CS640: Introduction to Computer Networks Notes

Jack Truskowski

December 8, 2017

Contents

1 9/7/17 History of Communication

- Python environment called Switchyard

1.1 Communication

- Spoken language ~ 100k years ago
- Written language ~ 8000 BC
- Carrier Systems ~ 2400 BC
- Telegraph/Morse - 1837
  - (Virtually) Instantaneous Communication
- Telephone - 1875
- Film/TV/Radio - 1894/1927/1896

1.2 Internet History (Machine-to-Machine Communication)

Telephone Network -> Circuit switching

- C. Shannon ('40s)
- circuit switching have to have a set of wires that connect you and only you to remote entity
- circuit switching is not a robust infrastructure (can't handle small disruptions)

1905s: 3 groups working on 'packet switching'

- Beren, Kleinrock, Davies
- Makes system more robust to outage
- Greater efficiency

1967 - ARPHnet (by the Government)

- plan for the first packet switched network

1969 - First 4-node ARPAnet (UCLA, SRI, UCSB, Utah)

- V. Cerf, R. Kahn, L. Roberts

- First application = Rlogin (remote login)

1972 - 15 nodes, RFC (request for comment) 001 = Network Control Protocol

- Facilitates consistency and inter-operability among a community of participants
- ICANN -> John Postal
- R. Tomlinson invents **email**

1973 - Ethernet 1974 - TCP/IP
1989 - 100k nodes

- V. Jacobson enhances TCP
- Invention of DNS (Domain name system)

1991 - HTTP invented (Tim Berners-Lee) 1993 - MOSAIC (graphical browser) invented (M. Andreeson)
1999 - Napster

- Google ~2000
- Facebook 2004

2007 - iPhone

2 9/12/17 Network Basics and 3 Perspectives on Networks

Perspectives:

1. Abstract
2. Internet
3. Architectural

2.1 Basics (Abstract Perspective)

- Nodes = Devices that facilitate communication (end-node, client) eg) computers, phones, routers, switches, and more
- Links = A physical medium that facilitates the transmission of signals

Point-to-Point Network:

```
[Comp] - [Comp]
         ^ protocol (eg PPP)
```

- Full Duplex Communication: allows 2-way communication at the same time

Multiple Access:

```
[Comp] [Comp] [Comp]
  \|-/
             ^shared medium (bus)
```

- Not **scalable**

Switched:

```
            [comp]
{             |
[router] - [router] - [comp]
      \      /
      [router] - [comp]
} <-represents more infrastructure
```

- Intermediate nodes enable scaling
- More than 2 nodes can communicate at the same time
- Circuit Switching:
- Establish a dedicated path between end nodes via signaling after which communication = stream of bits
- Packet Switching:

- Data is divided into packets (discrete blocks) and sent in shared environment toward the destination using **store and forward** nodes
- Robustness
- Efficient resource utilization

Network of Networks

```
{}  {} - [comp]
  \  / <- Must enable communication between variety of networks
   {} - [comp]
```

- This is **The Internet** (a network of networks)
- **Address**: Unique identifier which enables nodes to be targeted. Internet uses 'destination-based forwarding' to send packets ^ destination address in packets
- **Forwarding** Facilitated by nodes that are not directly connected to destination nodes. ^ process based on table lookups in store and forward devices. Results in transmitting packet on outgoing link
- **Routing**: process for establishing forwarding tables in routers.
- **Multiplexing**: sharing a resource among multiple entities eg) TDM (time division multiplexing), FDM (frequency division multiplexing)
  - **Statistical Multiplexing**:

```
Inputs ->
       -> [queue+server] -> Output
       ->
```

- There is some policy for aggregating packets as they come into these devices
- Demand > capacity = congestion -> for too long... = packet loss
- Network infrastructure and protocols are designed to support applications communicating with each other -> this drives the development and demand for more capability
- The challenge is that the systems are:

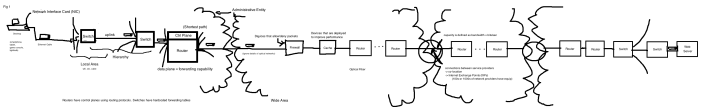
1. Huge
2. Complex
3. Dynamic

2.2 Internet Perspective

```
application -> [comp]~~~
                      ^NIC (network card)
```

3 9/14/17 Internet / Architectural Perspective

3.1 Internet Perspective



- Service providers run network administrative domains and are defined in tiers:

Tier 1) Very large providers with world wide footprint (ie AT&T) Tier 2) Single entities or providers with a more restricted infrastructure (ie IBM, Microsoft) Tier 3) Local service providers

3.2 Architectural Perspective

- Guide for implementation
  - Model for reasoning about complex systems
    - Primary abstraction = layers

    ex of layered architecture)

```
|-+|
| garments    |
| cloth       | - Stack: we move up by providing increased levels of service at
| yarn/thread |
| fibers      |
|-+|
```

- Abstract layered architecture for network:

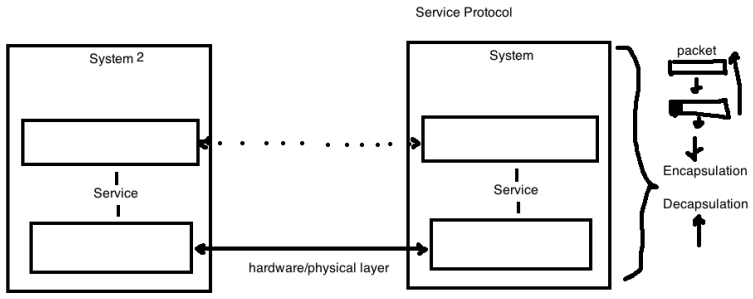
+
applications
<- Actual programs that facilitate communication
process-to-process
<- Multiplexing hosts and addressing reliability
host-to-host
<- Abstracts network complexities between hosts
hardware
<- Actual connections
+

4 9/19/17 Architectural Perspective / Performance

4.1 Primary abstraction = Layers

- Decompose complexity into manageable chunks
- Increased levels of 'service' as we move from bottom to top
- Protocols are defined at each layer
  - Protocols provide services for higher layers to communicate
- Set of layers that define the system/internet = **stack**

4.2 Protocols



- Define 2 interfaces:

1. **Service**: Defines local operations that can be performed - interface that can be performed - interface between layers on same system
2. **Peer**: Defines the form/meaning of messages exchanged between the same protocol layer on 2 different systems

4.2.1 Notes:

- Bits are only actually exchanged by hardware (at the hardware level)
- Potentially multiple protocols defined at each layer
- *NTP is a read-able RFC*
- Protocol is an overloaded term:
  - Description of how something behaves
  - Algorithms or state machine
  - Definitions

4.2.2 First Internet Architecture (OSI model)

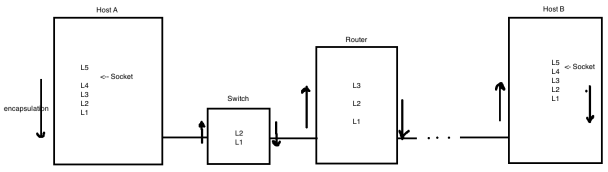
- Defined by ISO
- Open Systems Interconnection model (OSI), 7 layers

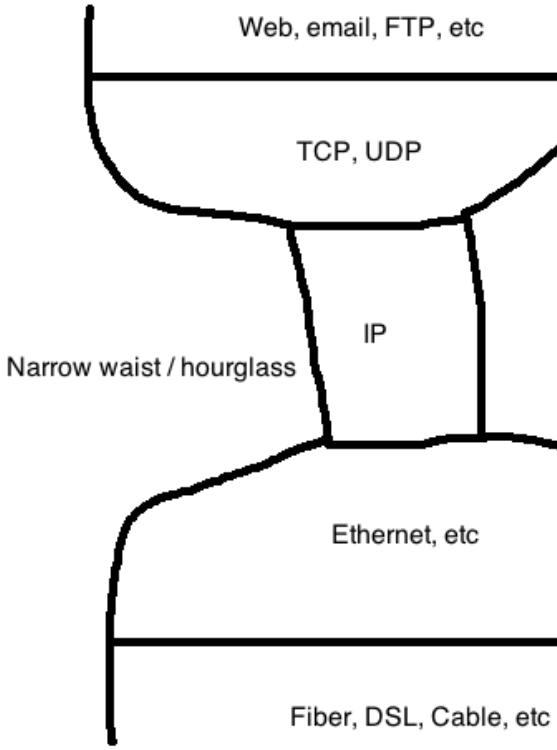
4.2.3 Internet Architecture

- **5 layer model**:

–
Layer
Name
Description
–
L5
Application
Defines interactions with users, and when to initiate/receive transfers
OSIs have additional layers here

L4
Transport
Defines logical channels between network and applications
L3
Network
Defines addressing and routing (ie IP)
L2
Link
Defines how hosts access physical media (ie Ethernet)
L1
Physical
Defines media, physical layout, and how bits are represented
–





- Layer 3 has one protocol = **Internet protocol (IP)**
- In practice, implementations don't necessarily respect layers

1. **IP Service Model**
   - Will be quizzed/tested on this
   - Gives us destination-based forwarding:
     (a) Connectionless
     (b) Packet-based
     (c) Best effort: packets can be delayed, dropped, reordered, or duplicated

4.3 Performance

- *The ability to make communication faster* -> has been a major driver of Internet technology

4.3.1 Basics:

- **Bandwidth**: The amount of data that can be transferred per unit of time

1. Link vs End-to-End:
    - Link = source to destination (basically bandwidth).
    - End to end = notion that there are destinations inbetween source & destination
        - Performance can be defined in terms of **Latency** (time to send message from one host to another)
          Latency = Propogation + Queuing + Transmit Propogation delay ~= distance / speed of light Queuing delay = [0...Queue1+Q2...Qn | n = number of hops] Transmit delay = Amount of data to send / Bandwidth
            * Propogation Delay * Bandwidth = amount of data 'in flight' or 'in the pipe' = **delay, bandwidth product**

4.4 HW1

5 9/26/17 Information Theory Basics / Simple Reliability / Physical Layer

- Review: IP enables communications between networks
- Encapsulation as packet moves down, decapsulation as packets move up the stack **headers**
- **Scalability, Robustness, Performance**

5.1 Queuing

Latency = Propogation + Queuing + Transmit

- Queuing Models and Analysis is a huge area of study

- Basic Queue:

  input->[Buffer](Server)->output ^ packet processes

  - Buffer manangement policies (ie. FCFS, LCFS, priority)
  - Server: # and speed
  - Server Delay = outgoing bandwidth
  - **Little's Law:** Mean # of jobs in system = arrival rate * mean response time *arriving jobs -> [black box] -> departing jobs*
    - Applies when jobs entering = jobs leaving (no jobs are dropped or created)
    - ex) Avg forwarding time in a router is 100 nanoseconds. The I/O rate = 100k pps. What is the mean # pkts in buffer? Solution: Mean # packers in buffer = 100,000 * 0.0001 = **10**

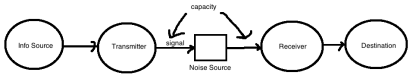
5.2 Information Theory Basics

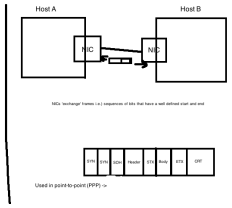
1700s: G. Boole invents Boolean Logic

- AND/OR/NOT, 0, 1

  Problem: Transmission of information over a **noisy channel** Objective: Reproduce a message at some point that was sent at another point

5.2.1 C. Shannon: 1948: "A Mathematical Theory of Communication"





-> foundation for Information Theory

Definitions: **Information** (measured in bits): The amount of uncertainty a message eliminates or only ___ uncertain to a receiver **Noise** distorts a message. Reduces information by increasing uncertainty **Redundancy:** Repetition of a message or part of a message that reduces information loss due to noise **Channel Capacity:** Amount of information + noise that can be processed per unit of time

1. Key Results:
   (a) Quantifies the average number of bits needed for communication in terms of **entropy** (quantifies uncertainty) -> **source coding theorem**
   (b) Proves that reliable communication is possible over a noisy channel if *transmit rate* < *capacity* -> **noisy channel coding theorem**

5.2.2 Handling Errors/Noise

- We should drop packets with errors as soon as possible Reasons
  1. Not useful for apps that assume reliable transfer
  2. Saves resources downstream

1. How do you know that a packet has an error? (at least one bit is flipped)
   (a) Packet = Datagram D = string of bits
   (b) Use algorithm A on D to generate code C = string of bits
   (c) Transmit D,C
   (d) Receiver gets D,C, uses Algorithm A on D to get C' and compares C and C'
   (e) The **expected noise** determiens the type of encoding that is required.
   Typical Methods of Encoding
      i. **Parity:** odd/even, 1-bit code
      ii. Checksum
      iii. **Cyclic Redundancy Check (CRC)** Need to know algorithm! Look it up in the book
         - Modular arithmetic
   (f) Error checking is done at almost ever layer in the protocol stack

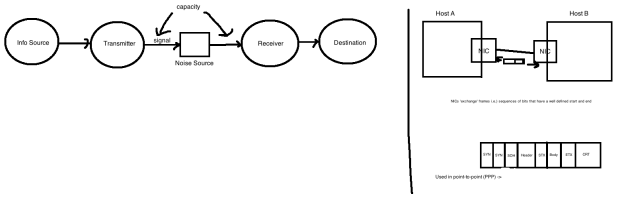
5.3 Layer 1: Physical Layer

Concerned with:

1. Characteristics of hardware and physical media

2. Signals on physical media, and framing -> where packets begin and end on physical media

3. Framing - where packets begin and end on physical media



- NICs 'exchange' frames ie) sequences of bits that have a well defined start and end
- How are start and end identified?
  (a) **Sentinel Approach**: Uses special characters to delineate start and end. SYN, STX, ETX
  (b) Byte Counting
  (c) Clock-based

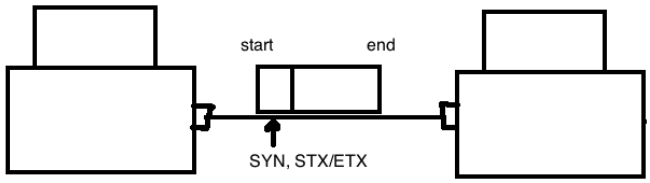
6 9/28/17 Framing

6.1 Layer 1 Continued

Framing:

1. Sentinal:

Sentinel Based Framing



1. Byte counting - SYN + #Bytes
2. Clock-based:
   - Fixed frame size
   - Includes special characters
   - Need to use timing to determine the start and end

Aspects that are included:

- Details of hardware (connectors, cables, etc)
  - Coax, Twisted pair, **Optical Fiber**
- Transmission of bits between hosts. That is how to represent 1s and 0s
- Transmission medium = copper
  - Transponder modulates voltage
    - 0 = no voltage

* 1 = voltage *How are multiple 1s and 0s distinguished?* Solution = clock
  Non-Return to Zero (NRZ): 0 = low, 1 = high voltage
  - Requires clock synchronization
  - Keep an average voltage to distinguish high and low

  NRZ-Inverted (NRZI): 0 = staying at same signal, 1 = changing signal *Manchester Encoding:* Explicit merge of clock and data, encode both 0 and 1 as transitions
  - 1 = high-to-low transition
  - 2 = low-to-high transition
  - Move back to baseline as needed inbetween clock edges

7 10/3/17 Data Link (Layer 2)

7.1 Project 1

- In the *Switchyard environment* -> enables switches, routers, etc to be built/emulated
  - Focused on sending / receiving / processing and forwarding packets
- Install VM on Linux-based system (see project homepage)
  - Switchyard + Mininet
- Do learning-switch exercise (.rst)
  - see detail on Project 1 page
- Submit program files by deadline (10/19)
- Grading: 25% = code review 25% = code does minimal subset of functions 25% = code runs correctly 25% = answering a set of questions that can be answered once the code runs

7.2 Layer 2 - Data Link

Concerned with transfer of data between adjacent hosts. Hosts are within same "local area" -> several hundred meters. There are 3 different types of channels between hosts.

1. **Point-to-Point**
    - Single or full duplex
    - Framing
    - Reliability
2. **Broadcast** = enable more systems to communicate
    - Challenge -> coordinating between hosts
        - Media Access Control protocol (MAC)
        - Include additional specifications and addressing schemes
3. **Switched**

7.3 MAC (Media Access Control)

1. Channel Partitioning
    - Time division multiplexing
    - Frequency division multiplexing
    - Code division multiplexing
2. Taking turns
    - Polling
    - Token passing

7.3.1 3. Random Access

- Nodes send/receive in no specific order
- Challenge: managing contention (ie. when two or more nodes want to send at the same time)

7.4 Aloha Network

Packet radio in late 60s / early 70s

- Send immediately
- Receivers will send an ACK

- If sender hears that another node is transmitting, then rest and resend (since original was assumed lost)
- If sender does not receive ACK within some time period, rest and resend
- Solves contention, but is inefficient

7.5 Ethernet

- Invented by Metcalf -> '73
- Spans layer 1 & 2
- Bus-based
  - Also hubs
  - Bridged Environments
  - Switched (Provides virtual point-to-point communication channels)

7.5.1 Ethernet Frames

- Well defined
- Include:
  1. Preamble
  2. Src/Dst addresses (48 bits)
  3. Payload
  4. Flags
  5. CRC Checksum
- Service: Connectionless and unreliable

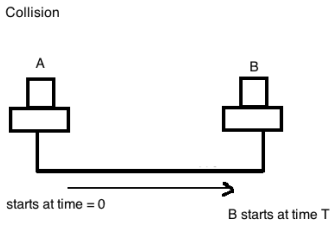
8 10/5/17 Ethernet MAC, Interconnects

8.1 Ethernet

8.1.1 802.3 ethernet protocol

ˆ wire-line ethernet *on quiz/exam!!*

- 10Mbps -> ie) it takes 0.1 μ s to signal 1 bit
- **MAC** = carrier sense multiple access colision detect (CSMA/CD)
    1. If the line is idle, send immediately, then wait for 9.6 μ s between frames (interframe gap)
    2. If line is busy, wait until it's clear + 9.6 μ s
    3. If a collision is detected, send jam signal (32-48bits), do *exponential backoff*
        - purpose of jamming signal is to ensure/reinforce that all nodes know there was a collision



ˆ Problem: ensuring the A sees collision B's message gets to A at time = 2T Solution: Be sure that A is still transmitting at time = 2T

- 802.3 specifies that 2Tmax = 51.2 μ s
    - Thus at 10Mbps this means min frame size = 512 bits (64B)
    - This also implies max length of an Ethernet bus segment = ˜2500m
    - Frame header = 18B
    - If data $<$ 46B, add padding
- Collision domain = CSMA/CD network where there will be a collision if 2 nodes transmit at the same time

8.1.2 Exponential backoff

1st collision: chose k from {0,1} then delay k*57.2 μ s 2nd collision: chose k from {0,1,2,3} then delay k*57.2 μ s ... 10th collision: max -> notify the higher layers that was unable to send frame

8.1.3 Ethernet efficiency

~ \(1/1+(5*tprop)/ttrans))

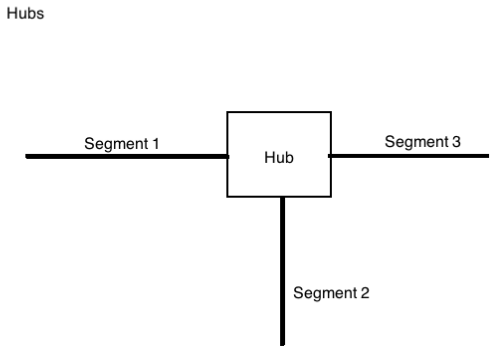
- larger frame in smaller network = more efficient network
  - *Quiz question*: how do things change if we move from 10Mbps to 100 Mbps

8.2 Ethernet Interconnects

Problem: How do we extend or interconnect Ethernet segments?

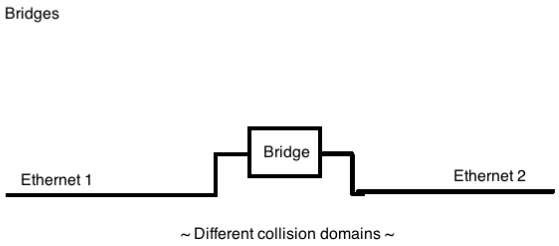
8.2.1 Interconnection devices

1. **Hub** - most simple ethernet interconnect which *does not* extend the



collision domain

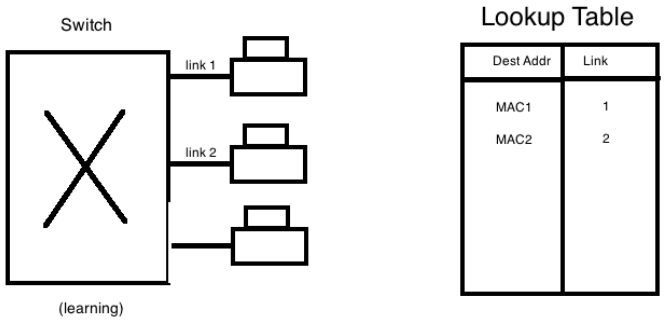
2. **Bridge** - device that connects together 2 different collision domains



- Collision domains are separate, bridges forward frames based on destination addresses
  - Bridges build forwarding tables or through fixed configuration
- Bridgin can enable larger LANs. However, managing large LANs depends on another protocol -> **Spanning Tree** which organizes LANS in a tree that assures a loop-free hierarchy
- Makes forwaring decisions based on Ethernet / MAC addresses

9 10/10/17 Switching, Wifi, Layer 3

9.1 Ethernet Switches

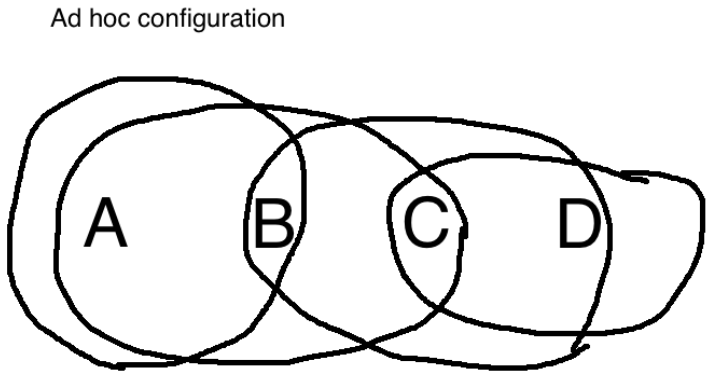


- Overcome the limitation of bus/hub by isolation collision domains to 2 hosts. Forwarding decisions are based on destination MAC addresses
- Basic function is to lookup destination address in a table and forward packet/frame on associated link/segment
- Switches build tables automatically:
  1. Empty Table
  2. If dest. addr. is NOT in table, forward frame on all links (except for the link that sent frame)
  3. When a frame arrives with source address that's not in the table, create new entry for addr/link
  4. Delete table entry if no frame is received from source after some timeout period

9.2 Wireless LANs

- Transmit / Receive data via antennas

- **802.11** = Set of standards for wireless LANs
- Wireless networks are different:
  - It's hard to transmit and listen at the same time -> due to antennas
  - Carrier sense is weaker
  - The air is not a perfect broadcast environment



- When B transmits, A and C can hear
- When A or C transmits, B can hear

9.2.1 Problems

- If A transmits to B, C cannot hear, so it will send to B at any time
  - This can corrupt the message A is sending = **The hidden terminal problem**
- If B is sending to A, C can hear it so it won't transmit. But, it could send to D since this transmission would not interfere with A's reception = **The exposed terminal problem**
- These problems are particular to carrier sense wireless ad hoc LAN

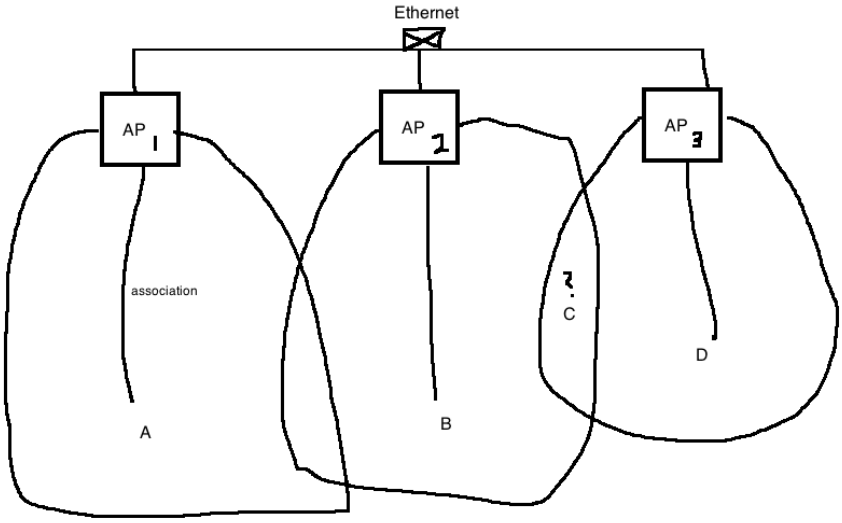
9.2.2 Solution: Multiple Access with Collision Avoidance (MACA) protocol*

- Solution to these problems
- Basically a connection setup protocol
    1. Before sending data, a sender transmits **Request to Send (RTS)** frame that includes a duration
    2. Receiver sends **Clear to Send (CTS)** that echos the duration
    3. Receiver sends ACK on successful receipt of frame, other nodes wait for this
- The CTS solves hidden terminal by informing nodes within range of the receiver how long they need to be quiet
- Any node that receives RTS but does not receive CTS knows that they are not close enough to the receiver to interfere -> solves exposed terminal

If there is a collision on FTS due to carrier sense, do random exponential backoff

9.2.3 Access Points

- Today's WiFi(802.11) LANs are facilitated through access points (APs) that acts as **switches**. APs in a local area are connected via wireline Ethernet



1. Association Protocol
   (a) WiFi node scan for access points by sending probe frame
   (b) All APs that receive probe frames send Probe Response Frame
   (c) Node selects AP with the strongest signal and sends Association Request Frame
   (d) AP responses with Association Response frame

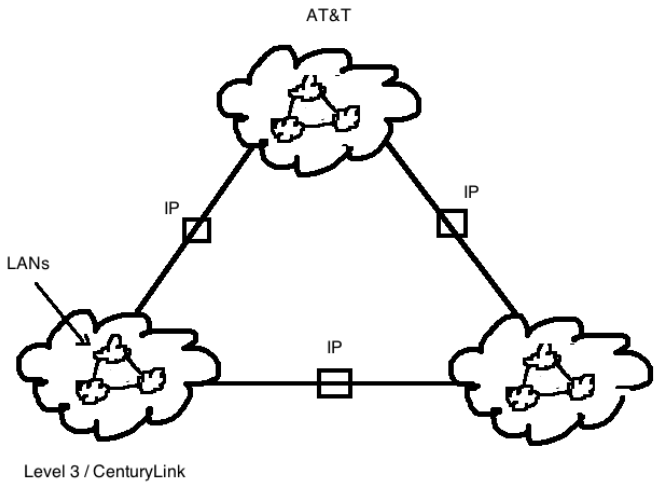
9.3 Layer 3 - Network Layer

- One of the most important aspects of Layer 3 - the one that makes it the narrow waist - is the unified addressing scheme (ie **Internet Protocol Addresses**)

10 10/12/17 Network Layer 3

- Addressing - Single scheme
- Routing protocols
- Routers
- **Major layer 3 activity**: move packets between networks

10.1 Layer 3 Architecture



- Autonomous System (AS) = Independent administrative domain <- ISPs

10.2 IP Addresses (IANA)

- 32-bit numbers organized in *dot notation*
  - ie: 192.128.65.80, each dot seperates 8 bit portion
  - 4 billion possible addresses
  - IPv4

10.2.1 Address allocation

- Address allocation to networks is a very big deal
  - Networks are defined by their address space
- Original IPv4 address space allocation was "classful"

Class	Address Space
A	[0(1bit),Network(7bit),Host(24bit)]
B	[1(1bit),0(1bit),Network(14bit),Host(16bit)]
C	[1(1bit),1(1bit),0(1bit),Network(21bit),Host(8bit)]
D	110... Multicast
E	1111... Experimental

- Routers examine the network portion of addresses to make forwarding decisions
- Classful allocation presents a number of scalability challenges:
  - Inflexible
  - Too many networks
  - Plus, we may not be able to manage large networks on Layer 2 by itself

10.2.2 Subnets

- Enables a single allocation of IP addresses to be divided into smaller 'networks'
  - Allows us to do routing using layer 3 packet forwarding in an administrative domain
- Subnets arrange for an extension into the host portion of an address:

```
 16bit  16bit      16bit      8bit      8bit
[ Net ][ Host ] -> [ Net ][ Subnet ID ][ Host ]
                       net + subnet ID = subnet number
```

- Subnets are identified by bitwise AND of IP address and the **Subnet mask**
  - Ex. 255.255.255.0 would expose the subnet in the prior example
  - This means that forwarding table must include subnet masks Forwarding table entries: $<$ subnet number, subnet mask, link $>$

10.2.3 Supernetting

- Goal: more flexibility in address space allocation. Supernet says allocate address space on powers of 2. Classless Interdomain Routing Addresses (**CIDR addresses**)
- CIDR indicates network addresses using "/"
  - ie) class C = /24 class B = /16 class A = /8
- CIDR is used in forwarding tables by allowing for adjacent networks to be "combined"

Net	Link
Class B1	Link A
Class B2	Link A

goes to...

/15	Link A

ON QUIZ How do packets get to hosts? Answer: build a table that maps IP addresses to MAC addresses associated with hosts.

10.2.4 Address Resolution Protocol (ARP)

- A dynamic protocol that operates by:
  1. Switch begins by looking for entry in table (IP address / MAC)
  2. Broadcast request over a LAN
  3. Node with IP address responds with $\text{ARP}_{\text{response}}$
  4. Each entry has a time to live

11 10/17/17 DHCP, Routers, Routing

11.1 Dynamic Host Control Protocol (DHCP)

- Enables hosts to have an address assigned for the local network
- Hosts send a special address request packet to address 255.255.255.255 and listens for a response at layer 2. Response comes from local DHCP server

11.2 Router Design

- Routers operate at layers 1-3 of the protocol stack. Physical hardware that ranges from low-cost home devices to "core systems" that cost $$$. The primary task of routers is to make (destination-based) forwarding decisions - using destination IP addresses <- network portion or subnet
- The other task of routers is to participate in **routing protocols** that establish local *fowarding tables* (unlike switches which use learning)

11.3 Router Types

Simple Routers:

- Single board (running linux)

[] [][][][]

- Uplink + 4 local ethernet ports / 1Gbps
- Limited management/config
- Limited protocol support
- Low cost

Access/Distribution Routers:

- Single board (often linux)
- Stackable/rack mount devices

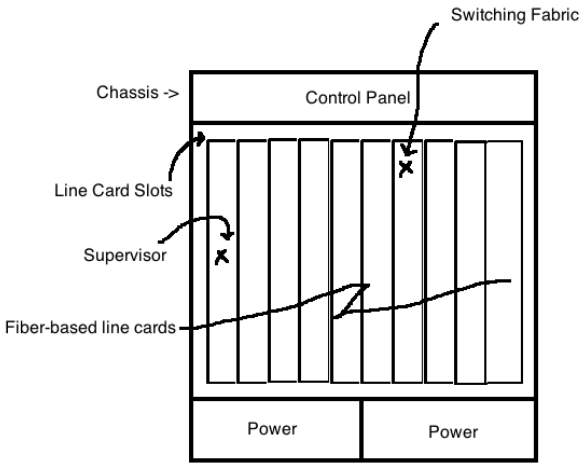
[-][-][-][-][-][-][-][-][-][-][-][-]

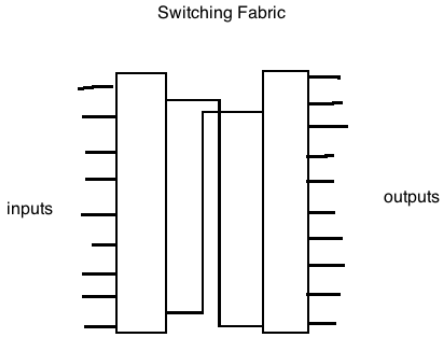
- Uplink + 48 ports / 10Gbps
- Full set of management/config capabilities
- Moderate cost

Core/Backbone Routers:

- Focus = transmission of large amounts of data over long paths
- Full or multi-rack systems

- Specialized ASIC's (application specific integrated circuits) running proprietary OS's
- Hundreds of ports/up to *400 Gbps*
- Hardware layout is focused on reliability and cooling





- Logical organization of routers:
  1. Control plane
     - configs and routing protocols
  2. Data plane
     - destination-based forwarding

11.4 Layer 3 Functions (cont)

Include:

- Specifying IP packet format
- Fragmentation / reassembly
- Error reporting
- Routing to establish forwarding tables

11.4.1 IP Packet Format

(see fig 3.16 in textbook)

- Version (4,6)
- Length of header
- Type of service
- Total length (header + payload)
- **IPID/flags/offset**
- TTL (time to live)
  - Each router it encounters decrements the TTL
- Protocol
- Checksum (header)
- Src/Dest Addresses
- Options

11.4.2 Framentation and Reassembly

- Router designers may select different sizes for pack max size. To facilitate this, routers should be able to chop up larger packets into small packets
- The resulting 'fragments' are identified via IPID and offset
- Fragmentation is a 'slow-path' process, thus we do MTU **(Maximum Transfer Unit)** discovery in TCP
  - If the packet is too large for a router, it sends an error message back

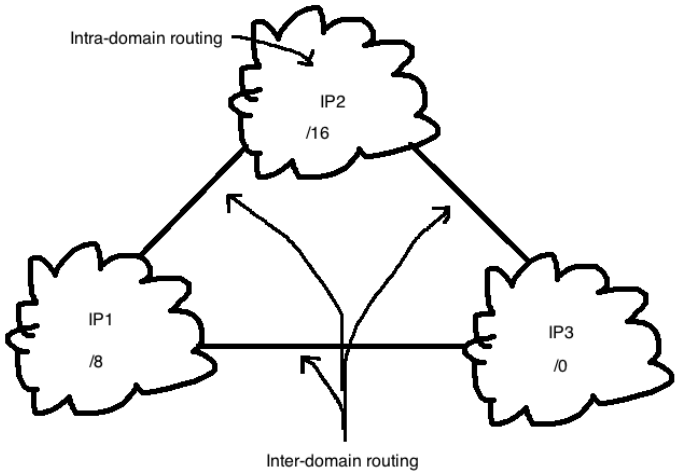
12 10/19/17 Routing

Quiz 2 on Tuesday!

12.1 Internet Control Message Protocol (ICMP)

- Designed to provide feedback from the network about status
- 13 message types
  - Ex: TTL = 0, Traceroute Ex: Echo, Ping

12.2 Routing

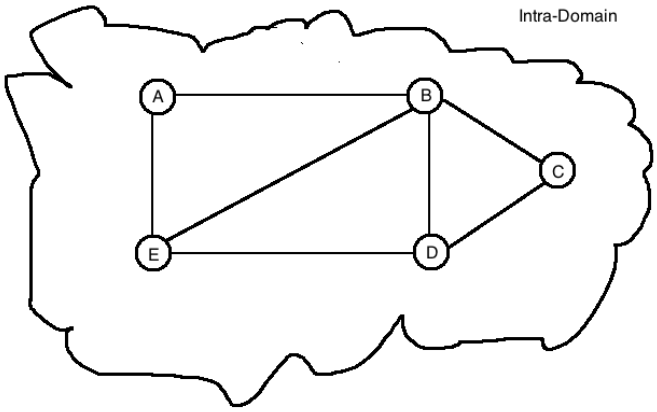


Routing = process that is used to establish forwarding tables in routers.

12.2.1 2 Levels of Routing

1. Intra-domain = routing within a domain (RIP, OSPF)
2. Inter-domain = routing between domains (BGP - don't worry for quiz 2)

12.2.2 Intra-Domain Routing



Methods for establishing paths:

1. Circuit-based (ie telephone)
2. Source-routing
3. Datagram / connectionless

1. Circuit-based
   (a) Establish virtual circuits between src and dest prior to sending data (using signaling protocols)
   (b) Data and nodes use VC identifiers to forward packets

   Pros: Good quality?
   Cons: Poor utilization?

2. Source routing
   - Packets have all info required to move to destination
   - Source must collect hop/path information out of band
   - Each packet has all hops from end-2-end

Pros: User control
Cons: User control!!

3. Datagram/connectionless
   - Each packet is forwarded independently based on destination address
   - Hosts don't know if the destination is reachable
   - Challenge:
     - Find 'lowest-cost' path between source and destination
     - This implies that a cost metric is assigned to each link
     - Easy if network is static, hard in dynamic network (ie the internet)
   - This implies the need for an algorithm that establishes shortest paths quickly and efficiently
     (a) Distance Vector Routing
     (b) Link State Routing: Djikstra's algorithm -> OSPF routing (open shorest path routing)

4. Distance Vector
   - Based on local computation by neighbor nodes. Nodes construct and send 1-d vector of distances to all other nodes
   - Belman-Ford algorithm:

```
Calculate D[i,j][h] for all i != j
          ^dist  ^hops
h=0, D(i,j)[0] = {0 if i=j, inf otherwise}
h=1, D(i,j)[1] = min/k=neighbors{d(i,k) + D(k,j)) for i != j}
h=2, ^
...
```

   - Objective is to converge on shortest paths
   - Nodes receive DVs from neighbors periodically or by trigger
   - Everytime a distance vector is received from a neighbor, recalculate distances
     - This can trigger a forwarding table update

Forwarding Table:

(a) Destination

(b) Cost

(c) Next Hop (link that packet will be sent out on)

ie)

Dest	Cost	Next hop
B	3	B
C	∞	-
D	∞	-
E	1	E
F	6	F

- Nodes send DVs to neighbors

```
if cost neighbor + cost to dest < cost dest:
    update entry for dest
```

- **IMPORTANT** Routing convergence

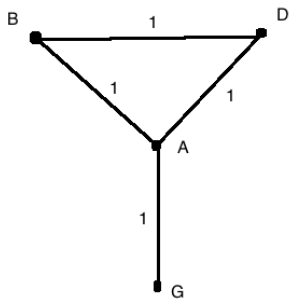
13 10/24/17 Link State Protocol

13.1 Cont...

- Distance vectors change because:
  1. A link goes down (edge cost = ∞)
  2. A configuration changed
- Making a link cost low results in more traffic over a particular link
  - ie) A high-bandwidth link can handle more traffic
- Only forward distance vectors when there is a change (or some time has elapsed)

13.2 Count to Infinity

13.2.1 The Problem



Distance Vector:

DV	A	B	D	G
A	0	1	1	1
B	1	0	1	2
D	1	1	0	2
G	1	2	2	0

1. Assume link A-G goes down. So, A advertises distance to G=∞: A:[0, 1, 1, ∞]
2. A receives B's DV: A:[0, 1, 1, 3]
3. A sends this DV
4. B recalculates DV: B:[1, 0, 1, 4]
5. This leads to the potential for loops, instability, and long convergence time

13.2.2 The Solutions

1. Set hop/exchange limit to **16**
2. Split horizon: don't send routes learned from neighbors back to those neighbors
3. Poison Reverse: Return routes to neighbors set to ∞

13.2.3 Routing Information Protocol (RIP)

- Standard implementation of a DV protocol

13.3 Link State Intra-domain Routing (Djikstra's Algo + Reliable Flooding)

- Uses Dijkstra's shortest path algorithm to establish forwarding tables. Assumes link state/costs for all links are available at all nodes before calculation

  Assume:

  1. Each node knows its own link state/cost
  2. Each node can communicate with its neighbors

13.3.1 Djistra's Algorithm

N = # of nodes in G l(i, j) = link cost between nodes i, j SPT = nodes comprising shortest path tree S = source C(n) = cost of path from S to n

1. Initialize SPT = {S} for each node not in SPT, C(n) = l(S,n)
2. while(SPT <> N): SPT = SPT U {w} such that C(w) is minimum for all w in (N-SPT)
   for each n in (N-SPT): C(n) = min(C(n), C(w) + l(w, n))

Ex)

1. SPT = {B}
2. Thus C(E) = 1, C(A) = 3, C(C) = 4, C(others) = ∞
3. SPT = {B, E}, now adjust costs

4. C(A) = min(3, 1+1) = 2
   C(D) = min(∞, 1+1) = 2
   C(F) = min(∞,)

14 10/26/17 Cost Metrics, Inter-Domain Routing

14.1 Quiz 2:

1. Problem = distinguish consecutive strings of 0s and 1s MRZ, INRZ, Manchester – *2.* RTT = 46.4us = 2Tmax @ 10Mbps, bit time = 0.1us Thus, min packet size is 46.4 + 48bits/0.1 = 512 bits @10Gbps = 46.4 + 48/0.001 = 46,448 Drawback = have to add padding for small data – inefficient! – *3.*

1. Switched ethernet builds forwarding tables and makes forwarding decisions based on these tables

2. No collision domains in switched ethernet, 802.3 requires collision domains

– *4.* Classes: A: [0, 7 - Net, 24 - Host] B: [1, 0, 10 - Net, 16 - Host] C: [1, 1, 1, 24? - Net, 8 - Host] Limitations:

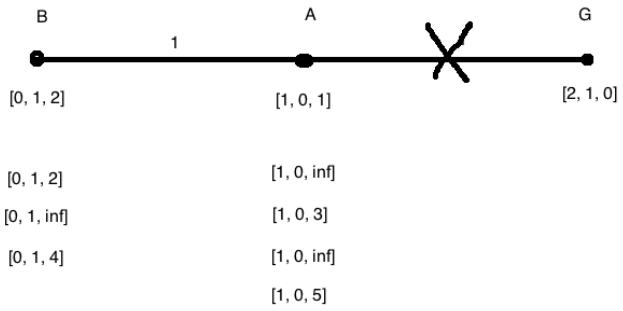
1. inefficient

2. inflexible with respect to routing within a domain

– *5.* Octagon

- No distance vectors are exchanged

14.2 Distance Vector Routing



In distance vector routing, whenever you receive an update from a node that is indicated as the next hop, you may have to raise the cost

14.3 Link State Routing

Link State = reliable flooding + Dijkstra's

- In the end, a forwarding table with the next hops is produced

Differences to DV routing:

1. Assemble a complete view of the network before executing a routing protocol
2. Use different protocol (Dijstra's)

14.3.1 Reliable Flooding

= Process for transmitting link-state packets throughout a network

- Format:
  - ID of initiating node

  - List of directly connected nodes
  - Link weights
  - Sequence #
  - TTL

- Link state change initiates reliable flooding

  1. Initial transmission of link state packets to neighbors
  2. Receiving node looks up src/dst record in map table
  3. If record *is not* in table, add it and broadcast to all neighbors (except original neighbor)
  4. Else if sequence # in the table is lower, replace entry and broadcast
  5. Else if sequence # in table is higher, do nothing
  6. Else if sequence # in table is same, do nothing

- This enables a consistent view of shortest paths to be established
- Reliability is enabled by:

  1. Hop by hop acknowledgement of link-state packets
  2. Protect update informationin packets via checksum
  3. Encryption
  4. Remove old (stale) updates via TTL

14.3.2 Open Shortest Path First (OSPF)

= instance of Link State Routing protocol

- Advantages over RIP:
  - Fast convergence
  - Loop free routes
- No count to infinity problems - no reliance on neighbor computations

14.4 Link Costs

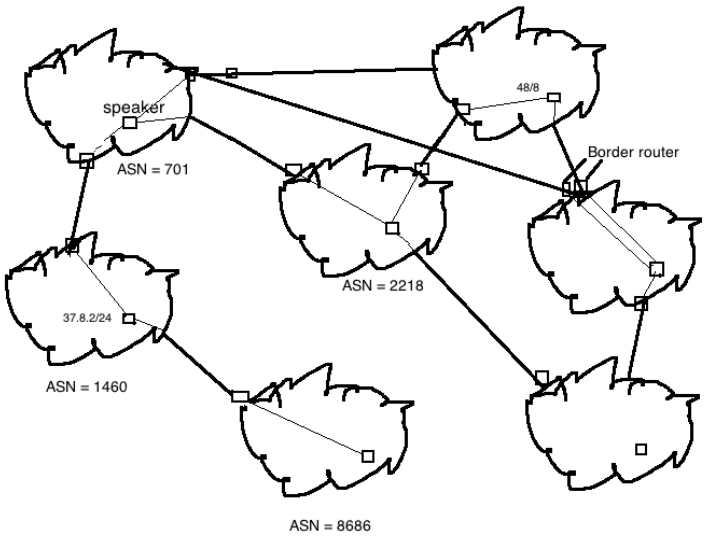
- 'Low cost' links are likely to carry traffic
- Static Metrics (ie. hop(1), **bandwidth**, distance) *Pros:* simple *Cons:* inflexible
- Dynamic Metrics (ie. latency, packet loss, volume) *Pros:* adapts to provide better performance *Cons:* control - complicated to manage
- Original cost metric: proportional to queue size
  - Problem: Moves packets towards short queues, ignores important information (ie. bandwidth, latency)
- Next metric: costs proportional to "average delay"
- In practice, smart configuration of static metrics works about the same as configuration with dynamic metrics
  - Most administrators prefer static metrics

14.5 Inter-domain Routing (Not on midterm)

- **Exterior Gateway Protocol** = original inter-domain routing protocol
  - Assumed an "internet backbone"
- Today's internet is organized by interconnections between **autonomous systems**
  1. Stub AS = small, single service provider to the rest of the internet
  2. Multi-homed = various sizes, multiple upstream providers
  3. Transit = provide local, = size and transit connectivity

15 10/31/17 Border Gateway Protocol

- Routing tends to be pretty fixed in relatively static networks
- For Inter-domain routing, there are two objectives:
  1. No loops
  2. Policy expression



15.1 Border Gateway Protocol (BGP)

- Developed in 1989
- BGP.4
- All networks have a **BGP speaker**
  - A router configured to interact with other BGP speakers in connected networks. Connections are either:
    1. Customer -$$$-> <-service- Provider
    2. Peer, connection is mutually beneficial
       * ie) UW Madison and other Big 10 schools

15.1.1 Information Exchange

- Speakers exchange information as follows:
  1. **Announcing/removing network addresses (uses CIDR)**
  2. Transit-only: Announce other reachable networks
  3. Full paths and attributes

  - Autonomous System = domain (AS)
  - Full path = sequence of ASNs defined by Path Vector Routing

- Information is exchanged when routes change:
  1. A destination prefix (set of IP addresses) becomes reachable
  2. A better path to a prefix becomes available
  3. The best path to a prefix becomes unreachable
  4. A destination prefix becomes unavailable

- BGP exchanges are based on Path Vector Routing. Similar to DV routing, but does not include a provably optimal algorithm for establishing paths

- Path vectors are exchanged as follows:
  1. Local network 'announces' the availability of a prefix by inlcluding the prefix and appending its ASN in a BGP update packet - send this update to all 'peer' (directly connected) speakers
     - Ex: path(X (address prefix), ASNY) + policy info
  2. Peer BGP speaker receive the announcement and then decide how to proceed -> typical case is to extend path by adding the local ASN and propogating to peers. -> *path(X, ASNY, ASNZ)*
  3. At next AS, path(X, ASNY, ASNZ, ASNA)
  4. Loop free routes are ensured by examining the set of ASNs in a path vector. If you see your own/local ASN in the path, do not propogate the vector

- Simple propogation of a path vector is is not the only choice
  - Networks can choose to filter updates
  - Routes can be aggregated via CIDR
  - Propogate a less direct path (in terms of AS hops)

15.1.2 Challenges

1. Number of networks (50k+ networks, 500k+ prefixes)
2. Different interpretations of policy and metrics
3. Need for flexibility
4. Need for trust

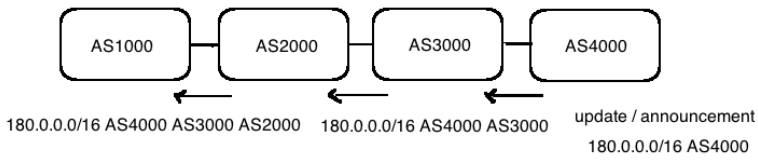
15.1.3 Operation:

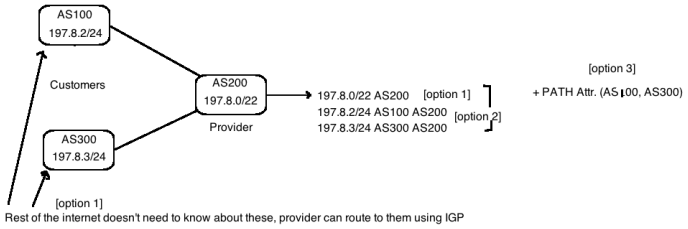
1. Establish session
2. Exchange all active routes
3. Exchange updates
4. Repeat

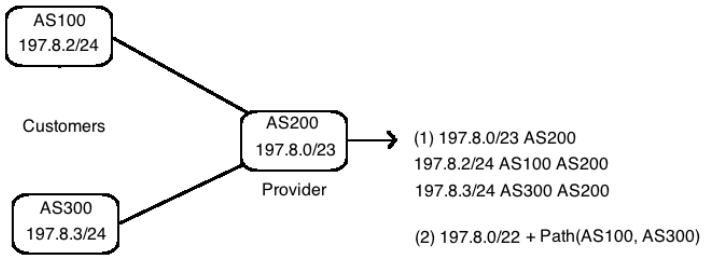
16 11/2/17 BGP Control, Multicast

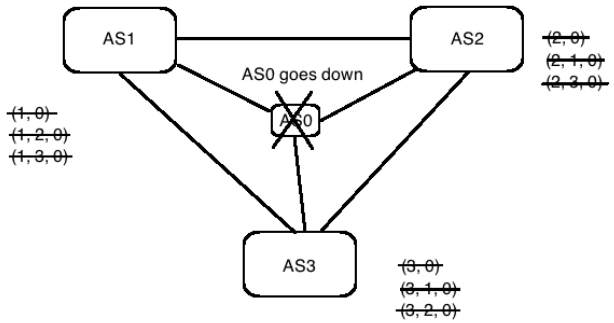
- BGP is used universally to construct a network of networks
- Convergence matters in BGP, but there are challenges related to scale and policy
- Also, **path exploration** is standard during updates
  - This refers to intermediate paths that can be used before convergence

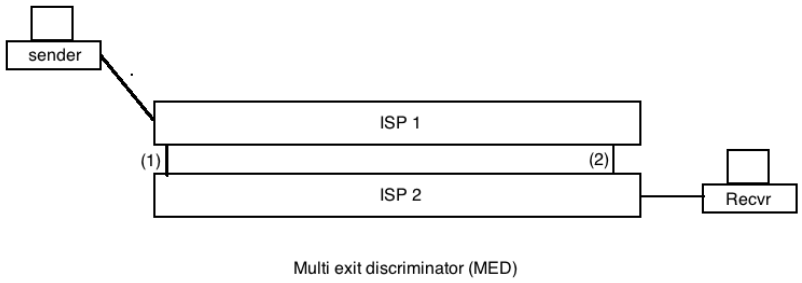
16.1 BGP Examples




AS100
197.8.2/24
Customers
AS200
197.8.0/22
Provider
AS300
197.8.3/24
197.8.0/22 AS200
197.8.2/24 AS100 AS200
197.8.3/24 AS300 AS200
[option 1]
[option 2]
[option 3]
+ PATH Attr. (AS100, AS300)
[option 1]
Rest of the internet doesn't need to know about these, provider can route to them using IGP


AS100
197.8.2/24
Customers
AS200
197.8.0/23
Provider
AS300
197.8.3/24
(1) 197.8.0/23 AS200
197.8.2/24 AS100 AS200
197.8.3/24 AS300 AS200
(2) 197.8.0/22 + Path(AS100, AS300)





Multi exit discriminator (MED)

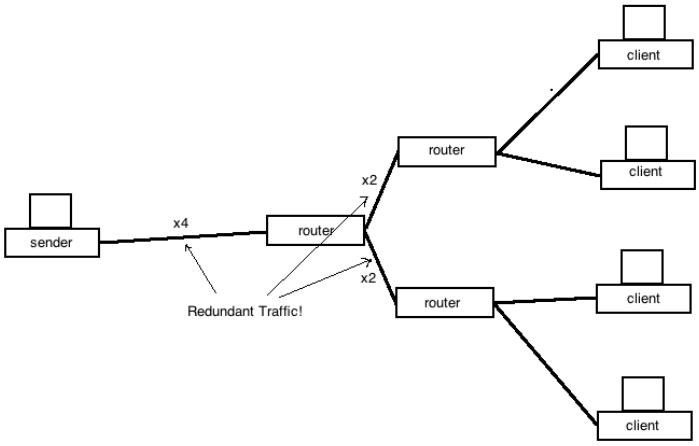
16.2 BGP Convergence Times

- Failover = ~3min

- Better path = ~1min
- Worse path = ~2min

16.3 Multicast Routing

- Observation: 1-to-many communication is inefficient in a unicast environment (ie. a single connection between server and *each* receiver)



To achieve an efficient transmission of data:

1. New method for routing
2. A way to duplicate packets
3. New addressing

Multicast routing was invented in 1989 -> RFC1112

- Senders transmit to a host group i.e. an IP address in the multicast space
- Members of a host group = clients that have indicated interest in content

- Routing must support the connection of senders to receivers (unicast must also continue to be supported)

Internet Group Management Protocol (IGMP) -> clients send IGMP packets upstream to their first hop router indicating interest in a multicast host group

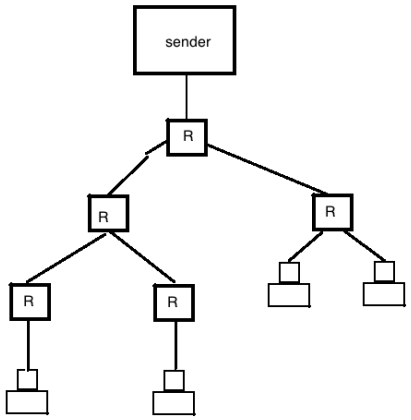
17 11/7/17 Multicast, IPv6

17.1 Multicast (cont)

- Clients use **IGMP** to notify their upstream routers that they want to receive content from a specific multicast channel (i.e. a multicast address)
- Changes in layer 3 infrastructure to support multicast:
  1. IGMP
  2. Routers must be able to duplicate packets
  3. Mechanism that sets up paths between server and client
     - Protocol + separate multicast forwarding table

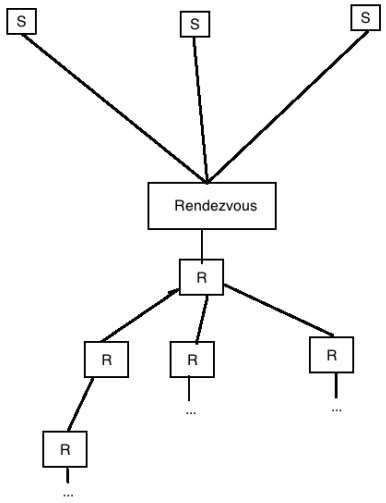
17.2 Routing Protocol Configuration

- There are two different configurations that are used in routing protocols:
  1. Source tree: Root of routing tree is at the server
     - Requires more memory in routers
     - Results in optimal paths
     - Good for situations where you have a small number of



senders/large number of receivers

2. Shared Tree: Root of routing tree is a special router called a rendezvous point
   - Requires less memory in routers
   - Can result in suboptimal paths
   - Good for environments with many senders



17.3 Protocol Independent Multicast (PIM)

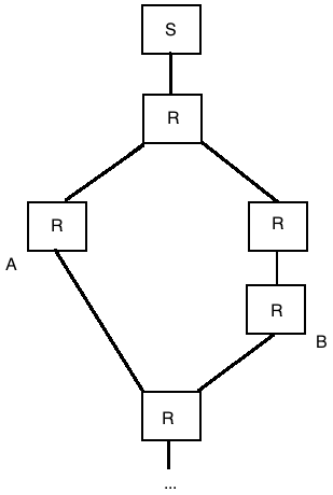
- Assumes that there is an underlying unicast routing protocol
- PIM operates in dense mode (DM) or sparse mode (SM)

17.3.1 Dense Mode

- Assumes all clients want to receive content. Clients must opt out
- Source tree is created via Reverse Path Flooding or via **Reverse Path Broadcast**

17.3.2 Reverse Path Broadcast

- Define parent-child relationships between routers



- R is a parent (ie it is on the forwarding path) if it has a minimum (lowest cost) path to S. In the example B is not a parent to S so it will not forward packets in the source tree
- Requires a global view of the network
  - Link state routing gives us a global view

- **SHOULD BE ABLE TO SET UP A SOURCE TREE FOR A NETWORK**

17.3.3 Sparse Mode

- Assumes clients will opt in on content they want to receive
- Utilizes a shared tree
- Requires a **Rendezvous Point (RP)** which is configured for a set of servers
- SM uses "explicit join" to establish paths to the RP. The request from the client (IGMP) triggers a series of path requests until the RP is encountered

17.4 IPv6

- 1991
- (IPv5 = Internet Stream Protocol, RFC1190)
- Basic limitation in IPv4 = 32 bit addresses
- v6 development focused on a number of issues: IPv6 features:
  1. 128-bit addresses
  2. Support for real-time QoS (quality of service)
  3. Security (IPSEC)
  4. Autoconfiguration
  5. Enhanced mobility support
  6. Multicast support
  7. Protocol extensions

18 11/9/17 IPv6, Mobile IP

18.1 IPv6 Addresses

- Allocation are classless
  - We continue to use "/" notation

- Standard representation of IPv6 address = 8, 16-bit values separated by colons

Allocations:

- unicast
- multicast
- any cast
- The IPv6 header was designed to be simpler than IPv4 - fewer fields and no options. Extension headers enable extra information to be communicated at layer 3
- The IPv6 header does not include checksum, no header length, no support for fragmentation

18.1.1 Unicast

- Indicated by addresses that begin with:
  - 001: There is structure in the remained of the allocation
    - First 64 bits = routing prefix
      - Routing prefix has structure that is meant to reflect the global network hierarchy. Prefix = Registry, Provier, Subscriber:
      - Registry = 5-bits
      - Provider = n-bits
      - Subscriber = 56-n bits
      - Plus, prefix = 48 bits (or more) of routing/network ID, followed by 16 bits (or less) of subnet ID
    - Followed by 64-bit interface identifier

18.2 V4 to V6 Transition

- Native IPv6 requires all routers to support this version of the protocol in order to transmit packet natively. Otherwise encapsulation / decapsulation is required
- BGP for v6 will be required to establish paths. v6 traffic depends on:

1. Content providers must make content available on v6
2. Clients must request content on v6

- Clients now operate "dual stack" hosts. This means that hosts request both v4 and v6 addresses. If both addresses are returned, host decides which one to use

18.3 Mobile IP

- RFC 3220
- IP routing = moving packets from source to destination network. What if we want to enable hosts to move without changing their IP address?
- Mobile IP enables a host to move between networks without having to change IP addresses
- Requires changes to infrastructure but not to host

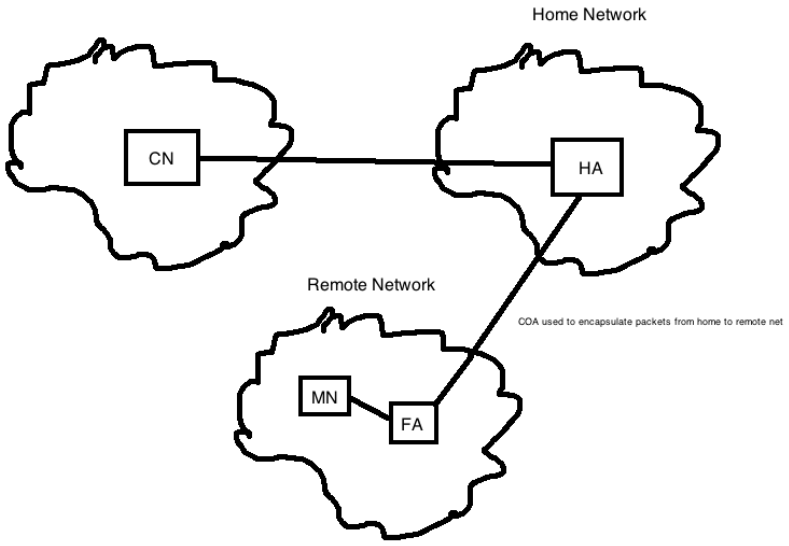
18.3.1 Entities

- **Mobile Node (MN)** - Assigned an IP that will not change (from its home network)
- **Home Agent (HA)** - Router in MN's home network. Will forward packets to MN when it is out of home network
- **Foreign Agent (FA)** - Forwards packets to MN when MN is in its network
- **Care-of-Address (CoA)** - Address that identifies the MN's current location (usually the IP of the FA)
- **Correspondent Node (CN)** - Host that MN is communicating with

18.3.2 Services

1. Agent discovery: HA and FA announce their presence via ICMP
2. Registration: MN registers its COA from the remote network with the HA
3. Encapsulation/Decapsulation is used to forward packets from the home agent to the MN via the COA

18.3.3 Operation



1. MN registers with FA wehn it is out of its home net
2. FA forwards req to HA which acknowledges
3. HA then encapsulates all packets sent to MN and fowards via COA

19 11/14/17 Intro to Transport UDP

19.1 Issues in Mobile IP

1. Suboptimal routing
    - Solution: Let the CN know the COA of the MN - in this case the CN can create its own tunnel to the MN
2. Managing placement and operation of Home/Foreign agents in large networks
3. Frequent movements of clients can lead to significant traffic at Home Agent
4. **Security**

19.2 Transport Layer (4)

(5)	*Many Applications*
Transport	< User Datagram Protocol / Transport Control Protocol
Network	
Data Link	
Physical	

- End-to-end connectivity
- Basic service provided by the transport layer is to multiplex between multiple applications at Layer 5 and the network at Layer 3
- Multiplexing is facilitated via **ports**
- Because the IP service model is so limited, there are additional functions that are possible at the transport layer:
    1. Connection control
    2. Error detection
    3. Reliable delivery
    4. In-order delivery
    5. Flow control
    6. Congestion control

19.3 Layer 4 Multiplexing / Demultiplexing

- Mux = data is passed from applictation layer down to network layer
    - encapsulate and send to layer 3
- Demux = when packets arrive from layer, examine transport header and pass up to the appropriate application
    - Ports: 16-bit IDs for both source process and destination process. Ports 0-1023 = well known < used by servers
    - 1024+ = ephermeral ports < used by clients

19.4 Sockets

- **Sockets:** Connect applications to the network (sockets "abstract" the network) by providing a unique handle that associates ports and processes
- Socket API defines the creation, attachment, send/receive and close mechanisms that enable apps to access the network
- *Create*: required to generate a socket handle that identifies a network connection
  - int socket(domain (internet), type, protocol(eg. TCP/UDP))
- Next step depends on whether app is a server or a client
- Assume app is a *server*, so prepare to accept incoming connections
  - int bind(socket, address, addr_{len})
  - int listen(socket, backlog)
  - int accept(socket, addr_{len})
  - int receive(socket, buffer, buff_{len}, flags)

19.5 User Datagram Protocol (UDP)

- Simpler, connectionless transport protocol. "datagram" service that is unreliable/unordered. It provides mux/demux and error detection (optional for IPv4)

19.5.1 Header

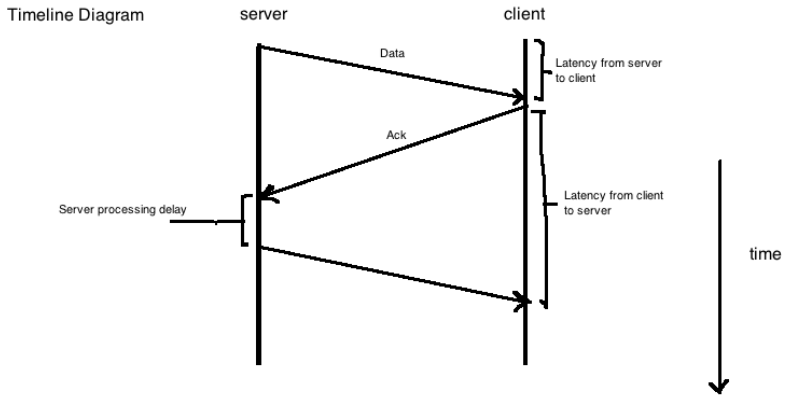
- 64 bits:
  - 16 bit source port
  - 16 bit destination port
  - 16 bit internet checksum < optional in IPv4, required in IPv6
  - 16 bit length field

20 11/16/17 Reliability, TCP

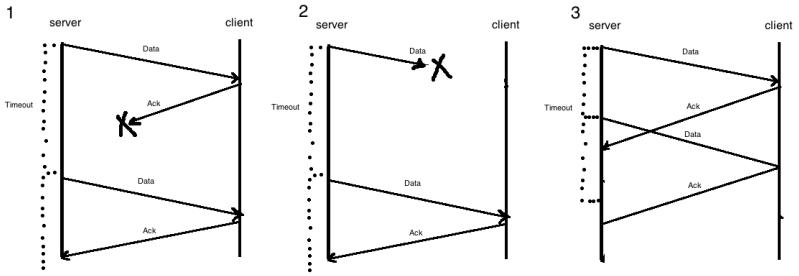
- UDP packets are "fully defined" by "Destination port/IP pair". ie) demux is based on this tuple

20.1 Reliable Transport

Goal: Offer a reliable service to applications. This is facilitated by offering an acknowledgement (ACK) in receipt of a packet.



Plus, the server uses a timeout mechanism to decide when to retransmit



(Timeout?) Buffers are required on both server and clients

20.1.1 Timeout/Timers

- Timeout signals play an important role in reliable transport. If they are too short, we may resend packets that were not lost
- If timeout is too long, performance goes down. Goal is to be sensitive to network conditions
- Basic mechanism = Exponentially weighted moving average (EWMA) of RTT from data sent to ACK received
  - Sample RTT is measured for every Data-Ack pair

Equations:

$$EstRTT = (\alpha * Es) + RTT + (\beta * SampleRTT)$$

$$\alpha + \beta = 1$$

$$0.8 < \alpha < 0.9$$

$$RTO = 2 * EstRTT (RetransmitTimeout = RTO)$$

20.1.2 Issues

- Sending one data packet at a time (ie. waiting for an ACK before sending the next data packet) may not be an efficient use of network bandwidth
- So, to take advantage of network resources, send multiple packets at a time
  - *Transport decides what to send and when to send it*
  - In UDP this is typically rate-based

- We use "sliding windows" on sender and receiver to improve network utilization and enable reliability
  - This implies the need to **flow control** to ensure that the receiver's buffer is not overflowed. This is a signal that the receiver sends to the server prior to data flow
- 2 other issues:
  1. The need to identify specific packets -> sequence numbers
  2. The need to manage buffers
- Sequence numbers have an upper limit before wrap around. Basic requirement:

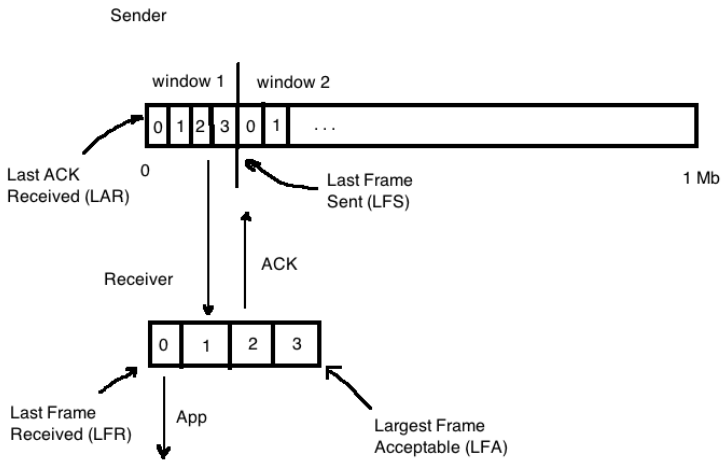
$$seqnumspace > numoutstandingpackets$$

- Simply stating that seq # space > # outstanding packets is insufficient
  - Assume 3-bit seq # space ie) seq #s range from 0-7.
    1. Sender sends pkts 0...6
    2. Receiver successfully receives these packets
    3. Receiver sends ACKs for all packets, which are lost
    4. Sender will resend 0...6
    5. Receiver expects seq# ...7, 0...6
  - So, there is a need for a larger seq# space
  - Max window(send window size) <= (Max Seq# + 1)/2

21 11/21/17 TCP

21.1 Reliability cont

How to make this efficient for network bandwidth? Send the max window size as often as possible



- Send window size = **SWS**
  - Upper bound on the # of unacknowledged packets (in-flight)

$$LFS - LAR <= SWS$$

- Sender maintains buffer in order to resend lost packets
- Receiver maintains buffer to ensure in-order, non-duplicate delivery to app
- Receive window size (**RWS**) = upper bound on out of order frames

$$LFA - LAR <= RWS$$

Implication: If packet arrives with seq # < LFR or > LFA, drop that packet

- Acks can be "grouped" in a **cumulative acknowledgement** which ACKs the sequence number of a group of packets that have successfully been received

21.2 Transport Control Protocol (TCP)

Features:

1. Connection oriented
2. Reliable transfer
3. Full duplex
4. Flow control
5. *Byte-oriented*
6. Congestion control

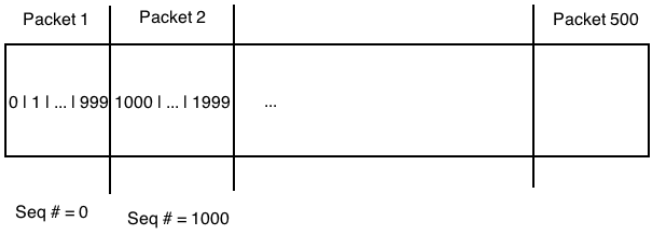
21.2.1 Header

Features:

- 16-bit src/dst port #s
- 32-bit seq/ack numbers
- 16-bit checksum
- 16-bit receive window size
- Flags: indicate packet type
  - SYN
  - FIN
  - ACK

21.2.2 Byte Oriented

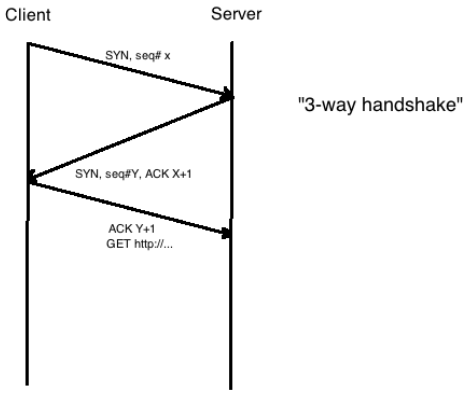
- TCP considers data as an "ordered byte stream" *(except for congestion control)
  - Implication: Sequence numbers reflect the first byte in a packet
  - Example: Assume a 500KB file with 1 KB maximum segment size and first byte sequence number = 0. TCP will construct 500 packets



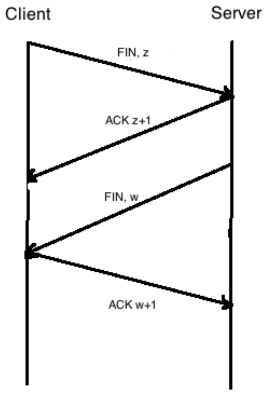
- ACKs are a bit tricker. ACKs reflect "the next byte expected". Thus, the ACK numbers for packets in the example would be:
  - ACK# for packet 1 = 1000
  - ACK# for packet 2 = 2000
- Initial sequence numbers are selected randomly by senders

21.2.3 Connection Management

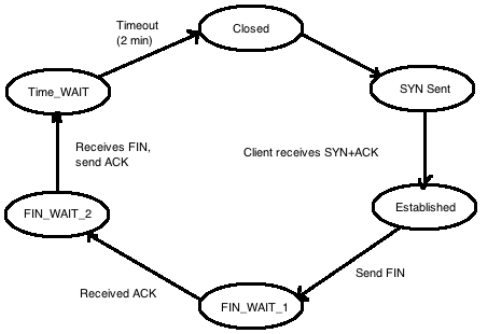
- See fig 5.7
- Connections are full duplex that are initiated by clients
- Connections are opened by preamble:



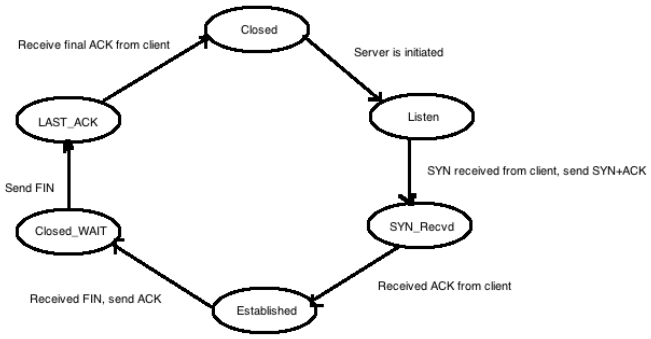
- Connections are concluded by an explicit tear-down sequence



Client States



Server States



22 11/28/17 TCP RTO Calculation, Congestion Control

22.1 Calculating RTO in TCP

- How can we improve over basic EWMA?

  1. **Karn/Partridge algorithm**
     - Don't sample the RTT on lost packets
     - Do exponential backoff on RTO for multiple timeouts
  2. **Jacobsen/Karls algorithm** - need to know algorithm

$$Diff = SampleRTT - EstRTT$$

$$EstRTT = EstRTT + (d * Diff), 0 < d < 1 (typically \ = 0.125)$$

$$Dev = Dev + d * (|Diff| - Dev)$$

$$RTO = \mu * EstRTT + \theta * Dev, \mu = 1, \theta = 4$$

     - Sensitive to Sample RTT variance

22.2 Congestion Control

- ~'87-'89
- The problem was that at the time, TCP was configured to send a full window of packets (ie flow control only) whenever possible
- Congestion control is based on the idea of **packet conservation**
  - For stability, a host in equilibrium should only inject a new packet when another packet has been received *(self pacing)*
- TCP Tahoe = original version of TCP with congestion control

22.2.1 Core Mechanism - AIMD

- Additive increase, multiplicative decrease = **AIMD**
- AIMD objective is to be sensitive to the available capactiy of an end-2-end path
- TCP Tahoe and subsequent variants use a congestion window (CWND) in addition to flow control (RWND)
- CWND limits the number of packets in flight to something less than RWND

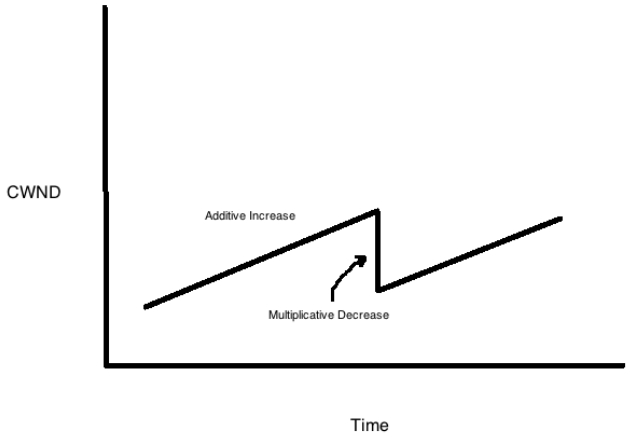
$$MaxWindow = min(CWND, RWND)$$

- Increase/decrase CWND depending on capacity of path
- Adjustments to CWND are based on signals from packets (data/ack) and in paritcular *loss* (as indicated by RTO)
- We will grow the size of CWND by probing for additional capacity

Algorithm:

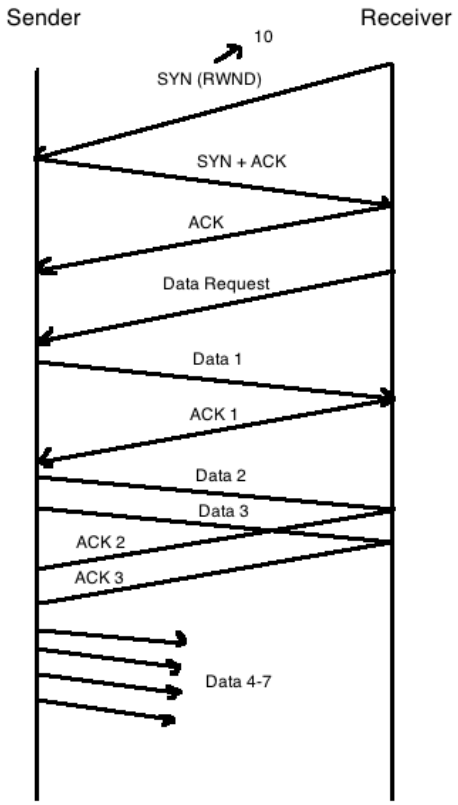
1. Increase CWND by 1 for each RTT
2. Decrease CWND by \$CWND$_{\overline{2\&ontimeoutCWND>=1}}$

23 11/30/17 Congestion Control, Congestion Avoidance



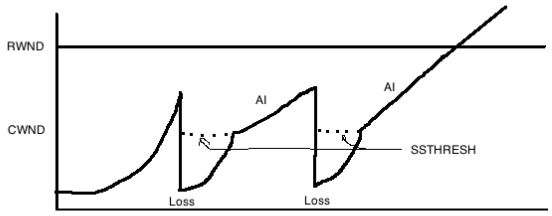
23.1 Slow Start in TCP

- Instead of starting a transfer with a full RWND (ie flow control limit), start sending at a slower rate, but something faster than simple additive increase



23.1.1 Algorithm

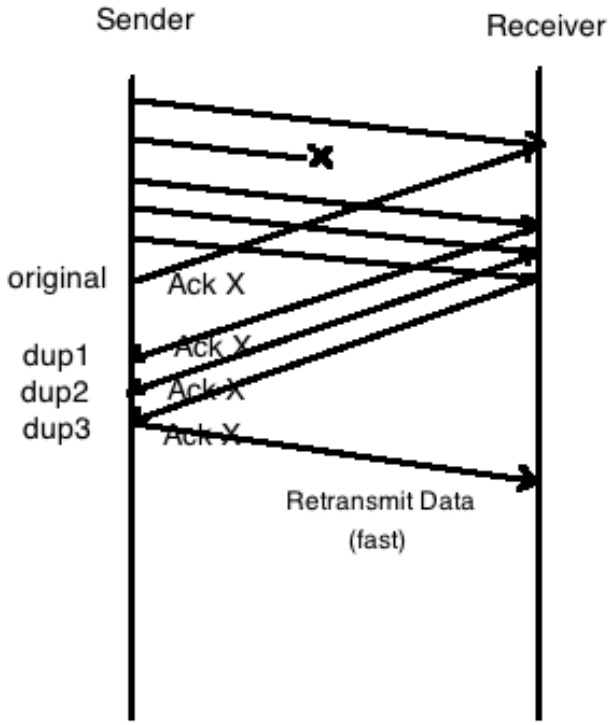
1. Start with CWND = 1
    - Set slow start threshold (SSTHRESH = \(inf))lp
2. Increase CWND by 1 for each packet acknowledged
3. When CWND >= SSTHRESH when transition to AIMD
4. When at a packet is lost, set SSTHRESH = CWND/2 (for each loss) and for any loss, set CWND = 1



- CWND >= 1
- SSTHRESH >= 2

23.2 TCP Reno

- Added 2 mechanisms to TCP Tahoe:
  1. Fast Retransmit
  2. Fast Recovery
- Acks returned from receiver can signal that a packet has been lost
- New signal for last packet:
  - triple duplicate ACK



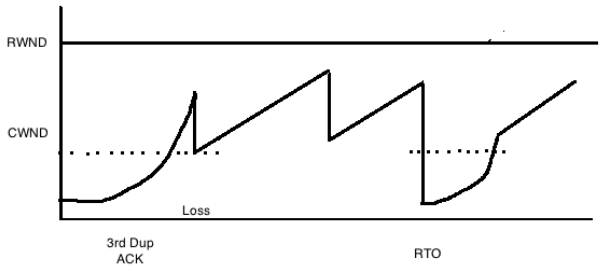
- **Fast Retransmit** algorithm:

    1. Upone receipt of 3rd duplicate ACK, retransmit packet that includes next byte expected

    2. After 3rd dup ACK, send one new packet for each ACK received

- **Fast Recovery:**

    1. Upon receipt of ACK for lost packet, set SSTHRESH = CWND/2, CWND = SSTHRESH, Start AI



23.3 Congestion Avoidance

- Instead of designing congestion control that leads to loss, can we avoid loss all together?
  - Host based: *TCP Vegas*
  - Network based: *Random Early Detection (RED)*

23.3.1 TCP Vegas

- Adjust send rate when there are signals that queues are growing on an end-to-end path. The signal is that *RTTs grow* (ie send rates go down)
- Vegas tries to identify when queues are growing and adjust send rate to stay just below capacity

Algorithm:

- $Diff = ExpectedRate - ActualRate$ where $Expected = CWND/BaseRTT$
- $ActualRate = CWND/SampleRTT$
- If $Diff < \alpha$ increase CWND linearly. α is typically $= 1$
- If $Diff > \beta$ decrease CWND linearly. β is typically $= 3$
- Else, leave CWND unchanged

Default behavior for TCP Vegas = TCP Reno when packets are being lost

24 12/5/17 Application Layer

24.1 Application Architectures

- Applications run on end hosts - 2 or more are needed
- Architecture specifies how to organize application on end hosts

Architectures:

1. Client - Server
    - Server: always on and receives and processes requests
    - Clients: sometimes/always on
    - ie) web browsing, email, etc
2. Peer-to-Peer (P2P)
    - ie) BitTorrent
    - Arbitrary hosts (peers) communicate directly
3. Hybrid
    - Peers communicate directly, but there is also an always on match-making server
    - ie) Skype, instant messaging

24.2 Application Layer Protocols

- Protocols specify:
    1. Types: request/response
    2. Syntax: field formats
    3. Semantics: meaning of fields
    4. Rules: for when and how a process sends/receives packets

24.3 The World Wide Web (WWW)

- Tim Bernes-Lee invented in 1989
- Organize information into a system of linked documents or objects
- Components:
  - Structural: Browsers, servers, caches
  - Semantics: HTTP, HTML/XML, URL

Browsers:

- Run on clients
- Generate well-formed HTTP requests
- Interpret HTML data
- ie) Chrome, Safari, Internet Explorer

Servers:

- Wait for requests on port 80 and TCP
- They house web objects and respond to client requests

Caches:

- Copies of frequently requested documents

24.4 HTTP (HyperText Transfer Protocol)

- Protocol for client/server communication
- Request: Uniform Resource Location (URL)
  - `http://foo.org/index.html`
- 8 different command/reqeust types
  - GET: get the document identified by the URL
  - POST: give information to the server

  - HEAD, PUT, DELETE
- Responses:
  - Status code: 200 OK, 404 not found
- 4 versions:
  - 0.9
  - 1.0
  - 1.1
  - 2.0

24.4.1 HTTP 1.0

- Stop and wait
- Separate TCP connections for each HTTP request
- Inefficient:
  1. Latency impact
  2. Connection setup and tear down

24.4.2 HTTP 1.1

- Persistent connection: same TCP with multiple HTTP requests
- Pipelining: send multiple HTTP requests without waiting for response
- HOL (head-of-line) blocking - impacts performance

24.4.3 HTTP 2.0

- Decrease in latency:
  - compression of headers
  - fixed HOL blocking problem
  - server PUSH

24.5 Domain Name System (DNS)

- It is difficult for humans to remember IP addresses
- DNS resolves names to IP addresses
- Originally: static list of name-to-IP mapping
- Hierarchical name space system for internet objects
  - DNS
- Names are read from left to right, separated by periods. Each suffix in a domain name is a domain

ie) cs.wisc.edu wisc.edu .edu

- Port 53 and UDP
- Rightmost port of domain called Top Level Domain (TLD)
  - Original TLDs: edu, com, gov, mil, org, net
  - Countries: .fr, .uk
  - Arbitrary TLDs
- Top level TLDs managed by ICANN

25 12/7/17 Domain Name System (DNS)

- Each level of the heirarchy is partitioned into zones
- Each zone is implemented by 2 or more servers
- Servers maintain a collection of resource records
- Types:
  - A = IPv4 address
  - AAAA = IPv6 address
  - NS = name server record
  - MX = mail server
  - CNAME = cannonical name

25.1 Content Delivery Network (CDN)

- Geographically distributed content over a network of proxy servers and data centers
- Mechanisms for selecting "best server":
    1. Better performance
    2. Reduced latency
    3. Load balancing
    4. Additional capacity
- How to pick a CDN server?
    1. Physical proximity
    2. Server load
    3. Congestion
    4. Cheapest path

25.2 Network Security

25.2.1 Security Services:

1. Privacy: Prevent unauthorized access to data
2. Authentication: Verifying ID of remote user
3. Integrity: Make sure messages have not been altered = assurance that information is trustworthy
4. Confidentiality: Encrypt messages to prevent adversary from understanding messages
    - Traffic confidentiality: concealing the quantity of traffic on destination

25.2.2 Symmetric-Key Encryption

- Sender and receiver share the same "secret key"
- Data Encryption Standard (DES)
  - 56-bit keys
  - 2^{56} search space for keys
  - because of parallelism DES became unusable
- 3-DES
  - 168-bit keys
- Advanced Encryption Standard (ADS)
  - 128, 192, or 256 bits

25.2.3 Public Key Ciphers

- Pair of keys owned by one participant
- Decryption using "private key"
- Encryption using "public key" -> shared
- Ex) Riverst Shamir Adlemann (RSA)
  - Keys are products of 2 large prime numbers
  - 1024 bits
  - High computational cost of factoring large prime numbers

25.2.4 Key Exchange

- Diffie-Hellman Key Exchange
- RSA tokens

25.2.5 Data Integrity

- Encryption does not guarantee data integrity since random bit flips can result in a plain text message that looks valid
- To address integrity and to prevent tampering of messages, we use "redundancy", **cryptographic checksum**
- To do this, we use "cryptographic has function"
- Output = Message Digest (MD)
  - MDs have special property that they produce some number of bits regardless of length of message

25.2.6 Authentication

- Note as simple as appending signatures to every message
- Replay and delayed-replay attacks
  - ex) credit card online purchase (replay), stock market, auction (delayed-replay)
- Use "session-keys" = symmetric-key ciphers

25.3 Challenge Response Protocol

- Simple one-way authentication protocol

25.4 Public Key Authentication

(assuming clock synchronization)

25.4.1 Public Key Auth (without clock sync)

25.4.2 Kerberos

- Trusted 3rd party with whom hosts share keys
- via 3rd party auth sequence is initiated

25.5 Issues in Security

- Threat models = how exactly bad guys attack?
- Key distribution = Public Key Infrastructure (PKI)
- Verification = how can we be sure systems are secure?