CSC458 Aid Sheet / Jason Pham / 1003078144 / 09/12/19 2-4 WY119

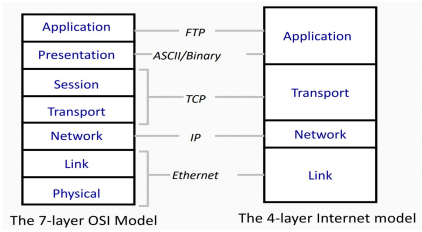
Internet Introduction

- Each packet is individually routed - No time guarantee for delivery - No guarantee of deliv. in sequence - No guarantee of deliv. at all - No guarantee of integrity - Packets can be fragmented/duplicated

Application Layer: Communication for specific apps; e.g HTTP.
Transport Layer: Provides reliable, in-sequence delivery of data from end-to-end on behalf of application. Communication between processes (e.g socket).
Network Layer: Provides "best-effort"(unreliable) delivery of datagrams. Logical communication between nodes. E.g IP
Link Layer: Carries data over point-to-point links between hosts and routers; or between routers and routers. E.g Ethernet
Sequence (FCS) Wait for access to the line. MAC requests PHY to send each bit of the frame. IP packet = Ethernet data
5. Rout1 accepts eth frame if destination = R1. Pass data to IP protocol
6. Use IP destination address to decide next hop, request link protocol to transmit packet
7. Create eth frame with FCS with dest set to next hop until final router
8. Final router accepts eth frame, checks IP header for dest address, if match, decapsulate TCP packet and pass to TCP protocol
9. Accept TCP "Connection setup" packet, establish connection by sending ACK.
10. Application receives request for TCP connection with "A"

In better terms

Layering



The 7-layer OSI Model — The 4-layer Internet model

Packet Switching and Circuit switching

- Each packet is routed packet-by-packet, using router's lookup table - Routers maintain no per-flow state - Different packets may take different paths - Several packets may arrive for the same output link at the same, therefore a packet switch has buffers.

Statistical Multiplexing: Network traffic rate changes frequently, so the more flows we have, the smoother the traffic. Because the buffer absorbs temp bursts, the egress link need not operate at rate N packets times R lines to link. But buffer has finite size so losses will occur.
SM Gain = 2C/R: The ratio of rates that give rise to a particular queue occupancy, or particular loss probability
Why packet switching:
Efficient use of expensive links: - Links are assumed to be expensive and scarce - Packet switching allows many bursty flows to share the same link efficiently - Circuit switching is very inefficient
Resilience to failure of links & routers: - If half the network went down, it's still possible for packets to be routed to destination

Physical/Link Layer

Wireless: Different freq have different prop (affected by atmospheric)
Fiber: Glass strand that propagates light (really high bandwidth)

Encoding Signals

NRZ encoding: high voltage = 1, low voltage = 0
NRZI: Invert signal on 1
Problem with NRZ(I): Hard to distinguish multiple of the same bit as clocks will become out of sync. Solution: send separate clock signal, keep messages short, embed clock signal in data signal
Manchester Coding: Make transitions in the middle of every bit period. Low to high: 0, High to low: 1. Signal rate is twice the bitrate
Advantage: Self clocking Disadvantage: 50% efficiency
4B/5B: Map data bits (which may lack transitions) into code bits 0000 -> 11110, 0001 -> 0100, …, 1111 -> 11101. 80% efficiency

Framing

Frames: Complete Link Layer messages
Sentinels: Special control code that marks boundaries of a frame a=bc | b = bd when for packet content contains sentinel code

Model of a link

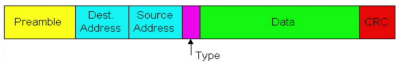
Bandwidth: info carrying capacity of channel. bits per second
Propagation delay = distance / speed of light in media
Transmission delay = message (bits) / rate (bps)
Latency: Propagation + Transmission + Queue (typically 1 way)
RTT: 2 * Latency
Throughput: Transfer Size / Transfer Time - Measure of system's ability to "pump out" data (NOT same as bandwidth)
Transfer Time: Time to get started shipping bits, Time to ship bits, Time to get stopped shipping the bits
Bandwidth-delay product: (i.e. measure of "data in flight" - Speed * latency (1 bps * 16 second delay = 16 bits)

Error Detection and Correction

(N, K) code: k bits in, N bits out -- simple memoryless mapping
Automatic Repeat reQuest (ARQ): Detect and retransmit
Forward Error Correction (FEC): Error correcting codes
Adv. to detection: Requires less bits/overhead | simpler processing
Adv. to error correct: Reduces number of retransmissions
Hamming Distance: Encoding 0 and 1 as 000 and 111. To detect *d* single bit errors, Hamming distance must be d+1. To correct 2d+1
1D Parity: Start with n bits and add another high bit so there are an even number of bits. Detects odd number of bit errors (does not correct)
2D Parity: Parity row/column, detects 1,2,3 bit errors, and many errors with >3 bits. Corrects all 1 bit errors
Checksums: Sum two 8 bits and take 1s complement, i.e 1s complement of the 1s complement sum of the data
Cyclic Redundancy Check (CRC): Given n bits of data, generate a k bit check sequence that gives a combined n+k bits that are divisible by a chosen divisor C(x). Message + Generator, divide right padded message by generator and append the remainder to message

Interconnecting LAN, IP

Carrier Sense: Wait for link to be idle to start transmitting
Collision Detection: Listen while transmitting, abort signal on collision, send jam signal, and start exponential backoff
Random Access: Exponential backoff
Limits on Ethernet Length: Needs to wait 2d to detect collision and needs to keep transmitting, max length is 2500m, min length of packet is 512 bits



Destination packet comes first for faster reading
Type: Indicates higher layer protocol, usually IP
Ethernet: Connectionless and unreliable
Ethernet MTU is 1500 Bytes.
ARP: Address resolution protocol is used for discovering link layer address e.g. MAC address.
Physical Layers: Uses hubs and repeaters to amplify signal
Hubs vs Repeaters: Rep. connect LANs, Hubs join multiple input lines
Limitations: One large collision domain, cannot support multiple LAN technologies, limitation of max nodes and distances
Bridges: Connects LANs, each segment is its own collision domain
Switches: Connects individual computers
Dedicated access: Host has direct connection to the switch
Full duplex: Each connection can send in both directions
Advantages over hubs/switches: Only forwards frames as needed, extends the geographic span of the network, improves privacy by limiting scope of frames, applies carrier sense/collision detection, joins segments using different technologies.
Disadvantages: Delay in forwarding frames, need to learn where to forward frames, higher cost
Cut-through switching: Forward frame before whole frame arrives
Self learning: Inspect source MAC, associate the address with the incoming interface, store mapping in table, TTL to forget mapping
Handling Misses: Unfamiliar destination is flooded to all interfaces
Spanning trees: Each switch thinks it's the root, sends a message out every interface, switches update their view of the root, switches compute their distance from the root
- Root needs to continue sending messages (failure root requires new root elected)
- Failure of non-root switch requires recomputing spanning tree

Comparing hubs/switches/routers:

	hubs	routers	switches
traffic isolation	no	yes	yes
plug & play	yes	no	yes
optimal routing	no	yes	no
cut through	yes	no	yes

IP: connectionless (mis-sequence), unreliable (may drop packets), best effort (only if necessary), datagram (individually routed)
Fragmentation: hosts use path MTU discovery to find smallest MTU

```
Class A   0 | Net ID |       Host ID          |
Class B   10 |     Net ID   |     Host ID     |
Class C   110 |              Net ID | Host ID|
Class D   1110 |    Multicast Group ID        |
Class E   11110 |     Reserved                 |
```

Subnetting: Within org to subdivide the organization's net ID
Classless Inter-domain routing: Prefix in form x/y where x indicates prefix of address, y indicates length of segment (length of match)
Prefix Aggregation: ISP serves 128.9.14.0/24 and 128.9.15.0/24, it can tell other routers to send all packets to 128.0.14.0/23
Error reporting: ICMP by router/end-host to report some types of error

Routing/Forwarding

Forwarding: Lookup in forwarding table, decrement TTL (update checksum), forward packet to outgoing interface, transmit packet to link
Longest prefix match: Maps each IP prefix to next-hop link | *Brute force is too slow*, use hardware content addressable memories,
Forward vs Route: Forwarding is on the data plane - directs data packet | Routing on control plane - computes path the packets will follow - end-to-end performances, balances network resources, transient disruptions
Distributed Bellman-Ford: Each node maintains distance vector (Dv) to all other nodes, each time a nodes Dv changes, it notifies its neighbours, and neighbours compute its Dvs and forwards
- Converges because at every iteration, distances reduces, lower bounded at 0 (distances are discrete)
- Algorithm runs for as longs as the longest shortest path
- For **N** nodes, takes **N - 1** steps at max.

Problems: Bad news travels slowly + Counting to infinity problem
- Set infinity to small integer - Split horizon: Don't advertise cost to the one you received it from

Dijsktra's: *Initialization*:

```
S = {u}
for all nodes in v:
    if v adjacent to u {
        D(v) = c(u, v)
    else D(v) = infinity
```

Loop until all nodes in S:

```
    find w not in S with smallest D(w):
        add w to S
        update D(v) for all v adj to w and not in S:
            D(v) = min( D(v), D(w) + c(w, v)   )
```

Forward table only contains next hop, don't write entire path
Link State Packet: ID of router that created LSP, list of neighbours/cost
Reliable Flooding: Resend LSP over all links (except incident link), if the sequence number is newer.
Robustness: LS: Node can advertise incorrect link cost, each node computes its own table | DV node can advertise incorrect path, each node's table used by others (error propagates)

Internet Topology and Routing

Autonomous Systems (AS): Routers/links managed by an institution
Hierarchy of AS: Tier 1 - national, Medium - regional, small - company
Internet Routing Architecture: Divided into AS, hierarchy of AS, Interaction between AS (internal topology is not shared)
Customer-provider: Customer is reachable to everyone and doesn't provide transit service
Peer-peer: Peers exchange traffic b/w customers - AS exports only customer routes to a peer - AS exports a peer's route to its customers
Tier 2: Provide downstream transit, but need at least one provider
Stub AS: Don't provide transit service to others
Backbone networks: Multiple points-of-presence (PoPs), lots between between PoPs, accommodate traffic demands and limit delay
Multihoming: Two or more providers (extra reliability)
Inferring AS relationships: Business relationships determine routing policies, routing policies determine paths, so look at chosen paths and infer policies | Challenges: Incomplete measurement data, real relationships are sometimes more complex
Routing in the Internet: Within an AS, the admin chooses an Interior Gateway Protocol, between AS's, the Internet uses an Exterior GP
Interior Gateway Protocol (IGP): Protocol used to exchange interdomain routing info among routers in the same domain.
Interior Routing Protocols: RIP: uses distance vector, no authentication OSPF: Link-state updates sent when required, runs Dijkstras, authenticated updates, AS may be partition into areas
Interdomain Routing: Destinations are IP prefixes, Nodes are AS, links are connections & business relationships
Challenges: Scale, privacy, policy (no notion of link cost)
LSP is Problematic: Divulges sensitive info, high processing overhead
DV: Adv: Hides internal topology Dis: Slow convergence
Path Vector Routing: Advertise the entire path, Path vector: for each destination, send entire path.
Border Gateway Protocol (BGP): Advertises complete paths (a list of AS's), local policies pick preferred path (ensuring business relationship)
Incremental Protocol: A node learns multiple paths to dest, applies policy to pick single active route *Announcement*: Upon new active route, add node id to path *Withdrawal:* send withdraw msg for dropped path
Flexible Policies: Route learned from customer preferred over route learned from peer, preferred over route learned from provider
Import Policy: Filter unwanted routes from neighbour, manipulate attributes to influence path selection
Export Policy: Filter routes you don't want to tell your neighbour, manipulate attributes to control what they see, don't announce routes from one peer to another or for network-management hosts
Joining BGP+IGP: Internal BGP to distribute information within an AS interacts with the IGP to compute forwarding tables
BGP Converges Slowly, if at all: Path vector avoids count-to-infinity, but AS must explore many paths. Most popular destination have very stable BGP routes,

Transport Protocols

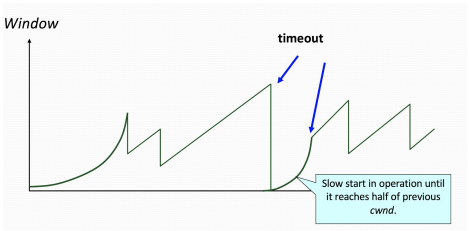
Transport Protocol: Provides logical communication between application processes running on different hosts
User Datagram Protocol (UDP): IP plus port numbers to support demultiplexing, optional error checking on the packet contents, lightweight and avoid overhead of reliable delivery/delay of ordered
Usage: Finer control over what data is sent/when, no delay for connection establishment, connection state, small header overhead
Transmission Control Protocol (TCP): Explicit setup/teardown, sends and receives a stream of bytes (not messages), reliable in-order delivery, prevent overflow of receiver's buffer space, congestion control
Supports: Checksum, sequence number, retransmission (for reliability)

IP packet should be no bigger than MTU, TCP segment is no more than Maximum Segment Size (MSS) bytes
MSS: TCP waits until having collected MSS Bytes from sending process. It's usually MTU - Header sizes
Initial Sequence Number (ISN): Sequence number for the very first byte, should not be 0 because reused port might receive an old packet in flight and might be associated with the new connection - requires changing the ISN over time (due to 32-bit clock)
TCP Handshake:
A ---------- SYN with A's ISN ----------> B
A <-- ---- SYN/ACK with B's ISN and A's ISN + 1 ------- B
A ---- ACK with Sequence number and B's ISN + 1 ---> B
If SYN packet is lost, SYN/ACK won't arrive so have a timeout
Automatic Repeat reQuest (ARQ): Receiver sends ACK when it receives packet, sender waits for ACK and timeouts if doesn't arrive
How Long Should Sender Wait: TCP sets timeout as a function of RTT, can estimate RTT by watching ACKs
EstimatedRTT = a * EstimatedRTT + (1 - a) * SampleRTT
Karn/Partridge Algo: Only collect samples for segments sent 1 single time
TCP Sliding Window: Stop/wait inefficient (only 1 segment is in flight), allow larger amount of data "in flight", allow sender ahead of receiver
Receiver Buffer: Window size is amount that can be sent with ACK, receiver advertises the window to the sender in TCP Header.
Fast Retransmission: When packet n is dropped, ACK says receiver still waiting on packet n (while getting n+1…), sender retransmits data after the triple duplicate ACK
Effectiveness: Good for long data transfers with high window size, low bursts in packet losses (not great for short transfers i.e. forced reload)
Tear Down of TCP:
A ------- FIN --------> B
A <--- FIN/ACK ----- B
A <----- FIN ---------- B
A ------- ACK ------> B

Congestion Control

Congestion Control: Keeping a set of senders from overloading the network

Flow Control: Keeping one fast sender from overwhelming a slow receiver (from filling its buffer)
- Arguably congestion is good since high congestion -> router buffers full -> high usage of network
- But router buffers full -> high delay, so delay/usage tradeoff exists
Congestion Collapse: Increase in network load to the point of a decrease in useful work done
- Possible causes include spurious retransmissions of packets still in flight and undelivered packets
- Ultimate goal is to maximize network throughput (power)
- Fairness also a consideration
Power: A simple metric of how well the network is doing: load/delay
Resource allocation: How nodes meet competing demands for resources
- E.g., link bandwidth and buffer space (when to say no, and to whom)
- Simplest approach: FIFO queue and drop tail packets (drop-tail queueing)
Congestion Control in TCP:
- TCP implements host-based, feedback-based, window-based congestion control
-TCP sources attempts to determine how much capacity is available
- TCP sends packets, then reacts to observable events (loss).
Congestion window: Maximum number of unacknowledged **bytes** in transit
- congestion-control equivalent of receiver window
- Actual TCP window = `min(congestion window, receiver window)`
Additive Increase, Multiplicative Decrease: Increase sending rate linearly, decrease multiplicatively (divide in half)
- Motivation: Consequences of over-sized window are much worse than having an under-sized window.
- Over-sized window: packets dropped and retransmitted
- Under-sized window: somewhat lower throughput
- Leads to TCP "sawtooth" in window vs. time graph
- Congestion window never drops below one MSS
Sending Rate: TCP sending rate is window/RTT. Not sawtooth.
Slow start: When TCP connection opened, Start with a small congestion window (1 MSS initially). Increase rate exponentially until first loss (not actually "slow", name is for historical reasons).



Triple duplicate ACK: One type of loss in TCP. Packet n is lost, but packets n+1, n+2, etc. arrive, Receiver sends duplicate acknowledgments, and the sender retransmits packet n quickly. Solution: Do a multiplicative decrease and keep going
Timeout: Second type of loss in TCP. Packet n is lost and detected via a timeout (E.g., because all packets in flight were lost). After the timeout, blasting away for the entire CWND would trigger a very large burst in traffic, so start CWND low and do another slow start phase.
Idle Period: Period of time during which the TCP connection is idle. After idle period, network conditions may have changed (eg. maybe more flows now traversing the link). Dangerous to blast away at old rate, so some TCP implementations repeat slow start.
Nagle's Algorithm: Algorithm for increasing TCP efficiency in interactive applications (eg. SSH).
- Motivation: Interactive applications send lots of small packets, which are wasteful due to being mostly header, so want to combine packets when possible. But, tradeoff with delay given that we may have to block the sending of a packet while waiting for more data to send. Need to balance competing tradeoffs.
- Algorithm: Wait if the amount of data is small (less than MSS)and some other packet is already in flight, that is, send at most one small packet per RTT.
Piggybacking: Using a data packet to also ACK a packet sent to you. Appealing, as increases efficiency. Works well
Delayed ACK: Delaying ACKing packets to you in the hopes application will come up with data to send so you can piggyback the ACK on a data packet.
- Efficiency/delay tradeoff. Wait is limited by forcing an ACK after 200 or 500ms and ACKing every other full-sized packet regardless of whether or not we have data available.

Queuing Mechanisms

End-to-end principle: Design principle for the Internet that keeps functionalities at the end hosts *Pros:* Flexible/easy to change *Cons:* Trust at the hands of least trusted component, high overhead
Drop-tail queuing: Leads to many packets dropping when link buffer becomes full **Slow Feedback:** Feedback comes when buffer is completely full, and filling buffer increases RTT
Random Early Detection (RED): Router notices that queue is getting backlogged, and randomly drops packets to signal congestion
Properties: Drops packet in proportion to each flow's rate (high-rate flows have more packets, hence high chance of being selected), desynchronizes sources (allowing aggregate flow to be steady)
Drop probability: Function w.r.t queue length, if buffer occupancy is above a threshold, begin RED
RED Avg. Qu Len: Moving average of the queue length is used so as to detect long term, congestion, yet allow short term bursts to arrive
AvgLen_n+1 = (1 - a) * AvgLen_n + a * Length_n
Problems with RED: Hard to tune parameters (i.e. threshold to start dropping, alpha (a), drop probability function)
Explicit Congestion Notification (ECN): Router marks packet with ECN bit, borrow two of the Type-Of-service bits in IPv4 header

Middleboxes

Middleboxes: Interposed b/w communicating hosts (firewalls, traffic shapers, proxy caches)
Networking Address Translation: Allows to share addresses among numerous devices, intended as a short-term remedy to IP addr depletion
NAT Box requires IP address, make the inside look like a single IP addr, hence local addresses are not globally unique.
Port-Translating NAT: Replace source address with NAT address, replace source port number with new port number, remote host respond using (NAT address, new port #)
NAT Mapping Table: Create entry for packet (source addr, source port),
if no packets arrive within time window, delete entry (soft state)
Objections: Port #s are meant for addressing processes, makes it hard to run a server behind a NAT, difficult to support p2p, routers are not supposed to look at port numbers, NAT violates end-to-end principle
Firewalls: Isolate internal net from larger internet
Challenges: Hard to inspect every packet for high speed, may have large/complex filtering rules.
Packet Filtering: Filter based on source/dest IP address, TCP/UDP source/dest port numbers, ICMP message type, TCP/SYN/ACK bits
Traffic Management: Traffic shaping (rate limiting), separate queues (use rules to group related packets, then round-robin schedule groups)
Application Gateways: Filter packets on application data, force all application data to go through gateway, require user login, apply policy
Motiviations: Enable detail policies, avoid rogue machines sending traffic, enable central logging, improve performance through caching
Web Proxy: Plays role of both client & server (sends request/response)
Proxy Caching: Enables faster response time and lower load on server
Getting req to proxy: Explicit configuration (requires user action to configure), transparent proxy (proxy intercepts packets to server)
Challenges of transparent proxies: Must ensure all packets pass by proxy (ex. place at border), overhead of reconstructing the request, may be viewed as violation of user privacy
Other functions: Anonymization, transcoding, prefetching, filtering

Software Defined Networking

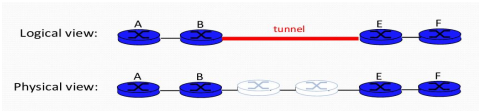
Making network applications is hard because network hardware is vertically integrated, closed and proprietary.
Software Defined Networking: The separation of the control plane from the data plane and the implementation of the control plane in software.
- Control plane now consists of software running on some machine (usually a distributed system to handle load).
- Much more room for innovation, much easier to configure networks, much lower barrier to entry for competition, lower cost
Network OS: Distributed system that runs on controllers in the network. Has a consistent, up-to-date global network view. Uses forwarding abstraction to get state information from forwarding elements, and give control directives to forwarding elements.
Control Program: Program that runs on network OS to implement some feature. Not a distributed system (network OS abstracts distributed systems aspects).
Forwarding Abstraction: Abstracts away forwarding hardware.
Openflow example of such an abstraction in real life.
Openflow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network.
Openflow: Open standard to run experimental protocols in production networks.
Flow: General notion of traffic flows (eg. Jim's traffic, Traffic to Canada, HTTP traffic). SDN Can apply actions to flows (Allow/deny, route/re-route, isolate, remove).
Matching: Individual bits in headers can be matched against to determine what actions to apply to a packet. This can be done down to individual bits across several headers, allowing for high granularity.

Network Security

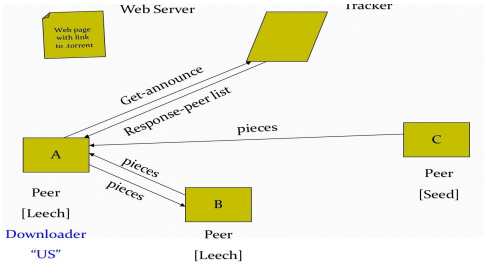
Network Telescope: Large piece of globally announced IP addresses, inbound traffic is almost always anomalous.
Network Security Goals: Availability, Protection, Authenticity, Data Integrity, Privacy
Internet Design vs Security:
Destination Routing: Shouldn't detect packet spoofing however
Packet Based: Difficult resource bound per communication, can't rely on source address for hogging issue
Global Addressing: Democratic communication, even to people who don't want to be talked to
Simple to join: Misbehaving routers can do bad things (no model of trust)
Power in end-hosts: Giving power to least trust user
"Ad-hoc" naming system: Fate sharing in hierarchical system, off route=more trusted elements
DoS: Saturate uplink bandwidth using legit requests/CPU time/Memory
Solution: Use CDN, admission control at the server
TCP SYN Flood: Send many connections with spoofed IP addresses. Victim allocs resource for each request until timeout until half-open connections are exhausted, then no more requests are accepted. Systems becomes unresponsive for legit traffic.
Defenses: - Reduce half-open connection timeout | - Drop half-open connections randomly - Send SYN-ACK cookies **:** Client sends SYN, server responds with SYN-ACK ISNs = H(src addr, src port, dest addr, dest port, rand). Honest client responds with ACK(ISNs). Server regenerates ISNs and checks that client response matches ISNs. Rand is derived from 32bit time counter. (Vuln to connection spoofing)
DDoS: Performing DoS attack across multiple compromised machines
DoS Aplenty: Attacker guesses TCP seq. num for an existing connection, attacker send reset packet to close connection. Most systems allow large window of acceptable sequence nums. Attack effective against long lived connections.
Congestion control DoS Attack: Generate TCP Flow to force target to repeatedly enter retransmission timeout state, difficult to detect because packet rate is low
Bellovin/Mockapetris Attack: User/hosts trust host-address mapping provided by DNS. Spoof reverse DNS to make host trust attacker. Gain control of DNS service for evil.org, select target machine in good.net, find trust relationships
DNS Rebinding Attack: Poisoning replaces victim domain -> attacker IP. Rebinding replaces attacker domain -> victim IP. Bypasses same origin policy - attacker and victim IP appear to belong to same domain
1. **Browser:** Attacker gets client to visit attacker's site
2. **Attacker:** Attacker controls DNS and returns resp. with short TTL, attacker's site serves a web page with a malicious script
3. **Browser:** Script makes request to attacker's website, making 2nd DNS query, which reaches attacker's DNS due to short TTL
4. **Attacker:** Attacker's DNS returns IP address of victim web server
5. **Browser:** Victim and attack server appear to be same origin. If browser is located on Intranet, rebinding may allow accessing!
Defenses: Browsers don't consider same IP addresses because sites might switch IP for load balancing. Browser can pin DNS/IP mapping to value of first DNS response. Block resolution of external names into local IP addresses at a local nameserver.
TCP Connection Spoof: TCP Handshake uses sequence number as weak auth. Attacker can forge packet with source set to client's address by guessing the sequence number, thereby connecting as the victim.
IP Spoofing Attack: Attacker opens connection A to get ISN, SYN flood B's queue, Send packets to A that resemble B's transmission, E cannot receive but may execute commands on A
Reconnaissance: Discover available resources (port scanning, host/applic. fingerprinting, traceroute, reverse DNS scanning, SNMP)
Packet Filtering: Usually done on router at perimeter of network
Problems: Attacker can send packets from valid ports, attacker can forge source, not effective for stopping packets from unwanted servers
Stateful packet filter: Only allow traffic initiated by the client, for each flow request keep a little state and sequence number, ensure packets received from Internet belong to an existing flow
Passive reconstruction of TCP Frame: Use passive network element to reconstruct TCP stream and check for harmful payload (virus sign.)
Problems: Can't tell if packet reached destination, how end-host manages overlapping TCP seq nums/overlapping fragments

Overlay Networks

Overlay Network: A logical network built on top of a physical network. Nodes are often end hosts.
Routing Overlay: Alternative routing strategies, no application level processing at the overlay nodes.
IP Tunneling: Type of routing overlay where packets are encapsulated inside an IP datagram.



6Bone: Deploying IPv6 over IPv4 via an IP tunnel.
Tor Network: An overlay to enhance security and privacy
- Obtain a list of nodes from directory servers, pick a random path to destination servers.
Communicating with mobile users: Mobile changes locations frequently -> IP Address changes. Soln: Fixed gateway with fixed IP that tracks mobile's address changes.
Multicast: Delivering same data to many receivers; avoid sending the same data many times. IP multicast not widely deployed so MBone tunneled between nodes.
RON (Resilient Overlay Networks): by building application overlay network, can increase performance and reliability.
Ron Can Outperform IP Routing:
IP doesn't adapt to congestion; RON can reroute. IP Routing depends on AS policies; RON can circumvent. RON has performance degradation, load on hosts, and is limited to a small number of nodes.
Types of P2P networks: Directory based (napster), Unstructured (Gnutella), Structured (distributed hash tables)
Limitations of Directory Based: Single point of failure and performance bottleneck. Locating content is centralized.
Bit Torrent:



Question. A TCP msg of sz 3KB is sent over a series of three routers. The MTU for the routers are 1.5KB, 0.8KB, 1KB. Assume IP hdr is 20 B, link layer headers are 30 B. Show the sequence of packets as they arrive to dest node.

Packet in router 1	Length (ip hdr + tcp msg)	offset
#1	20 + ((1500 – 20)/8)*8 = 20 + 1480	0
#2	20 + (1500 -20)/8)*8 = 20 + 1480	1480 / 8 = 185
#3	20 + (3000 – 1480 – 1480) = 20 + 40	370

Packets in 2nd router (and hence, in the 3rd router and the dst)	Length (ip hdr + tcp msg)	Offset
#1	20 + ((800 – 20) / 8 *8) = 20 + 776	0
#2	20 + (((1500 – 776) – 20) / 8) * 8 = 20 + 704	776 / 8 = 97
#3	20 + 776	97 + 704 / 8 = 185
#4	20 + 704	282
#5	20 + 40	370