Today’s Lecture

- Routing
  - BGP
  - IPv6 (brief)
- Advanced Routing
  - Multicast
  - MPLS
Global Internet

Outline
BGP

BGP-4: Border Gateway Protocol

- **AS Types** → Autonomous System
  - **Stub AS**: has a single connection to one other AS
    - carries local traffic only
  - **Multihomed AS**: has connections to more than one AS
    - refuses to carry transit traffic
  - **Transit AS**: has connections to more than one AS
    - carries both transit and local traffic
- Each AS has:
  - one or more border routers
  - one BGP *speaker* that advertises:
    - local networks
    - other reachable networks (transit AS only)
    - gives *path* information

Multi-Backbone Internet
BGP Example

- Speaker for AS2 advertises reachability to P and Q
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2

- Speaker for backbone advertises
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).
- Speaker can cancel previously advertised paths

IP Version 6

- Features
  - 128-bit addresses (classless)
  - multicast
  - real-time service
  - authentication and security
  - autoconfiguration
  - end-to-end fragmentation
  - protocol extensions
- Header
  - 40-byte “base” header
  - extension headers (fixed order, mostly fixed length)
    - fragmentation
    - source routing
    - authentication and security
    - other options

Addresses
Project 2

- Calendar Service
  - Client
  - Server
    - Iterative
    - Multi-Threaded
  - XML

Data Structure

- Server with group of calendars
  - Calendar Name
    - Think of it as a logical unit
    - No space in name
  - Event
    - Instance in a calendar
    - Brief identifier (one word)
    - Date
    - Start Time
    - Duration

Functionality

- Need the ability to
  - Add an event
  - Update an event
  - Remove an event
  - Get all of the events
    - For a day
    - All events slowly (for testing multi-threading)
**XML**

- eXtensible Markup Language
  - HTML – HyperText Markup Language
- Lends structure to data
  - Tags denoted by `< >`
  - Element ➔ demarcations of blocks of data
  - Attributes ➔ additional info on elements
- Key aspects
  - Human readable
  - Extensible ➔ add new tags as needed

**XML Structure**

```xml
<MyInfo>
  <Info> John Doe
      <Class Name="CSE40416" />
      <Class Name="CSE40422" />
  </Info>
</MyInfo>
```

**Client**

- Two modes of usage
  - Simple form
    - All data via command prompt
  - XML form
    - Read / send the XML data

```
./myCal CSE30264 add 022310 1100 1.25 Class
```

```
Calendar Action Date Time Identifier Duration
```
Client

- Not all commands take the same arguments
  - update
    - Needs all info
  - remove
    - Just needs time / date
  - get
    - Just needs the date
  - getslow
    - No date, will slowly give all calendar events

Message Structure

```xml
<?xml version="1.0" encoding="UTF-8"?>
<update>
  <Date>022310</Date>
  <Start>1100</Start>
  <Duration>1.25</Duration>
  <Info>MidTermExam</Info>
</update>
```

One example, structure is up to you

Client

- Send XML block to server
- Wait for response
  - Structure of response also up to you
- Server info
  - Save in a .mycal file in the local directory
  - ls -a to see all "hidden" files
  - Any form that you want for the file
  - Server IP + Port
Server

- Two versions
  - Iterative version
    - Just like Project 1
  - Multi-threaded version
    - Use pthreads, 1 thread / client
    - getslow will be used to test / compare

Server

- Store individual calendars in own format
  - Files on disk
  - One big file for the calendar
  - Keep in memory / etc.
- Up to you how to store it
- Server does need to note conflicts to client on any add or update events

getslow command

- Loop and send all events in a calendar
  - Send one event
  - Sleep for one second
  - Send one event
  - Sleep for one second
  - Repeat until done
More Notes

• No reference server / client
  – All your code
  – Free to form up however you see fit
  – XML-like in syntax
    • Human readable
  – Feel free to add bells / whistles

Features – 10%

• Need to add in a set of features
  – Write a client in a different language
    • Java
    • Perl
    • Ruby
    • C#?
  – Add descriptions
  – Add an alarm
  – Add attendees for a meeting
  – Determine when multiple people could meet

Multicast & MPLS

Outline
  Multicast for LS
  Multicast for DV
  Protocol Independent Multicast
  MPLS
Process Groups

- Any set of processes that want to cooperate
- Processes can join/leave a group
- A process can belong to many groups
- Groups can be either open or closed
- Use multicast rather than point-to-point messages
  - Group name (address) provides a useful level of indirection
- Example uses
  - Data dissemination (e.g., news)
  - Replicated servers

Multicast Addresses

- Subrange of IP address space reserved for MC (class D for IPv4)
- IPv4: 28 bits of possible MC addresses
- Ethernet: uses 23 bits for multicast
- Mapping 28 bits onto 23 bits: 32 IP addresses map into each one of the Ethernet addresses
- Ethernet host joins IP MC group by configuring device to receive Ethernet MC address. IP at host must inspect packet if actually directed to this host

Multicast Routing: LS

- Each host on a LAN periodically announces the groups it belongs to using IGMP
- Augment update message (LSP) to include set of groups that have members on a particular LAN
- Each router uses Dijkstra’s algorithm to compute shortest-path spanning tree for each source/group pair
- Each router caches tree for currently active source/group pairs
Multicast Routing: DV

• Reverse Path Broadcast
  – Each router already knows that shortest path to S goes through router N
  – When receive multicast packet from S, forward on all outgoing links (except one it arrived on), iff packet arrived from N
  – Eliminate duplicate broadcast packets by letting only “parent” for LAN (relative to S) forward
    • shortest path to S (learn from distance vector)
    • smallest address to break ties

DV (cont)

• Reverse Path Multicast
  – Goal: prune networks that have no hosts in group G
  – Step 1: determine if LAN is a leaf w/o members in G
    • leaf if parent is only router on the LAN
    • determine if any hosts are members of G using IGMP
  – Step 2: propagate “no members of G here” information
    • augment (destination, cost) update sent to neighbors with set of groups for which this network is interested in receiving multicast packets
    • only happens when multicast address becomes active

Protocol Independent Multicast

• PIM: sparse mode (PIM-SM) and dense mode
• Routers join/leave groups: Join/Prune messages
• Rendezvous Point (RP) for each group
• Shared trees and source-specific trees
Multiprotocol Label Switching

- **MPLS:**
  - enable IP capabilities on devices that do not have capability to forward IP datagrams in normal manner.
  - forward IP packets along "explicit routes".
  - support certain types of virtual private network services.

Destination-Based Forwarding
Label Distribution Protocol

Label Switching Routers

(a) ATM cell

(b) "Shim" header (for PPP, Ethernet, etc.)

PPP header | Label header | Layer 3 header

ATM cell

HEC

Layer 3 header

PPP header

Label

GFC | VPI | VCI | PTI | CLP | HEC | DLCI

PTI | CLP | VPI | VCI | GFC

PPP header | Label header | Layer 3 header
Benefits

Explicit Routing

- “Fish” network
- Resource Reservation Protocol (RSVP)