Computer Network

Santiago

May 1, 2020

摘要

Prof.David Wetherall from University of Washington
Computer Network Course

1 Week 6.1 - Transport Layer Overview

1.1 Where we are in the Course

- Starting the Transport Layer!
  - Builds on the network layer to deliver data across networks for applications with desired reliability or quality

1.2 Recall

- Transport layer provides end-to-end connectivity across the network
- See Figure 1
- Segments carry application data across the network
- Segments are carried within packets within frames
- See Figure 2

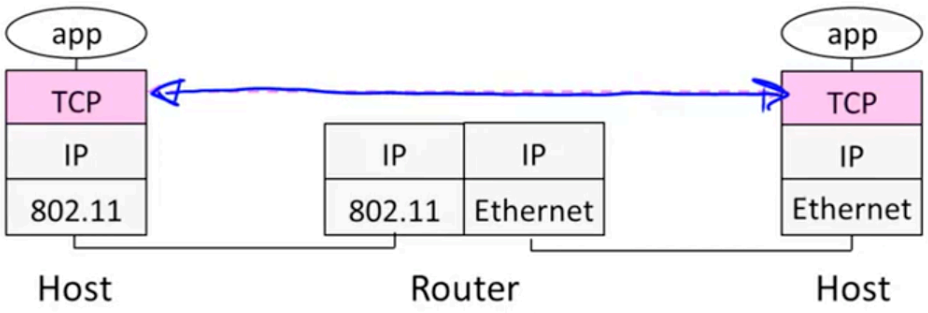


图 1: Recall 图一

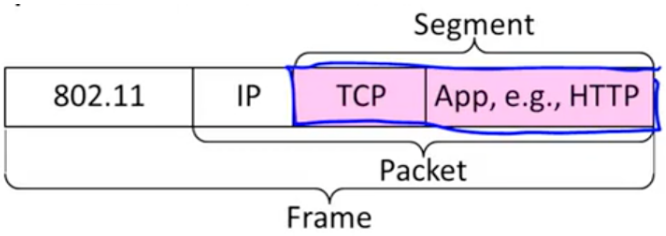


图 2: Recall 图二

1.3 Transport Layer Services

- Provide different kinds of data delivery across the network to applications
- See Figure 3

	Unreliable	Reliable
Messages	Datagrams (UDP)	
Bytestream		Streams (TCP)

图 3: Transport Layer Services

1.4 Comparison of Internet Transports

- TCP is full-featured, UDP is a glorified packet
- See Figure 4

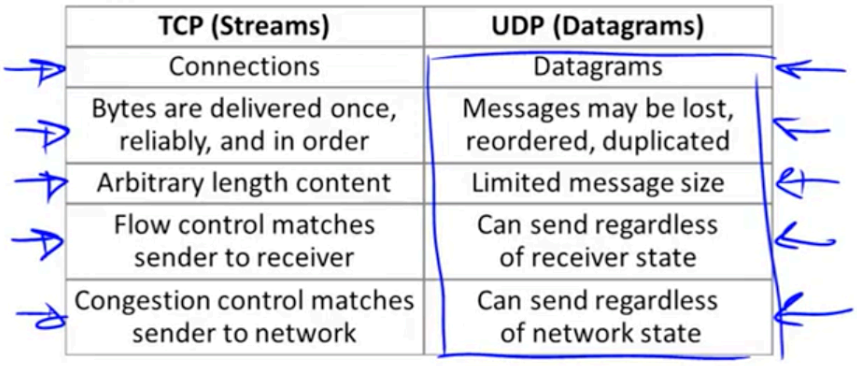


图 4: Comparison of Internet Transports

1.5 Socket API

- Simple abstraction to use the network
  - The "network" API (really Transport service) used to write all Internet apps
  - Part of all major OSes and languages; originally Berkeley (Unix) 1983
- Supports both Internet transport services (Streams and Datagrams)
- Sockets let apps attach to the local network at different ports
- See Figure 5
- Same AIP used for streams and Datagrams
- See Figure 6

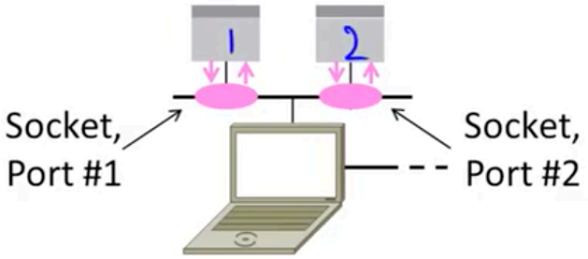


图 5: Socket API 图一

	Primitive	Meaning
Only needed for Streams	SOCKET	Create a new communication endpoint
	BIND	Associate a local address (port) with a socket
	LISTEN	Announce willingness to accept connections
	ACCEPT	Passively establish an incoming connection
	CONNECT	Actively attempt to establish a connection
To/From forms for Datagrams	SEND(TO)	Send some data over the socket
	RECEIVE(FROM)	Receive some data over the socket
	CLOSE	Release the socket

图 6: Socket API 图二

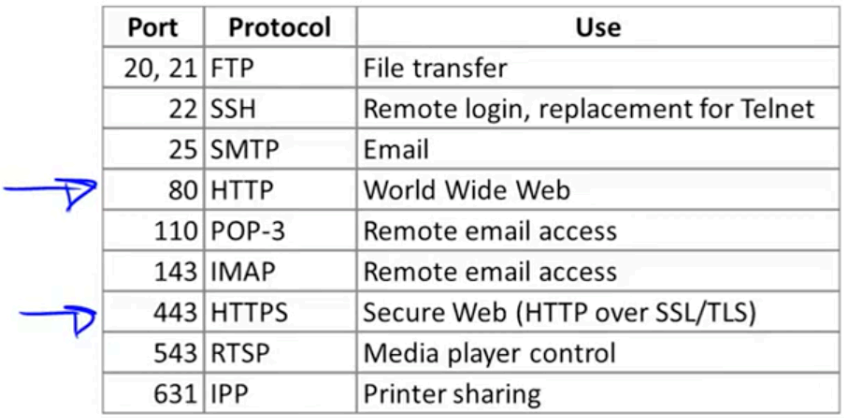
1.6 Ports

- Application process is identified by the tuple IP address, protocol, and port
  - Ports are 16-bit integers representing local ”mailboxes” that a process leases(租)
- Servers often bind to ”well-known ports”
  - <1024, require administrative privileges
- Clients often assigned ”ephemeral(短暂的)” ports

  - Chosen by OS, used temporarily

1.7 Some Well-Known Ports

- See Figure 7



Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

图 7: Some Well-Known Ports

1.8 Topics

- 下面是本次讨论的
- Service models
  - Socket API and ports
  - Datagrams, Streams
- 下面是下一次讨论内容
- User Datagram Protocol (UDP)
- Connections (TCP)

- Sliding Window (TCP)
- Flow control (TCP)
- Retransmission(中继；转播；重发) timers (TCP)
- 下面是再后来谈的内容
- Congestion control (TCP)

2 Week 6.2 - User Datagram Protocol (UDP)

2.1 Topic of UDP

- Sending messages with UDP
  - A shim(用木片或夹铁填孔隙) layer on packets
- See Figure 8

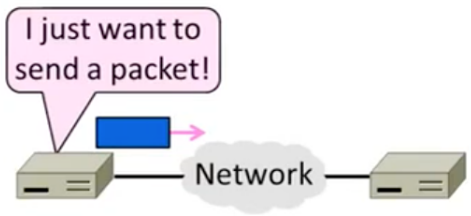


图 8: Topic of UDP

2.2 User Datagram Protocol (UDP)

- Used by apps that don't want reliability or bytestreams
  - Voice-over-IP (unreliable)

  - DNS, RPC (message-oriented)
  - DHCP (bootstrapping)

- (If application wants reliability and messages then it has work to do !)

2.3 Datagram Sockets

- See Figure 9

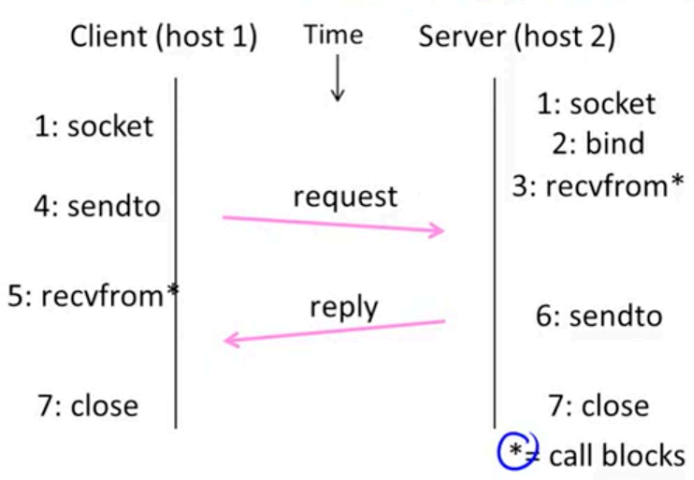


图 9: Datagram Sockets

2.4 UDP Buffering

- 下图中的消息队列就是一个 Buffer，接收和发送都是通过消息队列进行的，mux(multiplexer 多路调制器) 和 demex (解调器) 可以把数字量转换为模拟量传送到网络，并在另一个应用的端口进行解调。

- See Figure 10

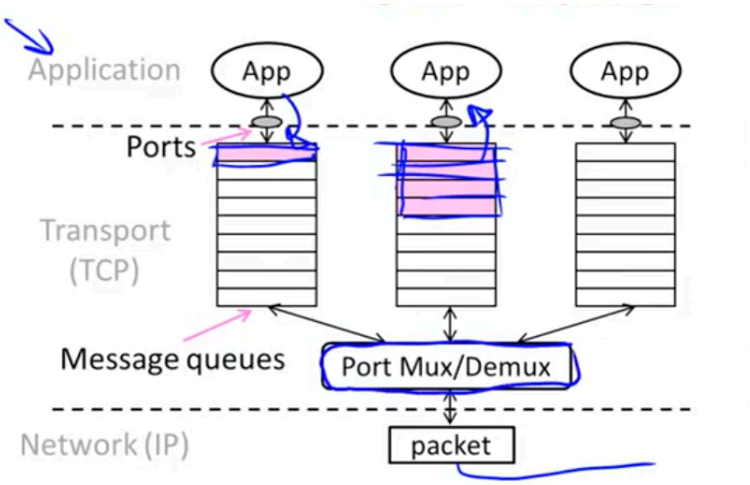


图 10: UDP Buffering

2.5 UDP Header

- Uses ports to identify sending and receiving application processes
- Datagram length up to 64K
- Checksum (16 bits) for reliability
- See Figure 11

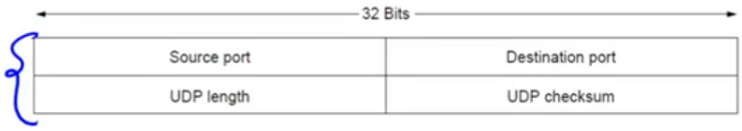


图 11: UDP Header 图一

- Optional checksum covers UDP segment and IP pseudoheader
  - Checks key IP fields (addresses)
  - Value of zero (全零) means "no checksum"

- See Figure 12

图 12: UDP Header 图二

3 Week 6.3 - Connection Establishment

3.1 Topic of Connection Establishment

- How to set up connections
  - We'll see how does it
- See Figure 13

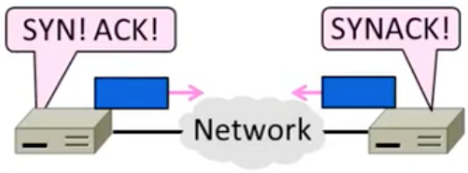


图 13: Topic of Connection Establishment

3.2 Connection Establishment

- Both sender and receiver must be ready before we start the transfer of data

  - Need to agree on a set of parameters
  - e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Like ”dialing” for a telephone call

3.3 Three-Way Handshake

- Used in TCP; opens connection for data in both directions
- Each side probes(探测) the other with a fresh Initial Sequence Number (ISN)
  - Sends on a SYNchronize(使同步) segment
  - Echo on an ACKnowlege(告知收到) segment
- Chosen to be robust event against delayed duplicates
- Three steps:
  - Client sends SYN(x)
  - Server replies with SYN(y) ACK(x+1)
  - Client replies with ACK(y+1)
  - SYNs are retransmitted if lost
- Sequence and ack numbers carried on further segments
- See Figure 14
- Suppose delayed, duplicate copies of the SYN and ACK arrive at the server!

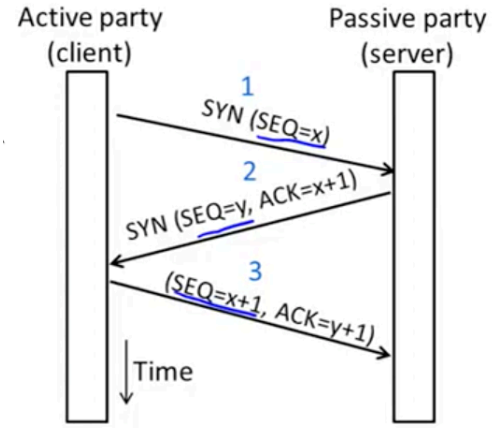


图 14: Three-Way Handshake 图一

- Improbable(不像会发生的), but anyhow(不管怎样) ...
- client 和 server 会拒绝这样的 signal。

- See Figure 15

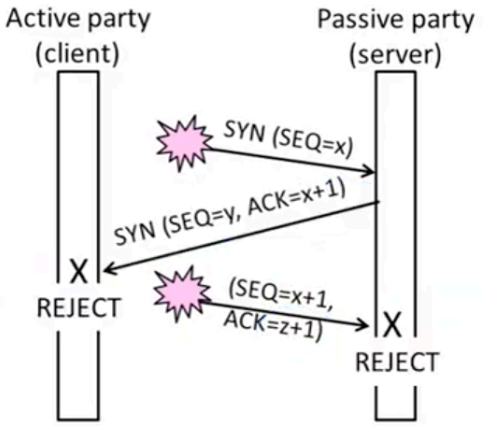


图 15: Three-Way Handshake 图二

- Connection will be cleanly rejected on both sides

3.4 TCP Connection State Machine

- Captures the states (rectangles) and transitions (arrows)
  - A/B means event A triggers(触发) the transition(转变), with action B
  - 比如下图 CONNECT 触发使得发出 SYS(Step1 of the 3-way handshake) , 注意：在 Client 和 Server 都运行这个状态机
- Follow the path of the client:
- See Figure 16

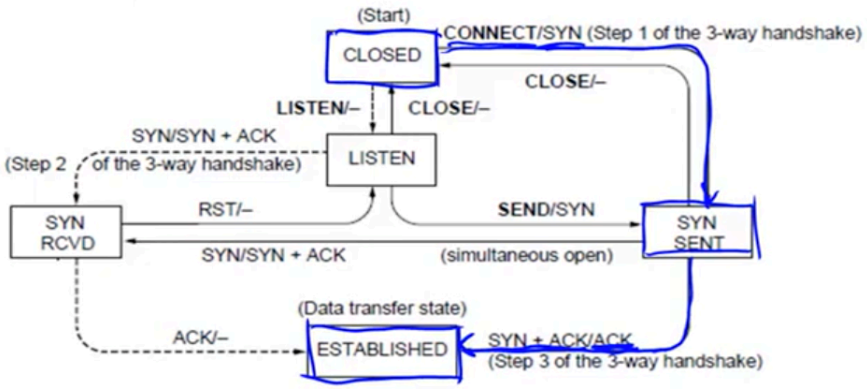


图 16: TCP Connection State Machine 图一

- And the path of the server:
- See Figure 17
- Again, with states ...
- See Figure 18
- Finite state machines are a useful tool to specify and check the handling of all cases the may occur

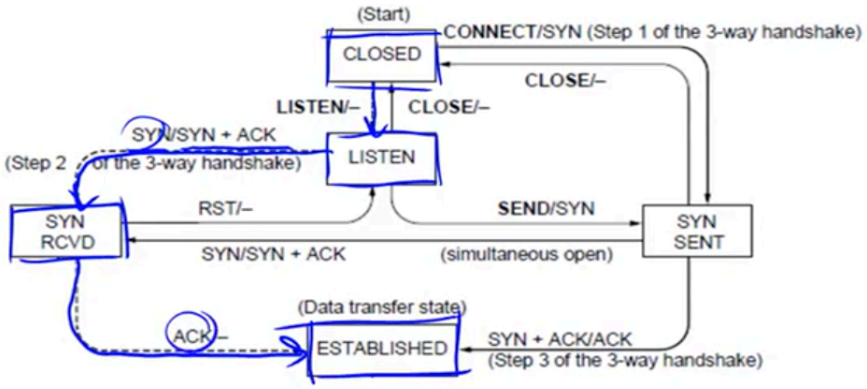


图 17: TCP Connection State Machine 图二

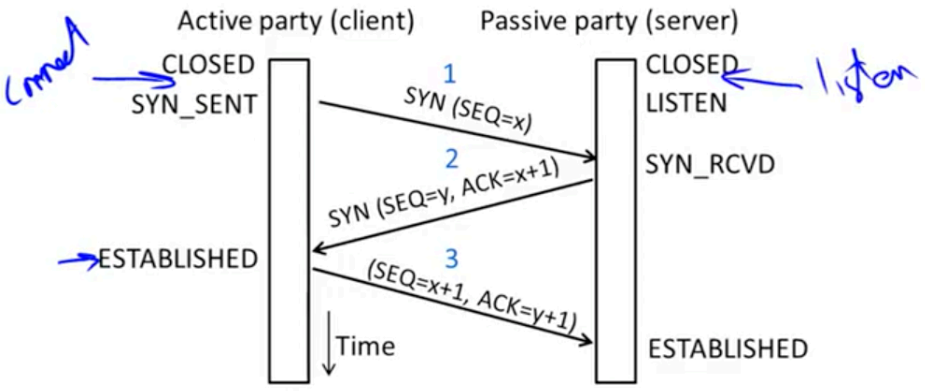


图 18: TCP Connection State Machine 图三

- TCP allows for simultaneous open
  - i.e., both sides open at once instead of the client-server pattern
  - Try at home to confirm it works

4 Week 6.4 - Connection Release

4.1 Topic of Connection Release

- How to release connections
  - We'll see how TCP does it
- See Figure 19

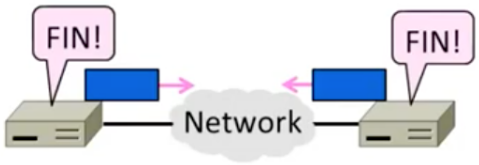


图 19: Topic of Connection Release

4.2 Connection Release

- Orderly release by both parties when done
  - Delivers all pending data and "hangs up"
  - Cleans up state in sender and receiver
- Key problem is to provide reliability while releasing
  - TCP uses a "symmetric" close in which both sides shutdown independently

4.3 TCP Connection Release

- Two steps:
  - Active sends FIN(x), passive ACKs

  - Passive sends FIN(y), active ACKs
  - FINs are retransmitted if lost
- Each FIN/ACK closes one direction of data transfer
- See Figure 20

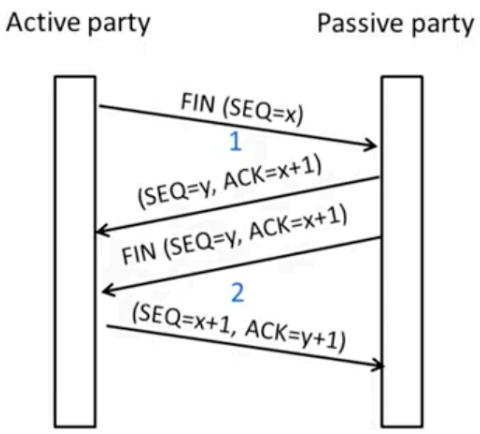


图 20: TCP Connection Release

4.4 TCP Connection State Machine

- See Figure 21

4.5 TCP Release

- Follow the active party
- See Figure 22
- Follow the passive party
- See Figure 23

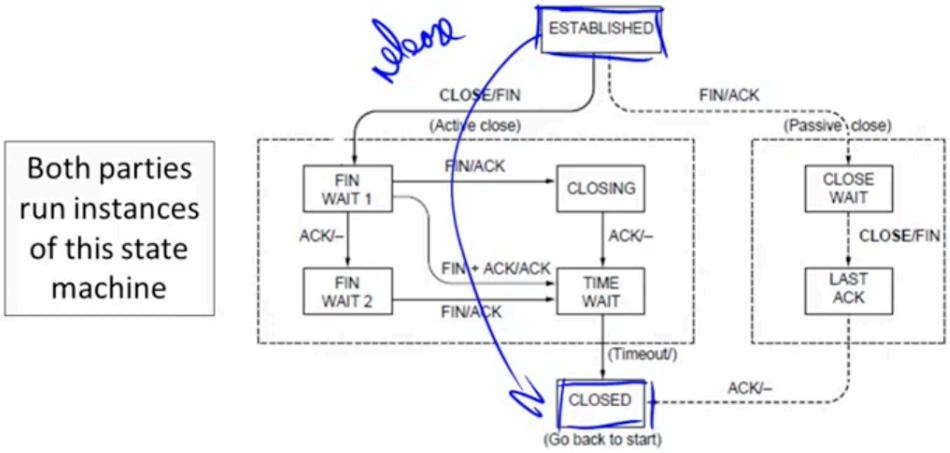


图 21: TCP Connection State Machine

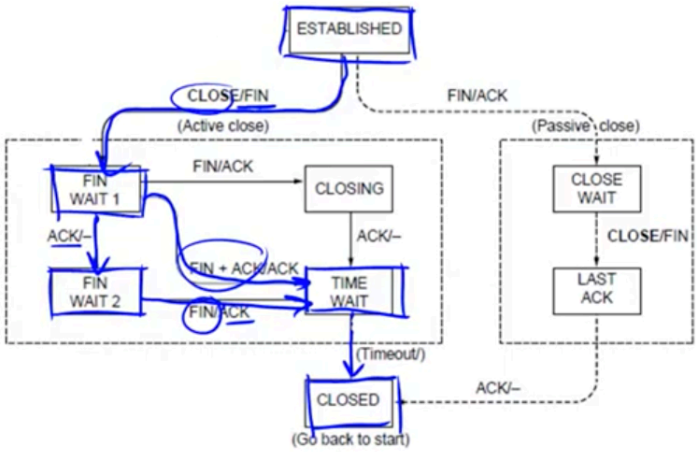


图 22: TCP Release 图一

- Again, with states ...
- See Figure 24

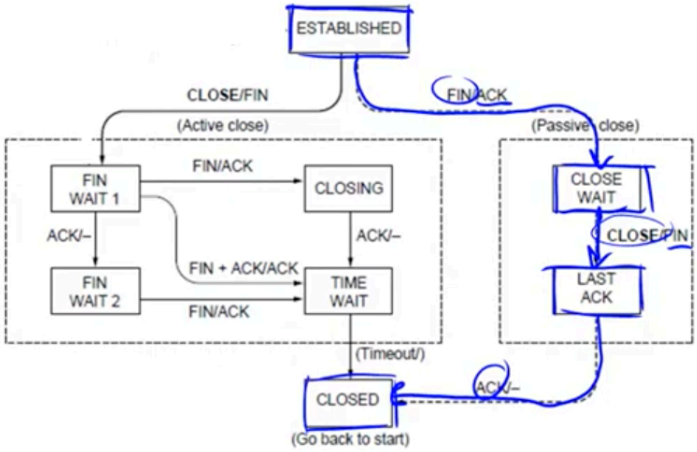


图 23: TCP Release 图二

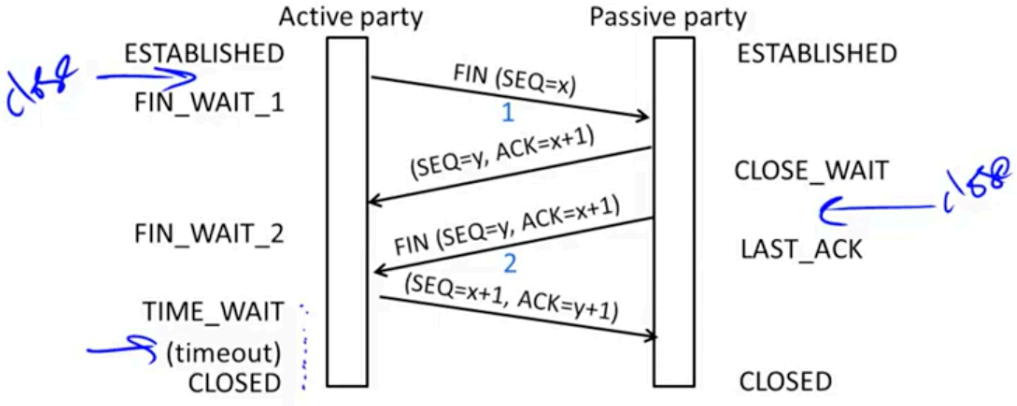


图 24: TCP Release 图三

4.6 TIME_WAIT State

- We wait a long time (two times the maximum segment lifetime of 60 seconds) after sending all segments and before completing the close
- Why?
  - ACK might have been lost, in which case FIN will be resent for

an orderly close

- Could otherwise interfere(干涉；妨碍) with a subsequent connection

5 Week 6.5 - Sliding Window

5.1 Topic of Sliding Window

- The sliding window algorithm
  - Pipelining and reliability
  - Building on Stop-and-Wait
- See Figure 25

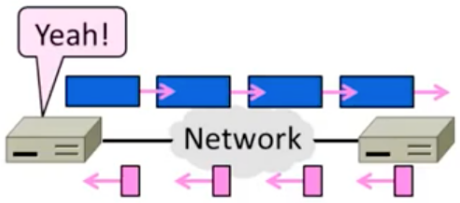


图 25: Topic of Sliding Window

5.2 Recall

- ARQ with one message at a time is Stop-and-Wait (normal case below)
- See Figure 26

5.3 Limitation of Stop-and-Wait

- It allows only a single message to be outstanding from the sender:

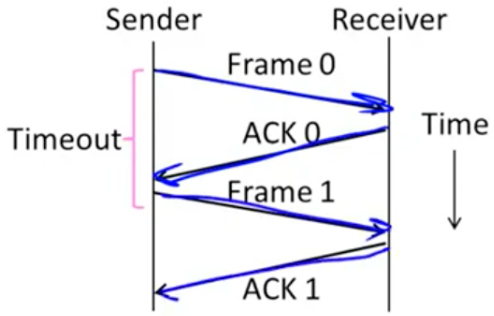


图 26: Recall

  - Fine for LAN (only one frame fit)
  - Not efficient for network paths with BD » 1 packet

- Example: R=1 Mbps, D=50 ms
  - RTT (Round Trip Time) = 2D = 100 ms
  - How meny packets/sec ?
    * 10 packets/sec = 100 kbps (10%)
  - What if R=10 Mbps?
    * 10 packets/sec = 100 kbps (1%)

5.4 Sliding Window

- Generalization of stop-and-wait
  - Allow W packets to be outstanding
  - Can send W packets per RTT (2D)
  - Pipelining improves performance
  - Need W=2BD to fill network path

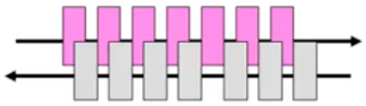


图 27: Sliding Window

- See Figure 27
- What W will use the network capacity?
- Ex: R=1 Mbps, D=50 ms
  - W = 2BD = $10^6 * 100 * 10^{-3}$ = 100Kb $\approx$ 10 packets
- Ex: What if R=10 Mbps?
  - W=2BD = 1000 Kb $\approx$ 100 packets

5.5 Sliding Window Protocol

- Many variations, depending on how buffers, acknowledgements, and retransmissions are handled
- Go-Back-N
  - Simplest version, can be inefficient
- Selective Repeat
  - More complex, better performance

5.6 Sliding Window - Sender

- Sender buffers up to W segments until they are acknowledged
  - LFS=LAST FRAME SENT, LAR=LAST ACK REC'D

- Sends while $LFS - LAR \leq W$

- See Figure 28

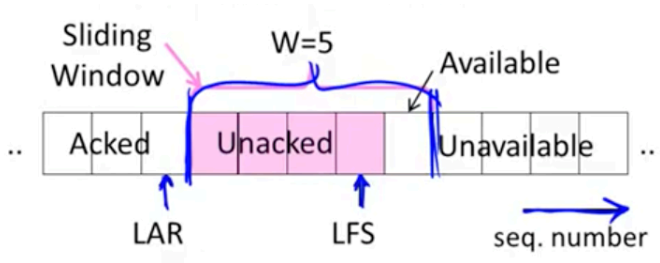


图 28: Sliding Window - Sender 图一

- Transport accepts another segment of data from the Application ...
  - Transport sends it (as $LFS - LAR \rightarrow 5$)

- See Figure 29

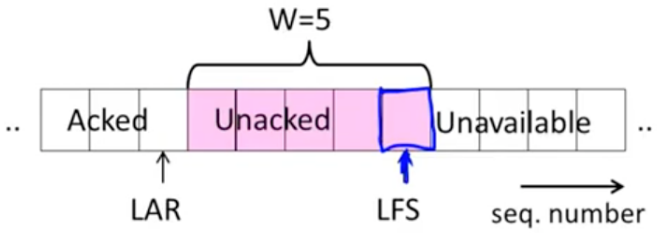


图 29: Sliding Window - Sender 图二

- Next higher ACK arrives from peer ...
  - Window advances, buffer is freed
  - $LFS - LAR \rightarrow 4$ (can send one more)

- See Figure 30

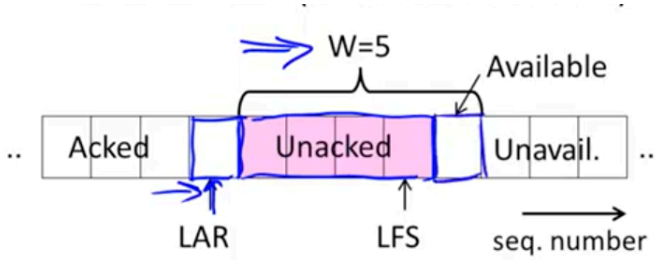


图 30: Sliding Window - Sender 图三

5.7 Sliding Window - Go-Back-N

- Receiver keeps only a single packet buffer for the next segment
  - State variable, LAS = LAST ACK SENT
- On receive:
  - If seq. number is LAS+1, accept and pass it to app, update LAS, send ACK
  - Otherwise discard (as out of order)

5.8 Sliding Window - Selective Repeat

- Receiver passes data to app in order, and buffers out-of-order segments to reduce retransmissions
- ACK conveys(传达，传输) highest in-order segment, plus hints about out-of-order segments
- TCP uses a selective repeat design; we'll see the details later
- Buffers W segments, keeps state variable, LAS = LAST ACK SENT
- On receive:

  - Buffer segments [LAS+1, LAS+W]
  - Pass up to app in-order segments from LAS+1, and update LAS
  - Send ACK for LAS regardless

5.9 Sliding Window - Retransmissions

- Go-Back-N sender uses a single timer to detect loss
  - On timeout, resends buffered packets starting at LAR+1
- Selective Repeat sender uses a timer per unacked segment to detect losses
  - On timeout for segment, resend it
  - Hope to resend fewer segments

5.10 Sequence Numbers

- Need more then 0/1 for Stop-and-Wait ...
  - But how many?
- For Selective Repeat, need W numbers for packets, plus W for acks of earlier packets
  - 2W seq. numbers
  - Fewer for Go-Back-N (W+1)
- Typically implement seq. number with an N-bit counter that wraps around at $2^N - 1$
  - E.g., N=8: ..., 253, 254, 255, 0, 1, 2, 3, ...

5.11 Sequence Time Plot

- See Figure 31

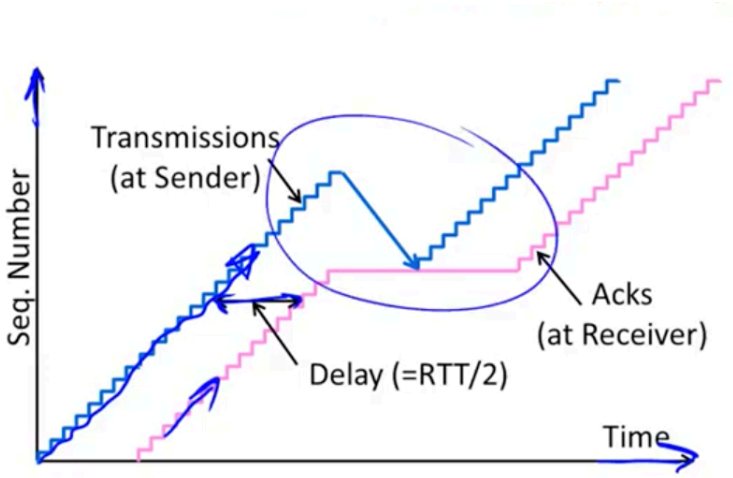


图 31: Sequence Time Plot 图一

- 上面图中的发送端曲折，是由于丢包导致的，下面图作一个分析。
- See Figure 32

6 Week 6.6 - Flow Control

6.1 Topic of Flow Control

- Adding flow control to the sliding window algorithm
  - To slow the over-enthusiastic sender
- See Figure 33

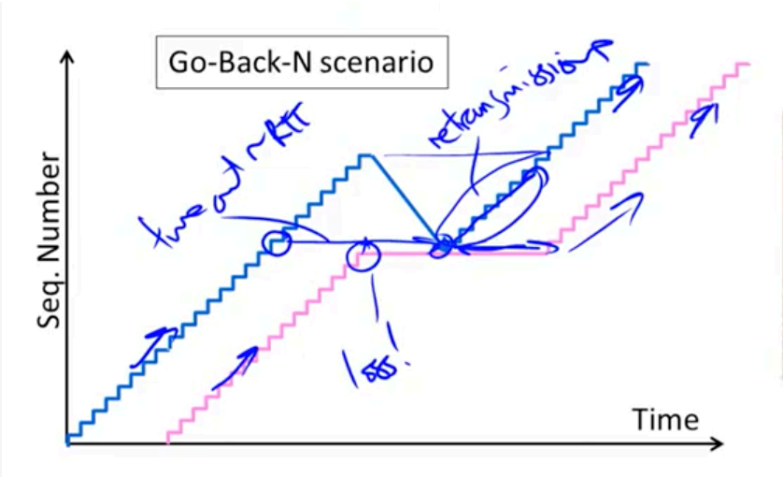


图 32: Sequence Time Plot 图二

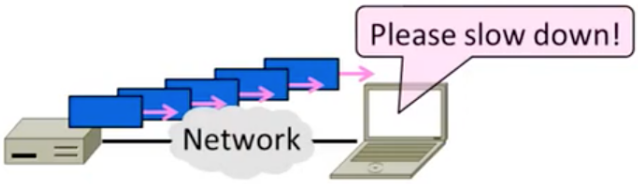


图 33: Topic of Flow Control

6.2 Problem

- Sliding window uses pipelining to keep the network busy
  - What if the receiver is overloaded?
- See Figure 34

6.3 Sliding Window - Receiver

- Consider receiver with W buffers

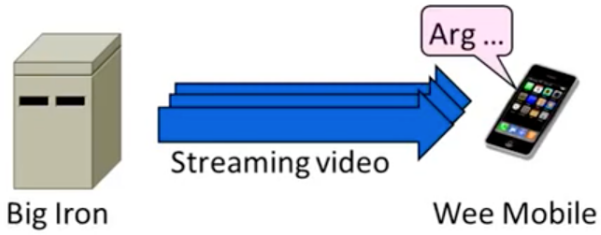


图 34: Problem

  - LAS=LAST ACK SENT, app pulls in-order data from buffer with recv() call

- See Figure 35

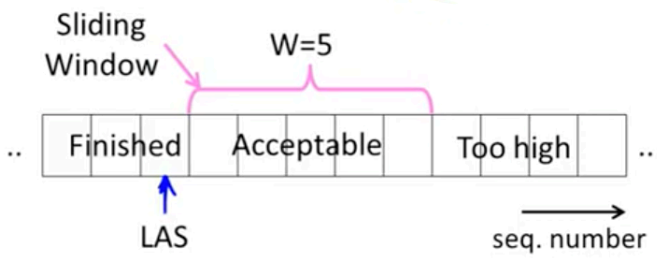


图 35: Sliding Window - Receiver 图一

- Suppose the next two segments arrive but app does not call recv()
  - LAS rises, but we can't slide window!
- See Figure 36
- If further segments arrive (even in order) we can fill the buffer
  - Must drop segments until app recvs!
- See Figure 37

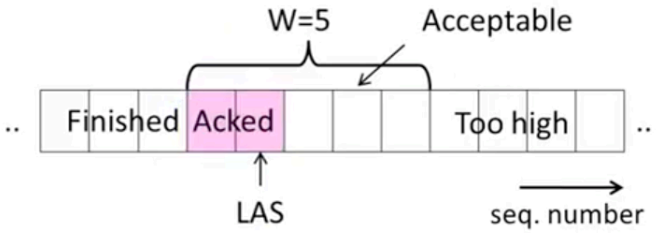


图 36: Sliding Window - Receiver 图二

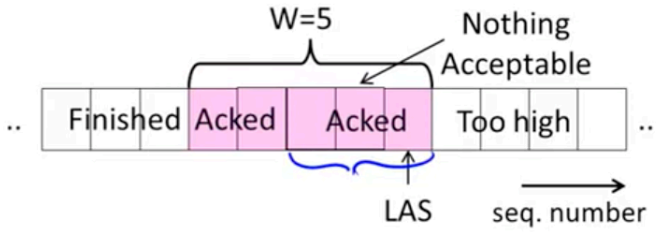


图 37: Sliding Window - Receiver 图三

- App recv() takes two segments
  - Window slides (phew 很慢很慢)
- See Figure 38

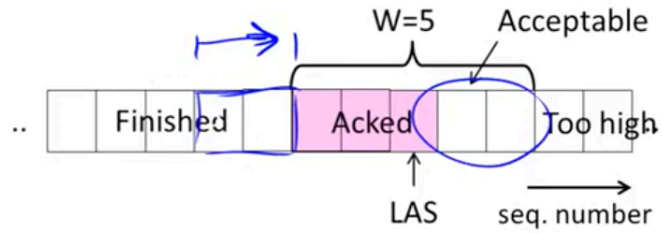


图 38: Sliding Window - Receiver 图四

6.4 Flow Control

- Avoid loss at receiver by telling sender the available buffer space
  - WIN=#Acceptable, not W (from LAS)
- See Figure 39

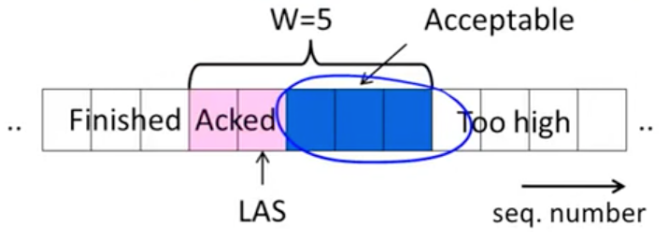


图 39: Flow Control 图一

- Sender uses the lower of the sliding window and flow control window (WIN) as the effective window size
- See Figure 40

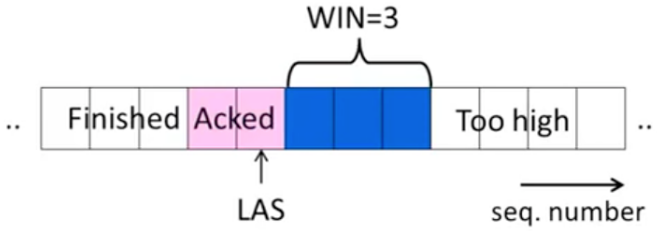


图 40: Flow Control 图二

- TCP-style example
  - SEQ/ACK sliding window
  - Flow control with WIN

- SEQ + length < ACK + WIN
- 4KB buffer at receiver
- Circular buffer of bytes

- See Figure 41

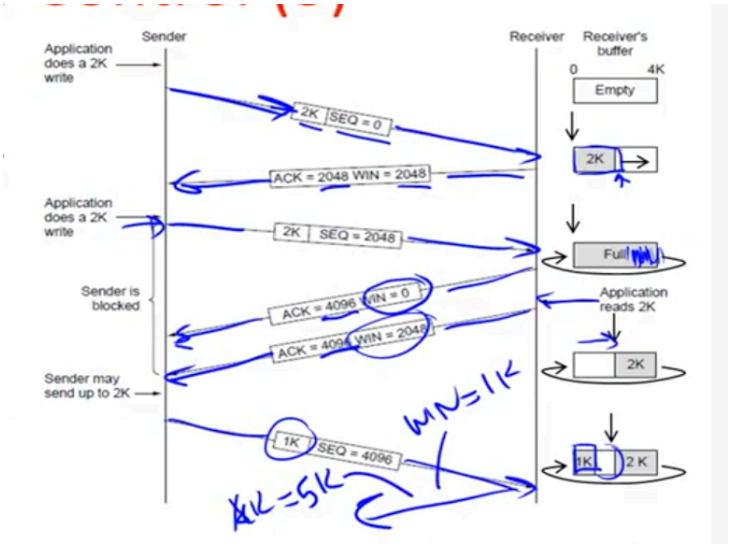


图 41: Flow Control 图三

7 Week 6.7 - Retransmission Timeouts

7.1 Topic of Retransmission Timeouts

- How to set the timeout for sending a retransmission
  - Adapting to the network path

- See Figure 42

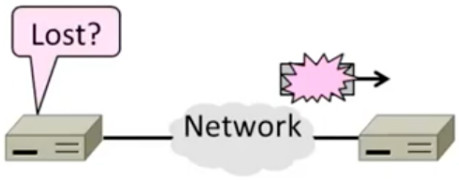


图 42: Topic of Retransmission Timeouts

7.2 Retransmissions

- With sliding window, the strategy for detecting loss is the timeout
  - Set timer when a segment is sent
  - Cancel timer when ack is received
  - If timer fires, retransmit data as lost
- See Figure 43

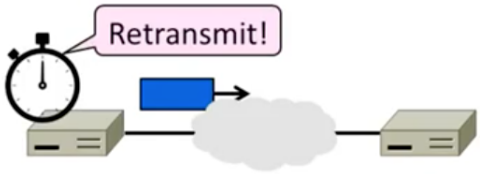


图 43: Retransmissions

7.3 Timeout Problem

- Timeout should be ”just right”
  - Too long wastes network capacity
  - Too short leads to spurious(伪) resends
- Easy to set on a LAN (Link)

  - Short, fixed, predictable RTT

- Hard on the Internet (Transport)
  - Wide range, variable RTT

7.4 Example of RTTs

- 下面这幅图描述了从 Barcelona 到海外再回来这个回路的 RTTs
- See Figure 44

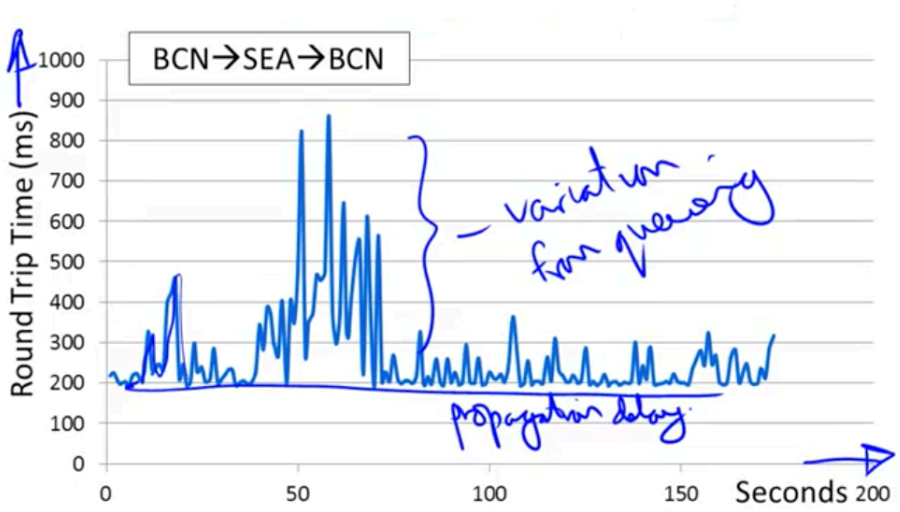


图 44: Example of RTTs 图一

- 下面进一步的说明
- See Figure 45

7.5 Adaptive Timeout

- Keep smoothed estimates of the RTT(1) and variance in RTT (2)

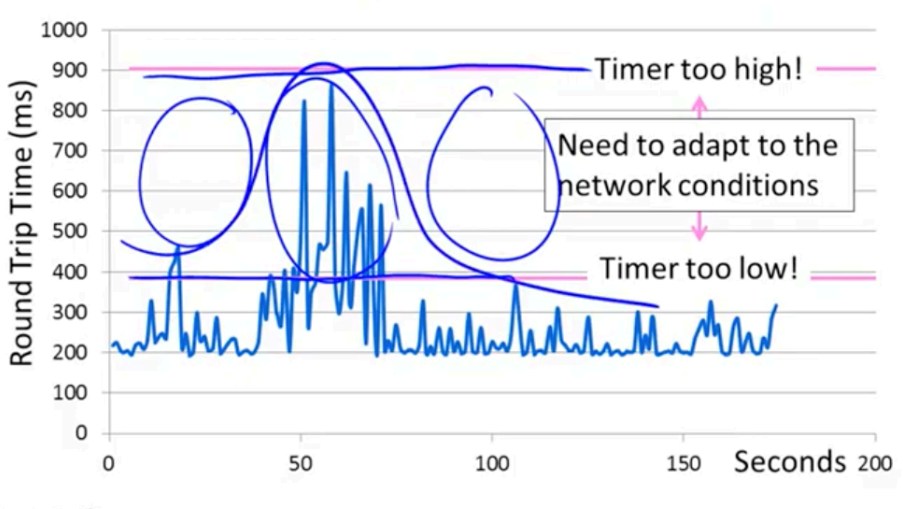


图 45: Example of RTTs 图二

  - Update estimates with a moving average
  - 1. $SRTT_{N+1} = 0.9 * SRTT_N + 0.1 * RTT_{N+1}$
  - 2. $Svar_{N+1} = 0.9 * Svar_N + 0.1 * |RTT_{N+1} - SRTT_{N+1}|$

- Set timeout to a multiple of estimates
  - To estimate the upper RTT in practice
  - TCP $Timeout_N = SRTT_N + 4 * Svar_N$
- Simple to compute, does a good job of tracking actual RTT
  - Little "headroom"(上升空间) to lower
  - Yet very few early timeouts
- Turns out to be important for good performance and robustness

7.6 Example of Adaptive Timeout

- 下图符合公式 $Timeout_N = SRTT_N + 4 * Svar_N$

- See Figure 46

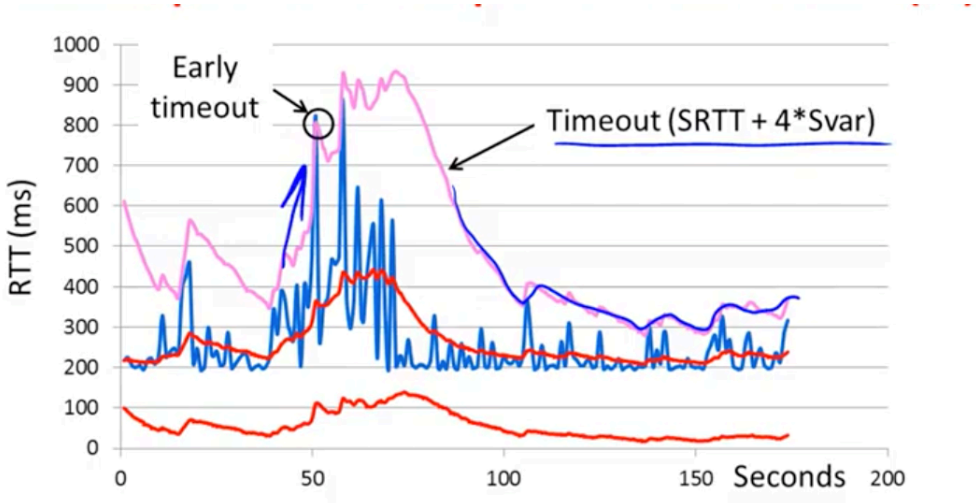


图 46: Example of Adaptive Timeout

8 Week 6.8 - Transmission Control Protoco TCP

8.1 Topic of TCP

- How TCP works!
  - The transport protocol used for most content on the Internet

8.2 TCP Features

- 本节课内容:
- A reliable bytestream service
- Sliding window for reliability

  - With adaptive timeout

- Flow control for slow receivers
- 下节课内容：
- Congestion control to allocate network bandwidth

8.3 Reliable Bytestream

- Message boundaries(边界) not preserved from send() to recv()
  - But reliable and ordered (receive bytes in same order as sent)
- See Figure 47

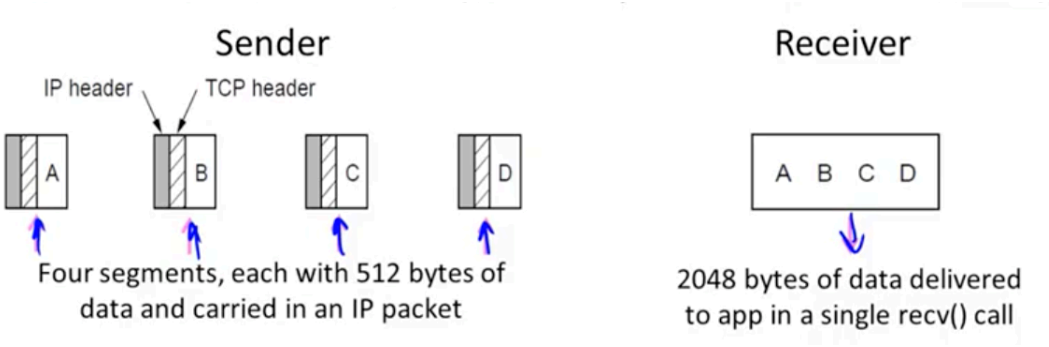


图 47: Reliable Bytestream 图一

- Bidirectional data transfer
  - Control information (e.g., ACK) piggybacks(背负着) on data segments in reverse direction
  - 发包并且把上个包的 ACK 连带着发送过来。
- See Figure 48

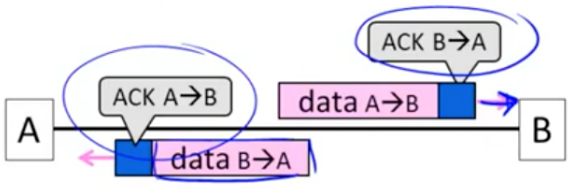


图 48: Reliable Bytestream 图二

8.4 TCP Header

- Ports identify apps (socket API)
  - 16-bit identifiers
- SEQ/ACK used for sliding window
  - Selective Repeat, with byte positions
- See Figure 49

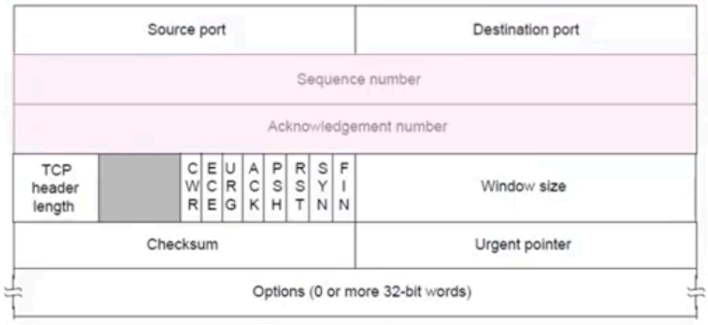


图 49: TCP Header

- SYN/FIN/RST flags for connections
  - Flag indicates segment is a SYN etc.

- Window size for flow control
  - Relative to ACK, and in bytes

8.5 TCP Sliding Window - Receiver

- Cumulative ACK tells next expected byte sequence number ("LAS+1")
- Optionally, selective ACKs (SACK) give hints for receiver buffer state
  - List up to 3 ranges of received bytes
- See Figure 50

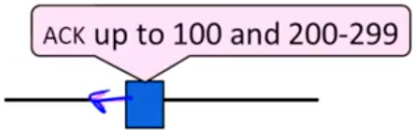


图 50: TCP Sliding Window - Receiver

8.6 TCP Sliding Window - Sender

- Uses an adaptive retransmission timeout to resent data from LAS+1
- Use heuristics(启发式) to infer loss quickly and resend to avoid time-outs
  - "Three duplicate ACKs" treated as loss
- See Figure 51

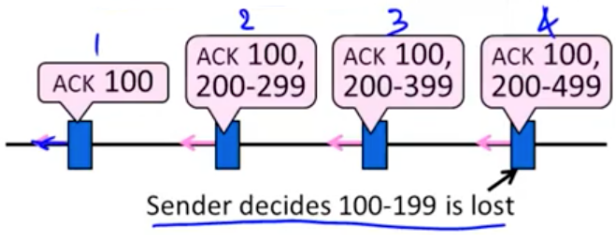


图 51: TCP Sliding Window - Sender

8.7 Other TCP Details

- Many, many quirks(怪癖) you can learn about its operation
  - But they are the details
- Biggest remaining(剩余的) mystery is the workings of congestion(拥塞) control
  - We'll tackle this next time!

9 Week 7.1 - Congestion Overview

9.1 Where we are in the Course

- More fun in the Transport Layer!
  - The mystery of congestion control
  - Depends on the Network layer too

9.2 Topic of Congestion Overview

- Understanding congestion, a "traffic jam" in the network
  - Later we will learn how to control it

9.3 Nature of Congestion

- Routers/switches have internal buffering for contention(争夺)
- See Figure 52

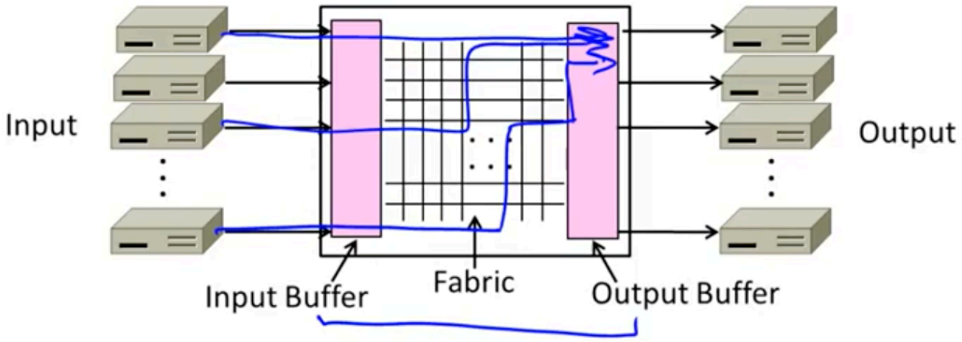


图 52: Nature of Congestion 图一

- Simplified view of per port output queues
  - Typically FIFO (First In First Out), discard when full
- See Figure 53

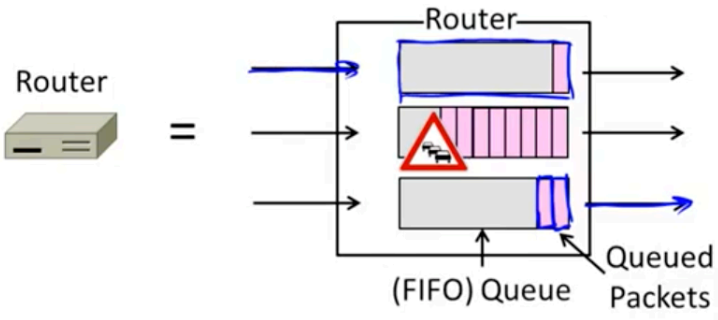


图 53: Nature of Congestion 图二

- Queues help by absorbing bursts(冲垮) when input > output rate

- But if input > output rate persistently, queue will overflow
  - This is congestion
- Congestion is a function of the traffic patterns - can occur even if every link have the same capacity

9.4 Effects of Congestion

- What happens to performance as we increase the load?
- See Figure 54

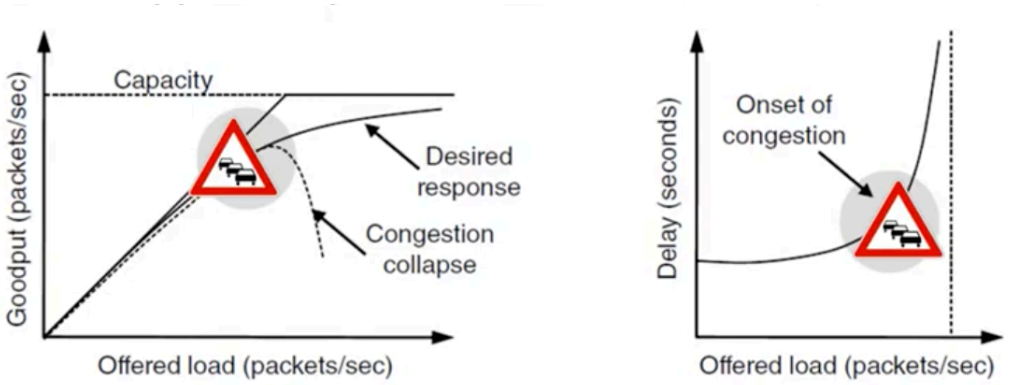


图 54: Effects of Congestion

- As offered load rises, congestion occurs as queues begin to fill:
  - Delay and loss rise sharply with more load
  - Throughput(吞吐量) falls below load (due to loss)
  - Goodput(有效吞吐量) may fall below throughout (due to spurious(伪) retransmissions)
- Nore of the above is good!
  - Want to operate network just before the onset of congestion

9.5 Bandwidth Allocation

- Important task for network is to allocate its capacity to senders
  - Good allocation is efficient and fair
- Efficient means most capacity is used but there is no congestion
- Fair means every sender gets a reasonable share the network
- Key observation:
  - In an effective solution, Transport and Network layers must work together
- Network layer witnesses(目睹)congestion
  - Onlu it can provide direct feedback
- Transport layer causes congestion
  - Only it can reduce offered load
- Why is it hard? (Just split equally!)
  - Number of senders and their offered load is constantly changing
  - Senders may lack capacity in different parts of the network
  - Network is distributed; on single party has an overall picture of its state
- Solution context:
  - Senders adapt concurrently based on their own view of the network
  - Design this adaption so the network usage as a whole is efficient and fair

  - Adaption is continuous since offered loads continue to change over time

9.6 Topics

- 本节课内容：
- Nature of congestion
- 下节课内容：
- Fair allocations
- AIMD control law
- TCP Congestion Control history
- ACK clocking
- TCP Slow-start
- TCP Fast Retransmit/Recovery
- Congestion Avoidance (ECN)

10 Week 7.2 - Fairness of Allocations

10.1 Topic of Fairness of Allocations

- What's a "fair" bandwidth allocation?
  - The max-min fair allocation

10.2 Recall

- We want a good bandwidth allocation to be fair and efficient
  - Now we learn what fair means
- Caveat(警告): in practice, efficiency is more important than fairness

10.3 Efficiency vs. Fairness

- Cannot always have both!
  - Example network with traffic $A \to B$, $B \to C$ and $A \to C$
  - How much traffic can we carry?
- See Figure 55

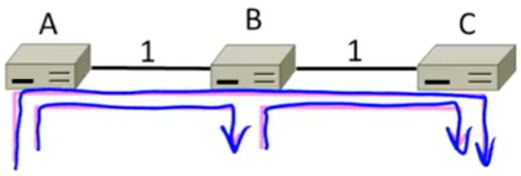


图 55: Efficiency vs. Fairness 图一

- If we care about fairness:
  - Give equal bandwidth to each flow
  - $A \to B$: $\frac{1}{2}$ unit, $B \to C$: $\frac{1}{2}$, and $A \to C$, $\frac{1}{2}$
  - Total traffic carried is $1\frac{1}{2}$ units
- See Figure 56
- If we care about efficiency:

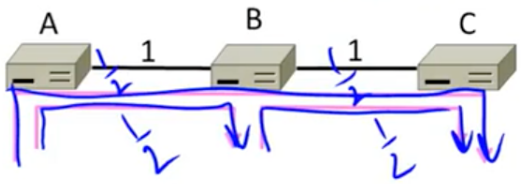


图 56: Efficiency vs. Fairness 图二

- Maximize total traffic in network
- $A \to B$: 1 unit, $B \to C$: 1, and $A \to C$, 0
- Total traffic rises to 2 units!

- See Figure 57

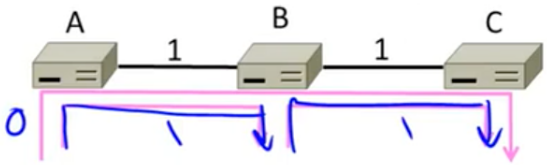


图 57: Efficiency vs. Fairness 图三

10.4 The Slippery(光溜溜) Notion of Fairness

- Why is "equal per flow" fair anyway?
  - $A \to C$ uses more network resources (two links) then $A \to B$ or $B \to C$
  - Host A sends two flows, B sends one
- Not productive to seek exact fairness
  - More important to avoid starvation
  - "Equal per flow" is good enough

10.5 Generalizing "Equal per Flow"

- Bottleneck for a flow of traffic is the link that limits its bandwidth
  - Where congestion occurs for the flow
  - For $A \rightarrow C$, link A-B is the bottleneck
- See Figure 58

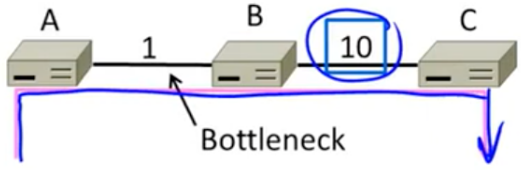


图 58: Generalizing "Equal per Flow" 图一

- Flows may have different bottlenecks
  - For $A \rightarrow B$ link A-B is the bottleneck
  - For $B \rightarrow C$ link B-C is the bottleneck, 可能后面网络中有比它更大的容量的线路，此时它就成了瓶颈。
  - Can no longer divide links equally ...
- See Figure 59

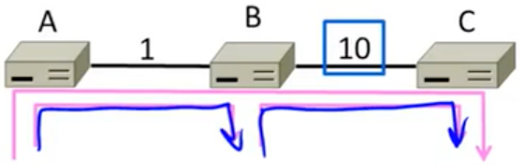


图 59: Generalizing "Equal per Flow" 图二

10.6 Max-Min Fairness

- Intuitively(直观地), flows bottlenecked on a link get an equal share of that link
- Max-min fair allocation is one that:
  - Increasing the rate of one flow will decrease the rate of a smaller flow
  - This "maximizes the minimum" flow
- To find it given a network, imagine "pouring(注入) water into the network"
  - 1. Start with all flows at rate 0
  - 2. Increase the flows until there is a new bottleneck in the network
  - 3. Hold fixed the rate of the flows that are bottlenecked
  - 4. Go to step 2 for any remaining flows

10.7 Max-Min Example

- Example: network with 4 flows, links equal bandwidth
  - What is the max-min fair allocation?
- See Figure 60
- When rate=1/3, flows B,C, and D bottleneck R4-R5
  - Fix B, C, and D, continue to increase A
- See Figure 61
- When $rate = \frac{2}{3}$, flow A bottlenecks R2-R3. Done.

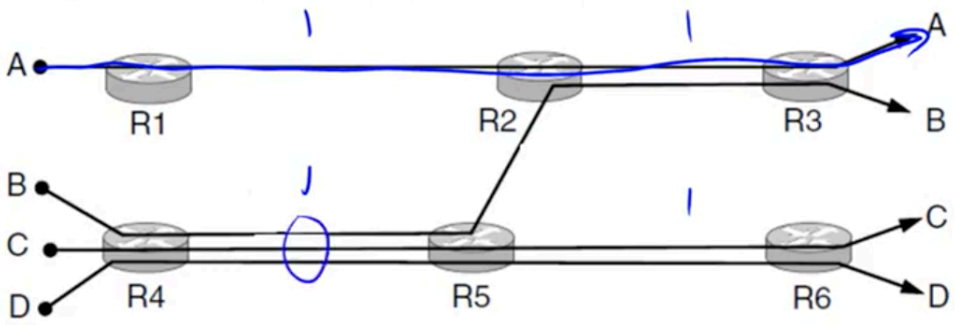


图 60: Max-Min Example 图一

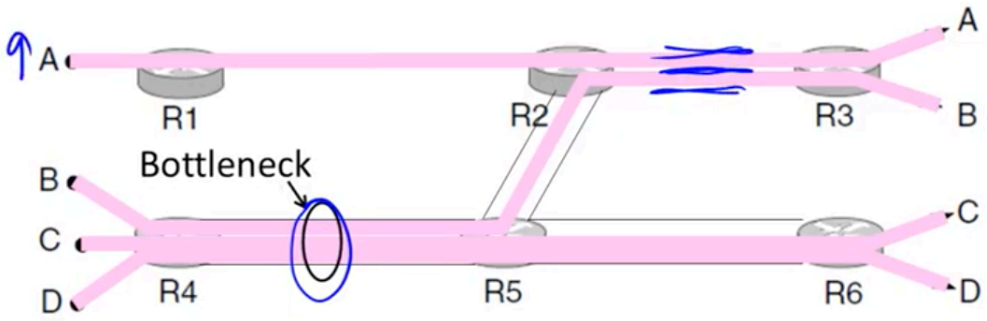


图 61: Max-Min Example 图二

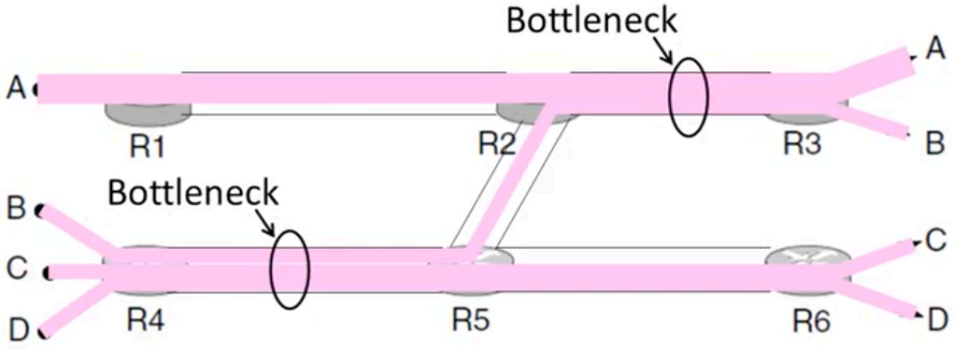


图 62: Max-Min Example 图三

- See Figure 62
- End with A=$\frac{2}{3}$, B, C, D=$\frac{1}{3}$, and R2-R3, R4-R5 full
  - Other links have extra capacity that can't be used
- See Figure 63

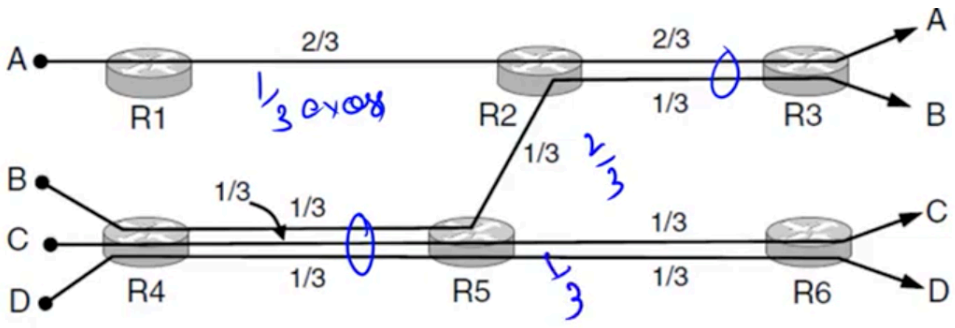


图 63: Max-Min Example 图四

10.8 Adapting over Time

- Allocation changes as flows start and stop
- See Figure 64

11 Week 7.3 - Additive Increase Multiplicative Decrease (AIMD)

11.1 Topic of AIMD

- Bandwidth allocation models
  - Additive Increase Multiplicative Decrease (AIMD) control law
- See Figure 65

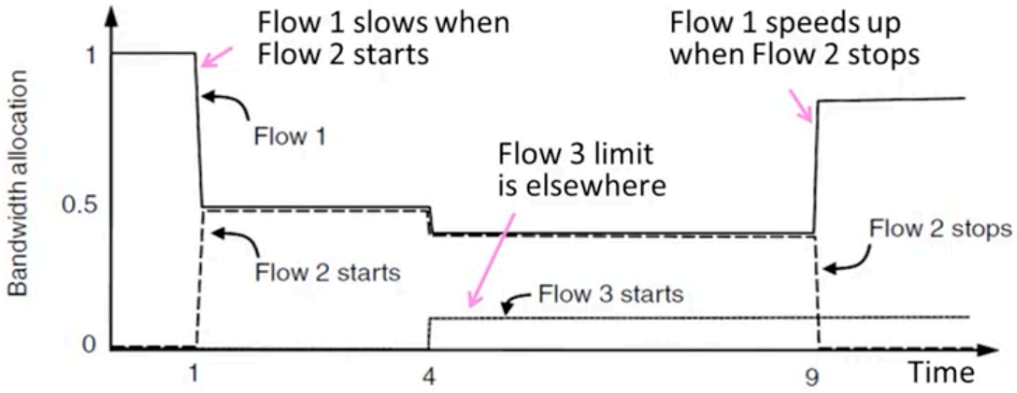


图 64: Adapting over Time

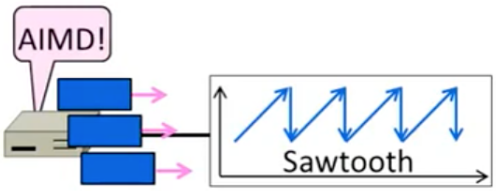


图 65: Topic of AIMD

11.2 Recall

- Want to allocate capacity to senders
  - Network layer provides feedback
  - Transport layer adjusts offered load
  - A good allocation is efficient and fair
- How should we perform the allocation?
  - Several different possibilities ...

11.3 Bandwidth Allocation Models

- Open loop versus closed loop
  - Open: reverse bandwidth before use
  - Closed: use feedback to adjust rates
- Host versus Network support
  - Who is sets/enforces allocations?
- Window versus Rate based
  - How is allocation expressed?
- TCP is a closed loop, host-driven, and window-based
- We'll look at closed-loop, host-driven, and window-based too
- Network layer returns feedback on current allocation to senders
  - At least tells if there is congestion
- Transport layer adjusts sender's behavior via window in response
  - How senders adapt is a control law

11.4 Additive Increase multiplicative Decrease

- AIMD is a control low hosts can use to reach a good allocation
  - Hosts additively increase rate while network is not congested
  - Hosts multiplicatively decrease rate when congestion occurs
  - Used by TCP
- Let's explore the AIMD game ...

11.5 AIMD Game

- Hosts 1 and 2 share a bottleneck
  - But do not talk to each other directly
- Router provides binary feedback
  - Tells hosts if network is congested
- See Figure 66

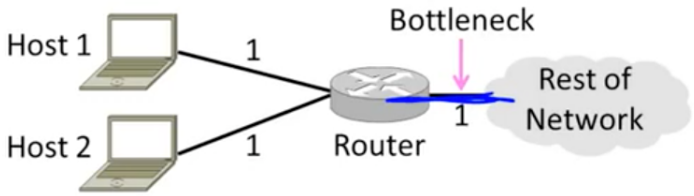


图 66: AIMD Game 图一

- Each point is a possible allocation
- AI and MD move the allocation
- See Figure 67
- Play the game !
- Always converge(汇集) to good allocation !
- See Figure 68

11.6 AIMD Sawtooth

- Produces a ”sawtooth” pattern over time for rate of each host

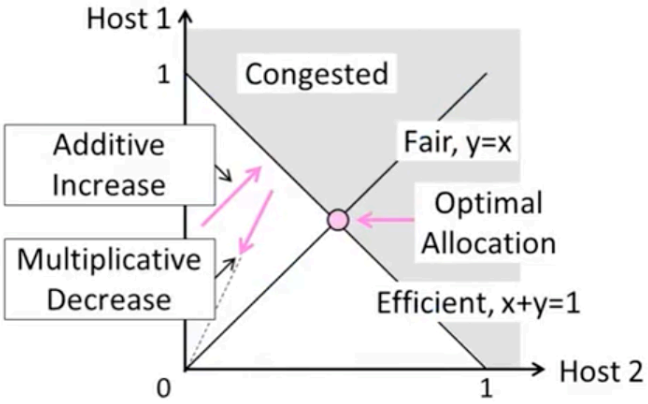


图 67: AIMD Game 图二

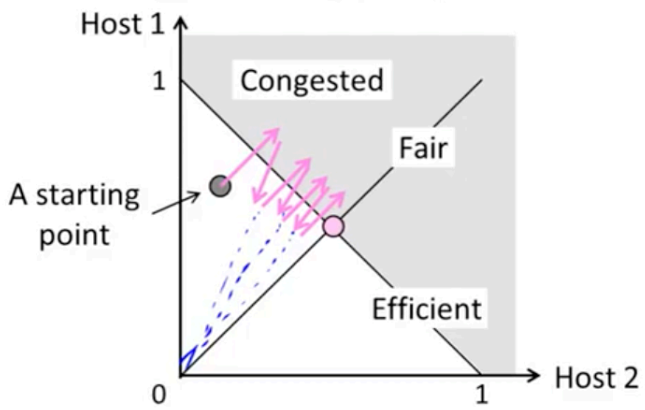


图 68: AIMD Game 图三

  - This is the TCP sawtooth (later)

- See Figure 69

11.7 AIMD Properties

- Converges to an allocation that is efficient and fair when hosts run it
  - Holds for more general topologies

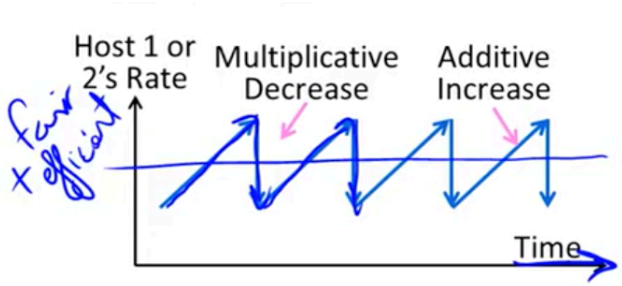


图 69: AIMD Sawtooth

- Other increase/decrease control laws do not! (Try MIAD, MIMD, AIAD)
- Requires only binary feedback from the network

11.8 Feedback Signals

- Several possible signals, with different pros/cons
  - We'll look at classic TCP that uses packet loss as a signal
- See Figure 70

Signal	Example Protocol	Pros / Cons
Packet loss	TCP NewReno Cubic TCP (Linux)	Hard to get wrong Hear about congestion late
Packet delay	Compound TCP (Windows)	Hear about congestion early Need to infer congestion
Router indication	TCPs with Explicit Congestion Notification	Hear about congestion early Require router support

图 70: Feedback Signals

12 Week 7.4 - History of TCP Congestion Control

12.1 Topic of History of TCP Congestion Control

- The story of TCP congestion control
  - Collapse, control, and diversification(多样化)

12.2 Congestion Collapse in the 1980s

- Early TCP used a fixed size sliding window (e.g., 8 packets)
  - Initially fine for reliability
- But something strange happened as the ARPANET grew
  - Links stayed busy but transfer rates fell by orders of magnitude(量纲)!
- Queues became full, retransmissions clogged(堵塞) the network, and goodput(有效吞吐量) fell
- See Figure 71

12.3 Van Jacobson (1950-)

- Widely credited with saving the Internet from congestion collapse in the late 80s
  - Introduced congestion control principles
  - Practical solutions (TCP Tahoe/Reno)
- Much other pioneering(开拓) work:

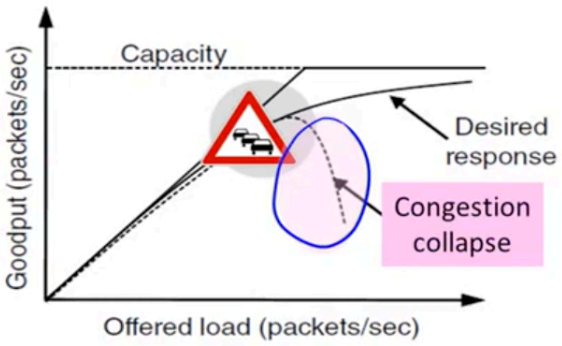


图 71: Congestion Collapse in the 1980s

- Tools like traceroute, tcpdump, pathchar
- IP header compression(压缩), multicast tools

12.4 TCP Tahoe/Reno

- Avoid congestion collapse without changing routers (or even receivers)
- Idea is to fix timeouts and introduce a congestion window (cwnd) over the sliding window to limit queues/loss
- TCP Tahoe/Reno implements AIMD by adapting cwnd using packet loss as the network feedback signal
- TCP behaviors we will study:
  - ACK clocking
  - Adaptive timeout (mean and variance)
  - Slow-start
  - Fast Retransmission
  - Fast Recovery
- Together, they implement AIMD

12.5 TCP Timeline

- See Figure 72

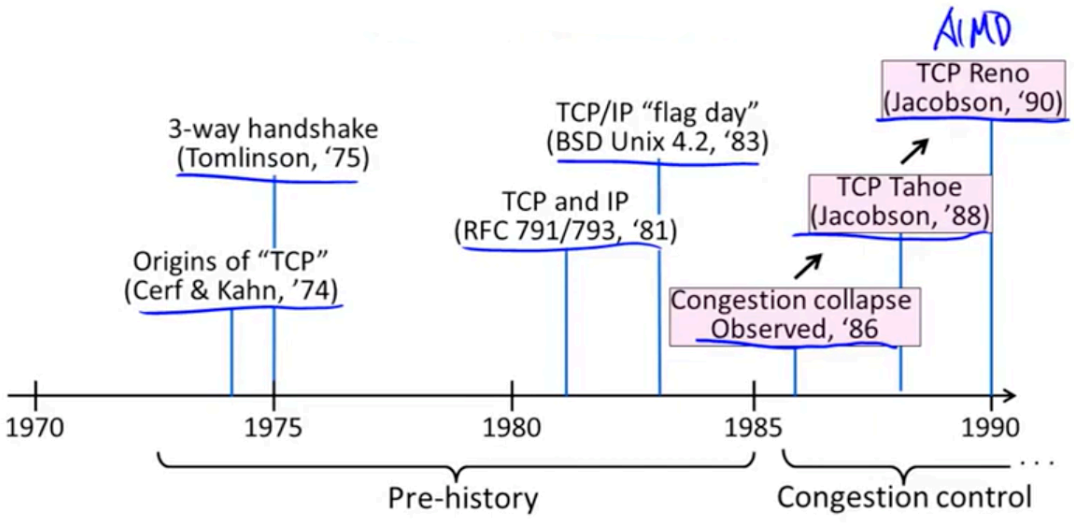


图 72: TCP Timeline 图一

- See Figure 73

13 Week 7.5 - TCP ACK Clocking

13.1 Topic of TCP ACK Clocking

- The self-clocking behavior of sliding windows, and how it is used by TCP
  - The "ACK clock"
- See Figure 74

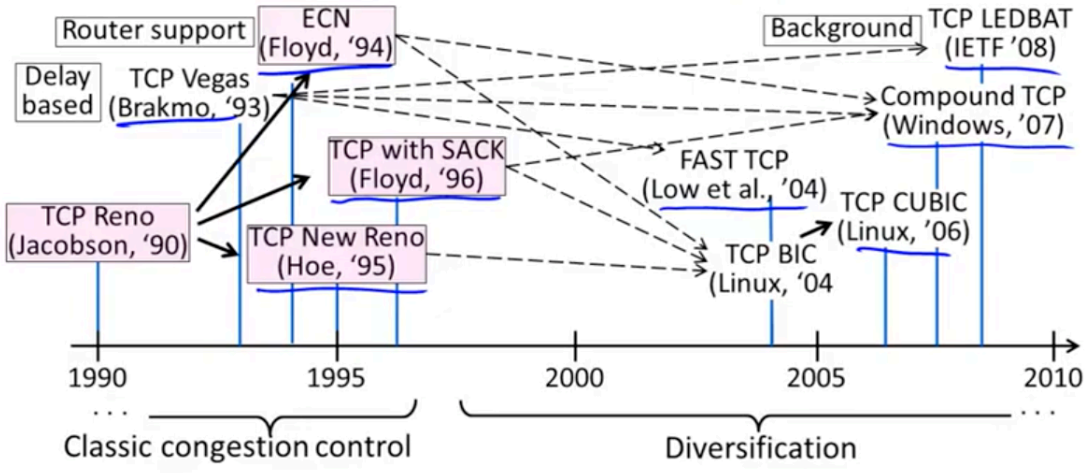


图 73: TCP Timeline 图二

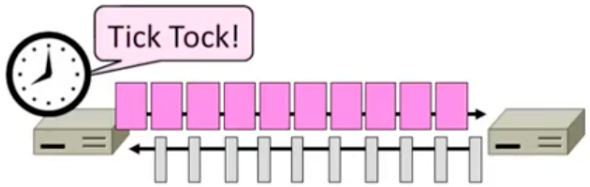


图 74: Topic of TCP ACK Clocking

13.2 Sliding Window ACK Clock

- Each in-order ACK advances(促进) the sliding window and lets a new segment enter the network
  - ACKs "clock" data segments
- See Figure 75

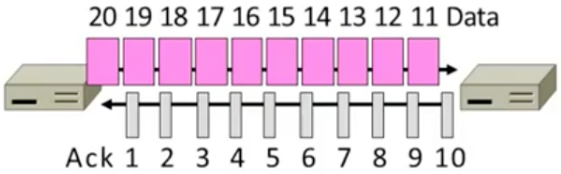


图 75: Sliding Window ACK Clock

13.3 Benefit of ACK Clocking

- Consider what happens when sender injects a burst of segments into the network
- See Figure 76

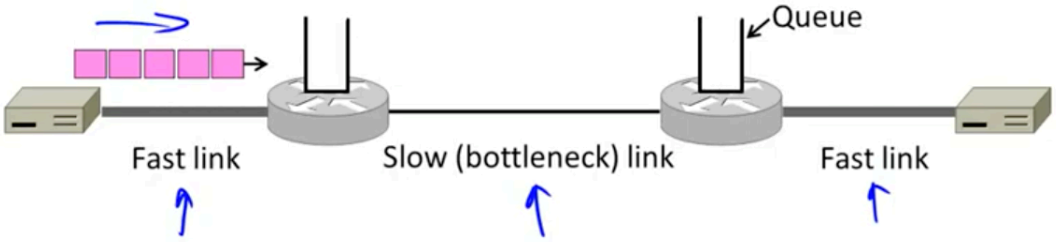


图 76: Benefit of ACK Clocking

- Segments are buffered and spread out on slow link
- See Figure 77
- ACKs maintain the spread back to the original sender
- See Figure 78
- Sender clocks new segments with the spread
  - Now sending at the bottleneck link without queuing!

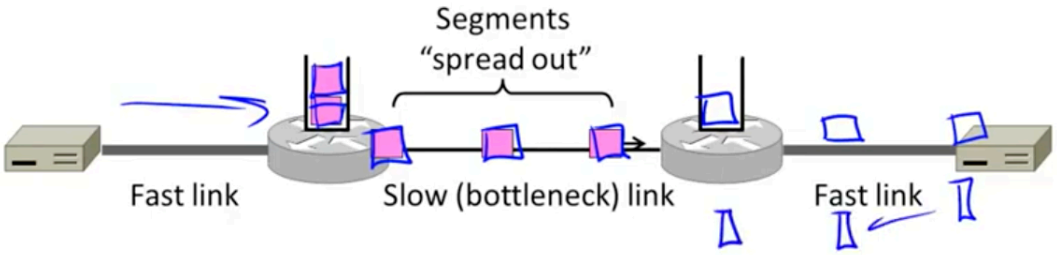


图 77: Benefit of ACK Clocking

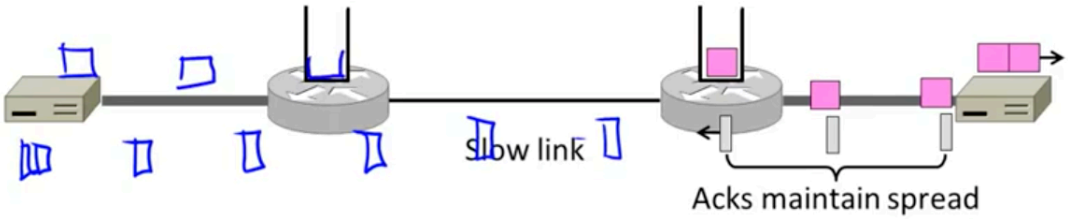


图 78: Benefit of ACK Clocking

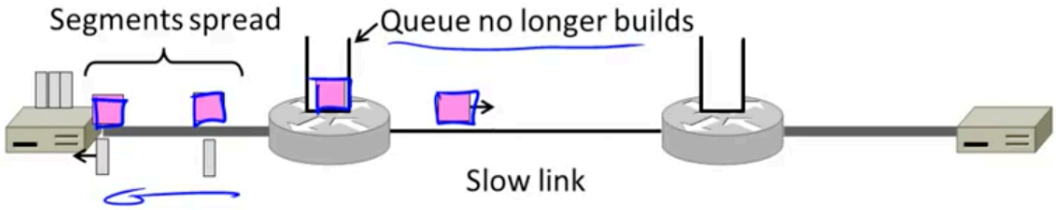


图 79: Benefit of ACK Clocking

- See Figure 79
- Helps the network run with low levels of loss and delay
- The network has smoothed out the burst of data segments

- ACK clock transfers this smooth timing back to the sender
- Subsequent data segments are not sent in bursts so do not queue up in the network

13.4 TCP Uses ACK Clocking

- TCP uses a sliding window because of the value of ACK clocking
- Sliding window controls how many segments are inside the network
  - Called the congestion window, or cwnd
  - Rate is roughly cwnd/RTT
- TCP only sends small bursts of segments to let the network keep the traffic smooth

14 Week 7.6 - TCP Slow Start

14.1 Topic of TCP Slow Start

- How TCP implements AIMD, part 1
  - ”Slow start” is a component of the AI portion of AIMD
- See Figure 80

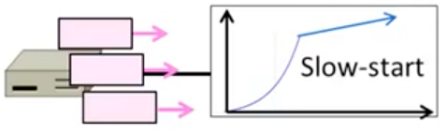


图 80: Topic of TCP Slow Start

14.2 Recall

- We want TCP to follow an AIMD control low for a good allocation
- Sender uses a congestion window or cwnd to set its rate ($\approx cwnd/RTT$)
- Sender uses packet loss as the network congestion signal
- Need TCP to work across a very large range of rates and RTTs

14.3 TCP Startup Problem

- We want to quickly near the right rate, $cwnd_{IDEAL}$, but it varies greatly
  - Fixed sliding window doesn't adapt and is rough on the network (loss!)
  - AI with small bursts adapts cwnd gently to the network, but might take a long time to become efficient

14.4 Slow-Start Solution

- Start by doubling cwnd every RTT
  - Exponential grownth (1, 2, 4, 8, 16, ...)
  - Start slow, quickly reach large values
- See Figure 81
- Eventually packet loss will occur when the network is congested
  - Loss timeout tells us cwnd is too large
  - Next time, switch to AI beforehand(预先)
  - Slowly adapt cwnd near right value

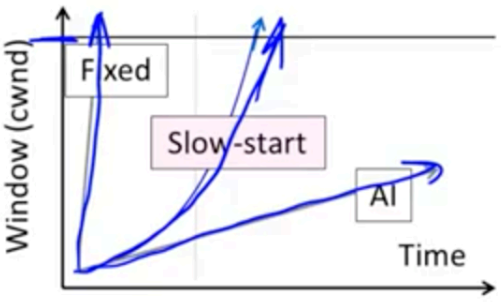


图 81: Slow-Start Solution 图一

- In terms of cwnd:
  - Expect loss for $cwnd_C \approx 2BD + queue$
  - Use $ssthresh = cwnd_C/2$ to switch to AI
- Combined behavior, after first time
  - Most time spend near right value
- See Figure 82

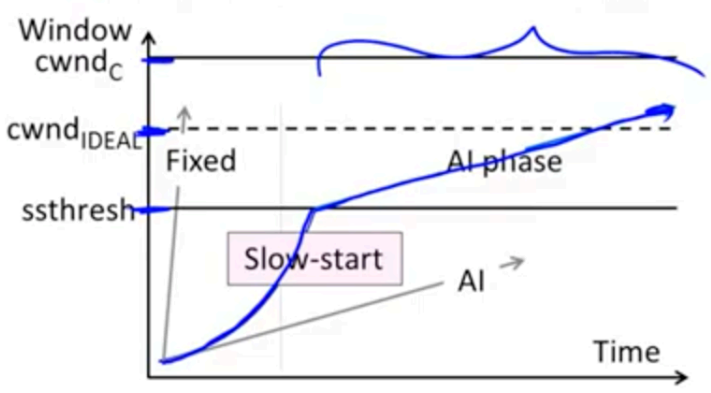


图 82: Slow-Start Solution 图二

14.5 Slow-Start (Doubling) Timeline

- 左侧的 cwnd 是一个一个地增加的
- See Figure 83

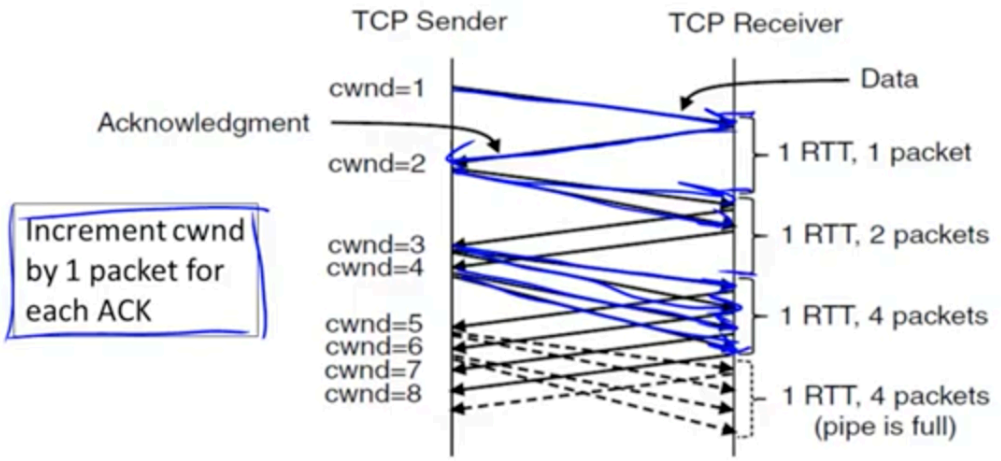


图 83: Slow-Start (Doubling) Timeline

14.6 Additive Increase Timeline

- 和上面那副图一样，左边的 cwnd 是一个一个地增加，这里增加的速度比上面慢多了。
- See Figure 84

14.7 TCP Tahoe (Implementation)

- Initial slow-start (doubling) phase
  - Start with $cwnd = 1$ (or small value)
  - $cwnd+ = 1$ packet per ACK

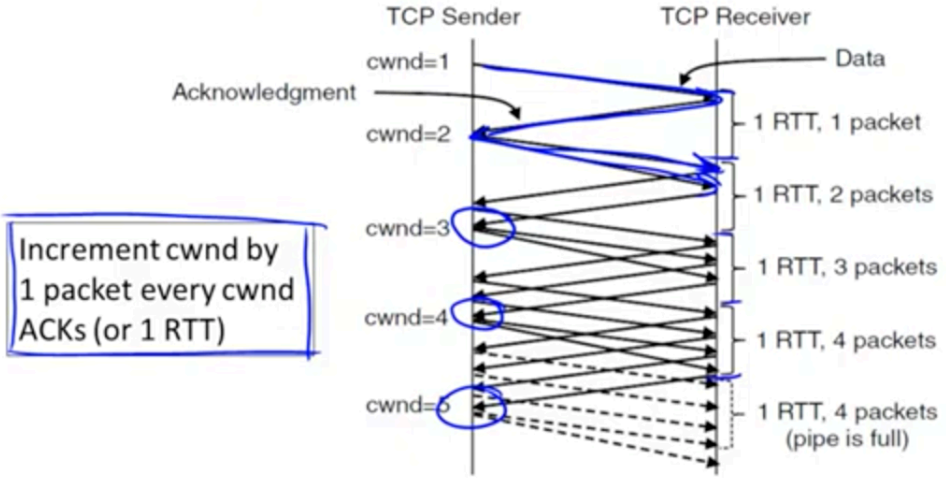


图 84: Additive Increase Timeline

- Later Additive Increase phase
  - $cwnd+ = 1/cwnd$ packets per ACK
  - Roughly adds 1 packet per RTT
- Switching threshold (initially infinity)
  - Switch to AI when $cwnd > ssthread$
  - Set $ssthresh = cwnd/2$ after loss
  - Begin with slow-start after timeout

14.8 Timeout Misfortunes(不幸)

- Why do a slow-start after timeout?
  - Instead of MD cwnd (for AIMD)
- Timeouts are sufficiently long that the ACK clock will have run down(撞倒)

  - Slow-start ramps(上坡) up the ACK clock

- We need to detect loss before a timeout to get to full AIMD
  - Done in TCP Reno (next time)

15 Week 7.7 - TCP Fast Retransmit : Fast Recovery

15.1 Topic of TCP Fast Retransmit : Fast Recovery

- How TCP implements AIMD, part 2
  - ”Fast retransmit” and ”fast recovery” are the MD portion of AIMD
- See Figure 85

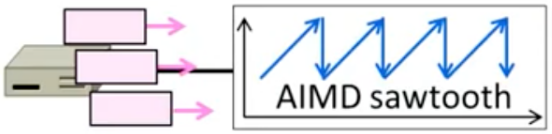


图 85: Topic of TCP Fast Retransmit : Fast Recovery

15.2 Recall

- We want TCP to follow an AIMD control law for a good allocation
- Sender uses a congestion window or cwnd to set its rate ($\approx cwnd/RTT$)
- Sender uses slow-start to ramp up(斜坡上升) the ACK clock, followed by Additive Increase

- But after a timeout, sender slow-starts again with cwnd=1 (as it no ACK clock)

15.3 Inferring(推斷) Loss from ACKs

- TCP uses a cumulative ACK
  - Carries highest in-order seq. number
  - Normally a steady advance
- Duplicate ACKs give us hints about what data hasn't arrived
  - Tell us some new data did arrive, but it was not next segment
  - Thus the next segment may be lost

15.4 Fast Retransmit

- Treat three duplicate ACKs as a loss
  - Retransmit next expected segment
  - Some repetition allows for reordering, but still detects loss quickly
- See Figure 86

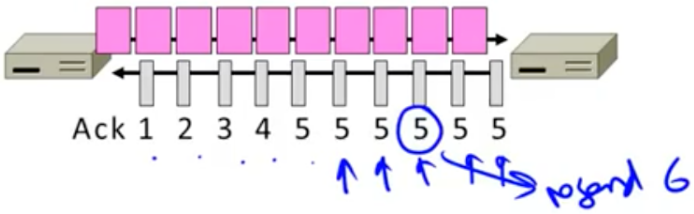


图 86: Fast Retransmit 图一

- See Figure 87

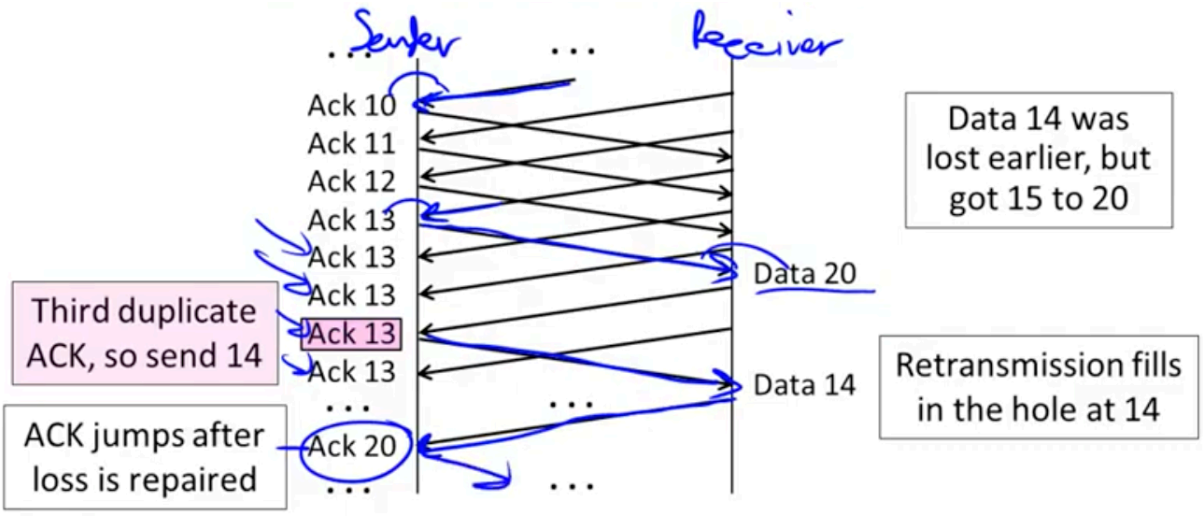


图 87: Fast Retransmit 图二

- It can repair single segment loss quickly, typically before a timeout
- However, we have quiet time at the sender/receiver while waiting for the ACK to jump
- And we still need to MD cwnd ...

15.5 Inferring(推斷) Non-Loss from ACKs

- Duplicate ACKs also give us hints about what data has arrived
  - Each new duplicate ACK means that some new segment has arrived
  - It will be the segments after the loss
  - Thus advancing the sliding window will not increase the number of segments stored in the network

15.6 Fast Recovery

- First fast retransmit, and MD cwnd
- Then pretend further duplicate ACKs are the expected ACKs
  - Lets new segments be sent for ACKs
  - Reconcile(調和) views when the ACK jumps
- See Figure 88

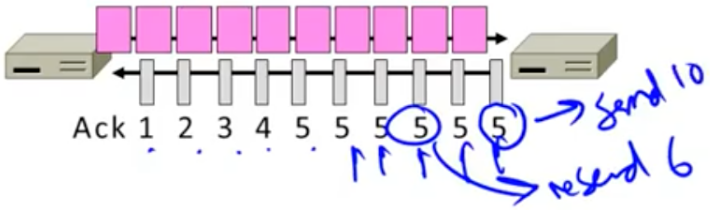


图 88: Fast Recovery 图一

- 如下图所示，把最后的 Ack 13 当成 Ack 20 可以让 Sender 和 Receiver 不用等待，持续发送与接收包。
- See Figure 89
- With fast retransmit, it repairs a single segment loss quickly and keeps the ACK clock running
- This allows us to realize AI<D
  - No timeouts or slow-start after loss, just continue with a smaller cwnd
- TCP Reno combines slow-start, fast retransmit and fast recovery
  - Multiplicative Decrease is $\frac{1}{2}$

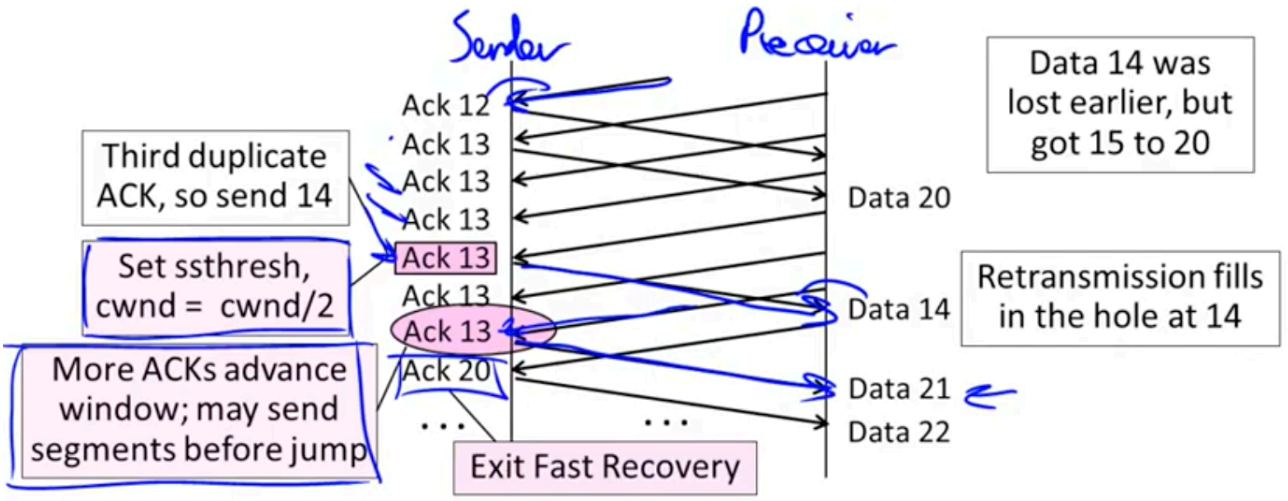


图 89: Fast Recovery 图二

- See Figure 90

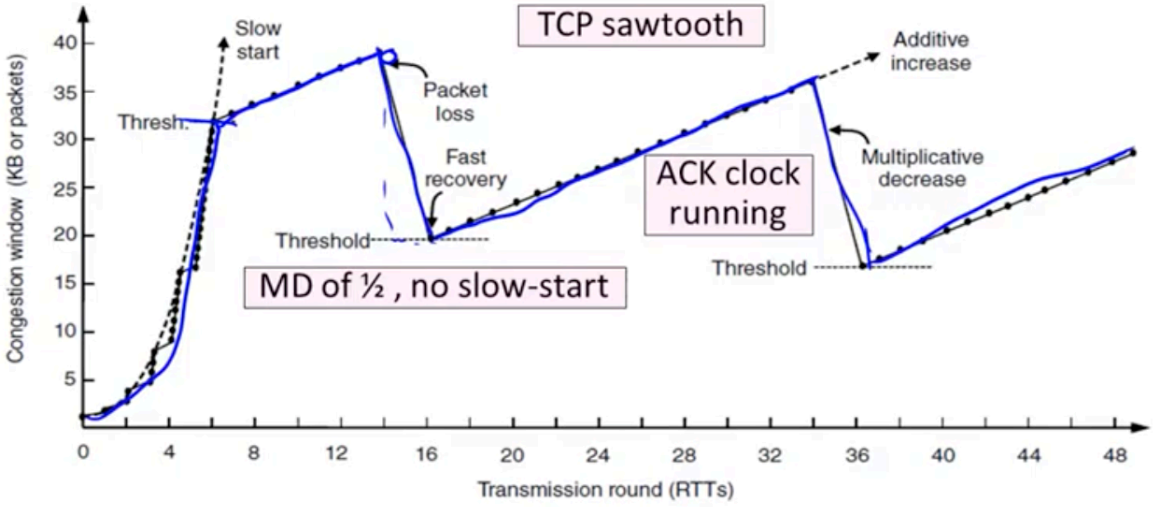


图 90: Fast Recovery 图三

15.7 TCP Reno, NewReno, and SACK

- Reno can repair one loss per RTT
  - Multiple losses cause a timeout
- NewReno further refines(提煉) ACK heuristics(試探)
  - Repairs multiple losses without timeout
- SACK is a better idea
  - Receiver sends ACK ranges so sender can retransmit without guesswork

16 Week 7.8 - Explicit Congestion Notification (ECN)

16.1 Topic of ECN

- How routers can help hosts to avoid congestion
  - Explicit Congestion Notification
- See Figure 91

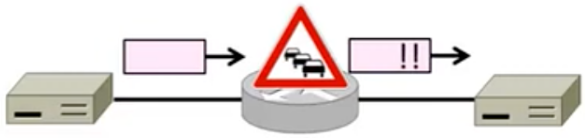


图 91: Topic of ECN

16.2 Congestion Avoidance vs. Control

- Classic TCP drives the network into congestion and then recovers
  - Needs to see to slow down
- Would be better to use the network but avoid congestion altogether(完全) !
  - Reduces loss and delay
- But how can we do this ?

16.3 Feedback Signals

- Delay and router signals can let us avoid congestion
- See Figure 92

Signal	Example Protocol	Pros / Cons
Packet loss	Classic TCP Cubic TCP (Linux)	Hard to get wrong Hear about congestion late
Packet delay	Compound TCP (Windows)	Hear about congestion early Need to infer congestion
Router indication	TCPs with Explicit Congestion Notification	Hear about congestion early Require router support

图 92: Feedback Signals

16.4 ECN (Explicit Congestion Notification)

- Router detects the onset(發作) of congestion via its queue

  - When congested, it marks affected packets (IP header)

- See Figure 93

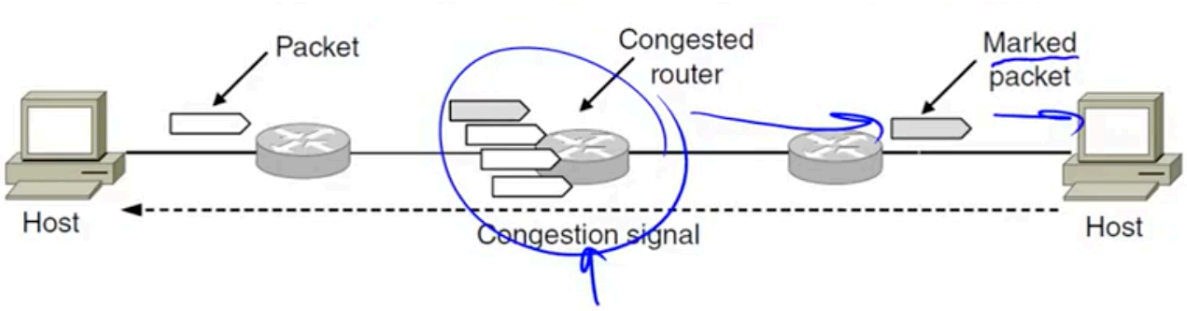


图 93: ECN (Explicit Congestion Notification)

- Marked packets arrive at receiver; treated as loss
  - TCP receiver reliably informs(告知) TCP sender of the congestion
- Advantages:
  - Routers deliver clear signal to hosts
  - Congestion is detected early, no loss
  - No extra packets need to be sent
- Disadvantages:
  - Routers and hosts must be upgraded

17 Week 8.1 - Application Layer Overview

17.1 Where we are in the Course

- Starting the Application Layer!

- Builds distributed ”network services” (DNS, Web) on Transport services

17.2 Recall

- Application layer protocols are often part of an ”app”
  - But don’t need a GUI, e.g., DNS
- See Figure 94

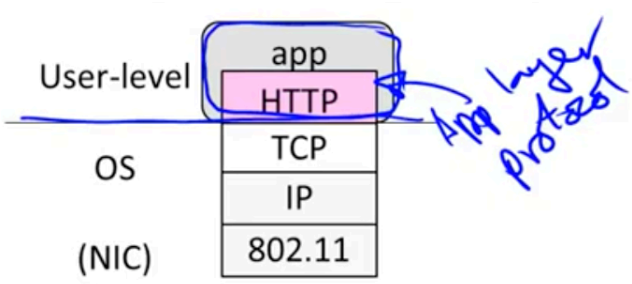


图 94: Recall 图一

- Application layer messages are often split over multiple packets
  - Or may be aggregated(集成體) in packet ...
- See Figure 95

17.3 Application Communication Needs

- Vary widely with app; must build on Transport services
- See Figure 96

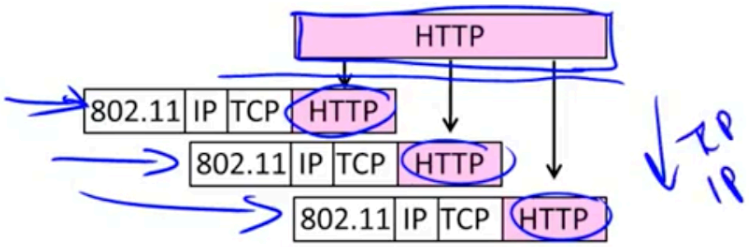


图 95: Recall 图二

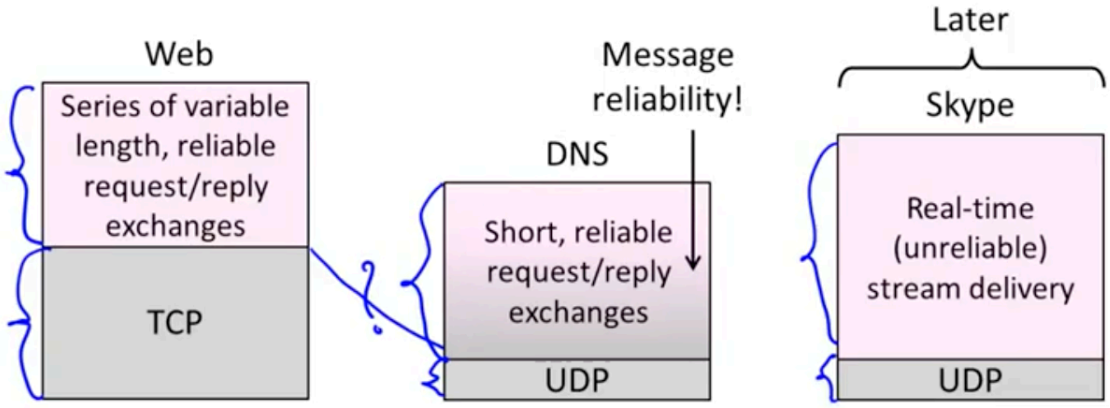


图 96: Application Communication Needs

17.4 OSI Session/Presentation Layers

- Remember this? Two relevant concepts ...
- See Figure 97

17.5 Session Concept

- A session is a series of related network interactions in support of an application task

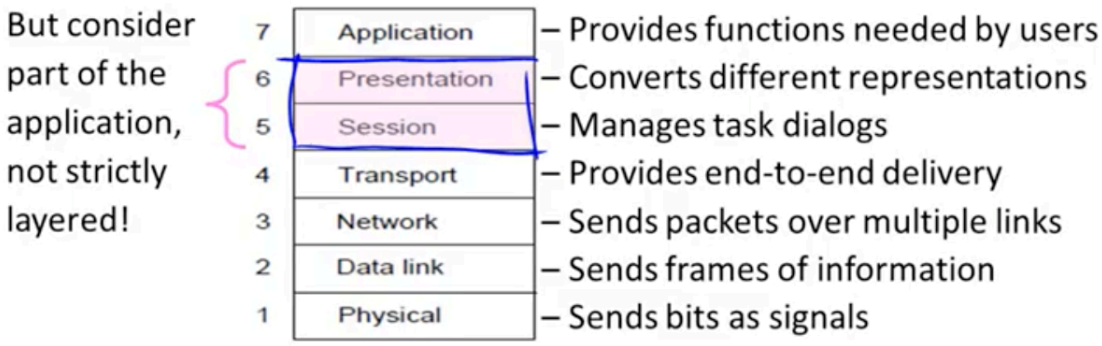


图 97: OSI Session/Presentation Layers

  - Often informal, not explicit

- Examples:

  - Web page fetches multiple resources
  - Skype call involves audio, video, chat

17.6 Presentation Concept

- Apps need to identify the type of content, and encode it for transfer

  - These are Presentation functions

- Examples:

  - Media (MIME) types, e.g., image/jpeg, identify the type of content
  - Transfer encodings, e.g., gzip, identify the encoding of the content
  - Application headers are often simple and readable versus packed for efficiency

17.7 Evolution(演化) of Internet Application

- Always changing, and growing ...
- See Figure 98

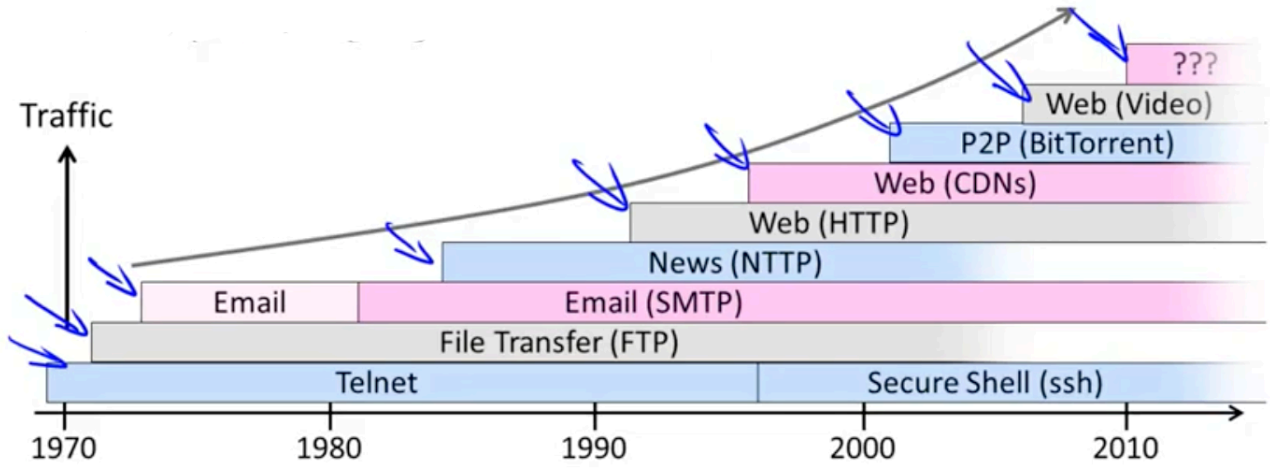


图 98: Evolution(演化) of Internet Application

- For a peek(窺視) at the state of the Internet:
  - Akamai's State of the Internet Report (quarterly(每季度的))
  - Cisco's Visual Networking Index
  - Mary Meeker's Internet Report
- Robust Internet growth, esp. video, wireless and mobile
  - Most traffic is video, will be 90% of Internet in a few years
  - Wireless traffic will soon overtake wired traffic
  - Mobile traffic is still a small portion (15%) of overall
  - Growing attack traffic from China, also U.S. and Russia

17.8 Topics

- 本次所讲的内容
- Evolving Internet applications
- 下次要讲的内容
- DNS (Domain Name System)
- HTTP (HyperText Transfer Protocol)
- Web proxies(代理人) and caching
- Content Distribution Networks
- Peer-to-peer (BitTorrent)
- 再晚一些要讲的内容
- Real-time applications (VoIP)

18 Week 8.2 - DNS Part 1

18.1 Topic of DNS Part 1

- The DNS (Domain Name System)
  - Human-readable host names, and more
  - Part 1: the distributed namespace
- See Figure 99

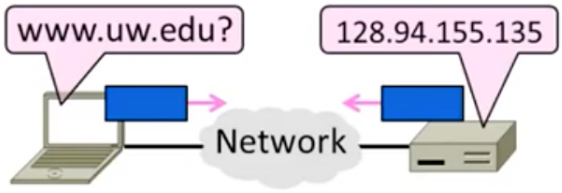


図 99: Topic of DNS Part 1

18.2 Names and Addresses

- Names are higher-level identifiers for resources
- Addresses are lower-level locators for resources
  - Multiple levels, e.g. full name → email → IP address → Ethernet address
- Resolution(解析) (or lookup) is mapping a name to an address
- See Figure 100

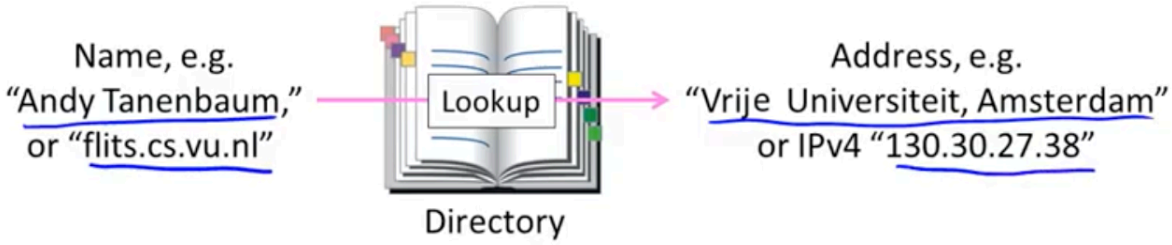


図 100: Names and Addresses

18.3 Before the DNS - HOSTS.TXT

- Directory was a file HOSTS.TXT regularly retrieved(取回) for all hosts from a central machine at the NIC (Network Information Center)

- Names were initially flat, became hierarchical (e.g., lcs.mit.edu) 85
- Neither manageable nor efficient as the ARPANET grew ...

18.4 DNS

- A naming service to map between host names and their IP addresses (and more)
  - www.uwa.edu.au → 130.95.128.140
- Goals:
  - Easy to manage (esp. with multiple parties)
  - Efficient (good performance, few resources)
- Approach:
  - Distributed directory based on a hierarchical namespace
  - Automated(使自動化) protocol to tie pieces together

18.5 DNS Namespace

- Hierarchical, starting from "." (dot, typically omitted)
- See Figure 101

18.6 TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
  - Starting in 1998; naming is financial, political, and international
- 22+ generic TLDs

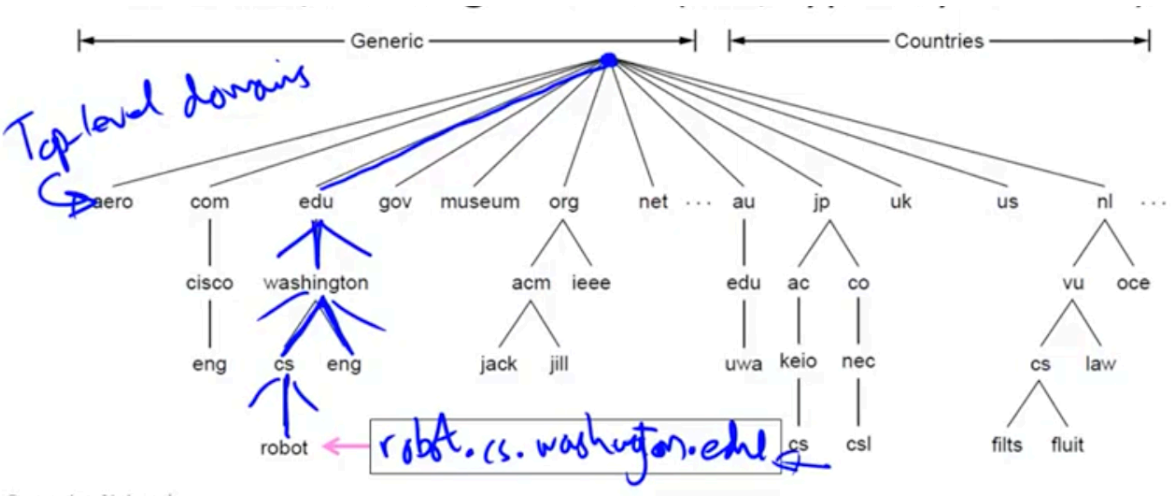


图 101: DNS Namespace

  - Initially .com, .edu, .gov, .mil, .org, .net
  - Added .aero, .museum, etc. from 2001 through .xxx in 2011
  - Different TLDs have different usage policies(有些可以乱用，但是如 edu 则是虚教育机构才可以)

- 250 country code TLDs
  - Two letters, e.g., ".au", plus international characters(比如汉字) since 2010
  - Widely commercialized, e.g., .tv (Tuvalu)
  - Many domain hacks, e.g., instagr.am (Armenia), goo.gl (Greenland)

18.7 DNS Zones

- A zone is a contiguous(毗鄰的) portion of the namespace
- See Figure 102

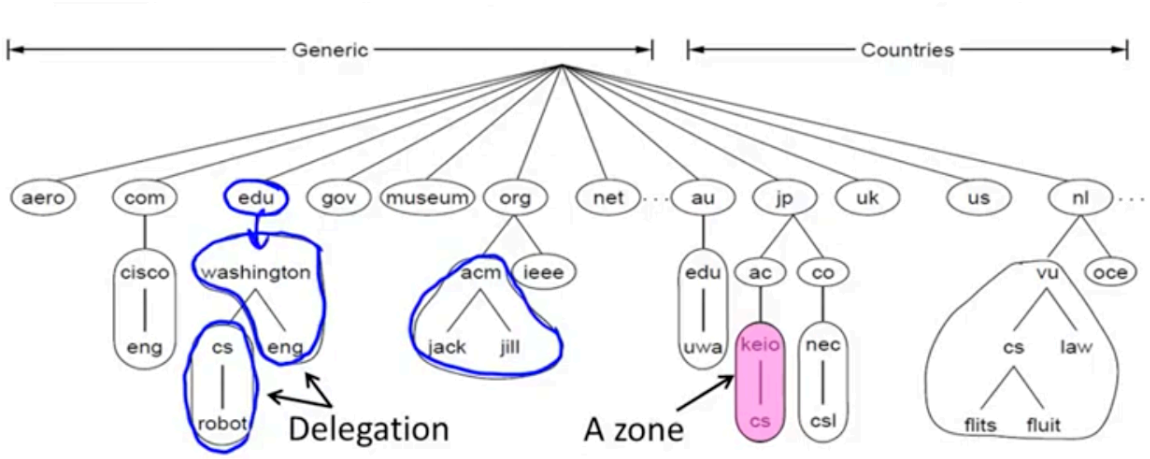


图 102: DNS Zones

- Zones are the basis for distribution
  - EDU Registrar administers .edu
  - UW administers washington.edu
  - CS&E administers cs.washington.edu
- Each zone has a nameserver to contact(聯F) for information about it
  - Zone must include contacts for delegations(分配; 委派), e.g., .edu knows nameserver for washington.edu
  - 也就是每个域至少需要一台 nameserver 来接收上一层所发来的信息，并回复下一层的信息到上一层。做的是桥梁的作用，把子域的孤岛信息由这个桥梁连接到上一层的码头上。

18.8 DNS Resource Records

- A zone is comprised(組成) of DNS resource records that give information for its domain names

- See Figure 103

Type	Meaning
SOA	Start of authority, has key zone parameters
A	IPv4 address of a host
AAAA (“quad A”)	IPv6 address of a host
CNAME	Canonical name for an alias
MX	Mail exchanger for the domain
NS	Nameserver of domain or delegated subdomain

图 103: DNS Resource Records 图一

- 下面是一个 DNS 资源记录的例子
- See Figure 104

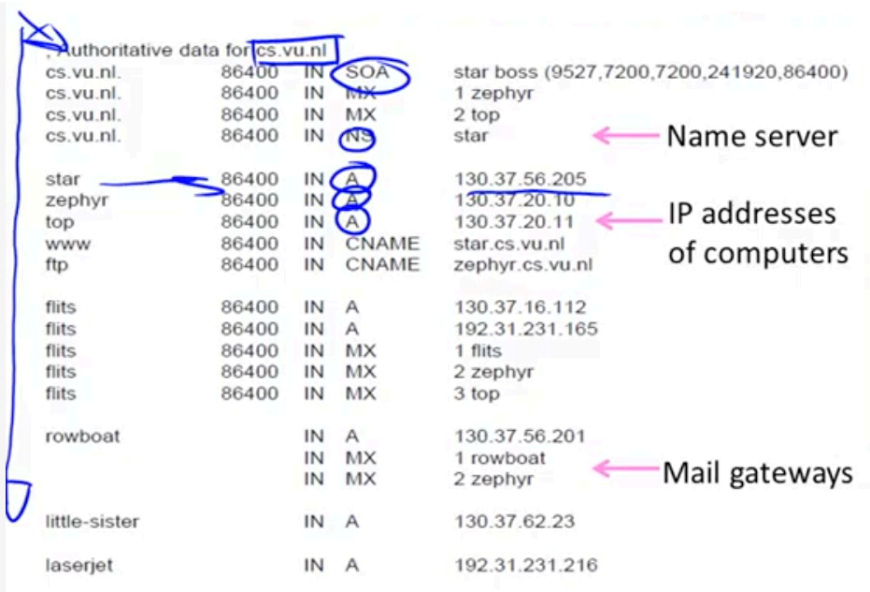


图 104: DNS Resource Records 图二

19 Week 8.3 - DNS Part 2

19.1 Topic of DNS Part 2

- Human-readable host names, and more
- Part2: Name resolution

19.2 Recall

- A zone is a contiguous portion of the namespace
  - Each zone is managed by one or more nameservers
- See Figure 105

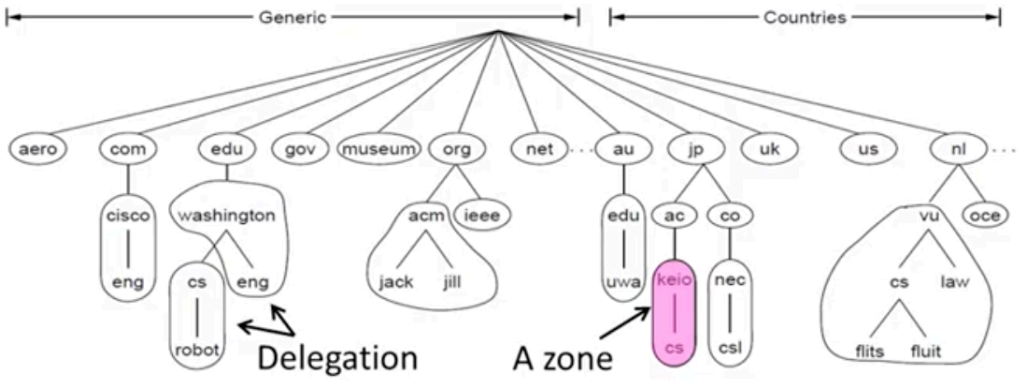


图 105: Recall

19.3 DNS Resolution(解析)

- DNS protocol lets a host resolve any host name (domain) to IP address
- If unknown, can start with the root nameserver and work down zones
- Let's see an example first ...

- flits.cs.vu.nl resolves robot.cs.washington.edu
- See Figure 106

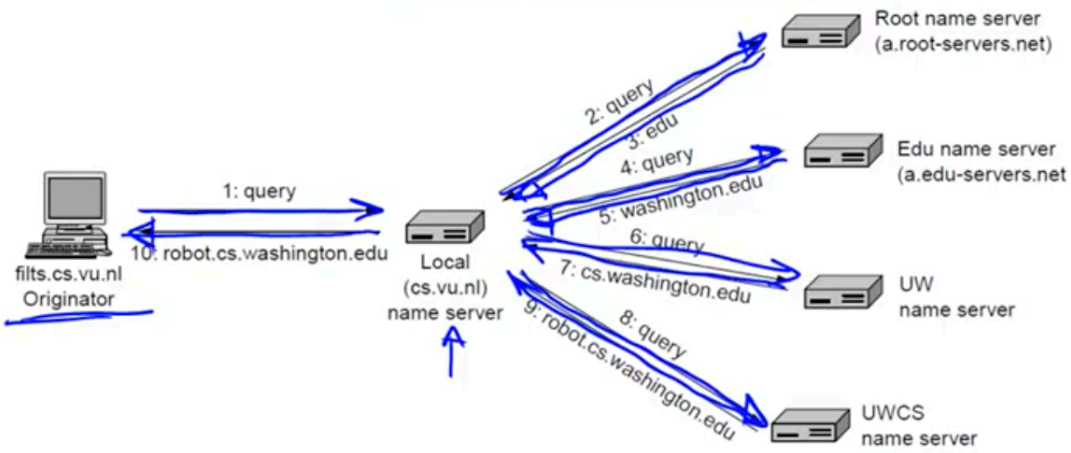


图 106: DNS Resolution(解析)

19.4 Iterative vs. Recursive Queries

- Recursive query
  - Nameserver completes resolution and returns the final answer
  - E.g., flits(輕快地飛（或移動）) → local nameserver
  - Lets server offload(卸去（累贅）) client burden(負荷) (simple resolvers) for manageability
  - Lets server cache over a pool of clients for better performance
- Iterative query
  - Nameserver returns the answer or who to contact next for the answer
  - E.g., local nameserver → all others

- Lets server ”file and forget”
- Easy to build high load servers

19.5 Caching

- Resolution latency should be low
  - adds delay to web browsing , 教授没说明原因。
- Cache query/responses to answer future queries immediately
  - Including partial (iterative) answers
  - Responses carry a TTL for caching ， 一般 TTL 设置为 1 天吧
- See Figure 107

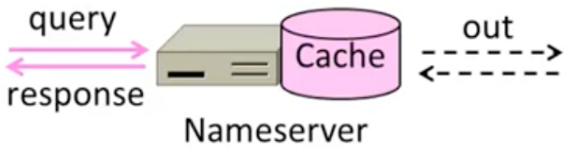


图 107: Caching 图一

- flits.cs.vu.nl resolves eng.washington.edu
  - And previous resolutions cut out most of the process
- See Figure 108

19.6 Local Nameservers

- Local nameservers typically run by IT (enterprise, ISP)
  - But may be your host or AP

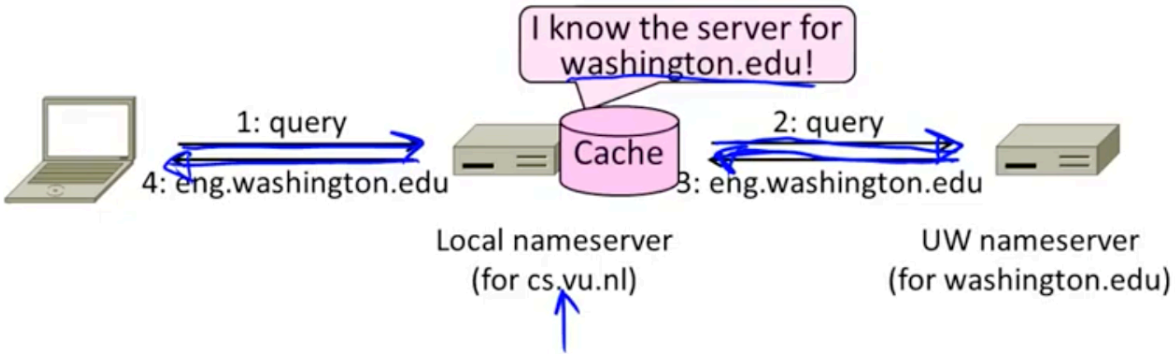


图 108: Caching 图二

  - Or alternatives e.g., Google public DNS

- Clients need to be able to contact their local nameservers
  - Typically configured via DHCP

19.7 Root Nameservers

- Root (dot) is served by 13 server names
  - a.root-servers.net to m.root-servers.net
  - All nameservers need root IP addresses
  - Handled via configuration file (named.ca)
- There are > 250 distributed server instances
  - Highly reachable, reliable service
  - Most servers are reached by IP anycast(任播是与单播（unicast）、广播（broadcast）和
    (Multiple locations advertise same IP ! Routes take client to the closest one. See §5.2.9)
  - Servers are IPv4 and IPv6 reachable

19.8 DNS Protocol

- Query and response messages
  - Built on UDP messages, port 53
  - ARQ for reliability; server is stateless!
  - Messages linked by a 16-bit ID field
- See Figure 109

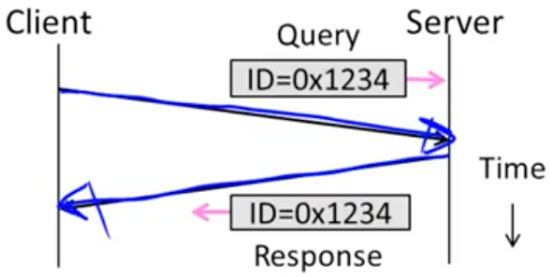


图 109: DNS Protocol 图一

- Service reliability via replicas
  - Run multiple nameservers for domain
  - Return the list; clients use one answer
  - Helps distribute load too
- See Figure 110
- Security is a major issue
  - Compromise(損害) redirects to wrong site!
  - Not part of initial protocols ...
- DNSSEC (DNS Security Extensions)

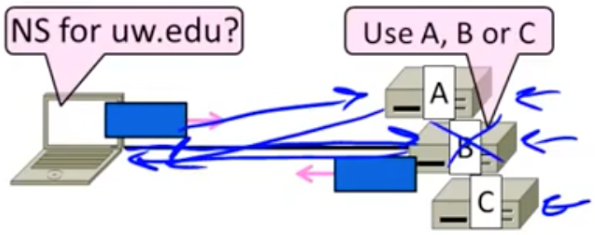


图 110: DNS Protocol 图二

- Long under development, now partially deployed. We'll look at it later

20 Week 8.4 - HTTP Introduction

20.1 Topic of HTTP Introduction

- HTTP, (HyperText Transfer Protocol)
  - Basis for fetching Web pages
- See Figure 111

图 111: Topic of HTTP Introduction

20.2 Sir Tim Berners-Lee (1955-)

- Inventor of the Web

  - Dominant(主要的) Internet app since mid 90s
  - He now directs the W3C
- Developed Web at CERN in 1989
  - Browser, server and first HTTP
  - Popularized via Mosaic (1993), Netscape
  - First WWW conference in 1994

20.3 Web Context

- See Figure 112

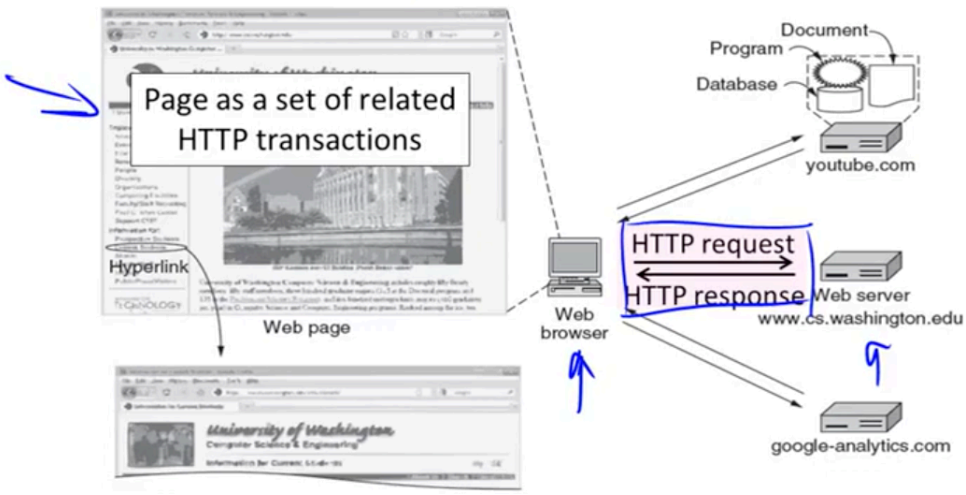


图 112: Web Context

20.4 HTTP Context

- HTTP is a request/response protocol for fetching Web resources
  - Runs on TCP, typically port 80

- Part of browser/server app

- See Figure 113

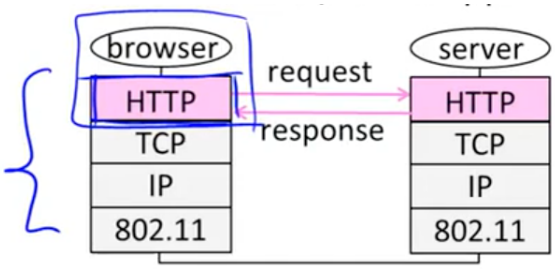


图 113: HTTP Context

20.5 Fetching a Web page with HTTP

- Start with the page URL:

- See Figure 114

图 114: Fetching a Web page with HTTP

- Steps:
  - Resolve the server to IP address (DNS)
  - Set up TCP connection to the server
  - Send HTTP request for the page
  - (Await HTTP response for the page)
  - ** Execute / fetch embedded resources / render(绘制)
  - Clean up any idle TCP connections

20.6 Static vs Dynamic Web pages

- Static web page is a file contents, e.g., image
- Dynamic web page is the result of program execution
  - Javascript on client, PHP on server, or both
- See Figure 115

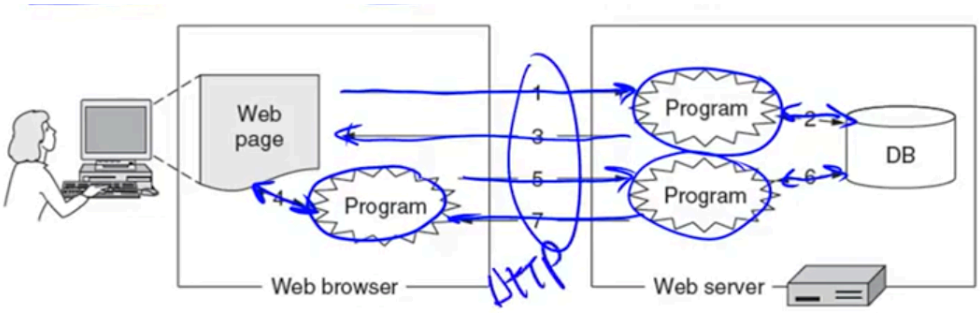


图 115: Static vs Dynamic Web pages

20.7 Evolution of HTTP

- Consider security (SSL/TLS for HTTPS) later
- See Figure 116

20.8 HTTP Protocol

- Originally a simple protocol, with many options added over time
  - Text-based commands, headers
- Try it yourself:
  - As a ”browser” fetching a URL

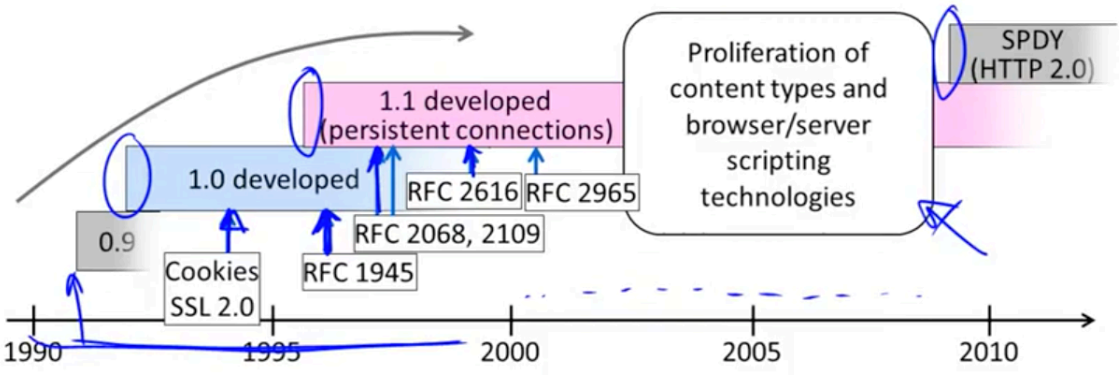


图 116: Evolution of HTTP

- Run "telnet en.wikipedia.org 80"
- Type "GET /wiki/Vegemite HTTP/1.0" to server followed by a blank line
- Server will return HTTP response with the page contents (or other info)

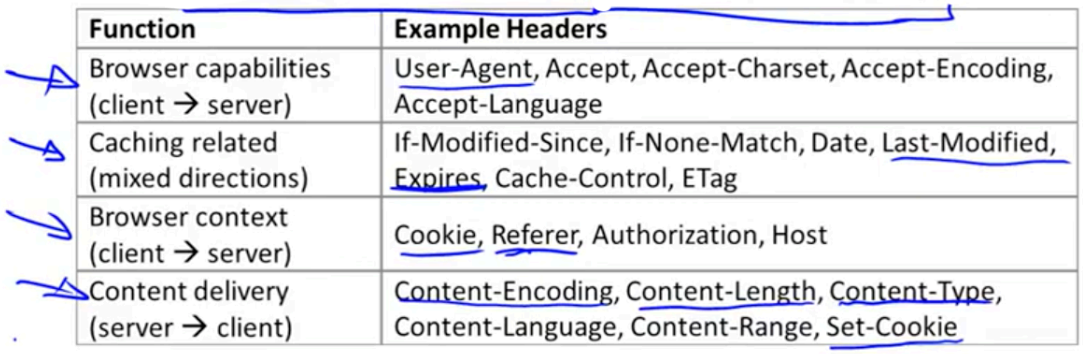
- Codes returned with the response
- See Figure 117

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Yes!

图 117: HTTP Protocol 图二

- Many header fields specify capabilities and content
  - E.g., Content-Type: text/html, Cookie: lect=8-4-http
- See Figure 118



Function	Example Headers
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

图 118: HTTP Protocol 图二

21 Week 8.5 - HTTP Performance

21.1 Topic of HTTP Performance

- Performance of HTTP
  - Parallel and persistent connections

21.2 PLT (Page Load Time)

- PLT is the key measure of web performance
  - From click until user sees page
  - Small increases in PLT decrease sales , 教授没有解释原因
- PLT depends on many factors

- Structure of page/content
- HTTP (and TCP!) protocol
- Network RTT and bandwidth

21.3 Early performance

- HTTP/1.0 uses one TCP connection to fetch one web resource
  - Made HTTP very easy to build
  - But gave fairly poor PLT ...
- See Figure 119

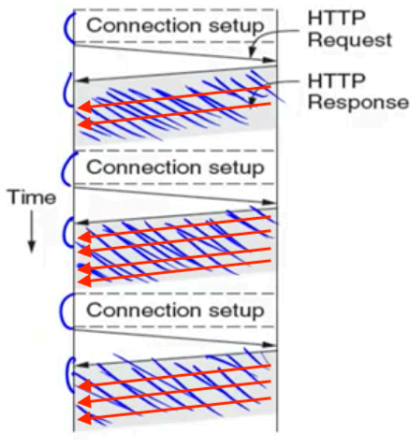


图 119: Early performance

- Many reasons why PLT is larger than necessary
  - Sequential request/responses, even when to different servers
  - Multiple TCP connection setups to the same server
  - Multiple TCP slow-start phases

- Network is not used effectively
  - Worse with many small resources / page

21.4 Ways to Decrease PLT

- 1. Reduce content size for transfer
  - Smaller images, gzip
- 2. Change HTTP to make better use of available bandwidth — 本次要讲
- 3. Change HTTP to avoid repeated transfers of the same content — 下次讲
  - Caching, and proxies
- 4. Move content closer to client — 再往后面会讲
  - CDNs [later]

21.5 Parallel Connections

- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handled concurrent requests for many clients
- How does this help ?
  - Single HTTP wasn't using network much ...
  - So parallel connections aren't slowed much

- Pulls in(节省) completion(結束) time of last fetch, 亦即整个运行时间缩短了。

21.6 Persistent Connections

- Parallel connections complete with each other for network resources
  - 1 parallel client ≈ sequential clients ?
  - Exacerbates(使加劇) network bursts, and loss
- Persistent connection alternative
  - Make 1 TCP connection to 1 server
  - Use it for multiple HTTP requests
- See Figure 120

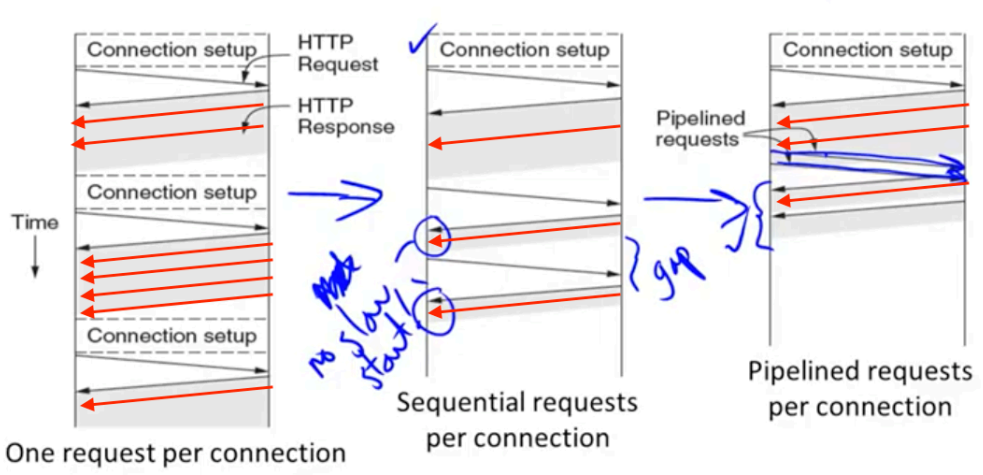


图 120: Persistent Connections

- Widely used as part of HTTP/1.1
  - Supports optional pipelining

  - PLT benefits depending on page structure(比如页面内容的多寡), but easy on network(增加网速则更易实现)

- Issues with persistent connections
  - How long to keep TCP connection ?
  - Can it be slower ? (Yes. But why ?)

22 Week 8.6 - HTTP Caching and Proxies

22.1 Topic of HTTP Caching and Proxies

- HTTP caching and proxies
  - Enabling content reuse
- See Figure 121

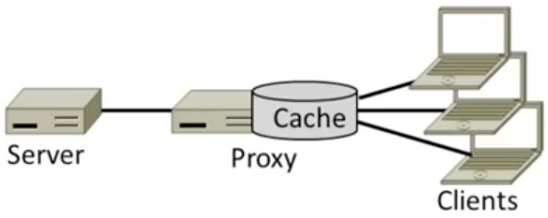


图 121: Topic of HTTP Caching and Proxies

22.2 Web Caching

- Users often revisit web pages
  - Big win from reusing local copy !
  - This is caching

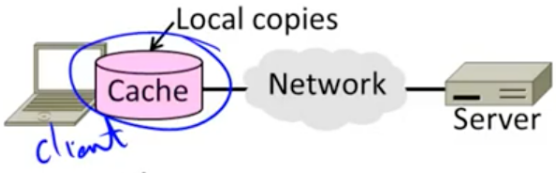


图 122: Web Caching 图一

- See Figure 122
- Key question:
  - When is it OK to reuse local copy ?
- Locally determine copy is still valid
  - Based on expiry(期滿) information such as ”expires” header from server
  - Or use a heuristic(🄴發式的) to guess (cacheable, freshly valid, not modified recently)
  - Content is then available right away
- Revalidate copy with remote server
  - Based on timestamp of copy such as ”Last-Modified” header from server
  - Or based on content of copy such as ”Etag” header from server
  - Content is available after 1 RTT
- Putting the pieces together:
- See Figure 123

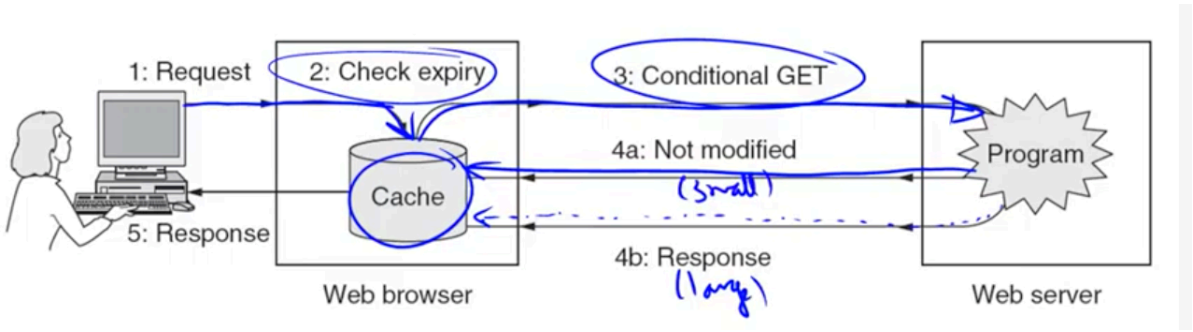


图 123: Web Caching 图二

22.3 Web Proxies

- Place intermediary between pool of clients and external web servers
  - Benefits for clients include greater caching and security checking
  - Organizational access policies too !
- Proxy caching
  - Clients benefit from larger, shared cache
  - Benefits limited by secure / dynamic content, as well as "long tail"
- Clients contact proxy; proxy contacts server
- See Figure 124

23 Week 8.7 - Content Delivery Networks (CDNs)

23.1 Topic of CDNs

- CDNs (Content Delivery Networks)

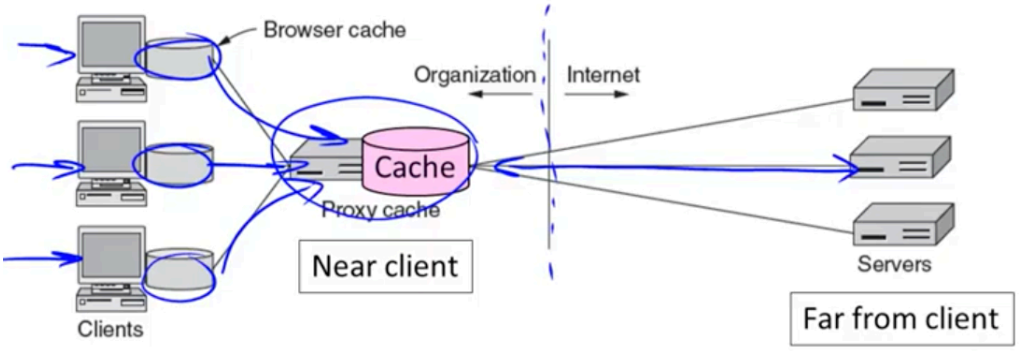


図 124: Web Proxies

  - Efficient distribution of popular content; faster delivery for clients

- See Figure 125

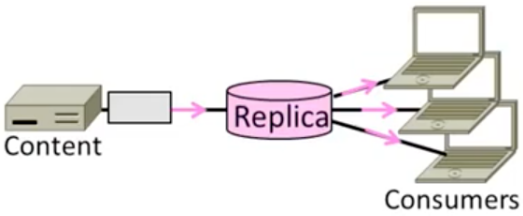


図 125: Topic of CDNs

23.2 Context(背景)

- As the web took off(飛起) in the 90s, traffic volumes grew and grew. This:

  - 1. Concentrated load on popular servers
  - 2. Led to congested networks and need to provision(供給) more bandwidth

  - 3. Gave a poor user experience

- Idea:
  - Place popular content near clients
  - Helps with all three issues above

23.3 Before CDNs

- Sending content from the source to 4 users takes $4 \times 3 = 12$ "network hops" in the example
- See Figure 126

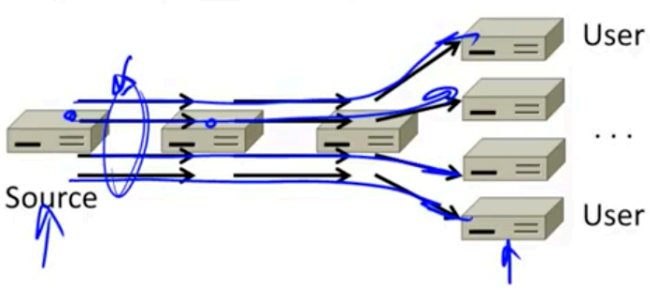


图 126: Before CDNs

23.4 After CDNs

- Sending content via replicas takes only $4 + 2 = 6$ "network hops"
- See Figure 127
- Benefits assuming popular content:
  - Reduces server, network load
  - Improves user experience (PLT)

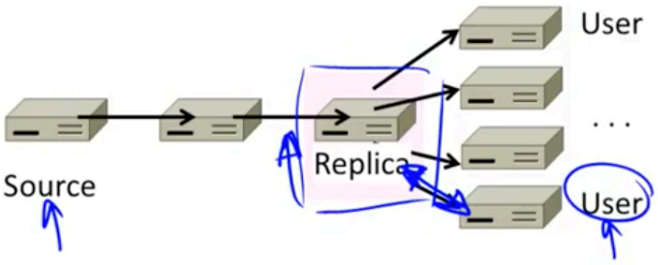


图 127: After CDNs

23.5 Popularity of Content

- Zipf's Law: few popular items, many unpopular ones; both matter
- See Figure 128

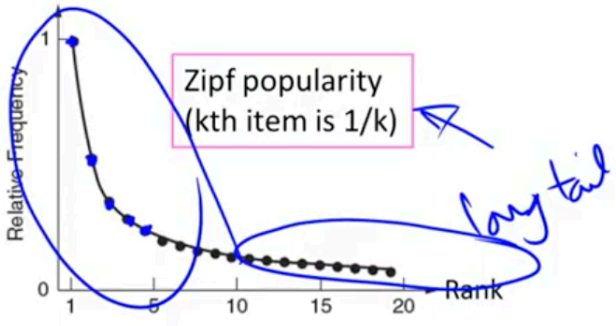


图 128: Popularity of Content

23.6 How to place content near clients ?

- Use browser and proxy caches
  - Helps, but limited to one client or clients in one organization
- Want to place replicas across the Internet for use by all nearby clients

- Done by clever use of DNS

23.7 Content Delivery Network

- 如下图所示，用域来划分 CDN 服务器服务的范围。
- See Figure 129

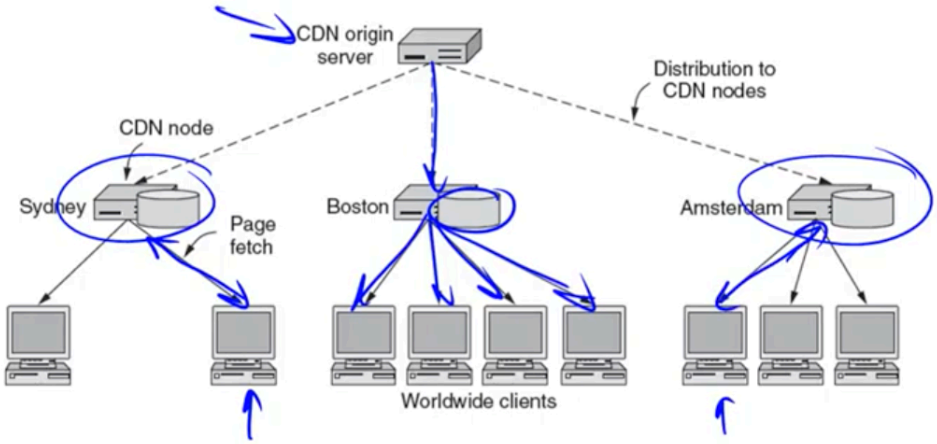


图 129: Content Delivery Network 图一

- DNS resolution of site gives different answers to clients
  - Tell each client the site is the nearest replica (map client IP)
- See Figure 130

23.8 Business Model

- Clever model pioneered(率先) by Akamai
  - Placing site replica at an ISP(ISP(Internet Service Provider) 所管辖的网络就叫骨干网，骨干网的服务商就叫 ISP) is win-win(双赢)

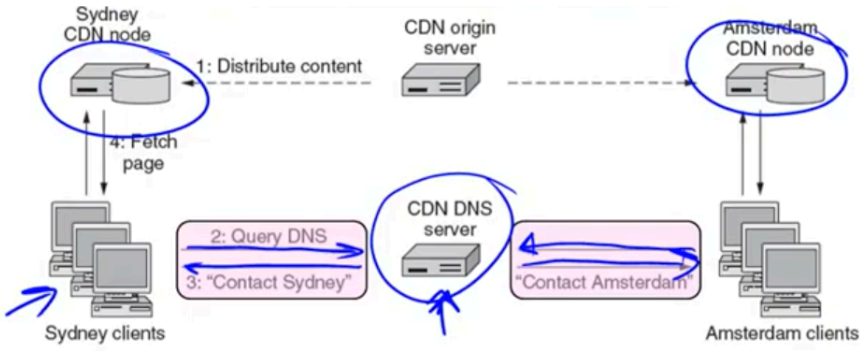


图 130: Content Delivery Network 图二

- Improves site experience and reduces bandwidth usage of ISP

- See Figure 131

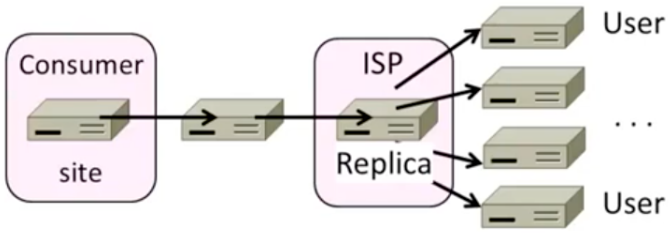


图 131: Business Model

24 Week 8.8 - Future of HTTP

24.1 Topic of Future of HTTP

- The Future of HTTP
  - How will we make the web faster ?
  - A brief look at some approaches

24.2 Modern Web pages

- Waterfall diagram shows progression(進展) of page load
- See Figure 132

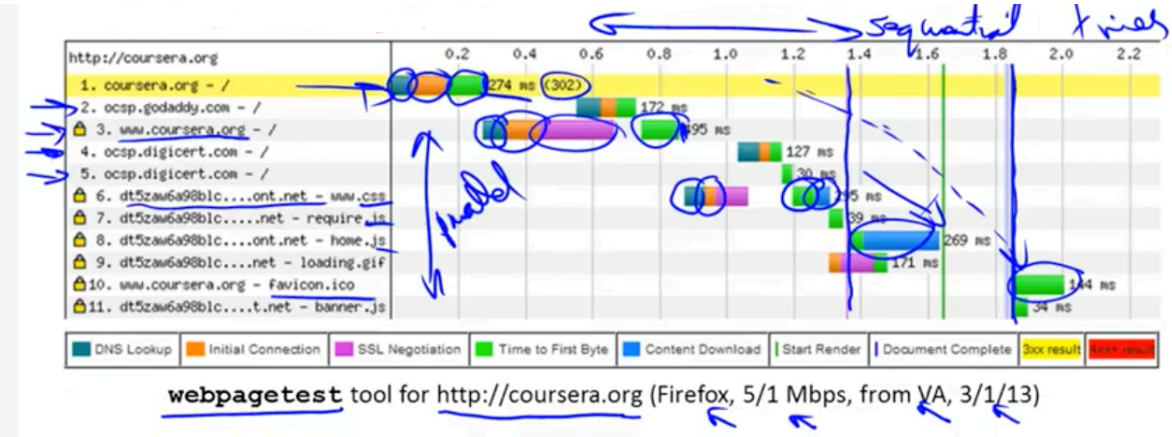


图 132: Modern Web pages 图一

- Yikes !
  - 23 requests
  - 1 Mb data
  - 2.6 secs
- See Figure 133
- Waterfall and PLT depends on many factors
  - Very different for different browsers
  - Very different for repeat page views
  - Depends on local computation as well as network
- See Figure 134

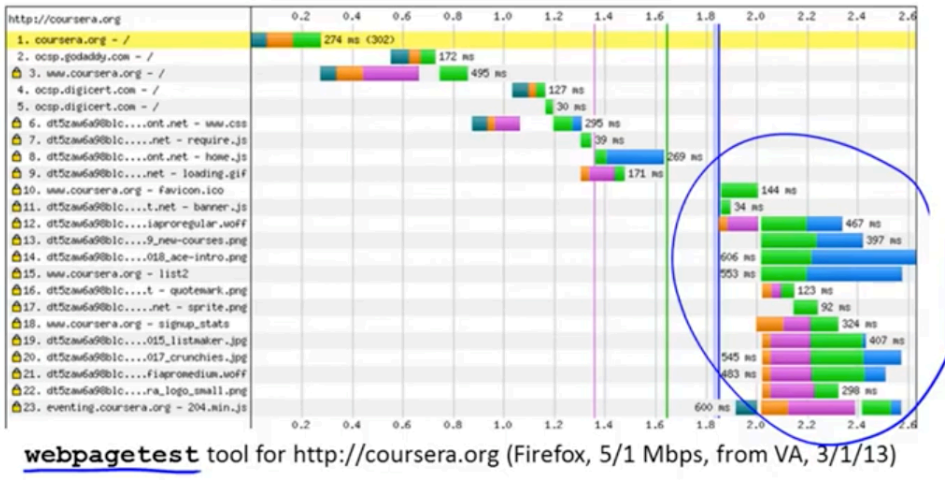


图 133: Modern Web pages 图二

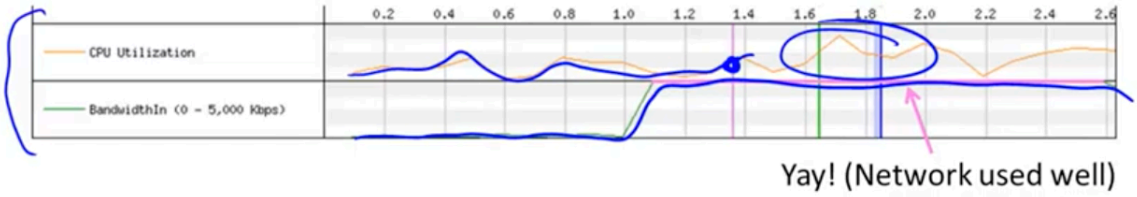


图 134: Modern Web pages 图三

24.3 Recent work to reduce PLT

- Pages grow ever more complex !
  - Larger, more dynamic, and secure
  - How will we reduce PLT ?
- 1. Better use of the network
  - HTTP/2 effort(努力) based on SPDY
- 2. Better content structures

  - mod_pagespeed server extension

24.4 SPDY ("speedy"(快速的)

- A set of HTTP improvements
  - Multiplexed (parallel) HTTP requests on one TCP connection
  - Client priorities for parallel requests
  - Compressed HTTP headers
  - Server push of resources
- Not being tested and improved
  - Default in Chrome, Firefox
  - Basis for an HTTP/2 effort

24.5 mod_pagespeed

- Observation:
  - The way pages are written affects how quickly they load
  - Many books on best practices for page authors and developers
- Key idea:
  - Have server re-write (compile) pages to help them load quickly !
  - mod_pagespeed is an example
- Apache server extension
  - Software installed with web server
  - Rewrites pages "on the fly" with rules based on best practices

- Example rewrite rules:
  - Minify Javascript
  - Flatten(把…弄平) multi-level CSS files
  - Resize images for client
  - And much more (100s of specific rules)

25 Week 8.9 - Peer to Peer Content Delivery BitTorrent

25.1 Topic of P2P content Delivery

- Peer-to-peer content delivery
  - Runs without dedicated(專用的) infrastructure
  - BitTorrent as an example
- See Figure 135

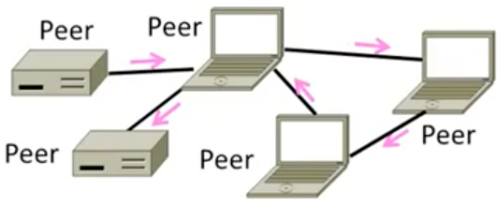


图 135: Topic of P2P content Delivery

25.2 Context

- Delivery with client/server CDNs:

  - Efficient, scales up(扩大规模) for popular content
  - Reliable, managed for good service
- ... but some disadvantages too:
  - Need for dedicated(專用的) infrastructure
  - Centralized control/oversight(監管)

25.3 P2P (Peer-to-Peer)

- Goal is delivery without dedicated infrastructure or centralized control
  - Still efficient at scale, and reliable
- Key idea is to have participants (or peers) help themselves
  - Initially Napster 1999 for music (gone)
  - Now BitTorrent 2001 onwards(（繼續）向前) (popular!)

25.4 P2P Challenges

- No servers on which to rely(依賴)
  - Communication must be peer-to-peer and self-organizing, not client-server
  - Leads to several issue at scale(大量的) ...
- 1. Limited capabilities
  - How can one peer deliver content to all other peers ?
- 2. Participation incentives(激勵)
  - Why will peers help each other ?

- 3. Decentralization
  - How will peers find content ?

25.5 Overcoming(攻克) Limited Capabilities

- Peer can send content to all other peers using a distribution tree
  - Typically done with replicas over time
  - Self-scaling(自缩放) capacity
- See Figure 136

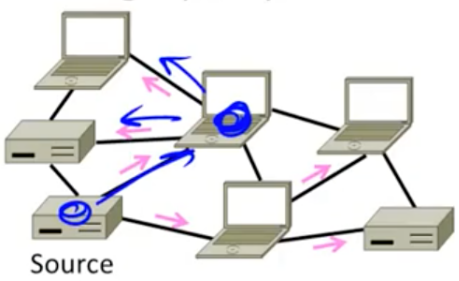


图 136: Overcoming Limited Capabilities

25.6 Providing Participation Incentives(激勵)

- Peer paly two roles:
  - Download(→) to help themselves, and upload(←) to help others
- See Figure 137
- Couple the two roles:
  - I'll upload for you if you upload for me
  - Encourages cooperation

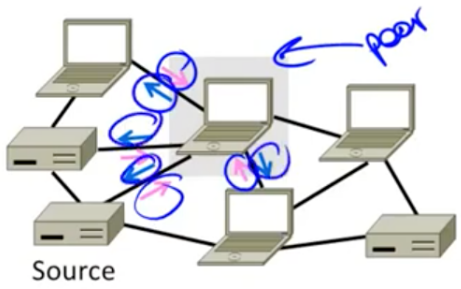


图 137: Providing Participation Incentives(激勵)

25.7 Enabling Decentralization

- Peer must learn where to get content
  - Use DHTs (Distributed Hash Tables)
- DHTs are fully-decentralized, efficient algorithms for a distributed index
  - Index is spread across all peers
  - Index lists peers to contact for content
  - Any peer can lookup the index
  - Started as academic work in 2001

25.8 BitTorrent

- Main P2P system in use today
  - Developed by Cohen in 2001
  - Very rapid growth, large transfers
  - Much of the Internet traffic today!
  - Used for legal and illegal content

- Delivers data using ”torrents”(洪流):
  - Transfers files in pieces for parallelism
  - Notable for treatment of incentives
  - Tracker or decentralized index (DHT)

25.9 BitTorrent Protocol

- Steps to download a torrent:
  - 1. Start with torrent description
  - 2. Contact tracker to join and get list of peers (with at least seed peer)
  - 2. Or, use DHT index for peers
  - Trade pieces with different peers
  - Favor(赞同) peers that upload to you rapidly; ”choke”(窒息) peers that don't(此处是省略结构，表示不快速给你传输的 peer) by slowing your upload to them
- All peers (except seed) retrieve(找回) torrent at the same time
- See Figure 138
- Dividing file into pieces gives parallelism for speed
- Choking unhelpful peers encourages participation
- See Figure 139
- DHT index (spread over peers) is fully decentralized
- See Figure 140

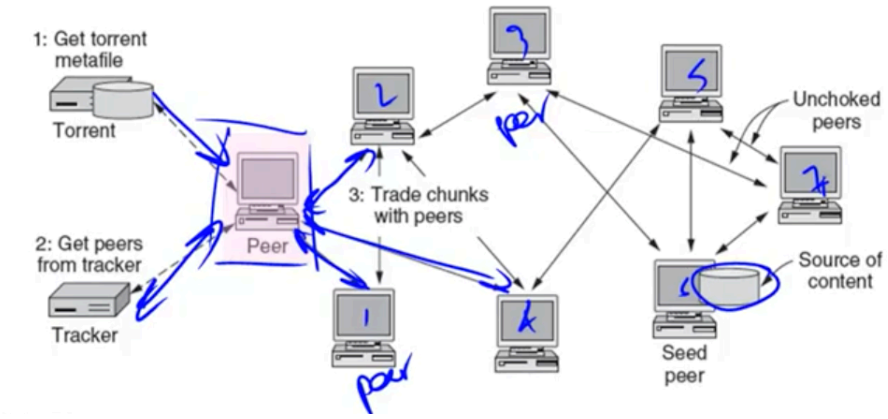


图 138: BitTorrent Protocol 图一

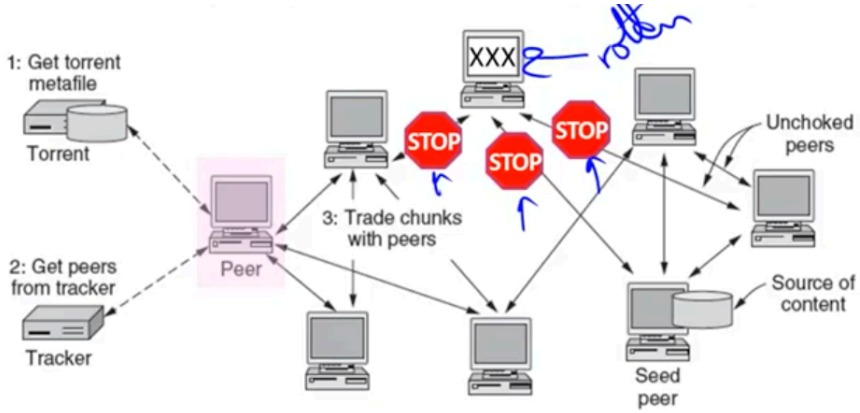


图 139: BitTorrent Protocol 图二

25.10 P2P Outlook(展望)

- Alternative to CDN-style client-server content distribution
  - With potential advantages
- P2P and DHT technologies finding more widespread use over time
  - E.g., part of skype, Amazon

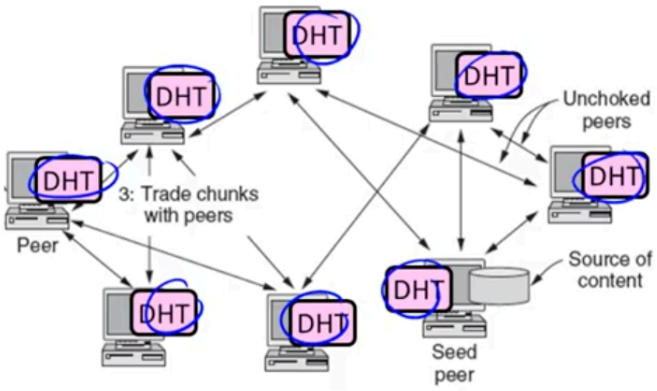


图 140: BitTorrent Protocol 图三

- Expect hybrid systems in the future

26 Week 9.1 - Quality of Service Overview

26.1 Where we are in the Course

- Revisiting the layers
  - Quality of Service (QOS) involves both the network and it users/applications

26.2 Topic of QOS

- QOS relates to the kind of service a user gets from the network
  - E.g., high/low bandwidth, delay, loss
  - Important issue for future Internet
- See Figure 141

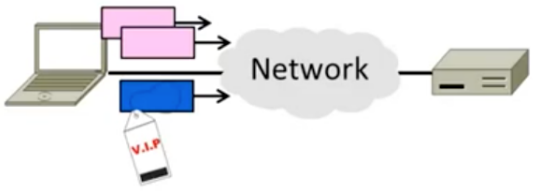


图 141: Topic of QOS

26.3 ”Best Effort”(最大努力) Service

- What we get in the Internet today with FIFO routers
  - Apps complete for bandwidth; queues add delay and loss
  - Try to deliver but no guarantee of bandwidth, delay, loss
- See Figure 142

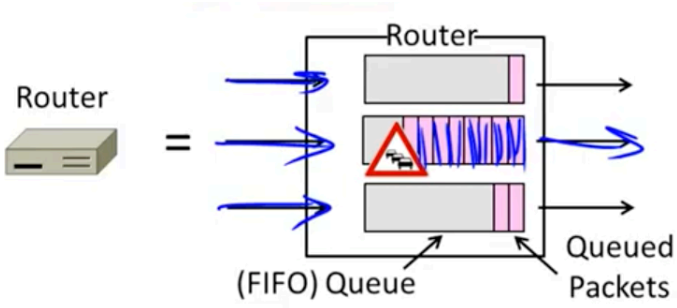


图 142: ”Best Effort”(最大努力) Service

26.4 QOS Motivation

- Best effort is not always enough!
  - May want performance guarantees
- What can’t be done:

- Guarantee more bandwidth or lower delay than exists in the network

- What can be done:
  - Control how bandwidth (hence(因为) delay/loss) is allocated to different users

26.5 Example - Skype and BitTorrent

- Home user runs Skype (VoIP only) and BitTorrent at the same time
  - Want low latency for Skype (real-time),
  - High throughput for TitTorrent (bulk(巨大的東西))
- See Figure 143

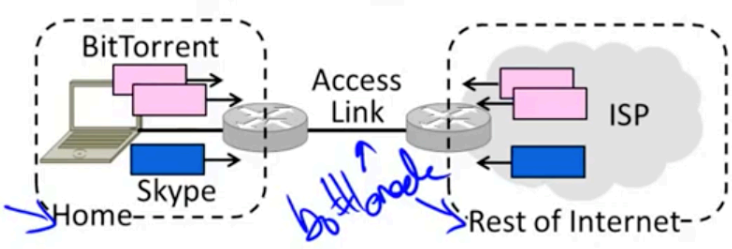


图 143: Example - Skype and BitTorrent 图一

- What happens with FIFO routers?
  - Skype and BitTorrent complete(完全占据) for bandwidth on access link (bottleneck)
  - Queues build at access routers ...
- See Figure 144

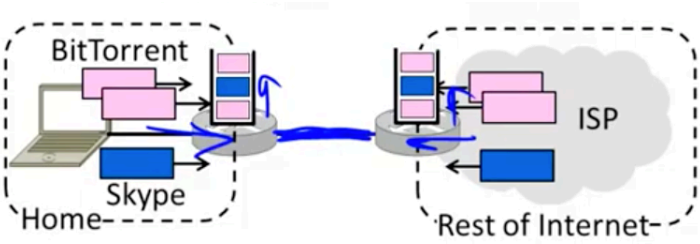


图 144: Example - Skype and BitTorrent 图二

- What happens ?
  - Skype call quality falls due to delay/loss
  - BitTorrent mostly unaffected by Skype
- What if we split the access link ?
  - Now Skype call quality is good ...
  - But BitTorrent loses bandwidth !
- See Figure 145

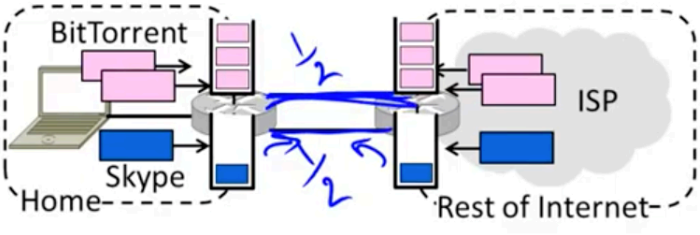


图 145: Example - Skype and BitTorrent 图三

- A better idea to try ?
  - Modify access routers to give priority to Skype packets on access link !

- See Figure 146

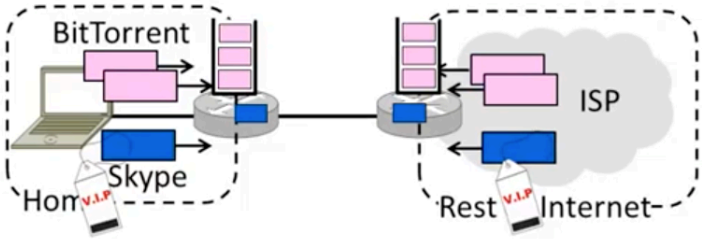


图 146: Example - Skype and BitTorrent 图四

- What happens now?
  - High Skype call quality and high BitTorrent throughput - both win !

26.6 Quality of Service

- Allocate bandwidth in a way that improves app/user performance
  - Guarantee bandwidth to an app
  - Satisfy multiple apps at once
  - Will explore(探索) ideas beyond priority
- To provide QOS, we need to know what apps require of the network
  - Need for bandwidth, delay, loss

26.7 Application Requirements

- HIGH stringency(严格性) means high bandwidth, low delay/loss
- See Figure 147

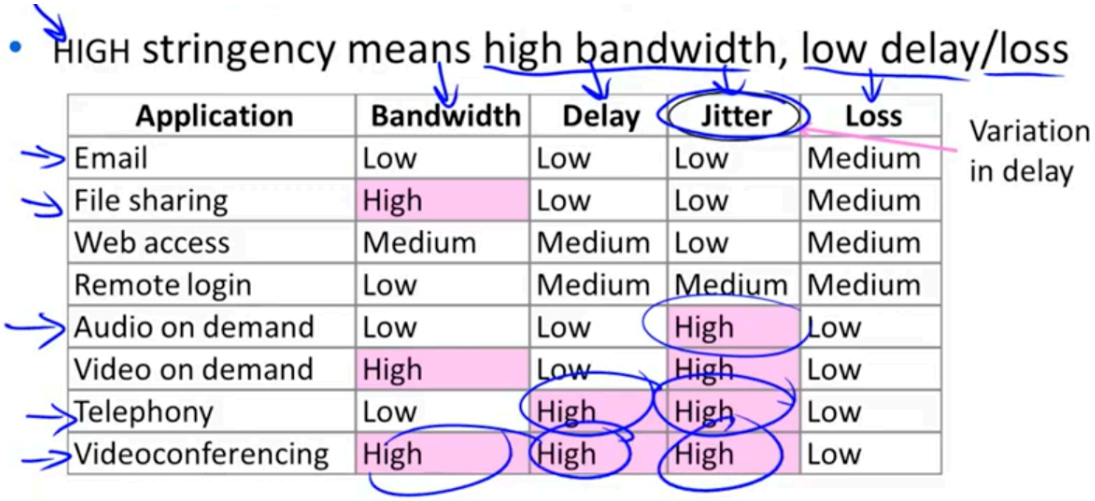


图 147: Application Requirements

26.8 Over-Provisioning(预留空间过大)

- A caveat(告誡) is that QOS matters only when there is a network bottleneck
  - Otherwise no queuing or loss ...
  - And hence no opportunity to improve
- Leads to over-provisioning approach:
  - Build heaps of network capacity!
  - Simple alternative(可供選擇的) to QOS, but not cost-effective and no guarantee

26.9 Topics

- 本次讲的内容
- Application requirements

- 下次要讲的内容
- Real-time transport (VoIP)
- Streaming media (video)
- 下次要讲的内容 – 未来的网络
- Fair Queuing
- Traffic Shaping
- Differentiated services
- Rate/Delay guarantees

27 Week 9.2 - Real Time Transport

27.1 Topic of Real Time Transport

- Sending interactive(交互的) real-time media over the network, e.g., VoIP
  - Using the best effort Internet
  - Playout(播放) buffer technique
- See Figure 148

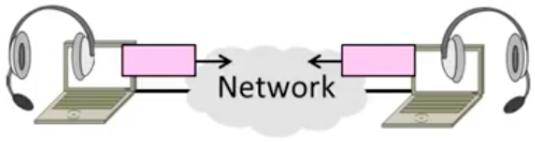


图 148: Topic of Real Time Transport

27.2 Challenge - Network Delay

- Consider one direction
  - Constant rate of media is generated at source, later consumed at receiver
  - Network must have enough bandwidth, and adds a delay
- See Figure 149

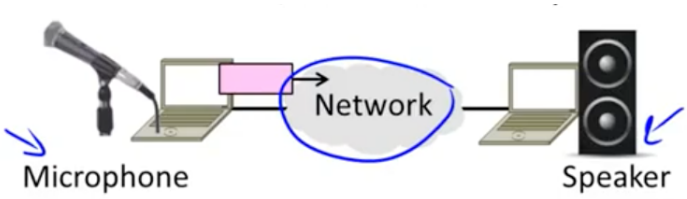


图 149: Challenge - Network Delay 图一

- Network delay is variable
  - Message latency plus queuing delay
  - Variability in delay is called jitter
- See Figure 150

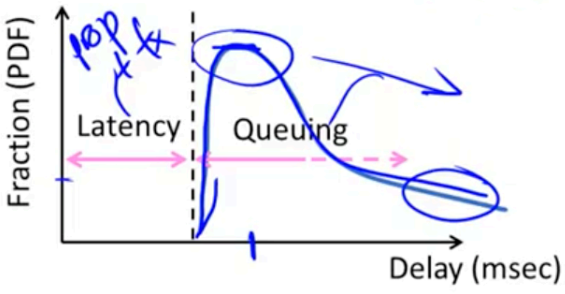


图 150: Challenge - Network Delay 图二

27.3 Playout

- Ideally want fixed, and small network delay for interactivity
  - Emulate(效仿) the telephone network
- See Figure 151

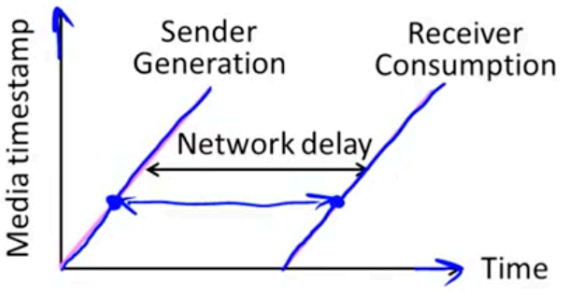


图 151: Playout 图一

- Media arrives at receiver after variable network delay
- See Figure 152

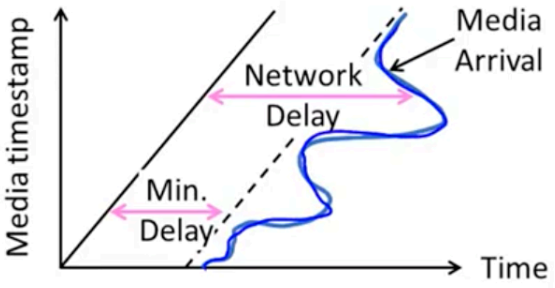


图 152: Playout 图二

27.4 Playout Buffer

- Put media in playout buffer at receiver until consumption time

– Smooth out(铺平) variable network delay

- See Figure 153

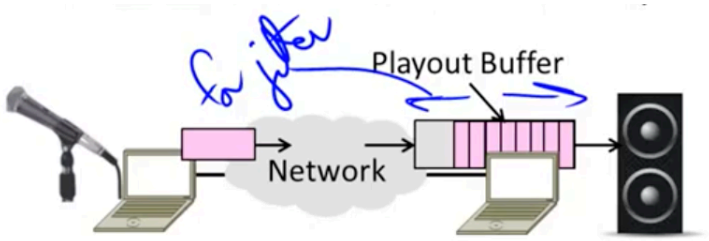


图 153: Playout Buffer 图一

- Media arrival curve determines(Ⓕ定) time in playout buffer and deadline
- 下图中粉红部分的水平宽度表示 buffer 所能缓存的播放时间。
- See Figure 154

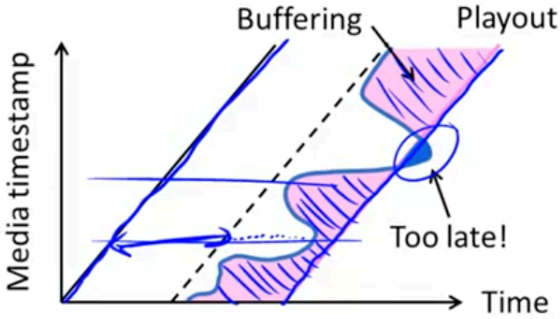


图 154: Playout Buffer 图二

- Pick largest acceptable network delay to set the playout point
- See Figure 155

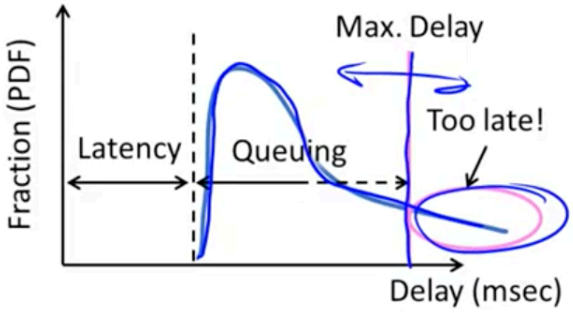


图 155: Playout Buffer 图三

- Tradeoff:
  - Larger acceptable network delay → larger buffer/delay, less loss
  - Smaller acceptable network delay → smaller buffer/delay, more loss
- Typically can't recover(失而復得) loss for interactive, real-time scenario
  - Instead, do without (glitch(小差錯))

27.5 Components of a Real-Time Session

- A call consists of several parts:
  - Call setup, with SIP
  - Session description, with SDP
  - Media transport, with RTP
  - Media playout, with buffer
- May have audio/video, multiple parties, mobility, etc.

27.6 RTP (Real-time Transport Protocol)

- Used to carry media to top of best effort UDP (§6.4.3)
  - Header has media format, timestamp, sequence number, etc.
  - Media follows in standard formats, e.g., G.711, MP4
- See Figure 156

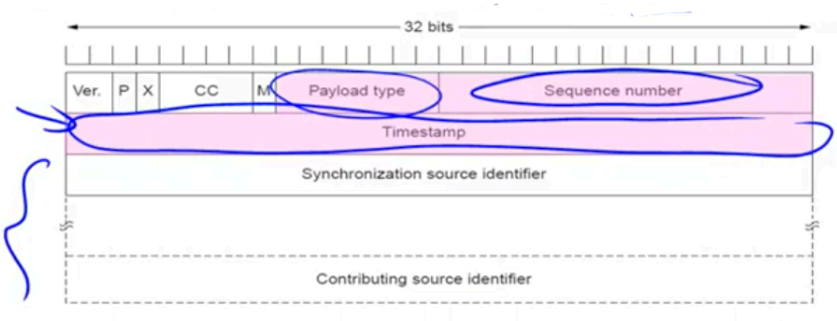


图 156: RTP (Real-time Transport Protocol)

27.7 SIP (Session Initiation Protocol)

- Open protocol for establishing voice and video calls over IP
  - Provides the signaling(发信号); media is carried directly with RTP (or other)
- This is not Skype
  - It uses a proprietary(專有的) protocol ...

27.8 SIP Singaling

- Signaling for call control

  - Like HTTP, uses simple method/response codes
  - Runs on UDP or TCP
  - SIP proxy servers and registrars(登記員) provide mobility (not shown)

- See Figure 157

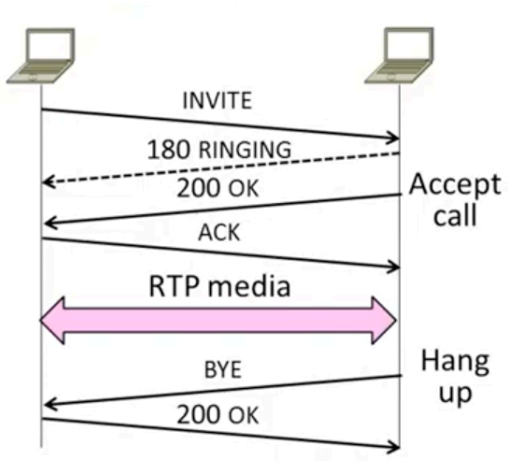


図 157: SIP Singaling

28 Week 9.3 - Streaming Media

28.1 Topic of Streaming Media

- Playback(回放) of media over the network
  - Using the best effort Internet
  - Coursera, YouTube, Netflix, etc.
  - Huge usage !

28.2 Streamed vs. Interactive Media

- Streamed is less demanding(苛刻的) case:
  - Only a single direction to consider
  - Low delay not essential(必要的); affects startup but not interactivity
  - Still need to handle bandwidth, jitter
- See Figure 158

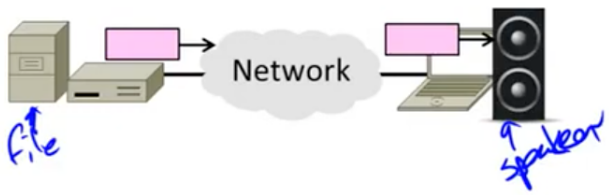


图 158: Streamed vs. Interactive Media

28.3 Handling Jitter

- As before, buffer media at receiver until ready for playout time
  - Smooth out variable network delay
- See Figure 159
- Use $HIGH$ and LOW watermarks to control source over/underfill (底部填充)
  - Start pulling media at low level
  - Stop pulling media at high level
- See Figure 160

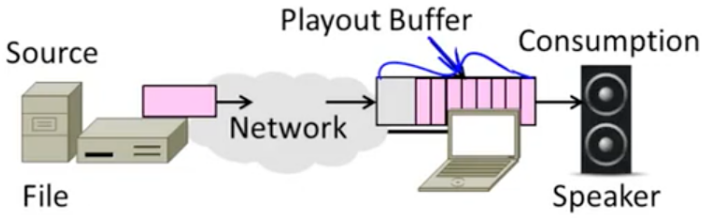


图 159: Handling Jitter 图一

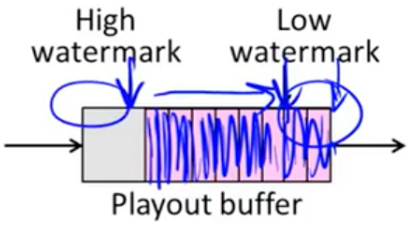


图 160: Handling Jitter 图二

28.4 Handling Bandwidth

- Send file in one of multiple encodings
  - Higher quality encodings require more bandwidth
  - Select best encoding given available bandwidth
- See Figure 161

28.5 Streaming over TCP or UDP ?

- UDP minimizes message delay for interactive, real-time sessions
- TCP is typically used for streaming
  - Low delay is not essential; startup

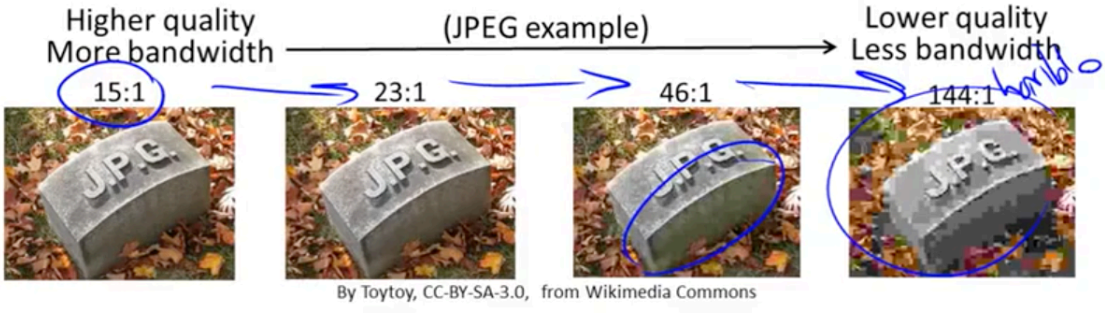


图 161: Handling Bandwidth

- Loss recovery(寻回) simplifies presentation(描述),TCP 已经把 Loss 等给做好了，这里就很简单提供 media 的描述就行了。
- HTTP/TCP passes through firewalls

28.6 Components of Streaming Media

- Session consists of several parts:
  - Signaling(发信号), e.g., with RTSP
  - Media transport, e.g., with HTTP
  - Media playout, with buffer
  - Evolving standards, e.g., HTML5
- Typically to an individual party
  - Use CDNs to reach many viewers

28.7 Streaming with RTSP

- Video started using HTTP to get metafile
- Invokes media player

  - Talks RTSP (Real-Time Streaming Protocol) to media server

- Media sent with, e.g., RTP over TCP/UDP
- See Figure 162

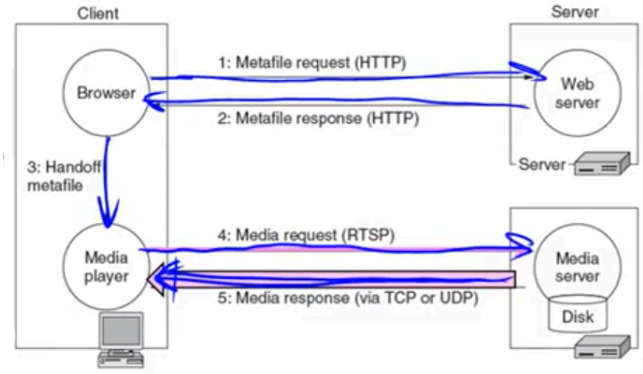


图 162: Streaming with RTSP

28.8 Streaming with HTTP

- Fetch media description data
  - Gives index of clips(片段), rates
- Fetch small segments
  - Put in playout buffer
- Adapt selection of encoding
  - Based on buffer occupancy(占用)
- Evolving(演变) standards, e.g., DASH
  - Leverages(影響力) HTTP and HTML5

  - Server is otherwise stateless

- See Figure 163

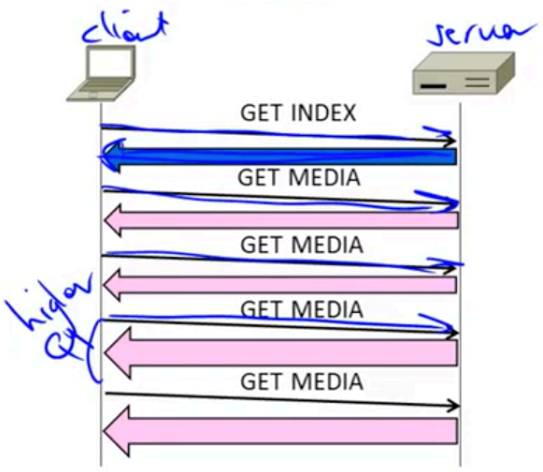


图 163: Streaming with HTTP

29 Week 9.4 - Fair Queuing

29.1 Topic of Fair Queuing

- Sharing bandwidth between flows
  - WFQ (Weighted Fair Queuing
  - Key building block for QOS
- See Figure 164

29.2 Sharing with FIFO Queuing

- FIFO "drop tail" queue:

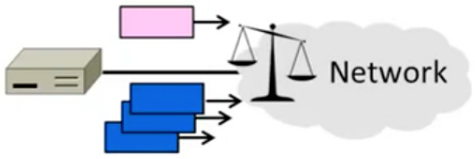


图 164: Topic of Fair Queuing

  - Queue packets First In First Out (FIFO)
  - Discard new packets when full
  - Typical router queuing model
- Sharing with FIFO queue
  - Multiple users or flows send packets over the same (output) link
  - What will happen ?
- Bandwidth allocation depends on behavior of all flows
  - TCP gives long-term(长期) sharing - with delay/loss, and RTT bias
  - Aggressive(富於攻擊性的) user/flow can crowd out others
- See Figure 165

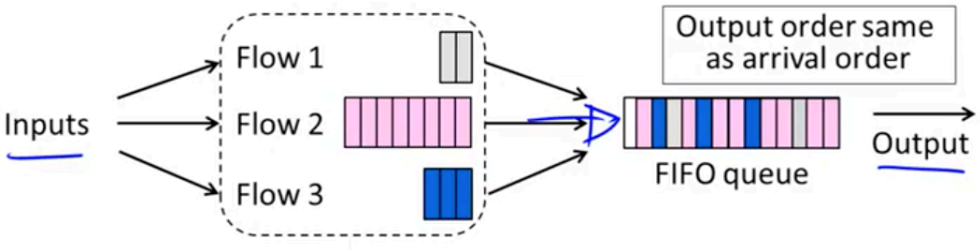


图 165: Sharing with FIFO Queuing

29.3 Round-Robin Queuing

- Idea to improve fairness:
  - Queue packets separately for each flow; take one packet in turn from each non-empty flow at the output time
- How well does this work?
- See Figure 166

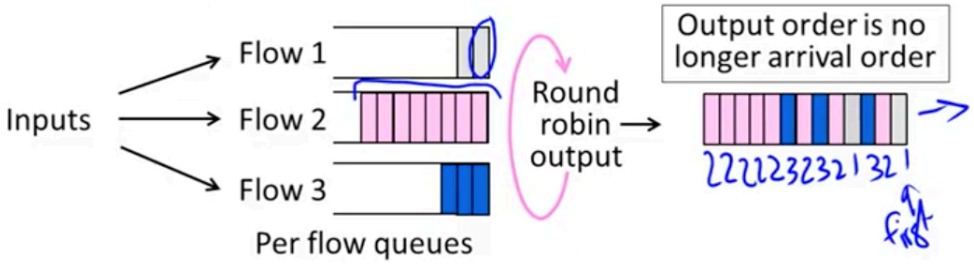


图 166: Round-Robin Queuing

- Flows don't see uncontrolled delay/loss from others!
- But different packet sizes lead to bandwidth imbalance
  - Might be significant, e.g., 40 bytes vs. 1500 bytes

29.4 Fair Queuing

- Round-robin but approximate bit-level fairness:
  - Approximate by computing virtual finish time
  - Virtual clock ticks once for each bit sent from all flows
  - Send packets in order of their virtual finish times, $Finish(j)_F$
  - Not perfect - don't preempt(抢占) packet being transmitted

- See Figure 167

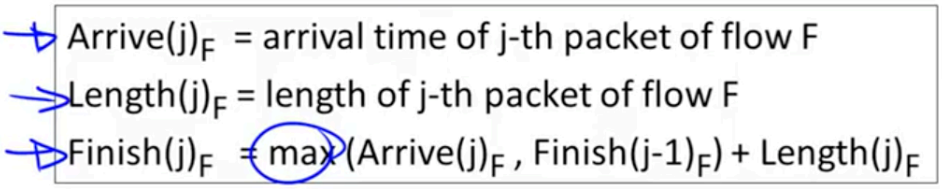


图 167: Fair Queuing 图一

- Suppose:
  - Flow 1 and 3 use 1000B byte packets, flow 2 uses 300B packets
  - What will fair queuing do ?
  - 下面公式中后一个到达时间是小于前一个完成的时间，也就是前一个在后一个到达之后才会传输完成。

$$
\begin{aligned}
&Let Finish(0)_F = 0, queues backlogged[Arrive(j)_F < Finish(j-1)_F] \\
&Finish(1)_{F1} = 1000, Finish(2)_{F1} = 2000, ... \\
&Finish(1)_{F2} = 300, Finish(2)_{F2} = 600, Finish(3)_{F2} = 900, 1200, 1500, ... \\
&Finish(1)_{F3} = 1000, Finish(2)_{F3} = 2000, ...
\end{aligned} \tag{1}
$$

- See Figure 168

29.5 WFQ (Weighted Fair Queuing)

- WFQ is a useful generalization(普遍化) of Fair Queuing:
  - Assign a weight, $Weight_F$, to each flow

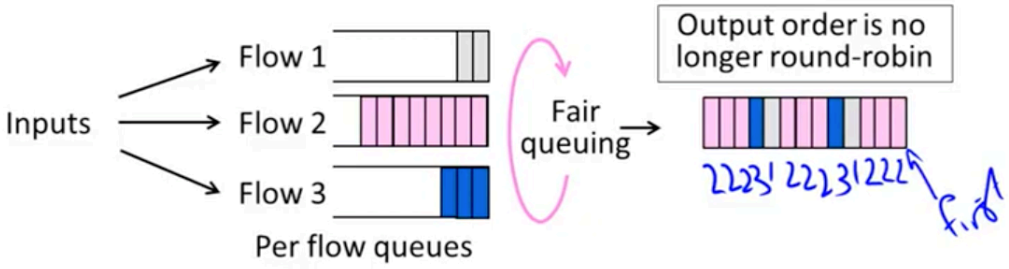


图 168: Fair Queuing 图二

  - Higher weight gives more bandwidth, e.g., 2 is 2 × *bandwidth*
  - Change computation of $Finish(j)_F to factor in Weight_F$
  - 表示完成时间认为变小了，所以就能多发送包。解决方式也非常巧妙。

- See Figure 169

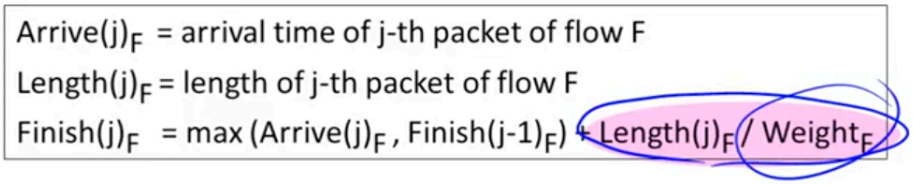


图 169: WFQ (Weighted Fair Queuing) 图一

- An example you can work through(（逐步）解🄵（困難的）問題) ...
- 下面的表格符合之前给定的公式。
- See Figure 170

29.6 Using WFQ

- Lots of potential !

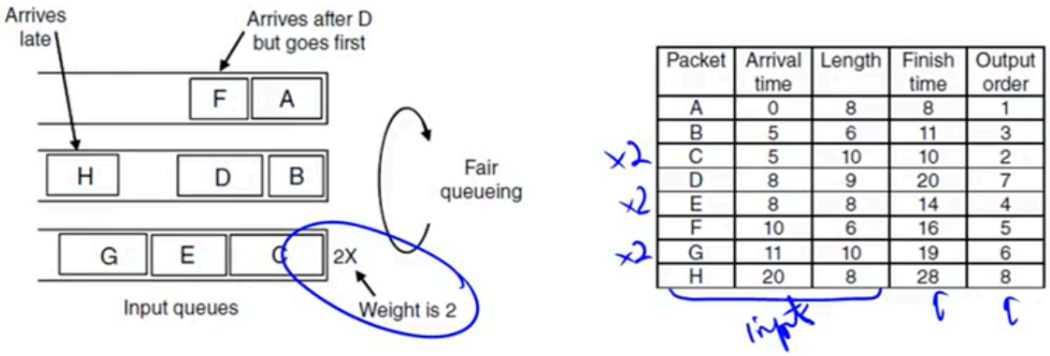


图 170: WFQ (Weighted Fair Queuing) 图二

  - Can prioritize and protect flows
  - A powerful building block

- Not yet a complete solution
  - Need to determine flows (user? application? TCP connection?)
  - Difficult to implement at high speed for many concurrent flows
  - Need to assign weights to flows

30 Week 9.5 - Traffic Shaping

30.1 Topic of Traffic Shaping

- Shaping traffic to constrain(約束) bursts
  - Token(象征) buckets
  - Key building block for QOS

- See Figure 171

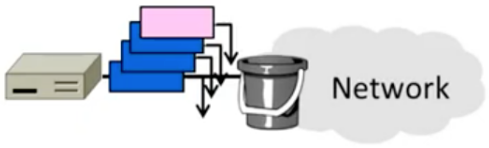


图 171: Topic of Traffic Shaping

30.2 Motivation

- Shaping traffic flows constrains(約束) the load they may place on the network
  - 1. Limiting the total traffic enables bandwidth guarantees
  - 2. Limiting bursts avoids unnecessary delay and loss
- How should we shape traffic ?
  - Real apps generate varying traffic - unrealisitc to smooth it out
- 这里讲的是在发送端搞一个水桶来乘这些 packets
- Flow A and flow B have the same average rate
  - 1 Mbps over 3.5 secs!
  - But they have very different behaviors!
- Average rate alone is not a good descriptor of behavior ...
- See Figure 172
- How should we describe traffic flows to the network?
  - Average rate matters; relates to long-term bandwidth
  - Burstiness also matters; relates to short-term bandwidth

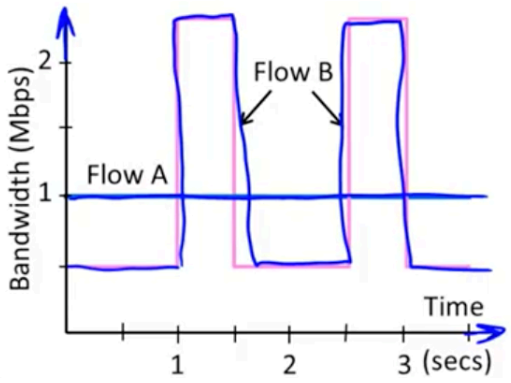


图 172: Motivation

- Two characteristics useful
  - More expressive(有表現力的) than average
  - Still relatively simple

30.3 Token Bucket

- (R, B) token bucket constrains:
  - Average rate of R bits/sec
  - Bursts (over R) of B bits
- See Figure 173
- Sending removes tokens (or credits)(代幣) from the bucket; no credit, no send
  - Fill rate of R bits/sec
  - Bucket capacity of B bits
- Constrains greatest traffic over time
- See Figure 174

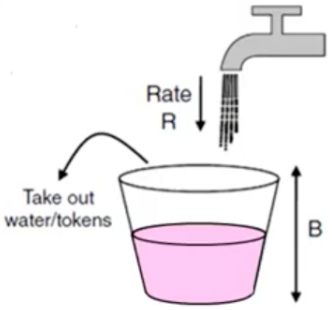


图 173: Token Bucket 图一

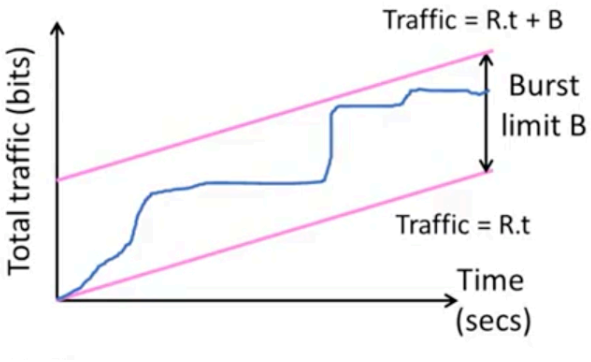


图 174: Token Bucket 图二

30.4 Shaping vs. Policing(監督)

- Shaping modifies traffic near the source to fit within an (R, B) profile(人物簡介)
  - Run (R, B) token bucket at the source
  - Pass sent packets to the network when there are tokens
  - Delay (queue) packets while more tokens arrive
- Lets user condition(習慣於) their traffic to meet the network contract(收縮)

- Policing verifies that traffic within the network fits an (R,B) profile
  - Run (R, B) token bucket at network edge
  - Let packets into the network when there are tokens
  - Demote or discard packets when there are insufficient(不足的) tokens
- Lets network check traffic to verify it meets the user's contract(収縮)

30.5 Usage for QOS

- Token buckets help the user and network regulate(調整) traffic for QOS
  - Network can limit the traffic for preferential(優先的) treatment
  - User can flexibly select that traffic
- Special treatment is implemented with other means such as WFQ

31 Week 9.6 - Differentiated Services

31.1 Topic of Differentiated Services

- Treating different traffic flows differently in the network
  - Coarse(粗糙) QOS (grades of service)
  - Gradually being deployed
- See Figure 175

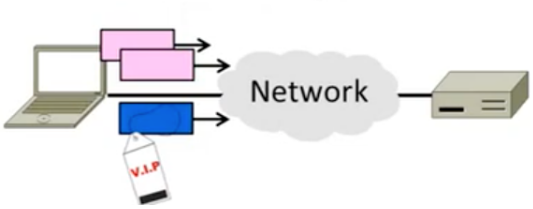


図 175: Topic of Differentiated Services

31.2 Motivation

- User runs Skype and BitTorrent
  - Or remote desktop, gaming, web, etc.
  - How can we give preference(優先) to flows?
- See Figure 176

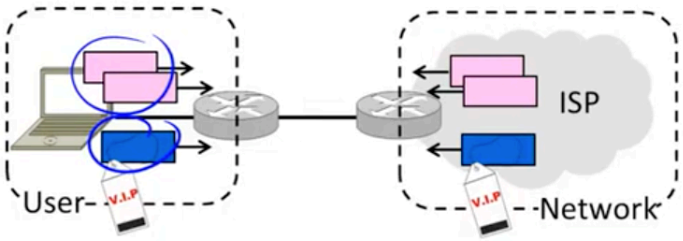


図 176: Motivation

31.3 Differentiated Services

- Idea is treat different kinds of traffic differently in the network
  - Have a few kinds of network service (GOLD, SILVER, BRONZE(青銅))
  - Different kinds get better or worse treatment in the network

  - Map apps to the right kind of service

- Architecture:
  - 1. User marks packet with desired service (e.g., Skype=GOLD)
  - 2. Network policies traffic levels at boundary(邊界) (token bucket)
  - 3. Network provides different forwarding (WFQ at routers)

- See Figure 177

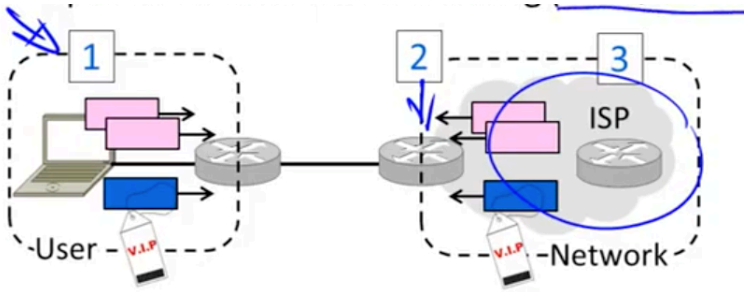


图 177: Differentiated Services

31.4 1. Marking packets

- Use bits in IPv4/IPv6 header to mark the kind of service
  - 6-bit DSCP (Differentiated Services Code Point)

- See Figure 178

- Many possible DSCP markings for different service/apps
  - Supported services depends on configuration of network

- See Figure 179

- Traffic is marked by user

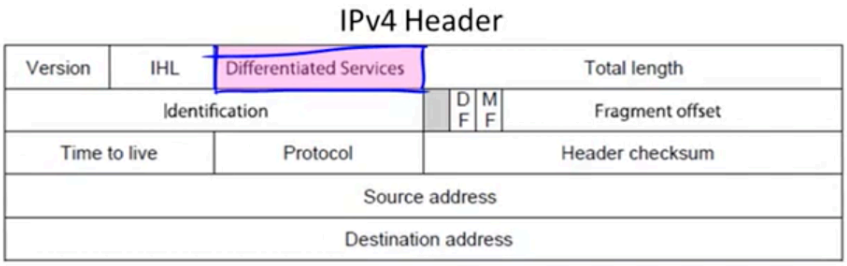


图 178: 1. Marking packets 图一

Service Name / Meaning	DSCP Value	Traffic Need (App example)
Default forwarding / Best effort	0	Elastic (BitTorrent)
Assured forwarding / Enhanced effort	10-38	Average rate (streaming video)
Expedited forwarding / Real-time	46	Low loss/delay (VoIP, gaming)
Precedence / e.g., Network control	48	High priority (Routing protocol)

图 179: 1. Marking packets 图二

  - Depends on local policies, e.g., gaming=expedited(加快)?

- May be done as part of host
  - Let OS or app classify their traffic
- May be done inside the network
- Using heuristic(启发式的), such as ports

31.5 2. Policing packets

- Network (ISP) checks incoming traffic meets service contract

  - Not more expedited traffic than agreed (and pard for!)
  - Only allowed markings, e.g., no network control from users
- Policing is done with token bucket
  - Can demote ”out of profile”(不在个人资料中) traffic by re-marking (e.g., default/best effort) or prioritizing for loss
- See Figure 180

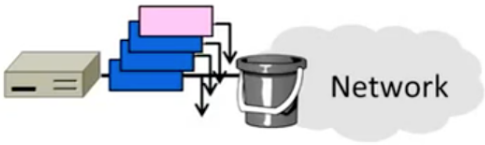


图 180: 2. Policing packets

31.6 3. Forwarding packets

- Network (ISP) routers use WFQ (and more) instead of FIFO
- The different kinds of service are the different flows/queues
- DSCP values are used to map packet to the right flow/queue
- Services are defined as ”per hop behaviors”
  - No guarantee for end-to-end service through a network
  - Need small amounts of high priority traffic for good service
- See Figure 181

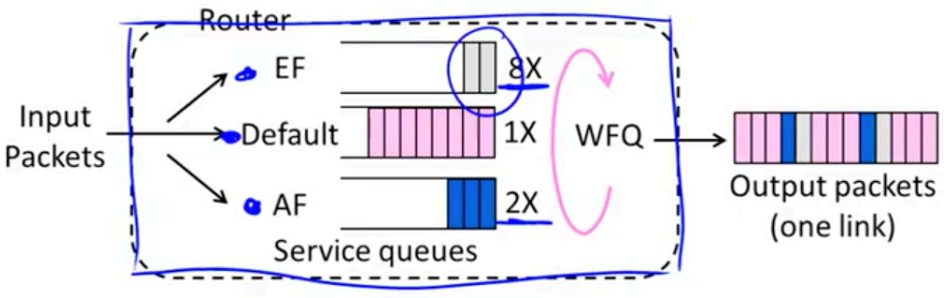


图 181: 3. Forwarding packets

31.7 Deployment

- QOS provides value when it is deployed across the network
  - Not much use if only(要是) your ISP!
- QOS is tightly tied(紧紧地绑定) to pricing
  - ”All my packets are high priority”
- 价格贵这个因素 Makes deployment slow/difficult ...

32 Week 9.7 - Rate and Delay Guarantees

32.1 Topic of Rate and Delay Guarantees

- Guaranteeing performance for traffic flows across in the network
  - This is ”hard QOS” with a firm guarantee for traffic flow
- See Figure 182

32.2 Motivation

- Sometimes we want guaranteed service - like the telephone network

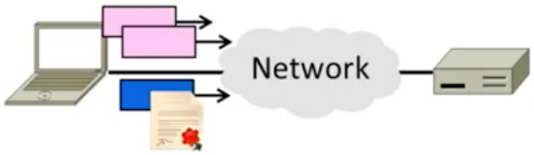


図 182: Topic of Rate and Delay Guarantees

  - Minimum rate and maximum delay regardless of how other flows behave(表現)
  - e.g., robotic control ?

- See Figure 183

図 183: Motivation

- Could provision(提供) a dedicated(専用的) circuit (or build a network), but expensive
- Can we have statistical multiplexing together with hard guarantees?

32.3 Admission(准許進入) Control

- Suppose we hav a flow F that needs $rate \geqslant RMbps$ and $delay \leqslant Dsecs$
- We must decide whether to admit or regect it from the network
  - This is admission control
  - Rejecting should be infrequent
- Key point is we need the ability to control load to make guarantees

32.4 Router Rate Guarantee

- WFQ can guarantee rate at a router
  - What rate will Flow F get?
- See Figure 184

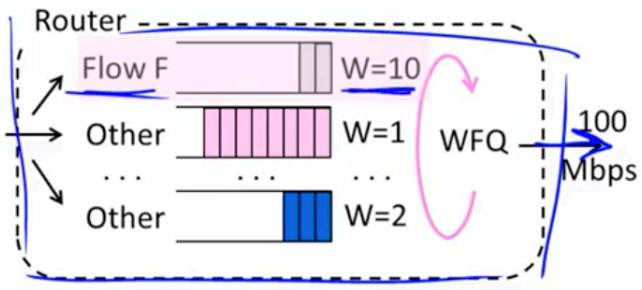


图 184: Router Rate Guarantee 图一

- Consider N flows with weight 1
  - Each flow gets 1/Nth share under load
  - Or at least 100/N Mbps
- Consider flow F with weight 10
  - Suppose weight of all flows is 100
  - Flow F gets $\geqslant \frac{10}{100} \times 100 = 10Mbps$
- See Figure 185

32.5 Network Rate Guarantee

- We can guarantee a minimum rate for a network path by Guaranteeing it at each router

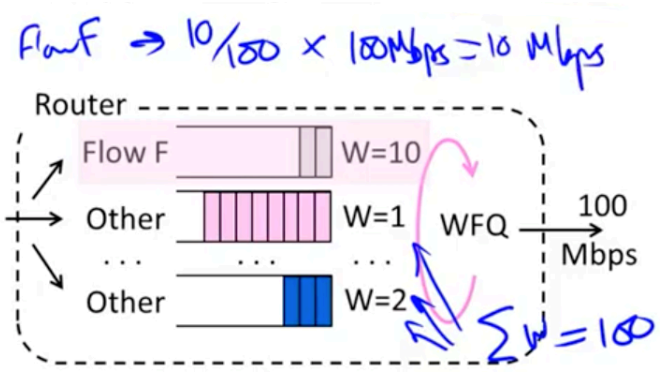


图 185: Router Rate Guarantee 图二

- See Figure 186

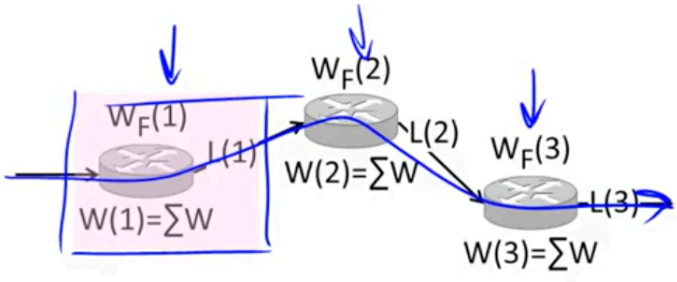


图 186: Network Rate Guarantee

- condition(使習慣於) for each router

$$
\begin{aligned}
&Forallroutersi: \\
&\frac{W_F(i)}{W(i)} \times L(i) \geqslant RMbps \\
&(Lforoutputlink)
\end{aligned}
\tag{2}
$$

32.6 Delay Guarantee

- What about the queuing delay?

  - How much larger than latency might the delay be, given rate guarantee?

- It depends on the traffic flow
  - If exceeds(超过) R Mbps then queues may build and delay will grow ...
- Need to shape traffic for guarantee
  - We'll use token buckets

32.7 Router Delay Guarantee

- Assume traffic flow F is shaped by an (R, B) token bucket
  - Long-term $rate \leqslant RMbps$
  - Short-term $rate \leqslant Bbits$
- See Figure 187

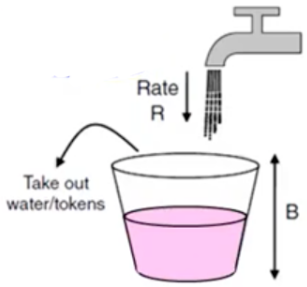


图 187: Router Delay Guarantee 图一

- What is delay of flow F at a router?
  - Traffic shaped by (R, B) token bucket

- WFQ with weight set for $rate \geqslant RMbps$

- See Figure 188

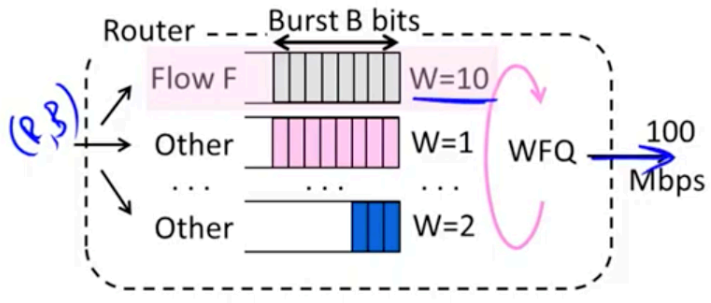


图 188: Router Delay Guarantee 图二

- In worst case B arrives all at once
  - So queuing delay is $\leqslant \frac{B}{R}$ seconds

32.8 Network Delay Guarantee

- What is the delay across N routers?
  - This is tricky! Each router add delays
  - Bound(極有可能的) of N*B/R is too loss
  - Intuitive(直覺的) argument follows ...
- If traffic is perfectly smooth at rate R (no bursts) then queuing delay is zero
  - Packet enters router just in time to leave
  - Delay is latency (propagation, Transmission)
- Observe if traffic pays for burst B at one router, it is smoothed for the next

- Burst delay is only paid once!

- See Figure 189

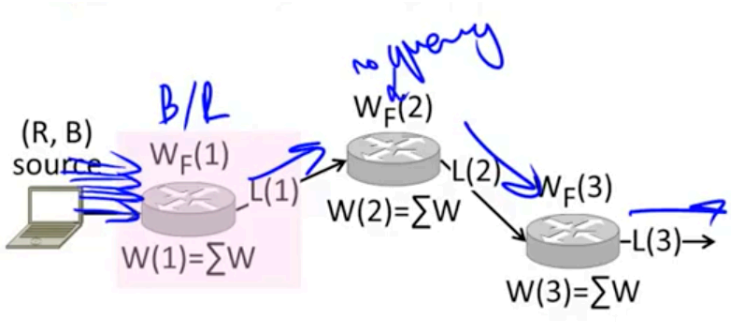


图 189: Network Delay Guarantee

- Delay across N routers: $Delay \leqslant Latencyterms + \frac{B}{R}$, 这里表示最多只存在一项 delay，从发送者角度，如果把水桶装满，最多需要花 $\frac{B}{R}$ 时间去抽干它。

32.9 Rate/Delay Guarantee

- Given a network with:
  - (R, B) shaped traffic flow
  - WFQ routers with proper weights
  - Sharing via statistical multiplexing(统计多路复用)
- We can guarantee the flow a minimum rate and maximum delay
  - Rate is $\geqslant RMbps$
  - Delay is $\leqslant latency + B/Rsecs$
  - regardless of how other flows behave

33 Week 10.1 - Network Security Overview

33.1 Where we are in the Course

- Revisiting the layers
  - Network security affects all layers because each layer may pose a risk

33.2 Topic of Network Security Overview

- Network security designs to protect against a variety of threats(多种的威胁)
  - Often build on cryptography
  - Just a brief overview. Take a course!
- See Figure 190

图 190: Topic of Network Security Overview

33.3 Security Threats

- "Security" is like "performance"
  - Means many things to many people
  - Must define the Properties we want
- Key part of network security is clearly stating(述説) the threat model

  - The dangers and attacker's abilities
  - Can't assess(评估) risk otherwise

- Some example threats
  - It's not all about encrypting messages

- See Table 1

attacker	**ability**	**threat**
Eavesdropper(窃听者)	Intercept messages	Read contents of message
Intruder(入侵者)	Compromised(損害) host	Tamper(篡改) with contents of message
Impersonator(假冒者)	Remote social engineering	Trick party into(进…里面) giving information
Extortionist(勒索者)	Remote / botnet(僵尸网络)	Disrupt network services

表 1: Security Threats

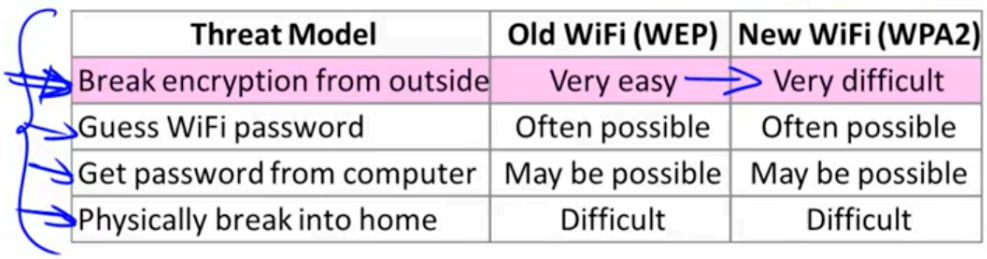
33.4 Risk Management

- Security is hard as negative goal, 想要阻止什么坏事发生是比较难的。
  - Try to ensure security proerties that don't let anything bad happen!

- Only as secure as the weakest link
  - Could be design flaw(缺陷) or bug in code

  - But often the weak link is elsewhere

- 802.11 security ... early on, WEP:
  - cryptography was flawed; can run cracking(开裂) software to read WiFi traffic
- Today, WPA2/908.11i security:
  - computationally infeasible(行不通的) to break!
- So that means 802.11 is secure against eavesdropping(偷聽)?
- Many possible threats
  - We just made the first one harder!
  - 802.11 is more secure against eavesdropping in that risk of successful attack is lower. But it is not "secure"
- See Figure 191



Threat Model	Old WiFi (WEP)	New WiFi (WPA2)
Break encryption from outside	Very easy	Very difficult
Guess WiFi password	Often possible	Often possible
Get password from computer	May be possible	May be possible
Physically break into home	Difficult	Difficult

图 191: Risk Management

33.5 Cryptology

- Rich history, especially spies / military
  - From the Greek "hidden writing"

- cryptography
  - Focus is encrypting information
- Cyptanalysis
  - Focus is how to break codes
- Modern emphasis(重視) is on codes that are ”computationally infeasible” to break
  - Takes too long compute solution

33.6 Uses of cryptography

- Encrypting information is useful for more than deterring(阻撓) eavesdroppers
  - Prove message came from real sender
  - Prove remote party is who they say
  - Prove message hasn't been altered
- Designing a secure cryptographic scheme(方案) is full of pitfalls(隱患; 陷阱)!
  - Use approved(認可的) design in approved way

33.7 Internet Reality

- Most of the protocols were Developed before the Internet grew popular
  - It was a smaller, more trusted world
  - So protocols lacked security ...

- We have strong security needs today
  - Clients talk with unverified servers
  - Servers talk with anonymous clients
  - Security has been retrofitted(加装)
  - This is far from ideal!

33.8 Topics

- —-本次所讲的内容—-
- Threat models
- —-Crypto—-
- Confidentiality
- Authentication
- —-Applied crypto—-
- Wireless security (802.11)
- Web security (HTTPS/SSL)
- DNS security
- Virtual Private Networks (VPNs)
- —-Connectivity—-
- firewalls
- Distributed denial-of-service

34 Week 10.2 - Message Confidentiality(保密)

34.1 Topic of Message Confidentiality(保密)

- Encrypting information to provide confidentiality
  - symmetric and public key encryption
  - Treat crypto functions as block boxes
- See Figure 192

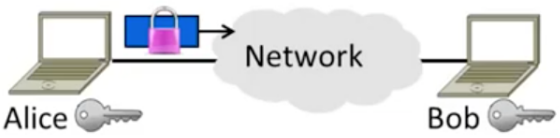


图 192: Topic of Message Confidentiality(保密)

34.2 Goal and Threat Model

- Goal is to send a private message from Alice to bob
  - This is called confidentiality
- Threat is Eve will read the message
  - Eva is a passive(被动) adversary(对抗者) (observes)
- See Figure 193

34.3 Encryption/Decryption Model

- Alice encrypts private message (plaintext(纯文本)) using key

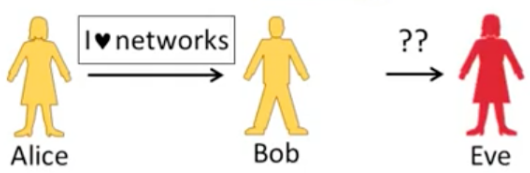


图 193: Goal and Threat Model

- Eve sees ciphertext(密文) but can't relate(关联) it to private message
- Bob decrypts using key to obtain the private message
- See Figure 194

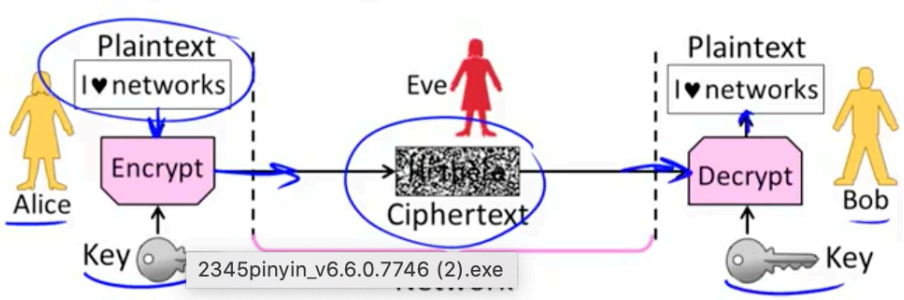


图 194: Encryption/Decryption Model

- Encryption is a reversible mapping
  - ciphertext is confused(乱糟糟) plaintext
- Assume attacker knows algorithm
  - Security does not rely on its secrecy
- Algorithm is parameterized by keys
  - Security does rely on key secrecy

- Must be distributed (Achilles' heel(阿喀琉斯的[illegible]踵，致命弱點))

- Two main kinds of encryption:
- 1. symmetric key encryption, e.g., AES
  - Alice and Bob share secret key
  - Encryption is a bit mangling(絞壞；撕爛) box
- 2. Pablic key encryption, e.g., RSA
  - Alice and Bob each have a key in two parts: a public part (widely known), and a private part (only owner knows)
  - Encryption is based on mathematics (e.g., RSA) is based on difficulty of factoring)

34.4 Symmetric (Secret key) Encryption

- Alice and Bob have the same secret key, K_{AB}
  - Anyone with the secret key can encrypt/decrypt
- See Figure 195

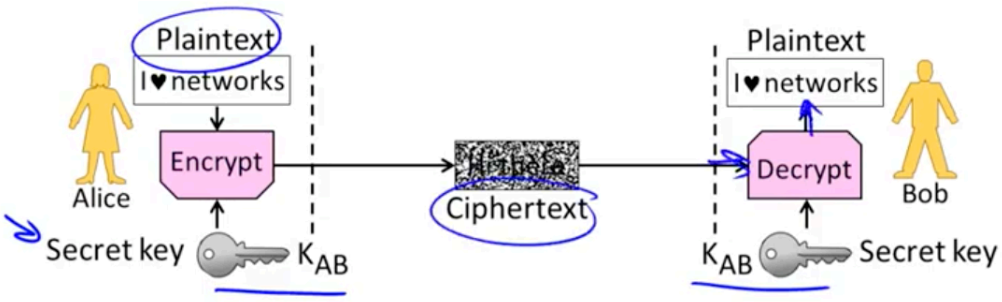


图 195: Symmetric (Secret key) Encryption

34.5 Public Key (Asymmetric) Encryption

- Alice and Bob each have public/private key pair ($K_B/{K_B}^{-1}$)
  - Public keys are well-known, private keys are secret to owner
- See Figure 196

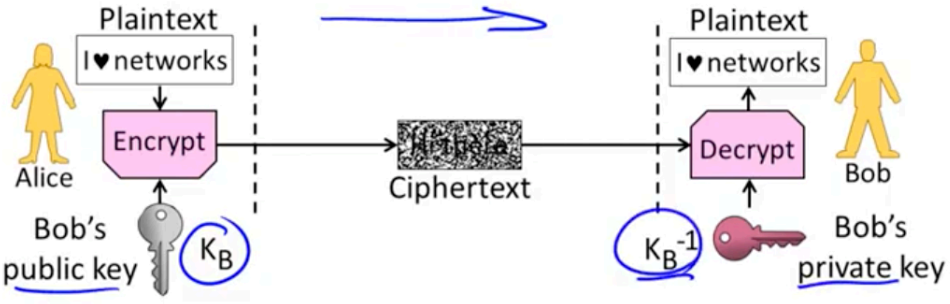


图 196: Public Key (Asymmetric) Encryption

- Alice encrypts with Bob's public Key K_B; anyone can send
- Bob decrypts with his private key ${K_B}^{-1}$; only he can do so

34.6 Key Distribution

- This is a big problem on a network!
  - Often want to talk to new parties
- symmetric encryption problematic
  - Have to first set up shared secret
- Public key idea has own difficulties
  - Need trusted directory service
  - We'll look at certificates later

34.7 Symmetric vs. Public Key

- Have complementary(互補的) properties
- See Figure 197

Property	Symmetric	Public Key
Key Distribution	Hard – share secret per pair of users	Easier – publish public key per user
Runtime Performance	Fast – good for high data rate	Slow – few, small, messages

图 197: Symmetric vs. Public Key

34.8 Winning Combination

- Alice uses public key encryption to send Bob a small private message
  - It's a key! (Say 256 bits.)
- Alice and Bob send large messages with symmetric encryption
  - Using the key they now share
- The key is called a session key , 零时生成的密匙叫做会话密匙。
  - generated for short-term use

35 Week 10.3 - Message Authentication(认证方式)

35.1 Topic of Message Authentication

- Encrypting information to provide authenticity(真实性) (=correct sender) and integrity (=unaltered)
  - Confidentiality(保密) isn't enough

35.2 Goal and Threat Model

- Goal is to let Bob verify the message came from Alice and is unchanged
  - This is called integrity/authenticity
- Threat is Trudy will tamper(篡改) with messages
  - Trudy is an active adversary(对抗者) (interferes(干扰))
- See Figure 198

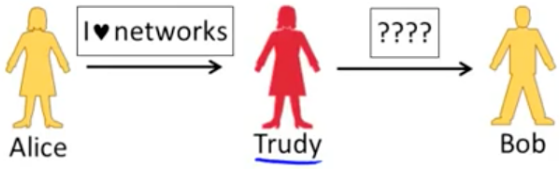


图 198: Goal and Threat Model

35.3 Encryption Issues

- What will hapen if Trudy flips some of Alice's message bits?

  - Bob will decrypt it, and Bob will receive an altered message

- Typically encrypt blocks of data
- What if Trudy reorders message?
  - Bob will decrypt, and Bob will receive altered message
- See Figure 199

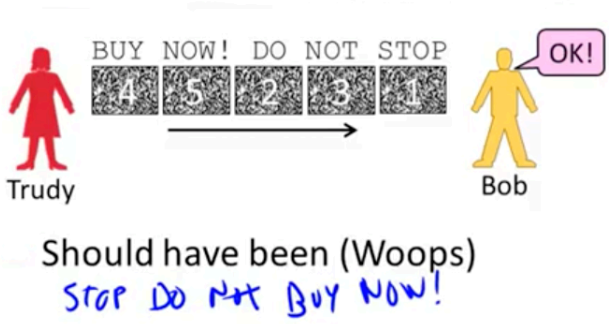


图 199: Encryption Issues

35.4 MAC (Message Authentication Code)

- MAC is a small token(类似于游戏机代币) to validate the integrity/authenticity of a message
  - Send the MAC along with message
  - Validate MAC, process the message
  - Example: HMAC scheme(方案)
- See Figure 200
- Kind of symmetric encryption operation - key is shared
  - Lets Bob validate unaltered message came from Alice

图 200: MAC (Message Authentication Code) 图一

  - Doesn't let Bob convince Charlie(说服) that Alice sent the message, 由于没有 Alice 的签名，所以 Bob 并不能向他人证明这是 Alice 发送的消息。

- See Figure 201

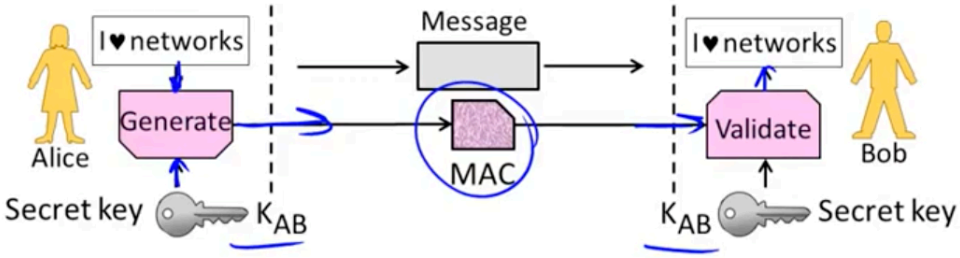


图 201: MAC (Message Authentication Code) 图二

35.5 Digital Signature

- Signature validates the integrity/authenticity of a message
  - Send it along with the message
  - Lets all parties validate
  - Example: RSA signatures
- See Figure 202

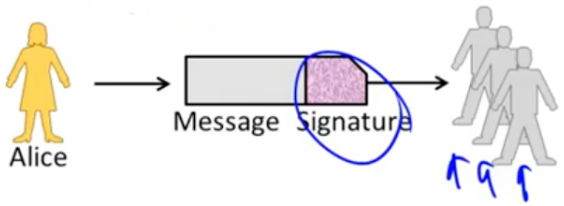


图 202: Digital Signature 图一

- Kind of public key operation - public/private key parts
  - Alice signs with private key, $K_A{}^{-1}$, Bob verifies with public key, K_A
  - Does let Bob convince Charlie that Alice sent the message
- See Figure 203

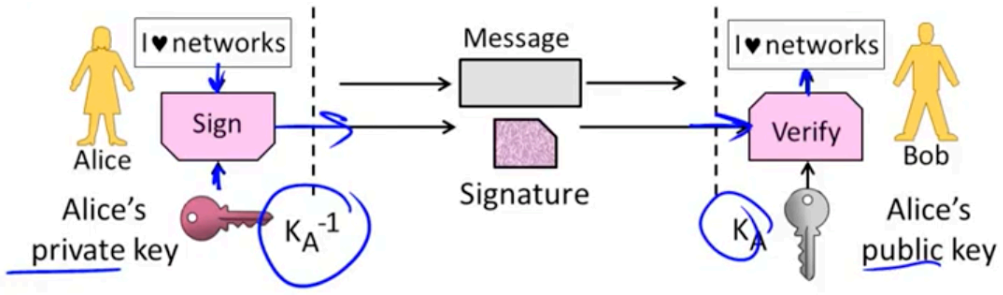


图 203: Digital Signature 图二

35.6 Message Digest or Cryptographic Hash

- Digest/Hash is a secure checksum
  - Deterministically(确定性地) mangles(撕裂) bits to pseudo-random(伪随机) output (like CRC)
  - Can't find messages with same Hash

- Acts as a fixed-length descriptor of messages - very useful!

- See Figure 204

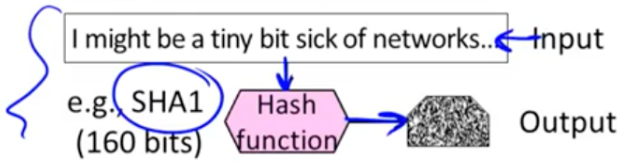


图 204: Message Digest or Cryptographic Hash

35.7 Speeding up Signatures

- Same tension(悬念) as for confidentiality:
  - Public key has keying advantages
  - But it has slow performance
- Use a technique to speed it up
  - Message digest(訊息摘要) stands for message
  - Sign the digest instead fo full message
- Conceptually(从概念上讲) as before except sign the hash of message
  - Hash is fast to compute, so it speeds up overall operation
  - Hash stands for message as can't find another with same hash
- See Figure 205

35.8 Preventing(预防) Replays

- We normally want more then confidentiality, integrity, and authenticity for secure messages!

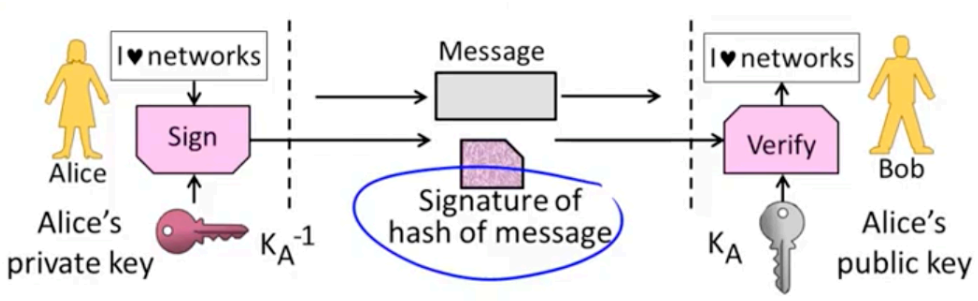


图 205: Speeding up Signatures

  - Want to be sure message is fresh

- Don't want to mistake old message for new one - a replay
  - Acting on it again may cause trouble
- Replay attack:
  - Trudy records Alice's messages to Bob
  - Trudy later replays them (unread) to Bob; she pretends to be Alice
- To prevent replays, include proof of freshness in messages
  - Use a timestamp, or nonce
- See Figure 206

35.9 Takeaway(带走)

- Cryptographic designs can give us integrity, authenticity and freshness as well as confidentiality. Yay!
- Real protocol designs combine the properties in different ways

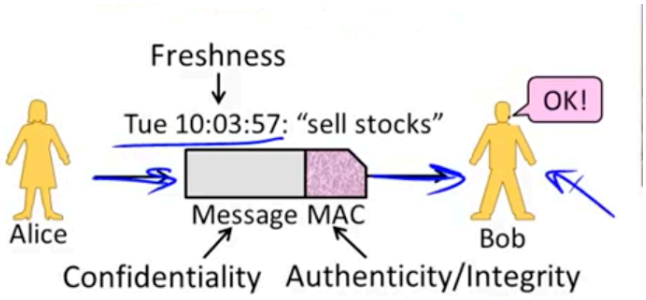


图 206: Preventing(预防) Replays

- We'll see some examples
- Note many pitfalls(陷阱) in how to combine, as well as in the primitives(原始的) themselves. 这段话的意思是，他们的组合版本和他们每个单独的版本都可能存在缺陷，使用被证实有效的加密方法。

36 Week 10.4 - Wireless Security

36.1 Topic of Wireless Security

- Securing wireless networks
  - Focus on 802.11
- See Figure 207

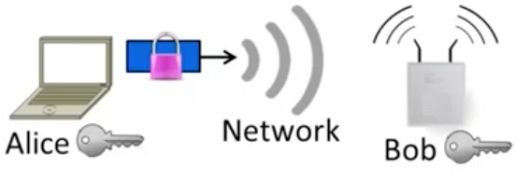


图 207: Topic of Wireless Security

36.2 Goal and Threat Model

- Unlike wired, wireless messages are broadcast to all nearby receivers
  - Don't need physical network access
  - Heightens(加高) security problems
- See Figure 208

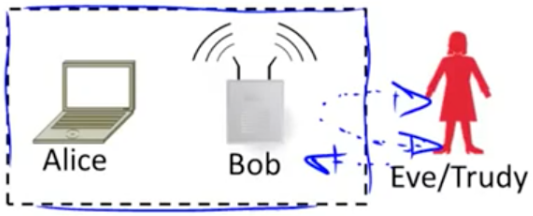


图 208: Goal and Threat Model

- Two main threats:
  - eavesdropping(偷聽) on conversations
  - Unauthorized access to network
- We'll consider 802.11 setting
  - Assume external attacker can send/receive wireless messages

36.3 802.11 Security

- Security is based on passwords
  - For access control and confidentiality and integrity/authenticity
- 802.11 standard (1999) used WEP
  - For "Wired Equivalent Privacy"

  - Badly flawed(帯缺陷的), easily broken

- 802.11i standard in 2004
  - WiFi Protected Access or WPA
  - This is what you should use
- Security is part of 802.11 protocol
  - Encrypted message between client and AP; removed after AP
- See Figure 209

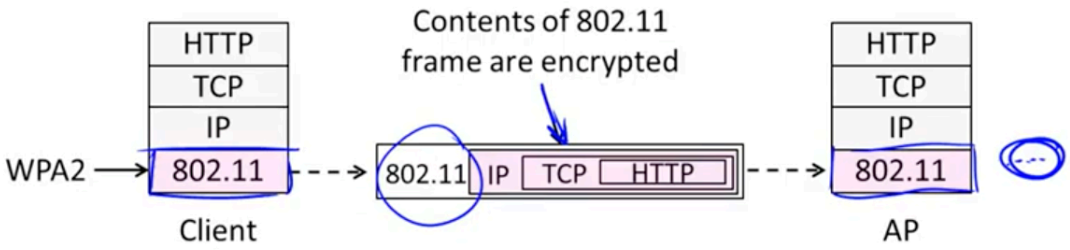


图 209: 802.11 Security

36.4 Home Network

- AP is set up with network password
- Each client also knows password
- Client proves it knows password
  - AP grants(准予) network access if successful
- See Figure 210
- For access, client Authenticates(確認…的真實性) to AP

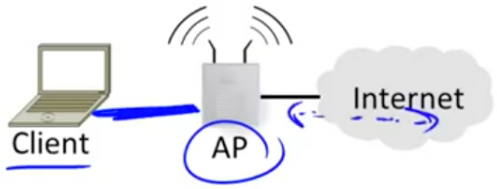


图 210: Home Network 图一

  - Both compute a shared session key based on the password
  - If client knows the session key it has proved that is has the password

- For usage, client/AP encrypt messages
  - For confidentiality, integrity/authenticity
  - No access without the session key
  - Also group key for AP to reach all clients, 这里的 group key 是 AP 广播消息到所有的终端。

- Master key(从 WiFi 密码算出的值) is from password; nonces(随机数 (计算机网络科学用语))(用来验证这个终端或 AP 是活着的) for freshness
  - K_S lets client talk to AP; K_G lets AP talk to all clients

- 在密码学中，消息认证码（英语：Message authentication code，缩写为 MAC），又译为消息鉴别码、文件消息认证码、讯息鉴别码、信息认证码，是经过特定算法后产生的一小段信息，检查某段消息的完整性，以及作身份验证。它可以用来检查在消息传递过程中，其内容是否被更改过，不管更改的原因是来自意外或是蓄意攻击。同时可以作为消息来源的身份验证，确认消息的来源。

- 互相发送 nonce 表示不信任对方是活着的。然后 AP 分发 K_S 给所有的终端，如果终端可以看到正确的 K_S，则认为它和一个无误的 AP 交流，然后发送“知道了”给 AP，如果 AP 解密后发现是终端发来的“知道了”，那一切都正常。然后就可以相互之间发送消息了。
- See Figure 211

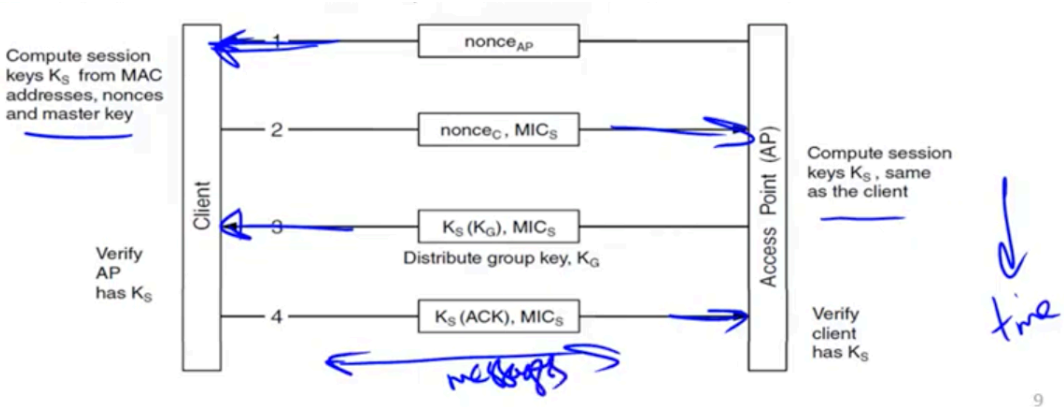


图 211: Home Network 图二

36.5 Enterprise Network

- Network has Authentication(身份验证) server
- Each client has own credentials(資格證書)
- AP lets client talk to auth. server
  - Grants network access if successful
- See Figure 212

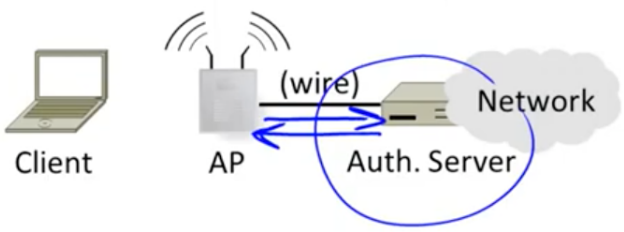


图 212: Enterprise Network

37 Week 10.5 - Web Security

37.1 Topic of Web Security

- Securing the web
  - Focus on SSL/TLS for HTTPS
  - Including certificates

37.2 Goal and Threat Model

- Much can go wrong on the web!
  - Clients encounter(遭遇) malicious(怀有恶意) content
  - Web servers are target of break-ins(入室盗窃)
  - Fake content/servers trick(欺诈) users
  - Data sent over network is stolen ...
- Goal of HTTPS is to secure HTTP
- We focus on network threats:
  - 1. Eavesdropping(窃听) client/server traffic
  - 2. Tampering(篡改) with client/server traffic
  - 3. Impersonating(冒充) web servers

37.3 HTTPS Context

- HTTPS (HTTP Secure) is an add-on
  - Means HTTP over SSL/TLS
  - SSL (Secure Sockets Layer) precedes TLS (Transport Layer security)
- See Figure 213

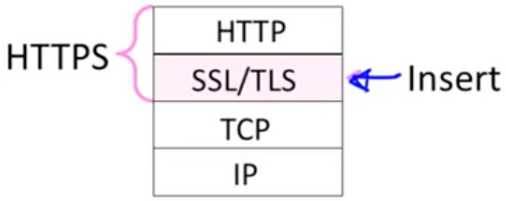


图 213: HTTPS Context

- SSL came out of Netscape
  - SSL2 (flawed(有缺陷的)) made public in 1995
  - SSL3 fixed flaws in 1996
- TLS in the open standard
  - TLS 1.0 in 1999, 1.1 in 2006, 1.2 in 2008
- Motivated by secure web commerce(贸易)
  - Slow adoption(采用), now widespread use
  - Can be used by any app, not just HTTP

37.4 SSL Operation

- Protocol provides:
  - 1. Verification of identity of server (and optionally client)
  - 2. Message exchange between the two with confidentiality, integrity, authenticity and freshness
- Consists of authentication(身份验证) phase (that sets up encryption) followed by data transfer phase

37.5 SSL/TLS Authentication(身份验证)

- Must allow clients to securely connect to servers not used before
  - Client must Authenticate server
  - Server typically doesn't identify client
- Uses public key authentication
  - But how does client get server's key?
  - With certificates(证书, 凭证)

37.6 Certificates(证书, 凭证)

- A certificate binds public key to an identity(身份), e.g., domain
  - Distributes public keys when signed by a party you trust
  - Commonly in a format called X.509
- See Figure 214

图 214: Certificates(证书, 凭证)

37.7 PKI (Public Key Infrastructure)

- Adds hierarchy to certificates to let many parties(各方) issue(分发)
  - Issuing(分发) parties are called CAs (Certificate Authorities(官方))
- See Figure 215

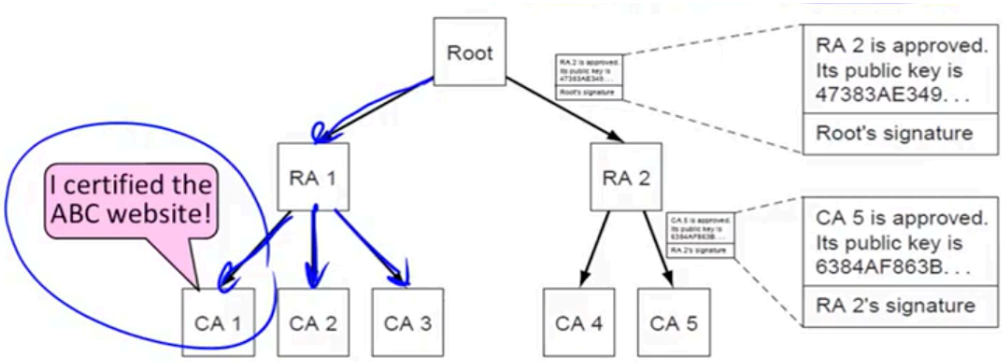


图 215: PKI (Public Key Infrastructure) 图一

- Need public key of PKI root and trust in servers on path to verify a public key of website ABC, 逐级保证。
  - Browser has Root's public key

  - {RA1's key is X} signed Root
  - {CA1's key is Y} signed RA1
  - {ABC's key is Z} signed CA1
- Browser/OS has public keys of the trusted roots of PKI
  - > 100 root certificates!
  - That's a problem ...
  - Inspect(查看) your web browser
- See Figure 216

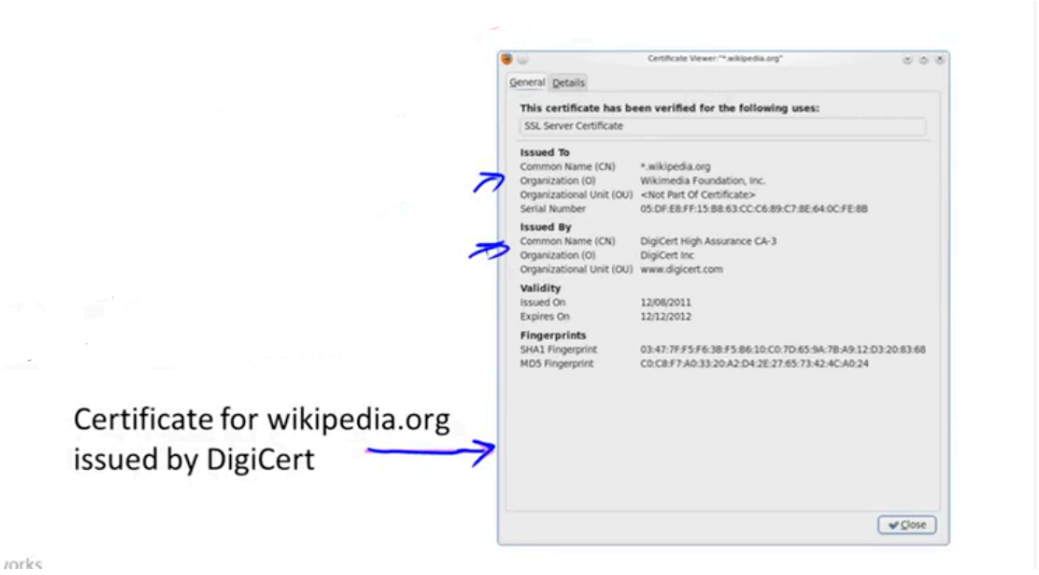


图 216: PKI (Public Key Infrastructure) 图二

- Real-world complication(使更复杂化（或更困难）的事物):
  - Public keys may be compromised(放弃)
  - Certificates must then be revoked(被调销)

- PKI includes a CRL (Certificate Revocation(撤消) List)
  - Browsers use to weed(除（地面的）杂草) out bad keys

37.8 SSL3 Authentication

- 下面的图老师并没有细说，这是一个稍微复杂一点的过程。后面自己可以研究一下，老师只是顺便过一下。
- See Figure 217

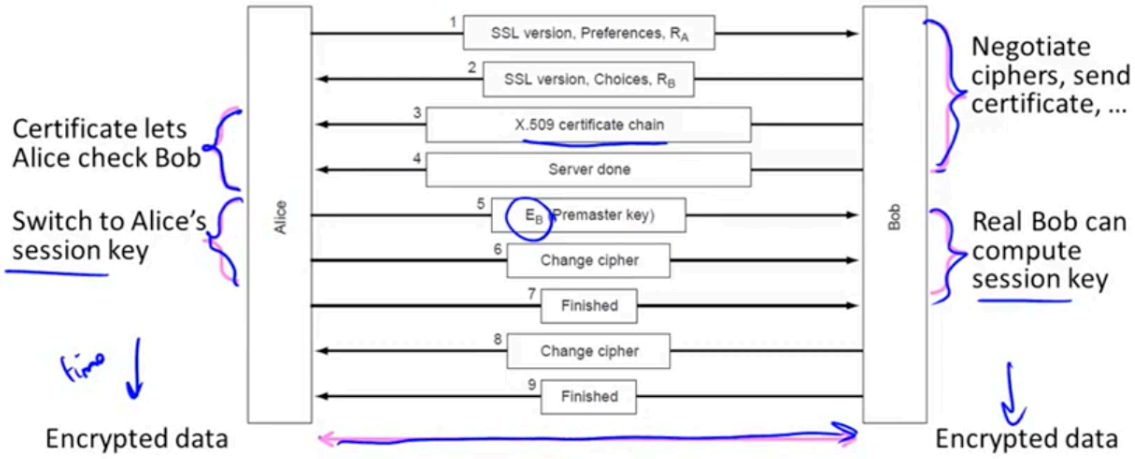


图 217: SSL3 Authentication

37.9 Takeaways

- SSL/TLS is a secure transport
  - For HTTPS and more, with the usual confidentiality, integrity, authenticity
  - Very widely used today

- Client Authenticates web server
  - Done with PKI and certificates
  - Major area of complexity and risk

38 Week 10.6 - DNS Security

38.1 Topic of DNS Security

- Securing Internet naming
  - DNS security Extensions (DNSSEC)
- See Figure 218

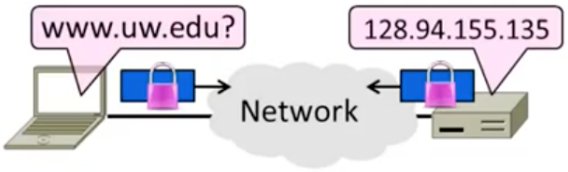


图 218: Topic of DNS Security

38.2 Goal and Threat Model

- Naming is a crucial Internet service
  - Binds host name to IP address
  - Wrong binding can be disastrous
- See Figure 219
- Goal is to secure the DNS so that the returned binding is correct

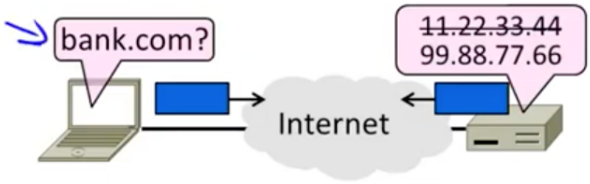


图 219: Goal and Threat Model

  - Integrity/authenticity vs. confidentiality

- Attacker can intercept(拦截)/tamper(篡改) with message on the network

38.3 DNS Spoofing(欺骗攻击)

- Hang on - how can a network attacker corrupt(破坏) the DNS?
- Trudy can trick(欺诈) a nameserver into caching the wrong binding
  - By using the DNS protocol itself
  - This is called DNS spoofing
- To spoof, Trudy returns a fake DNS response that appears to be true
  - Fake response contains bad binding
- See Figure 220
- Lots of questions!
  - 1. How does Trudy know when the DNS query is sent and what it is for?
  - 2. How can Trudy supply a fake DNS reply that appears to be real?

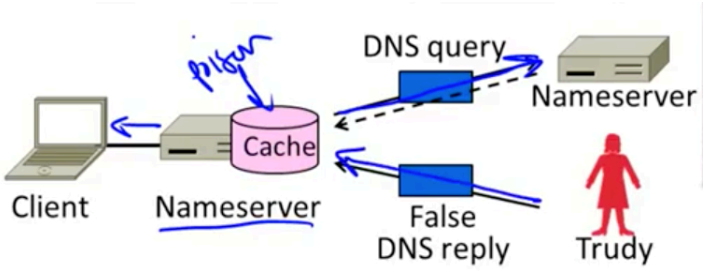


图 220: DNS Spoofing(欺骗攻击)

  - 3. What happens when the real DNS reply shows up?

- There are solutions to each issue …
- 1. How does Trudy know when the query is sent and what it is for?
- Trudy can make the query(询问) herself! 这些本地 DNS 不会限制本地服务器访问它，所以可以利用这一点来访问本地 DNS。
  - Nameserver works for many clients
  - Trudy is just another client
- 2. How can Trudy supply a fake DNS reply that appears to be real?
- A bit more difficult. DNS checks:
  - Reply is from authoritative(权威性的) nameserver (e.g., .com)
  - Reply ID that matches the request
  - Reply is for outstanding(尚待解决的) query
- (Nothing about content though ...)
- Techniques:
  - Put IP of authoritative(权威性的) nameserver as the source IP address

  - ID is 16 bits (64K). Send many guesses! (Or if a counter, sample(尝试) to predict.)
  - Send reply right after query
- Good chance of succeeding!
- 3. What happens when the real DNS reply shows up?
- Likely not be a problem
  - There is no outstanding query after fake reply is accepted
  - So real reply will be discarded

38.4 DNSSEC (DNS Security Extensions)

- Extends DNS with new record types
  - RRSIG for digital signatures of records
  - DNSKEY for public keys for validation
  - DS for public keys for delegation(授权)
  - First version in 1997, revised(修正) by 2005
- Deployment requires software upgrade at both client and server
  - Root servers upgraded in 2010
  - followed by uptick(上涨) in deployment

38.5 DNSSEC - New Records

- As well as the usual A, NS records
- (A 记录：将域名指向一个 IPv4 地址（例如：100.100.100.100），需要增加 A 记录)

- (NS 记录：域名解析服务器记录，如果要将子域名指定某个域名服务器来解析，需要设置 NS 记录)
- RRSIG
  - Digital signatures of domain records
- DNSKEY
  - Public key used for domain RRSIGs
- DS
  - Public keys for delegated domain
- NSEC/NSEC3
  - Authenticated(认证) denial(拒绝，否定) of existence

38.6 DNSSEC - Validating Replies

- Clients query DNS as usual, then validate replies to check that content is authentic(真正的)
- Trust anchor(锚点) is root public keys
  - Part of DNS client configuration
- Trust proceeds down DNS hierarchy
  - Similar concept to SSL certificates
- Client queries www.uw.edu as usual，这里不仅是 DNS 签名，而且必须是授权的签名。这两个要点保证了攻击者无法伪造。

  - Replies include signatures/keys, 如果是攻击者的话, 既没有签名, 也没有对应于签名的 key，这里涉及到 RRSIG 和 DNSKEY 两个物件。

- Client validates answer，这里用到了 DS，用以标定授权了的域名的 Key，DNSKEY 必须是 DS 众多个授权域名中的一个，下图中的所有的域都需要确认为授权的域名，这就杜绝了攻击者自己搞得域名:
  - 1. K_{ROOT} is a trust anchor
  - 2. Use K_{ROOT} to check K_{EDU}
  - 3. Use K_{EDU} to check $K_{UW.EDU}$
  - 4. Use $K_{UW.EDU}$ to check IP
- See Figure 221

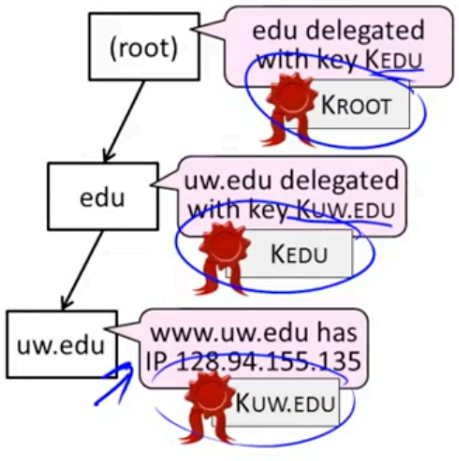


图 221: DNSSEC - Validating Replies

- Other features too:
  - authoritative(权威性的) answers a domain record doesn't exist (NSEC/NSEC3)，提供一个反例库。

  - Optional anti-spoofing to bind query and reply，老师没细讲
  - Flags related to deployment，老师没细讲...

38.7 Takeaways

- DNS spoofing is possible without added security measures
  - Large problem in practice!
- DNSSEC adds authentication (only) of replies to the DNS
  - Using a hierarchy of public keys

39 Week 10.7 - Firewalls

39.1 Topic of Firewalls

- Firewalls
  - Protecting hosts by restricting(限制) network connectivity
- See Figure 222

图 222: Topic of Firewalls

39.2 Motivation

- The best part of IP connectivity
  - You can send to any other host
- The worst part of IP connectivity
  - Any host can send packets to you!
  - There's nasty(令人厌恶的, 不友好的) stuff out there ...

40 Goal and Threat Model

- Goal of firewall is to implement a boundary to restrict(限制) IP connectivity:
  - You can talk to hosts as intended
  - Trudy can't talk to you over network
- See Figure 223

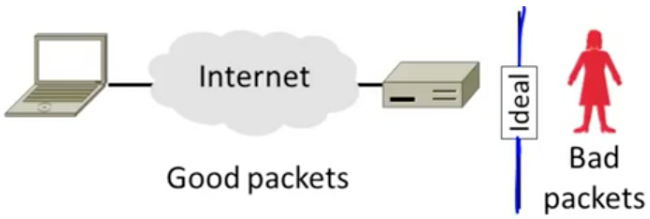


图 223: Goal and Threat Model

40.1 Recall Middleboxes

- Sit "inside the network" but perform "more then IP" processing on packets to add new functionality

  - NAT(Network Address Translation) box, Firewall/Intrusion(入侵) Detection System

- See Figure 224

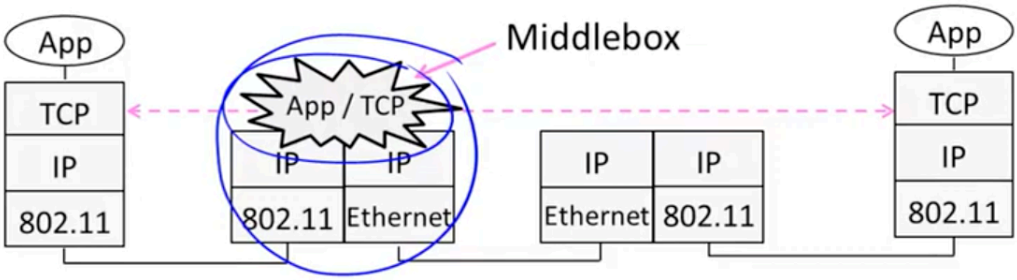


图 224: Goal and Threat Model

40.2 Firewall as Middlebox

- 用防火墙替代中间件。
- See Figure 225

40.3 Operation

- Firewall has two sides:
  - Internal (organization) and external (Internet)
- Fore each packet that tries to cross, decide whether to:
  - ACCEPT = pass unaltered(未经过篡改的); or DENY = discard silently
  - Decision is a local policy; firewall centralizes IT job
- See Figure 226

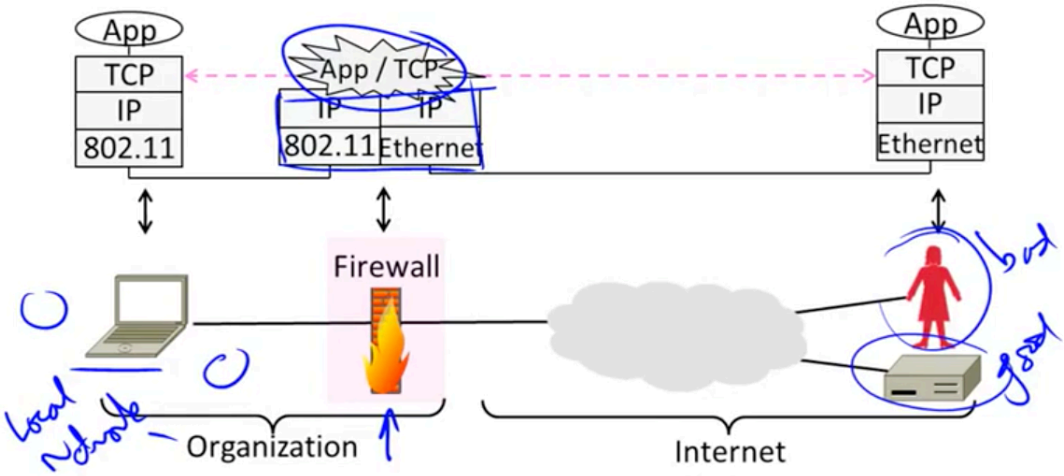


图 225: Firewall as Middlebox

图 226: Operation

40.4 Design

- Key tension(矛盾):
  - How to translate desired policies into packet filtering rules
- policies are high-level statements
  - Relate to usage of apps, content
- Packet filtering is low-level
  - Limited viewpoint(视角) in the network, e.g., no app messages,

encryption

- Stateless firewall
  - Simplest kind of firewall
  - Implements static packet filter rules
  - Typically using TCP/UDP ports
  - E.g., deny TCP port 23 (telnet)
  - Can allow/disallow many types of services and destinations
- Stateful firewall
  - A step up from stateless
  - Implements stateful packet filter rules that track packet exchanges
  - NAT example: accept incoming TCP packets after internal host connects, 内部机器主动连接外部服务器，NAT 才会接收来自该服务器的 TCP 包。
- Application layer firewall:
  - Another step up
  - Implements rules based on app usage and content
  - E.g., inspect(检查) content for viruses
  - Tries to look beyond packets by emulating(模拟) higher layers, e.g., by reassembling(重新装配) app messages

40.5 Deployment

- Firewall is placed around internal/external boundary

  - Classic setup includes DMZ(DeMilitarized(非军事化) Zone) to put busy Internet hosts on the outside for better separation

- See Figure 227

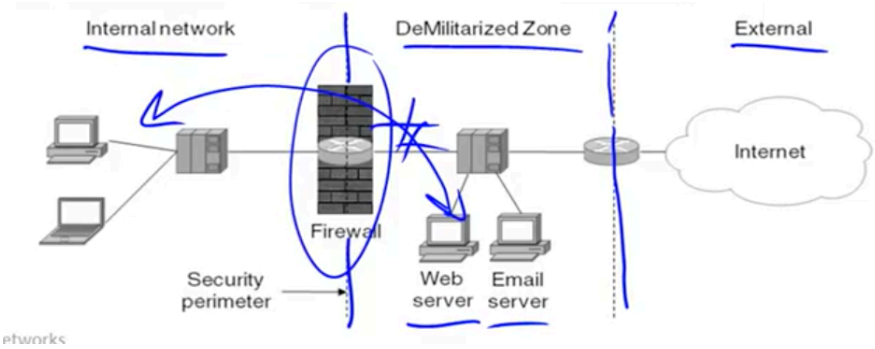


图 227: Operation

- Various device options:
  - Specialized(专用的) network firewall
  - Firewall in boundary device, e.g., AP
  - Firewall as part of host, e.g., in OS
- Tradeoff:
  - Centralizing simplifies IT job
  - Distributing improves protection, visibility into apps(應用程序可見性), and performance

41 Week 10.8 - Virtual Private Networks (VPNs)

41.1 Topic of VPN

- Virtual Private Networks (VNSs)

  - Run as cloesed networks on on Internet
  - Use IPSEC to secure messages

41.2 Motivation

- The best part of IP connectivity
  - You can send to any other host
- The worst part of IP connectivity
  - Any host can send packets to you!
  - There's nasty(令人厌恶的) stuff out there ...
- Often desirable to seperate network from the Internet, e.g., a company
  - Private network with leased(租借) lines
  - Physically separated from Internet
- See Figure 228

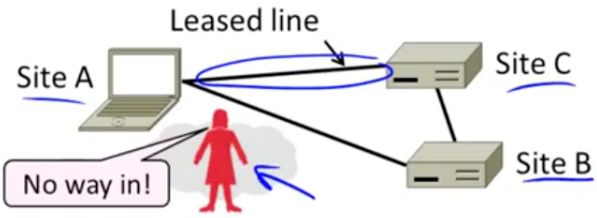


图 228: Motivation 图一

- Idea: Use the public Internet instead of leased lines - cheaper!
  - Logically separated from Internet ...
  - This is Virtual Private Network (VPN)
- See Figure 229

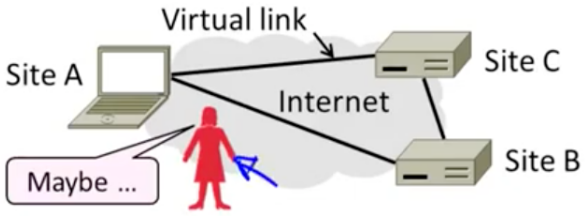


图 229: Motivation 图二

41.3 Goal and Threat Model

- Goal is to keep a logical network (VPN) separate from the Internet while using it for connectivity
  - Threat is Trudy may access VPN and intercept(侦听) or tamper(篡改) with messages
- See Figure 230

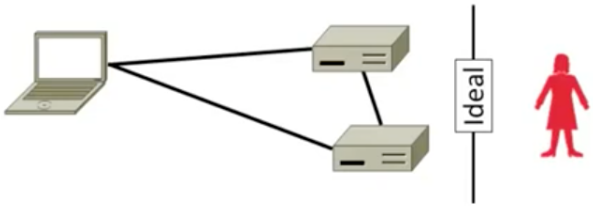


图 230: Goal and Threat Model

41.4 Tunneling(隧道)

- How can we build a virtual link? With tunneling!
  - Hosts in private network send to each other normally
  - To cross virtual link (tunnel), endpoints encapsulate(封装；装入胶囊；密封) packet

- See Figure 231

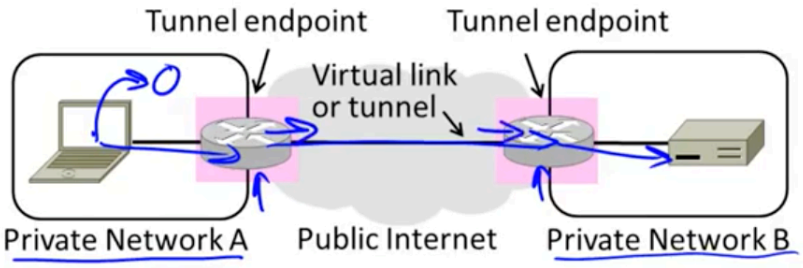


图 231: Tunneling(隧道) 图一

- Tunnel endpoints encapsulate IP packets ("IP in IP")
  - Add/modify outer IP header for delivery to remote endpoint
- See Figure 232

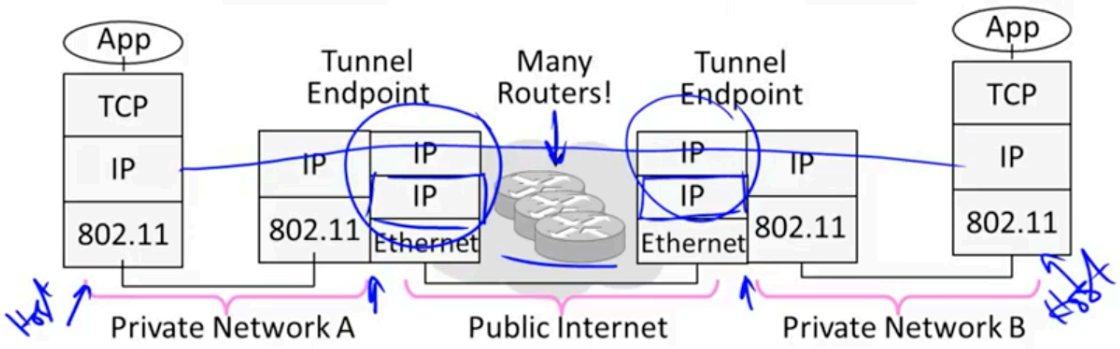


图 232: Tunneling(隧道) 图二

- Simplest encapsulation wraps(包裹) packet with another IP header
  - Outer (tunnel) IP header has tunnel endpoints as source/destination

  - Inner packets has private network IP addresses as source/destination

- See Figure 233

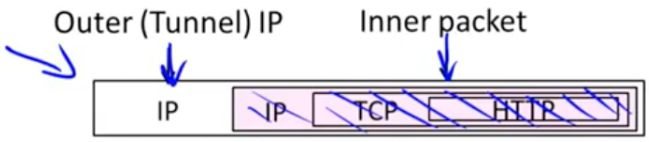


图 233: Tunneling(隧道) 图三

- Tunneling alone is not secure ...
  - No confidentiality, integrity, authenticity
  - Trudy can read, inject her own messages
  - We require cryptographic protections
- IPSEC (IP Security) is often used to secure VPN tunnels

41.5 IPSEC (IP Security)

- Longstanding(长期存在的) effort(努力) to secure the IP layer
  - Adds confidentiality, integrity/authenticity
- IPSEC operation:
  - Keys are set up for communicating host pairs
  - Communication become more connection-oriented(面向连接的)
  - Header and trailer added to protect IP packets
- See Figure 234

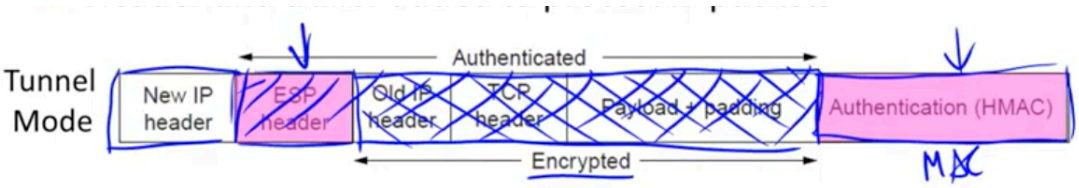


图 234: IPSEC (IP Security)

- (ESP: encapsulating Security Protocol, 和 HMAC 是 IPSEC 所要添加的内容, 主要功能是：加解密，签名什么的)

41.6 Takeaways

- VPNs are useful for building networks on top of Internet
  - Virtual links encapsulate packets
  - Alters(变更) IP Connectivity for hosts
- VPNs need crypto to secure messages
  - Typically IPSEC is used for confidentiality, integrity,authenticity

41.7 Week 10.9 - Distributed Denial of Servive (DDOS)

41.8 Topic of DDOS

- Distributed Denial-of-Service (DDOS)
  - An attack on network available

41.9 Motivation

- The best part of IP connectivity
  - You can send to any other host

- The worst part of IP connectivity
  - Any host can send packets to you!
- Flooding(淹没) a host with many packets can interfere(干扰) with its IP connectivity
  - Host may become unresponsive(反应迟钝的)
  - This is a form of denial-of-service(拒绝服务)
- See Figure 235

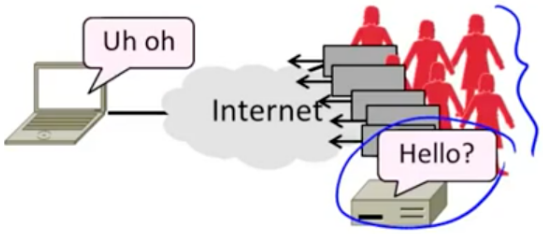


图 235: Motivation

41.10 Goal and Threat Model

- Goal is for host to keep network connectivity for desired services
  - Threat is Trudy may overwhelm(淹没) host with undesired traffic

41.11 Internet Reality

- Distributed Denial-of-Service is a huge problem today
  - Akamai(阿卡迈科技) Q3-12 reports DDOS against US banks peaking at 65 Gbps ...

- There are no great solutions
  - CDNs(Content Delivery Networks), network traffic filtering, and best practices all help

41.12 Denial-of-Service

- Denial-of-service means a system is made unavailable to intended users
  - Typically, because its resources are consumed by attackers instead
- In the network context:
  - "System" means server
  - "Resources" mean bandwidth (network) or CPU/memory (host)

41.13 Host Denial-of-Service

- Strange packets can sap(削弱) host resources!
  - "Ping of Death" malformed(畸形的) packet
  - "SYN flood" sends many TCP connect requests and never follows up
  - Few bad packets can overwhelm host
- Patches(补丁) exist for these vulnerabilities(漏洞)
  - Read about "SYN cookies" for interest

41.14 Network Denial-of-Service

- Network DOS needs many packets
  - To saturate(使充满) network links

- Causes high congestion/loss

- See Figure 236

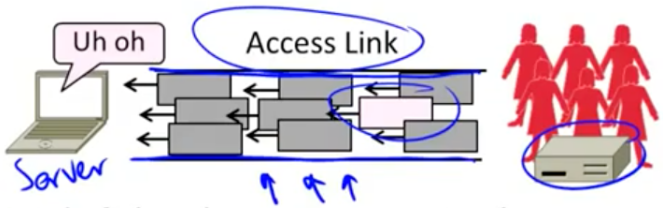


图 236: Network Denial-of-Service

- Helpful to have many attackers ... or Distributed Denial-of-Service

41.15 Distributed Denial-of-Service (DDOS)

- Botnet provides many attackers in the form of compromised(妥协的) hosts

  - Hosts send traffic flood to victim
  - Network saturates near victim

- See Figure 237

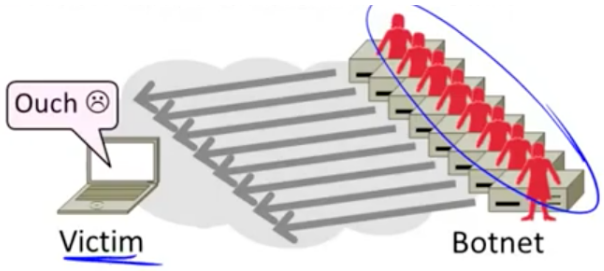


图 237: Distributed Denial-of-Service (DDOS)

41.16 Complication(混乱): Spoofing(欺骗)

- Attackers can falsify(篡改) their IP address
  - Put fake source address on packets
  - Historically network doesn't check
  - Hides location of the attackers
  - Called IP address spoofing
- See Figure 238

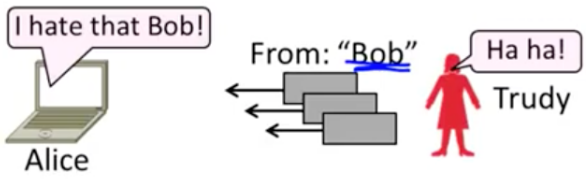


图 238: Complication(混乱): Spoofing(欺骗) 图一

- Actually, it's worse than that
  - Trudy can trick(欺诈) Bob into really sending packets to Alice
  - To do so, Trudy spoofs Alice to Bob
- See Figure 239

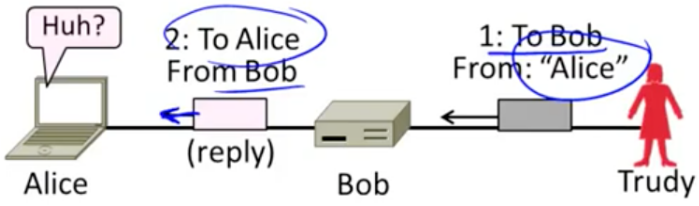


图 239: Complication(混乱): Spoofing(欺骗) 图二

41.17 Best Practice: Ingress Filtering

- Idea: Validate the IP source address of packets at ISP boundary (Duh!)
  - Ingress Filtering is a best practice, but deployment has been slow
- See Figure 240

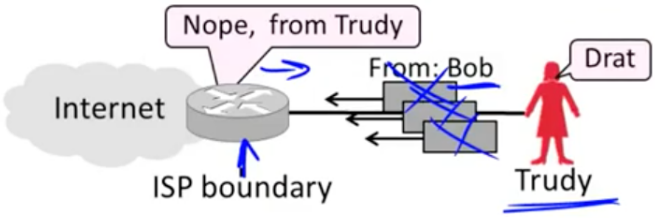


图 240: Best Practice: Ingress Filtering

41.18 Flooding Defenses

- 1. Increase network capacity around the server; harder to cause loss
  - Use a CDN for high peak capacity
- 2. Filter out attack traffic within the network (at routers)
  - The earlier the filtering, the better
  - Ultimately(最终) what is needed, but ad hoc measures(临时措施) by ISPs(Internet service providers) today

41.19 Week 10.10 - Farewell For Now (Last One)

- 完结撒花