CS4450

Computer Networks: Architecture and Protocols

Lecture 19
MPLS, Multicast

Spring 2018

Rachit Agarwal

Traffic Engineering (TE)

- Connectivity is necessary but not sufficient

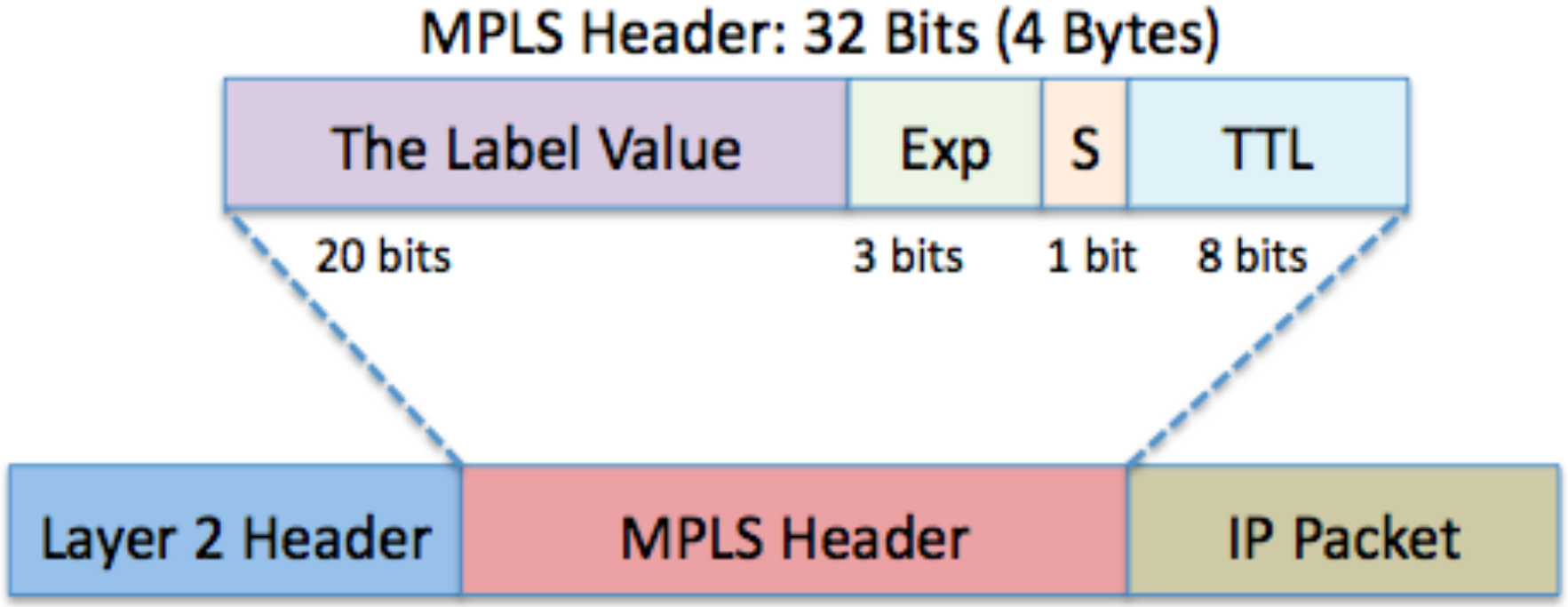
- Need to also provide performance
- Requires that links on the path not be overloaded
    - otherwise, high queueing delay

    - provide reasonable bandwidth to connections by spreading load
- TE is a way of distributing load on the network
    - i.e., not all packets travel the “shortest path”
- **One way to do this: spread load over MPLS paths**

Multiprotocol Label Switching (MPLS)

- Operators wanted more flexibility in routing traffic
    - Normal routing just used destination address…
- Wanted ability to route on larger aggregates
    - First decide if flow belongs to aggregate, then route aggregate
    - Example: all traffic from LA to NY follow same path
- Wanted ability to route at finer granularity
    - Not all packets with same destination take same path
- Solution: insert a “label” before IP header
    - Route based on that label

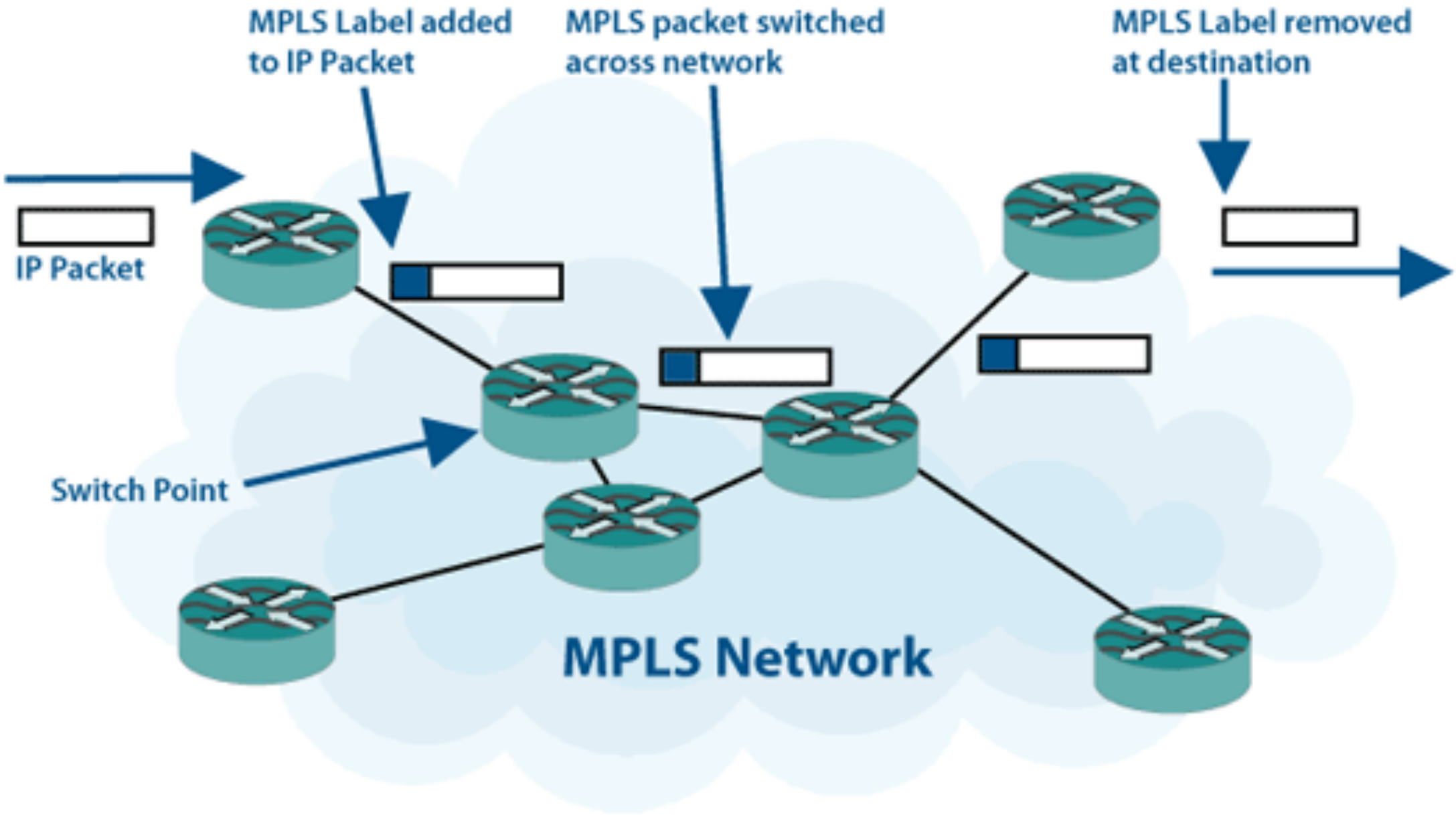
MPLS Header


MPLS Header: 32 Bits (4 Bytes)
The Label Value
Exp
S
TTL
20 bits
3 bits
1 bit
8 bits
Layer 2 Header
MPLS Header
IP Packet

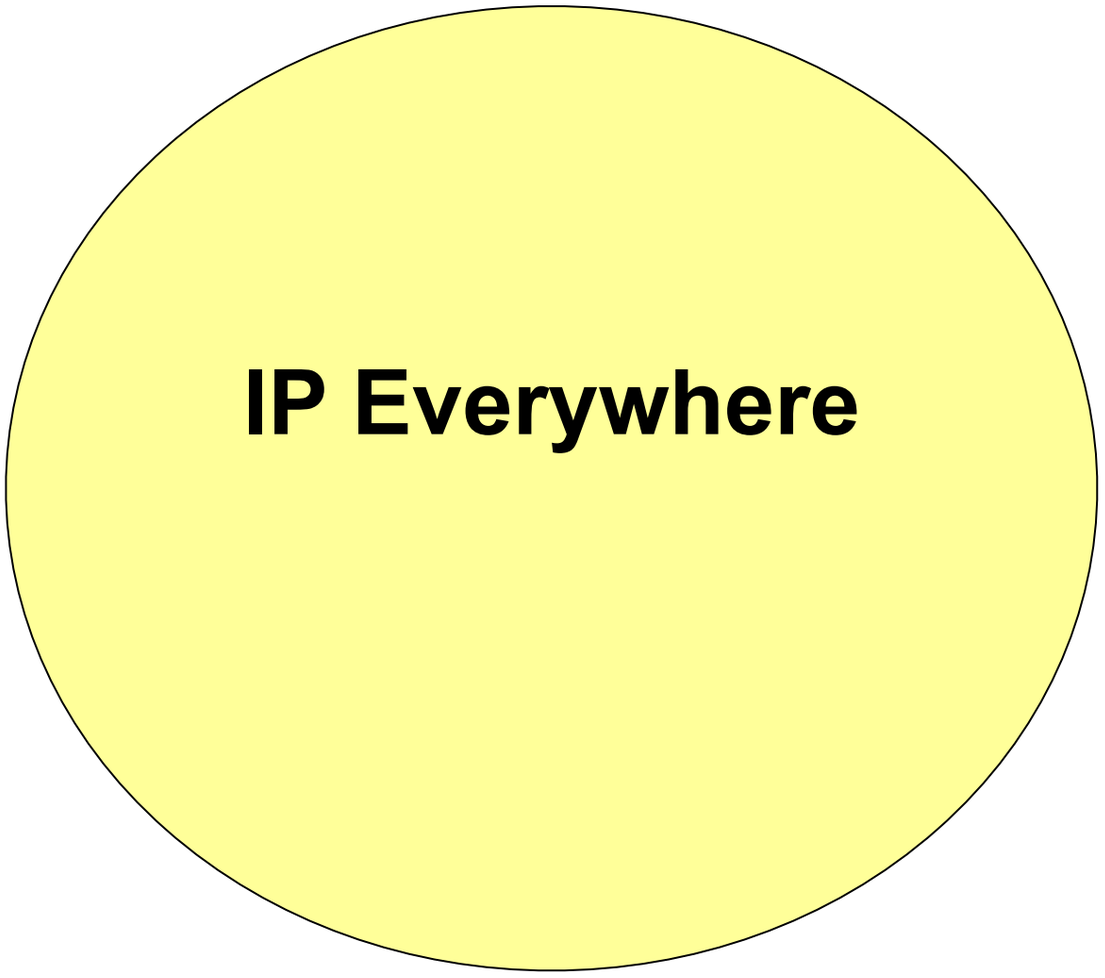
Using MPLS

- Make a distinction between edge and core routers
- Edge routers inspect IP header, insert MPLS label
- Core routers route based on MPLS label
- Must set up forwarding state for MPLS labels
  - Done in a variety of ways, for a variety of goals
  - Supporting failover paths, TE,…

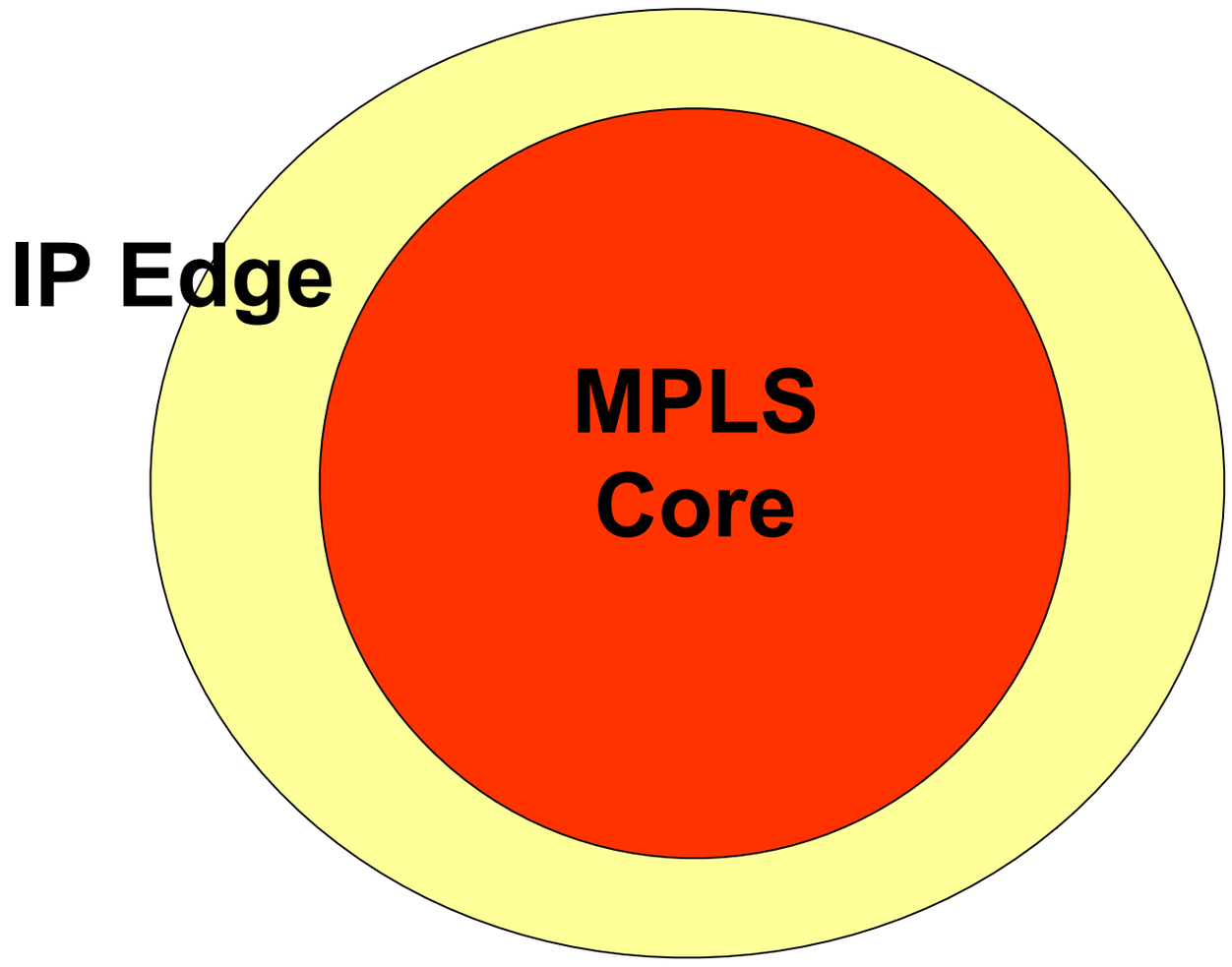
How MPLS Works



Theoretical Model of Carrier Network



Actual View of Most Carrier Networks



Another Case of Edge/Core Split

- Edge: has all the intelligence/functionality

- Core: dumb plumbing providing connectivity
- An example of modularity:
    - Keep core simple, fast, cheap (but has to be distributed)

    - Let edge be complex and slower (no distribution)

- Distinction should be recognized more broadly
- Made edge software on x86s, core hardware

MPLS is widely used

- Extremely important practically, not intellectually
- Because it is not tightly tied to a single purpose
    - Used for VPNs, TE, etc.
- Each use is ad hoc, rather than overall paradigm
- Like the IPv6 flow ID: all mechanism, no policy
- If IPv6 happened sooner, we wouldn’t need MPLS

MPLS and TE

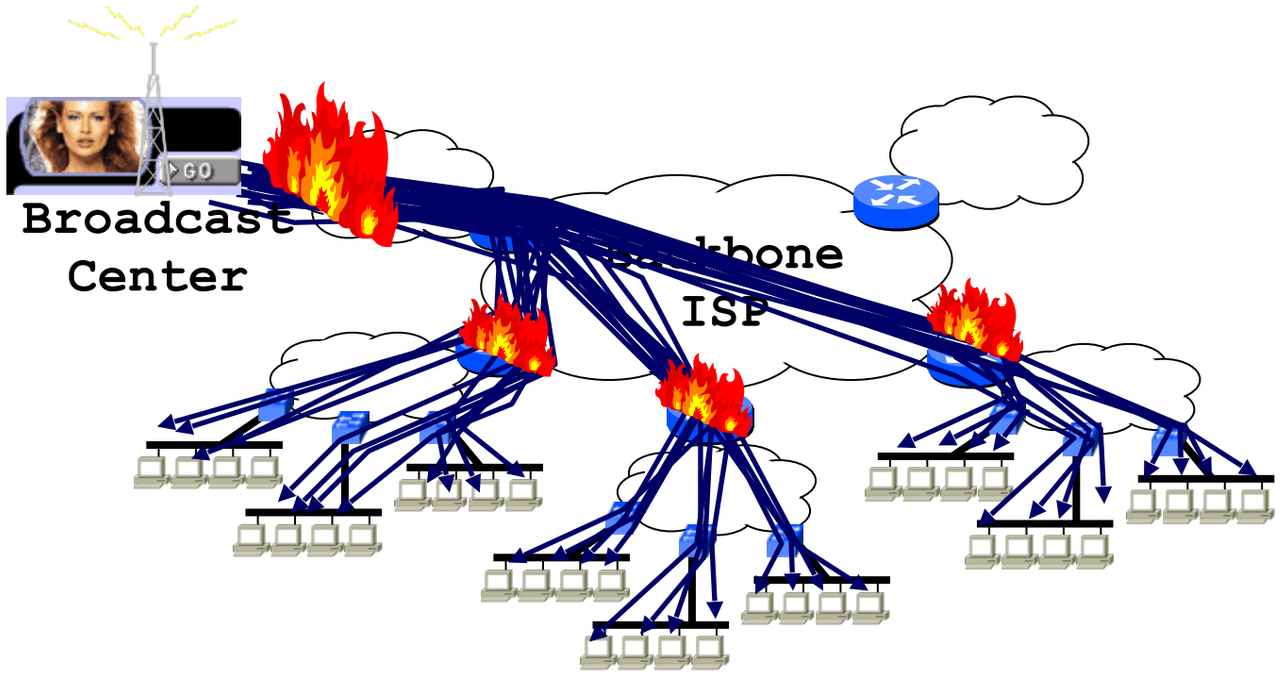
- MPLS:
  - Enables operators to nail up paths between two points
  - Think of an MPLS tunnel as a virtual link (layer 2.5)
- Most modern backbones are built out of MPLS
  - With backup paths to deal with failure
- Load is spread by having multiple MPLS paths between any two points, and then adjusting how load is split between them….

Multicast

Motivating Example: Internet Radio

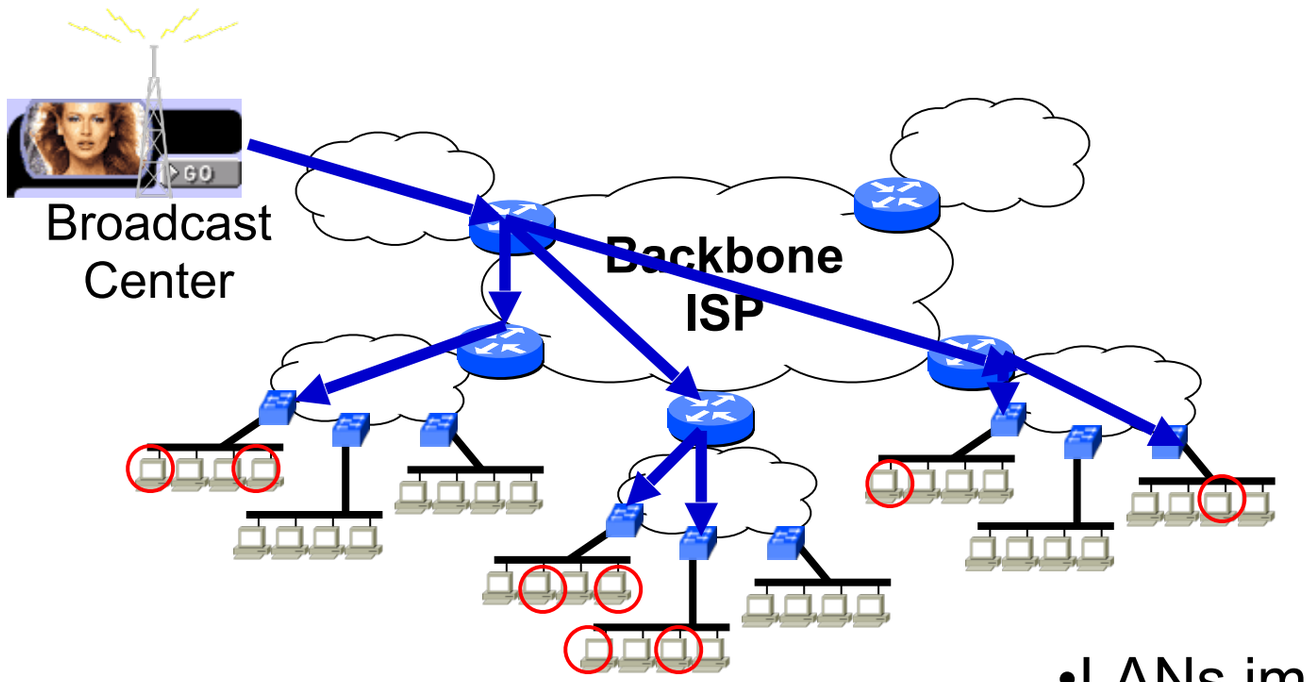
- Internet concert
  - More than 1M simultaneous online listeners
  - Could we do this with parallel unicast streams?
- Bandwidth usage
  - If each stream was 1Mbps, concert requires > 1 Tbps
- Coordination
  - Hard to keep track of each listener as they come and go
- Multicast addresses both problems….

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: single sent packet delivered to many dests**

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 = Network Interface Card
  - NIC normally only listens for packets sent to unicast MAC address A and broadcast address ff:ff:ff:ff:ff:ff
  - After being instructed to join group G, NIC also listens for packets sent to multicast address G
- Send to group G
  - Packet treated like a broadcast packet, sent everywhere
- Scalability:
  - State: Only host NICs keep state about who has joined
  - Bandwidth: Requires broadcast over subnet
- Limitation: just over single subnet

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 (mistake!)

Requirements for the design

- Receivers join group G (using IGMP message)
  - Internet Group Management Protocol

- Senders send packet to destination G
  - With no knowledge of who the receivers are

- Intradomain network routes packets to all receivers
  - All the responsibility placed on the network

- Must be much more efficient than flooding

- Need not deal with groups across multiple domains

IP Multicast Routing

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

Distance Vector Multicast Routing Protocol

- Elegant extension to DV routing
- Will cover two main steps in DVRMP
  - Reverse Path Flooding
  - Truncation (pruning)
- **Discussion is drastically oversimplified!**

General Strategy

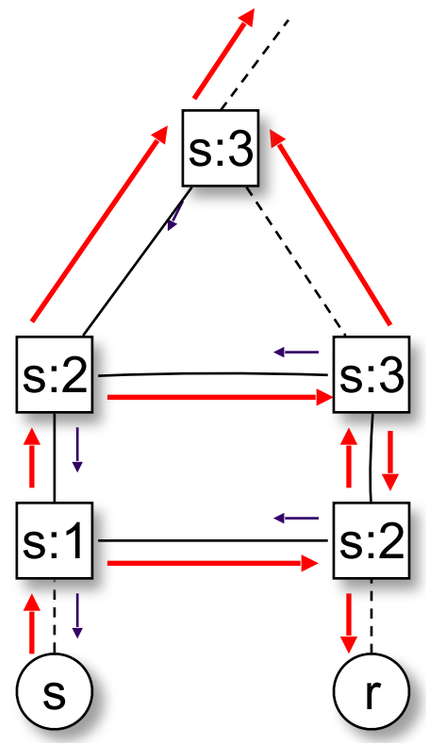
- Start by flooding packets along a tree
  - Flooding in the Internet requires some thought
  - In particular, how do you flood without loops?
- Prune portions of tree that don't have members
  - So only the first few packets of a multicast flow wasted

General Tactics

- Construct a tree from a source to all destinations
  - This is done by using the **reverse-paths** from all destinations to the source
  - That is, packets from multicast source S follows paths that unicast routing would take from each D **to** S.
- Why reverse paths?
  - Forward paths from source to all destinations not guaranteed to be a tree (**why?**)
  - Reverse paths are set of paths from all destinations to source, and this must be a tree (for dest-based routing)
- Packets sent along tree (copied when tree splits)

Reverse Path Flooding (RPF)

- If incoming link is shortest path to source
  - Send on all links except incoming
  - Otherwise, drop
- Issues:
  - Every link receives each multicast packet
  - Some links (LANs) may receive multiple copies
  - Can be avoided by knowing your parent in r-tree



RPF is not enough!

- This is 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
  - Don’t worry how you know you are a leaf…
  - (poisoned reverse is involved!)

Sending Joins

- Hosts that want to join group send IGMP message
  - "I want to join group G"
  - To first-hop router
- This router knows whether it has local members
- If it gets flooded messages from a source S, but has no local members (and is a leaf node), then it prunes itself from tree.

Pruning Details

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

DVMRP Review

- Packets are initially broadcast everywhere
    - Using reverse paths to prevent loops
- Leaf nodes send prunes if they have no members
    - Prunes travel toward source (using forward path)
- **Result**
    - When all prunes have been sent (and none have timed out), then all packets from source S travel the subtree that connects S to all members of the group
        - In the reverse direction!

Distance Vector Multicast Scaling

- State requirements:
    - O(Sources × Groups) active state
- How to get better scaling?
    - Hierarchical Multicast
    - Core-based Trees
- What you need to know:
    - General strategy
    - Resulting paths (per source delivery trees)

Core-Based Trees (CBT)

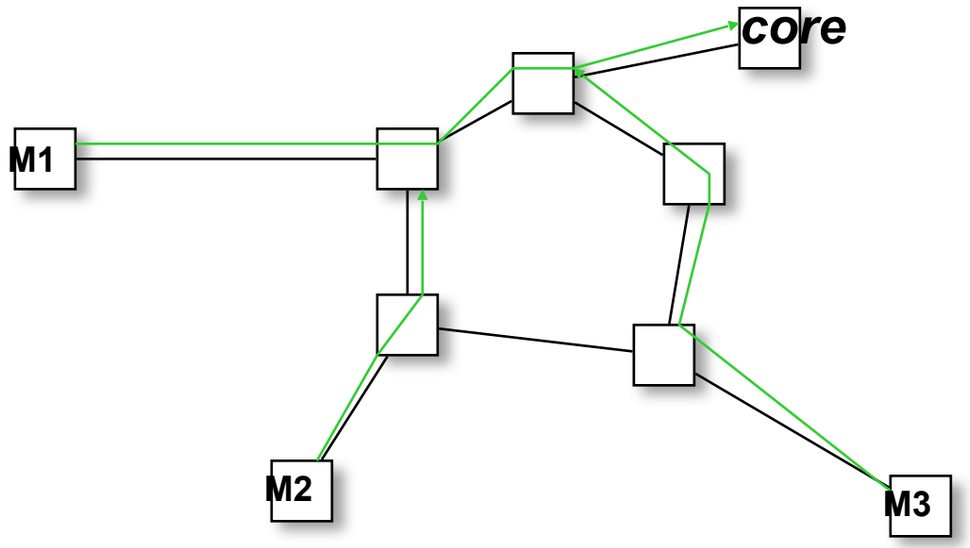
- Pick “rendezvous point” for the group (called core)
    - The mapping between group G and core IP address is known (somehow)
- Build tree from all members to that core
    - (using forward-path unicast routing)
    - Shared tree
- More scalable:
    - Reduces routing table state from O(S x G) to O(G)
    - No initial flooding

Sending Packets

- Members:
    - Send on tree (broadcast)
- Nonmembers:
    - Encapsulate packet and send to core
        - Using core's IP address
    - Core then sends it on tree

Establishing Shared Tree

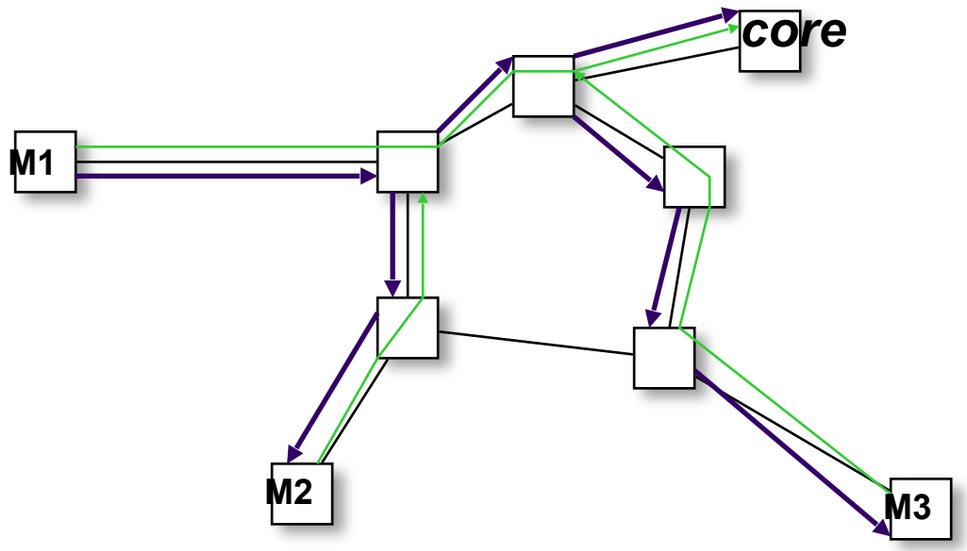
- Group members: M1, M2, M3Send on tree (broadcast)



control (join) messages

Use Shared Tree for Delivery

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



control (join) messages
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 core per group
    - This is a problem in many settings
    - Core must exist before members join
    - But what if core is far from members?
- What you need to know: everything.

Special-Case: Single-Source Mcast

- For each SSM group, only a single sender
    - Which serves as the core (perfectly located!)
- Well-suited to live event usage
    - A natural single source
    - Potential large audience for simultaneous reception

Barriers to Multicast

- Hard to change IP
  - Multicast means changes to IP
    - Unicast IP remains same, but IP now must include multicast
  - Details of multicast were very hard to get right
    - Years-long effort with many brilliant people
      - Deering, Jacobson, Estrin, Handley, etc.
- Not always consistent with ISP economic model
  - Charging done at edge, but single packet from edge can explode into millions of packets within network

Review of Multicast

- DVMRP:
  - Per-source trees (reverse path!)
  - Flood then prune
  - Issues: scalability (state) and flooding
- CBT:
  - Shared tree
  - Built by receiver joins sent to core
  - Any sender can reach tree by going to core

What Makes Interdomain Mcast Hard?

- Can’t flood then prune in a global network
- If you use CBT, where do you place cores?
  - Can be solved using large key-value stores
  - And a hierarchical set of cores

Multicast vs Caching

- If delivery need not be simultaneous, caching (as in CDNs) works well, and needs no change to IP
- This is true for almost all online applications except:
  - Gaming
  - Videoconferences
  - …..

Any Questions

Network Security

My definition of “network security”

- “network security” ≠ “security in a connected world”

- If network magically transfers data between known parties, there is no “network security” problem
- There are many other security problems
    - Distributed system (if A lies to B, does system crash?)
    - Operating system (Can A’s system be compromised?)
    - …
- But these may not require network solutions

A few non-network security issues

- Browser “drive-by” exploits
- Server vulnerabilities
- Spam
- Phishing
- Account theft
- ....

Two Kinds of Network Security Goals

- Core concern: accomplishing communication
  - Getting the data from A to B intact
  - Knowing it was from intended party, to intended party
- Also: Keeping bystanders as ignorant as possible
  - Making sure C, D, etc. don't know what A and B did

Core Security Requirements

- Availability: Will the network deliver data?
- Authentication: Who is sending me data?
- Integrity: Do messages arrive in original form?
- Provenance: Who is responsible for this data?
  - Not who sent the data, but who created it
  - Important because communication may not be directly between actors, but through intermediaries
- (i.e., did these headlines really come from CNN?)

Keeping Bystanders Ignorant

- Privacy: can others read data I send?
- Anonymity: can I avoid revealing my identity?
- Freedom from traffic analysis: can someone tell when I am sending and to whom?
- Today, will ignore latter two and focus on privacy

List of Goals

- Availability
- Authentication
- Integrity
- Provenance
- Privacy

Any Questions

Public Key Crypto Provides

- Way to authenticate yourself: signature
- Way to ensure privacy: encryption
    - with rcvr's public key
- Way to verify integrity: hash function (or MAC)
- Way to verify provenance: signature
- In short, crypto provides all but availability!

Protecting Availability

How can availability be harmed?

- Problems in basic protocols
    - Persistent outages due to natural events
- External vulnerabilities in basic protocols
    - Attackers can prevent protocols from functioning
- Internal vulnerabilities in basic protocols
    - If attackers compromise routers, can prevent network from functioning
- Denial-of-service attacks
    - Overwhelming one or more resources

How Can We Avoid These?

- Problems in basic protocols
    - **Good design and careful operation**
- External vulnerabilities in basic protocols
    - **Good design and careful operation**
- Internal vulnerabilities in basic protocols
    - **Good design and careful operation**
- Denial-of-service attacks
    - **Requires new thinking….**

Denial of Service (DoS)

- Attacker prevents legitimate users from using something (network, server)
- Motives?
    - Retaliation, extortion, commercial advantage, etc.
- Often done via some form of flooding
    - Overwhelming some resource….
- Can be done at different semantic levels
    - Network: clog a link or router with a huge rate of packets
    - Transport: overwhelm victim’s ability to handle connections
    - Application: overwhelm victim’s ability to handle requests

Mechanism

- Attacker sends traffic to victim as fast as possible
    - It will often use (many) spoofed source addresses …
- Using multiple hosts (zombies) yields a **Distributed Denial-of-Service** attack, aka **DDoS**
- Traffic is varied (sources, destinations, ports, length) so no simple filter matches it
- If attacker has enough zombies, often doesn't need to spoof - victim can't shut them down anyway! :-(

Unfortunate Fact

- Easy to identify hosts that participate in attacks.
- But they typically have well-meaning owners
    - They've just been compromised
- Cannot just disconnect all compromised hosts!
    - Customers would sue their provider!
- Need to allow them to function, while preventing them from bringing down the Internet…

Attack on Dyn: 10/21/2016

- Dyn provides DNS service to many companies
- Attack took the form of DNS lookup requests
    - From tens of millions of IP addresses
    - Internet-connected devices (printers, cameras, baby monitors, etc.) infected with Mirai malware.
- Estimated load: 1.2 terabits per second
- Defense:
    - Anycast, internal filtering, external scrubbing, ??

Any Questions

Defending Against DDoS

- In short, we have no systematic defense
- Can do ad hoc scrubbing
  - Try to identify attacking traffic and block it
  - While allowing real traffic through
  - But this is a losing game….
- How could we change the architecture to defend against DDoS?

Architectural Approaches to DDoS?

- Talk to your neighbors. 3 minutes.
- If you could redesign architecture, how would you design it so that DDoS attacks could be:
  - Mitigated
  - Prevented
  - Blocked
- And if you don’t have to change the architecture, even better!

Fight-Fire-With-Fire (mitigation)

- Typically
  - Victim has N customers whose traffic is overwhelmed by M attackers, even though M<<N
  - Bringing down the victim's server
  - Because customers send intermittently, attackers are sending at full line rate
- "Crazy" Defense
  - **Don't slow attackers down, just speed up customers**
  - Ask customers to send more rapidly, and then randomly filter traffic at server to serve only a small fraction
  - Customers get their share of the service N/(N+M)
  - Can't do better than that without distinguishing attackers from customers

Capabilities (prevention)

- Internet is “default-on”: Anyone can send to anyone without asking for permission
- One way to deal with DDoS is to force people to ask for permission to send (get “capabilities”)
- When attackers start up, can refuse to renew their permission to send
- Complicated design, huge change to architecture
  - Yuck. Cure worse than disease….

Shut-Up Packets (blocking)

- Embed logic in NIC to handle shut-up requests
    - NIC out of reach of OS, can't be easily compromised
- If host A sends a shut-up packet to host B, then host B's NIC prevents B from sending packets to host A's address (for some period of time)

- Easy to support in NIC. Subtle points in design.
- Doesn't shut down hosts. Merely allows destinations to say "don't send traffic to me"
- Best approach so far for DDoS in my opinion

Any Questions