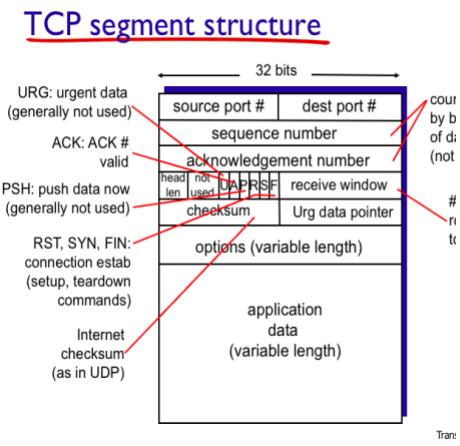
----------RDT----------

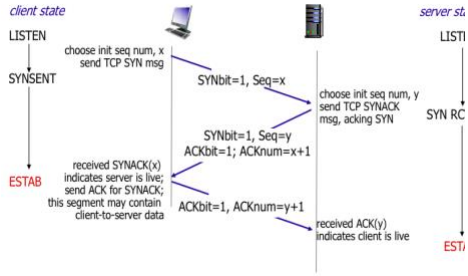
1.0: Reliable transfer, Reliable channel
2.0: channel with bit errors
Question: recover from errors

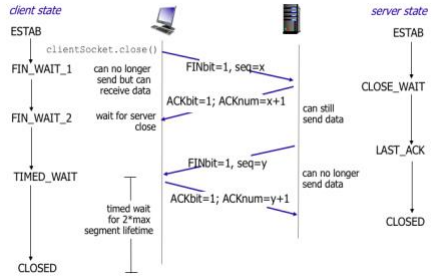
Solution: 1.error detection. 2. Feedback(ACK, NAK)
2.1: sender, handles grabled ACK/NAK
Solution: sequence number

2.2: NAK-free
3.0: channels with errors and loss
Solution: time

----------TCP----------



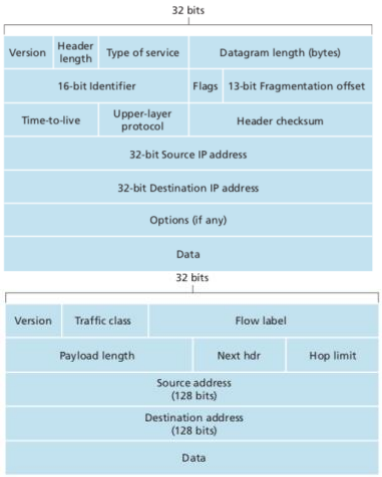


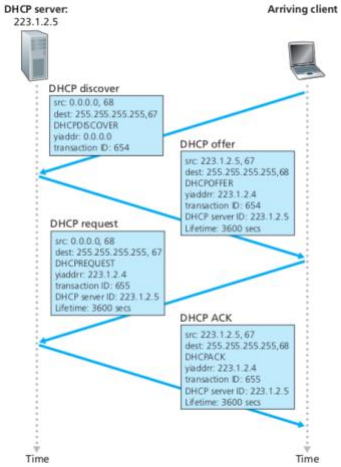


TCP Congestion Control:
Slow Start: double cwnd every RTT
Congestion Avoidance:
cwnd += (MSS/cwnd)*MSS upon every incoming non-duplicate ACK
Time out: ssthresh = cwnd/2, cwnd=1

Fast retransmit / fast recovery:
ssthresh = cwnd/2
cwnd = ssthresh + 3MSS
retransmit the lost packet
increase cwnd by 1 MSS upon every duplicate ACK

----------IPv4 and IPv6----------





IPv4	IPv6
The IPv4 datagram consists of lot of fields out of which some are optional.	The IPv6 datagram consists of minimum number of fields.
The header size of IPv4 is a not fixed value. The size of a typical IPv4 header is 20-byte.	The header size of IPv6 datagram is fixed 40-byte.
The optional fields are also included in the header of IPv4.	The optional fields are maintained using the extension header.
The length of the datagram is 16-bit field and is specified by the Datagram length.	The length of the datagram is a 20-bit field and it is specified by the Flow label.
The source address is 32-bit field.	The source address is 128 bit field.
The destination address is a 32-bit field.	The destination address is 128-bit field.
It consist of Fragmentation and Reassembly, Header checksum and Options fields.	It does not consist of Fragmentation and Reassembly, Header checksum and Options fields because they are costlier and time-consuming.
The total length in IPv4 is the total length of IPv4 packet including header in bytes.	The payload length does not include the size of header in IPv6. It includes the length of additional headers used in bytes.

----------Link-state vs. distance-vector----------

Properties:	Centralized algorithm	Distributed algorithm
Computation of least cost path:	Done between source and destination with complete and global knowledge about the network.	Done by the routers in an iterative and distributed manner.
Knowledge while starting the algorithm:	Connectivity between all nodes and costs of all links are taken as inputs.	At the beginning of each node, the knowledge of the costs of its own directly attached links or connected neighbors are only known.

Example for Centralized routing algorithm: Open Shortest Path First (OSPF).
• OSPF takes the centralized approach as a complete topological map, which is constructed by each router for the entire autonomous system.
Example for Distributed routing algorithm: Border Gateway Protocol (BGP).
• BGP takes the decentralized approach for routing the packet, like each node maintains it cost estimates, and each router has a forwarding table across multiple autonomous systems.

Link State routing algorithm	Distance Vector routing algorithm
The network topology and all the link costs are the input to this algorithm.	The input to the algorithm is all the associated costs with the current node to all its neighbors.
It computes the least-cost path from source to destination with a complete knowledge on the network.	It computes the least-cost path in an iterative and distributed manner.
The shortest path is calculated using dijkstras algorithm.	The shortest cost path is calculated using Bellman Ford algorithm.
Open Shortest Path First (OSPF) is an example of link state routing algorithm.	Routing Information Protocol (RIP) is an example of distance vector routing algorithm.
CPU utilization is more when compared to distance vector routing algorithm.	CPU utilization is less when compared to link state routing algorithm.
More memory space is required when compared to distance vector routing algorithm.	Less memory space is required when compared to link state routing algorithm.
Convergence time is fast.	Convergence time is very slow.
The tables that are created are routing table, neighbor table and topology table.	The table that is created is routing table.
Updates are multicasted.	Updates are broadcasted.

----------Link-layer---------

Services:
Framing, Link access (MAC), Reliable delivery, Error detection and correction

Broadcast channel:
1.One node transfer at full rate 2. Fairness 3. Decentralized
4. Simple and efficient

Category:
Channel partitioning, random access, taking turns

----------CSMA/CD ----------

1. The adapter obtains a datagram from the network layer, prepares a link-layer frame, and puts the frame adapter buffer.
2. If the adapter senses that the channel is idle (that is, there is no signal energy entering the adapter from the channel), it starts to transmit the frame. If, on the other hand, the adapter senses that the channel is busy, it waits until it senses no signal energy and then starts to transmit the frame.
3. While transmitting, the adapter monitors for the presence of signal energy coming from other adapters using the broadcast channel.
4. If the adapter transmits the entire frame without detecting signal energy from other adapters, the adapter is finished with the frame. If, on the other hand, the adapter detects signal energy from other adapters while transmitting, it aborts the transmission (that is, it stops transmitting its frame).
5. After aborting, the adapter waits a random amount of time and then returns to step 2.

----------Router vs. Switch----------

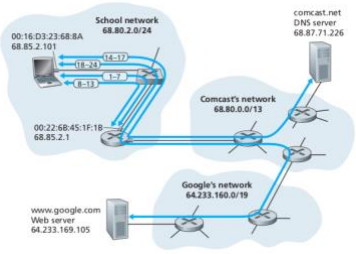
	Hubs	Routers	Switches
Traffic isolation	No	Yes	Yes
Plug and play	Yes	No	Yes
Optimal routing	No	Yes	No

----------Switch self-learning----------

1. The switch table is initially empty.
2. For each incoming frame received on an interface, the switch stores in its table (1) the MAC address in the frame's *source address field*, (2) the interface from which the frame arrived, and (3) the current time. In this manner the switch records in its table the LAN segment on which the sender resides. If every host in the LAN eventually sends a frame, then every host will eventually get recorded in the table.
3. The switch deletes an address in the table if no frames are received with that address as the source address after some period of time (the **aging time**). In this manner, if a PC is replaced by another PC (with a different adapter), the MAC address of the original PC will eventually be purged from the switch table.

----------daily life on web----------

1. The operating system on Bob's laptop creates a **DHCP request message** (Section 4.4.2) and puts this message within a **UDP segment** (Section 3.3) with destination port 67 (DHCP server) and source port 68 (DHCP client). The UDP segment is then placed within an **IP datagram** (Section 4.4.1) with a broadcast



5.7.2 Still Getting Started: DNS and ARP

When Bob types the URL for www.google.com into his Web browser, he begi long chain of events that will eventually result in Google's home page bein played by his Web browser. Bob's Web browser begins the process by creat **TCP socket** (Section 2.7) that will be used to send the **HTTP request** (Sectio to www.google.com. In order to create the socket, Bob's laptop will need to the IP address of www.google.com. We learned in Section 2.5, that the **DNS p col** is used to provide this name-to-IP-address translation service.

8. The operating system on Bob's laptop thus creates a **DNS query message** (Se 2.5.3), putting the string "www.google.com" in the question section of the DN message. This DNS message is then placed within a UDP segment with a dest tion port of 53 (DNS server). The UDP segment is then placed within an IP da gram with an IP destination address of 68.87.71.226 (the address of the DNS returned in the DHCP ACK in step 5) and a source IP address of 68.85.2.101.
9. Bob's laptop then places the datagram containing the DNS query message an Ethernet frame. This frame will be sent (addressed, at the link layer) to gateway router in Bob's school's network. However, even though Bob's la

knows the IP address of the school's gateway router (68.85.2.1) via the DHC ACK message in step 5 above, it doesn't know the gateway router's MAC address. In order to obtain the MAC address of the gateway router, Bob's lap top will need to use the **ARP protocol** (Section 5.4.1).

10. Bob's laptop creates an **ARP query** message with a target IP address of 68.85.2.1 (the default gateway), places the ARP message within an Ethernet frame with a broadcast destination address (FF:FF:FF:FF:FF:FF) and sends t Ethernet frame to the switch, which delivers the frame to all connected devices, including the gateway router.
11. The gateway router receives the frame containing the ARP query message on th interface to the school network, and finds that the target IP address of 68.85.2.1 the ARP message matches the IP address of its interface. The gateway router thu prepares an **ARP reply**, indicating that its MAC address of 00:22:6B:45:1F:1B corresponds to IP address 68.85.2.1. It places the ARP reply message in an Ethernet frame, with a destination address of 00:16:D3:23:68:8A (Bob's laptop) and sends the frame to the switch, which delivers the frame to Bob's laptop.
12. Bob's laptop receives the frame containing the ARP reply message and extrac the MAC address of the gateway router (00:22:6B:45:1F:1B) from the ARP reply message.
13. Bob's laptop can now (*finally!*) address the Ethernet frame containing the DNS query to the gateway router's MAC address. Note that the IP datagram in this frar has an IP destination address of 68.87.71.226 (the DNS server), while the frame h a destination address of 00:22:6B:45:1F:1B (the gateway router). Bob's laptop sends this frame to the switch, which delivers the frame to the gateway router.

IP destination address (255.255.255.255) and a source IP address of 0 since Bob's laptop doesn't yet have an IP address.

2. The IP datagram containing the DHCP request message is then place an **Ethernet frame** (Section 5.4.2). The Ethernet frame has a destinat addresses of FF:FF:FF:FF:FF:FF so that the frame will be broadcast t devices connected to the switch (hopefully including a DHCP server) frame's source MAC address is that of Bob's laptop, 00:16:D3:23:68:
3. The broadcast Ethernet frame containing the DHCP request is the firs sent by Bob's laptop to the Ethernet switch. The switch broadcasts th ing frame on all outgoing ports, including the port connected to the r
4. The router receives the broadcast Ethernet frame containing the DHCP on its interface with MAC address 00:22:6B:45:1F:1B and the IP datag extracted from the Ethernet frame. The datagram's broadcast IP destina address indicates that this IP datagram should be processed by upper la cols at this node, so the datagram's payload (a UDP segment) is thus **de plexed** (Section 3.2) up to UDP, and the DHCP request message is extr from the UDP segment. The DHCP server now has the DHCP request
5. Let's suppose that the DHCP server running within the router can allo addresses in the **CIDR** (Section 4.4.2) block 68.85.2.0/24. In this exa IP addresses used within the school are thus within Comcast's addres

5.7.3 Still Getting Started: Intra-Domain Routing to the DNS Server

14. The gateway router receives the frame and extracts the IP datagram containi the DNS query. The router looks up the destination address of this datagram (68.87.71.226) and determines from its forwarding table that the datagram sh be sent to the leftmost router in the Comcast network in Figure 5.32. The IP gram is placed inside a link-layer frame appropriate for the link connecting t school's router to the leftmost Comcast router and the frame is sent over this
15. The leftmost router in the Comcast network receives the frame, extracts th datagram, examines the datagram's destination address (68.87.71.226) and determines the outgoing interface on which to forward the datagram towa the DNS server from its forwarding table, which has been filled in by Com cast's intra-domain protocol (such as **RIP**, **OSPF** or **IS-IS**, Section 4.6) as as the **Internet's inter-domain protocol**, **BGP**.
16. Eventually the IP datagram containing the DNS query arrives at the DNS se The DNS server extracts the DNS query message, looks up the name www.google.com in its DNS database (Section 2.5), and finds the **DNS rec**

record that contains the IP address (64.233.169.105) for www.google.com. (assuming that it is currently cached in the DNS server). Recall that this cache data originated in the **authoritative DNS server** (Section 2.5.2) for googlecon The DNS server forms a **DNS reply message** containing this hostname-to-IP- address mapping, and places the DNS reply message in a UDP segment, and t segment within an IP datagram addressed to Bob's laptop (68.85.2.101). This datagram will be forwarded back through the Comcast network to the school's router and from there, via the Ethernet switch to Bob's laptop.

17. Bob's laptop extracts the IP address of the server www.google.com from the DNS message. *Finally*, after a *lot* of work, Bob's laptop is now ready to cont the www.google.com server!

CDMA:

Channel partitioning

$Z = d_i * c_m$

$D_i = \text{sum}(Z_{i,m} * c_m)/M$

Let's suppose the DHCP server allocates address 68.85.2.101 to Bob's laptop. The DHCP server creates a **DHCP ACK message** (Section 4.4.2) containing this IP address, as well as the IP address of the DNS server (68.87.71.226), the IP address for the default gateway router (68.85.2.1), and the subnet block (68.85.2.0/24) (equivalently, the "network mask"). The DHCP message is put inside a UDP segment, which is put inside an IP datagram, which is put inside an Ethernet frame. The Ethernet frame has a source MAC address of the router's interface to the home network (00:22:6B:45:1F:1B) and a destination MAC address of Bob's laptop (00:16:D3:23:68:8A).

6. The Ethernet frame containing the DHCP ACK is sent (unicast) by the router to the switch. Because the switch is **self-learning** (Section 5.4.3) and previously received an Ethernet frame (containing the DHCP request) from Bob's laptop, the switch knows to forward a frame addressed to 00:16:D3:23:68:8A only to the output port leading to Bob's laptop.
7. Bob's laptop receives the Ethernet frame containing the DHCP ACK, extracts the IP datagram from the Ethernet frame, extracts the UDP segment from the IP datagram, and extracts the DHCP ACK message from the UDP segment. Bob's DHCP client then records its IP address and the IP address of its DNS server. It also installs the address of the default gateway into its **IP forwarding table** (Section 4.1). Bob's laptop will send all datagrams with destination address outside of its subnet 68.85.2.0/24 to the default gateway. At this point, Bob's laptop has initialized its networking components and is ready to begin processing the Web page fetch. (Note that only the last two DHCP steps of the four presented in Chapter 4 are actually necessary.)

5.7.4 Web Client-Server Interaction: TCP and HTTP

18. Now that Bob's laptop has the IP address of www.google.com, it can create the **TCP socket** (Section 2.7) that will be used to send the **HTTP GET** message (Section 2.2.3) to www.google.com. When Bob creates the TCP socket, the TCP in Bob's laptop must first perform a **three-way handshake** (Section 3.5.6) with the TCP in www.google.com. Bob's laptop thus first creates a **TCP SYN** segment with destination port 80 (for HTTP), places the TCP segment inside an IP datagram with a destination IP address of 64.233.169.105 (www.google.com), places the datagram inside a frame with a destination MAC address of 00:22:6B:45:1F:1B (the gateway router) and sends the frame to the switch.
19. The routers in the school network, Comcast's network, and Google's network forward the datagram containing the TCP SYN towards www.google.com, using the forwarding table in each router, as in steps 14–16 above. Recall that the router forwarding table entries governing forwarding of packets over the inter-domain link between the Comcast and Google networks are determined by the **BGP** protocol (Section 4.6.3).
20. Eventually, the datagram containing the TCP SYN arrives at www.google.com. The TCP SYN message is extracted from the datagram and demultiplexed to the welcome socket associated with port 80. A connection socket (Section 2.7) is created for the TCP connection between the Google HTTP server and Bob's laptop. A TCP SYNACK (Section 3.5.6) segment is generated, placed inside a datagram addressed to Bob's laptop, and finally placed inside a link-layer frame appropriate for the link connecting www.google.com to its first-hop router.
21. The datagram containing the TCP SYNACK segment is forwarded through the Google, Comcast, and school networks, eventually arriving at the Ethernet card in Bob's laptop. The datagram is demultiplexed within the operating system to the TCP socket created in step 18, which enters the connected state.
22. With the socket on Bob's laptop now (*finally!*) ready to send bytes to www.google .com, Bob's browser creates the HTTP GET message (Section 2.2.3) containing the URL to be fetched. The HTTP GET message is then written into the socket, with the

GET message becoming the payload of a TCP segment. The TCP segment is placed in a datagram and sent and delivered to www.google.com as in steps 18–20 above.

23. The HTTP server at www.google.com reads the HTTP GET message from the TCP socket, creates an **HTTP response** message (Section 2.2), places the requested Web page content in the body of the HTTP response message, and sends the message into the TCP socket.
24. The datagram containing the HTTP reply message is forwarded through the Google, Comcast, and school networks, and arrives at Bob's laptop. Bob's Web browser program reads the HTTP response from the socket, extracts the html for the Web page from the body of the HTTP response, and finally (*finally!*) displays the Web page!

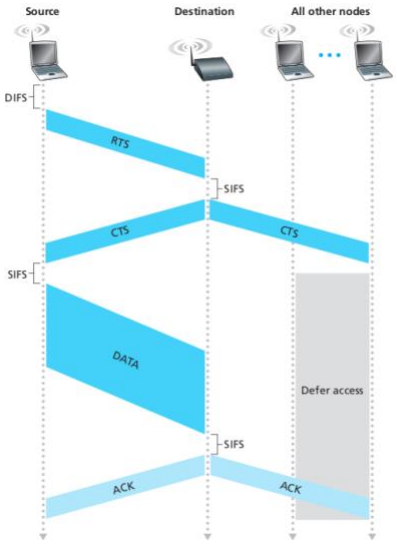
----------CSMA/CA----------

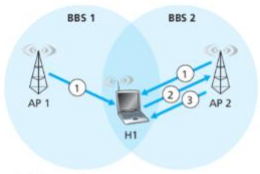
CA over CD:

- The ability to detect collisions requires the ability to send (the station's ov nal) and receive (to determine whether another station is also transmitting same time. Because the strength of the received signal is typically very compared to the strength of the transmitted signal at the 802.11 adapte costly to build hardware that can detect a collision.
- More importantly, even if the adapter could transmit and listen at the sam (and presumably abort transmission when it senses a busy channel), the a would still not be able to detect all collisions, due to the hidden terminal pr and fading, as discussed in Section 6.2.

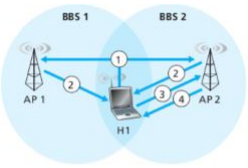
Details:

1. If initially the station senses the channel idle, it transmits its frar period of time known as the **Distributed Inter-frame Space (D** Figure 6.10.
2. Otherwise, the station chooses a random backoff value using bir tial backoff (as we encountered in Section 5.3.2) and counts do when the channel is sensed idle. While the channel is sensed bu value remains frozen.
3. When the counter reaches zero (note that this can only occur wh is sensed idle), the station transmits the entire frame and then w acknowledgment.
4. If an acknowledgment is received, the transmitting station knows that it has been correctly received at the destination station. If the station has a frame to send, it begins the CSMA/CA protocol at step 2. If the acknow ment isn't received, the transmitting station reenters the backoff phase i 2, with the random value chosen from a larger interval.





a. Passive scanning
1. Beacon frames sent from APs
2. Association Request frame sent: H1 to selected AP
3. Association Response frame sent: Selected AP to H1



a. Active scanning
1. Probe Request frame broadcast from H1
2. Probes Response frame sent from APs
3. Association Request frame sent: H1 to selected AP
4. Association Response frame sent: Selected AP to H1

802.11 frame

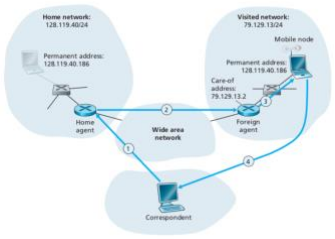
Ethernet frame:

Preamble (8 bytes)	Destination Address (6 bytes)	Source Address (6 bytes)	Type (2 bytes)	Data (46 to 1,500 bytes)	CRC (4 bytes)

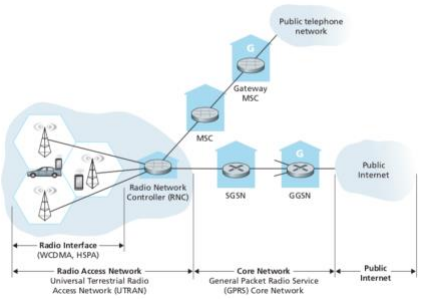
802.11 frame:

Frame control (2 bytes)	Duration (2 bytes)	Add1 (6 bytes)	Add2 (6 bytes)	Add3 (6 bytes)	Seq Control (2 bytes)	Add4 (6 bytes)	Data (0 to 2312 bytes)	CRC (4 bytes)

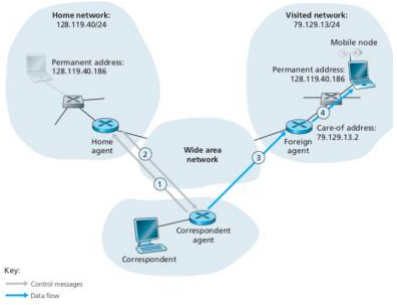
Indirect routing:



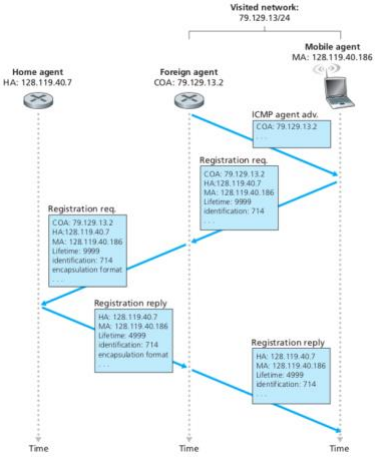
3G



Direct Routing:



Mobile IP registration:



Handoff

1. The old base station (BS) informs the visited MSC that a handoff is to be performed and the BS (or possible set of BSs) to which the mobile is to be handed off.
2. The visited MSC initiates path setup to the new BS, allocating the resources needed to carry the rerouted call, and signaling the new BS that a handoff is about to occur.
3. The new BS allocates and activates a radio channel for use by the mobile.
4. The new BS signals back to the visited MSC and the old BS that the visited-MSC-to-new-BS path has been established and that the mobile should be informed of the impending handoff. The new BS provides all of the information that the mobile will need to associate with the new BS.
5. The mobile is informed that it should perform a handoff. Note that up until this point, the mobile has been blissfully unaware that the network has been laying the groundwork (e.g., allocating a channel in the new BS and allocating a path from the visited MSC to the new BS) for a handoff.
6. The mobile and the new BS exchange one or more messages to fully activate the new channel in the new BS.
7. The mobile sends a handoff complete message to the new BS, which is forwarded up to the visited MSC. The visited MSC then reroutes the ongoing call to the mobile via the new BS.
8. The resources allocated along the path to the old BS are then released.

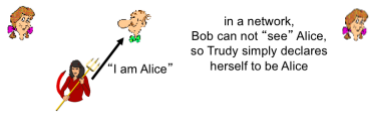
Network security

Confidentiality: only sender, intended receiver should "understand" message contents
Authentication: sender, receiver want to confirm identity of each other
Message integrity: sender, receiver want to ensure message not altered without detection
Access and availability: service must be accessible and available to users

Authentication:

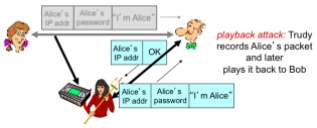
Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

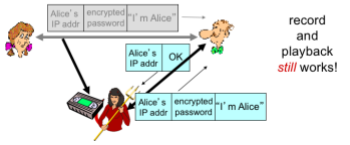


Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



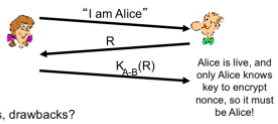
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key

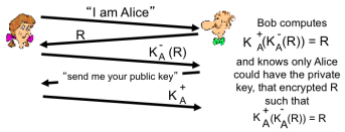


Failures, drawbacks?

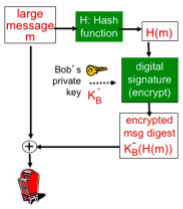
ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

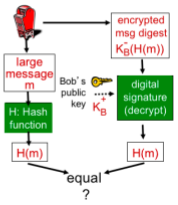
ap5.0: use nonce, public key cryptography



Bob sends digitally signed message:



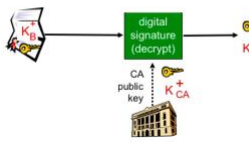
Alice verifies signature, integrity of digitally signed message:



Digital signature:

Certification authority

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere)
  - apply CA's public key to Bob's certificat public key



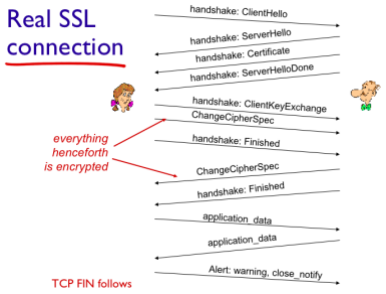
SSL

1. client sends list of algorithms it supports, client nonce
2. server chooses algorithms from list; sends choice + certificate + server nonce
3. client verifies certificate, extracts server's key, generates pre_master_secret, encryp server's public key, sends to server
4. client and server independently compute and MAC keys from pre_master_secret a
5. client sends a MAC of all the handshake m
6. server sends a MAC of all the handshake

Why two random nonces?

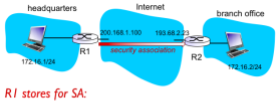
- why two random nonces?
- suppose Trudy sniffs all messages between & Bob
- next day, Trudy sets up TCP connection w Bob, sends exact same sequence of record
  - Bob (Amazon) thinks Alice made two separate for the same thing
  - solution: Bob sends different random nonce fo connection. This causes encryption keys to be on the two days
  - Trudy's messages will fail Bob's integrity chec

Real SSL:



----------VPN----------

Example security association:



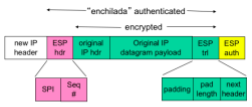
R1 stores for SA:

- 32-bit SA identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

- data integrity
- origin authentication
- replay attack prevention
- confidentiality

Security Association database:

- endpoint holds SA state in *security association database (SAD)*, where it can locate them during processing.
- with n salespersons, 2 + 2n SAs in R1's SAD
- when sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- when IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.



- ESP trailer: Padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key

R1: Convert original datagram to IPsec datagram

- appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- encrypts result using algorithm & key specified by SA.
- appends to front of this encrypted quantity the "ESP header, creating "enchilada".
- creates authentication MAC over the *whole enchilada*, using algorithm and key specified in SA;
- appends MAC to back of enchilada, forming *payload*;
- creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload

IPsec seq#:

- for new SA, sender initializes seq. # to 0
- each time datagram is sent on SA:
  - sender increments seq # counter
  - places value in seq # field
- goal:
  - prevent attacker from sniffing and replaying a packet
  - receipt of duplicate, authenticated IP packets may disrupt service
- method:
  - destination checks for duplicates
  - doesn't keep track of *all* received packets; instead uses a window

IPsec Services:

IPsec datagram:

Firewalls

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections

prevent illegal modification/access of internal data

- e.g., attacker replaces CIA's homepage with something else

allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

Limitations

- *IP spoofing:* router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
- *tradeoff:* degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks

Intrusion detection systems

- packet filtering:
  - operates on TCP/IP headers only
  - no correlation check among sessions
- *IDS: intrusion detection system*
  - *deep packet inspection:* look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
  - examine correlation among multiple packets
    - port scanning
    - network mapping
    - DoS attack