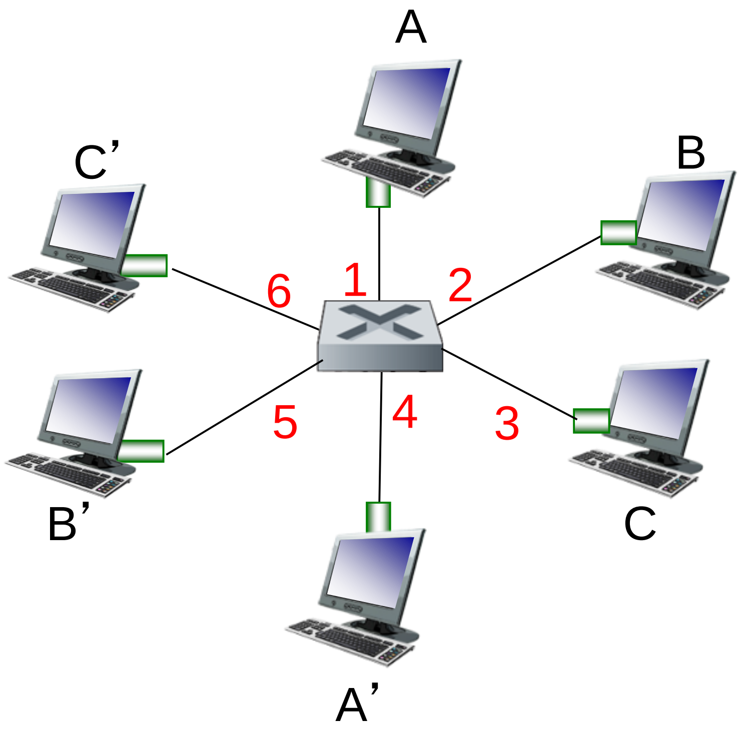
- hosts are unaware of presence of switches

plug-and-play, self-learning

- switches do not need to be configured

Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions



switch with six interfaces (1,2,3,4,5,6)

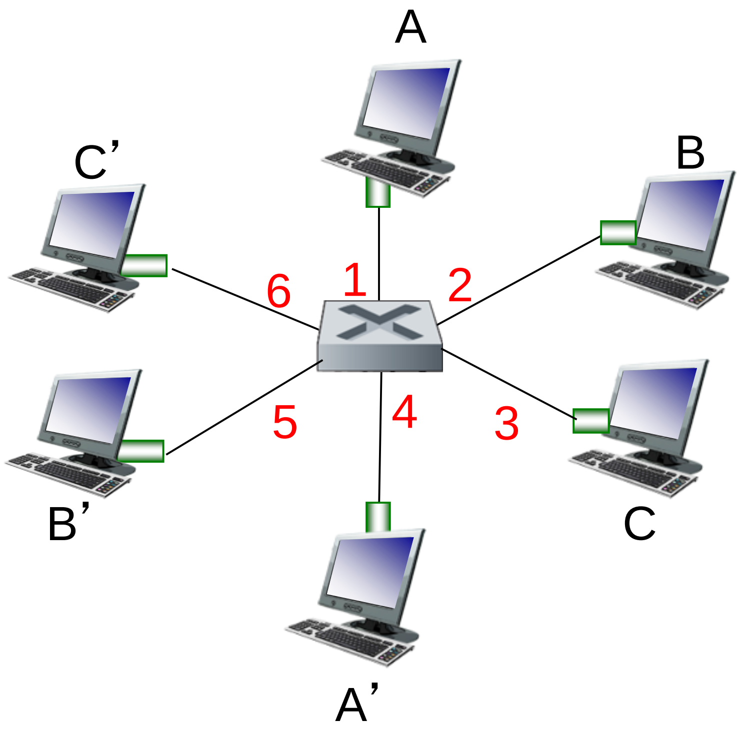
Switch forwarding table

Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

- *A: each switch has a switch table, each entry:*
  - *(MAC address of host, interface to reach host, time stamp)*
  - *looks like a routing table!*

Q: how are entries created, maintained in switch table?

- *something like a routing protocol?*

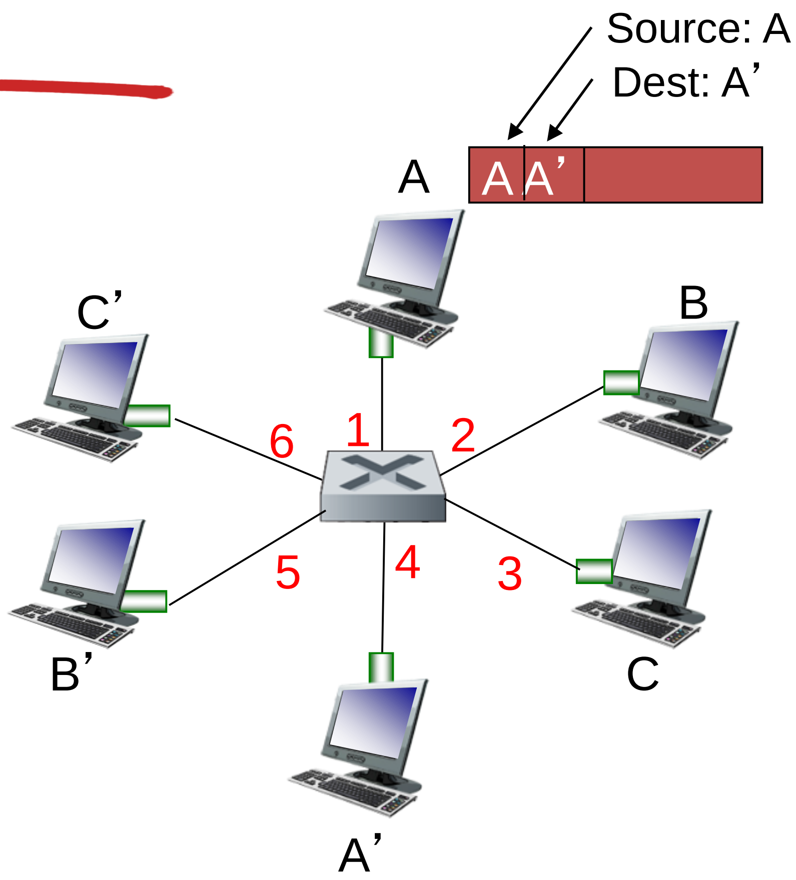


switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

switch *learns* which hosts can be reached through which interfaces

- when frame received, switch "learns" location of sender: incoming LAN segment
- records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Switch: frame filtering/forwarding

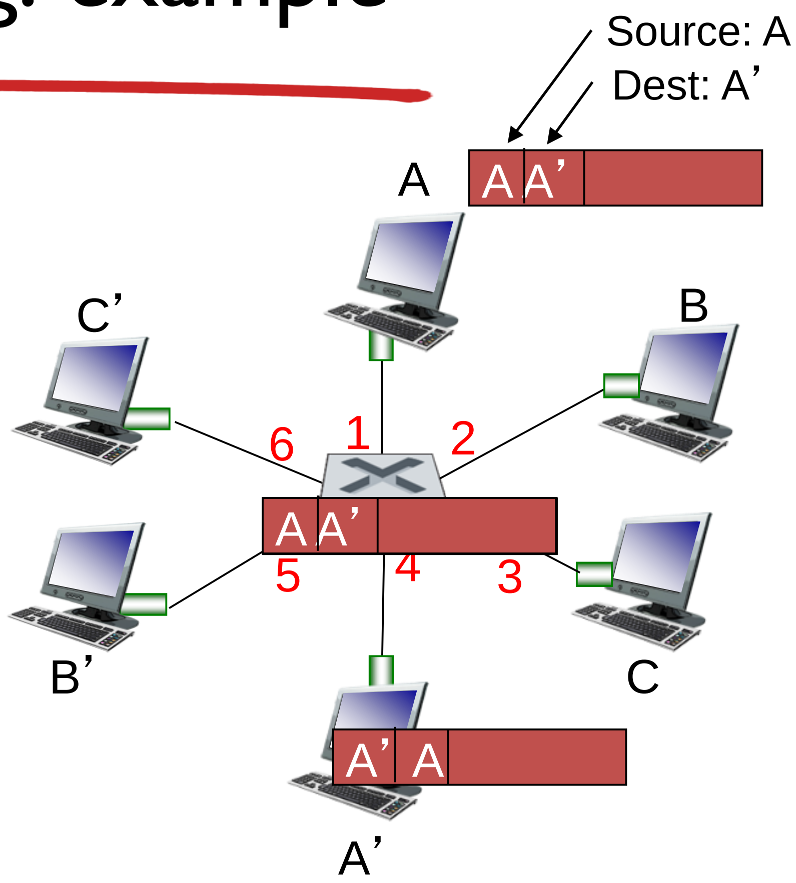
when frame received at switch:

```
1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
  then {
    if destination on segment from which frame arrived
      then drop frame
        else forward frame on interface indicated by entry
     }
    else flood  /* forward on all interfaces except arriving
                   interface */
```

Self-learning, forwarding: example

frame destination, A', locaton unknown: *flood*

- destination A location known: *selectively send on just one link*

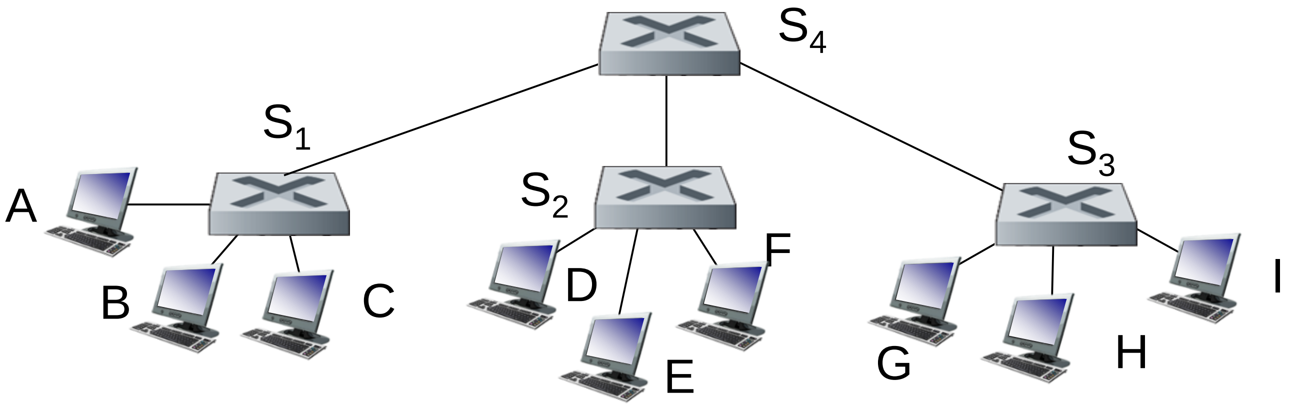


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

switches can be connected together

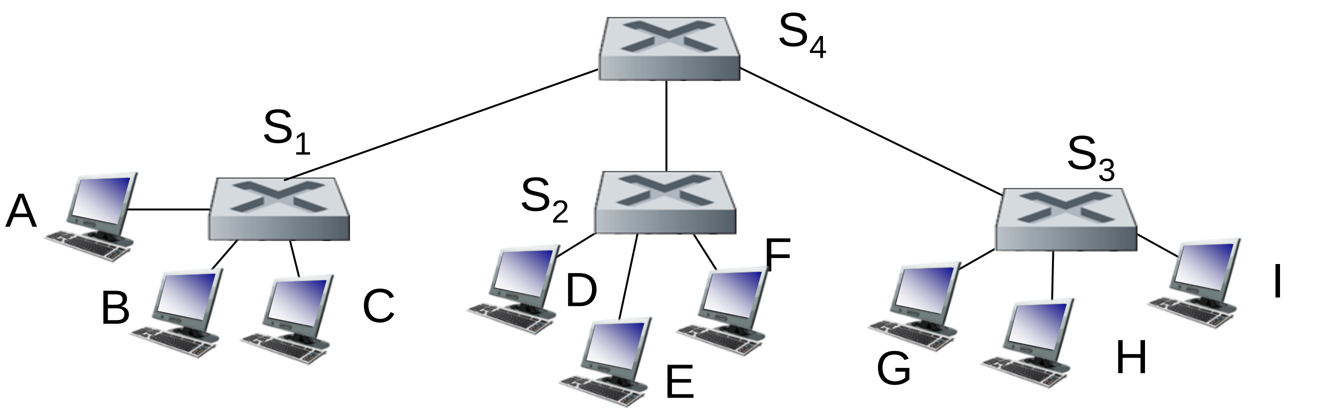


Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?

- *A:* self learning! (works *exactly* the same as in single-switch case!)

Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3, S_4

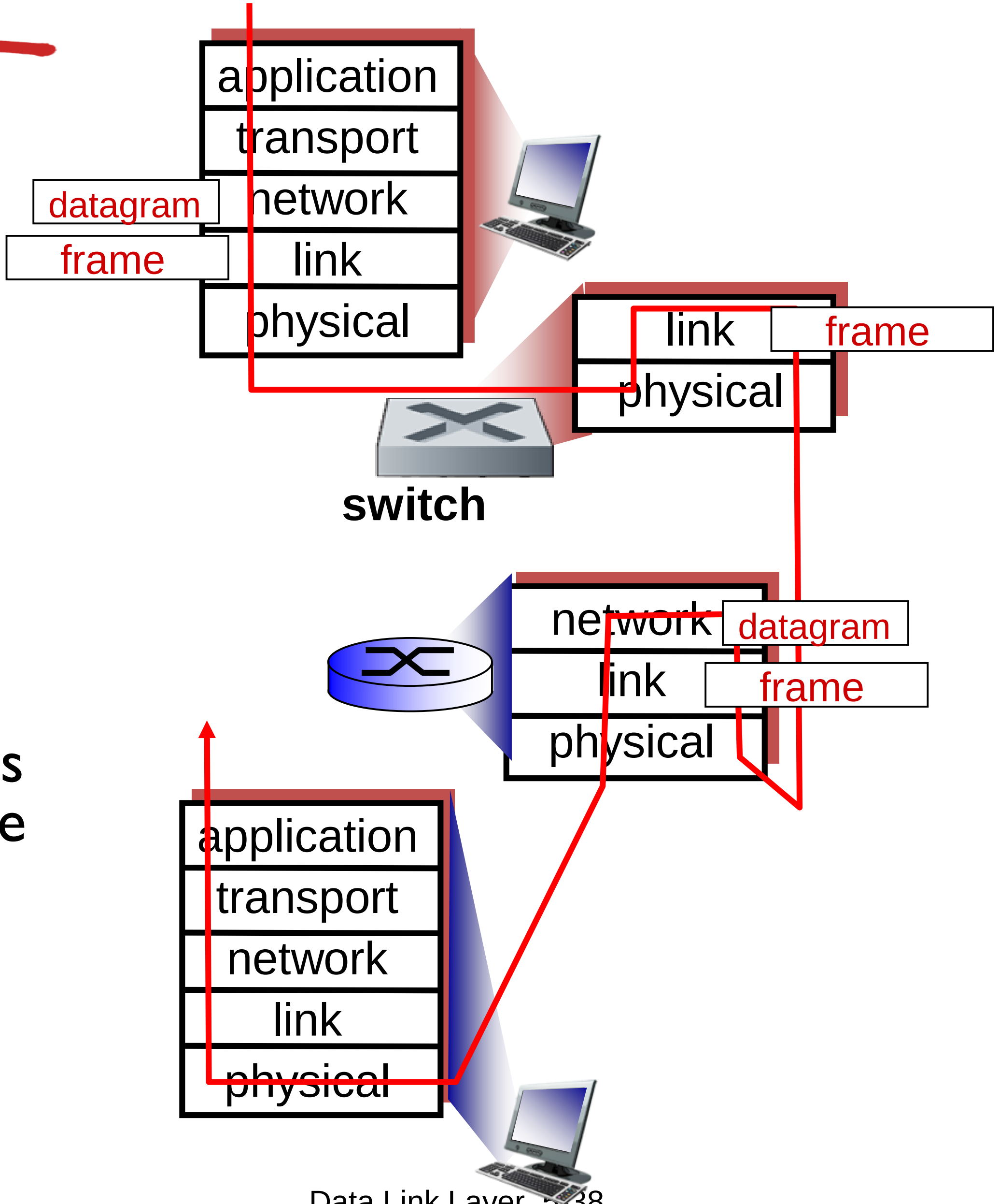
Switches vs. routers

both are store-and-forward:

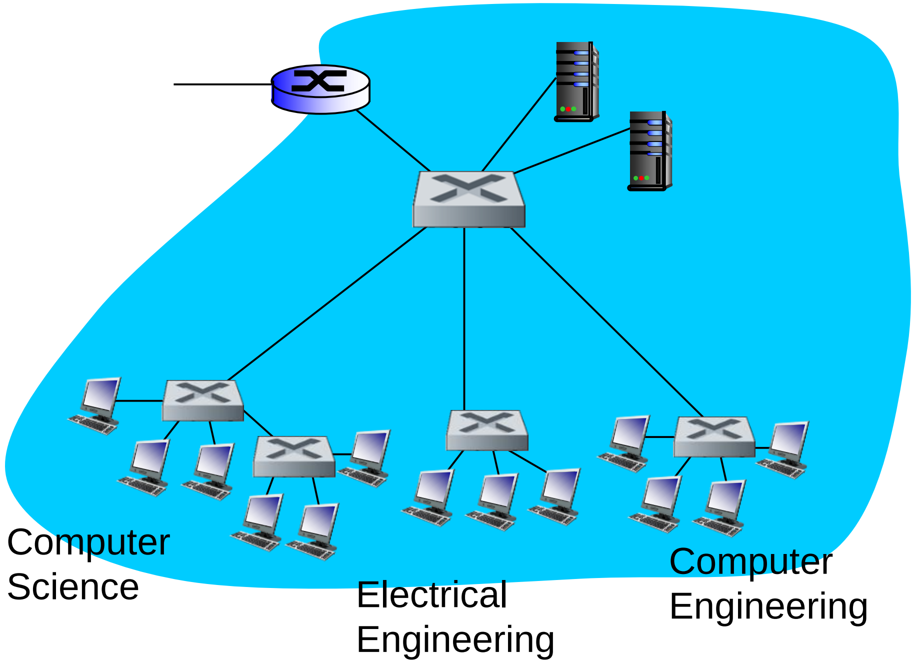
- *routers:* network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



VLANs: motivation



consider:

CS user moves office to EE, but wants connect to CS switch?

single broadcast domain:

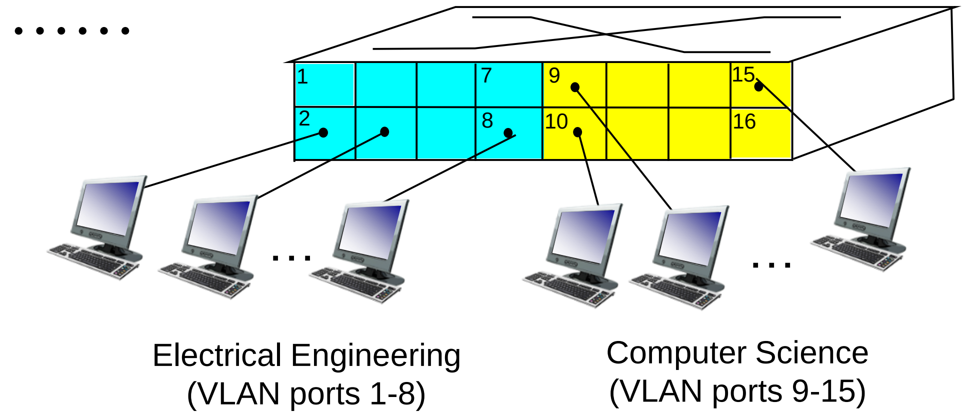
- all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
- security/privacy, efficiency issues

VLANs

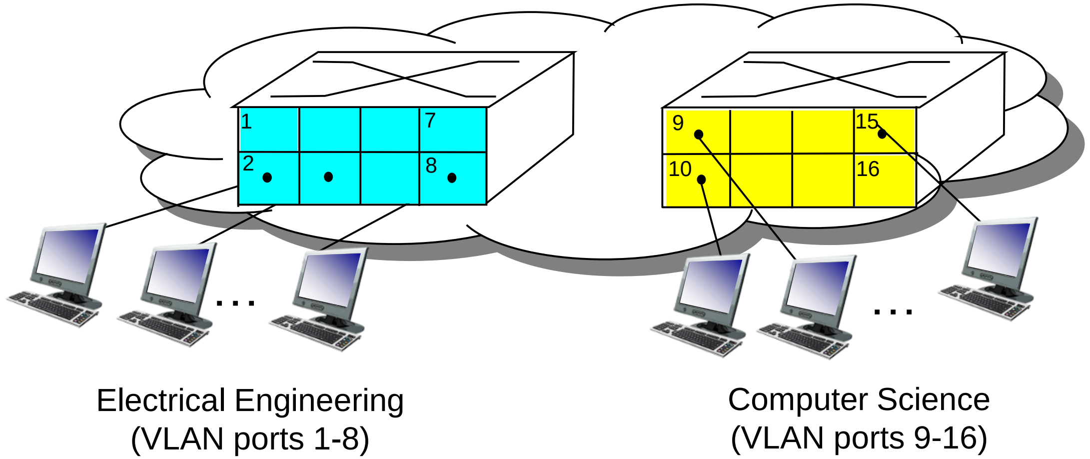
Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……



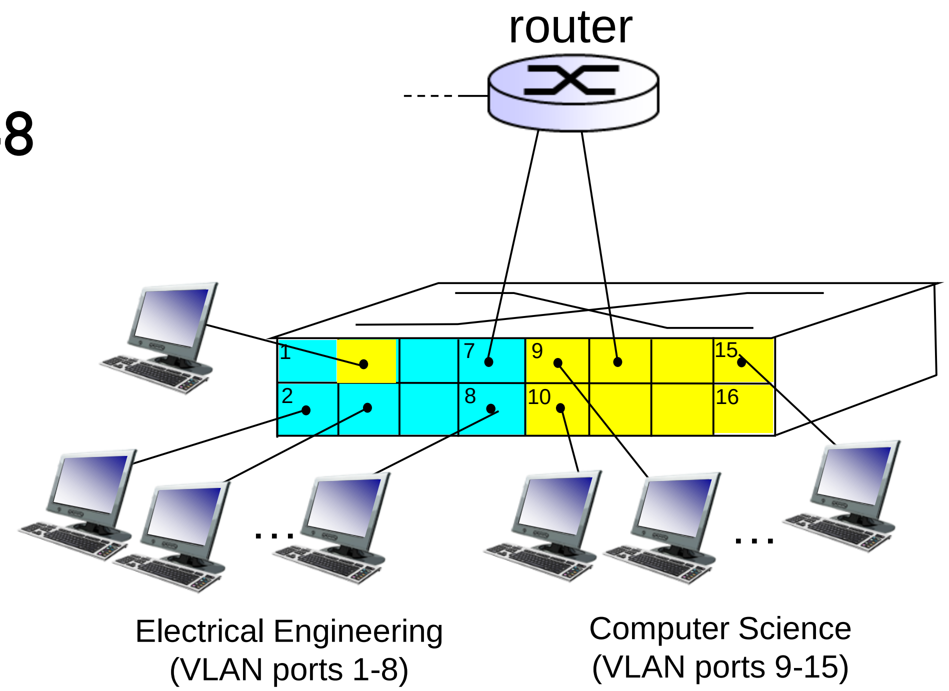
… operates as *multiple* virtual switches



Port-based VLAN

traffic isolation: frames to/from ports 1-8 can *only* reach ports 1-8

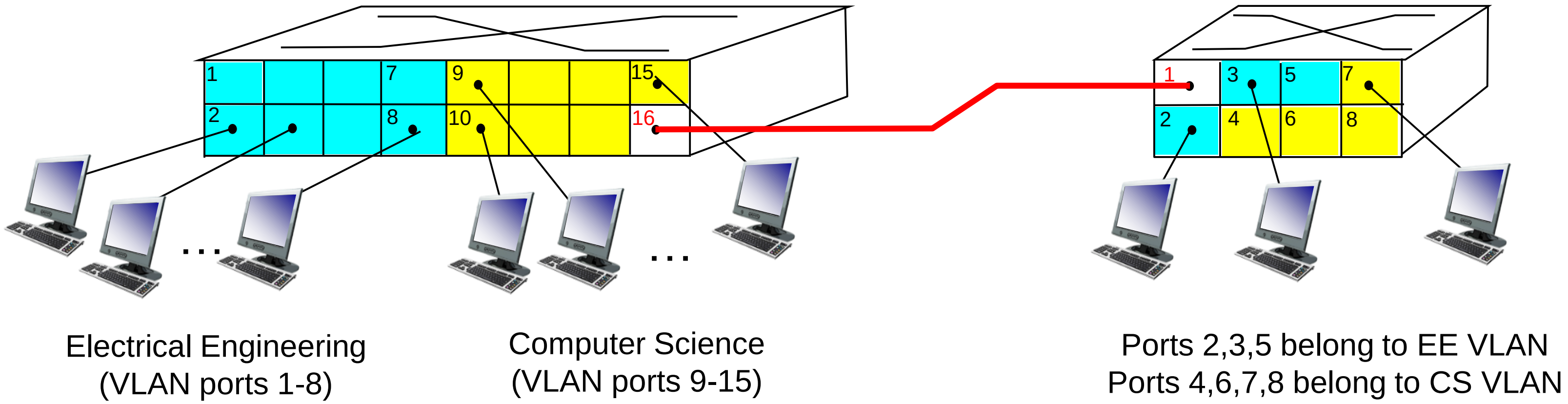
can also define VLAN based on MAC addresses of endpoints, rather than switch port



- *dynamic membership:* ports can be dynamically assigned among VLANs

- *forwarding between VLANS:* done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers

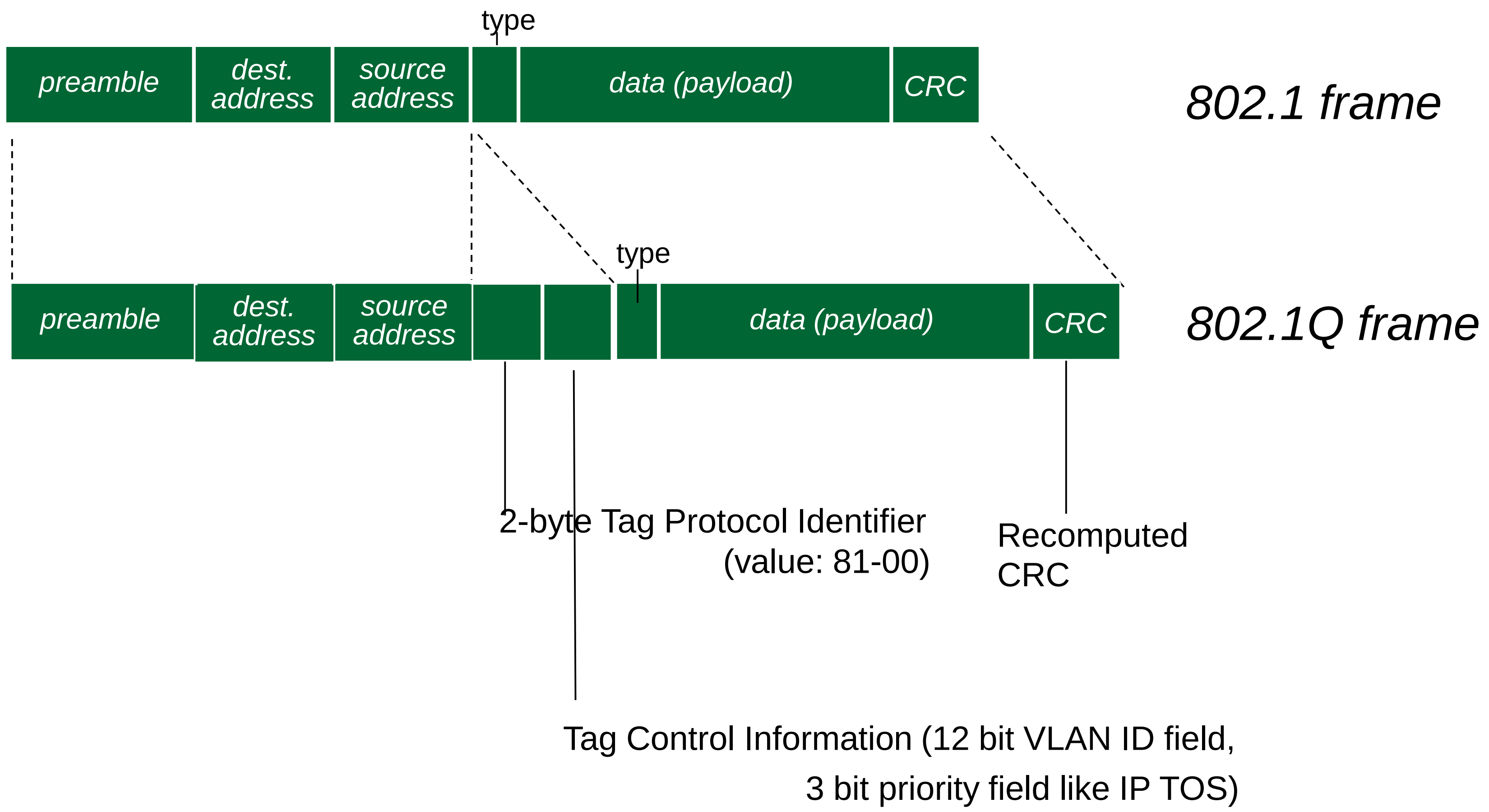
VLANS spanning multiple switches



trunk port: carries frames between VLANS defined over multiple physical switches

- frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
- 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format



Data center networks

10's to 100's of thousands of hosts, often closely coupled, in close proximity:

e-business (e.g. Amazon)
content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
search engines, data mining (e.g., Google)

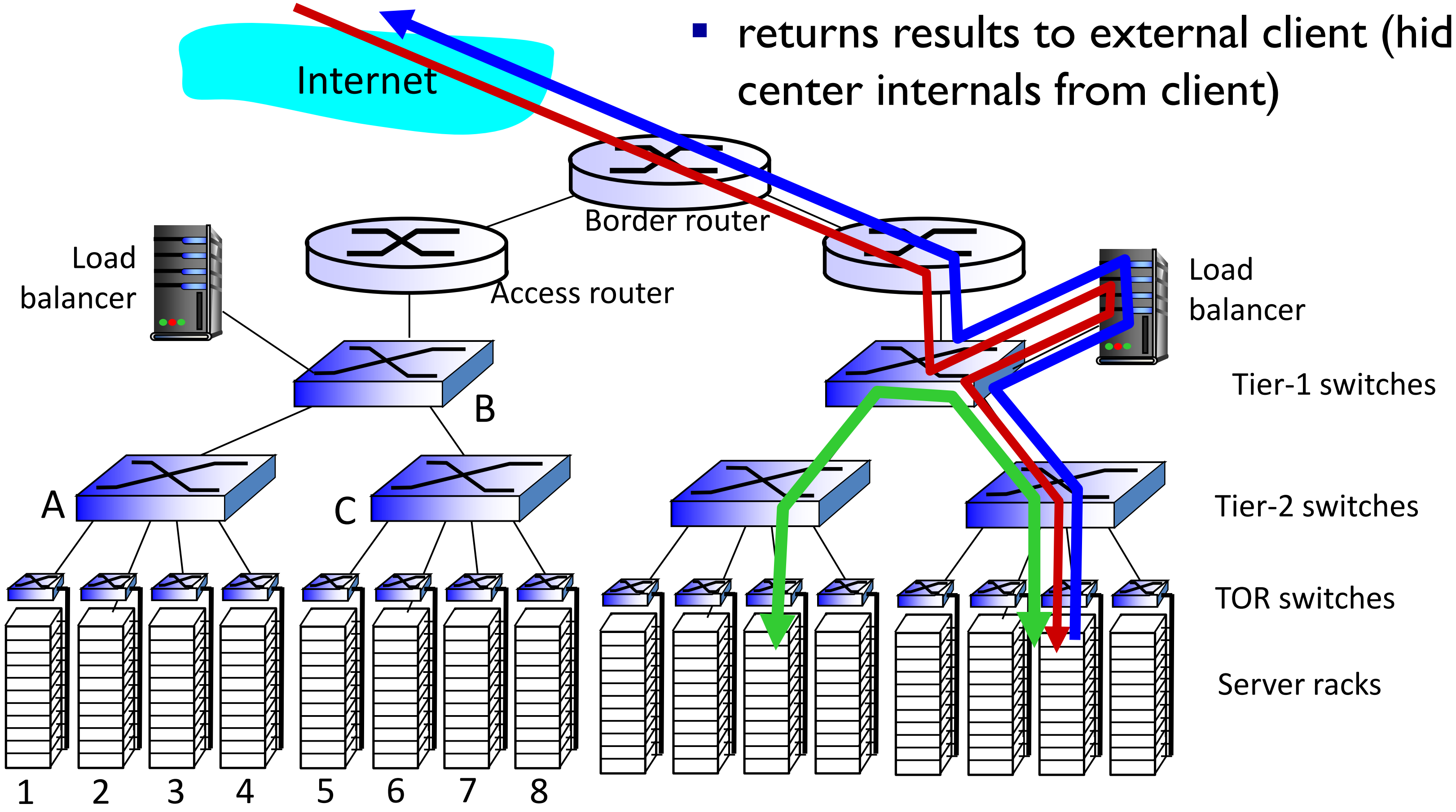
- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Data center networks

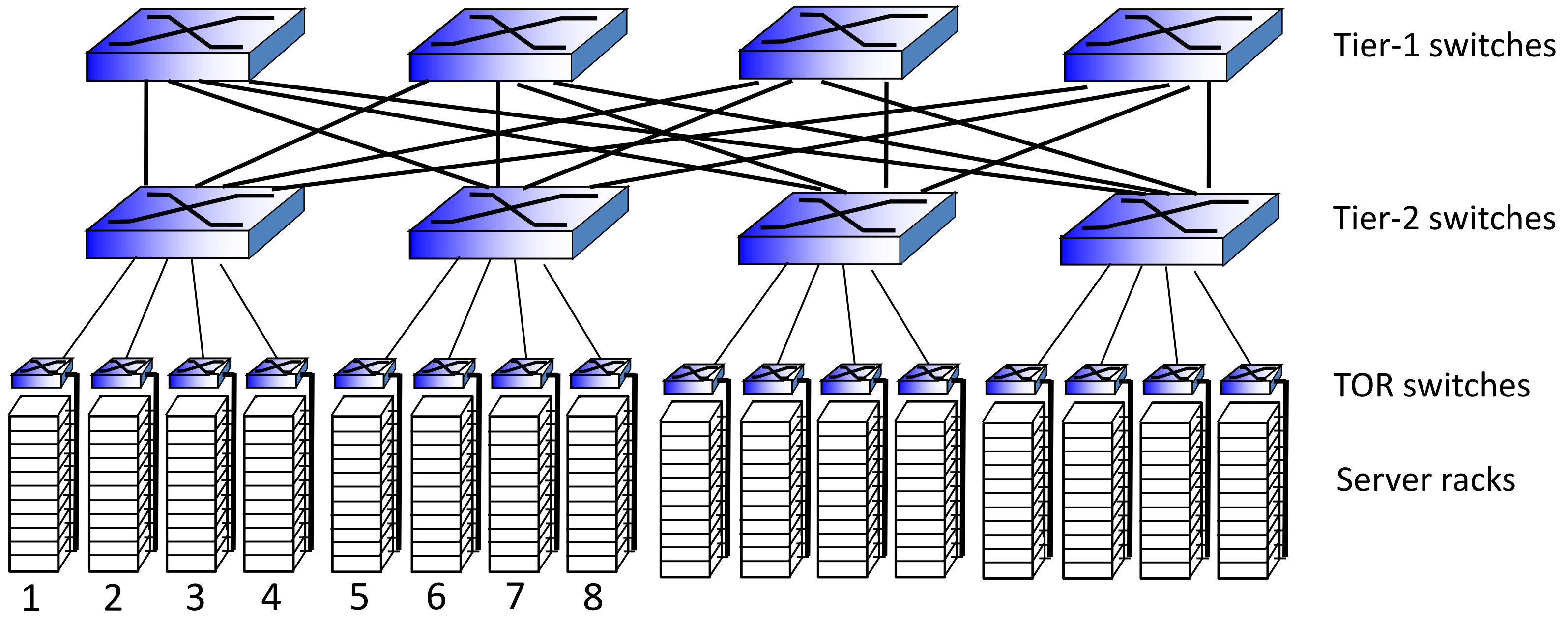
load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Data center networks

- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request

Synthesis: a day in the life of a web request

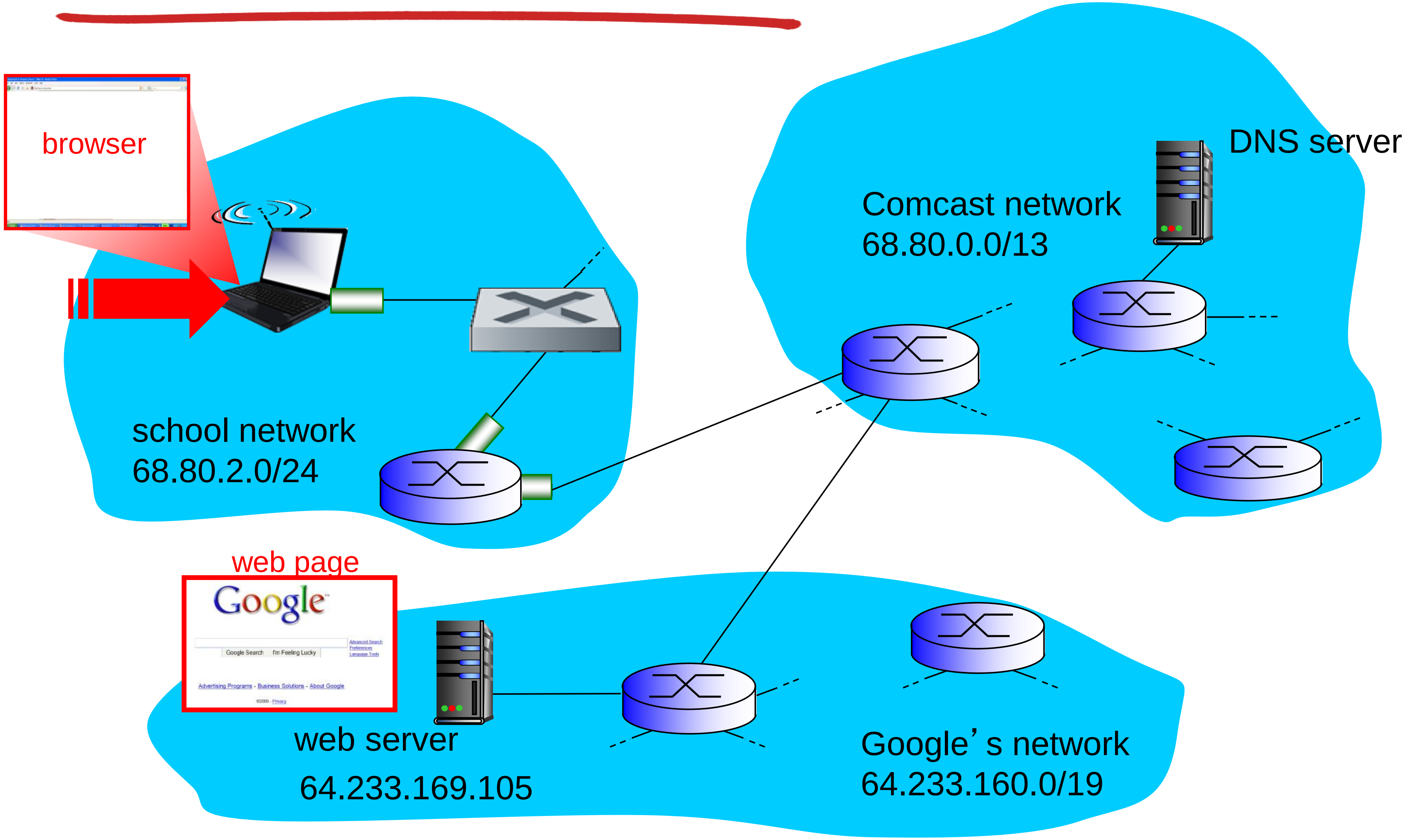
journey down protocol stack complete!

- application, transport, network, link

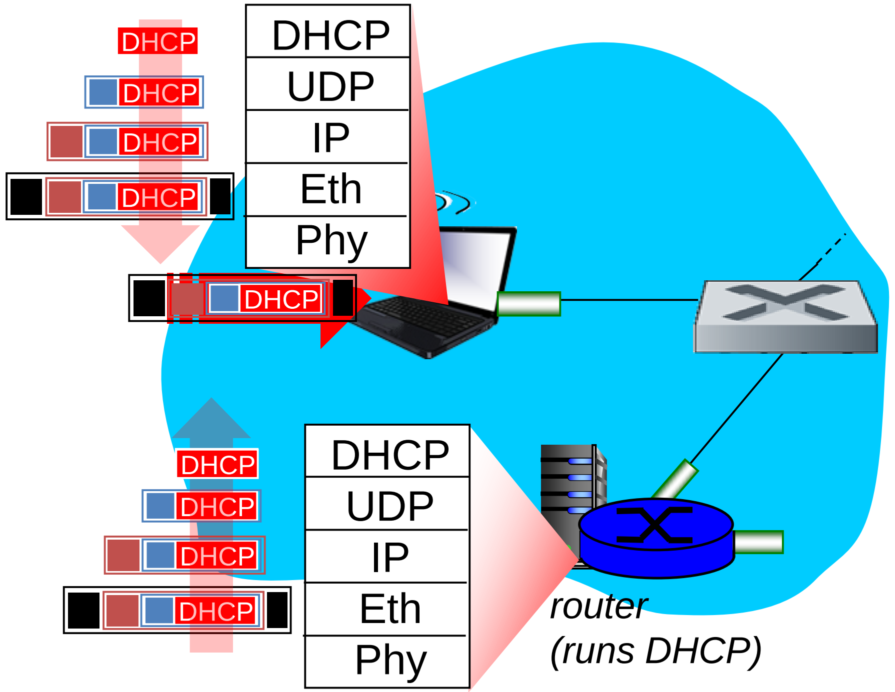
putting-it-all-together: synthesis!

- *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
- *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario



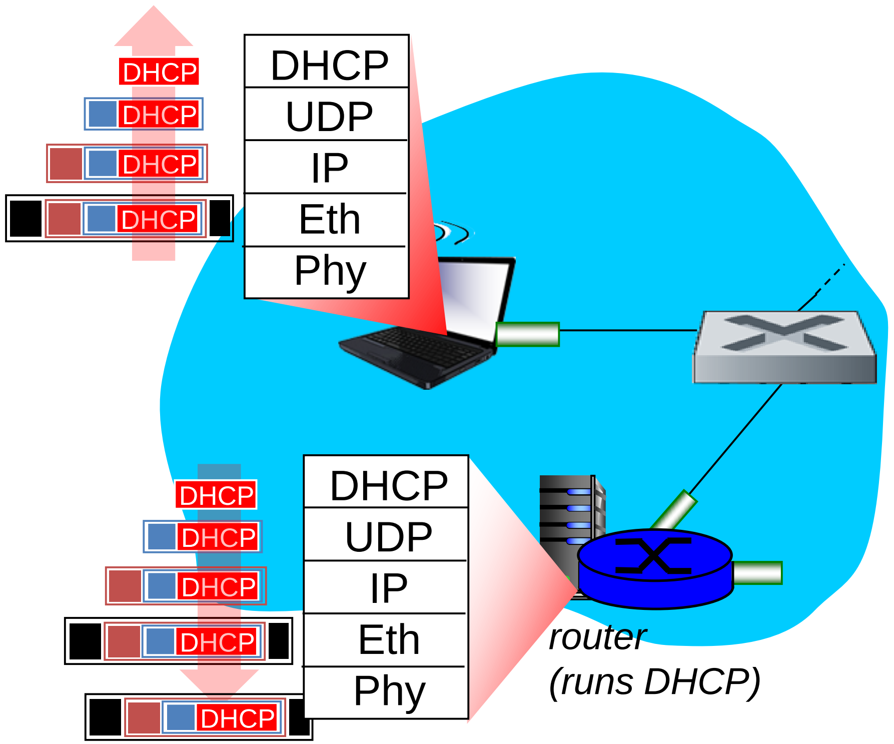
A day in the life… connecting to the Internet



connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*

- DHCP request *encapsulated* in *UDP*, encapsulated in *IP*, encapsulated in *802.3* Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

A day in the life… connecting to the Internet

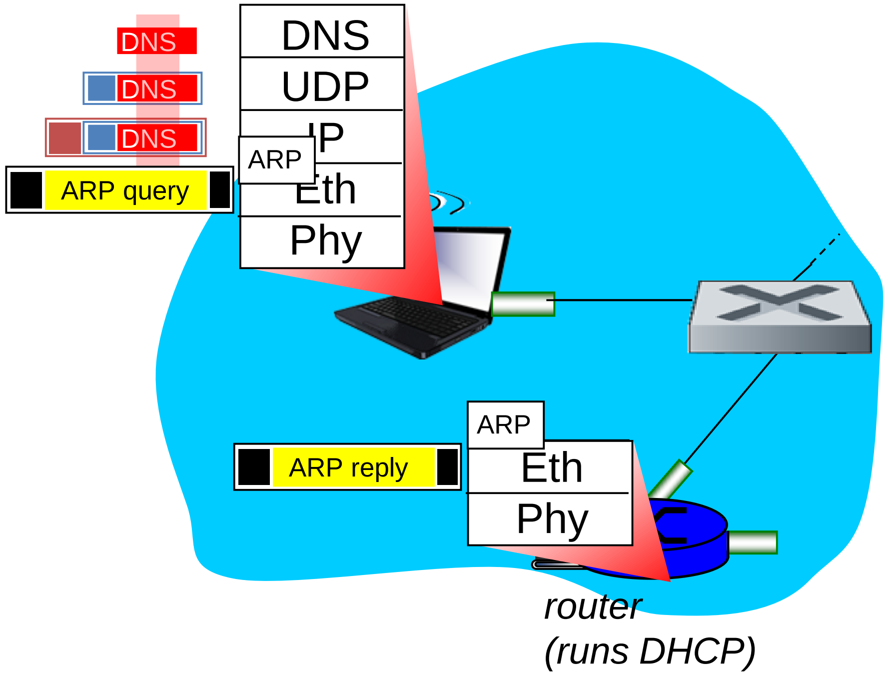


DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- encapsulation at DHCP server, frame forwarded (*switch learning*) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

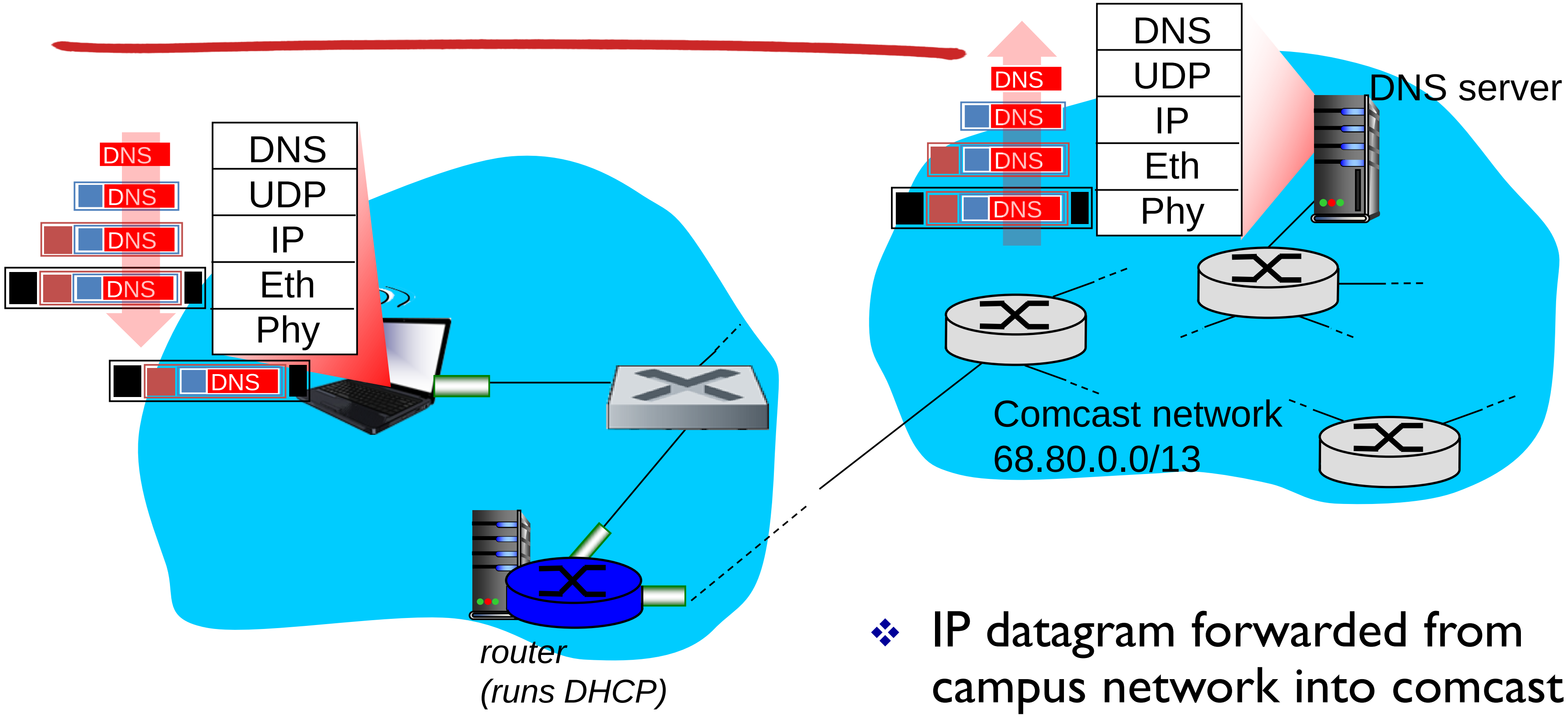
A day in the life… ARP (before DNS, before HTTP)



before sending *HTTP* request, need IP address of www.google.com: *DNS*

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: *ARP*
- *ARP query* broadcast, received by router, which replies with *ARP reply* giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

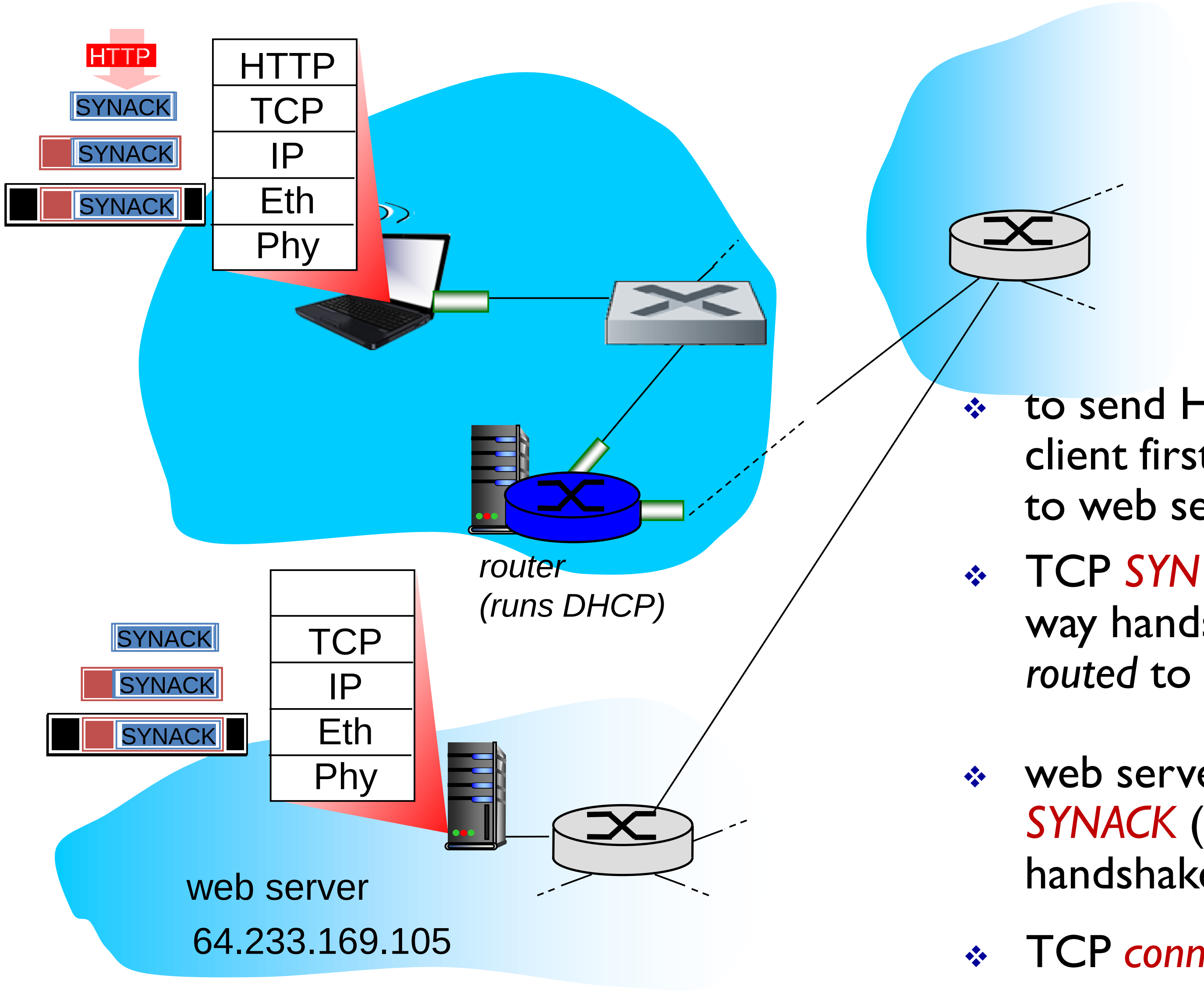
A day in the life… using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

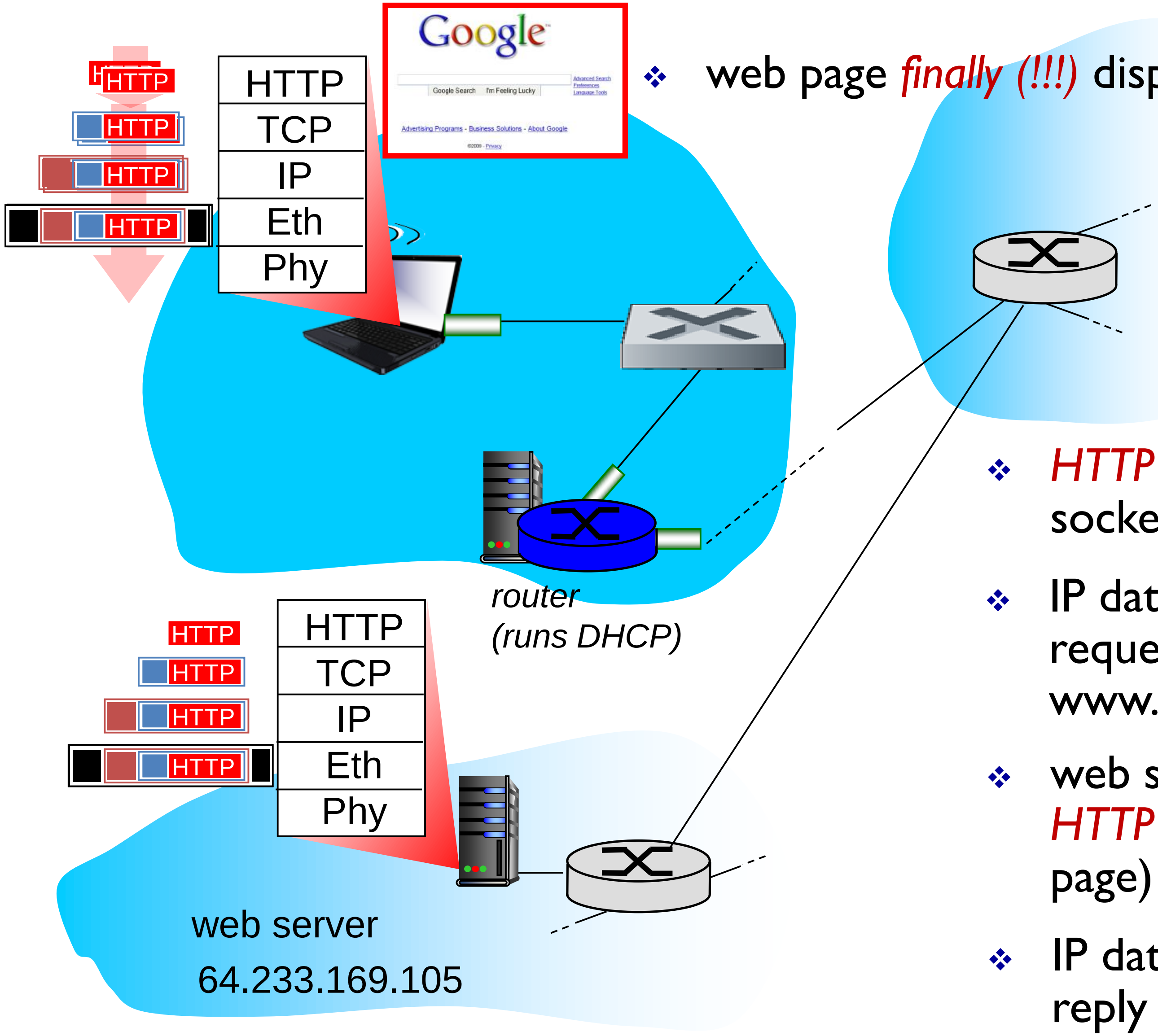
- IP datagram forwarded from campus network into comcast network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- demux' ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens *TCP socket* to web server
- TCP *SYN segment* (step 1 in 3-way handshake) *inter-domain routed* to web server
- web server responds with *TCP SYNACK* (step 2 in 3-way handshake)
- TCP *connection established!*

A day in the life… HTTP request/reply



- web page *finally (!!!)* displayed

- *HTTP request* sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with *HTTP reply* (containing web page)
- IP datagram containing HTTP reply routed back to client

Scaling a LAN network

Self-learning Ethernet switches work great at small scales, but buckle at larger scales

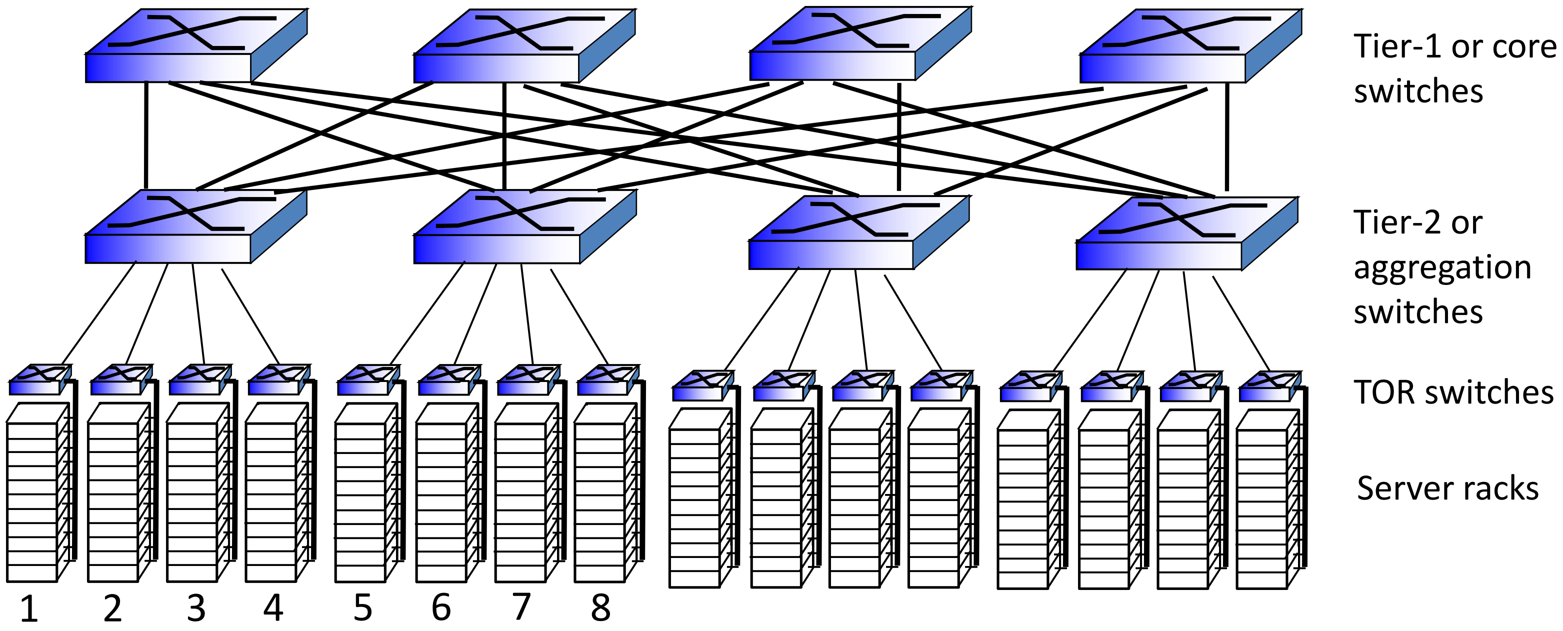
- Broadcast overhead of self-learning linear in the total number of interfaces
- Broadcast storms possible in non-tree topologies

Goals

- Scalability to a very large number of machines
- Isolation of unwanted traffic from unrelated subnets
- Ability to accommodate general types of workloads (Web, database, MapReduce, scientific computing, etc.)

Typical DC network components

- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



DC network design questions

Core and aggregation switches much faster than ToR switches
How much faster should core and aggregation switches need to be than ToR switches?
How many ports do core/aggregation switches need to support for a given number of ToR switch ports?
How many cables need to be run in total for a N machine datacenter?
What bisection bandwidth can be achieved?

Q: Why can't we just build a single BIG switch to interconnect all machines?

DC network topologies

Fat-tree (used ambiguously to mean Clos as well as a simple hierarchical design)
Clos family
Hypercube
Torus

Why simpler hierarchies not good enough?

High co
High ov
among e

width

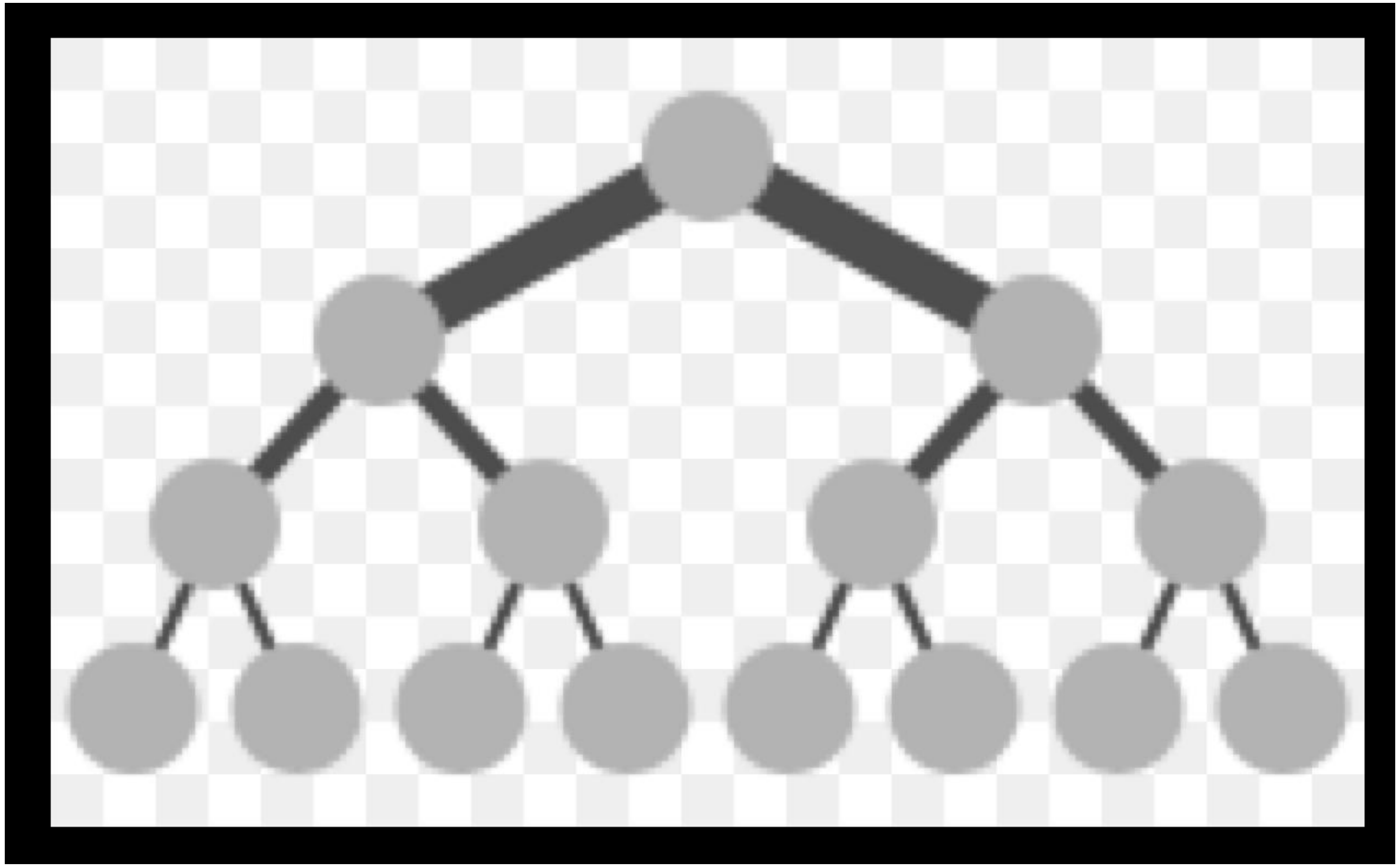
	Hierarchical design			Fat-tree		
Year	10 GigE	Hosts	Cost/ GigE	GigE	Hosts	Cost/ GigE
2002	28-port	4,480	\$25.3K	28-port	5,488	\$4.5K
2004	32-port	7,680	\$4.4K	48-port	27,648	\$1.6K
2006	64-port	10,240	\$2.1K	48-port	27,648	\$1.2K
2008	128-port	20,480	\$1.8K	48-port	27,648	\$0.3K

Table 1: The maximum possible cluster size with an oversubscription ratio of 1:1 for different years.

Figure 2: Current cost estimate vs. maximum possible number of hosts for different oversubscription ratios.

Fat tree topology

Core branches, i.e., those near the top of the hierarchy, are fatter or higher in capacity



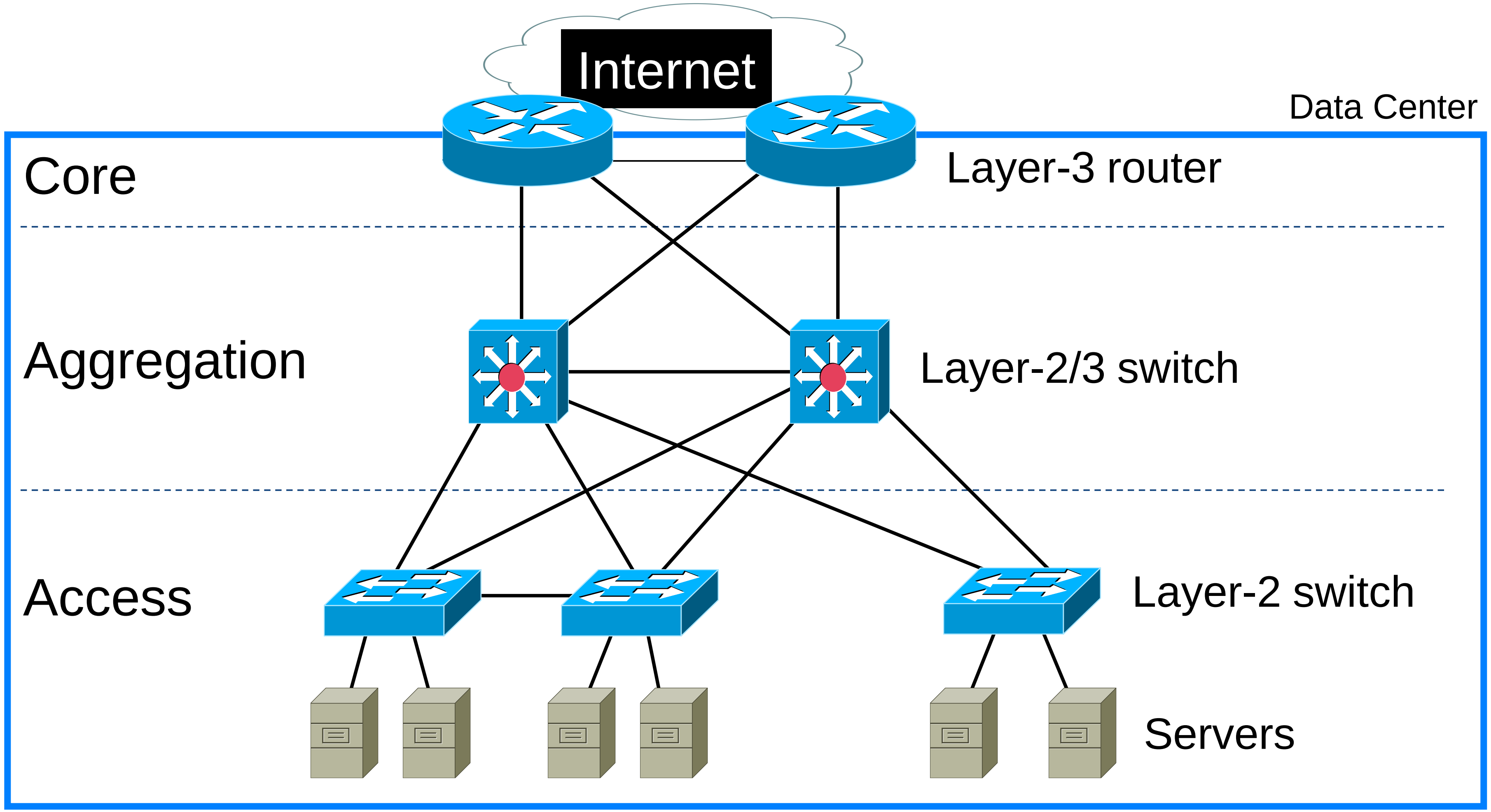
DCN Optimization: a Fat-tree Case Study

Background of Current DCN Architectures
Desired properties in a DC Architecture
Fat tree based solution
Evaluation
Conclusion

Background

- *Topology:*
  - 2 layers: 5K to 8K hosts
  - 3 layer: >25K hosts
  - Switches:
    - Leaves: have N GigE ports (48-288) + N 10 GigE uplinks to one or more layers of network elements
    - Higher levels: N 10 GigE ports (32-128)
- *Multi-path Routing:*
  - Ex. ECMP
    - without it, the largest cluster = 1,280 nodes
    - Performs static load splitting among flows
    - Lead to oversubscription for simple comm. patterns
    - Routing table entries grows multiplicatively with number of paths, cost ++, lookup latency ++

Common Data Center Topology



Background

- ***Oversubscription:***
  - Ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology
  - Lower the total cost of the design
  - Typical designs: factor of 2:5:1 (400 Mbps)to 8:1(125 Mbps)
- ***Cost:***
  - Edge: $7,000 for each 48-port GigE switch
  - Aggregation and core: $700,000 for 128-port 10GigE switches
  - Cabling costs are not considered!

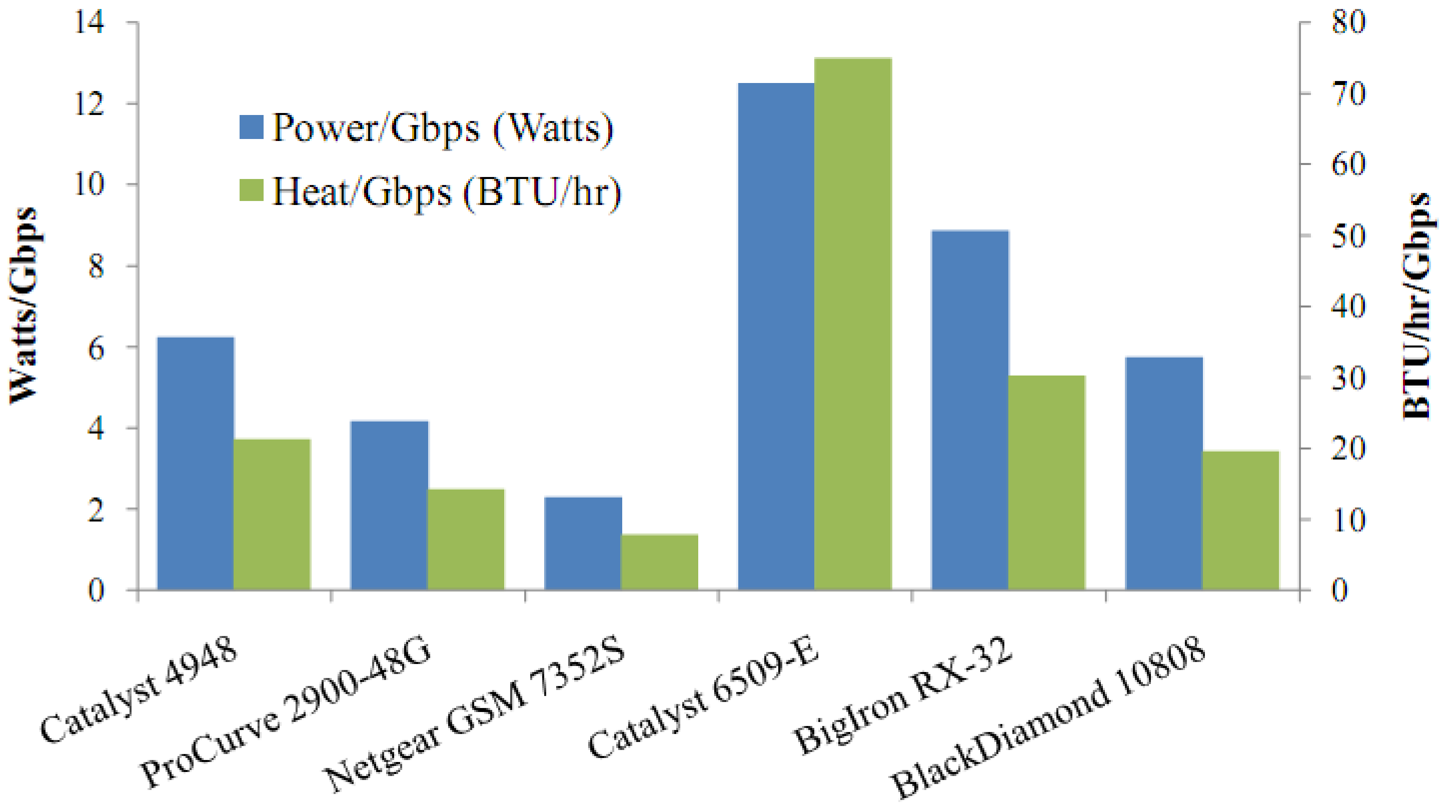
Current Data Center Network Architectures

- **Leverages specialized hardware and communication protocols, such as InfiniBand, Myrinet.**
  - These solutions can scale to clusters of thousands of nodes with high bandwidth
  - Expensive infrastructure, incompatible with TCP/IP applications
- **Leverages commodity Ethernet switches and routers to interconnect cluster machines**
  - Backwards compatible with existing infrastructures, low-cost
  - Aggregate cluster bandwidth scales poorly with cluster size, and achieving the highest levels of bandwidth incurs non-linear cost increase with cluster size

Problems with common DC Topology

Single point of failure
Over subscript of links higher up in the topology
- Trade off between cost and provisioning

Cost of maintaining switches



Properties of the solution

Backwards compatible with existing infrastructure

- No changes in application
- Support of layer 2 (Ethernet)

Cost effective

- Low power consumption & heat emission
- Cheap infrastructure

Allows host communication at line speed

Clos Networks/Fat-Trees

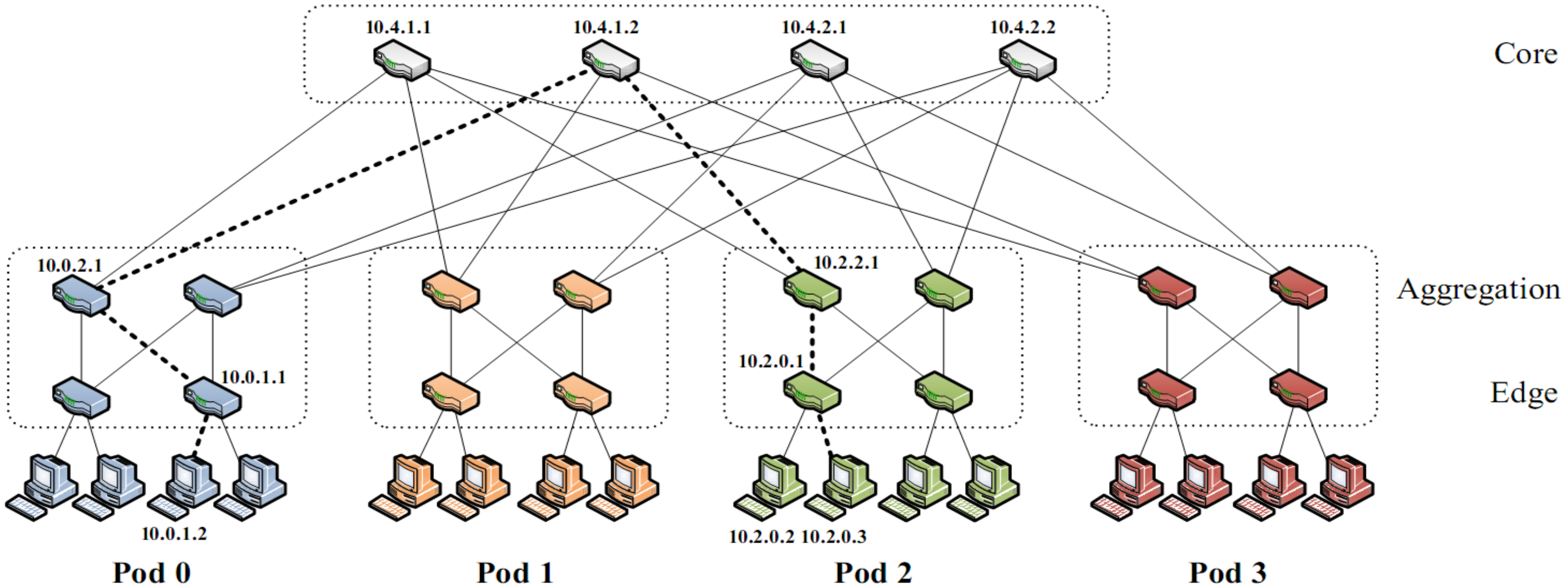
Adopt a special instance of a Clos topology

Similar trends in telephone switches led to designing a topology with high bandwidth by interconnecting smaller commodity switches.

FatTree-based DC Architecture

Inter-connect racks (of servers) using a fat-tree topology

K-ary fat tree: three-layer topology (edge, aggregation and core)
each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
each edge switch connects to k/2 servers & k/2 aggr. switches
each aggr. switch connects to k/2 edge & k/2 core switches
$(k/2)^2$ core switches: each connects to k pods



FatTree-based DC Architecture

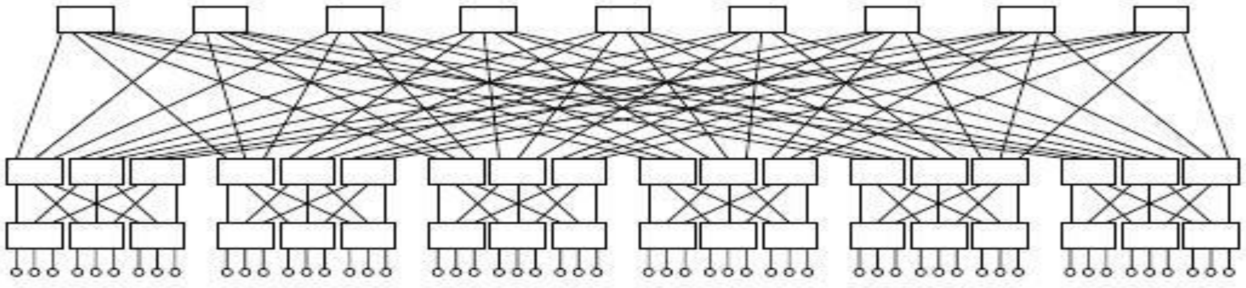
Why Fat-Tree?

- Fat tree has identical bandwidth at any bisections
- Each layer has the same aggregated bandwidth

Can be built using cheap devices with uniform capacity

- Each port supports same speed as end host
- All devices can transmit at line speed if packets are distributed uniform along available paths

Great scalability: k-port switch supports $k^3/4$ servers



FatTree Topology is great, But…

- Does using fat-tree topology to inter-connect racks of servers in itself sufficient?
- What routing protocols should we run on these switches?
- Layer 2 switch algorithm: data plane flooding!
- Layer 3 IP routing:
  - shortest path IP routing will typically use only one path despite the path diversity in the topology
  - if using equal-cost multi-path routing at each switch independently and blindly, packet re-ordering may occur; further load may not necessarily be well-balanced
  - Aside: control plane flooding!

Problems with Fat-tree

- Layer 3 will only use one of the existing equal cost paths
  - Bottlenecks up and down the fat-tree
    - Simple extension to IP forwarding
- Packet re-ordering occurs if layer 3 blindly takes advantage of path diversity ; further load may not necessarily be well-balanced
- Wiring complexity in large networks
  - Packing and placement technique

FatTree Modified

- Enforce a special (IP) addressing scheme in DC
  - unused.PodNumber.switchnumber.Endhost
  - Allows host attached to same switch to route only through switch
  - Allows inter-pod traffic to stay within pod

FatTree Modified

Use two level look-ups to distribute traffic and maintain packet ordering

- First level is prefix lookup
  - used to route down the topology to servers
- Second level is a suffix lookup
  - used to route up towards core
  - maintain packet ordering by using same ports for same server
  - Diffuses and spreads out traffic

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

FatTree Modified

Diffusion Optimizations (routing options)

1. **Flow classification**, Denote a *flow* as a sequence of packets; pod switches forward subsequent packets of the same flow to same outgoing port. And periodically reassign a minimal number of output ports
   - Eliminates local congestion
   - Assign traffic to ports on a per-flow basis instead of a per-host basis, Ensure fair distribution on flows

FatTree Modified

2. Flow scheduling, Pay attention to routing ***large flows***, edge switc
detect any outgoing flow whose size grows above a predefined thresho
and then send notification to a ***central scheduler***. The central schedul
tries to assign non-conflicting paths for these large flows.

- Eliminates global congestion
- Prevent long lived flows from sharing the same links
- Assign long lived flows to different links

Fault Tolerance

In this scheme, each switch in the network maintains a BFD (Bidirectional Forwarding Detection) session with each of its neighbors to determine when a link or neighboring switch fails

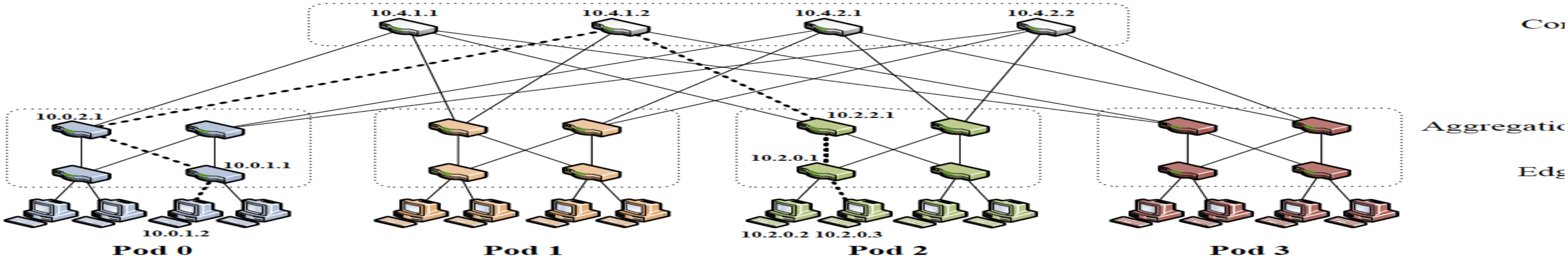
- Failure between upper layer and core switches
  - Outgoing inter-pod traffic, local routing table marks the affected link as unavailable and chooses another core switch
  - Incoming inter-pod traffic, core switch broadcasts a tag to upper switches directly connected signifying its inability to carry traffic to that entire pod, then upper switches avoid that core switch when assigning flows destined to that pod

Fault Tolerance

Failure between lower and upper layer switches

- Outgoing inter- and intra pod traffic from lower-layer,
  - the local flow classifier sets the cost to infinity and does not assign it any new flows, chooses another upper layer switch
- Intra-pod traffic using upper layer switch as intermediary
  - Switch broadcasts a tag notifying all lower level switches, these would check when assigning new flows and avoid it
- Inter-pod traffic coming into upper layer switch
  - Tag to all its core switches signifying its ability to carry traffic, core switches mirror this tag to all upper layer switches, then upper switches avoid affected core switch when assigning new flaws

Example: uniform Clos topology [UCSD]

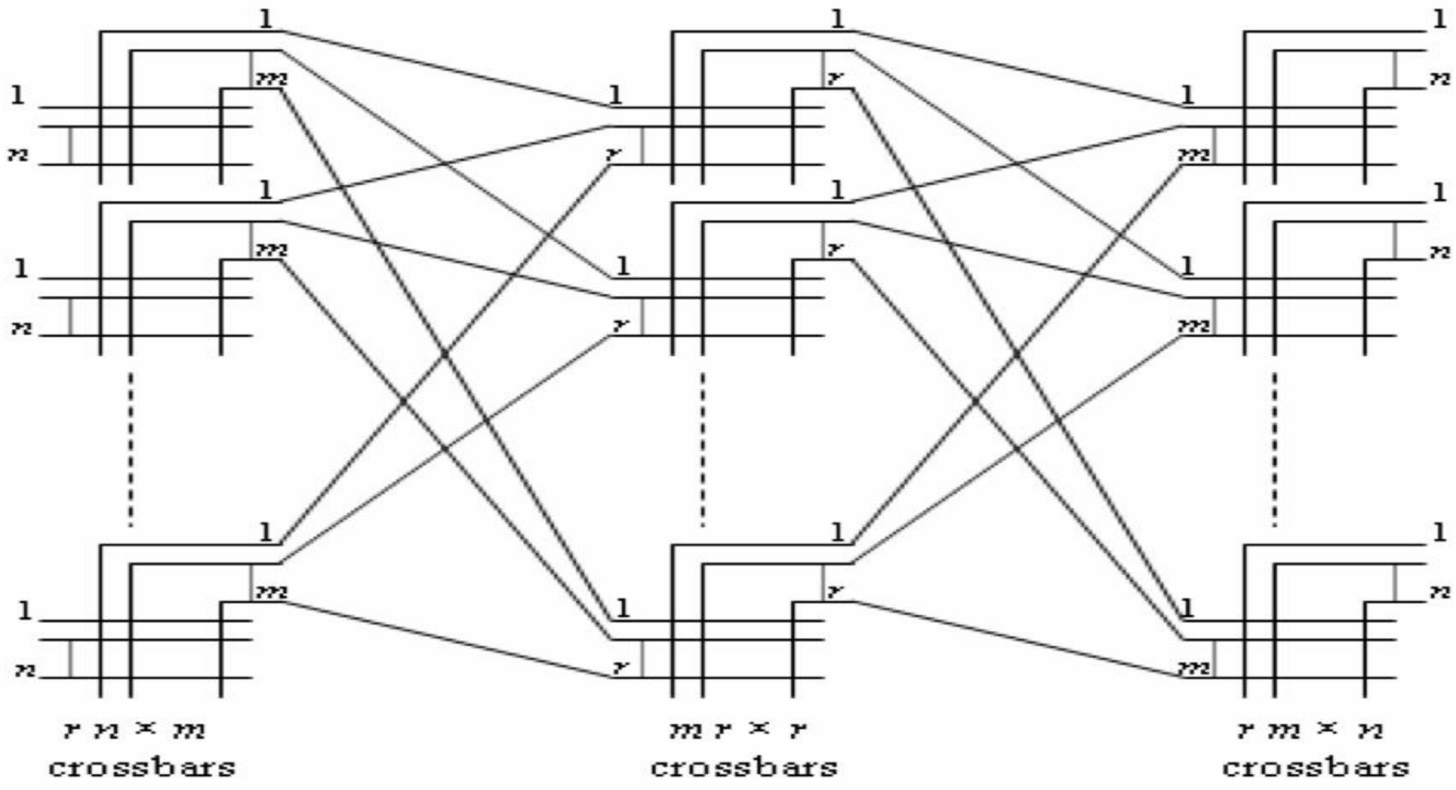


ure 3: Simple fat-tree topology. Using the two-level routing tables described in Section 3.3, packets from source 10.0.
tination 10.2.0.3 would take the dashed path.

[UCSD] A Scalable Commodity Data Center Network Architecture

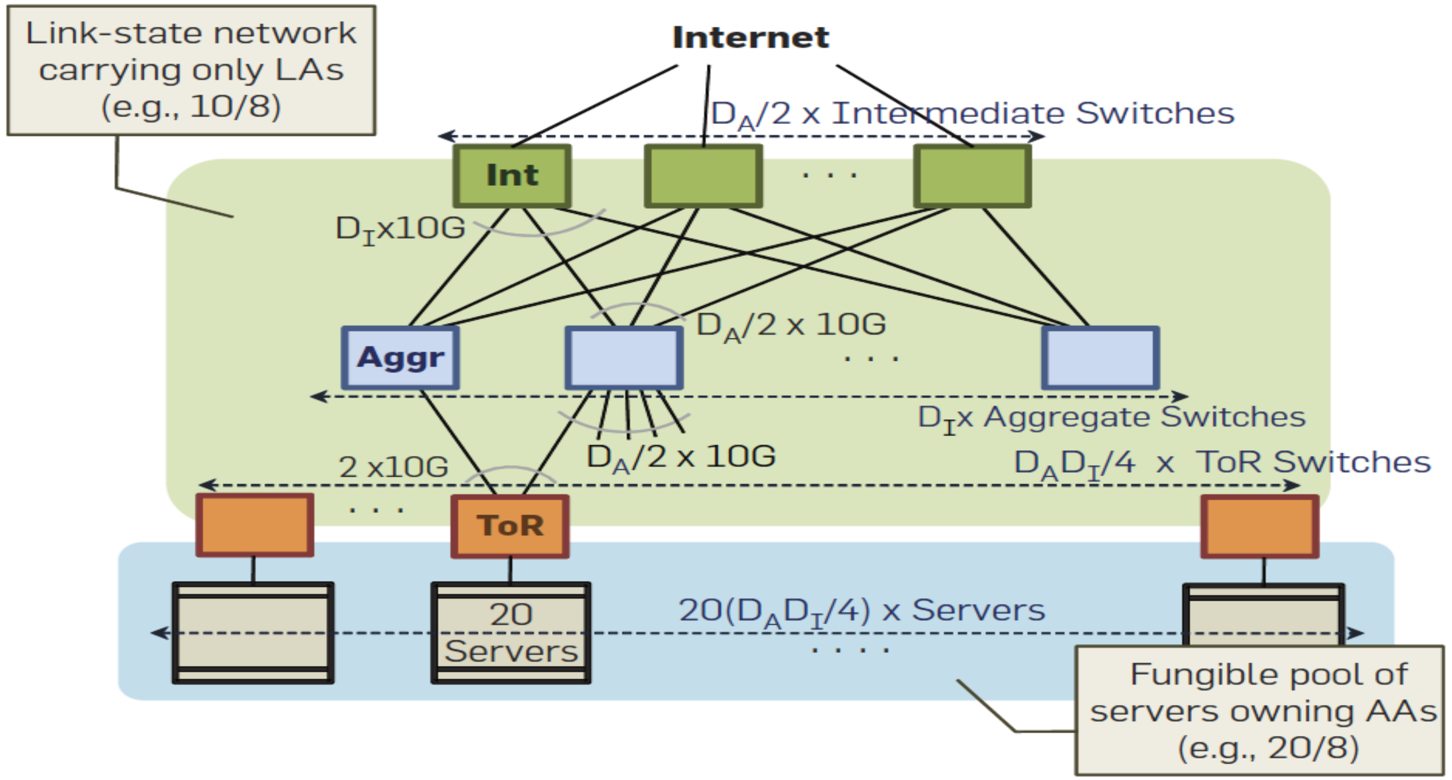
Clos family

Ingress, intermediate, and egress switches where each stage's links form a bipartite graph



VL2: Clos case study (Microsoft)

Figure 4. An example Clos network between aggregation and intermediate switches provides a richly connected backbone well suited for VLB. The network is built with two separate address families—topologically significant locator-specific addresses (LAs) and flat application-specific addresses (AAs).



VL2: Addressing and routing

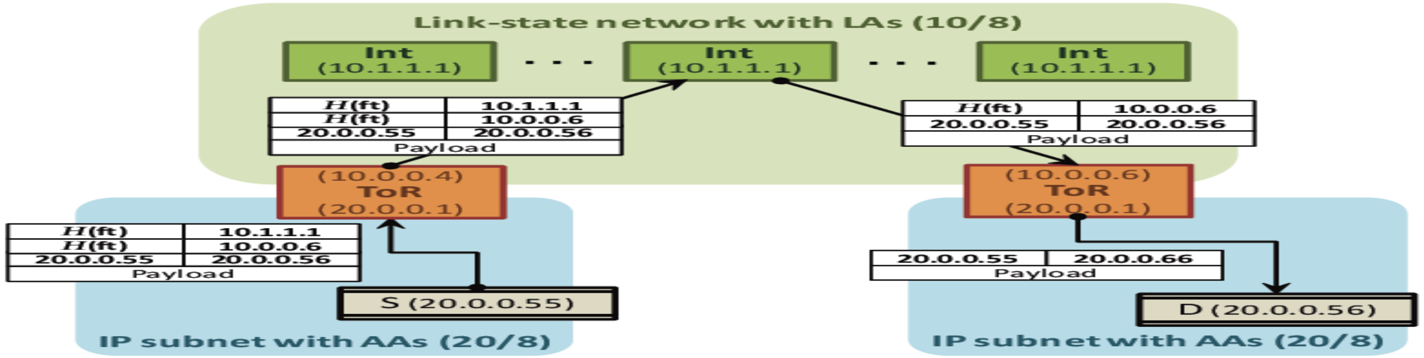


Figure 6: VLB in an example VL2 network. Sender S sends packets to destination D via a randomly-chosen intermediate switch using IP-in-IP encapsulation. AAs are from 20/8, and LAs are from 10/8. H(ft) denotes a hash of the five tuple.

Valiant load balancing

Randomization for efficient, load-balanced routing
[VLB]

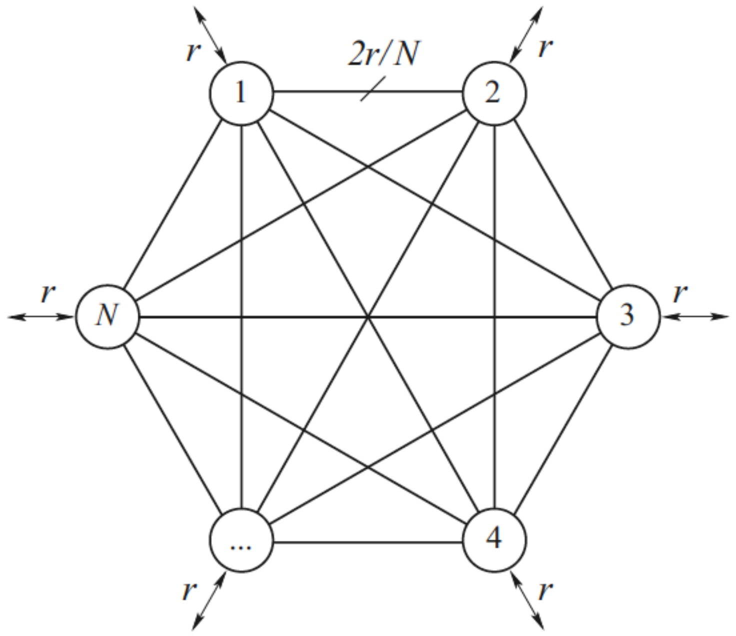


Fig. 2.1 VLB in a network of N identical nodes each having capacity r. A full mesh of logical links of capacity $2r/N$ connect the nodes

[VLB] Valiant Load-Balancing: Building Networks That Can Support All Traffic Matrices

VL2: Directory for AA<->LA mappings

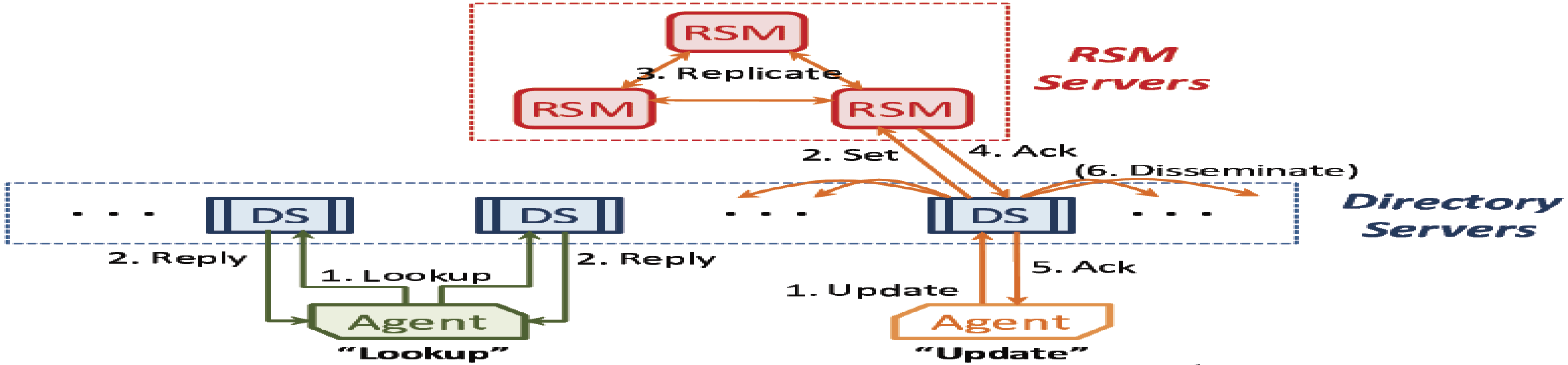


Figure 7: VL2 Directory System Architecture

BCube: relies on more server ports

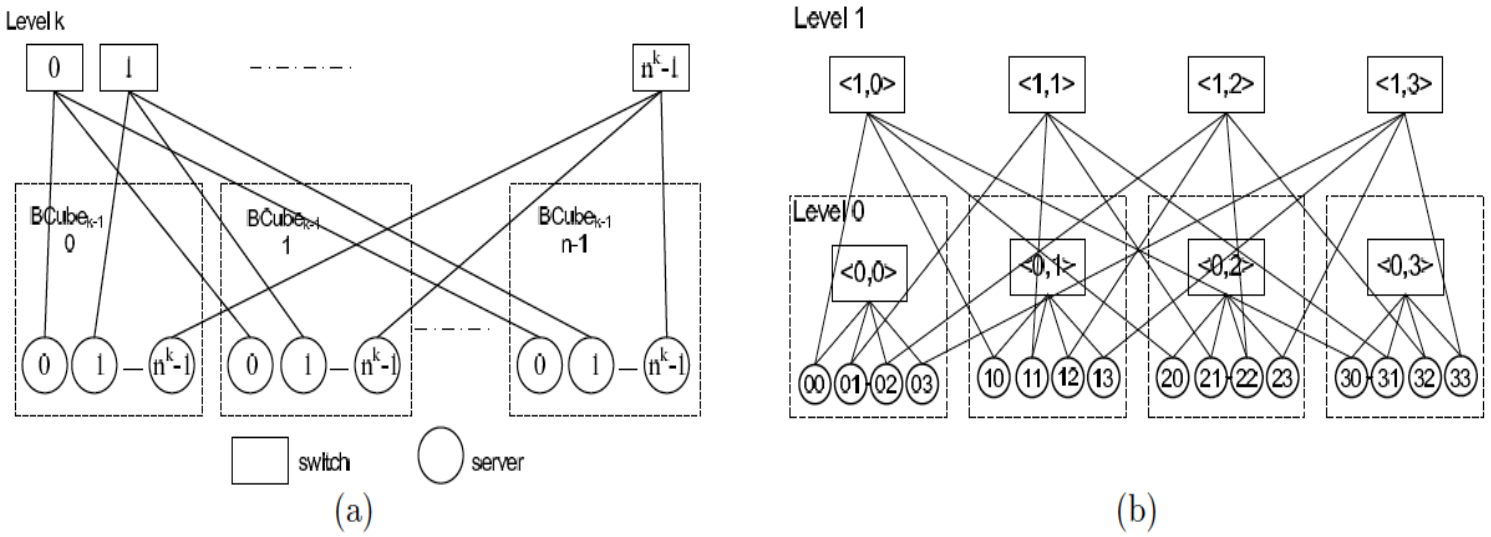
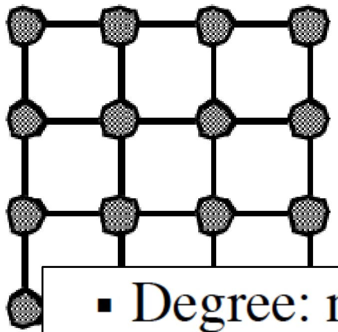
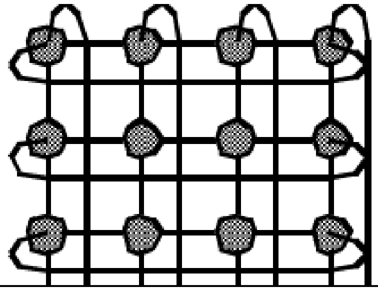
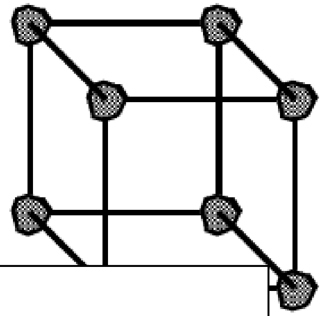


Figure 1: (a)BCube is a leveled structure. A BCube$_k$ is constructed from n BCube$_{k-1}$ and n^k n-port switches. (b) A BCube$_1$ with $n = 4$. In this BCube$_1$ network, each server has two ports.

Other topologies from “supercomputing”

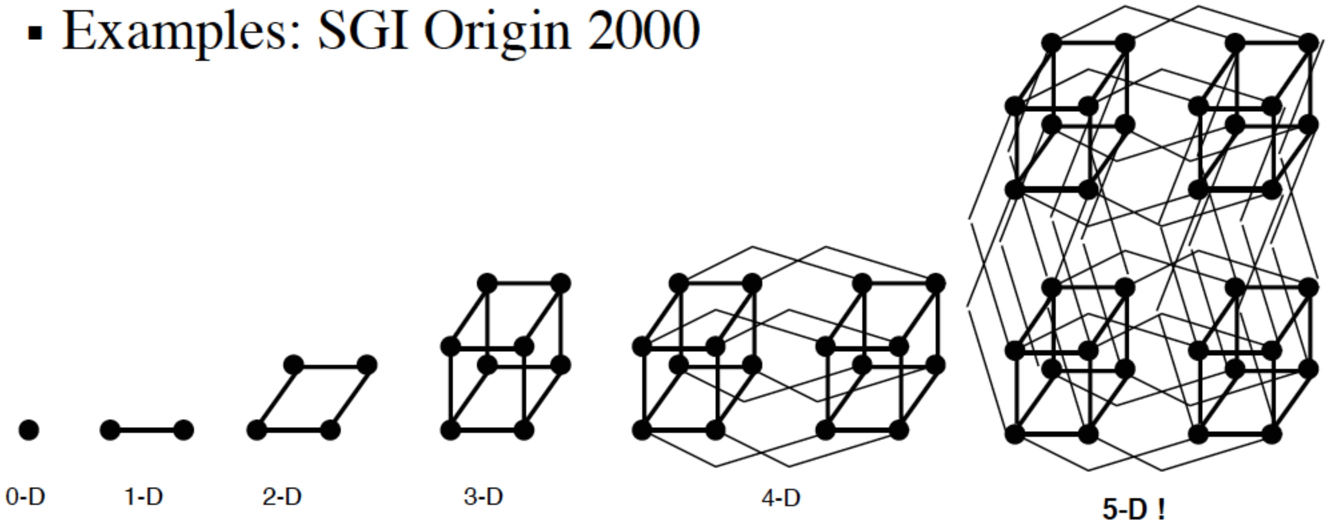






- Degree: n = logN
- Distance O(logN) Hops
- Good bisection BW
- Examples: SGI Origin 2000

Hypercube



Optic

Optical
switche
Optical
But opt
MEMS (
Optical ;
(elepha

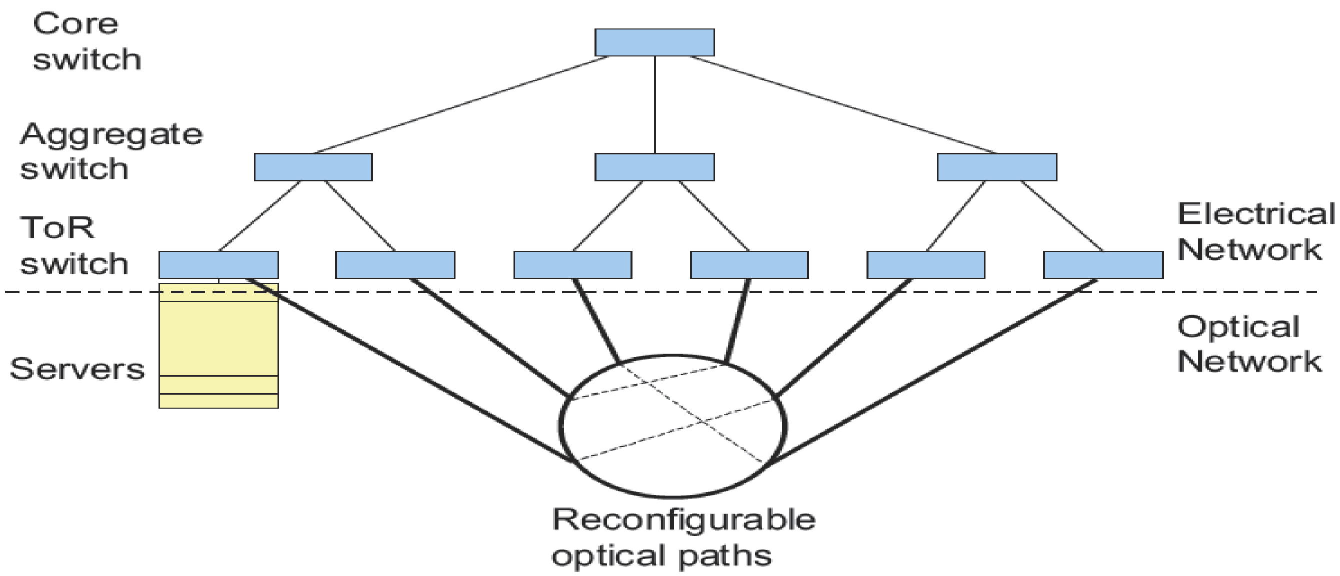


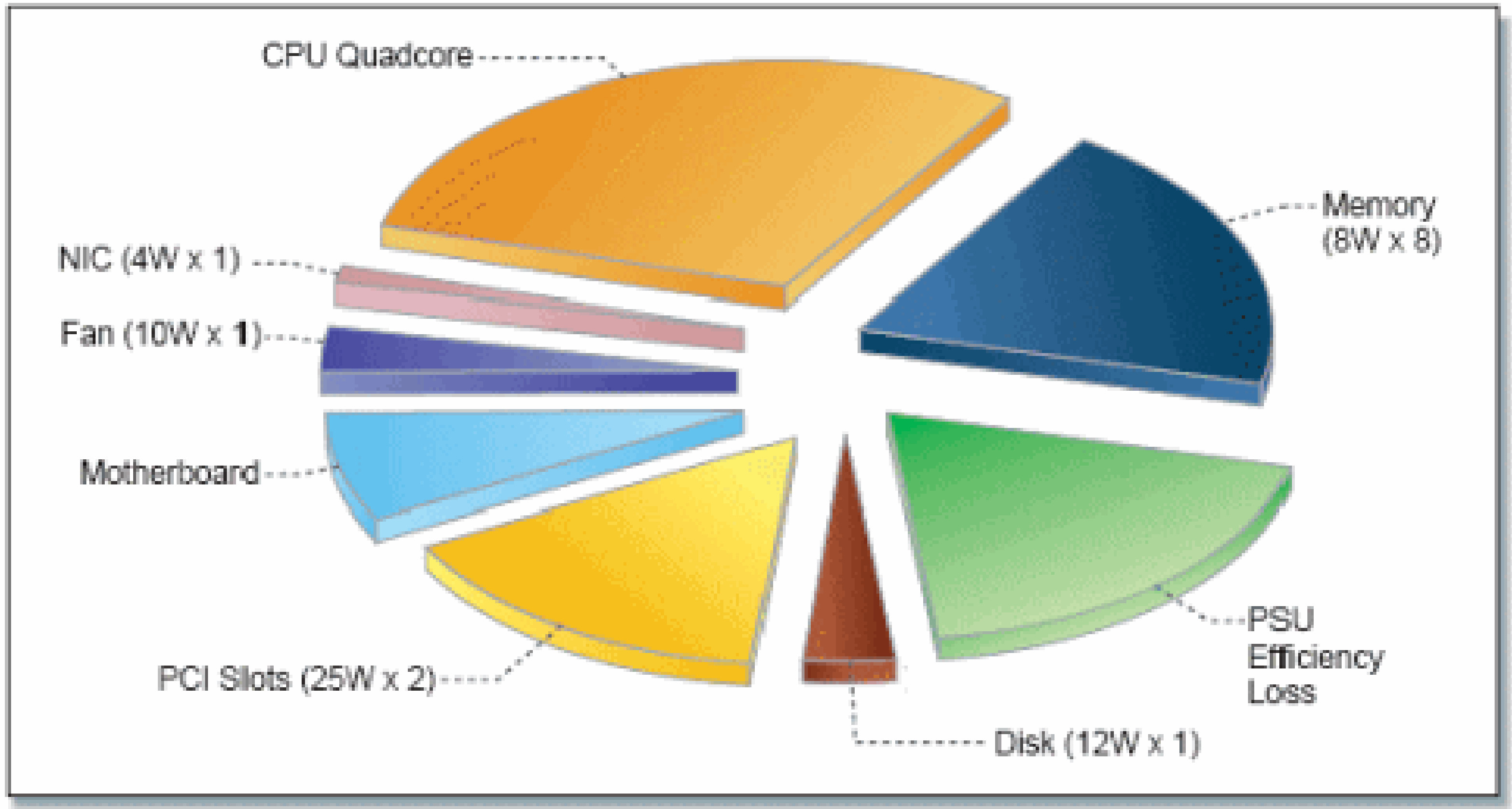
Figure 1: HyPaC network architecture

	System requirements
Control plane	1. Estimating cross-rack traffic demands 2. Managing circuit configuration
Data plane	1. De-multiplexing traffic in dual-path network 2. Maximizing the utilization of circuits when available (optimization)

Table 1: Fundamental requirements of HyPaC architecture.

Energy usage numbers

Typical US household: ~1000kWh per month or ~30kW
Typical d
Typical 1 thousand
W for hi
Switches



Switch power consumption

Generally sm topologies

Topology	Server count	Switches	Switch pow. / Server pow.
1 Gbps server link			
FatTree	3456	Cisco 2224TP (80W, 720 count)	0.17
VL2	2880	Cisco 2224TP (80W, 144 count), Cisco Nexus 5548P (390 W, 24 count)	0.08
10 Gbps server link			
FatTree	3456	Cisco Nexus 5548P (390 W, 360 count)	0.40
VL2	2304	Cisco 6001 (750W, 48 count), Cisco 6004 (2900W, 6 count)	0.23

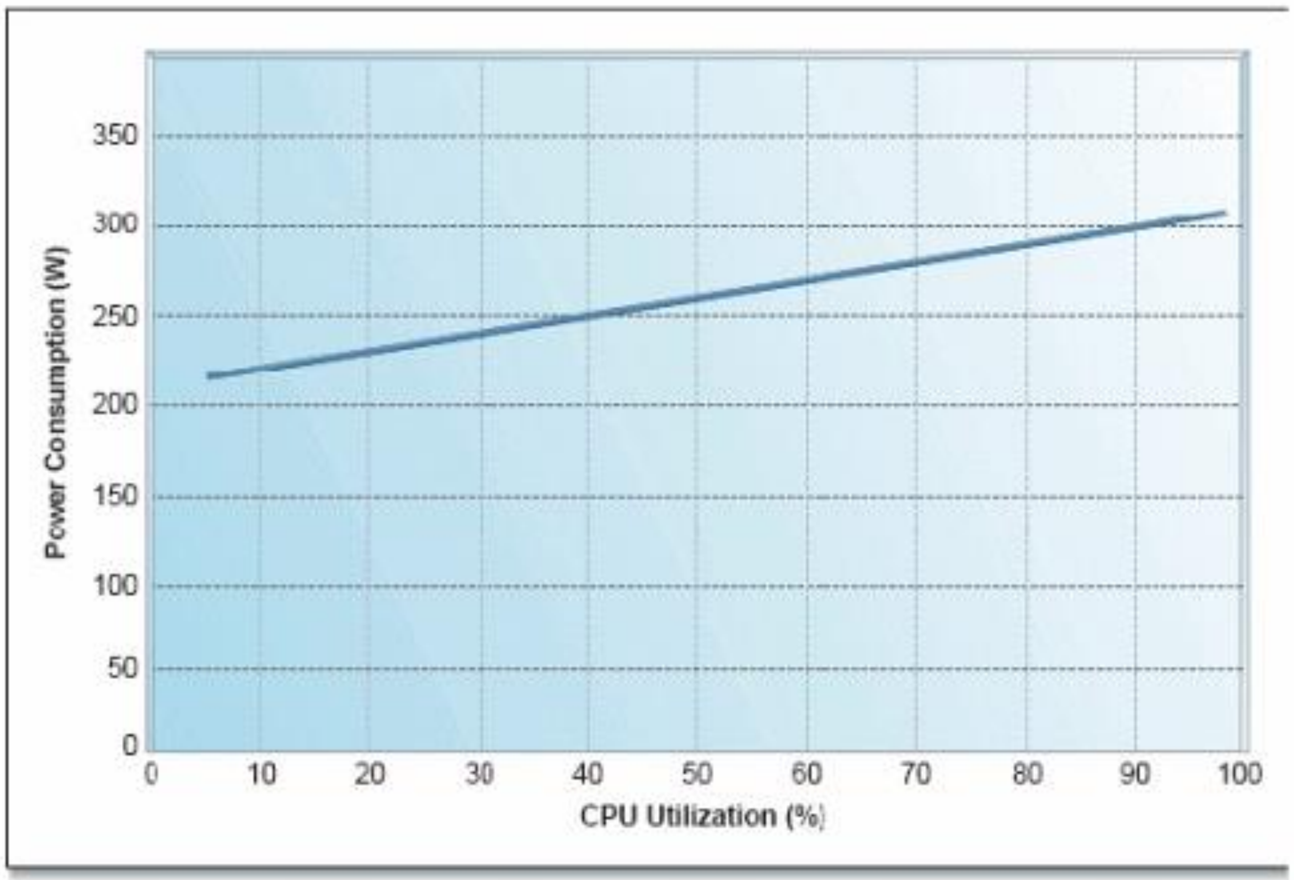
Figure 1 *Ratio of network to server energy use at typical operating conditions. Switch power use data from Cisco [13]. Server power use of Acer Altos T350 F2 at a load of 30% is 98.5 W [46].*

Techniques to reduce energy

Dynamic voltag   ces CV^2f by reducing voltag

Generally not power   reased usage

Shutting down   network: widely studied



Let' s take a breath and take home

journey down protocol stack *complete* (except PHY)
solid understanding of networking principles, practice
….. could stop here …. but *lots* of interesting topics in data centers!

- wireless
- multimedia
- security
- network management