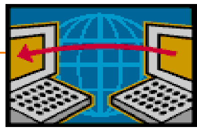


Miscellaneous Topics

EE122 Fall 2012

Scott Shenker

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley

Q/A on Project 3

Today's Lecture: Dim Sum of Design

- Quality-of-Service
- Multicast

    Announcements…

- Wireless addendum
- Advanced CC addendum

Extending the Internet Service Model

Internet Service Model

- Best-Effort: everyone gets the same service
    - No guarantees
- Unicast: each packet goes to single destination

Extending the service model

- Better than best-effort: **Quality of Service (QoS)**
- More than one receiver: **Multicast**

Quality of Service (QoS)

Summary of Current QoS Mechanisms

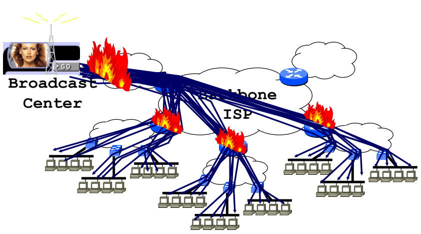
Blah, blah, **priority scheduling,** blah, blah….

Multicast

Motivating Example: Internet Radio

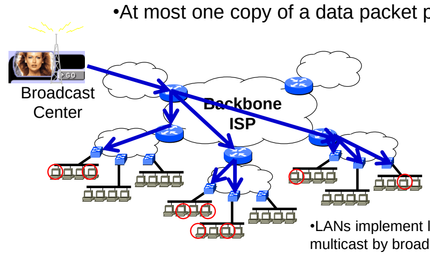
- Internet concert
  - More than 100,000 simultaneous online listeners
  - Could we do this with parallel unicast streams?
- Bandwidth usage
  - If each stream was 1Mbps, concert requires > 100Gbps
- Coordination
  - Hard to keep track of each listener as they come and go
- Multicast addresses both problems….

Unicast approach does not scale…



Instead build data replication trees

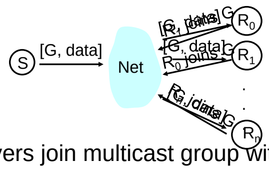
- Copy data at routers
- At most one copy of a data packet per link



- LANs implement link layer multicast by broadcasting

- Routers keep track of groups in real-time
- Routers compute trees and forward packets along them

Multicast Service Model



- Receivers join multicast group with address G
- Sender(s) send data to address G
- Network routes data to each of the receivers
- Multicast both delivery and rendezvous mechanism
  - Senders don't know list of receivers
  - The latter is often more important than the former

Multicast and Layering

- Multicast can be implemented at different layers
  - Link layer
    - e.g. Ethernet multicast
  - Network layer
    - e.g. IP multicast
  - Application layer
    - e.g. End system multicast
- Each layer has advantages and disadvantages
  - Link: easy to implement, limited scope
  - IP: global scope, efficient, but hard to deploy
  - Application: less efficient, easier to deploy [not covered]

Multicast Implementation Issues

- How is join implemented?
- How is send implemented?
- How much state is kept and who keeps it?

Link Layer Multicast

- Join group at multicast address G
  - NIC normally only listens for packets sent to unicast address A and broadcast address B
  - After being instructed to join group G, NIC also listens for packets sent to multicast address G
- Send to group G
  - Packet is flooded on all LAN segments, like broadcast
- Scalability:
  - State: Only host NICs keep state about who has joined
  - Bandwidth: Requires broadcast on all LAN segments
- Limitation: just over single LAN

Network Layer (IP) Multicast

- Performs inter-network multicast routing
  - Relies on link layer multicast for intra-network routing
- Portion of IP address space reserved for multicast
  - 2^{28} addresses for entire Internet
- Open group membership
  - Anyone can join (sends IGMP message)
    - Internet Group Management Protocol
  - Privacy preserved at application layer (encryption)
- Anyone can send to group
  - Even nonmembers

How Would YOU Design this?

- 5 Minutes….

IP Multicast Routing

- Intra-domain (know the basics here)
  - **Source Specific Tree**: Distance Vector Multicast Routing Protocol (DVRMP)
  - **Shared Tree**: Core Based Tree (CBT)
- Inter-domain [not covered]

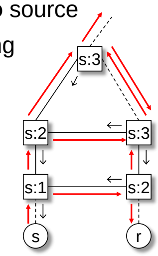
Distance Vector Multicast Routing Protocol

- Elegant extension to DV routing
  - Using reverse paths!
- Use shortest path DV routes to determine if link is on the source-rooted spanning tree
- Three steps in developing DVRMP
  - Reverse Path Flooding
  - Reverse Path Broadcasting
  - Truncated Reverse Path Broadcasting (pruning)

Reverse Path Flooding (RPF)

If incoming link is shortest path **to** source

- Send on all links except incoming
- Otherwise, drop

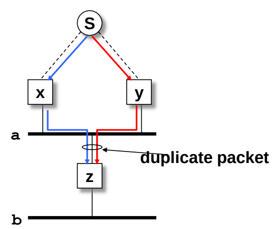


Issues: (fixed with RPB)

- Some links (LANs) may receive multiple copies
- **Every** link receives each multicast packet

Other Problems

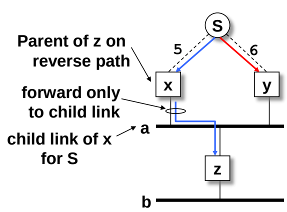
- Flooding can cause a given packet to be sent multiple times over the same link



- Solution: Reverse Path Broadcasting

Reverse Path Broadcasting (RPB)

- Choose single parent for each link along reverse shortest path to source
- Only parent forwards to child link
- Identifying parent links
  - Distance
  - Lower address as tie-breaker



Even after fixing this, not done

- This is still a broadcast algorithm – the traffic goes everywhere
- Need to "Prune" the tree when there are subtrees with no group members
- Networks know they have members based on IGMP messages
- Add the notion of "leaf" nodes in tree
  - They start the pruning process

Pruning Details

- Prune (Source,Group) at leaf if no members
  - Send Non-Membership Report (NMR) up tree
- If all children of router R send NMR, prune (S,G)
  - Propagate prune for (S,G) to parent R
- On timeout:
  - Prune dropped
  - Flow is reinstated
  - Down stream routers re-prune
- Note: a soft-state approach

Distance Vector Multicast Scaling

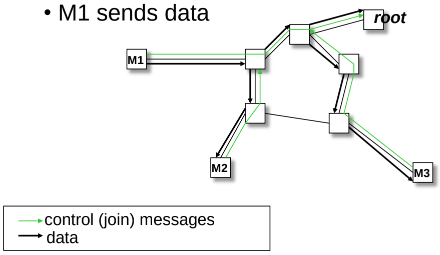
- State requirements:
  - O(Sources × Groups) active state
- How to get better scaling?
  - Hierarchical Multicast
  - Core-based Trees

Core-Based Trees (CBT)

- Pick “rendevouz point” for the group (called core)
- Build tree from all members to that core
  - Shared tree
- More scalable:
  - Reduces routing table state from O(S x G) to O(G)

Use Shared Tree for Delivery

- Group members: M1, M2, M3
- M1 sends data



Core-Based Tree Approach

- Build tree from all members to core or root
  - Spanning tree of members
- Packets are broadcast on tree
  - We know how to broadcast on trees
- Requires knowing root per group

Barriers to Multicast

- Hard to change IP
  - Multicast means changes to IP
  - Details of multicast were very hard to get right
- Not always consistent with ISP economic model
  - Charging done at edge, but single packet from edge can explode into millions of packets within network

Multicast vs Caching

- If delivery need not be simultaneous, caching (as in CDNs) works well, and needs no change to IP

31

Announcements

32

Clarification on Homework 3

- All links in the homework are full duplex.
  - Can send full line rate in both directions simultaneously

33

Announcements

- HW4 will just be a nongraded worksheet
- Review sessions (i.e., extended office hours) will be scheduled during class time of reading week
  - Will ask you to send in questions beforehand….
- In response to several queries, sometime after the SDN lecture I will give a short (and very informal) talk on my experience with Nicira.

34

Wireless Review

35

History

- MACA proposal: basis for RTS/CTS in lecture
  - MACA proposed as replacement for carrier sense
  - Contention is at receiver, but CS detects sender!
- MACAW paper: extended and altered MACA
  - **Implications of data ACKing**
  - Introducing DS in exchange: RTS-CTS-DS-Data-ACK
    - Shut up when hear DS or CTS (DS sort of like carrier sense)
  - Other clever but unused extensions for fairness, etc.
- 802.11: uses carrier sense *and* RTS/CTS
  - RTS/CTS often turned off, just use carrier sense
  - When RTS/CTS turned on, shut up when hear either
  - RTS/CTS augments carrier sense

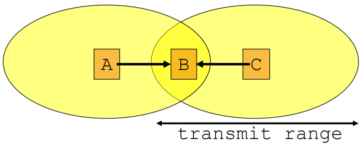
36

Why Isn't MACA Enough?

- That's what we now discuss, by repeating a few slides from previous lecture
- In what follows, we assume that basic reception is symmetric (if no interference):
  - If A is in range of B, then B is in range of A
- Any asymmetries in reception reflect interference from other senders

37

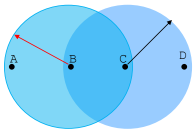
Hidden Terminals: Why MACA is Needed



- A, C can both send to B but can't hear each other
  - A is a *hidden terminal* for C and vice versa
- Carrier Sense by itself will be ineffective
  - **If A is already sending to B, C won't know to defer.**

38

Exposed Terminals: Alleged Flaw in CS



- Exposed node: using carrier sense
  - B sends a packet to A
  - C hears this and decides not to send a packet to D
  - *Despite the fact that this will not cause interference!*
- Carrier sense prevents successful transmission!

39

Key Points in MACA Rationale

- No concept of a global collision
  - Different receivers hear different signals
  - Different senders reach different receivers
- Collisions are at receiver, not sender
  - Only care if receiver can hear the sender clearly
  - It does not matter if sender can hear someone else
  - As long as that signal does not interfere with receiver
- Goal of protocol:
  - Detect if receiver can hear sender
  - Tell senders who might interfere with receiver to shut up

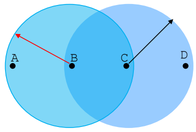
40

What Does This Analysis Ignore?

- Data should be ACKed!
  - In wireless settings, data can be easily lost
  - Need to give sender signal that data was received
- How does that change story?
- Connection is now two-way!
  - Congestion is at both sender and receiver
  - Carrier-sense is good for detecting the former
  - MACA is good for detecting the latter

41

Exposed Terminals Revisited



- Exposed node: with only MACA, both B and C send
- *But when A or D send ACKs, they won't be heard!*
- *Carrier-sense prevents B, C sending at same time*

42

802.11 Overview (oversimplified)

- Uses carrier sense *and* RTS/CTS
  - RTS/CTS often turned off, just use carrier sense
- If RTS/CTS turned on, shut up when hear either
  - CTS
  - Or current transmission (carrier sense)
- What if hear only RTS, no CTS or transmission?
  - You can send (after waiting small period)
    - CTS probably wasn't sent

What Will Be on the Final?

- General awareness of wireless (lecture, section)
- Reasoning about a given protocol
  - If we used the following algorithm, what would happen?
- You are **not** expected to know which algorithm to use; we will tell you explicitly:
  - RTS/CTS
  - Carrier Sense
  - Both

Back to Congestion Control

Quick Review of Advanced CC

- Full queues: RED
- Non-congestion losses: ECN
- High-speeds: Alter constants: HSTCP
- Fairness: Need isolation
- Min. flow completion time: Need better adjustment
- Router-Assisted CC: *Isolation*: WFQ, AFD
  Adjustment: RCP

Why is Scott a Moron?

Or why does Bob Briscoe think so?

Giving equal shares to "flows" is silly

- What if you have 8 flows (to different destinations), and I have 4…
  - Why should you get twice the bandwidth?
- What if your flow goes over 4 congested hops, and mine only goes over 1?
  - Why not penalize you for using more scarce bandwidth?
- And what is a flow anyway?
  - TCP connection
  - Source-Destination pair?
  - Source?

flow rate fairness
dismantling a religion
<draft-briscoe-tsvarea-fair-01.pdf>

status: individual draft
final intent: informational
intent next: tsvwg WG item after (or at) next draft

Bob Briscoe
Chief Researcher, BT Group
IETF-68 tsvwg Mar 2007

Charge people for congestion!

- Use ECN as congestion markers
- Whenever I get ECN bit set, I have to pay $$$
- No debate over what a flow is, or what fair is…
  - Just send your packets and pay your bill
- Can avoid charges by sending when no congestion
- Idea started by Frank Kelly, backed by much math
  - Great idea: simple, elegant, effective

50

Why isn't this the obvious thing to do?

- Do you really want to sell links to highest bidder?
  - Will you ever bandwidth when Bill Gates is sending?
  - He just sends as fast as he can, and pays the bill
- Can we embed economics at such a low level?
  - Charging mechanisms usually at higher levels
- Is this the main problem with congestion control?
  - Is this a problem worth solving? Bandwidth caps work…
- This approach is fundamentally right…
  - …but perhaps not practically relevant

51

Datacenter Networks

52

What makes them special?

- Huge scale:
  - 100,000s of servers in one location
- Limited geographic scope:
  - High bandwidth (10Gbps)
  - Very low RTT
- Extreme latency requirements (especially the tail)
  - With real money on the line
- Single administrative domain
  - No need to follow standards, or play nice with others
- Often "green field" deployment
  - So can "start from scratch"…

53

This means….

- Can design "from scratch"
- Can assume very low latencies
- Need to ensure very low queuing delays
  - Both the average, and the tail….
  - TCP terrible at this, even with RED
- Can assume plentiful bandwidth internally
- As usual, flows can be elephants or mice
  - Most flows small, most bytes in large flows…

54

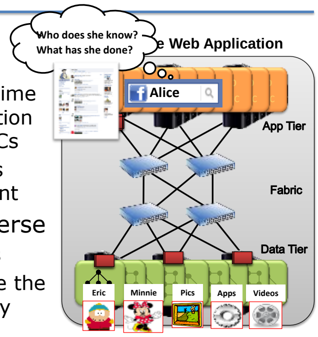
Deconstructing Datacenter Packet Transport

Mohammad Alizadeh, Shuang Yang, Sachin Katti, Nick McKeown, Balaji Prabhakar, Scott Shenker

Stanford University U.C. Berkeley/ICSI

Transport in Datacenters

- Latency is King
  - Web app response time depends on completion of 100s of small RPCs
  - Tail of distribution is particularly important
- But, traffic also diverse
  - Mice AND Elephants
  - Often, elephants are the root cause of latency



Transport in Datacenters

- Two fundamental requirements
  - **High fabric utilization**
    - Good for all traffic, esp. the large flows
  - **Low fabric latency (propagation + switching)**
    - Critical for latency-sensitive traffic
- Active area of research
  - DCTCP[SIGCOMM'10], D3[SIGCOMM'11] HULL[NSDI'11], D^2TCP[SIGCOMM'12] PDQ[SIGCOMM'12], DeTail[SIGCOMM'12]

vastly improve performance, but very complex

Goal of pFabric

- Subtract complexity, not add to it
- Start from scratch and ask:
  - What components do we need for DC CC?
  - How little mechanism can we get away with?
- None of what we say here applies to general CC, it is limited to DC setting

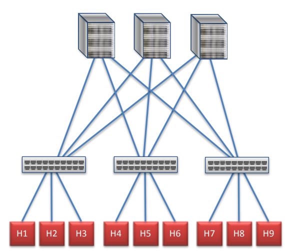
pFabric in 1 Slide

Packets carry a single priority #
- e.g., prio = remaining flow size

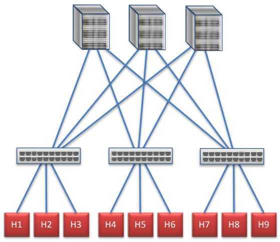
pFabric Switches
- Very small buffers (e.g., 10-20KB)
- Send highest priority / drop lowest priority pkts
  - Give priority to packets with least remaining in flow

pFabric Hosts
- Send/retransmit aggressively
- Minimal rate control: just prevent congestion collapse

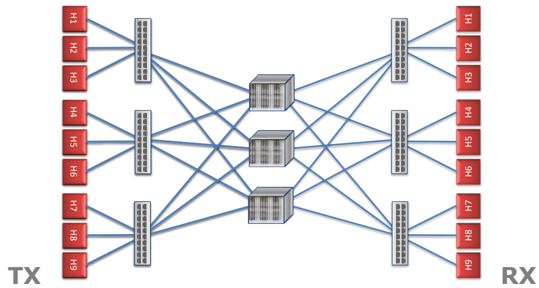
DC Fabric: Just a Giant Switch!



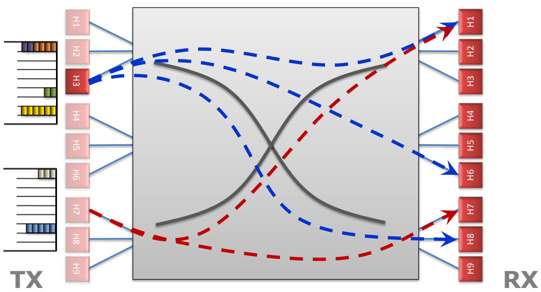
DC Fabric: Just a Giant Switch!



DC Fabric: Just a Giant Switch!



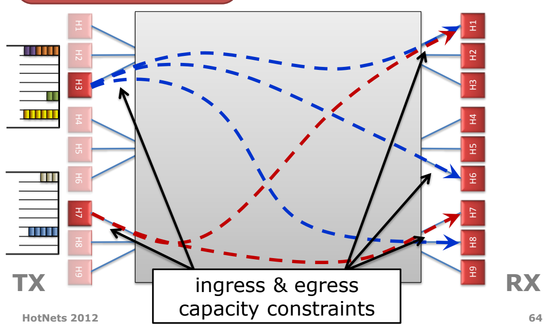
DC Fabric: Just a Giant Switch!



DC transport = Flow scheduling on giant switch

Objective?

➢ Minimize avg FCT



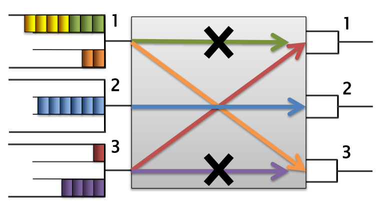
Min. FCT with Simple Queue

- Just use shortest-job first…..
- But here we have many inputs and many outputs with capacity constraints at both

"Ideal" Flow Scheduling

Problem is NP-hard ☹ [Bar-Noy et al.]

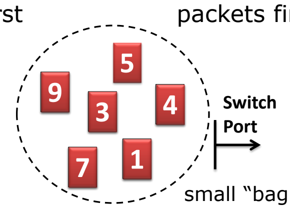
– Simple greedy algorithm: **2-approximation**



pFabric Design

pFabric Switch

- ➢ **Priority Scheduling** send higher priority packets first
- ➢ **Priority Dropping** drop low priority packets first



small "bag" of packets per-port

prio = remaining flow size

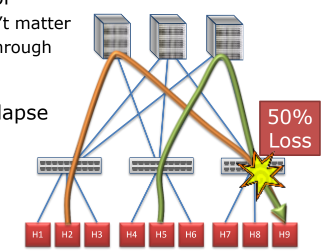
Near-Zero Buffers

- Easiest way to keep delays small?
  - Have small buffers!
- Buffers are very small (~1 BDP)
  - e.g., C=10Gbps, RTT=15μs → BDP = 18.75KB
  - Today's switch buffers are 10-30x larger

pFabric Rate Control

- Priority scheduling & dropping in fabric also simplifies rate control
  - Queue backlog doesn't matter
  - Priority packets get through

One task:
Prevent congestion collapse when elephants collide



pFabric Rate Control

- Minimal version of TCP
  1. Start at line-rate
     - Initial window larger than BDP
  2. No retransmission timeout estimation
     - Fix RTO near round-trip time
  3. No fast retransmission on 3-dupacks
     - Allow packet reordering

Why does this work?

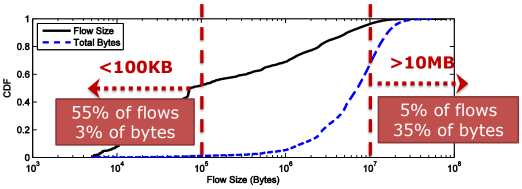
Key observation:
Need the highest priority packet destined for a port **available at the port** at any given time.

- **Priority scheduling**
  - ➢ High priority packets traverse fabric as quickly as possible
- **What about dropped packets?**
  - ➢ Lowest priority → not needed till all other packets depart
  - ➢ Buffer larger than BDP → more than RTT to retransmit

Evaluation

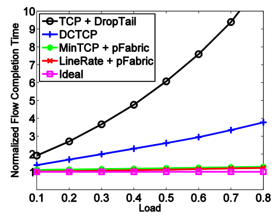
- 54 port fat-tree: 10Gbps links, RTT = ~12μs
- Realistic traffic workloads
  - Web search, Data mining

* From Alizadeh et al. [SIGCOMM 2010]

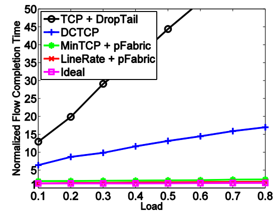


Evaluation: Mice FCT (<100KB)

Average



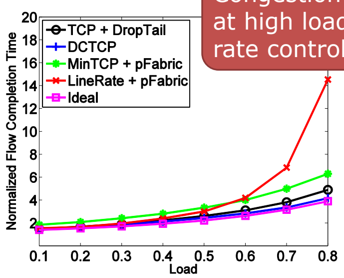
99^{th} Percentile



Near-ideal: almost no jitter

Evaluation: Elephant FCT (>10MB)

Congestion collapse at high load w/o rate control



Summary

pFabric's entire design:
Near-ideal flow scheduling across DC fabric

- **Switches**
  - Locally schedule & drop based on priority
- **Hosts**
  - Aggressively send & retransmit
  - Minimal rate control to avoid congestion collapse