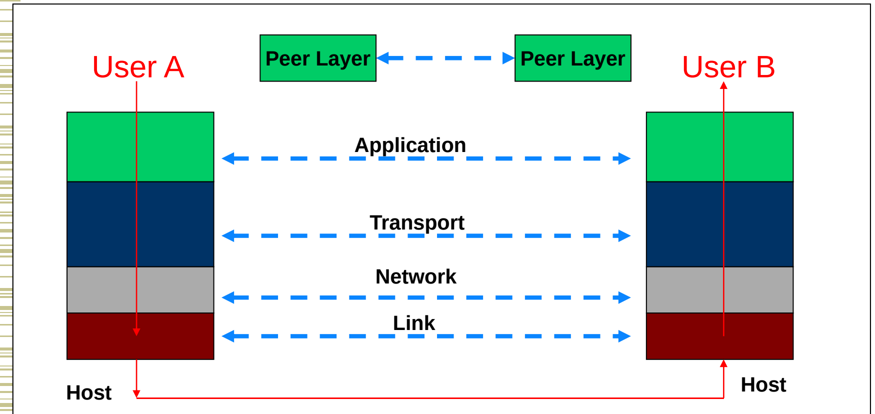
# 15-441 Computer Networking

Lecture 29 – Final Review

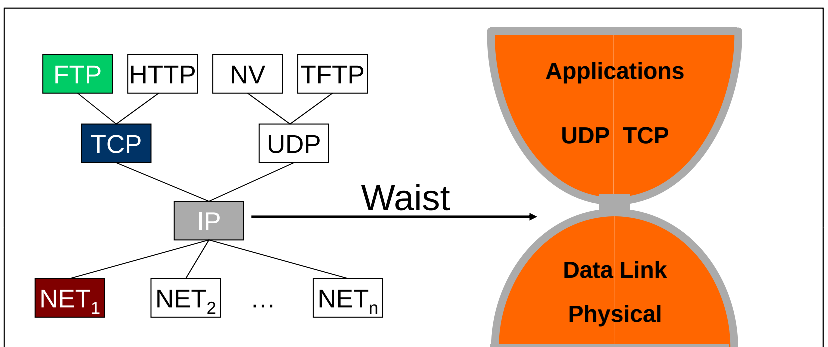
## What is Layering?




Modular approach to network functionality



## The Internet Protocol Suite



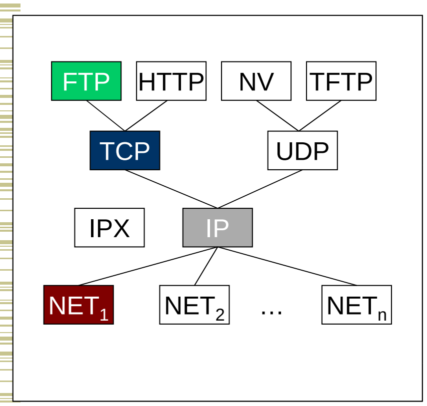

**The Hourglass Model**

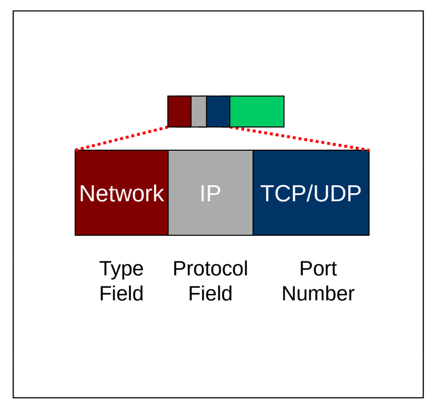
The waist facilitates interoperability



## Protocol Demultiplexing

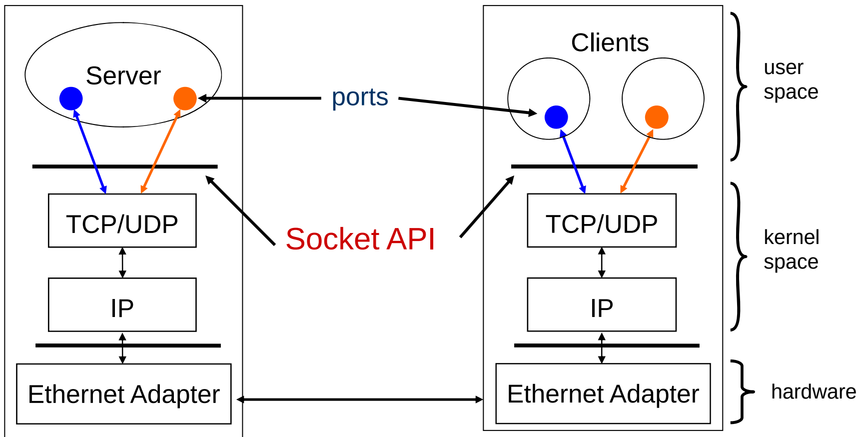
- Multiple choices at each layer



# Server and Client

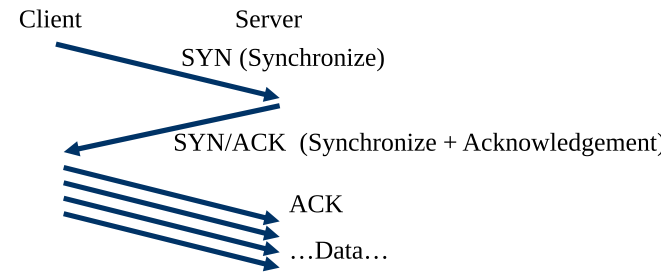
Server and Client exchange messages over the network through a common Socket API






# One more detail: TCP

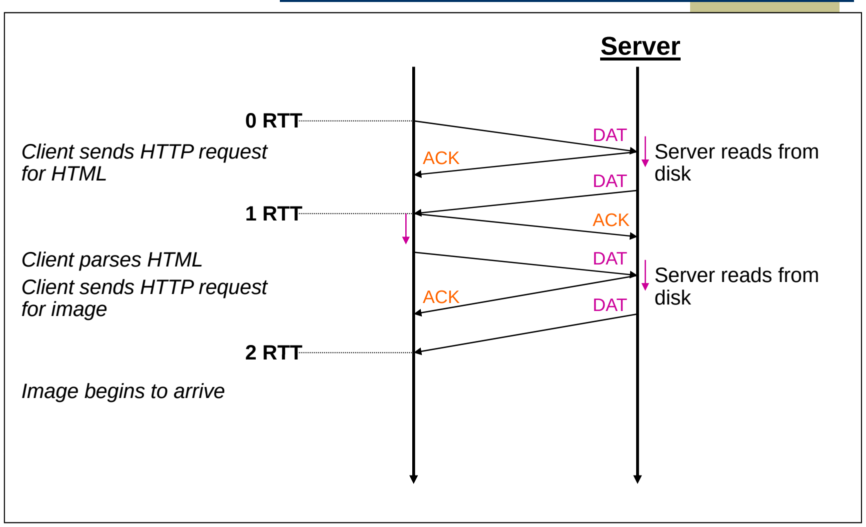
- TCP connections need to be set up
  - "Three Way Handshake":




2: TCP transfers start slowly and then ramp up the bandwidth used (so they don't use too much)

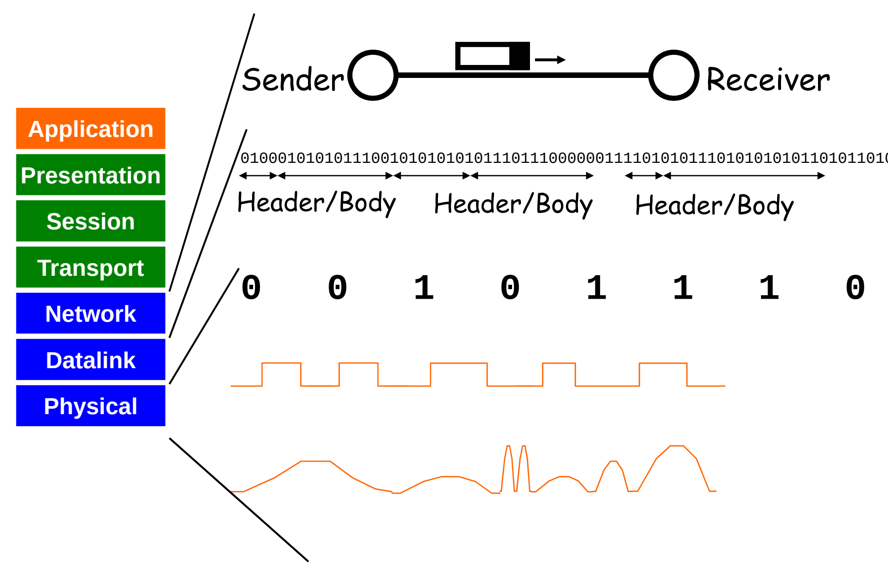


# Persistent Connection Solution



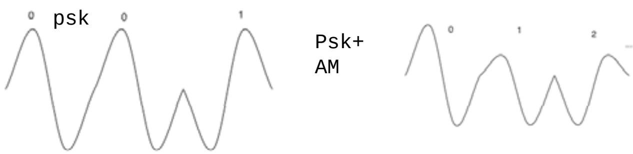



# From Signals to Packets



# Past the Nyquist Limit

- More aggressive encoding can increase the channel bandwidth.
  - Example: modems
    - Same *frequency* - number of symbols per second
    - Symbols have more possible values




- Every transmission medium supports transmission in a certain frequency range.
  - The channel bandwidth is determined by the transmission medium and the quality of the transmitter and receivers
  - Channel capacity increases over time



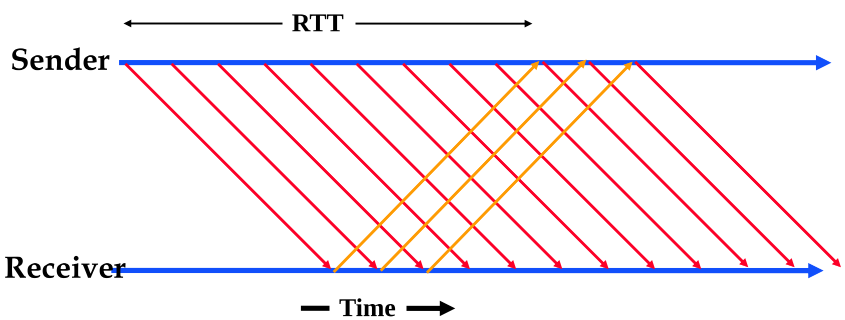
# Why Encode?

0 1 0 1 How many more ones?

NRZ
NRZI
Manchester



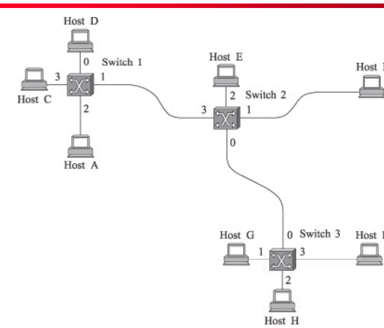
# Bandwidth-Delay Product

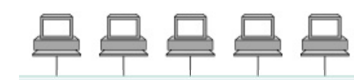



$$\text{Max Throughput} = \frac{\text{Window Size}}{\text{Roundtrip Time}}$$



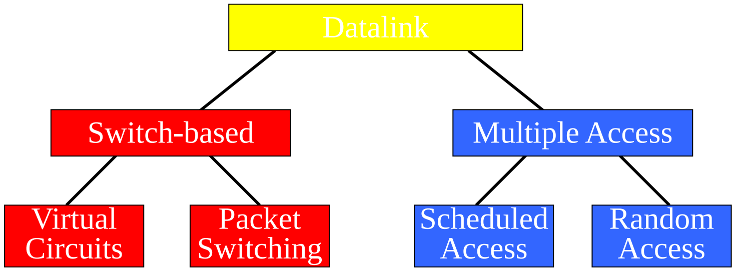
# Datalink Architectures



- Point-Point with switches
- Media access control.

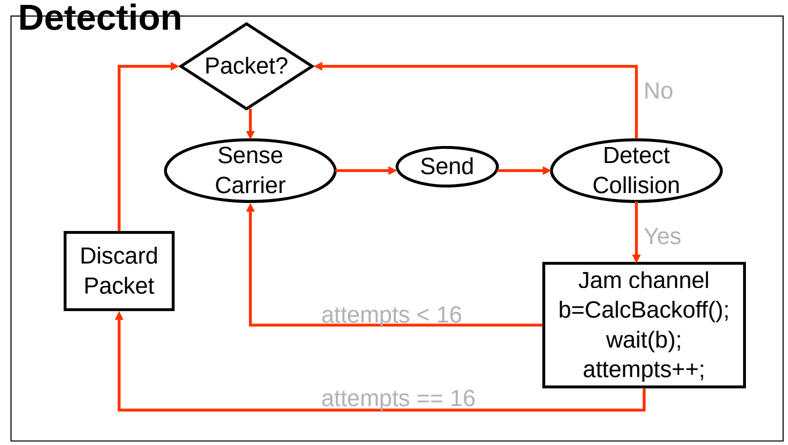
## Datalink Classification






## Ethernet MAC (CSMA/CD)

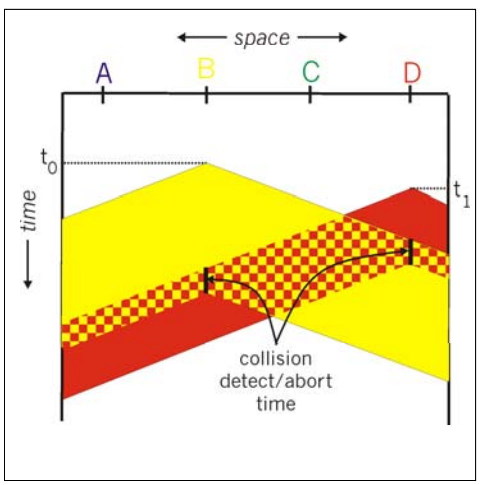
- **Carrier Sense Multiple Access/Collision Detection**






## Minimum Packet Size

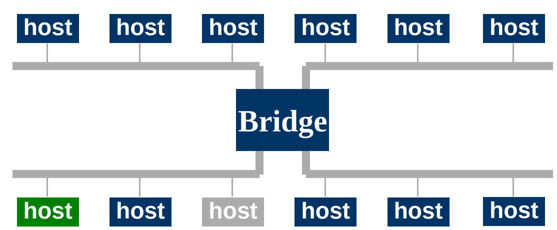
- **What if two people sent really small packets**
  - » **How do you find collision?**





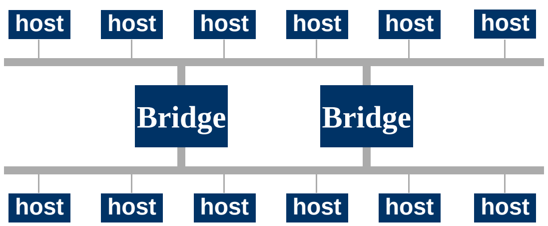

## Learning Bridges

- Manually filling in bridge tables?
  - Time consuming, error-prone
- Keep track of source address of packets arriving on every link, showing what segment hosts are on
  - Fill in the forwarding table based on this information



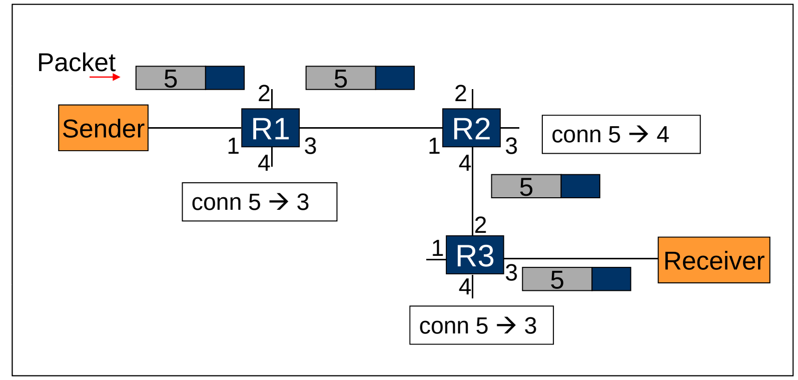
# Spanning Tree Bridges

- More complex topologies can provide redundancy.
  - But can also create loops.
- What is the problem with loops?
- Solution: spanning tree



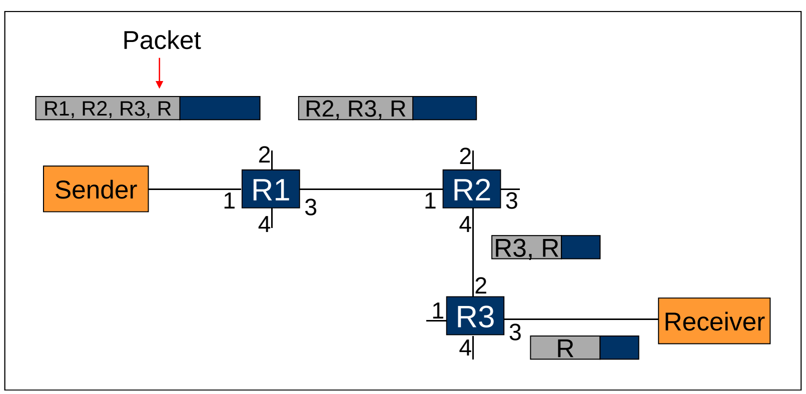



# Simplified Virtual Circuits Example



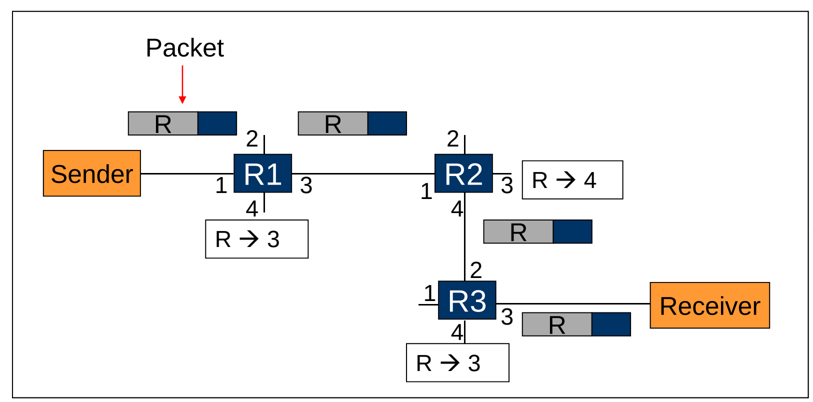



# Source Routing Example

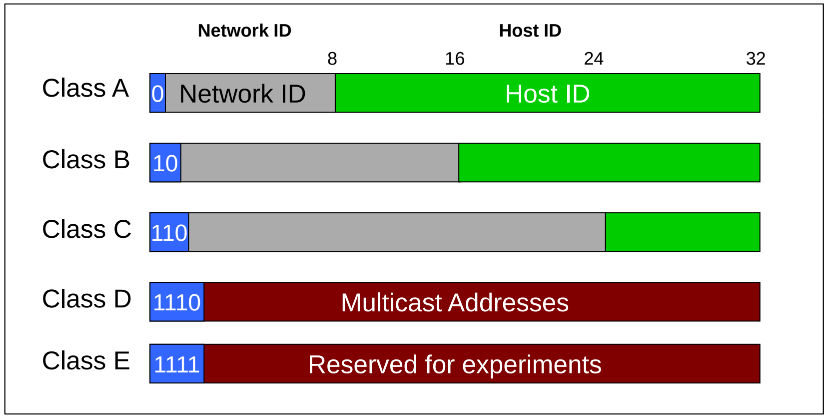





# Global Address Example



# IP Address Classes
(Some are Obsolete)





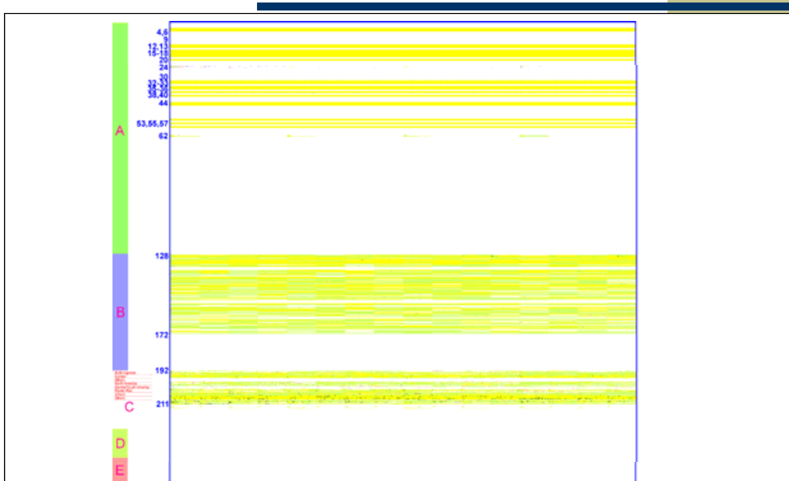

# ARP Cache Example

- Show using command “arp -a”

```
Interface: 128.2.222.198 on Interface 0x1000003
  Internet Address      Physical Address      Type
  128.2.20.218          00-b0-8e-83-df-50     dynamic
  128.2.102.129         00-b0-8e-83-df-50     dynamic
  128.2.194.66          00-02-b3-8a-35-bf     dynamic
  128.2.198.34          00-06-5b-f3-5f-42     dynamic
  128.2.203.3           00-90-27-3c-41-11     dynamic
  128.2.203.61          08-00-20-a6-ba-2b     dynamic
  128.2.205.192         00-60-08-1e-9b-fd     dynamic
  128.2.206.125         00-d0-b7-c5-b3-f3     dynamic
  128.2.206.139         00-a0-c9-98-2c-46     dynamic
  128.2.222.180         08-00-20-a6-ba-c3     dynamic
  128.2.242.182         08-00-20-a7-19-73     dynamic
  128.2.254.36          00-b0-8e-83-df-50     dynamic
```



# IP Address Utilization (‘97)

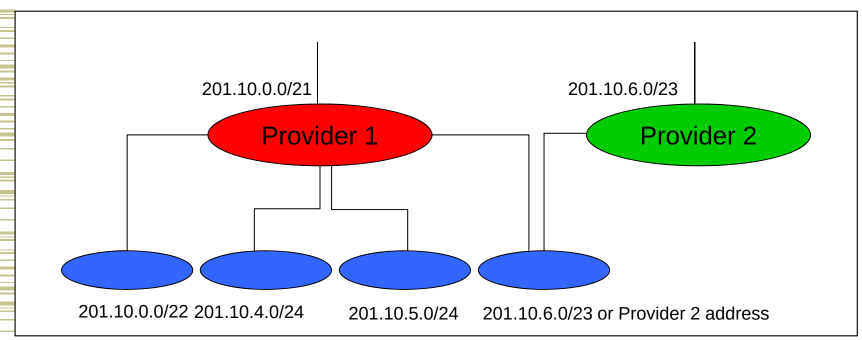



http://www.caida.org/outreach/resources/learn/ipv4space/ -- broken



# CIDR Implications

- Longest prefix match!!



# IP Service Model

- Low-level communication model provided by Internet
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram

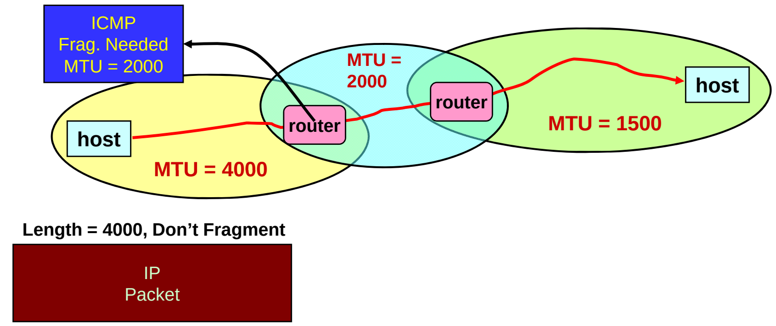
**IPv4 Packet Format**

| 0 | 4 | 8 | 12 | 16 | 19 | 24 | 28 | 31 |
|---|---|---|---|---|---|---|---|---|
| version | HLen | TOS | | Length | | | | |
| Identifier | | | | Flag | Offset | | | |
| TTL | | Protocol | | Checksum | | | | |
| Source Address | | | | | | | | |
| Destination Address | | | | | | | | |
| Options (if any) | | | | | | | | |
| Data | | | | | | | | |

**Header** (version through Options)



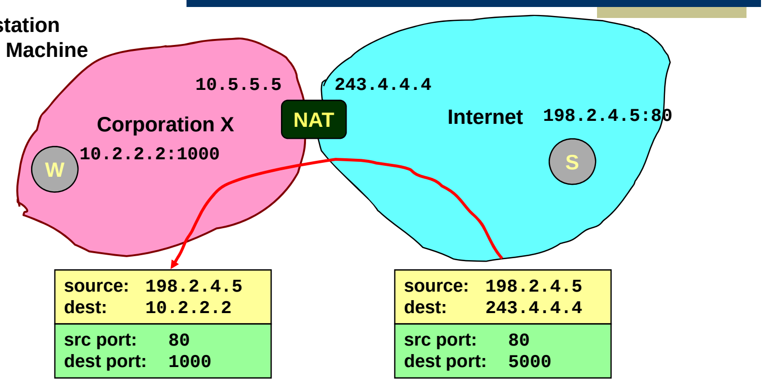
# IP MTU Discovery with ICMP






# NAT: Server Response

**W: Workstation**
**S: Server Machine**




| Int Addr | Int Port | NAT Port |
|---|---|---|
| 10.2.2.2 | 1000 | 5000 |

- Firewall acts as proxy for client
  - Acts as destination for server messages
  - Relabels destination to local addresses

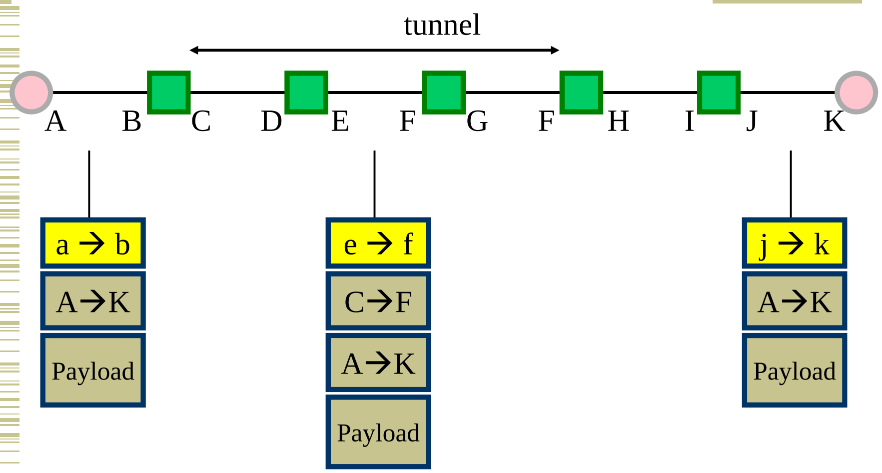


# IPv6

- "Next generation" IP.
- Most urgent issue: increasing address space.
  - 128 bit addresses
- Simplified header for faster processing:
  - No checksum (why not?)
  - No fragmentation (?)
- Support for guaranteed services: priority and flow id
- Options handled as "next header"
  - reduces overhead of handling options

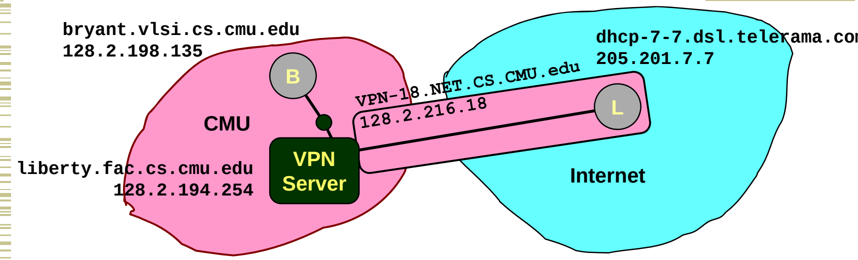
| V/Pr | Flow label | |
|---|---|---|
| Length | Next | Hop L |
| Source IP address | | |
| Destination IP address | | |

# Tunneling Example






# CMU CS VPN Example




- CS has server to provide VPN services
- Operation
  - Running echo server on CMU machine 128.2.198.135
  - Run echo client on laptop connected through DSL from non-CMU ISP
- With VPN
  - server connected to
  - VPN-18.NET.CS.CMU.EDU
  - (128.2.216.18)
- Effect
  - For other hosts in CMU, packets appear to originate from within CMU



# Comparison of LS and DV Algorithms

**Message complexity**

- LS: with n nodes, E links, O(nE) messages
- DV: exchange between neighbors only

**Speed of Convergence**

- LS: Relatively fast
  - Complex computation, but can forward before computation
  - may have transient loops
- DV: convergence time varies
  - may have routing loops
  - count-to-infinity problem
  - faster with triggered updates

**Space requirements:**

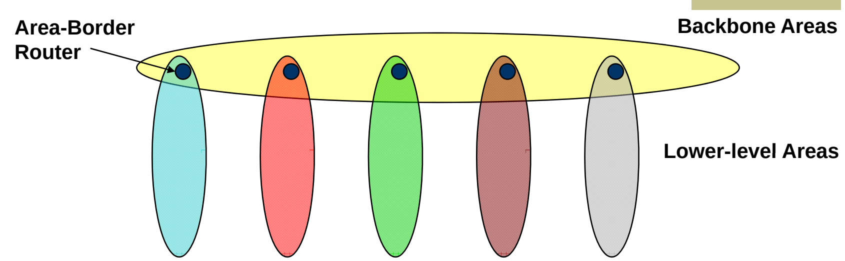
- LS maintains entire topology
- DV maintains only neighbor state

**Robustness: router malfunctions**

- LS: Node can advertise incorrect link cost
  - Each node computes its own table
- DV: Node can advertise incorrect path cost
  - Each node's table used by others (error propagates)

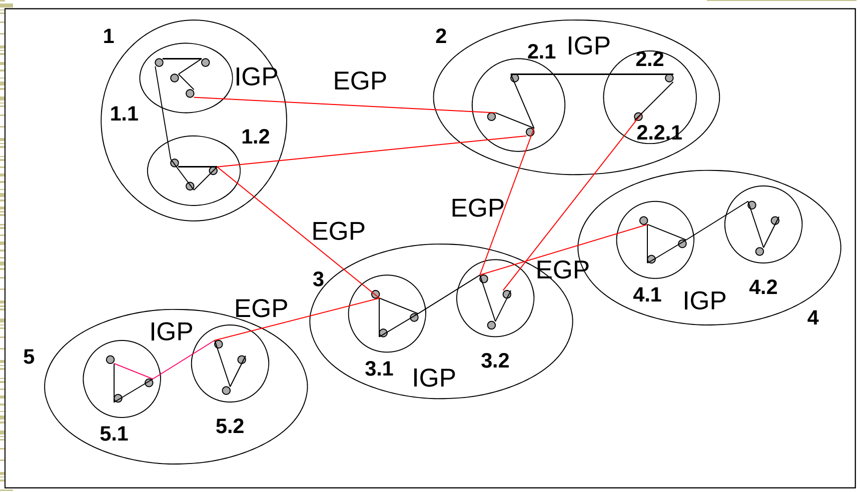


# Routing Hierarchy



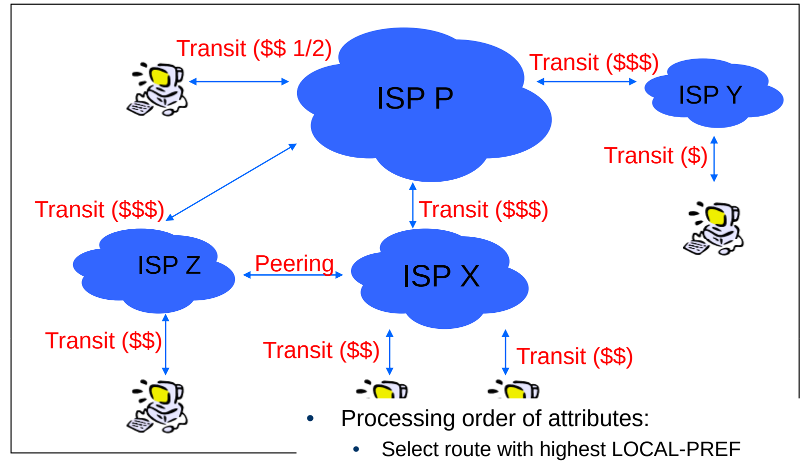

- Partition Network into "Areas"
  - Within area
    - Each node has routes to every other node
  - Outside area
    - Each node has routes for other top-level areas only
    - Inter-area packets are routed to nearest appropriate border router
- Constraint: no path between two sub-areas of an area can exit that area

# Example






# Transit vs. Peering

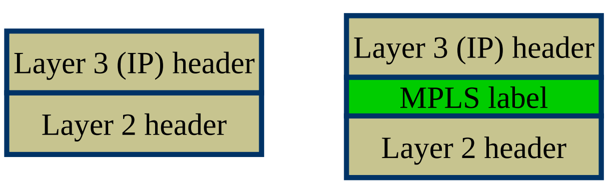



Path vector

- Processing order of attributes:
  - Select route with highest LOCAL-PREF
  - Select route with shortest AS-PATH
  - Apply MED (if routes learned from same neighbor)

# Multi Protocol Label Switching - MPLS

- Selective combination of VCs + IP
  - Today: MPLS useful for traffic engineering, reducing core complexity, and VPNs
- Core idea: Layer 2 carries VC label
  - Could be ATM (which has its own tag)
  - Could be a "shim" on top of Ethernet/etc.:
  - Existing routers could act as MPLS switches just by examining that shim -- no radical re-design. Gets flexibility benefits, though not cell switching advantages




# DNS Records

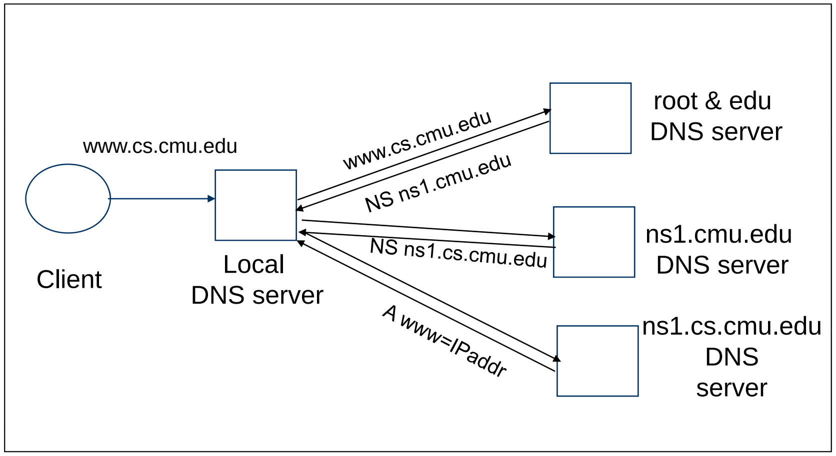
RR format: **(class, name, value, type, ttl)**

- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

**FOR IN class:**

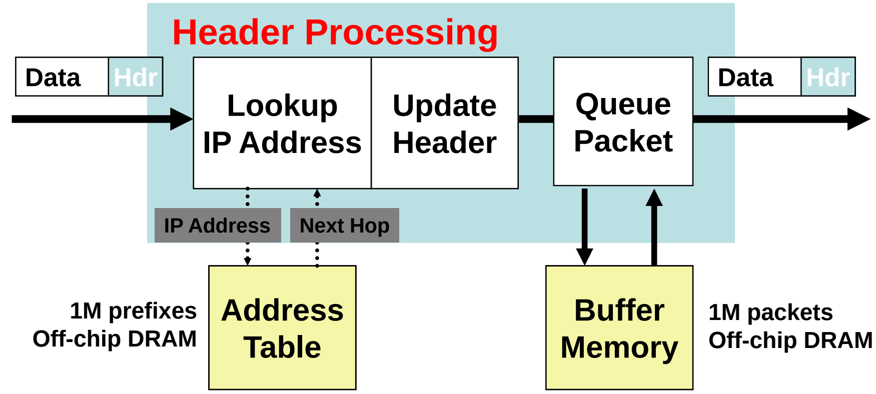
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is name of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "canonical" (the real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

# Typical Resolution



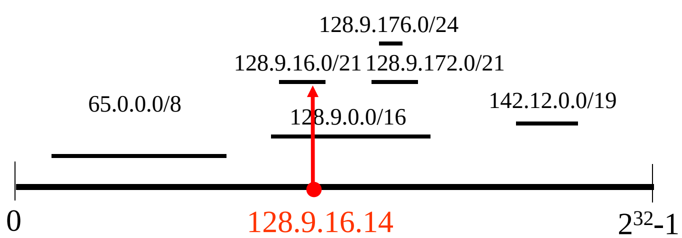



# Generic Router Architecture





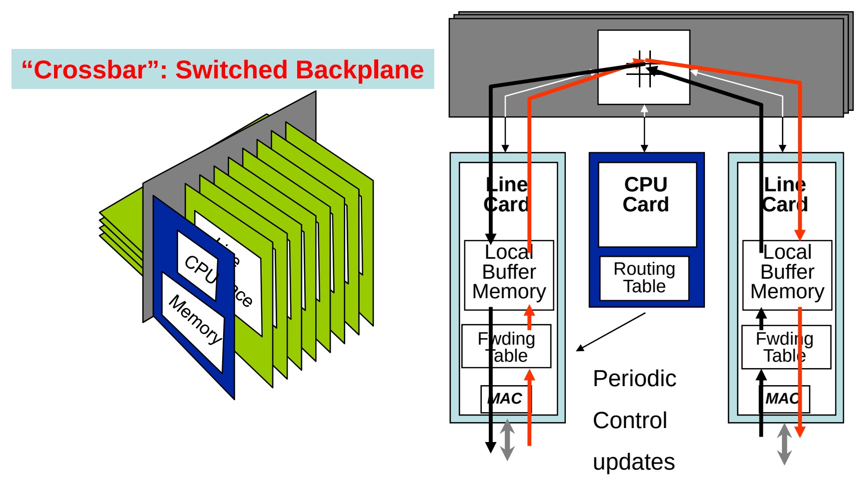

# IP Lookups find Longest Prefixes




Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.



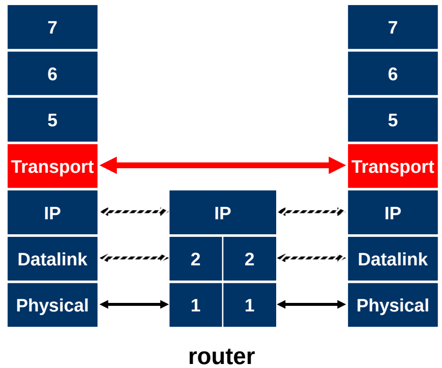
# Third Generation Routers




Typically <50Gb/s aggregate capacity

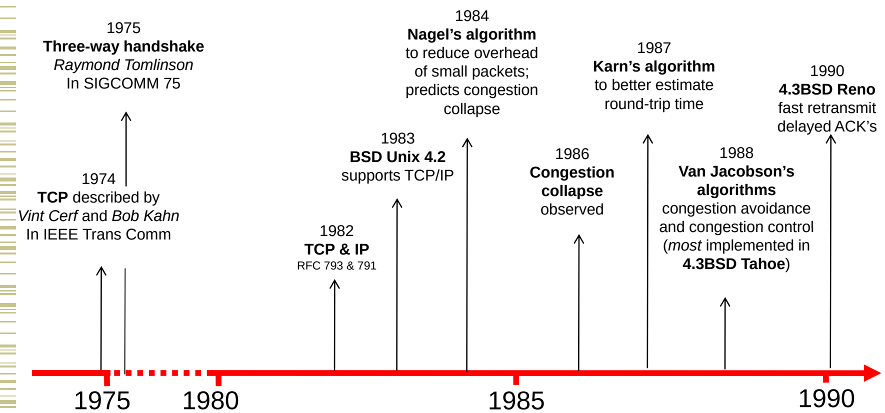
# Transport Protocols

- Lowest level end-to-end protocol.
  - Header generated by sender is interpreted only by the destination
  - Routers view transport header as part of the payload



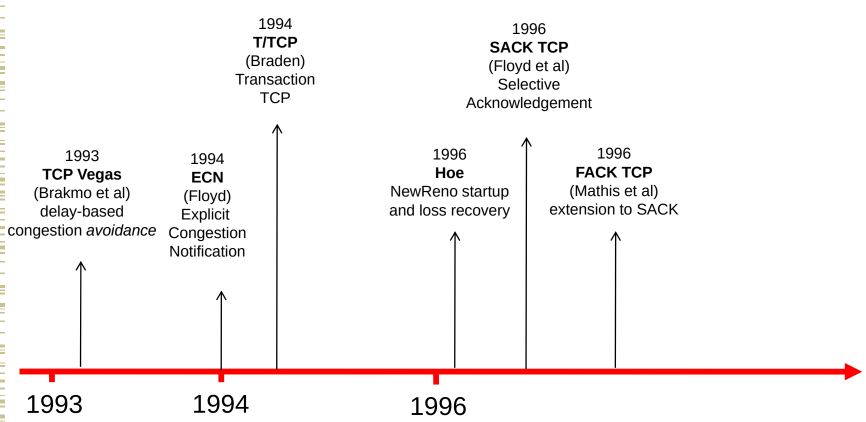



# Evolution of TCP



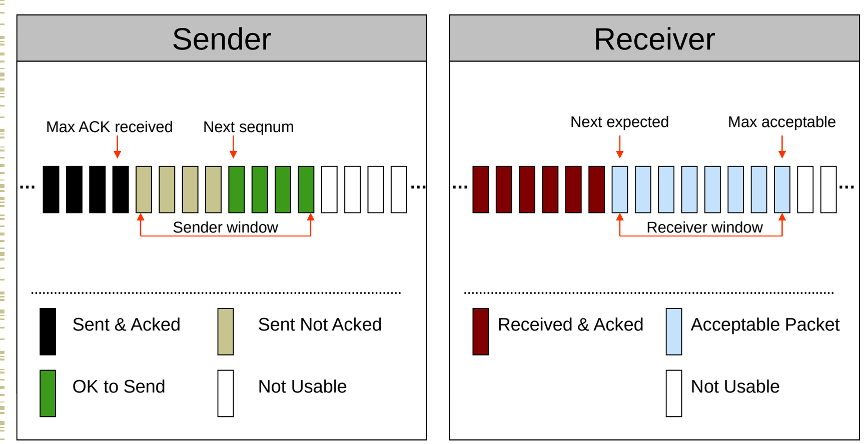



# TCP Through the 1990s

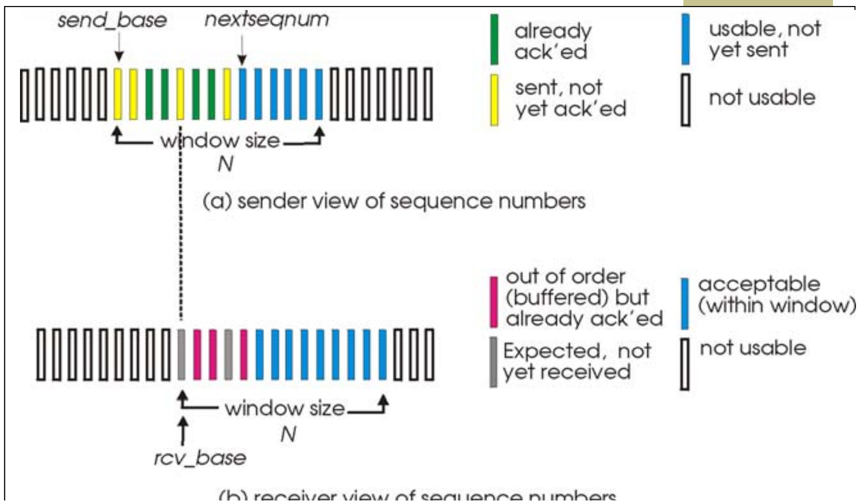





# Sender/Receiver State



# Selective Repeat: Sender, Receiver Windows

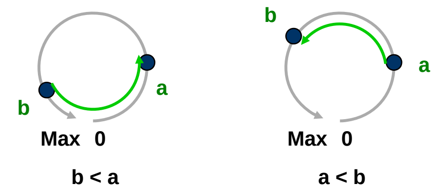



Compare: stop-and-wait, go-back-n, selective repeat



# Sequence Numbers

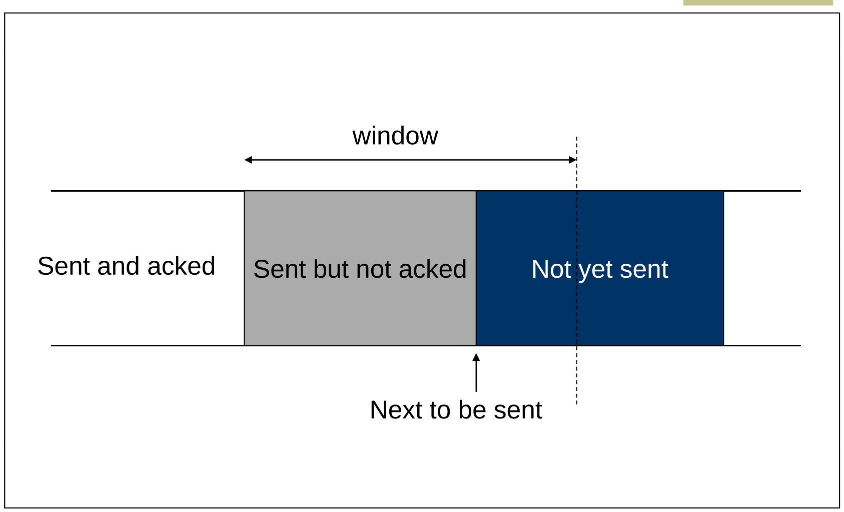
- 32 Bits, Unsigned → for bytes not packets!
  - Circular Comparison




- Why So Big?
  - For sliding window, must have
    |Sequence Space| > |Sending Window| + |Receiving Window|
    - No problem
  - Also, want to guard against stray packets
    - With IP, packets have maximum lifetime of 120s
    - Sequence number would wrap around in this time at 286MB/s

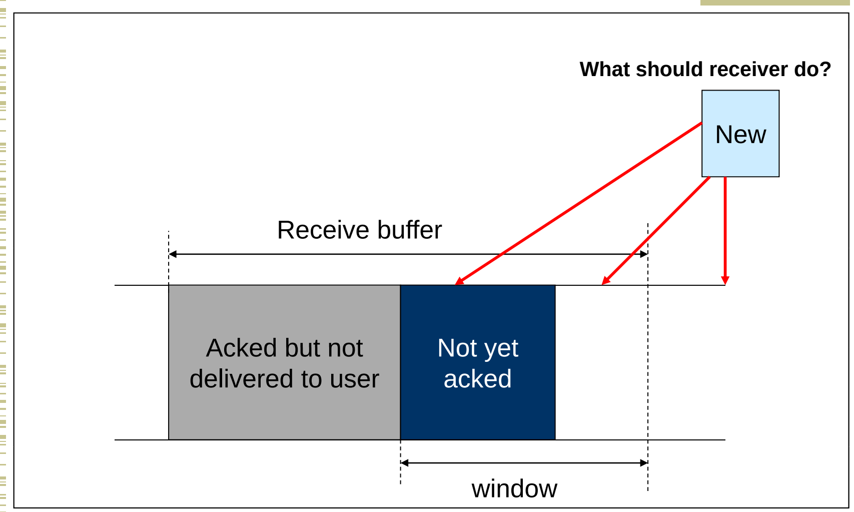


# Window Flow Control: Send Side



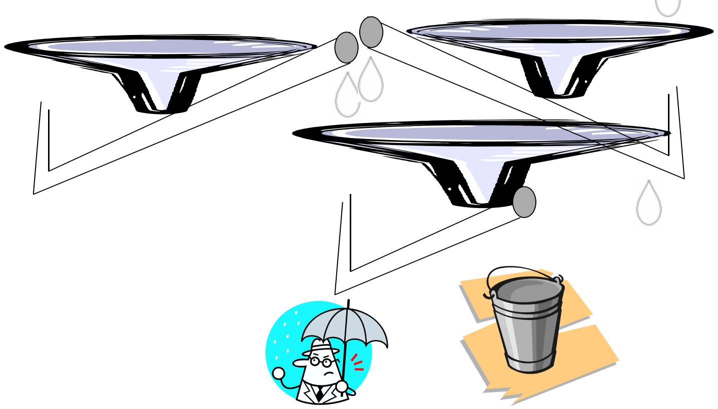



# Window Flow Control: Receive Side



# Plumbers Gone Wild 2!

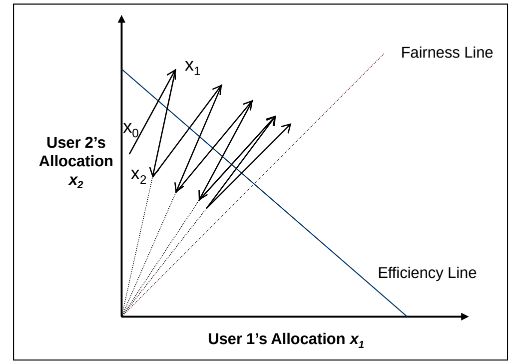
- Now what?
- Feedback from the bucket or the funnels?





# What is the Right Choice?

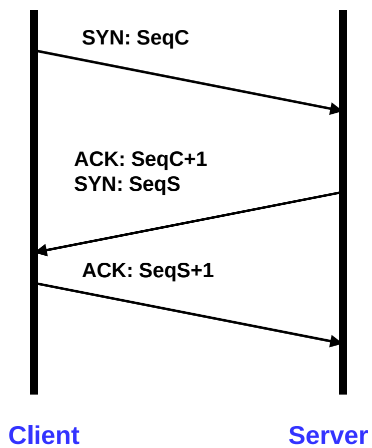
- Constraints limit us to AIMD
  - Can have multiplicative term in increase (MAIMD)
  - AIMD moves towards optimal point





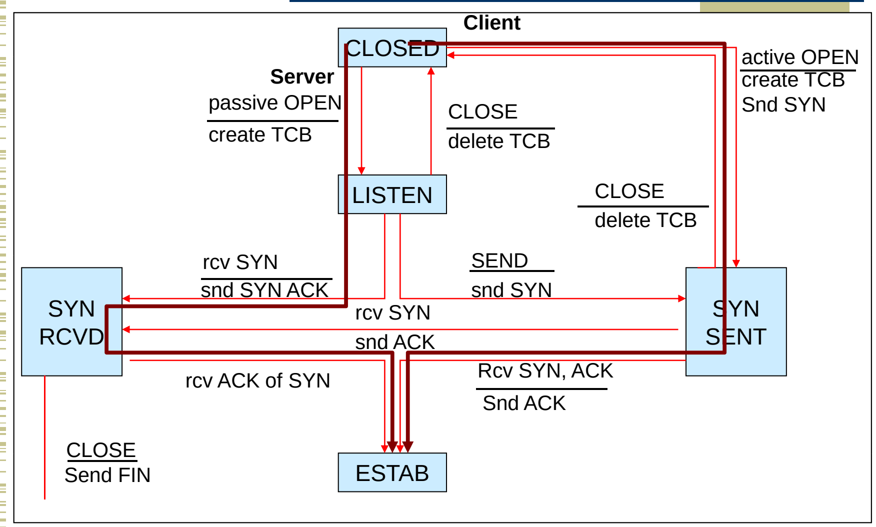

# Establishing Connection: Three-Way handshake

- Each side notifies other of starting sequence number it will use for sending
  - Why not simply chose 0?
    - Must avoid overlap with earlier incarnation
    - Security issues
- Each side acknowledges other's sequence number
  - SYN-ACK: Acknowledge sequence number + 1
- Can combine second SYN with first ACK

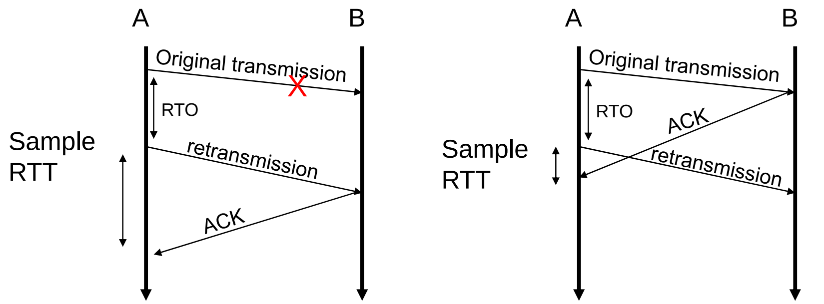





# TCP State Diagram: Connection Setup



# RTT Sample Ambiguity




- Karn's RTT Estimator
  - If a segment has been retransmitted:
    - Don't count RTT sample on ACKs for this segment
    - Keep backed off time-out for next packet
    - Reuse RTT estimate only after one successful transmission

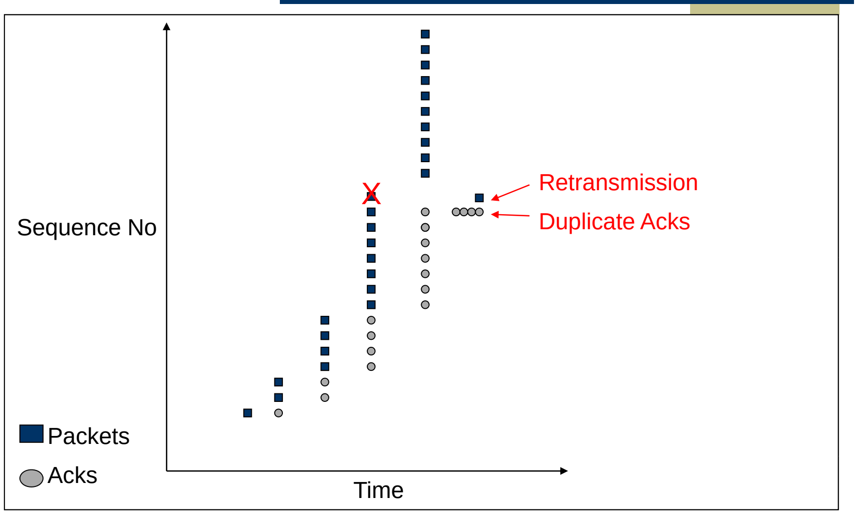


# Jacobson's Retransmission Timeout

- Key observation:
  - At high loads, round trip variance is high
- Solution:
  - Base RTO on RTT and standard deviation
    - RTO = RTT + 4 * rttvar
  - new_rttvar = $\beta$ * dev + (1- $\beta$) old_rttvar
    - Dev = linear deviation
    - Inappropriately named – actually smoothed linear deviation

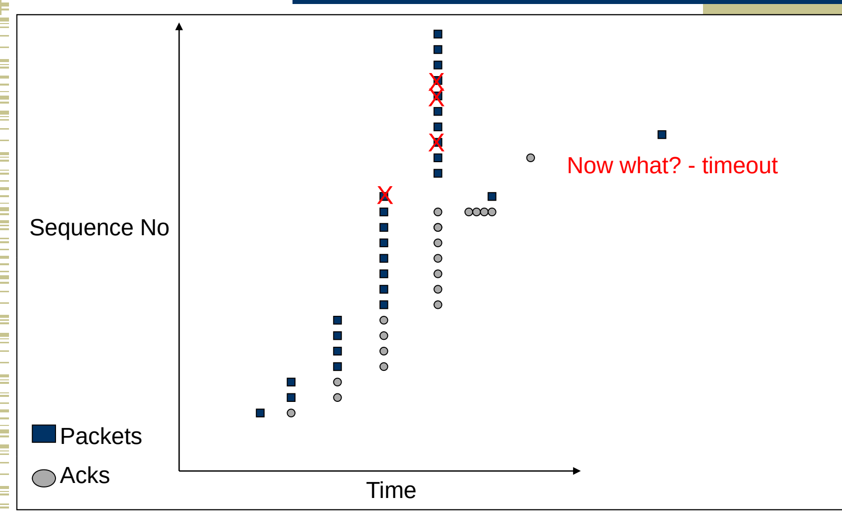


# Fast Retransmit

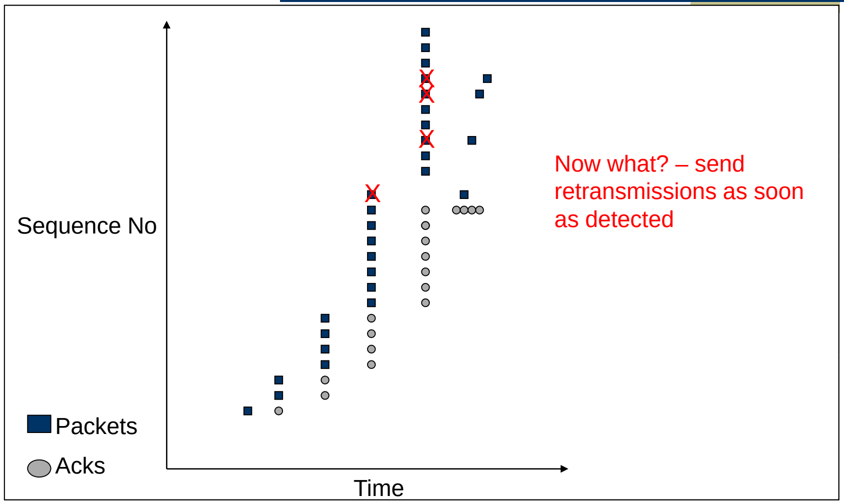





# TCP (Reno variant)



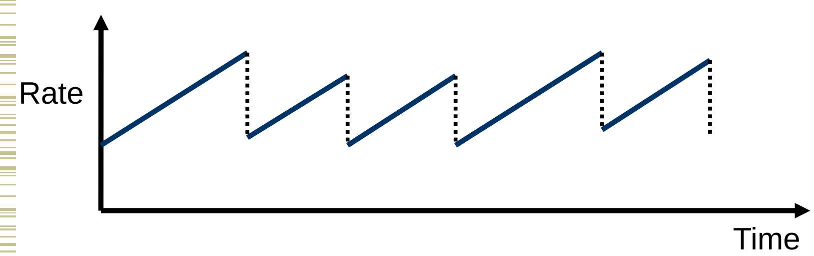
# SACK






# AIMD

- Distributed, fair and efficient
- Packet loss is seen as sign of congestion and results in a multiplicative rate decrease
  - Factor of 2
- TCP periodically probes for available bandwidth by increasing its rate

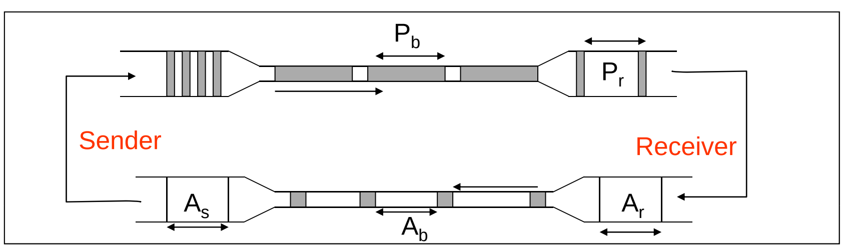





# TCP Packet Pacing

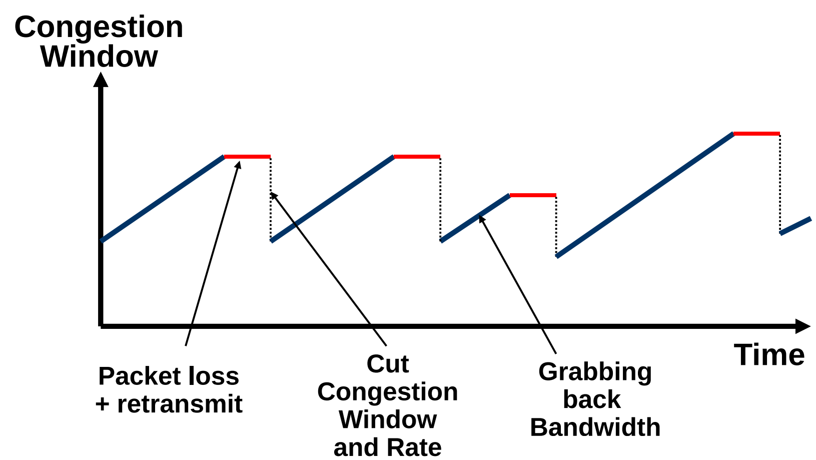
- Congestion window helps to "pace" the transmission of data packets
- In steady state, a packet is sent when an ack is received
  - Data transmission remains smooth, once it is smooth
  - Self-clocking behavior

Packet Conservation



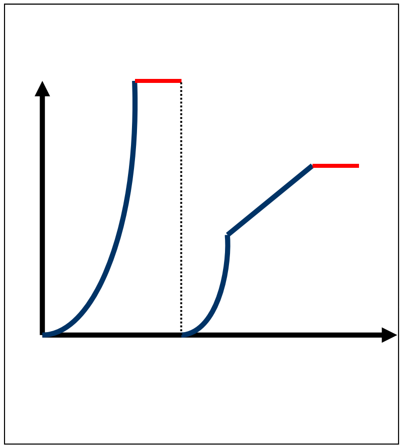



# Congestion Avoidance Behavior



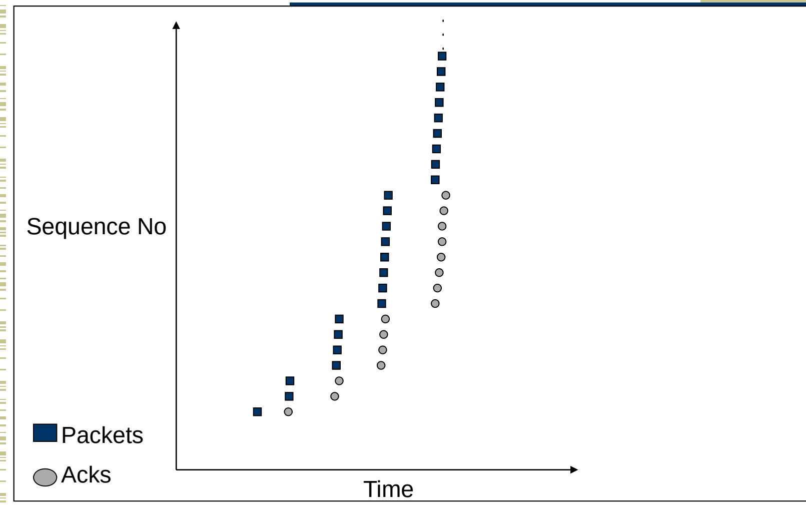
# Slow Start Packet Pacing

- How do we get this clocking behavior to start?
  - Initialize cwnd = 1
  - Upon receipt of every ack, cwnd = cwnd + 1
- Implications
  - Window actually increases to W in RTT * $\log_2(W)$
  - Can overshoot window and cause packet loss



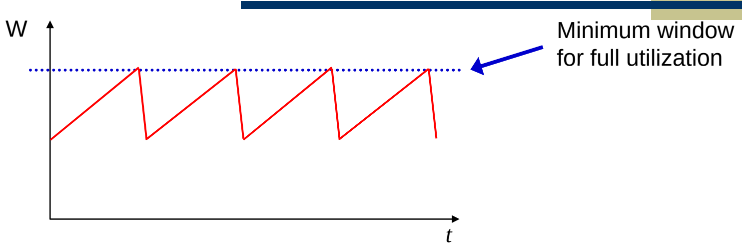


# Slow Start Sequence Plot





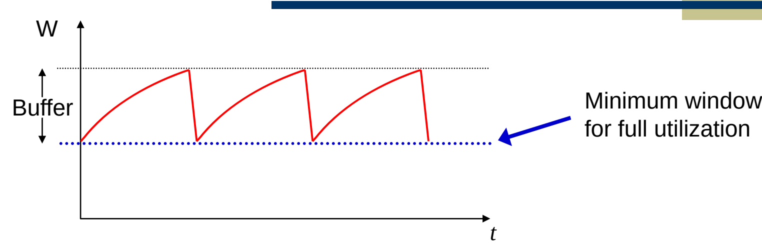

# Summary Unbuffered Link




- The router can't fully utilize the link
  - If the window is too small, link is not full
  - If the link is full, next window increase causes drop
  - With no buffer it still achieves 75% utilization



# Summary Buffered Link




- With sufficient buffering we achieve full link utilization
  - The window is always above the critical threshold
  - Buffer absorbs changes in window size
    - Buffer Size = Height of TCP Sawtooth
    - Minimum buffer size needed is RTT * BW
  - Delay? Between RTT and 2*RTT

## TCP (Summary)

- General loss recovery
  - Stop and wait
  - Selective repeat
- TCP sliding window flow control
- TCP state machine
- TCP loss recovery
  - Timeout-based
    - RTT estimation
  - Fast retransmit
  - Selective acknowledgements



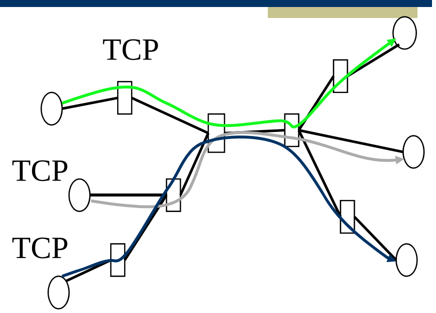
## TCP (Summary)

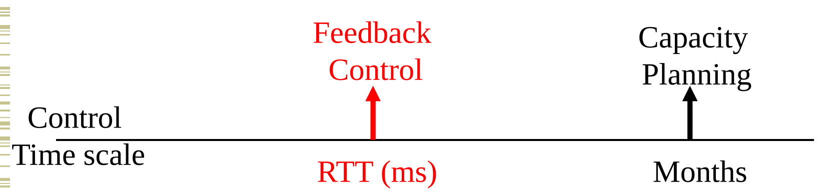
- Congestion collapse
  - Definition & causes
- Congestion control
  - Why AIMD?
  - Slow start & congestion avoidance modes
  - ACK clocking
  - Packet conservation
- TCP performance modeling
  - How does TCP fully utilize a link?
    - Role of router buffers



## Congestion Control in Today's Internet

- End-system-only solution (TCP)
  - dynamically estimates network state
  - packet loss signals congestion
  - reduces transmission rate in presence of congestion
  - routers play little role

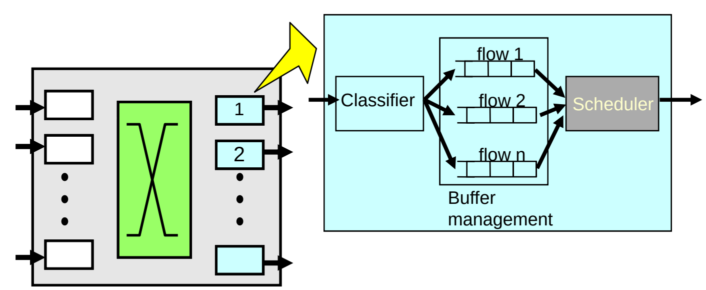





## Router Mechanisms

- Buffer management: when and which packet to drop?
- Scheduling: which packet to transmit next?

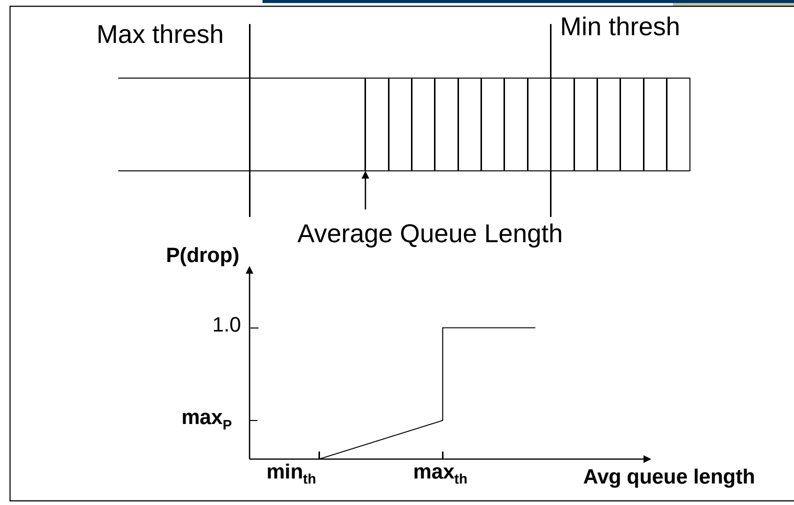


# Typical Internet Queueing

- FIFO (scheduling discipline) + drop-tail (drop policy)
  - Cong control at edges
  - No flow differentiation
  - Lock out
    - Random drop
    - Drop front
  - Full queues
    - Early random drop (RED)
    - Explicit congestion notification
    - decbit



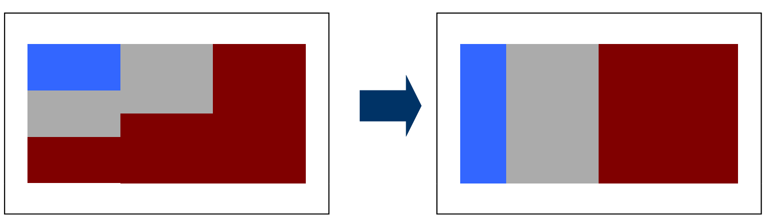
# RED Operation





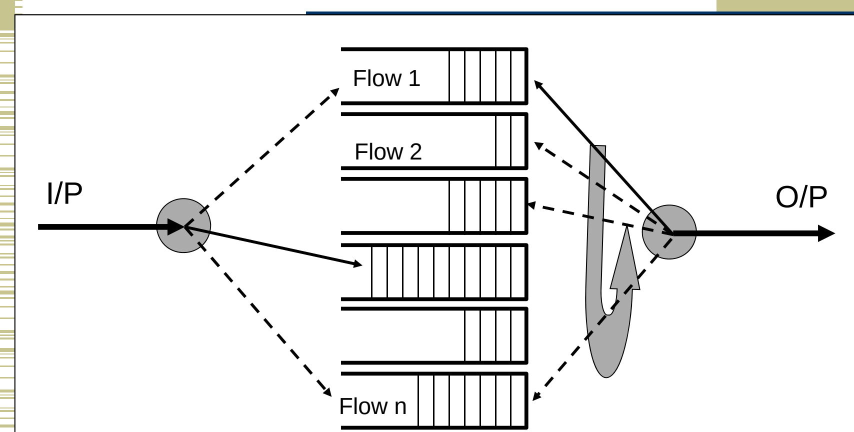

# Fair Queuing

- Mapping bit-by-bit schedule onto packet transmission schedule
- Transmit packet with the lowest $F_i$ at any given time
  - How do you compute $F_i$?



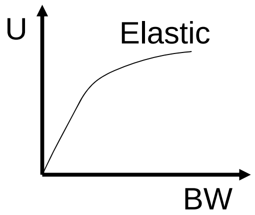


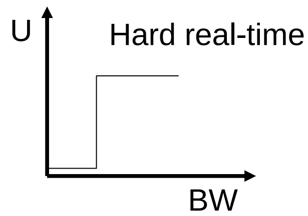
# FQ Illustration

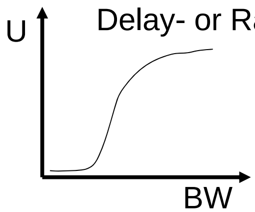



Variation: Weighted Fair Queuing (WFQ)

# Utility Curve Shapes



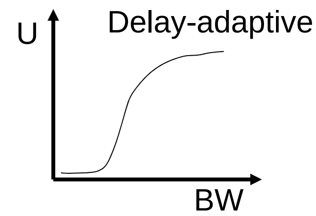




Stay to the right and you are fine for all curves



# Admission Control

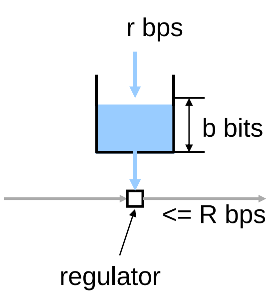
- If U is convex → inelastic applications
  - U(number of flows) is no longer monotonically increasing
  - Need admission control to maximize total utility
- **Admission control** → deciding when adding more people would reduce overall utility
  - Basically avoids overload

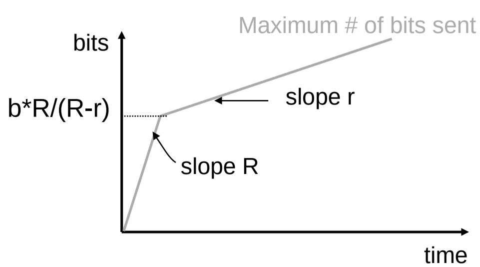





# Token Bucket

- Parameters
  - r – average rate, i.e., rate at which tokens fill the bucket
  - b – bucket depth
  - R – maximum link capacity or peak rate (optional parameter)
- A bit is transmitted only when there is an available token



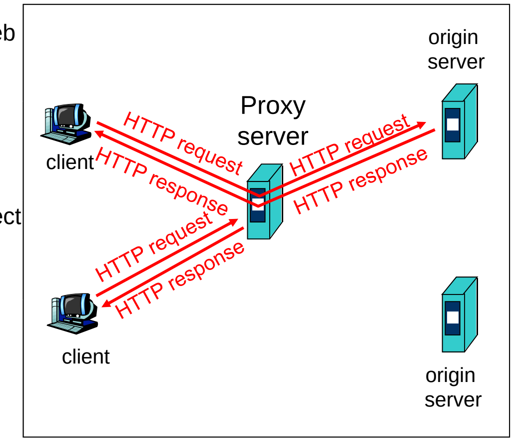



# Guarantee Proven by Parekh

- Given:
  - Flow *i* shaped with token bucket and leaky bucket rate control (depth *b* and rate *r*)
  - Network nodes do WFQ
- Cumulative queuing delay $D_i$ suffered by flow *i* has upper bound
  - **$D_i$ < b/r,** (where r may be much larger than average rate)
  - Assumes that $\Sigma r$ < link speed at any router
  - All sources limiting themselves to *r* will result in no network queuing

# Web Proxy Caches

- User configures browser: Web accesses via cache
- Browser sends all HTTP requests to cache
  - Object in cache: cache returns object
  - Else cache requests object from origin server, then returns object to client






# W/Caching Example (3)

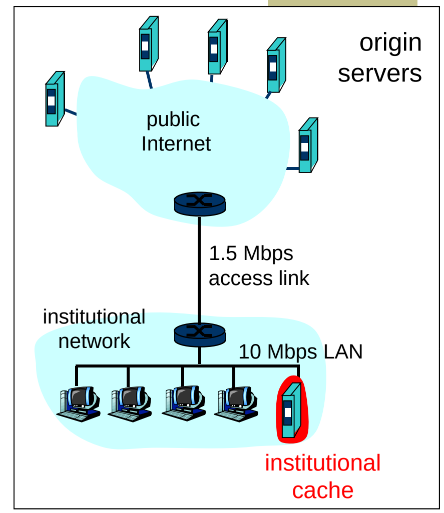
Install cache

- Suppose hit rate is .4

Consequence

- 40% requests will be satisfied almost immediately (say 10 msec)
- 60% requests satisfied by origin server
- Utilization of access link reduced to 60%, resulting in negligible delays
- Weighted average of delays

= .6*2 sec + .4*10msecs < 1.3 secs



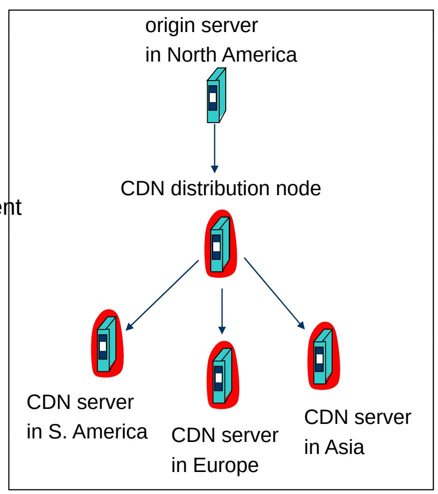



# Content Distribution Networks (CDNs)

- The content providers are the CDN customers.

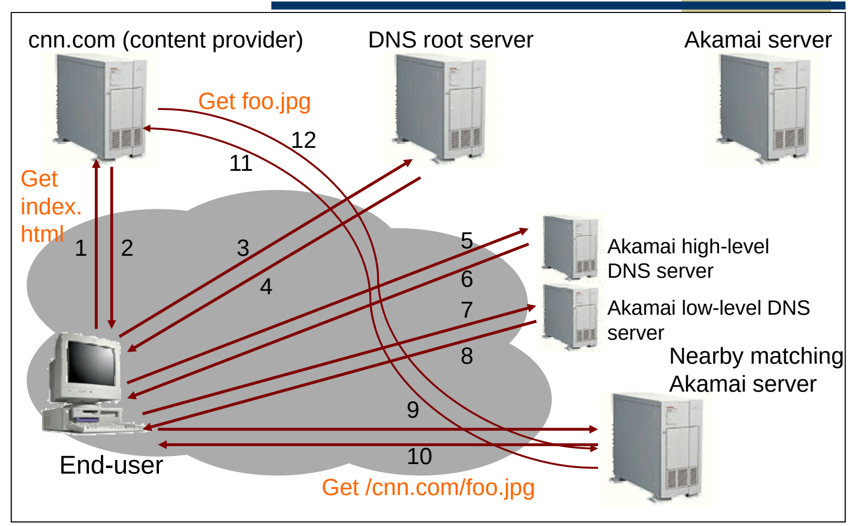
Content replication

- CDN company installs hundreds of CDN servers throughout Internet
  - Close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers

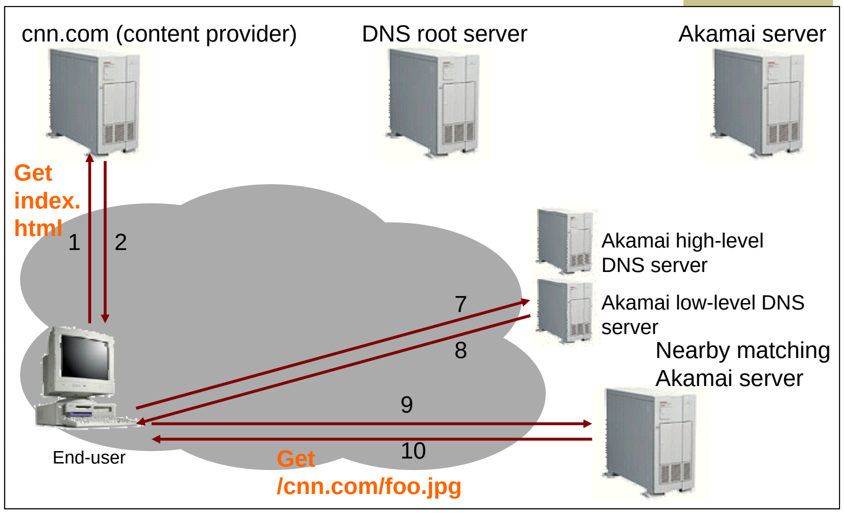





# How Akamai Works



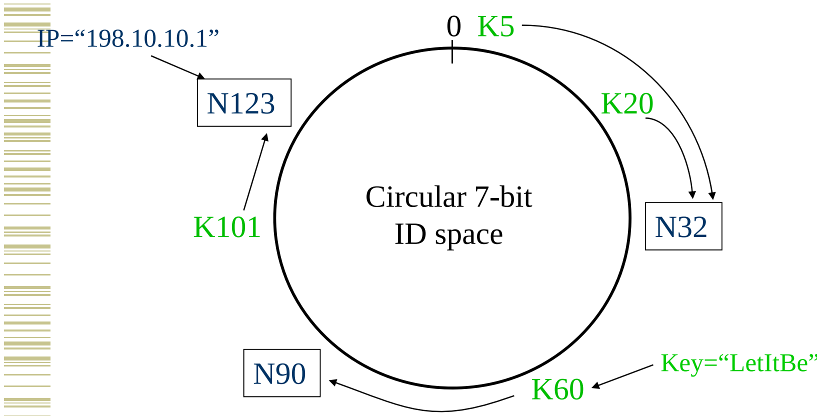
## Akamai – Subsequent Requests






## Consistent Hashing Example

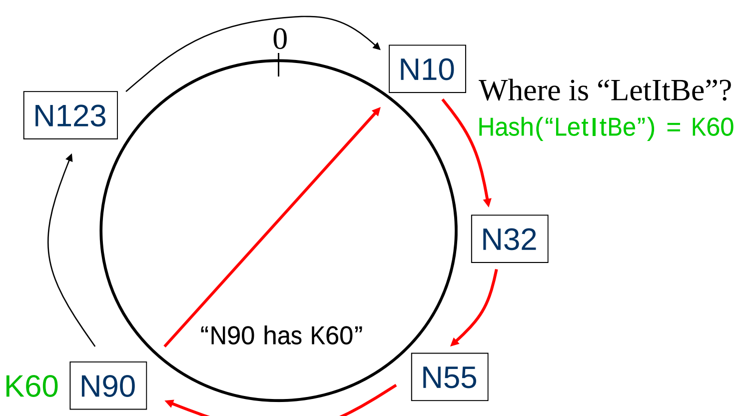
**Rule: A key is stored at its successor: node with next higher or equal ID**






## Lookups strategies

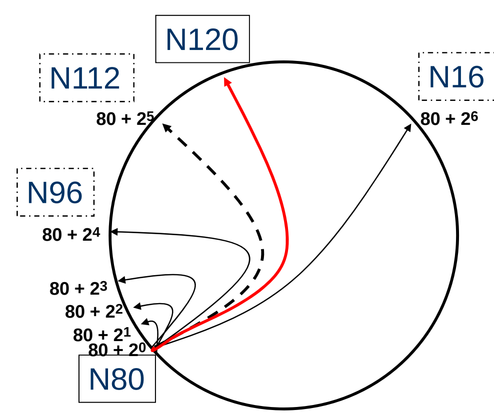
- Every node knows its successor in the ring
- Requires *O(N)* lookups





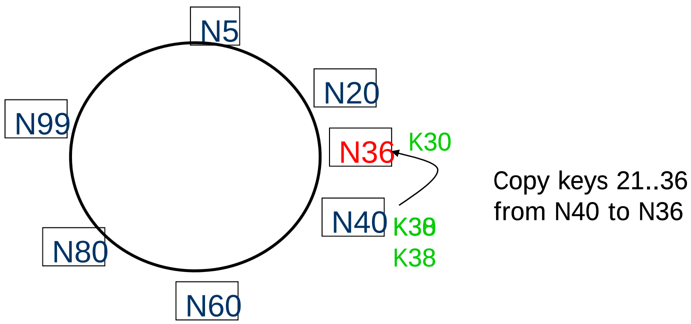

## Reducing Lookups: Finger Tables

- Each node knows *m* other nodes in the ring (it has m fingers)
- Increase distance exponentially
- Finger *i* points to successor of $n+2^{i-1}$ i=1..m



## Join: Transfer Keys

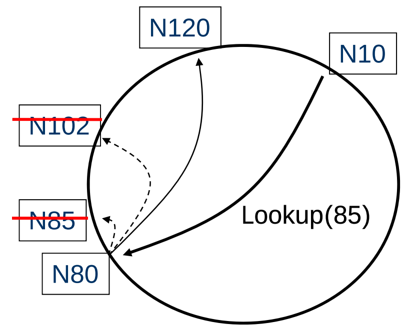
- Only keys in the range are transferred






## Handling Failures

- **Problem:** Failures could cause incorrect lookup
- **Solution:** *Fallback:* keep track of a list of immediate successors



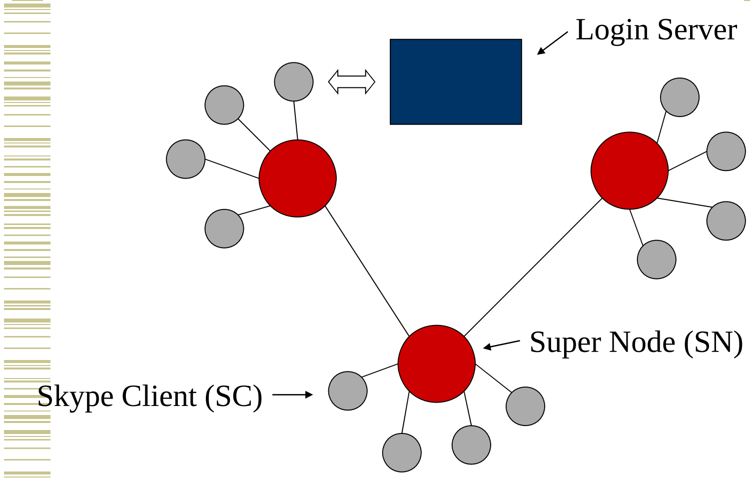



## Approaches to P2P

- Centralized
- Flooding
- Supernodes
- Routing
  - Structured
  - Un-structured



## Skype Architecture



# Routing Queries in Freenet

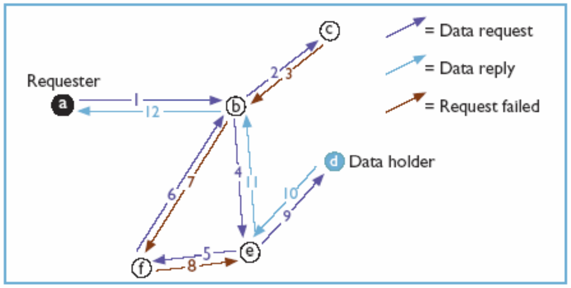



*Figure 1. Typical request sequence. The request moves through the network from node to node, backing out of a dead-end (step 3) and a loop (step 7) before locating the desired file.*

After success, node ***a*** creates a link in its routing table for the key to node ***d***.

Note: alternatively, any node on path from ***d*** to ***a***, e. g., ***e***, can name itself as originator of data.

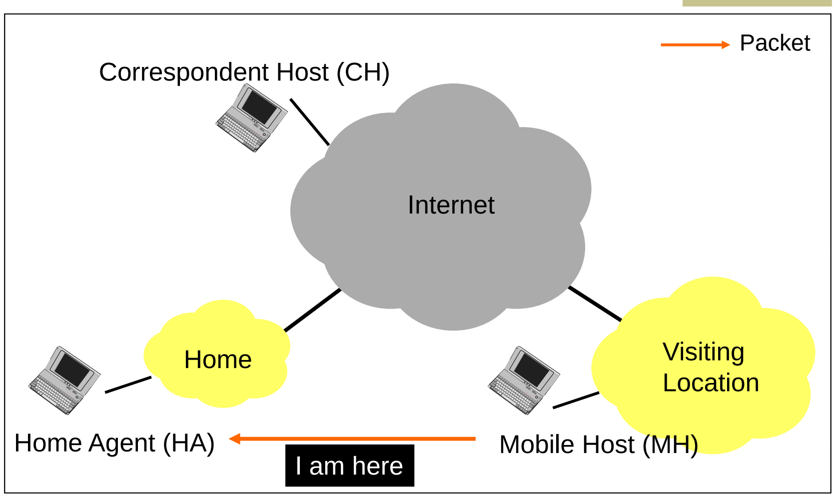


# Routing to Mobile Nodes

- Obvious solution: have mobile nodes advertise route to mobile address/32??
- What are some possible solutions?
  - DHCP? (changing IP?)
  - TCP?
  - Learning bridges (e.g., at CMU)
  - Encapsulated PPP
  - Interception & forwarding

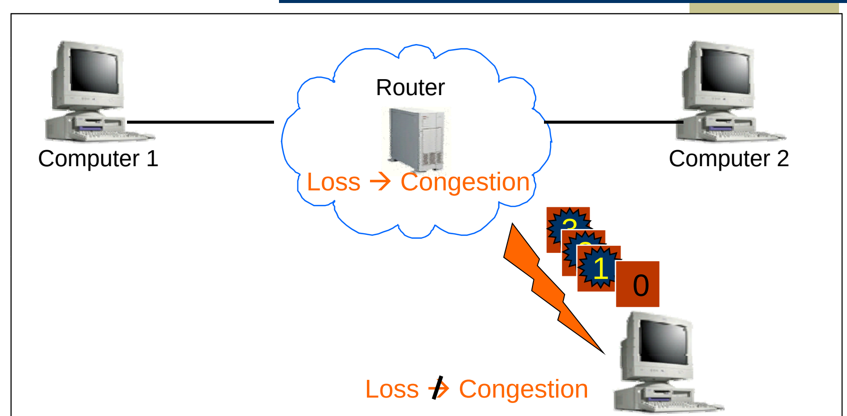


# Mobile IP (MH Moving)





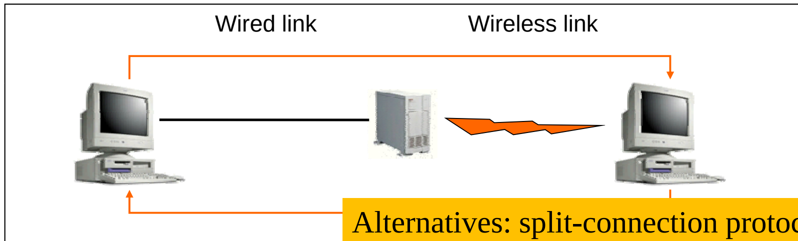

# Wireless Bit-Errors




Burst losses lead to coarse-grained timeouts
Result: Low throughput

# Approach Styles (End-to-End)

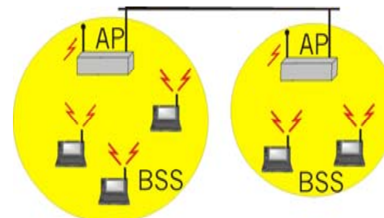
- Improve TCP implementations
  - Not incrementally deployable
  - Improve loss recovery (SACK, NewReno)
  - Help it identify congestion (ELN, ECN)
    - ACKs include flag indicating wireless loss
  - Trick TCP into doing right thing → E.g. send extra dupacks




Alternatives: split-connection protocols, ECC, local retransmit

# IEEE 802.11 Wireless LAN

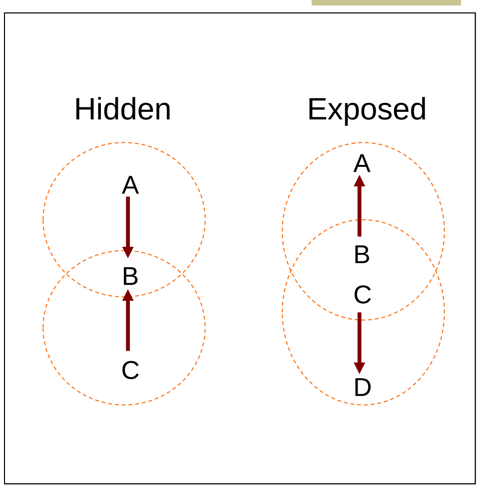
- Wireless host communicates with a base station
  - Base station = access point (AP)
- **Basic Service Set (BSS) (a.k.a. "cell") contains:**
  - **Wireless hosts**
  - **Access point (AP): base station**
- **BSS's combined to form distribution system (DS)**






# CSMA/CD Does Not Work

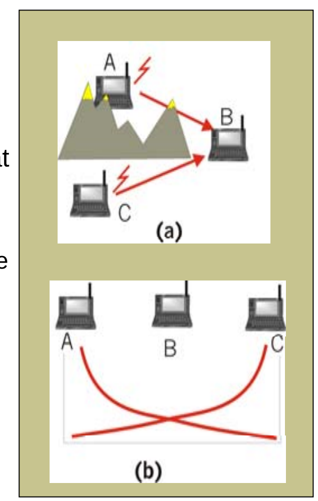
- Collision detection problems
  - Relevant contention at the receiver, not sender
    - Hidden terminal
    - Exposed terminal
  - Hard to build a radio that can transmit and receive at same time






# Hidden Terminal Effect

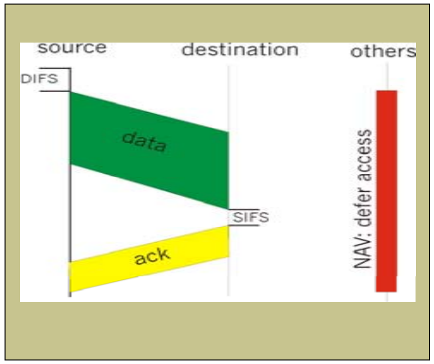
- Hidden terminals: A, C cannot hear each other
  - Obstacles, signal attenuation
  - Collisions at B
  - Collision if 2 or more nodes transmit at same time
- CSMA makes sense:
  - Get all the bandwidth if you're the only one transmitting
  - Shouldn't cause a collision if you sense another transmission
- Collision detection doesn't work
- **CSMA/CA: CSMA with C**ollision **A**voidance



# IEEE 802.11 MAC Protocol: CSMA/CA

802.11 CSMA: sender
- If sense channel idle for **DISF (Distributed Inter Frame Space)**
  then transmit entire frame (no collision detection)
- If sense channel busy
  then binary backoff

802.11 CSMA receiver:
- If received OK
  return ACK after **SIFS (Short IFS)**
  (ACK is needed due to lack of collision detection)






# Important Lessons

- Many assumptions built into Internet design
  - Wireless forces reconsideration of issues
- Link-layer
  - Spatial reuse (cellular) vs wires
  - Hidden/exposed terminal
  - CSMA/CA (why CA?) and RTS/CTS
- Network
  - Mobile endpoints – how to route with fixed identifier?
  - Link layer, naming, addressing and routing solutions
    - What are the +/- of each?
- Transport
  - Losses can occur due to corruption as well as congestion
    - Impact on TCP?
  - How to fix this → hide it from TCP or change TCP



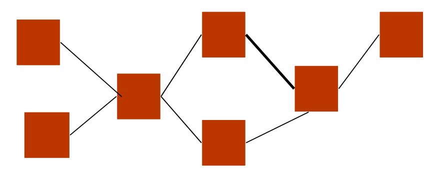
# Ad Hoc Networks

- All the challenges of wireless, plus:
  - No fixed infrastructure
  - Mobility (on short time scales)
  - Chaotically decentralized
  - Multi-hop!
- Nodes are both traffic sources/sinks and forwarders, no specialized routers
- The biggest challenge: <u>routing</u>

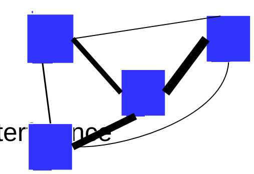


# Traditional Routing vs Ad Hoc

- Traditional network:
  - Well-structured
  - ~O(N) nodes & links
  - All links work ~= well

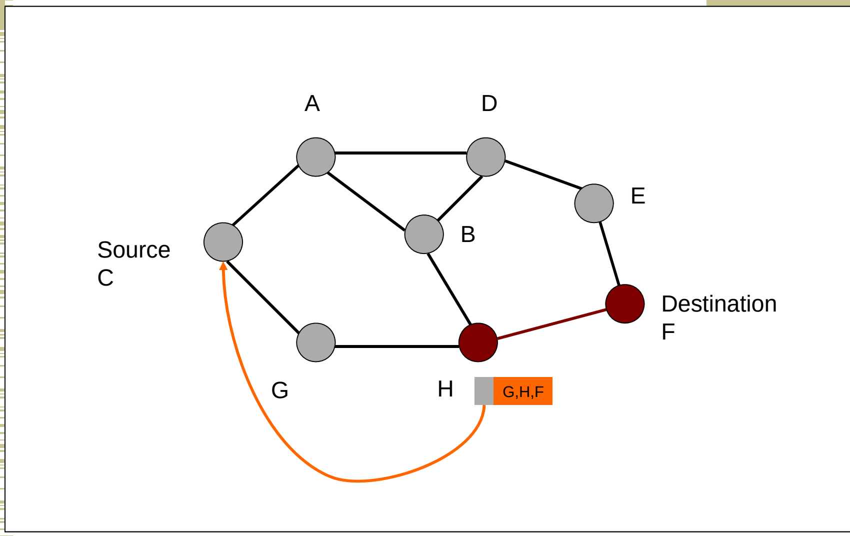


- Ad Hoc network
  - O(N^2) links - but most are bad!
  - Topology may be really weird
    - Reflections & multipath cause strange interference
  - Change is frequent



Traditional routing fails: DV loops, LS overhead, updates are power hungry, $N^2$ links: Instead proposed are: DSDV, AODV, DSR

# H Responds to Route Request



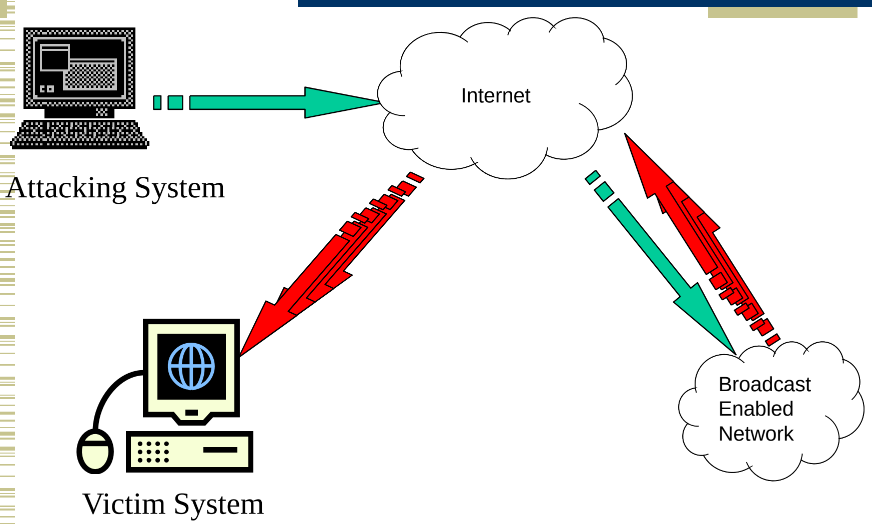



# Important Lessons

- Wireless is challenging
  - Assumptions made for the wired world don't hold
- Ad-hoc wireless networks
  - Need routing protocol but mobility and limited capacity are problems
  - On demand can reduce load; broadcast reduces overhead
- Special case 1 – Sensor networks
  - Power is key concern
  - Trade communication for computation
- Special case 2 – Vehicular networks
  - No power constraints but high mobility makes routing even harder, geographical routing

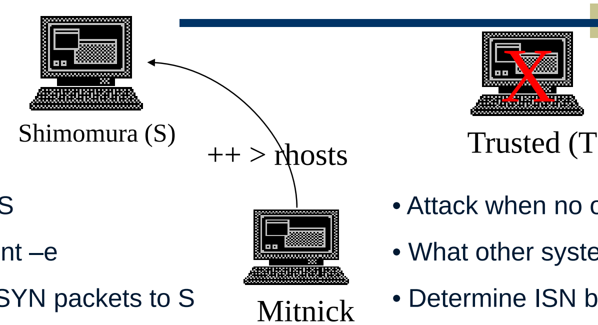


# Smurf Attack






# An Example

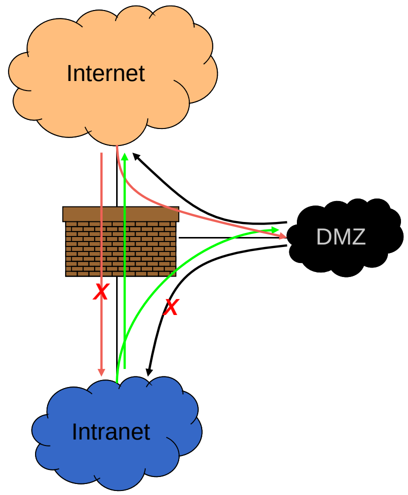



- Finger @S
- showmount –e
- Send 20 SYN packets to S
- SYN flood T
- Send SYN to S spoofing as T
- Send ACK to S with a guessed number
- Send "echo + + > ~/.rhosts"

- Attack when no one is around
- What other systems it trusts?
- Determine ISN behavior
- T won't respond to packets
- S assumes that it has a session with T
- Give permission to anyone from anywhere

# Typical Firewall Configuration

- Internal hosts can access DMZ and Internet
- External hosts can access DMZ only, not Intranet
- DMZ hosts can access Internet only
- Advantages?
  - If a service gets compromised in DMZ it cannot affect internal hosts






# Sample Firewall Rule

Allow SSH from external hosts to internal hosts

Two rules

Inbound and out

How to know a p

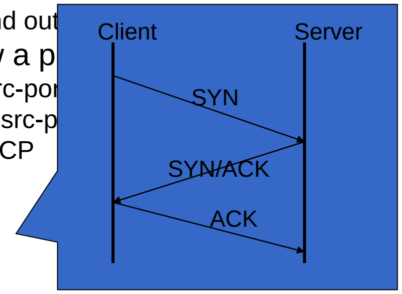
Inbound: src-por

Outbound: src-p

Protocol=TCP

Ack Set?

Problems?




| Rule | Dir | Src Addr | Src Port | Dst Addr | Dst Port | Proto | Ack Set? | Action |
|---|---|---|---|---|---|---|---|---|
| SSH-1 | In | Ext | > 1023 | Int | 22 | TCP | Any | Allow |
| SSH-2 | Out | Int | 22 | Ext | > 1023 | TCP | Yes | Alow |



# What do we need for a secure comm channel?

- Authentication (Who am I talking to?)
- Confidentiality (Is my data hidden?)
- Integrity (Has my data been modified?)
- Availability (Can I reach the destination?)

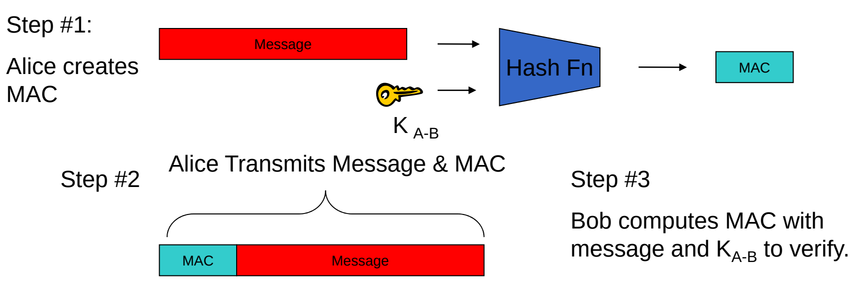


# The Great Divide

| | Symmetric Crypto (Private key) (E.g., AES) | Asymmetric Crypto (Public key) (E.g., RSA) |
|---|---|---|
| Shared secret between parties? | Yes | No |
| Speed of crypto operations | Fast | Slow |

# Symmetric Key: Integrity

- Hash Message Authentication Code (HMAC)

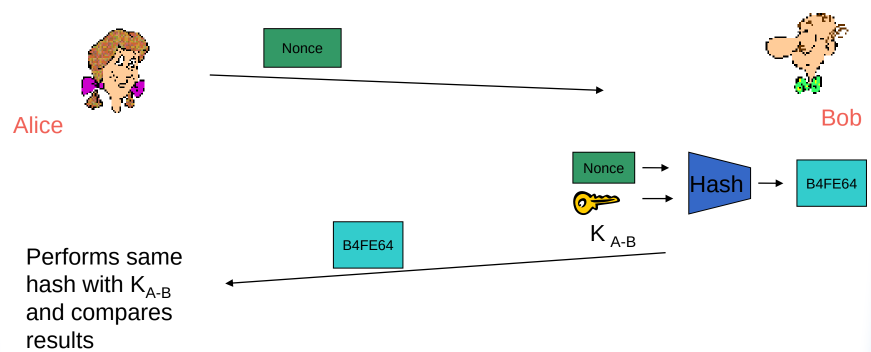



Why is this secure?
How do properties of a hash function help us?



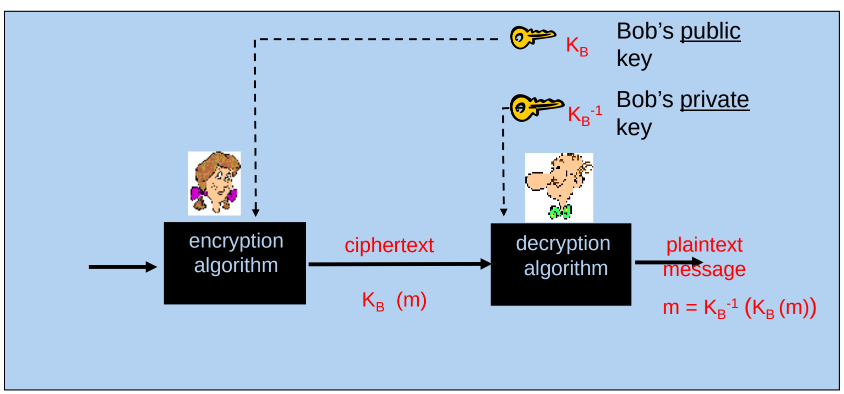
# Symmetric Key: Authentication

- A "Nonce"
  - A random bitstring used only once. Alice sends nonce to Bob as a "challenge". Bob Replies with "fresh" MAC result.





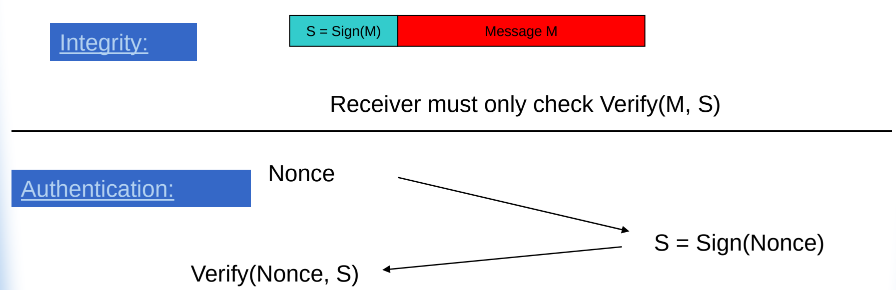

# Asymmetric Key: Confidentiality





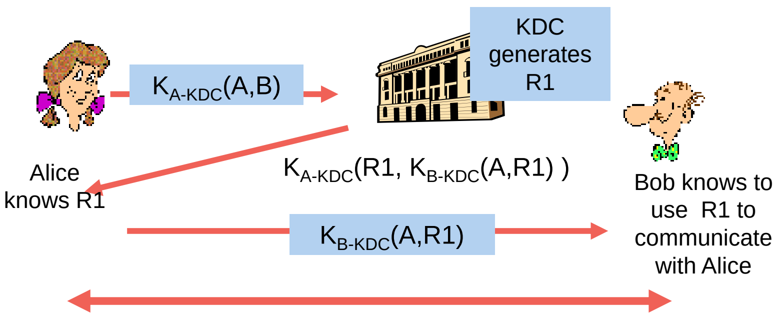

# Asymmetric Key: Integrity & Authentication

- We can use Sign() and Verify() in a similar manner as our HMAC in symmetric schemes.



# Key Distribution Center (KDC)

*Q:* How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?




Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption



# Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- An entity E registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - Certificate contains E's public key AND the CA's signature of E's public key.