Final Review

Lecture 16

Material List (after Mid-term 1)

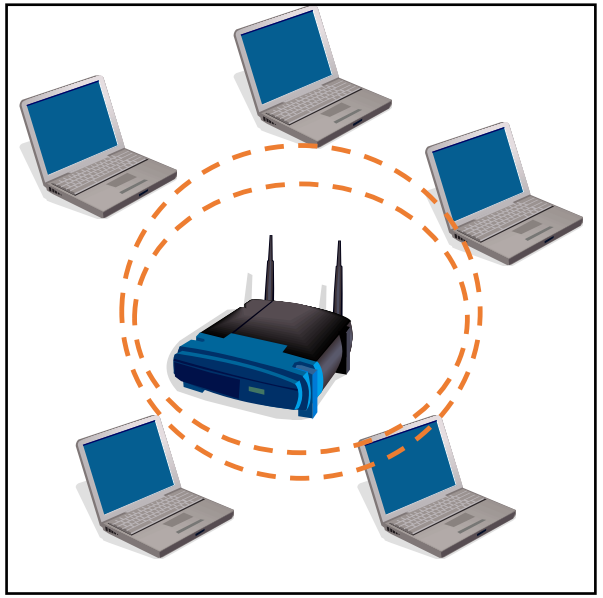
- Lecture 11:Wireless networking
- Lecture 12: Wireless Routing & TCP
- Lecture 13: Datacenter Networking
- Lecture 14: Data-driven Design (TCP Remi)
- Lecture 15: P2P Networks

Wireless

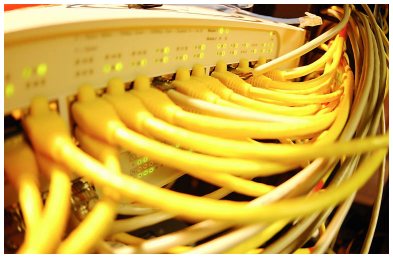
Wireless Networking

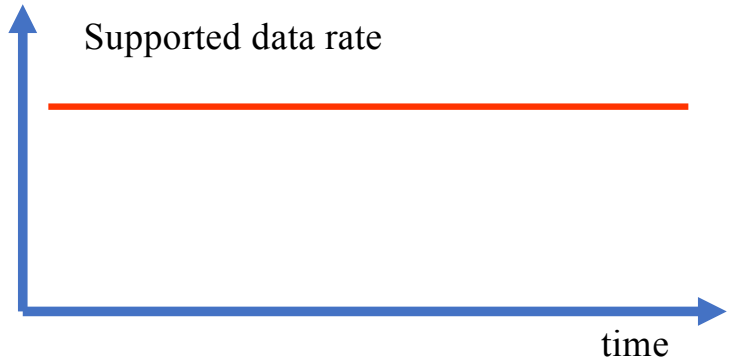
- Changes in Physical Medium

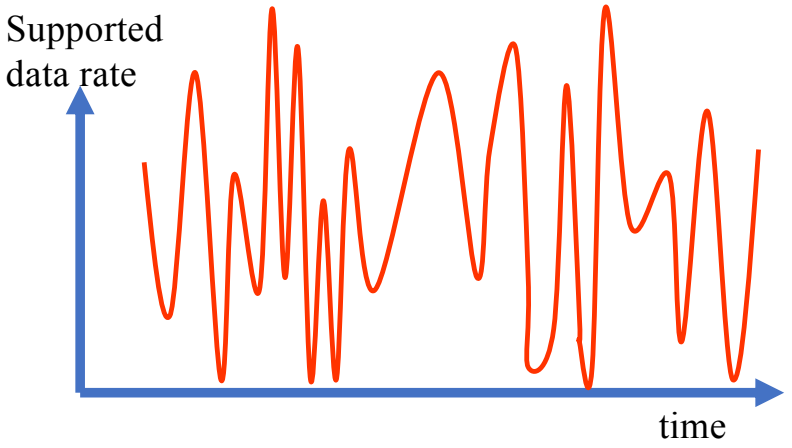
- Key artifacts
  - Link fluctuations
  - Interference



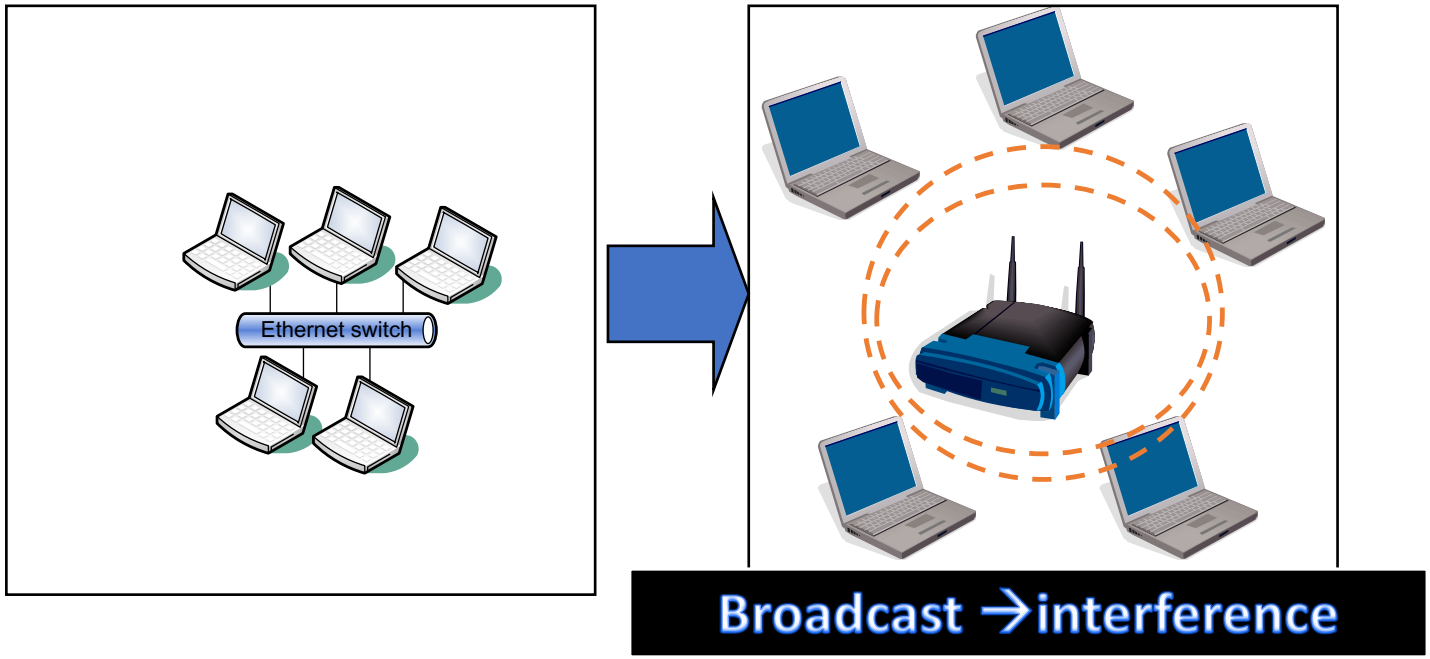
The Difference: # 1







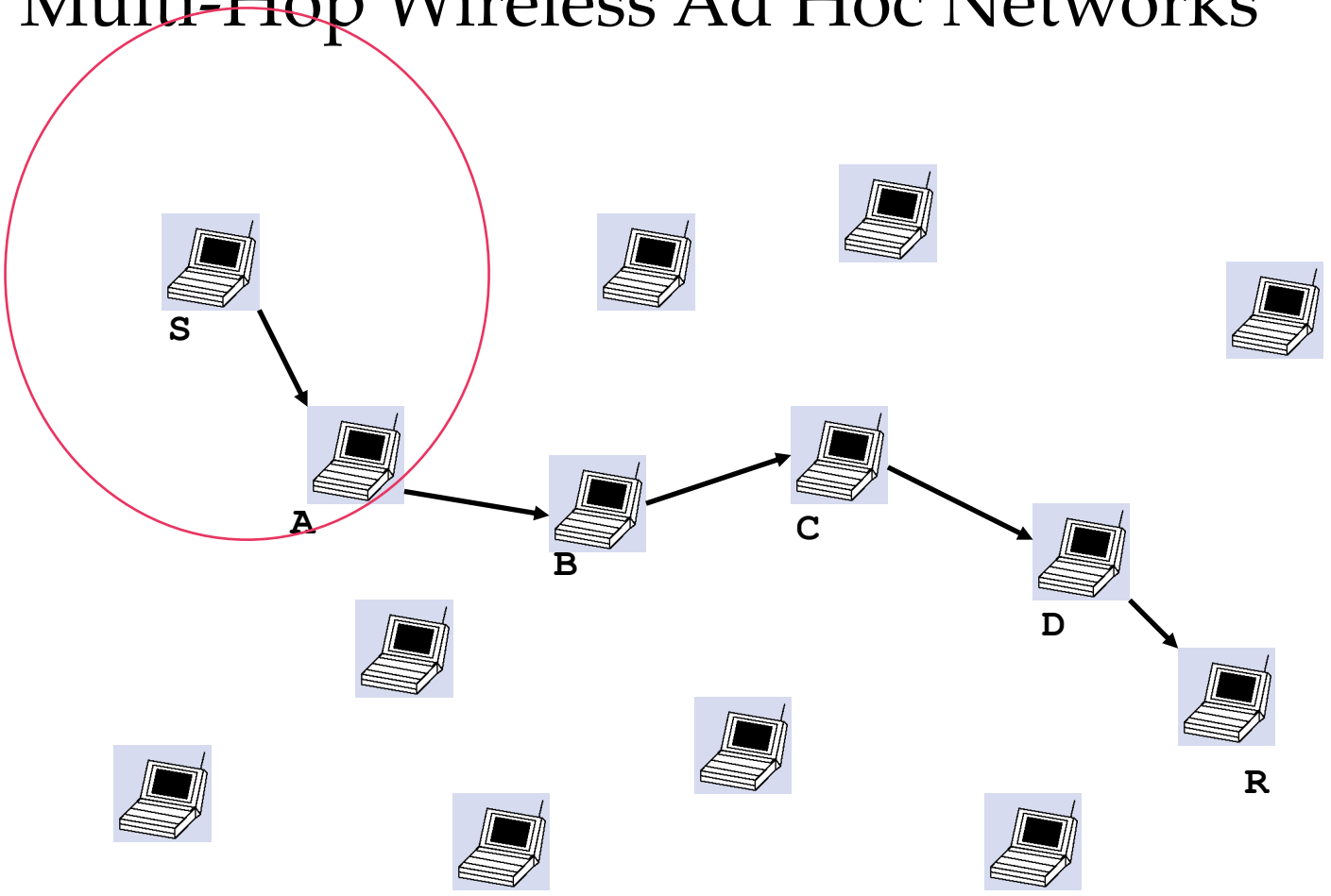
The Difference #2



Wireless Networking Solutions

- LINK Layer: 802.11 CSMA/CA
  - The problems of Hidden terminal & Exposed terminal
  - What triggered them (roles of transmitter & receiver)
  - How to address them
  - What is CSMA, what is CA
  - How to avoid interference using channel assignment
- Networking Layer: Multi-hop routing
  - Why wireless multi-hop routing is harder/less efficient than wired multi-hop routing
  - What really happens (RTS/CTS/transmission)
  - TCP makes it even harder, why?

Multi-Hop Wireless Ad Hoc Networks



Courtesy of Tianbo Kuang and Carey Williamson University of Calgary)

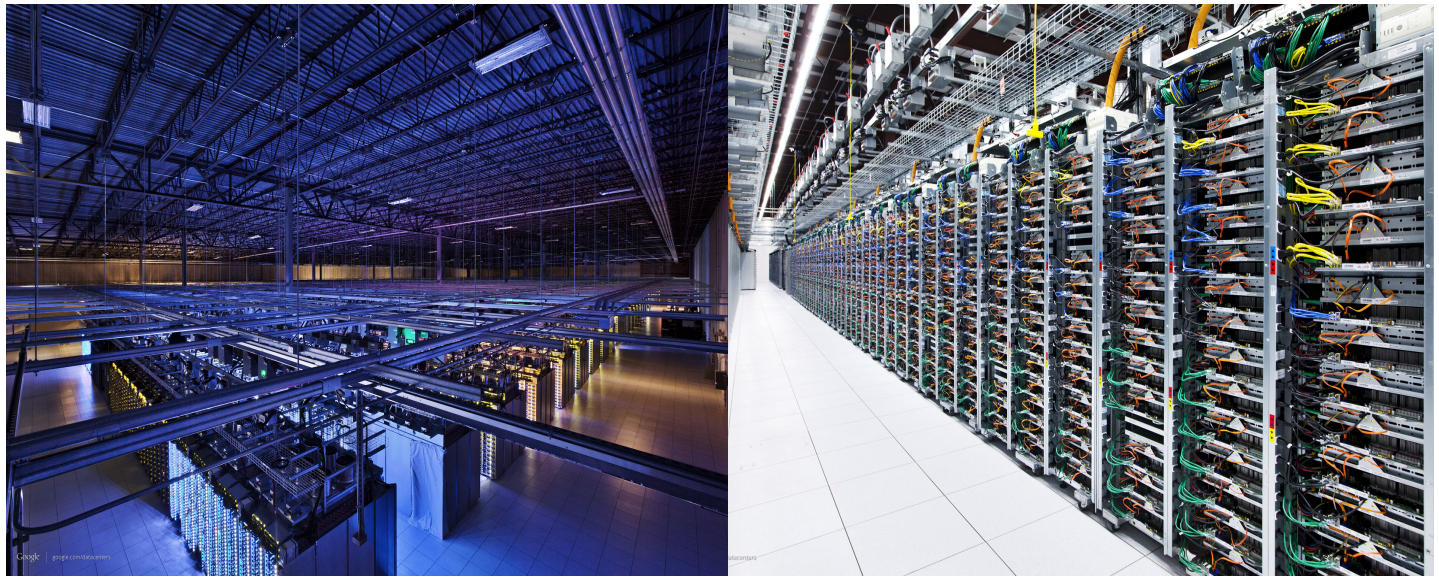
Wireless Networking Solutions

- Transport Layer: Wireless TCP
  - Running TCP over wireless.. What is the problem, how does it differ from running TCP over just wired networks
  - 2 groups of strategies to address wireless TCP
    - Masking
    - Notification
    - Where are they implemented? Sender, Receiver, intermediate nodes?
    - Good and bad of these approaches

Data Center

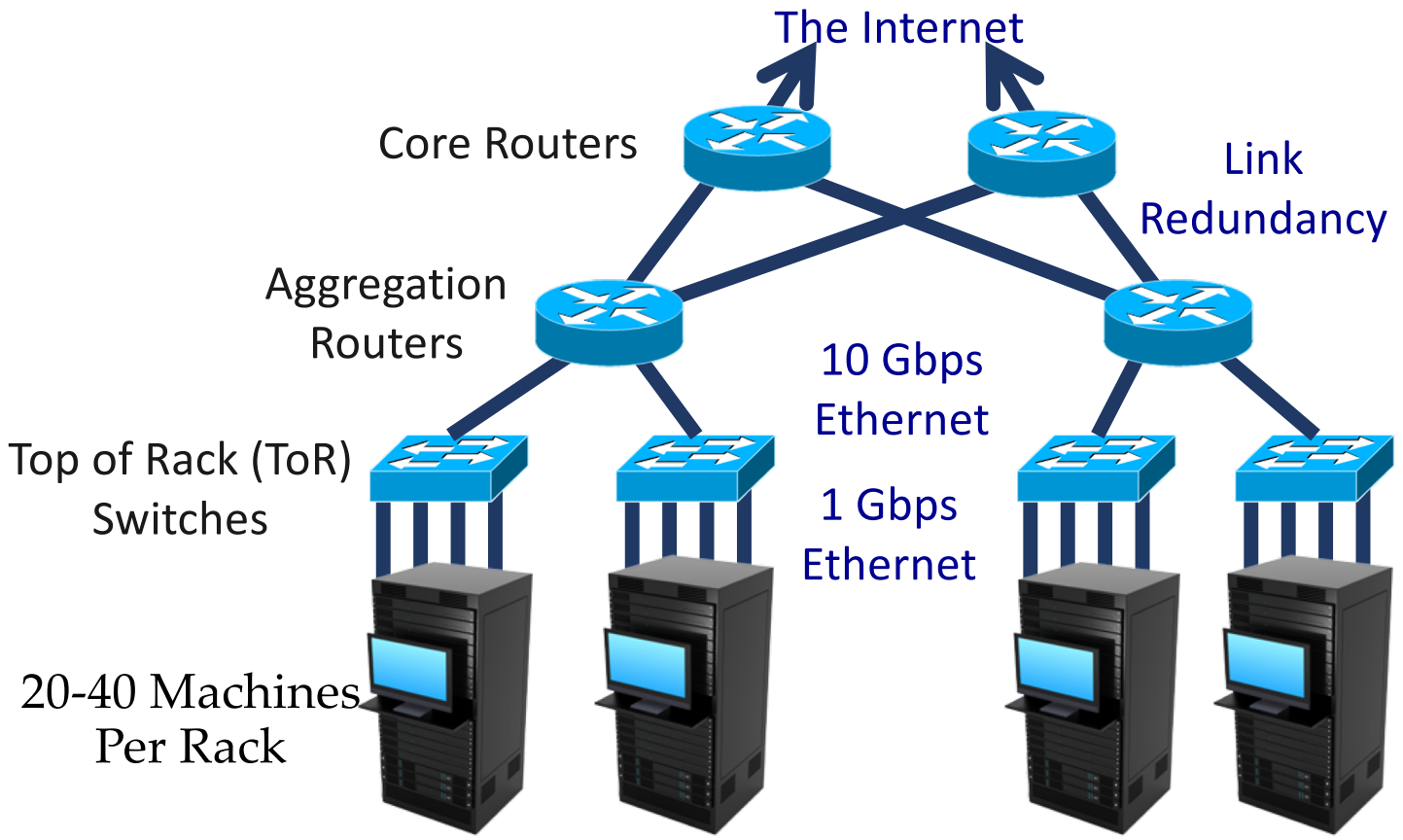
Today's Data Centers

- Warehouse of servers

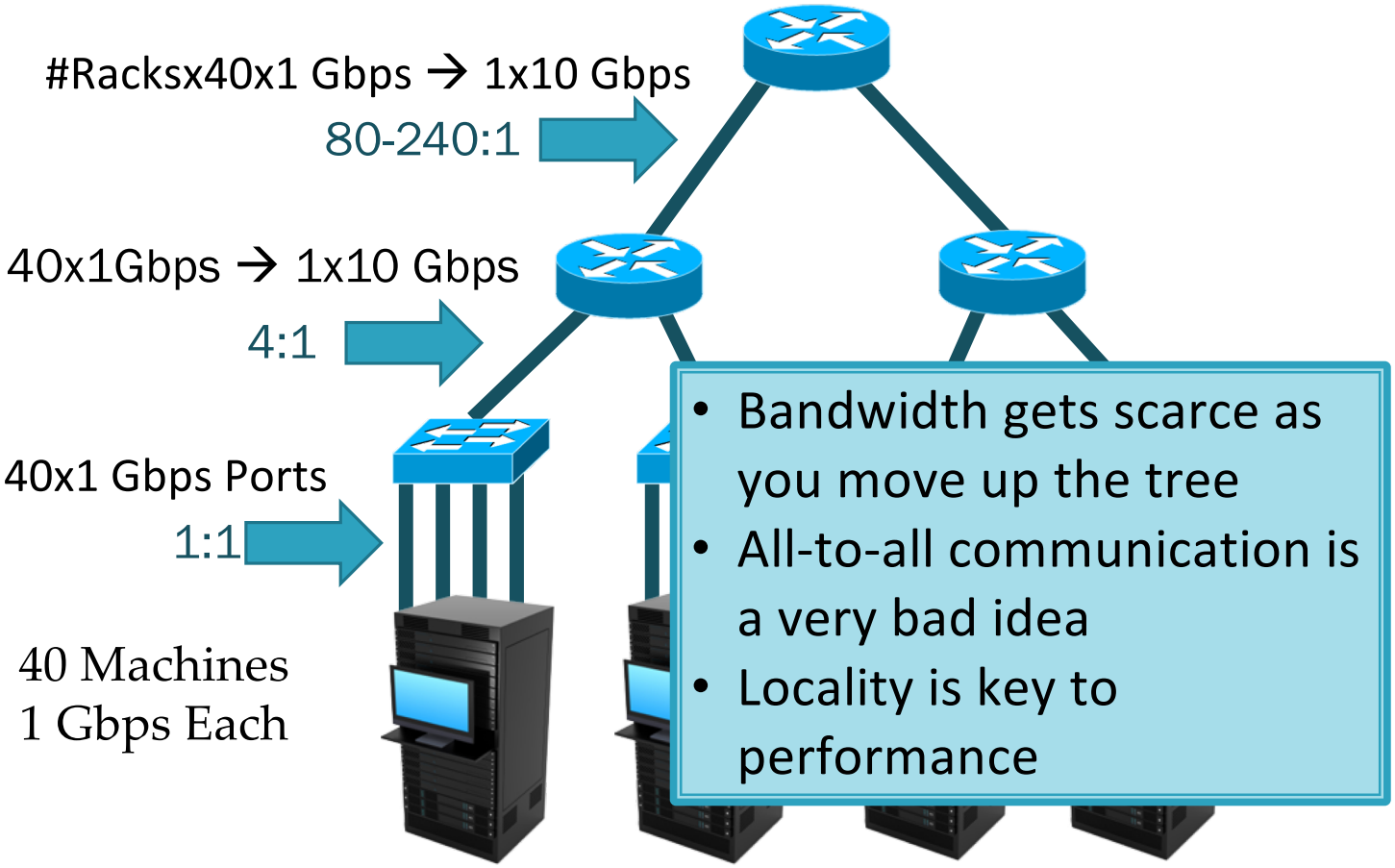


https://www.youtube.com/watch?v=zDAYZU4A3w0

Typical Data Center Topology



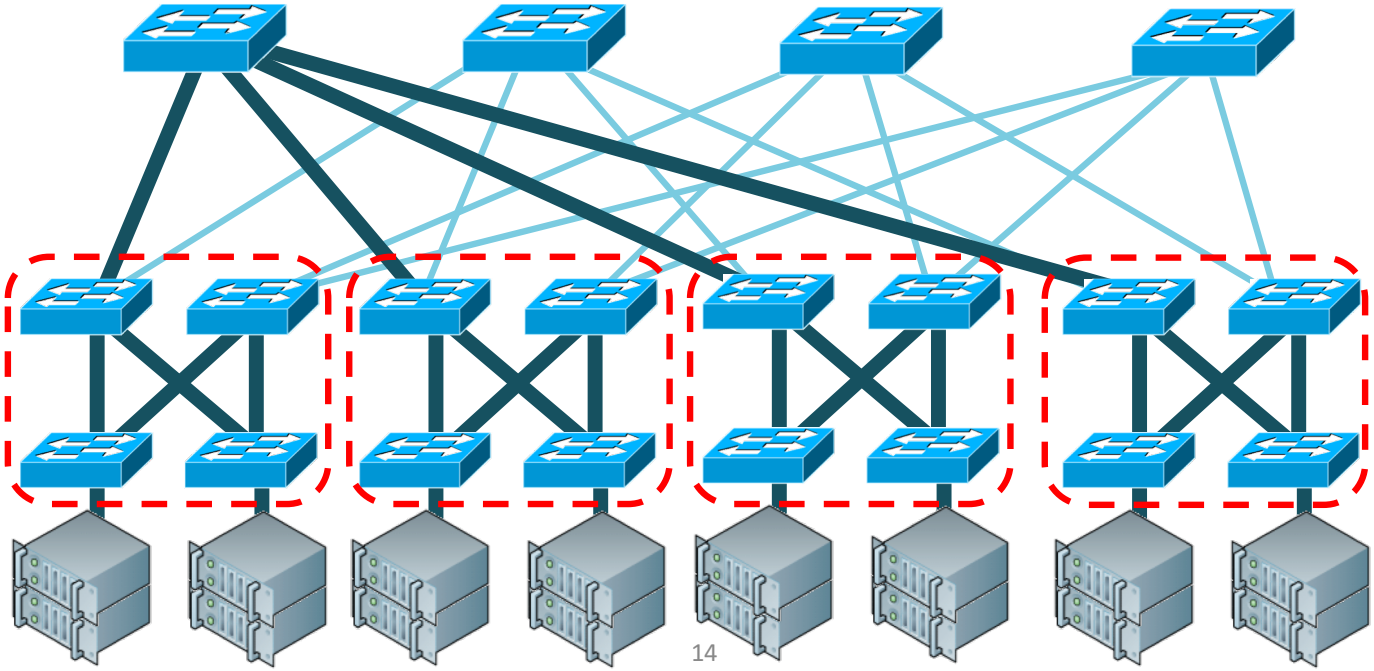
Problem: Oversubscription



Fat-tree

To build a K-ary fat tree
- K-port switches
- K pods, each with K switches
- $K^3/4$ hosts
- $(K/2)^2$ core switches

In this example K=4
- 4-port switches
- 4 pods, each with 4 switches
- $K^3/4 = 16$ hosts
- $(K/2)^2 = 4$ core switches



Summary

- Network configuration in data center
  - Why it differs from Internet
- What is the problem of oversubscription
- How do we solve the problem of oversubscription
  - Fat tree
  - How do you build fat tree
  - What happens if one router breaks (Hint: Facebook video)
- What is the problem of traffic hotspot in Data center
  - What are the solutions (wired solution vs. wireless solution) to address this problem

Summary (cont)

- TCP in Data center
  - Difference in requirements
    - Partition/aggregation pattern
    - Delay sensitive
    - Elephant flow vs. mice flow
    - Can change sender/receiver since they are controlled by the admin
  - Dirty slate approach: DCTCP
    - Adapting existing TCP
    - Explicit notification
    - Adjust window in proportional to congestion
  - Clean slate approach: D^3
    - Completely modify TCP to perform delay-based resource reservation
    - How to implement this??

Data driven Design

Data-Driven Network Design

- Type 1: Understand “real” network behaviors from ”in the wild” measurements
  - Many existing network protocol designs were driven by insights (which might be obsolete)

- Type 2: Use machine learning, especially reinforcement learning to identify rules for network functionalities

TCP Remy

- So far, human designers have created TCP's congestion-control algorithms **by hand**
  - Making implicit assumptions
- The **emergent behavior** of a flock of machines running TCP is complicated and difficult to characterize
- Remy: machine generated TCP algorithm based on observations of the current network state

http://web.mit.edu/remy/

Remy: a program that generates congestion-control schemes offline

Input:

- Prior assumptions (what network may do)
- Goal (what app wants)

Output: CC algorithm for a TCP sender (RemyCC)

Time: a few hours

Cost: \$5–\$10 on Amazon EC^2

Remy Structure

- Remy searches for the best congestion-control algorithm
- Optimizes expected objective over prior assumptions
- Makes tractable by limiting available state
- Remy tracks three congestion signals

r_ewma:	moving average of interval between acks
s_ewma:	. . . between sender timestamps echoed in acks
rtt_ratio:	ratio of last RTT to smallest RTT so far

Remy maps each state to an action

$\textsc{Rule}(r\_ewma, s\_ewma, rtt\_ratio) \rightarrow \langle m, b, \tau \rangle$

m	Multiple to congestion window
b	Increment to congestion window
τ	Minimum interval between two outgoing packets

On ack:

- $\langle m, b, \tau \rangle \leftarrow \textsc{Rule}(r\_ewma, s\_ewma, rtt\_ratio)$
- $\texttt{cwnd} \leftarrow m \cdot \texttt{cwnd} + b$

Send packet if:

- $\texttt{cwnd} > \texttt{FlightSize}$, and
- last packet sent $> \tau$ ago

Things that need extra care

- Remy assumes that the designer can model the network explicitly, i.e. making some assumptions on network model
  - What about mismatched set of assumptions

P2P

Summary

- How P2P differs from traditional client/server architecture

- Unstructured P2P
  - Design for ???
  - What does “unstructured” mean?
  - Which one is easy to find in unstructured P2P: popular items vs. non-popular items, why?
- Structured P2P
  - What is the structure for? Why do we need structure
  -

Why Do We Need Structure?

- Without structure, the search problem is harder
  - Anything can be anywhere
  - Need to look through many (or all) nodes to have confidence in finding object X

- So how do we add structure? Ans: Add rules
  - Give everyone (every machine) a name
  - Give every object a name
  - Match up all object names to machine names
    - If you are looking for X, mapping(X)→Y, talk to node Y

- Challenges
  - This mapping must be easy, embedded into the network
  - It must be consistent as the machines in network change
    - One new machine should not disrupt everything

Structured Overlays

- Application infrastructure for large-scale apps

- Two key pieces of functionality
  - Scalable any to any communication in network of size N
    - Much larger scale compared to distance vector/link state
    - Each machine only needs to know Log(N) other nodes
    - Routing from A->B takes at most Log(N) steps or hops
  - Deterministic key-node mapping
    - Allows peer rendezvous using common name
    - This is the object to node mapping we need
    - We call it Key-Based Routing

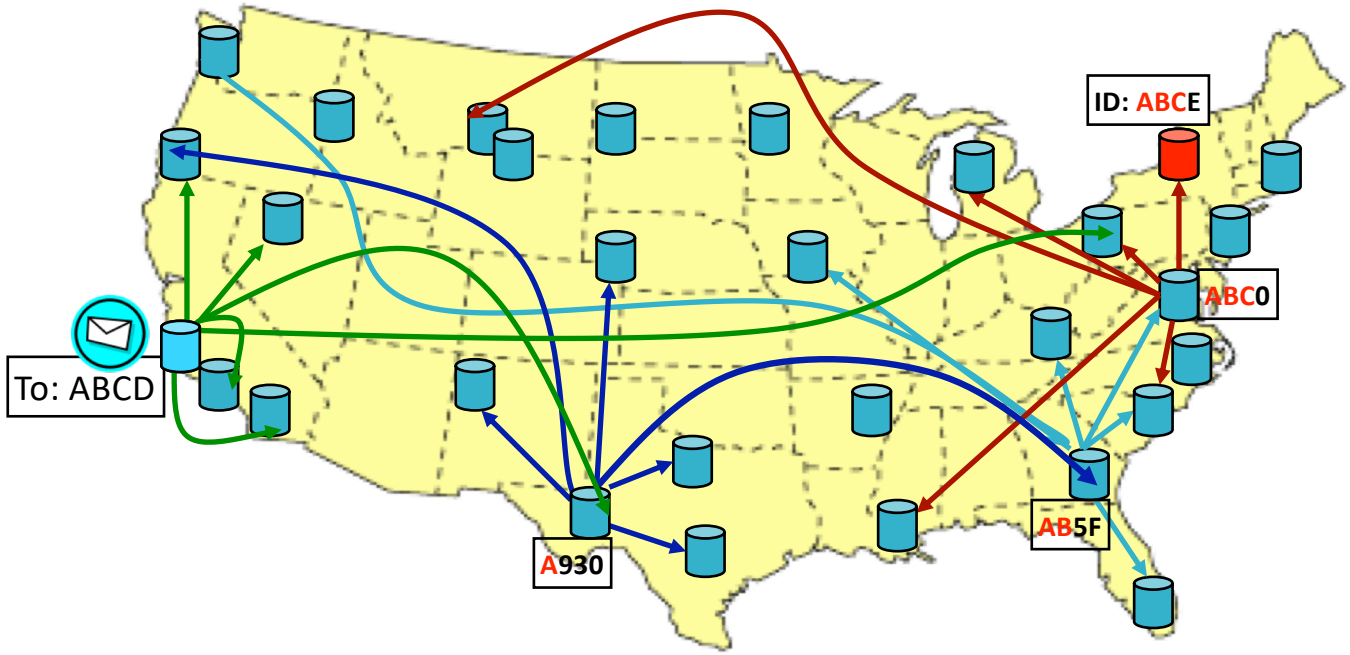
Peer-to-Peer Protocols

- Unstructured Peer to Peer Approaches
  - Napster, Gnutella, KaZaa
  - ❖probabilistic search (optimized for the hay, not the needle)
  - ❖locality-agnostic routing (resulting in high network b/w costs)

- Structured Peer to Peer Overlays
  - the first protocols (2001): Tapestry, Pastry, Chord, CAN
  - then: Kademlia, SkipNet, Viceroy, Symphony, Koorde, Ulysseus…
  - ❖distinction: how to choose your neighbors
    - ❖Tapestry/Chimera, Pastry: latency-optimized routing mesh
  - ❖distinction: application interface
    - ❖distributed hash table: put (key, data); data = get (key);
    - ❖Tapestry/Chimera: decentralized object location and routing

Structured Overlays at 10000 ft

- Node IDs and keys from randomized namespace (SHA-1)
  - incremental routing towards destination ID
  - each node has small set of outgoing routes, e.g. prefix routing
  - log (n) neighbors per node, log (n) hops between any node pair



This is it!

- Lecture 11:Wireless networking
- Lecture 12: Wireless Routing & TCP
- Lecture 13: Datacenter Networking
- Lecture 14: Data-driven Design (TCP Remi)
- Lecture 15: P2P Networks

Q&A Session