Network Attacks

CMSC 23200, Spring 2026, Lecture 9

Grant Ho and David Cash

University of Chicago, 04/21/2026
Slides adapted from Blasé Ur, Peyrin Kao, Vern Paxson, and Borja Sotomayor

Logistics

- Assignment 4 Due on Thursday (04/23) @ 11:59pm
  - Both Part (a): Crypto II + Part (b): Networking due then

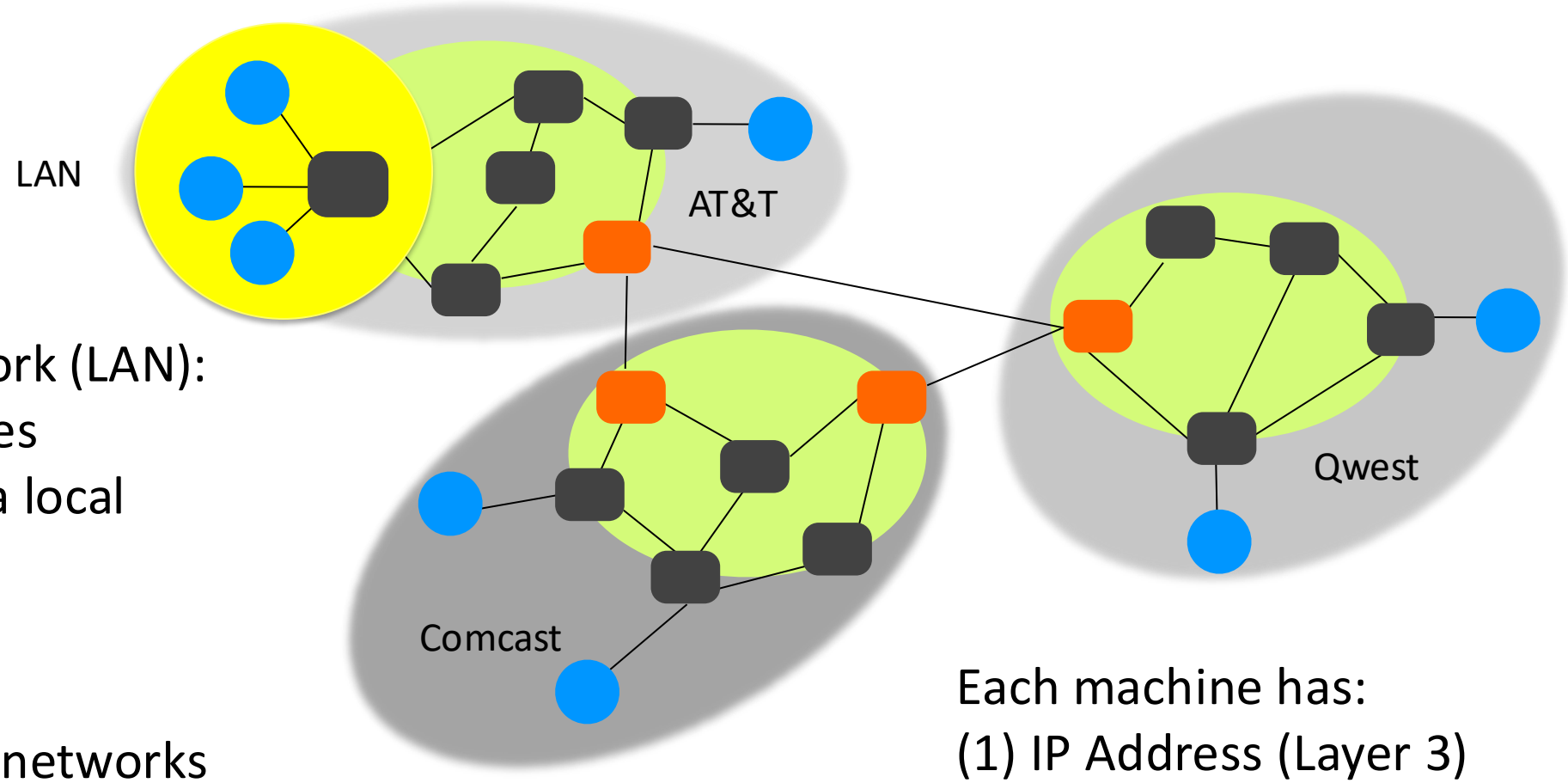
- No assignment for the following 2 weeks:
  Midterm on May 5 in-class

- Final Exam Time: **Wednesday, May 27 @ 6pm**
  - COMBINED TIME FOR BOTH SECTIONS

Outline

- ARP/DHCP Security
- WPA2: Wifi security
- TCP/UDP Attacks
- DNS Security

Recall: The Internet From 10,000 Feet



Local Area Network (LAN):

- Set of machines connected in a local network

Internet (IP):

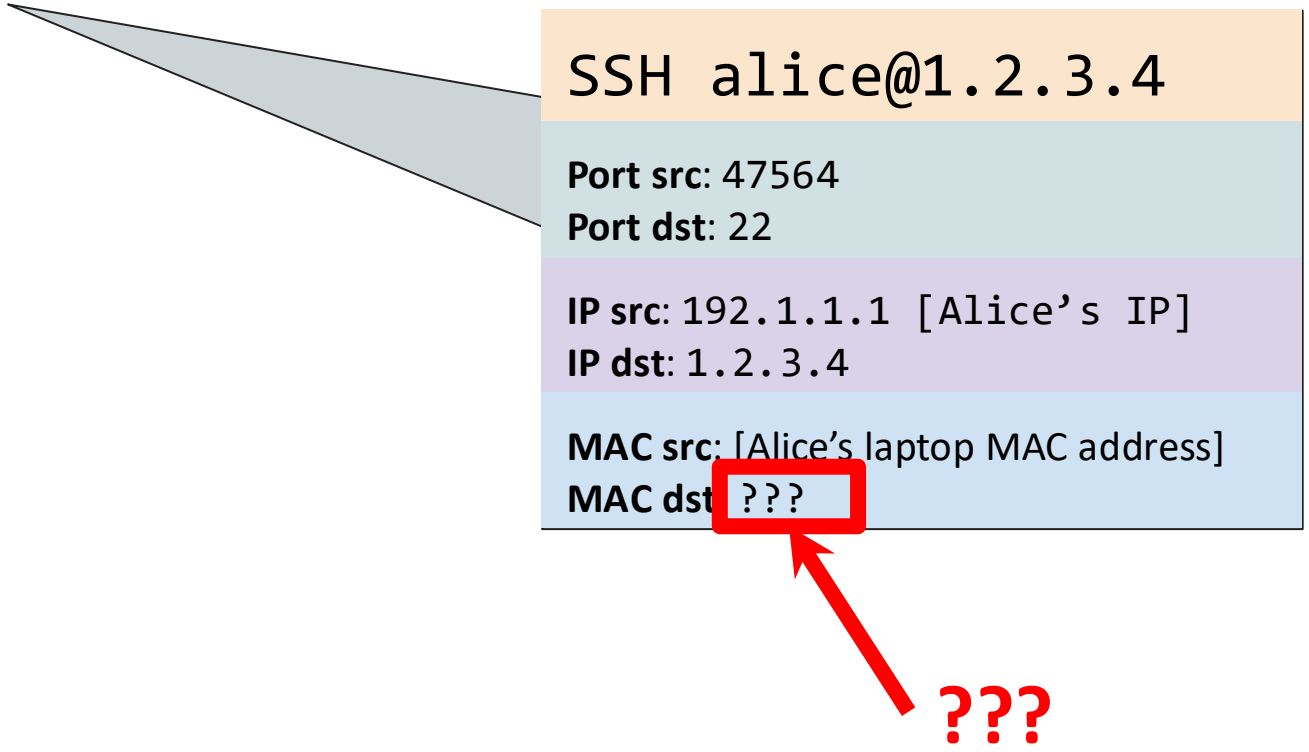
- Set of smaller networks connected via routers

Each machine has:
(1) IP Address (Layer 3)
(2) MAC Address (Layer 2)

Address Resolution Protocol (ARP)

The Problem: Alice knows Bob's IP address & wants to send him data (e.g., Alice performs ssh login to VM [Bob] @ IP address = `1.2.3.4`)

- What should she fill in for the Layer 2 header (MAC Address)?

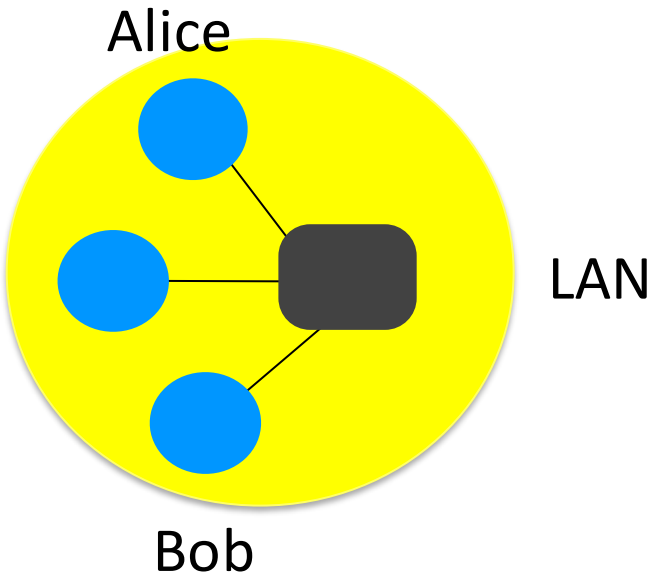


Address Resolution Protocol (ARP)

The Problem: Alice knows Bob's IP address & wants to send him data (e.g., Alice performs ssh login to VM [Bob] @ IP address = `1.2.3.4`)

- What should she fill in for the Layer 2 header (MAC Address)?

ARP: Translates IP addresses to MAC addresses



Address Resolution Protocol (ARP)

The Problem: Alice knows Bob's IP address & wants to send him data (e.g., Alice performs ssh login to VM [Bob] @ IP address = `1.2.3.4`)

- What should she fill in for the Layer 2 header (MAC Address)?

ARP: Translates IP addresses to MAC addresses

1. Alice checks her ARP cache to see if she already knows Bob's MAC address.
2. If Bob's MAC addr not in the cache, Alice **broadcasts** to everyone on the LAN: "What is the MAC address of **`1.2.3.4`**?"
3. Bob responds by sending a message only to Alice: "My IP is **`1.2.3.4`** and my MAC address is **`ca:fe:f0:0d:be:ef`**."
   Everyone else does nothing.
4. Alice caches Bob's MAC address & uses it.

Address Resolution Protocol (ARP)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC

Alice

Bob

Charlie

Dave

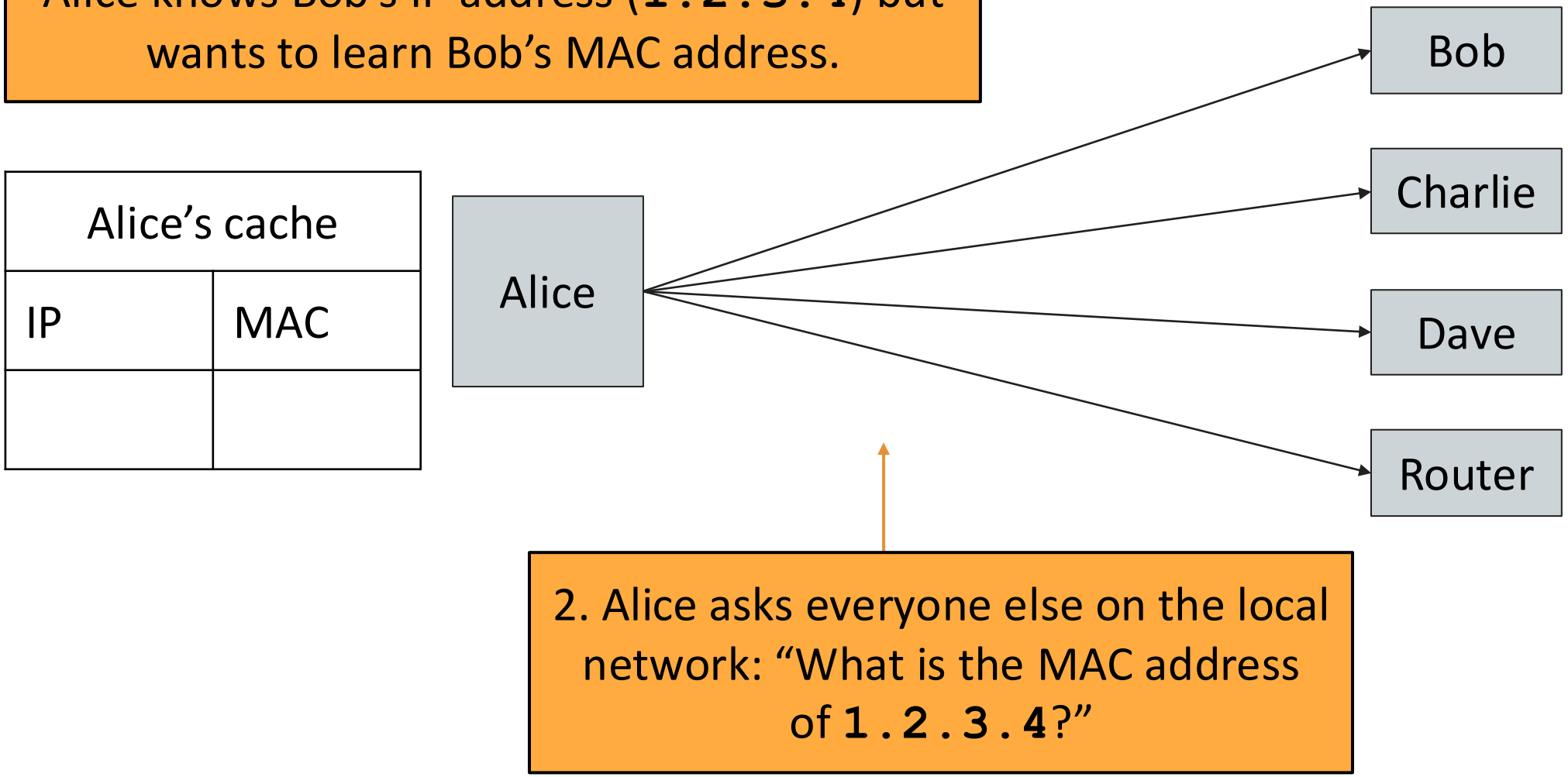
Router

1. Alice checks her cache to see if she already knows the MAC address corresponding to `1.2.3.4`.

Not in the cache: she must make a request to find out.

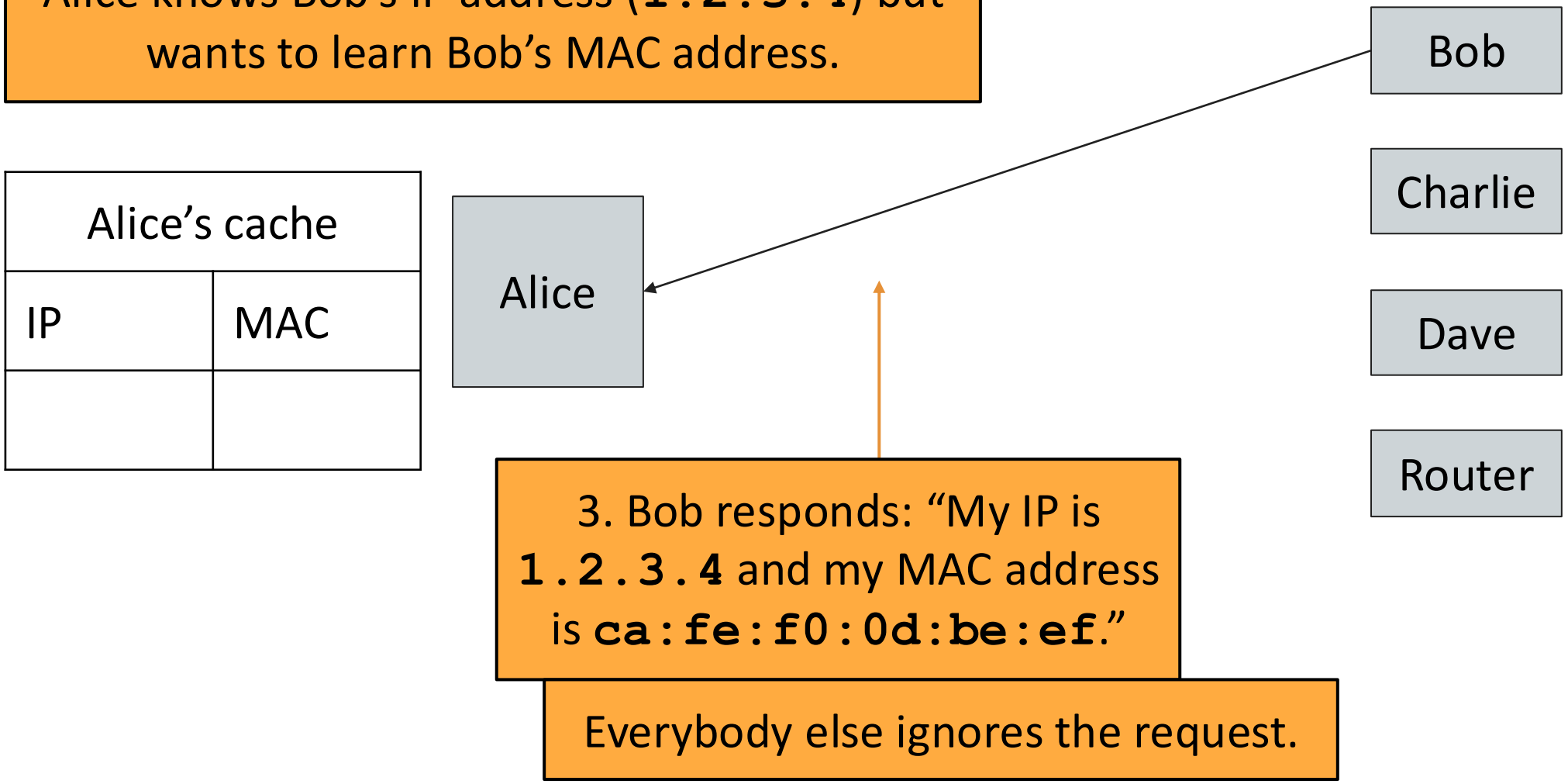
Address Resolution Protocol (ARP)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.



Address Resolution Protocol (ARP)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.



Address Resolution Protocol (ARP)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC
`1.2.3.4`	`ca:fe:f0:0d:be:ef`

Alice

Bob

Charlie

Dave

Router

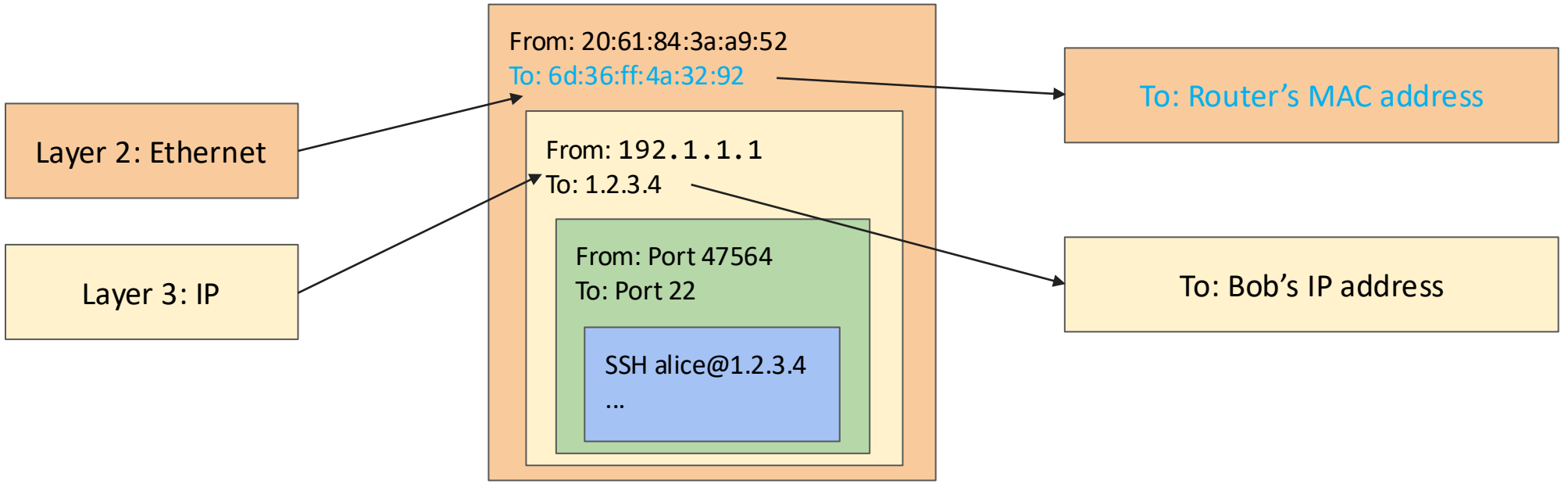
4. Alice adds Bob's MAC address to her cache.

NOTE: All received ARP replies are cached, even if no request was sent!!

Address Resolution Protocol (ARP)

If Bob is outside of the LAN, Alice uses router's MAC address as dest

- If Alice wants to send a packet to Bob, she sends the packet to the router
- The router can forward the packet to other LANs to reach Bob



Spoofing Attacks

- Anybody can send their own packets through the network
- **Spoofing**: Lying about the identity of a packet's sender
  - The attacker can lie about source addresses in the packet header
  - Example: Mallory sends a message and says the message is from Bob
- All 3 types of attackers can spoof packets
  - However, some spoofing attacks may be harder if the attacker can't read or modify packets

Spoofing Attacks on ARP (v1)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC

Alice

Bob

Charlie

Mallory

Router

1. Alice checks her cache to see if she already knows the MAC address corresponding to `1.2.3.4`.

Since her cache is empty, she must make a request to find out.

Attacks on ARP (v1)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

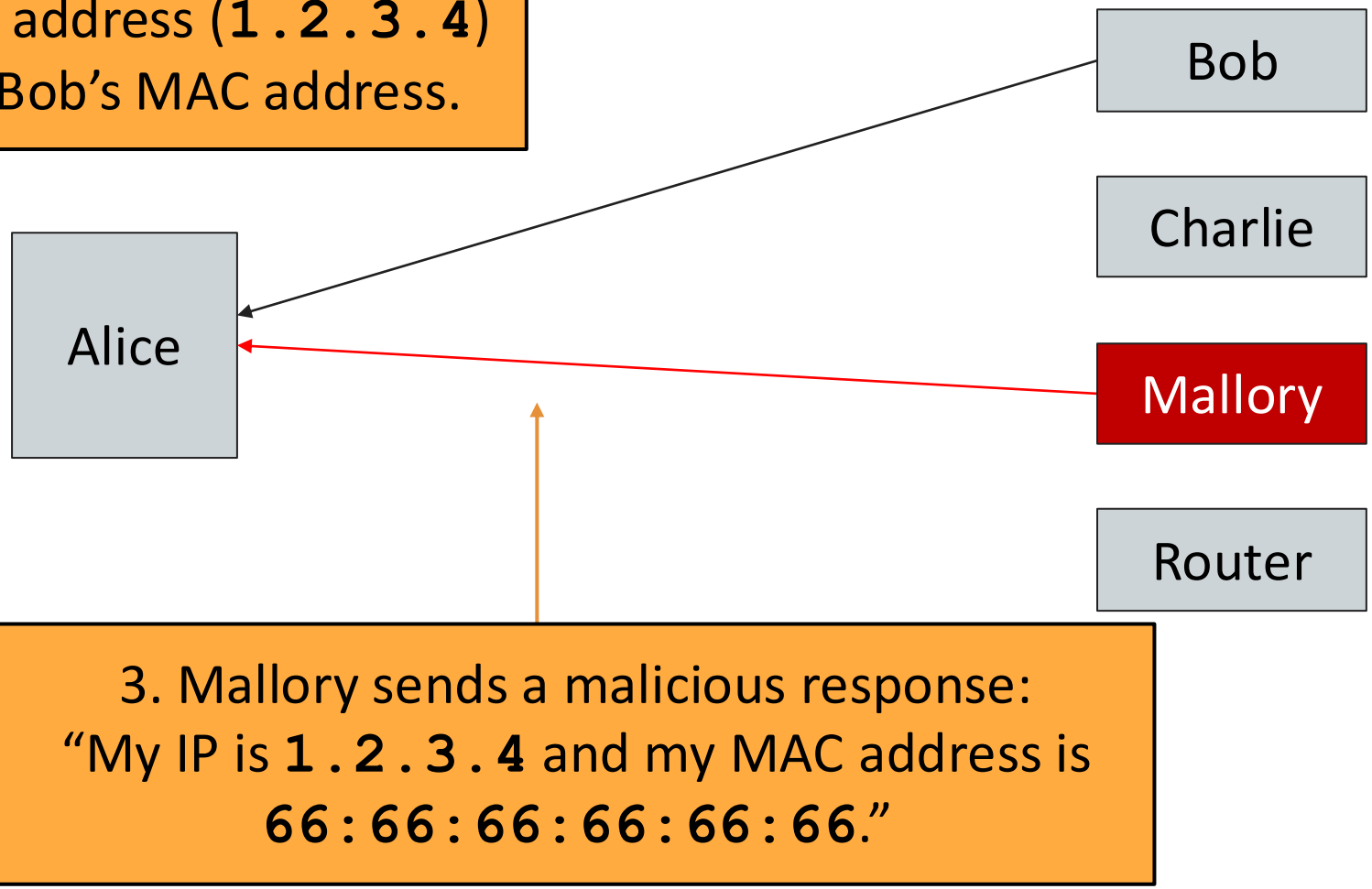
Alice's cache	
IP	MAC

2. Alice asks everyone else on the local network: "What is the MAC address of `1.2.3.4`?"

Attacks on ARP (v1)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC



Attacks on ARP (v1)

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC
`1.2.3.4`	`66:66:66:66:66:66`

Alice

Bob

Charlie

Mallory

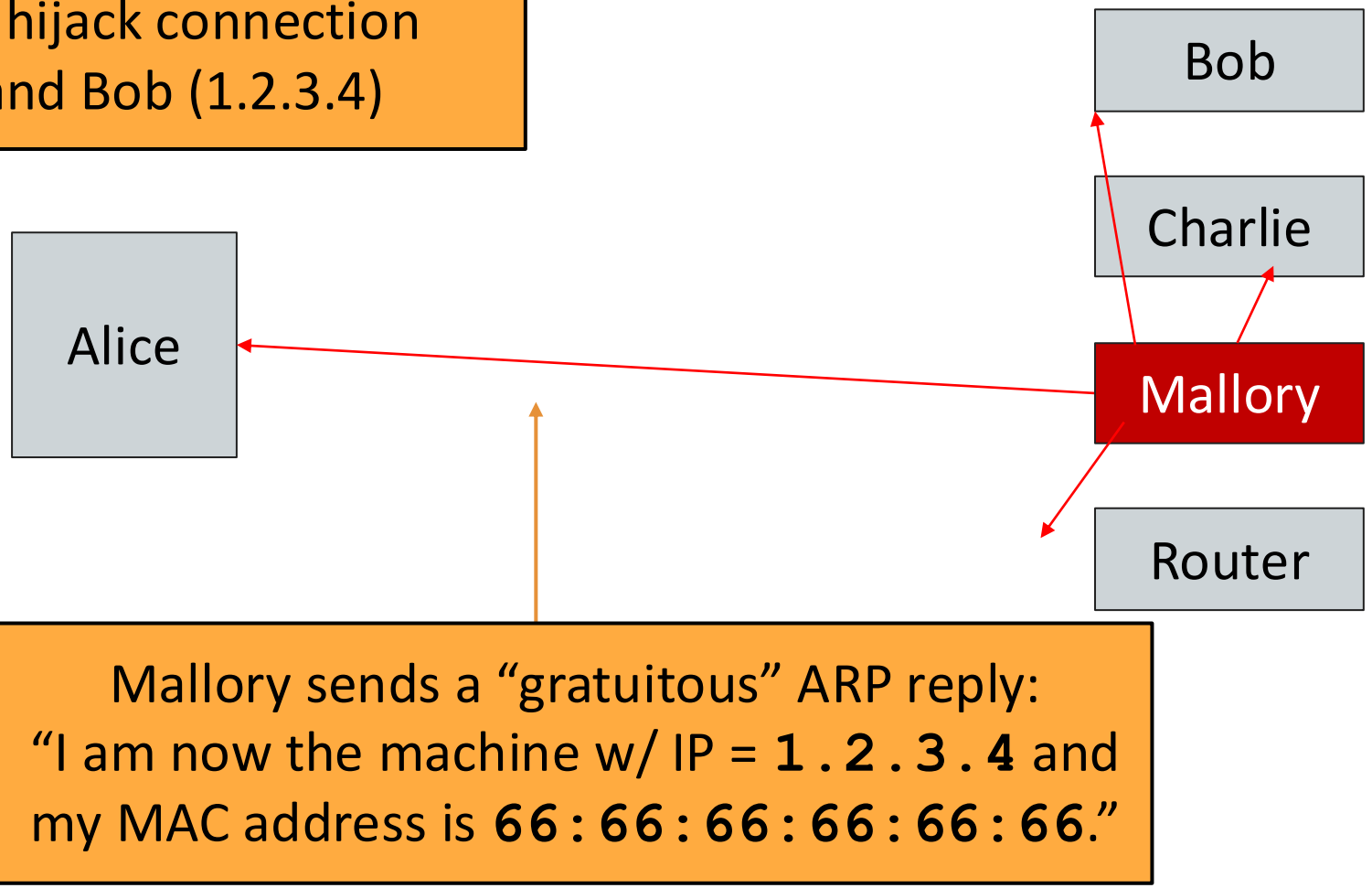
Router

4. Alice adds Mallory's malicious MAC address to her cache for Bob's IP addr!

Attacks on ARP (v2)

Mallory wants to hijack connection between Alice and Bob (1.2.3.4)

Alice's cache	
IP	MAC
`1.2.3.4`	`1:1:1:1`

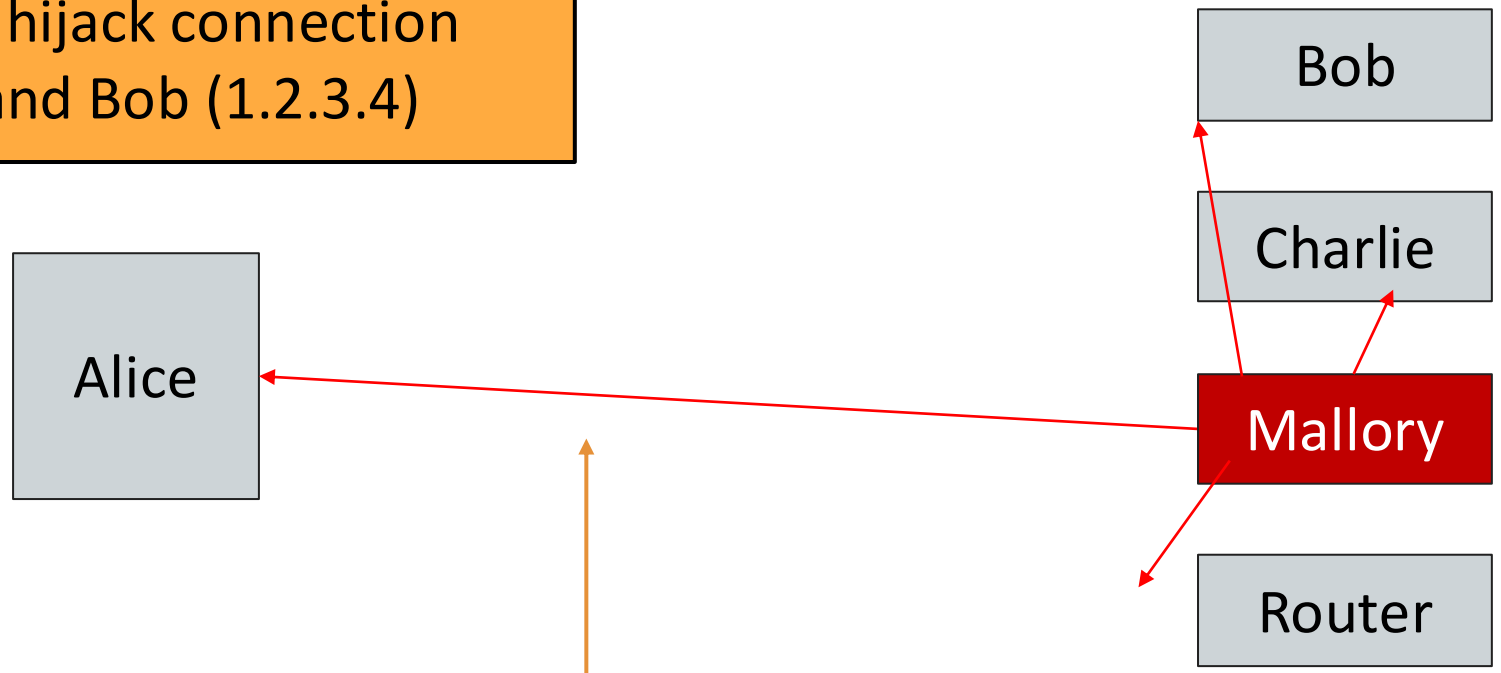


Mallory sends a "gratuitous" ARP reply:
"I am now the machine w/ IP = **`1.2.3.4`** and my MAC address is **`66:66:66:66:66:66`**."

Attacks on ARP (v2)

Mallory wants to hijack connection between Alice and Bob (1.2.3.4)

Alice's cache	
IP	MAC
`1.2.3.4`	`66:66:66:66:66:66`



Mallory sends a "gratuitous" ARP reply:
"I am now the machine w/ IP = **`1.2.3.4`** and my MAC address is **`66:66:66:66:66:66`**."

Attack: ARP Spoofing

- Alice has no way of verifying the ARP responses
- ARP protocol assumes that responses are accurate information and machines obey this assumption

- ARP spoofing requires Mallory to be in the same LAN as Alice
- ARP spoofing lets Mallory become a man-in-the-middle (MITM)
  - Alice thinks that Bob's MAC address is **`66:66:...:66`** (Mallory's MAC address)
  - When Alice sends a message to Bob, she is actually sending the msg to Mallory
  - Mallory can modify the message and then send the modified message to Bob

ARP Spoofing: Defenses

- Repeater Hubs vs. Switches
- Use **switches** to avoid broadcasts and ARP requests
  - When Alice wants to msg Bob, she sends the msg to a switch on the LAN
  - The switch maintains a cache of IP <-> MAC mappings
  - If Bob's MAC address in the cache, the switch sends the msg directly to Bob
  - Otherwise, the switch broadcasts the message
- Benefits of switches
  - Efficiency: Fewer broadcast requests
  - Security: Reduces the number of messages broadcast to the entire LAN & isolation: can create "virtual" VLANs in software (guest Wifi vs. main Wifi)

Dynamic Host Configuration Protocol (DHCP)

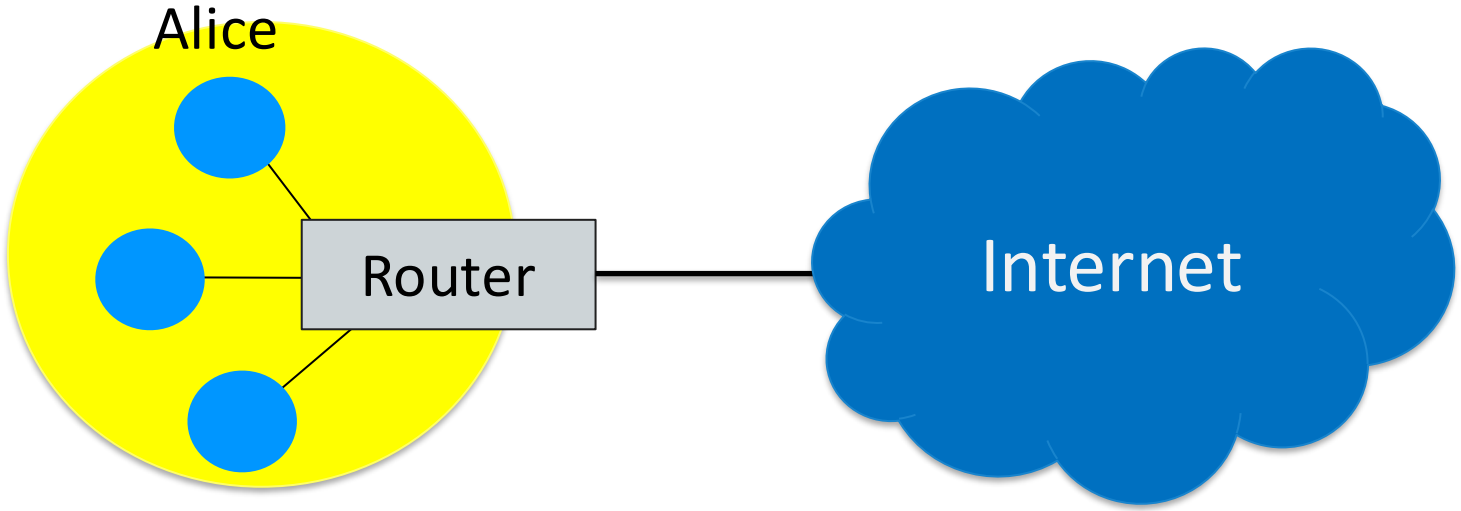
Coffee Shop

You go to a Coffee shop and want to use the Wifi

Wait… I don't have an IP address to send/recv data. *How do I get one?*

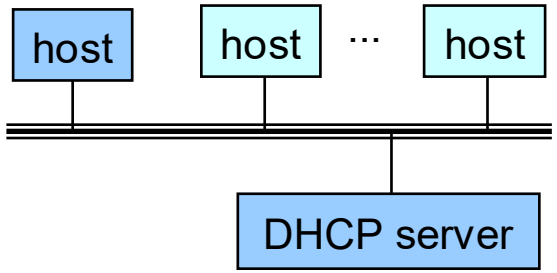
DHCP: Initial Network Configuration

- To connect to a network, a user needs:
  - An IP address so that other people can contact the user
  - The IP address of the router (gateway) so that the user can contact machines outside of the LAN
  - The IP address of the DNS server (IP <-> hostname resolution)



DHCP: Initial Network Configuration

- New host connecting to network doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know *who to ask* for an IP address
  - So, host doesn't know what destination address to use
- Solution: *shout* to "**discover**" server that can help
  - Broadcast a server-discovery message
  - DHCP Server(s) sends a reply offering an address & config details



DHCP = Dynamic Host Configuration Protocol

Steps of the DHCP Handshake (Protocol)

1. **Client Discover:** The client broadcasts a request for a configuration
2. **DHCP Offer**: Any DHCP server can respond with a configuration offer
   - Usually only one DHCP server responds, but can be multiple
   - The offer includes an IP address for the client & additional info (DNS server + gateway)
   - The offer also has an expiration time (how long the user can use this configuration)
3. **Client Request**: The client broadcasts which configuration it has chosen
   - If multiple DHCP servers made offers, the ones that weren't chosen discard their offer
   - The chosen DHCP server gives the offer to the client
4. **DHCP Acknowledgement**: The chosen server confirms that its configuration has been given to the client

Dynamic Host Configuration Protocol (DHCP)

Alice's configuration	
My IP	???
DNS Server	???
Gateway	???

Alice

Bob

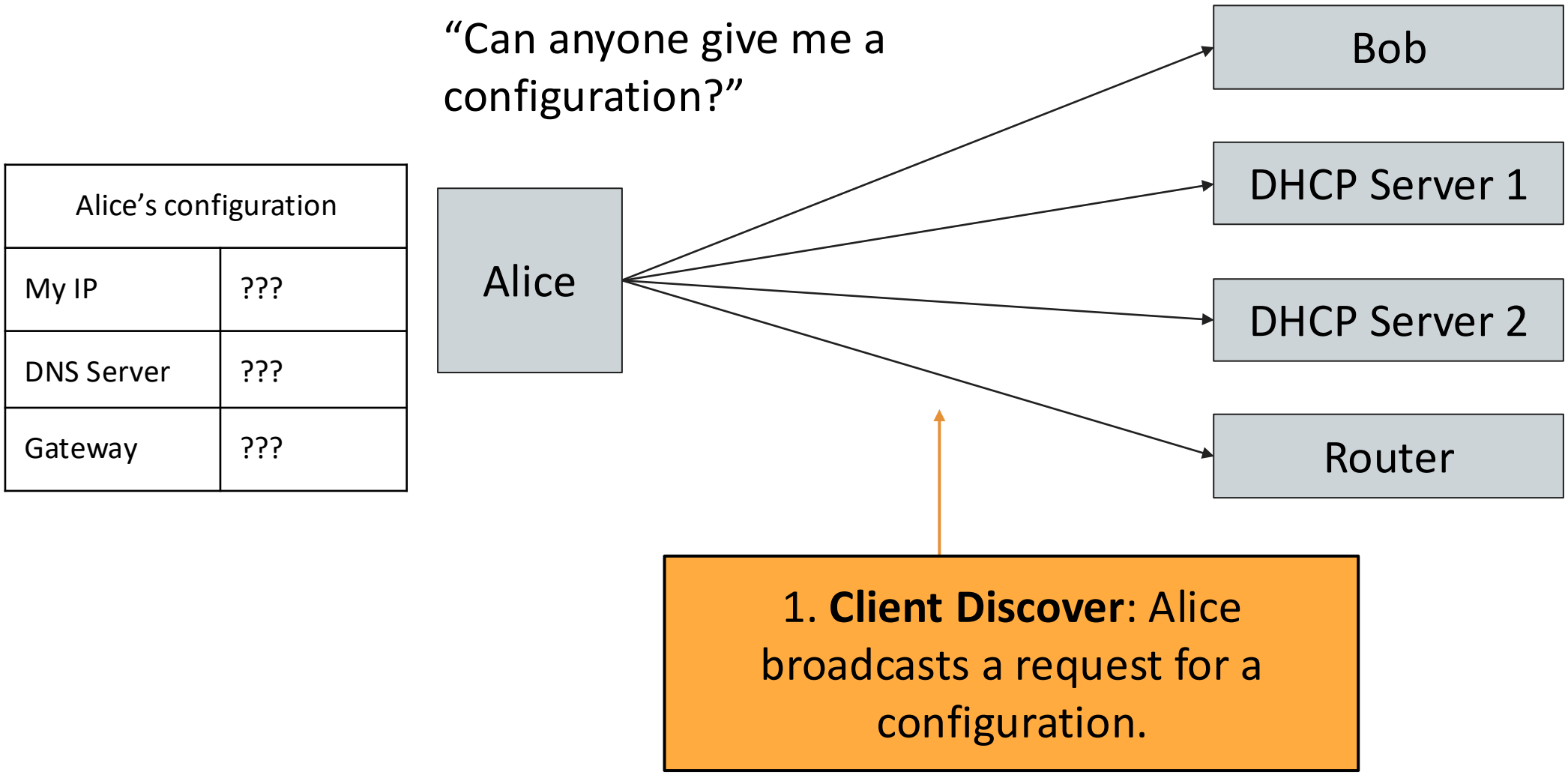
DHCP Server 1

DHCP Server 2

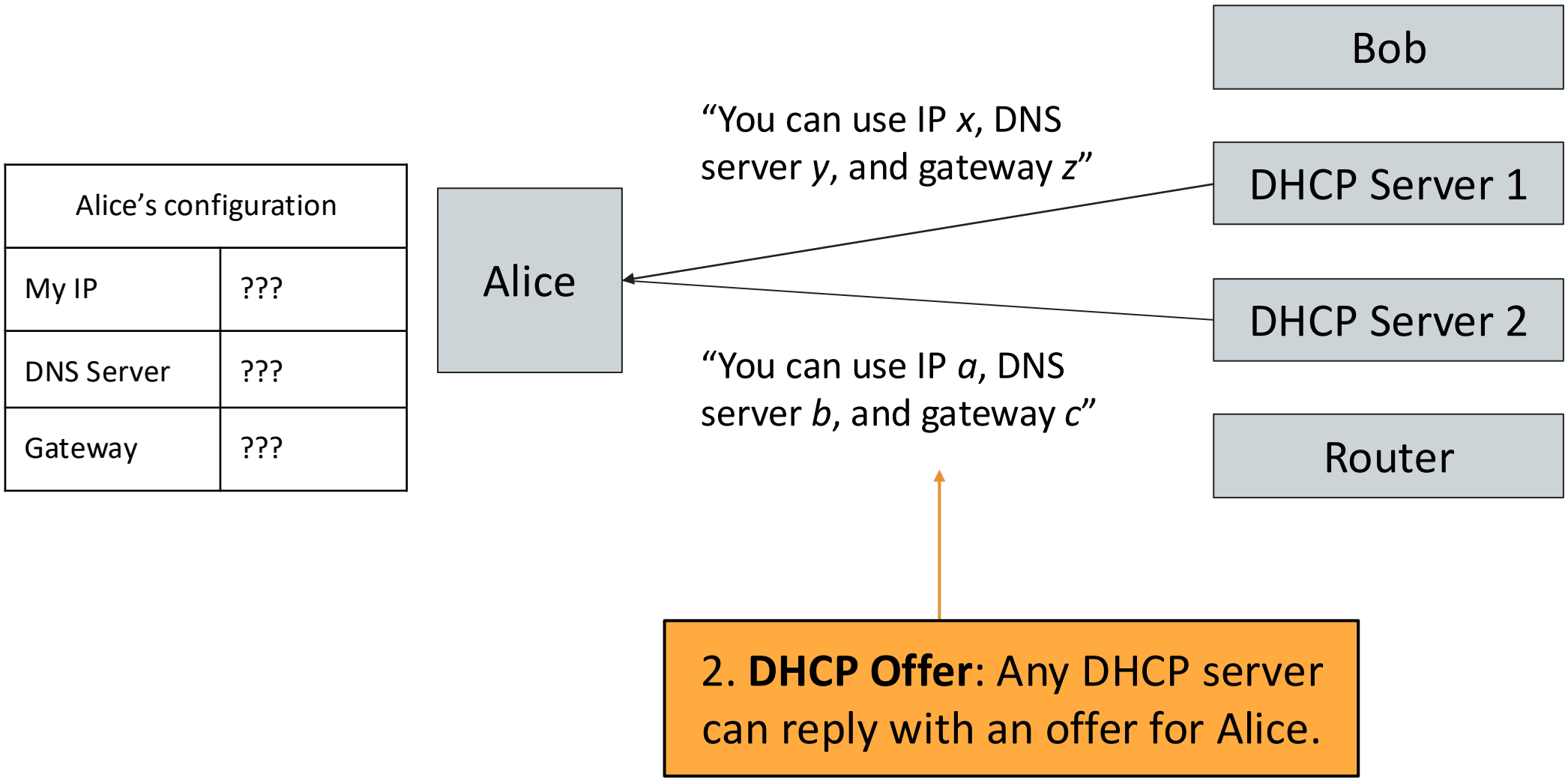
Router

Alice wants to connect to the network, but she's missing a configuration.

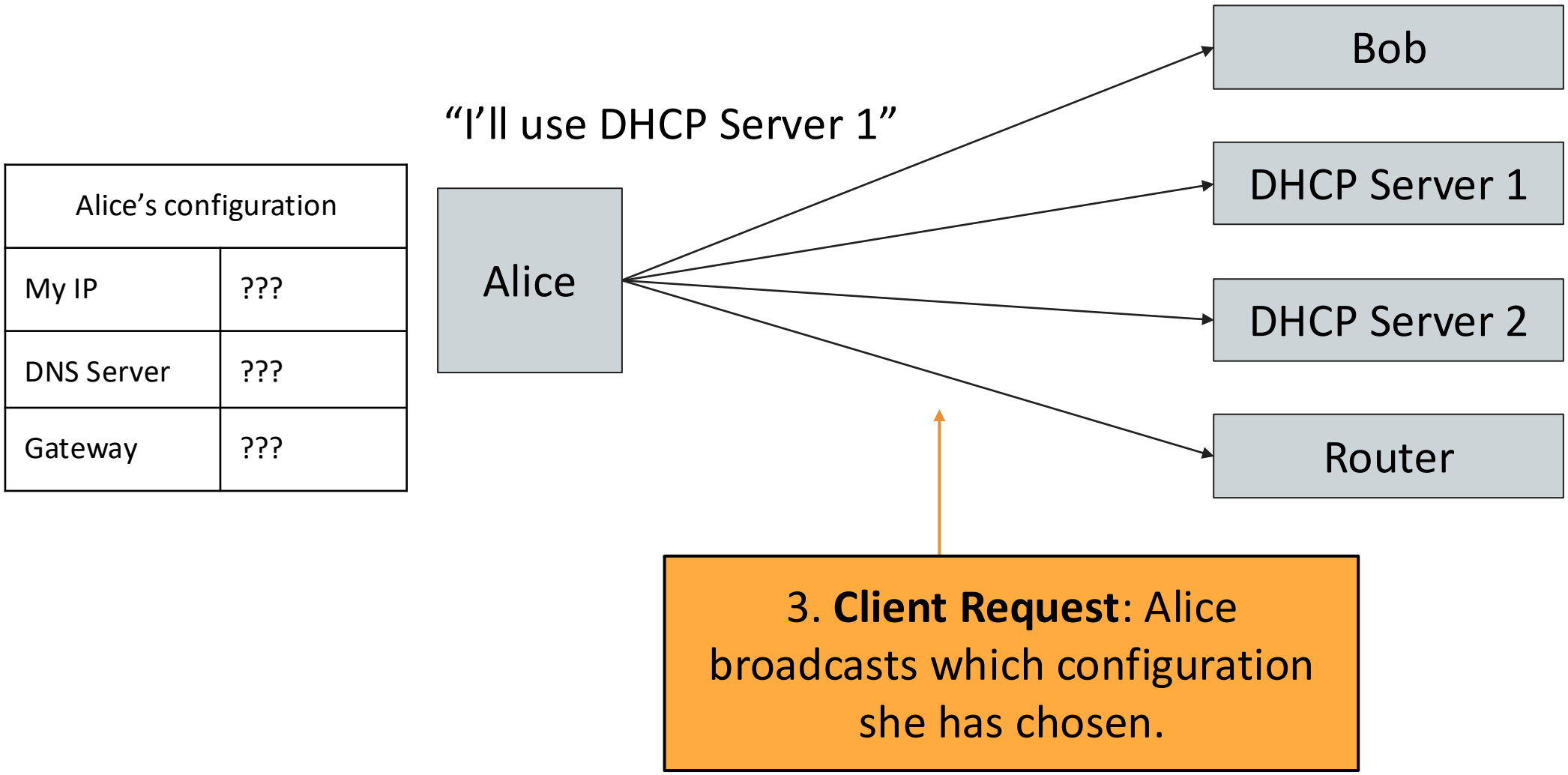
Dynamic Host Configuration Protocol (DHCP)



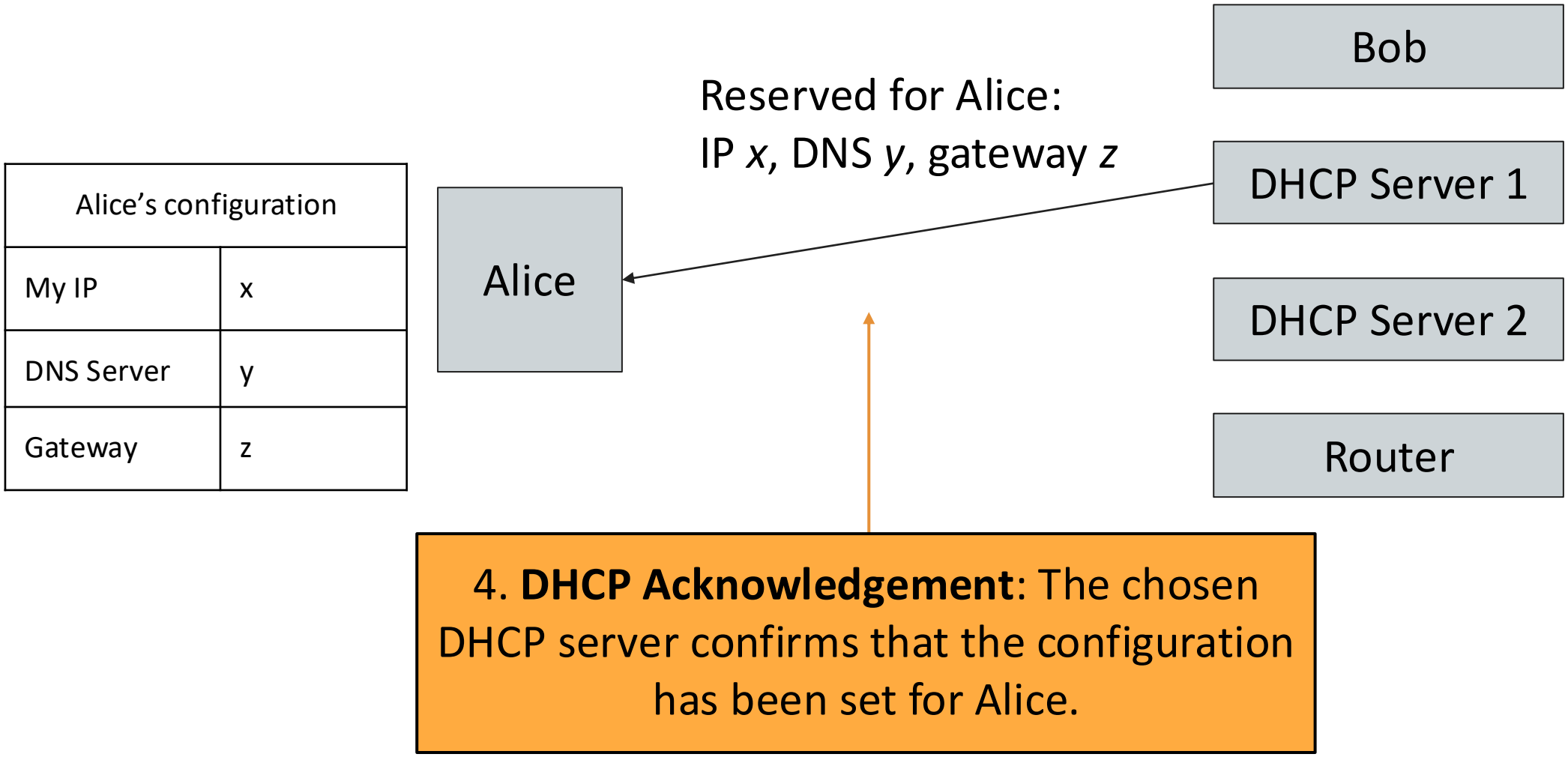
Dynamic Host Configuration Protocol (DHCP)



Dynamic Host Configuration Protocol (DHCP)



Dynamic Host Configuration Protocol (DHCP)



DHCP Attacks

Alice has no way of verifying the DHCP response

- Spoofing: Any attacker on the local network can claim to have a configuration (Step 2)

Alice usually expects only one DHCP server to respond, so she will accept the first response

- **Race condition**: If attacker responds faster, Alice accepts the attacker's response

DHCP attacks require Mallory (attacker) to be in the same LAN as Alice

DHCP attacks let Mallory become a man-in-the-middle (MITM) attacker

- e.g., Mallory claims the gateway router's address is Mallory's address
  - When Alice tries to send a msg to rest of Internet, she actually sends it to Mallory
  - Mallory can modify the message before sending it to its destination

DHCP Defenses

- DHCP attacks are hard to defend against
- No root of trust: When we first connect, there's nobody we can trust
- Instead, we rely on defenses provided in higher layers (e.g., TLS)

ARP and DHCP

- The attacks on ARP and DHCP are very similar
  - Spoofing: The attacker claims to have an answer
  - Elevate on-path attackers to in-path (MITM) attackers
- Main vulnerabilities
  - Broadcast protocols: Requests are sent to everyone on the LAN, so the attacker can see every request
  - No trust anchor: There is no way to verify that responses are legitimate

Outline

- ARP/DHCP Security
- WPA2: Wifi security
- TCP/UDP Attacks
- DNS Security

Wi-Fi

- **ARP & DHCP:** Protocols for getting configuration / address info once a machine has joined a LAN
- **Wi-Fi**: layer 2 protocol that wirelessly connects machines in a LAN (sending packets b/t machines on a LAN)
  - Alternative is Ethernet, which uses wires to connect machines in a LAN
- Parts of a Wi-Fi network
  - **Access point**: A machine that will help you connect to the network
  - **SSID**: The name of the Wi-Fi network
  - Password: Optionally, a password to secure Wi-Fi communications

WPA2: Wi-Fi Security

- **Wi-Fi Protected Access 2** (**WPA2**): A protocol for securing Wi-Fi network communications with cryptography

- Design goals
  - Everyone with the Wi-Fi password can join the network
  - An attacker who ***does not know the Wi-Fi password*** cannot read or tamper with data sent over Wi-Fi
  - Messages sent over the network are encrypted with keys

WPA Handshake (Symmetric Key Sharing)

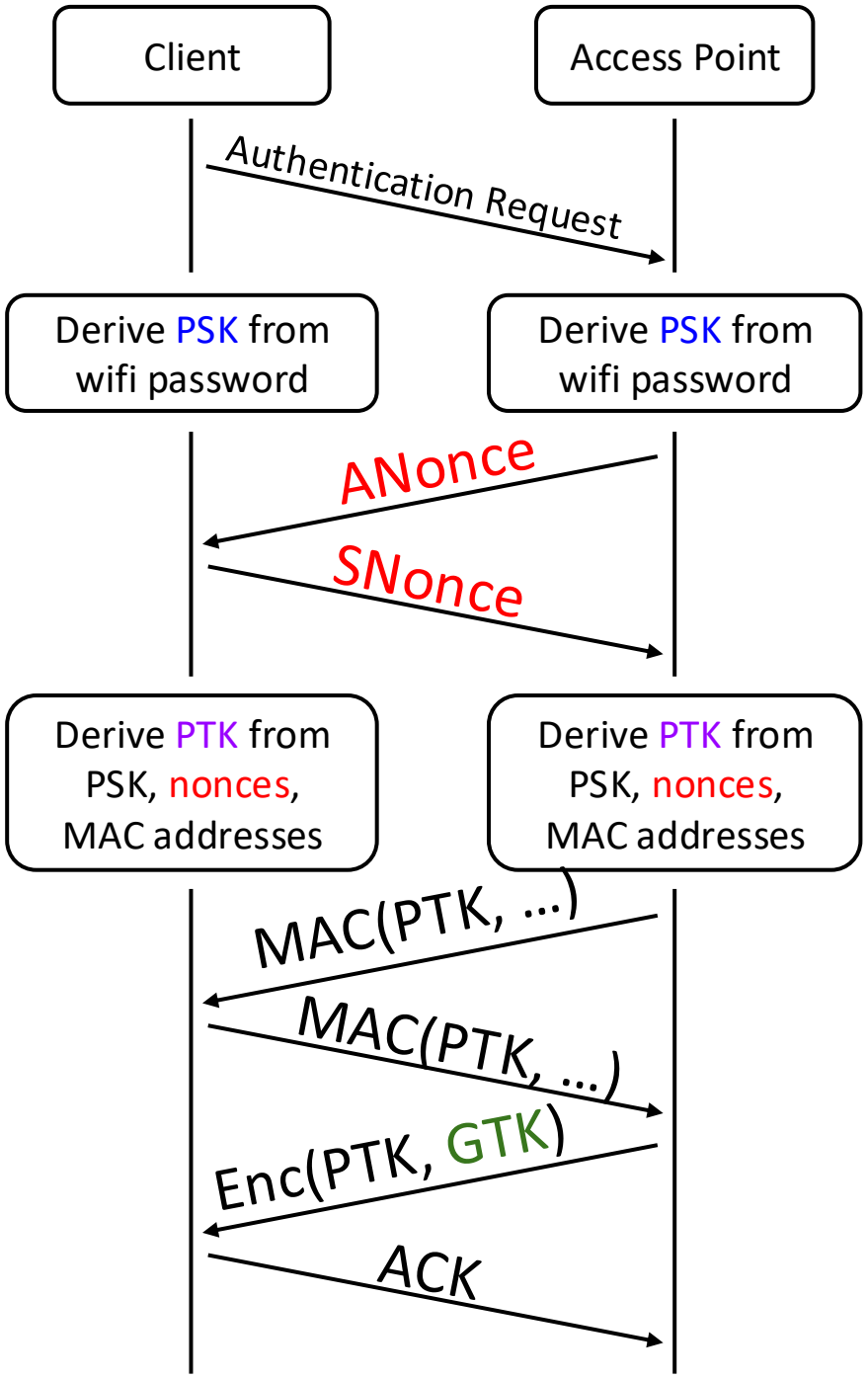
Password: $secret!

Client

Access Point

WPA Handshake (Symmetric Key Sharing)

1. The client sends an authentication request to the access point
2. Both use the password to derive the *PSK* (pre-shared key)
3. Both exchange random nonces
4. Both use the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
5. Both exchange MACs to ensure no one has tampered with the nonces, and that the PTK was correctly derived
6. The access point encrypts and sends the *GTK* (group temporal key) to the client, used for broadcasts that anyone can decrypt
7. The client acknowledges receiving the GTK

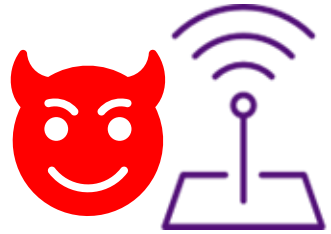


WPA Handshake: Establishing Shared Secret Key

- Both sides derive secret keys (*PTK*) for communication
  - Wi-Fi password → PSK
  - PSK + nonces + MAC addresses → PTK
  - The PTK is used to encrypt and authenticate all future communication
  - Note: The PTK is different for every user session, because of the nonces

- The access point encrypts and sends the *GTK* to the client
  - The GTK is used for messages broadcast to the entire network (e.g., ARP, DHCP, etc.)
  - Everyone on the network uses the same GTK

WPA-PSK Attacks



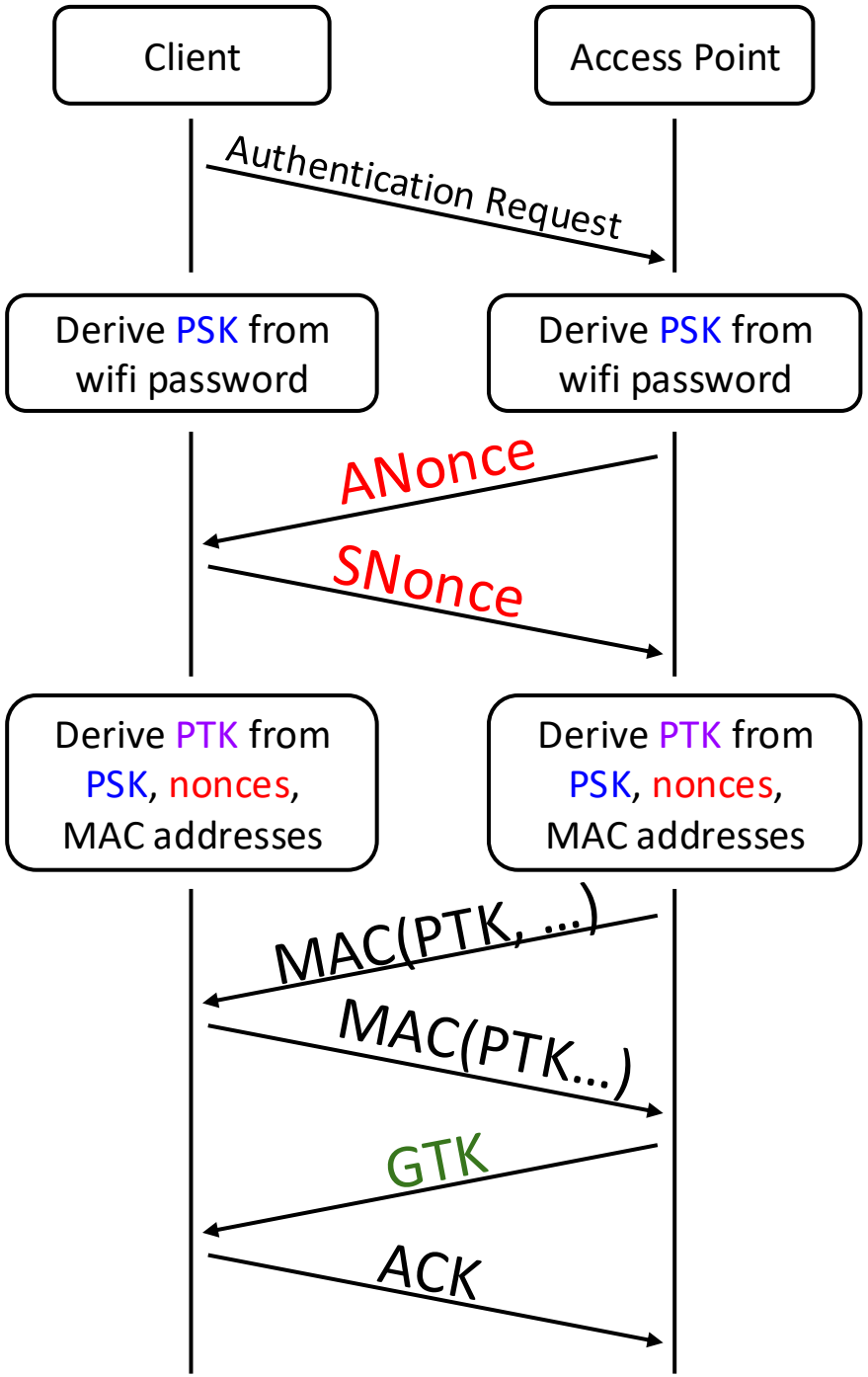
SSID: ORDWifi
Password: $secret!

Rogue Access Point (AP):
Pretend to be a valid AP, once client connects -> MITM Attacker

WPA-PSK Attacks

Eavesdropper already on Wi-Fi:

- All Wi-Fi msg's are broadcast
- Attacker can derive all keys & therefore decrypt/MAC everything correctly:
  - Wi-Fi password → *PSK*
  - Nonces are broadcast → *PTK*

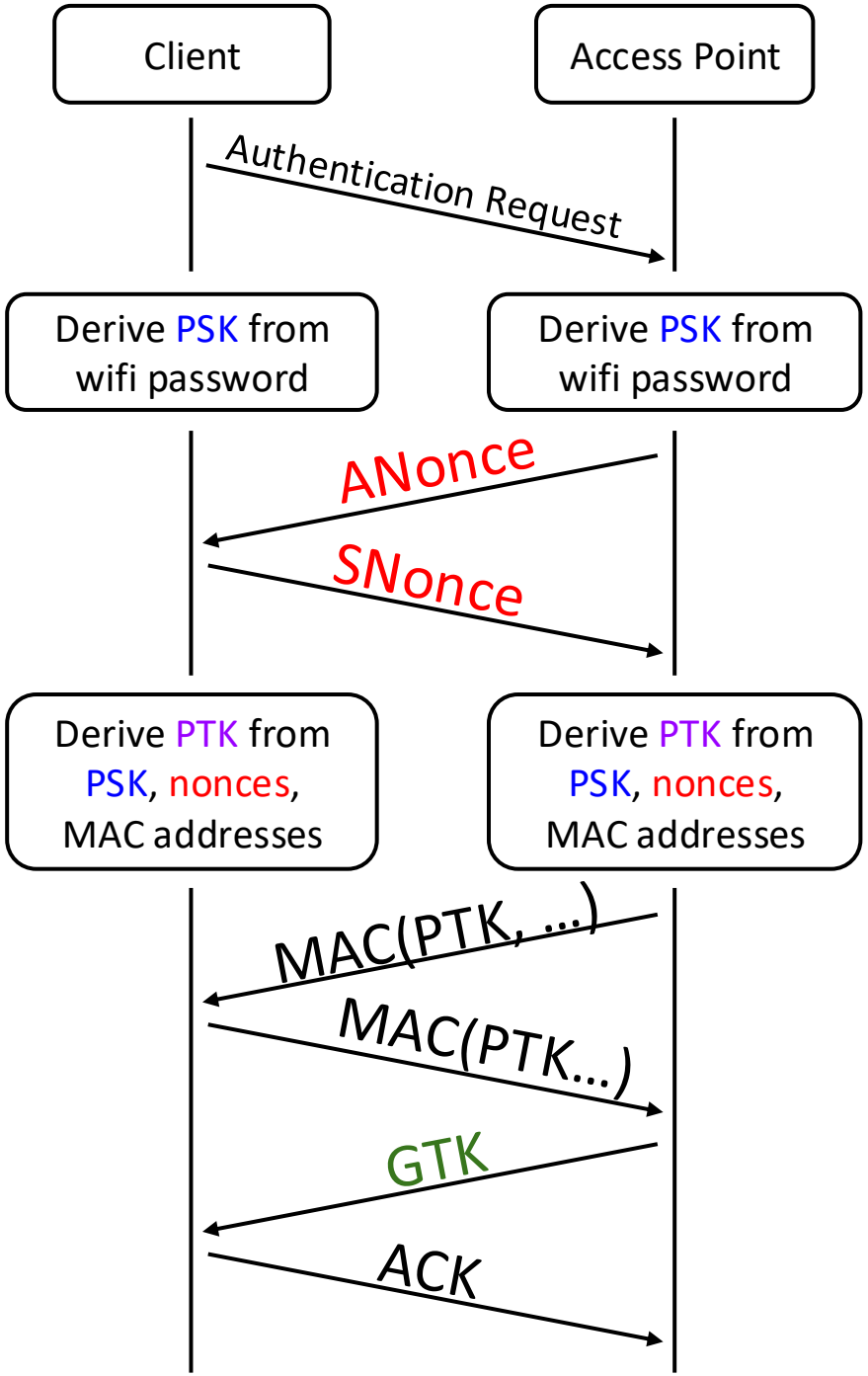


WPA-PSK Attacks

What about an attacker who doesn't know the Wi-Fi password?

Offline brute-force attack: People tend to choose bad passwords -> attacker can store past info to check if they guessed it

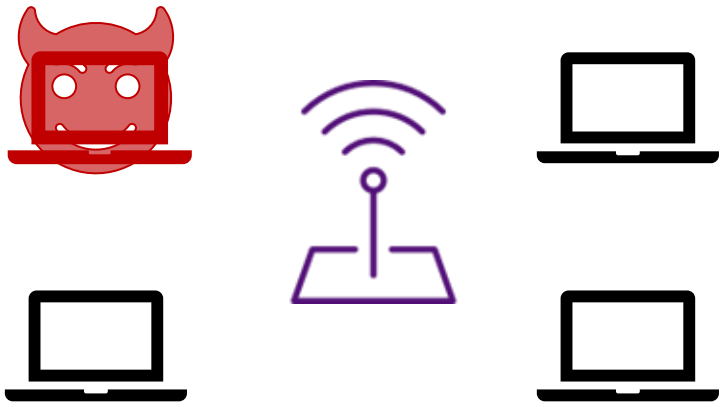
- Nonces are sent unencrypted, and client and AP MAC addresses are public
- Eavesdropper guesses a password and derives:
  - Wi-Fi password → *PSK*
  - *PSK* + nonces + MAC addresses → *PTK*
  - Eavesdropper checks: MAC from the guess matches the MAC that was sent?

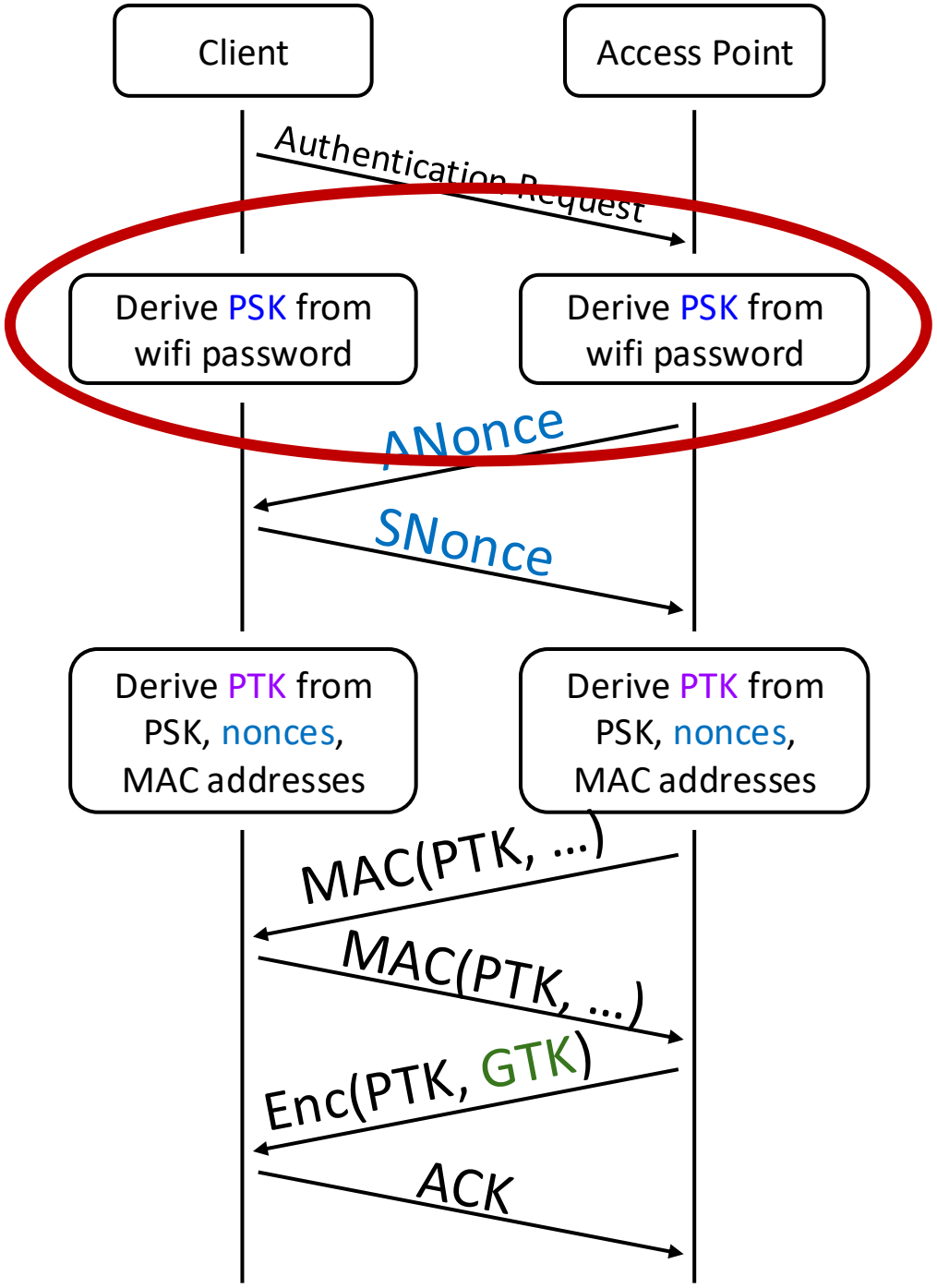


WPA Weakness

- **Problem:** Everyone uses the same shared Wi-Fi password

- **Therefore:** an attacker can generate the symmetric keys (PTK) for everyone they see joining the network





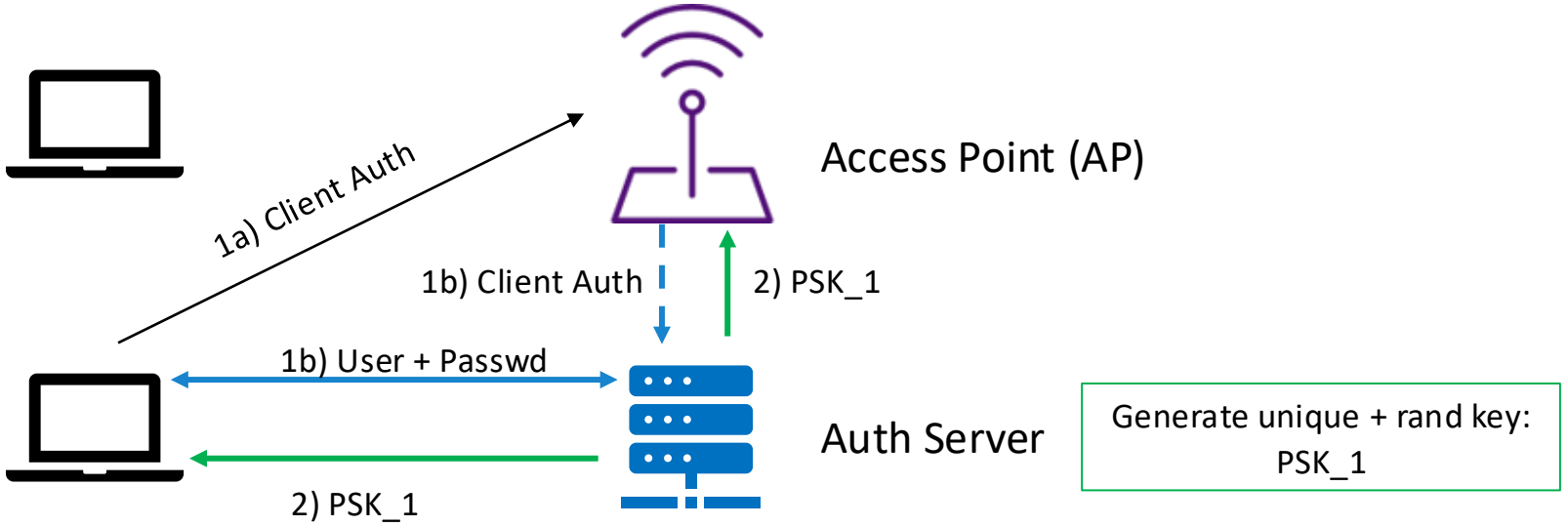
WPA-Enterprise

- Core issue: Every client starts w/ the same *PSK* to derive *PTK*
  - Fix: Have each user use their own unique username + password
  - This is the model that eduroam use!
- Instead of using a PSK from common password, use a key generated (unique per user) randomly by an auth server
  - Authentication server has a digital certificate to prove legitimacy
  - Your machine first forms a secure channel to the authentication server, which lets you enter your username and password
  - If the username + password correct, auth server sends a one-time key to use instead of a PSK to both the client and the AP (over a separate secure channel)
- The rest of the handshake proceeds normally

WPA Enterprise

Solution: Generate unique Wi-Fi session keys based on user accounts

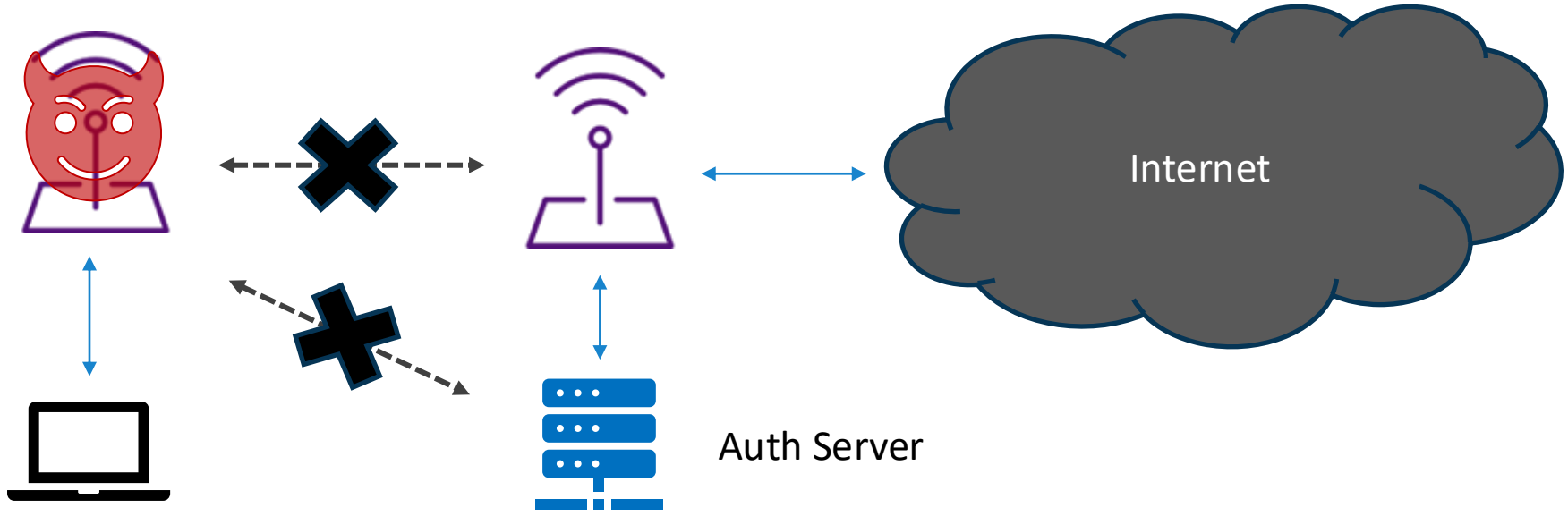
- Add additional authentication server (w/ a certificate verifying identity)
- AP configured to have a secure channel w/ Auth Server during setup
- Users now authenticate to the auth server when connecting to Wi-Fi
- If auth succeeds, auth server sends random key (PSK) to user & AP



WPA-Enterprise Attacks

WPA Enterprise defends against:

- **Rogue AP attack [if unauthorized user]**: The APs must authenticate themselves, which the attacker can't do (so attacker & user have no network access!)



WPA-Enterprise Attacks

WPA Enterprise defends against:

- **Rogue AP attack [if unauthorized user or no client validation of cert]**: The APs must authenticate themselves to the auth server, which the attacker can't do
- **Other Wi-Fi users:** Every user has unique PSK & derives unique session keys (PTK's)
- **Brute-force attack**: The generated PSK is too long & random to brute-force

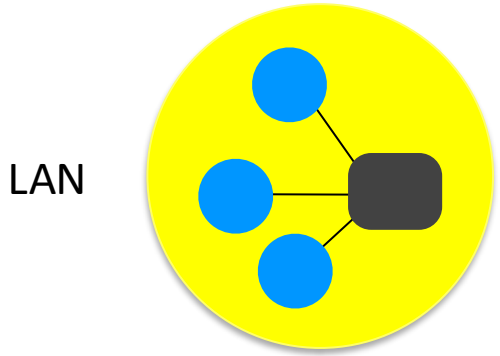
However, still vulnerable to (1) ARP or DHCP spoofing, and (2) Rogue AP's that have network access

– Can defend against Rogue AP w/ server-side cert validations; otherwise the malicious AP can just relay & MITM the user's network connections.

Outline

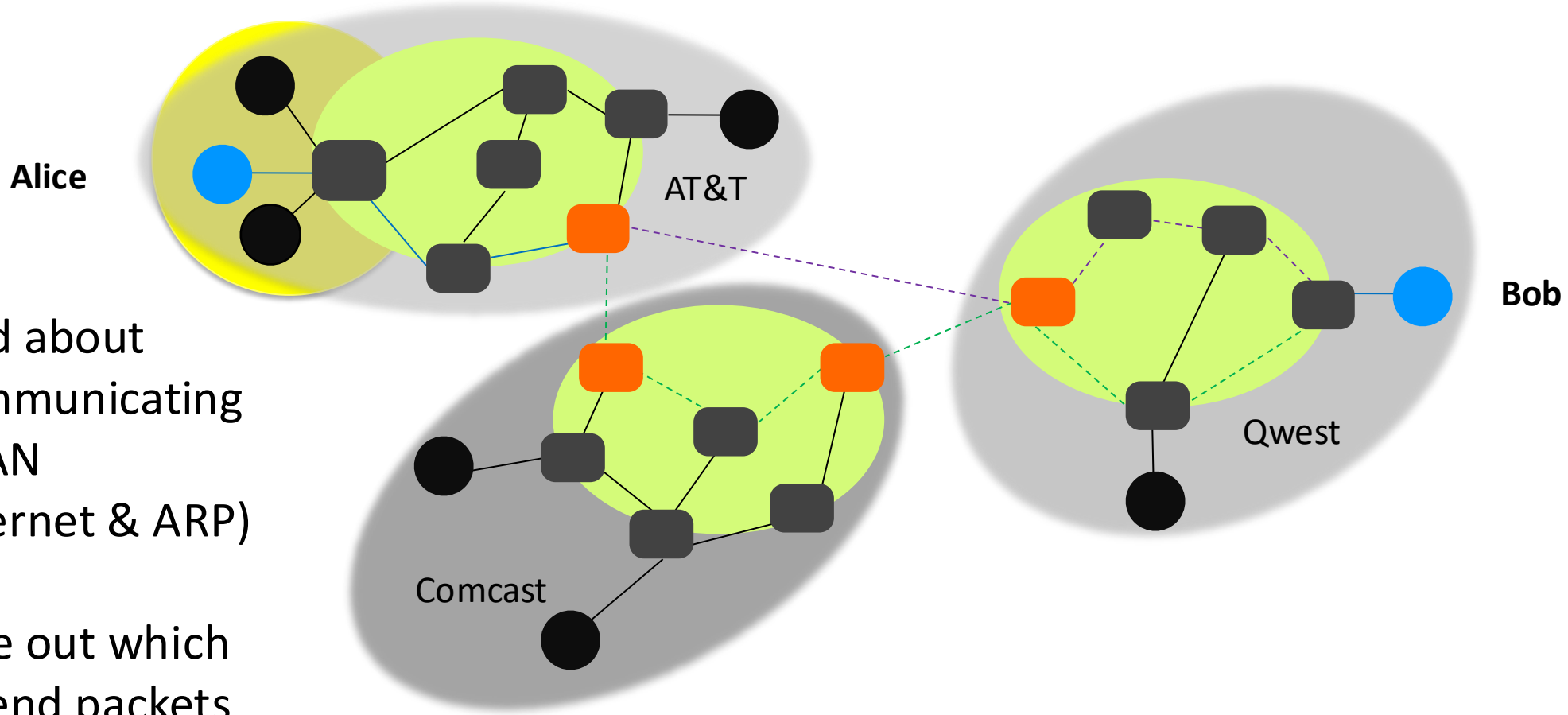
- ARP/DHCP Security
- WPA2: Wifi security
- TCP/UDP Attacks
- DNS Security

Brief Aside: BGP: Routing Across the Internet



Previously: talked about
protocols for communicating
within a single LAN
(e.g., Wi-Fi / Ethernet & ARP)

BGP: Routing Across the Internet



Previously: talked about protocols for communicating within a single LAN (e.g., Wi-Fi / Ethernet & ARP)

How do we figure out which route (path) to send packets across the Internet?

Answer: BGP: Border Gateway Protocol (BGP)

BGP Prefix Hijacking

4/25/2019
02:30 PM

Marc Laliberte
Commentary

Connect Directly

0 COMMENTS
COMMENT NOW

Login

100% 0%

How a Nigerian ISP Accidentally Hijacked the Internet

For 74 minutes, traffic destined for Google and Cloudflare services was routed through Russia and into the largest system of censorship in the world, China's Great Firewall.

On November 12, 2018, a small ISP in Nigeria made a mistake while updating its network infrastructure that highlights a critical flaw in the fabric of the Internet. The mistake effectively brought down Google — one of the largest tech companies in the world — for 74 minutes.

To understand what happened, we need to cover the basics of how Internet routing works. When I type, for example, HypotheticalDomain.com into my browser and hit enter, my computer creates a web request and sends it to Hypothtetical.Domain.com servers. These servers likely reside in a different state or country than I do. Therefore, my Internet service provider (ISP) must determine how to route my web browser's request to the server across the Internet. To maintain their routing tables, ISPs and Internet backbone companies use a protocol called Border Gateway Protocol (BGP).

https://www.darkreading.com/cloud/how-a-nigerian-isp-accidentally-hijacked-the-internet/a/d-id/1334482

BGP Defenses

Some attempts to mitigate Prefix Hijacking (S-BGP / BGP-Sec)

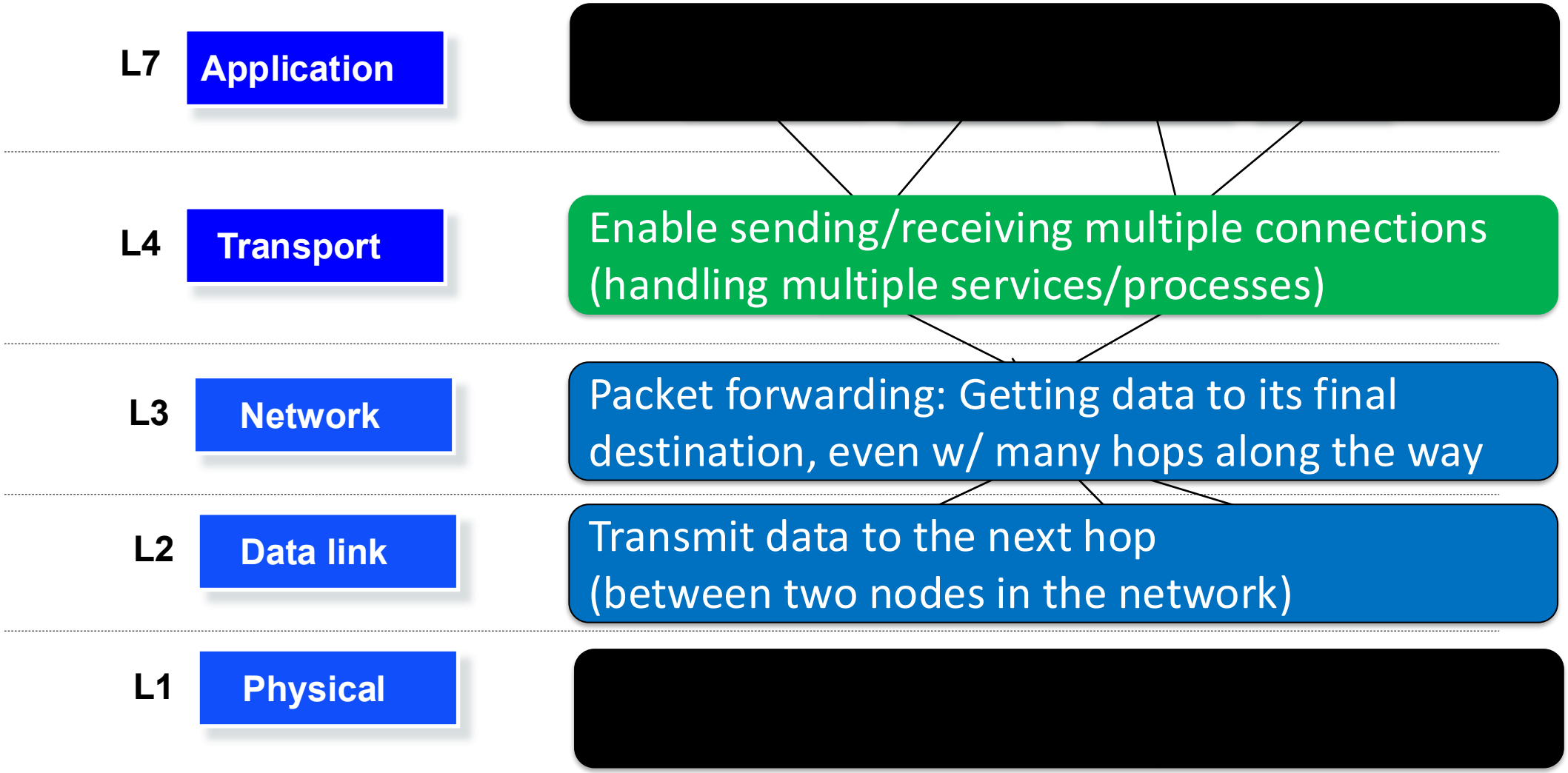
- Idea: Cryptographically sign routes
- Problems: Costly & Slow adoption

Instead: often rely on higher layer's security (e.g., TLS)

Outline

- ARP/DHCP Security
- WPA2: Wifi security
- TCP/UDP Attacks
- DNS Security

Network Stack (OSI Layers)



Transport Layer Goals

1. Handle multiple connections streams (ports) &
2. Get ALL of the data to its destination

UDP: User Datagram Protocol

UDP: Simple transport protocol that adds ports to support traffic multiplexing (multiple simultaneous connections)

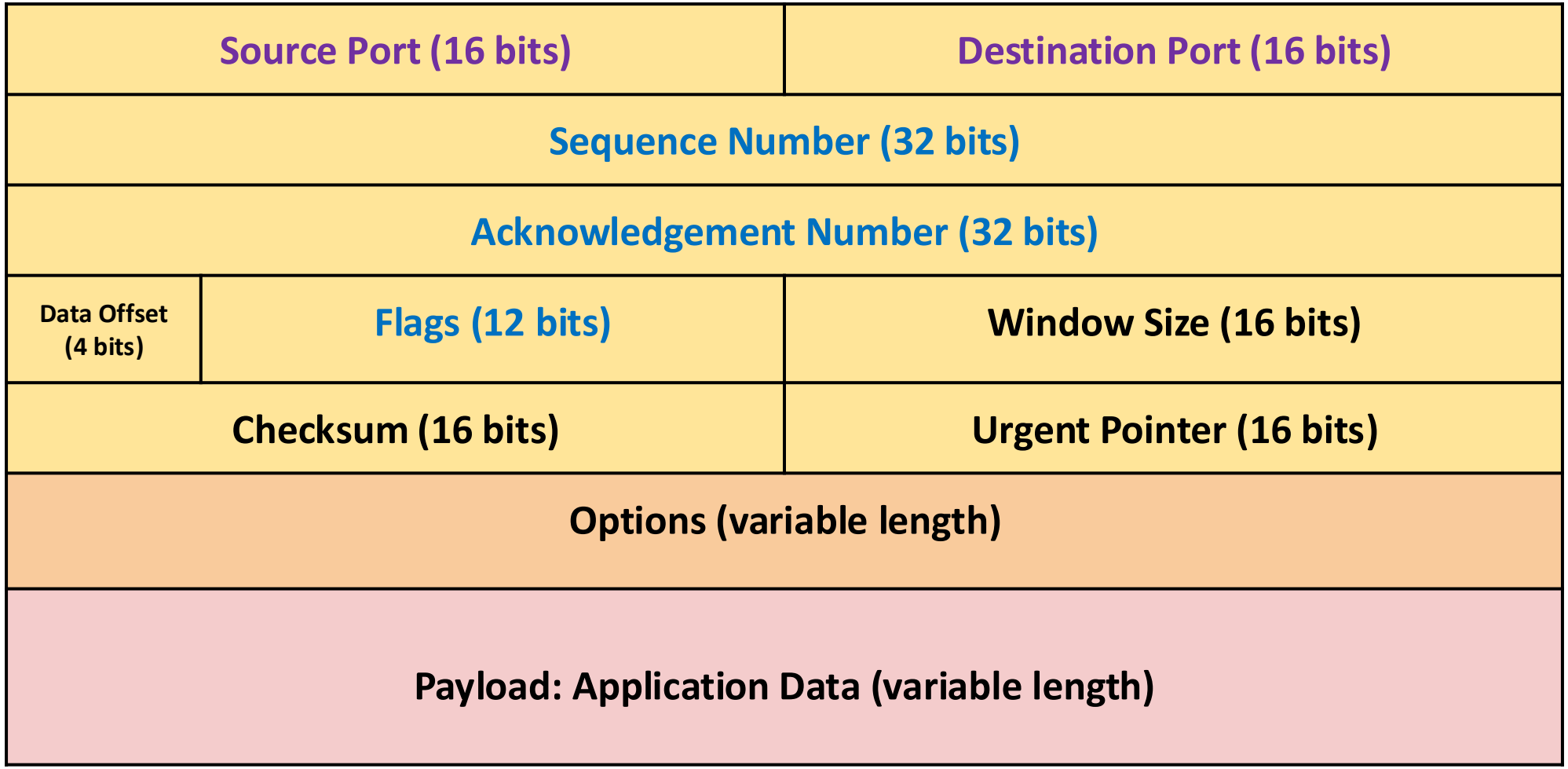
UDP datagram header + payload

Source Port (16 bits)	**Destination Port (16 bits)**
Length (16 bits)	**Checksum (16 bits)**
Payload: Application Data (variable length)	

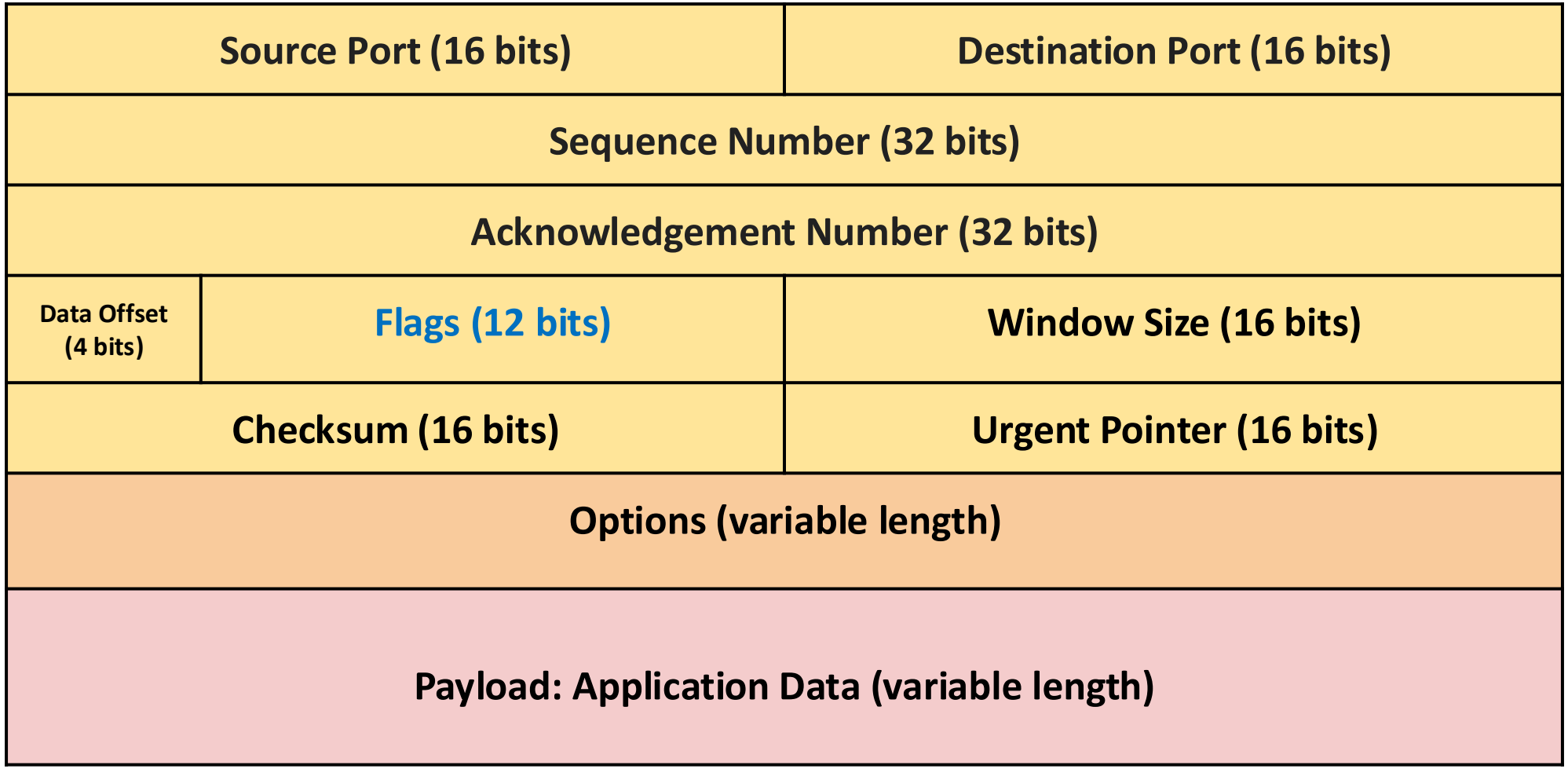
TCP: Reliable Data Streams

- Routing on the Internet is full of hazards: packets being dropped, re-ordered, and duplicated
  - Faulty router, shark nibbling on underwater cable, router power outage, etc.
- Most applications want a stream of bytes delivered reliably and in-order between different hosts
- Transmission Control Protocol (TCP) provides reliable + in-order byte streams, along with other services (e.g., congestion control)
  - Uses a combination of ***sequence numbers*** and ***flags*** to ensure reliable & controlled connections

TCP Packet Structure



TCP Packet Structure



TCP Sequence Numbers

- Two data streams in a TCP session, one in each direction
- Bytes in data stream numbered with a 32-bit sequence number
- Every packet has sequence number that indicates where data belongs
- Receiver sends acknowledgement number that indicates data received

H	e	l	l	o		s	e	r	v	e	r
50	51	52	53	54	55	56	57	58	59	60	61

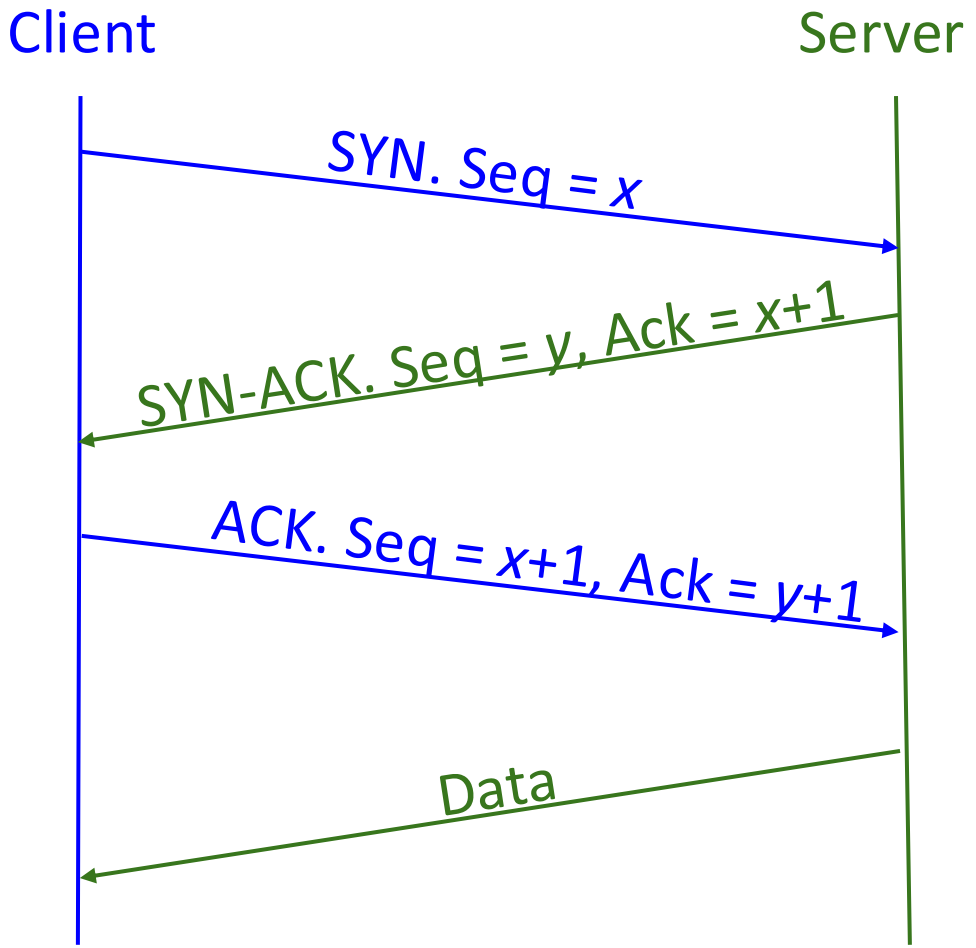
H	e	l	l	o		c	l	i	e	n	t
25	26	27	28	29	30	31	32	33	34	35	36

Example: Bytes from the client are numbered starting at 50.

Example: Bytes from the server are numbered starting at 25.

TCP Setup: 3-Way Handshake

1. Client chooses an random initial sequence number (ISN) *x*: sends a SYN (synchronize) packet to the server
2. Server chooses an random ISN *y* for bytes it will send and responds with a SYN-ACK packet
3. Client responds with an ACK packet
4. Once both hosts have synchronized sequence numbers, the connection is "established"



TCP: Sending and Receiving Data

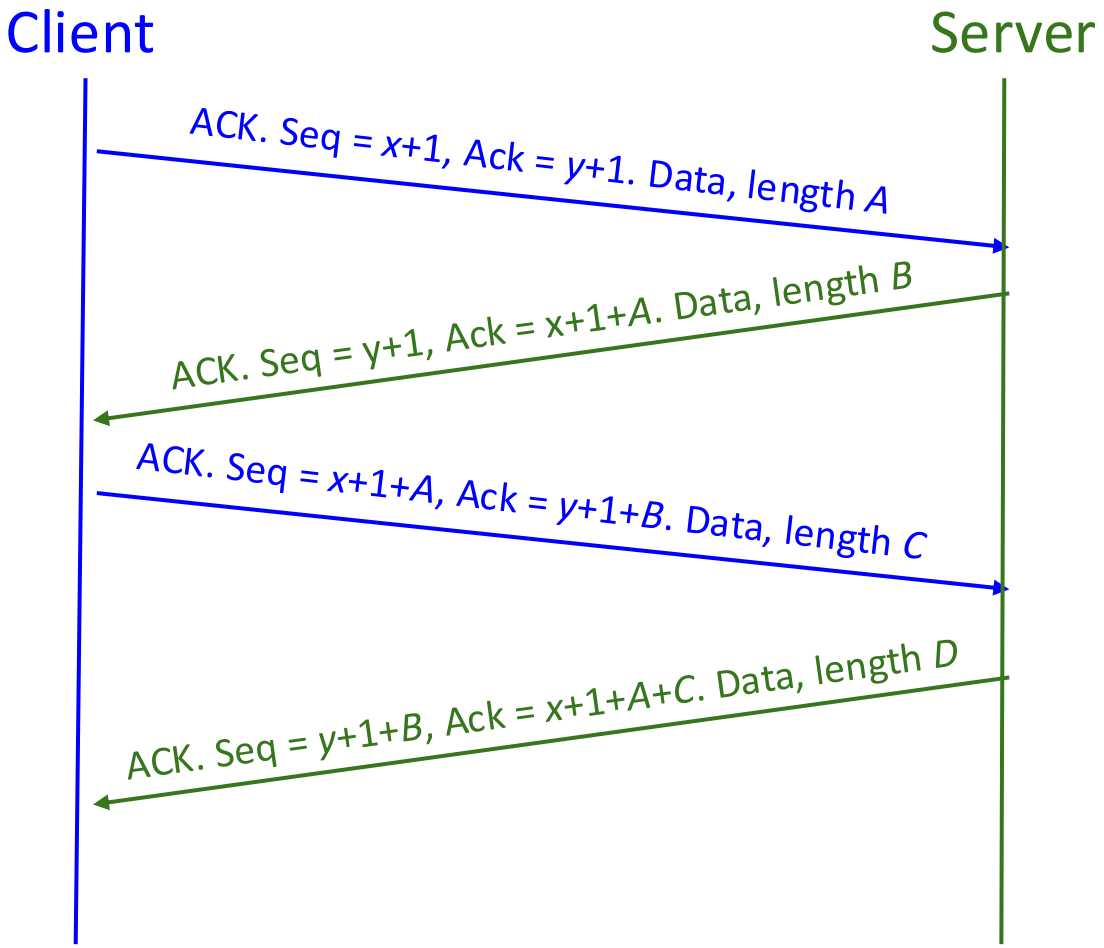
TCP headers use seq #'s to provide ordering and reliable delivery

A packet's sequence number = the sequence number of the first byte of its current data

- Byte *i* of the byte stream is represented by sequence number *x* + *i* (client) or *y* + *i* (server)

A packet's ACK number = the sequence number of the byte it expects to receive next

- Equal to (sequence number) + (length of data) for the last received packet



TCP: Retransmission (Reliable Delivery)

- If a packet is dropped (lost in transit):
  - *Recipient won't send an ACK, so the sender will not receive the ACK*
  - *The sender repeatedly tries to send the packet again until it receives the ACK*
- If a packet is received by recipient, but the ACK is dropped:
  - *Sender won't receive an ACK, so they try to send packet again*
  - *The recipient ignores the duplicate data and sends the ACK again*

TCP Flags

SYN

- Indicates the beginning of the connection

ACK

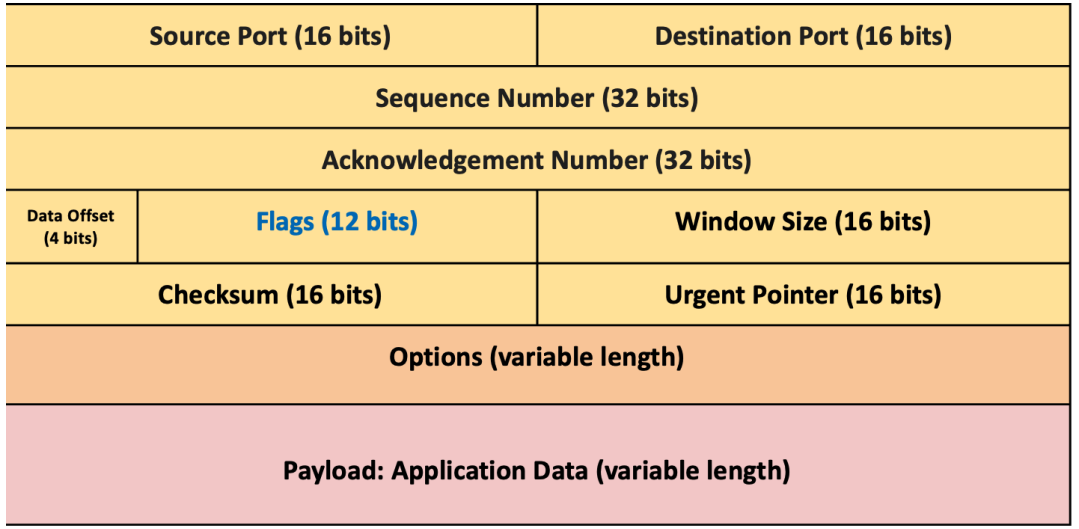
- Acknowledges the receipt of something (in the ack number)
- Pretty much always set except the very first packet

FIN

- One way to end the connection
- Requires an acknowledgement
- No longer sending packets, but will continue to receive

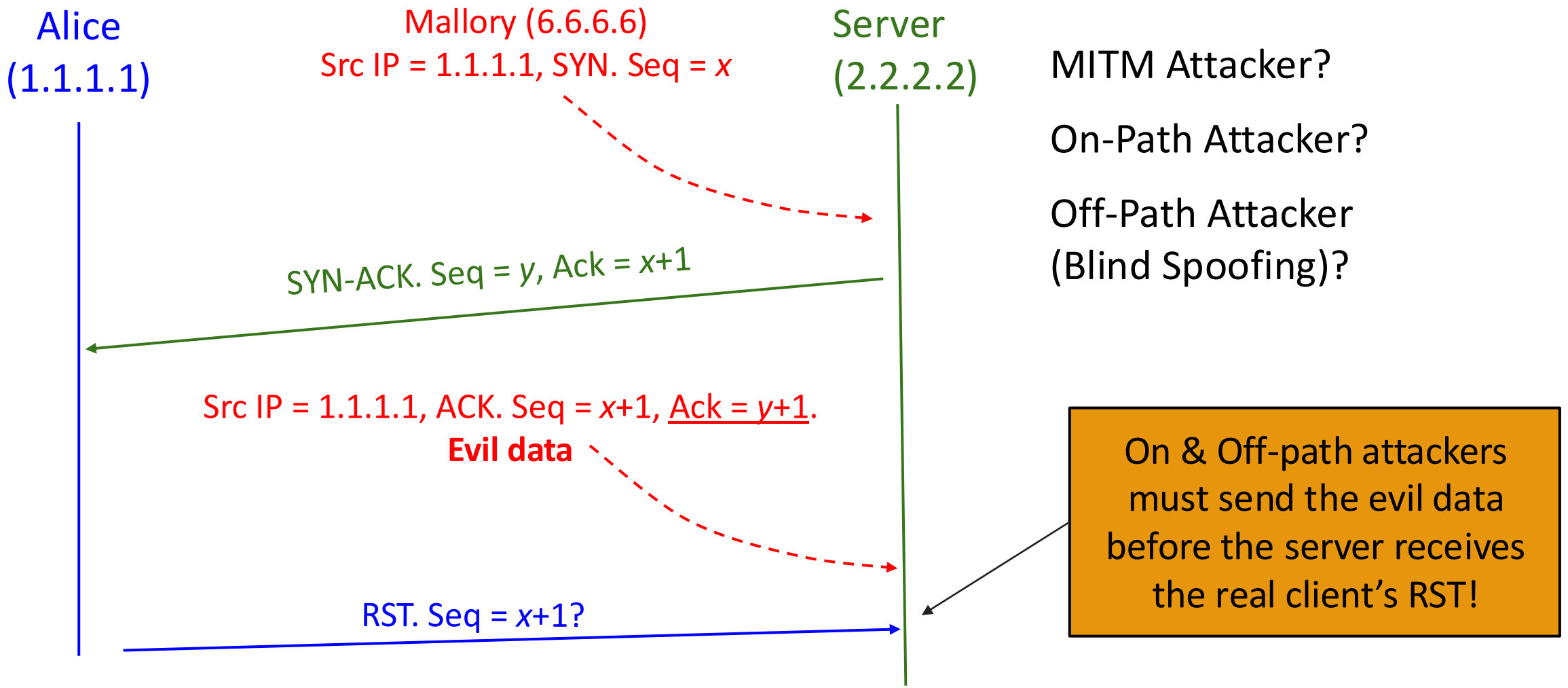
RST

- One way to end a connection
- Does not require an acknowledgement
- No longer sending or receiving packets



Attacks: TCP Spoofing

(Spoof connection to appear to come from someone else [Alice])



MITM Attacker?

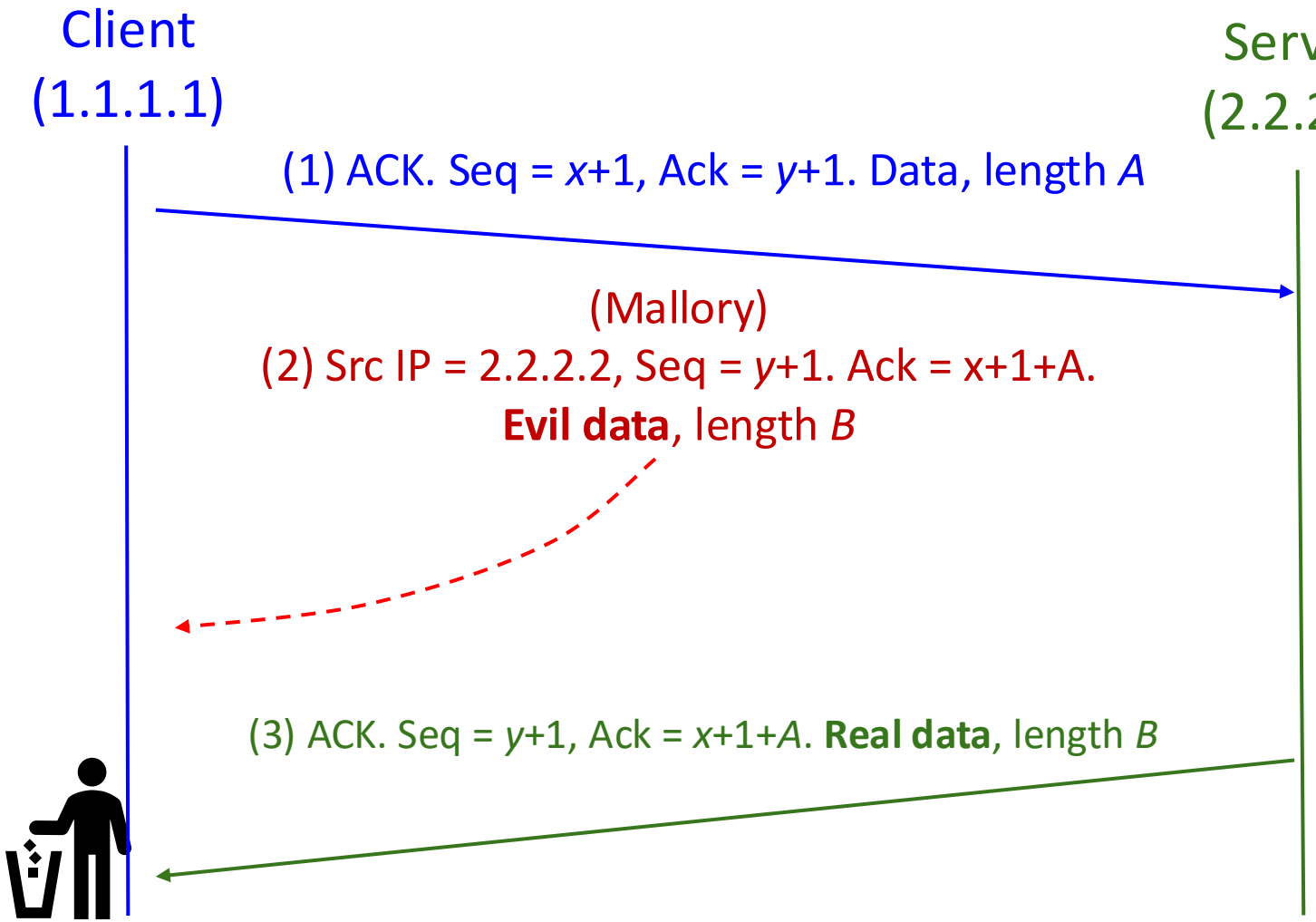
On-Path Attacker?

Off-Path Attacker (Blind Spoofing)?

On & Off-path attackers must send the evil data before the server receives the real client's RST!

Attacks: TCP Data Injection

(Tampering with an existing session to modify / inject data into a connection)



MITM Attacker?

On-Path Attacker?

Off-Path Attacker (Blind Injection)?

Packet (3) will be ignored by the client since the client already processed the malicious packet!

TCP & UDP Attacks

- TCP provides no confidentiality or integrity
  - Instead, we rely on higher layers (like TLS) to prevent those kind of attacks

- Defense against off-path attackers rely on choosing random sequence numbers
  - Bad randomness can lead to trivial off-path attacks:
    TCP sequence numbers used to be based on the system clock!

- UDP: Attacks even easier! No sequence numbers to guess/forge.

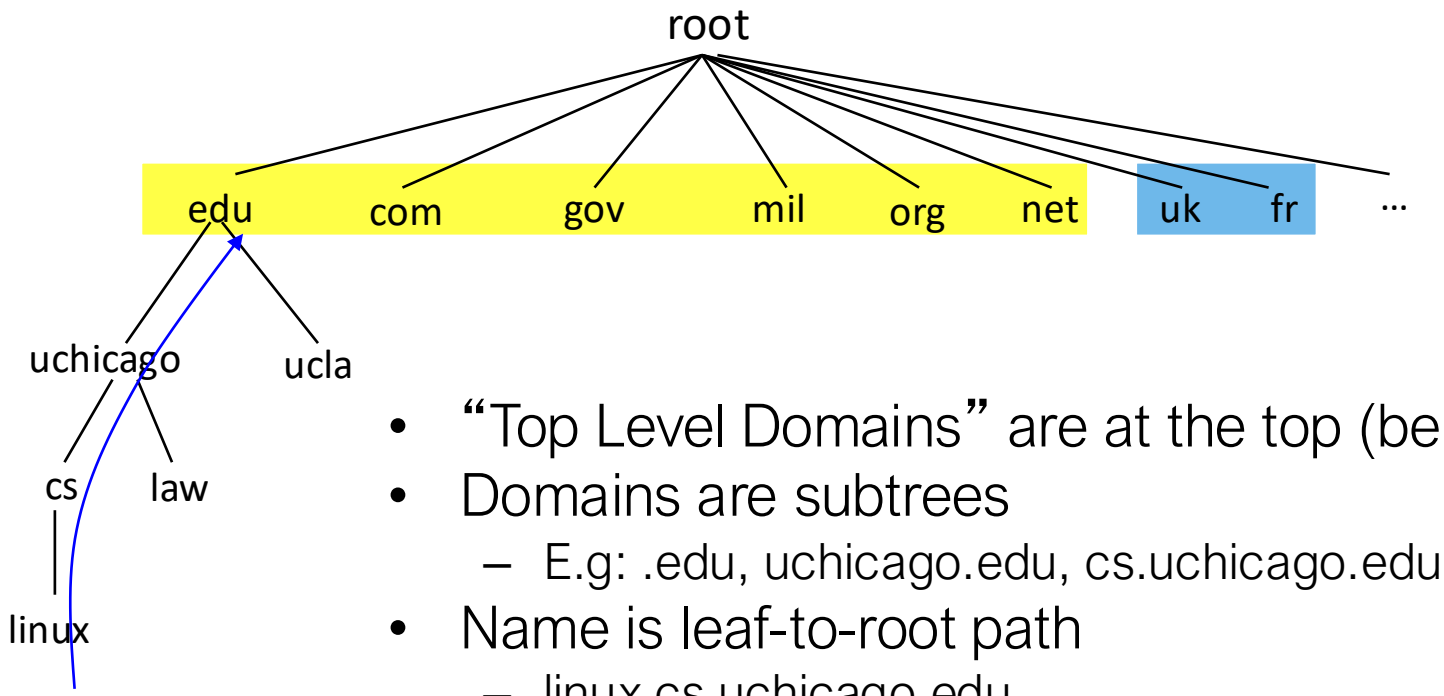
Outline

- ARP/DHCP Security
- WPA2: Wifi security
- TCP/UDP Attacks
- DNS Security

DNS (Domain Name System)

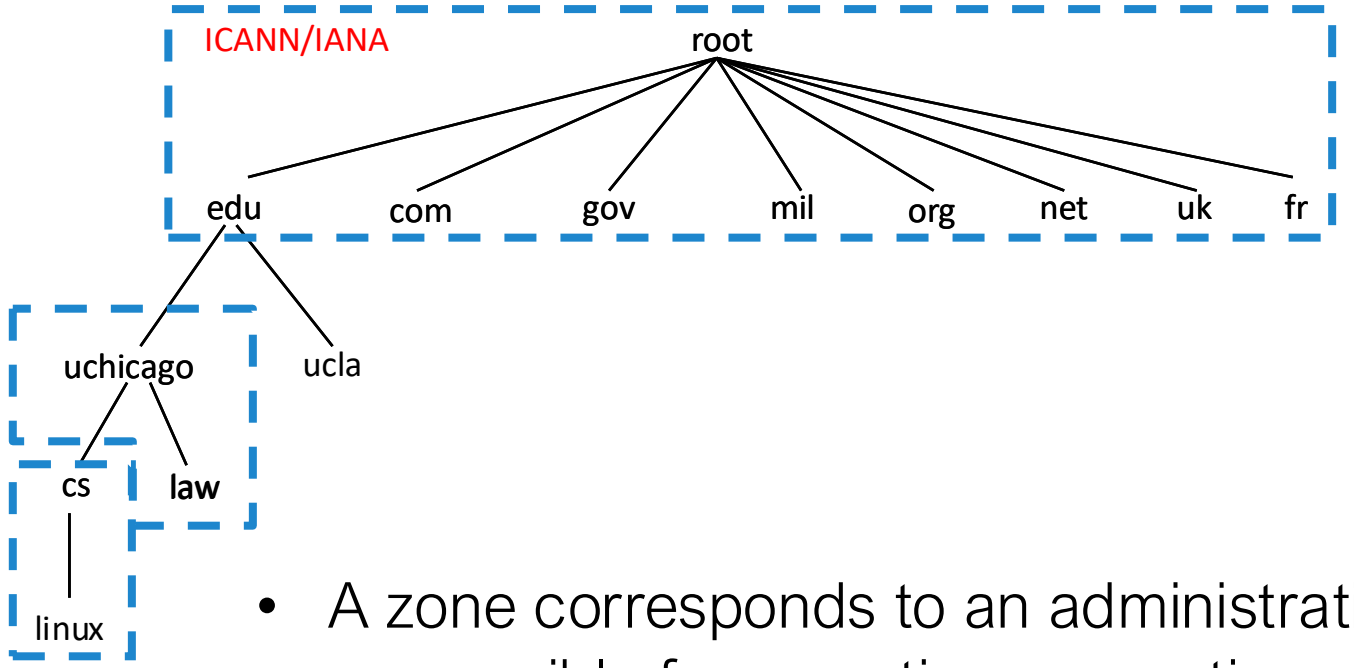
- Host (machine) IP addresses: e.g., *128.135.11.239*
  - A number used by protocols (e.g., BGP)
- Host names: e.g., *super.cs.uchicago.edu*
  - Usable by humans
- Domain Name System (DNS): how we map from hostnames to IP addresses

Hierarchical Namespace



- "Top Level Domains" are at the top (below root)
- Domains are subtrees
  - E.g: .edu, uchicago.edu, cs.uchicago.edu
- Name is leaf-to-root path
  - linux.cs.uchicago.edu
- Name collisions trivially avoided!
  - each domain's responsibility

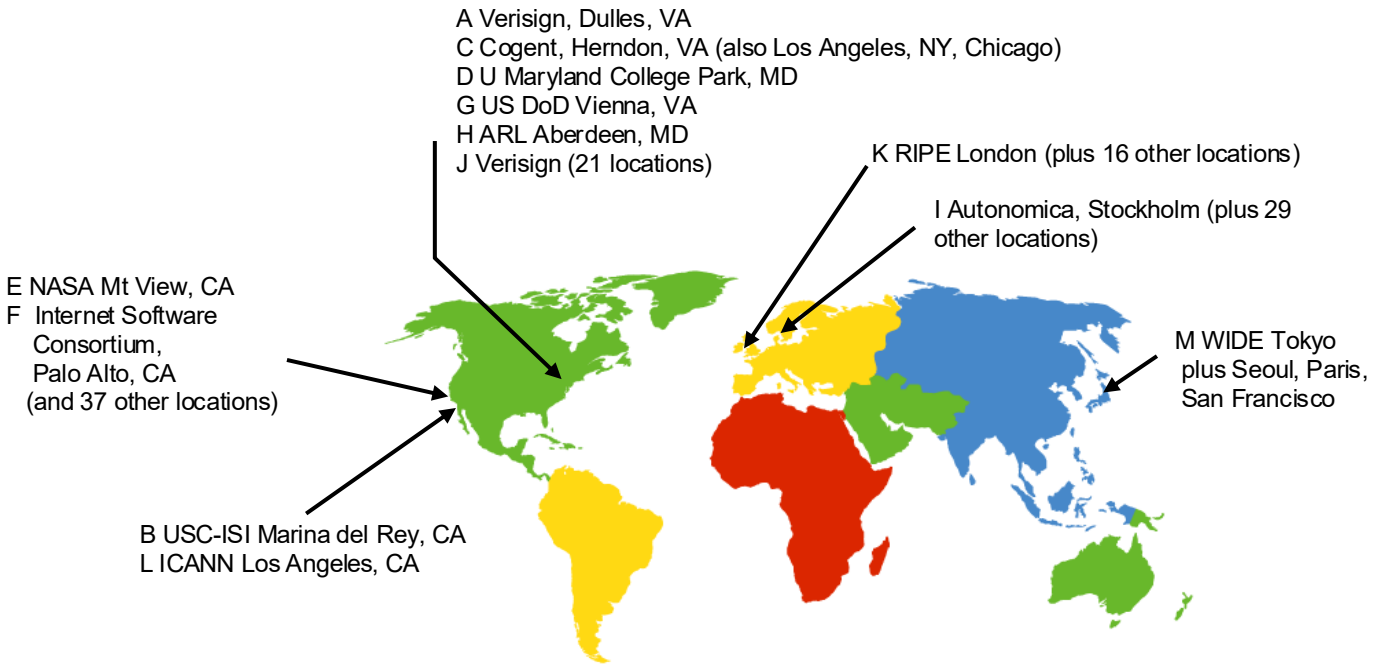
Hierarchical Administration



- A zone corresponds to an administrative authority responsible for a contiguous portion of hierarchy
- UChicago controls law.uchicago.edu and *.cs.uchicago.edu
  while CS controls *.cs.uchicago.edu

DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers.org/)
- All replicated via anycast



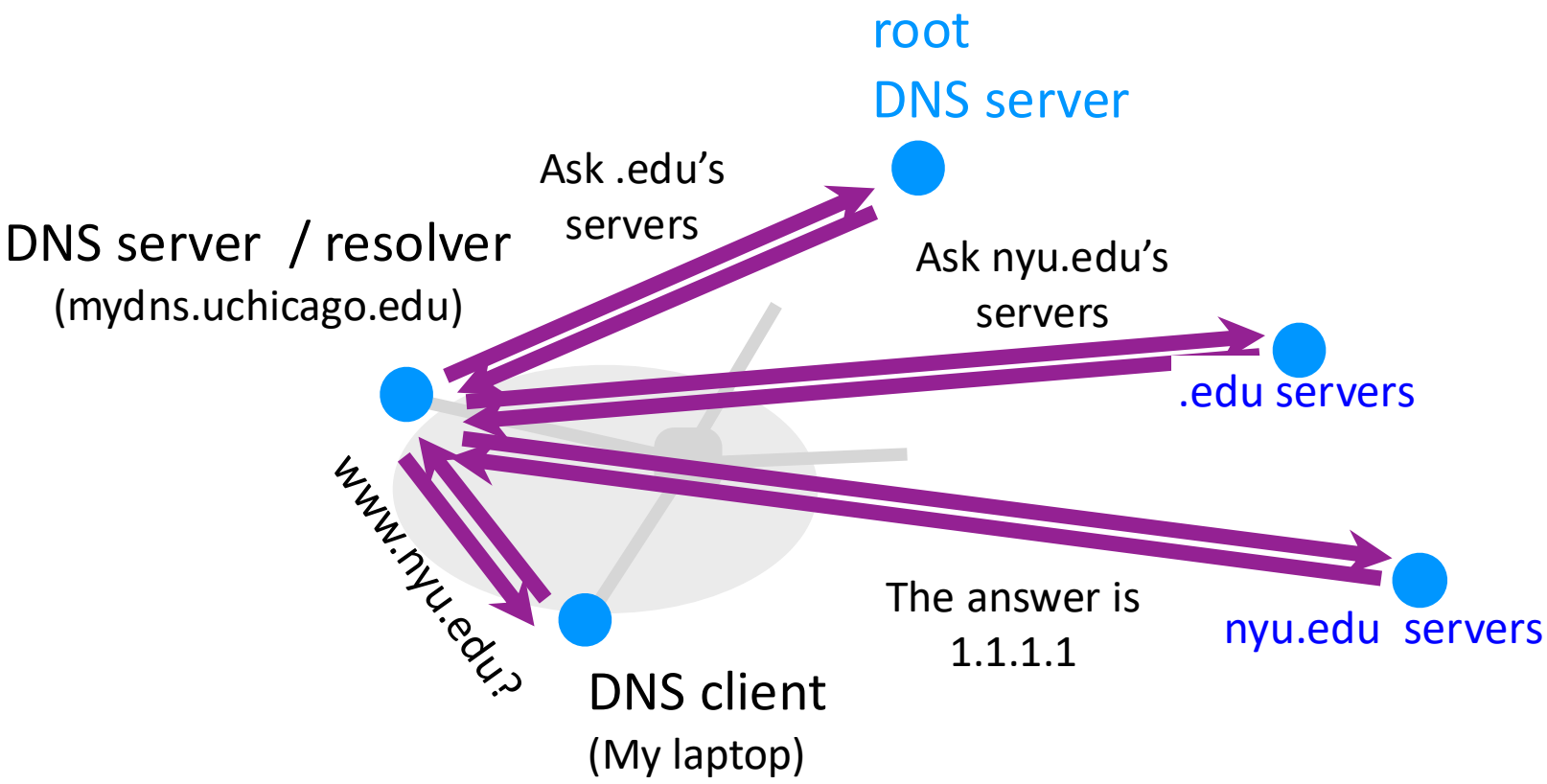
DNS Records

- DNS servers store Resource Records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: *(→ Address)*
  - name = hostname
  - value = IP address
- Type = NS: *(→ Name Server)*
  - name = domain
  - value = name of DNS server for domain
- Type = MX: *(→ Mail eXchanger)*
  - name = domain in email address
  - value = name(s) of mail server(s)

Registering a Domain

- Example: you want “blaseur.com”
- Register blaseur.com at registrar (e.g., Dreamhost)
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts your name server’s info into the .com TLD server
- You store resource records in your domain’s DNS (name)server
  - e.g., type A record for www.blaseur.com
  - e.g., type MX record for blaseur.com

DNS Query (Lookup)



DNS Packet Format: UDP Header

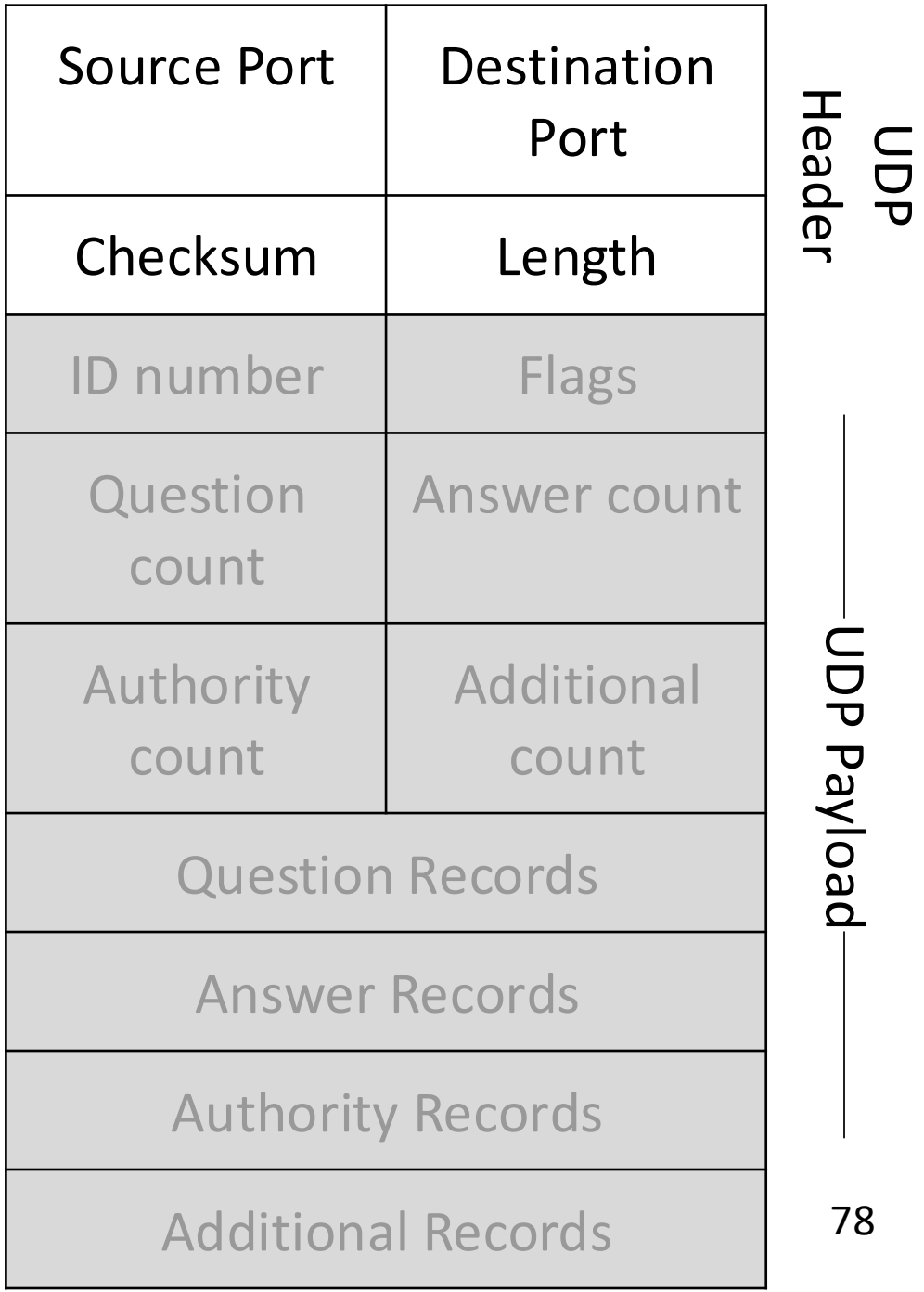
DNS is designed to be lightweight and fast: uses UDP for transport protocol

Source port (16 bits): Chosen by the client

- Can be randomized for security, as we'll see later

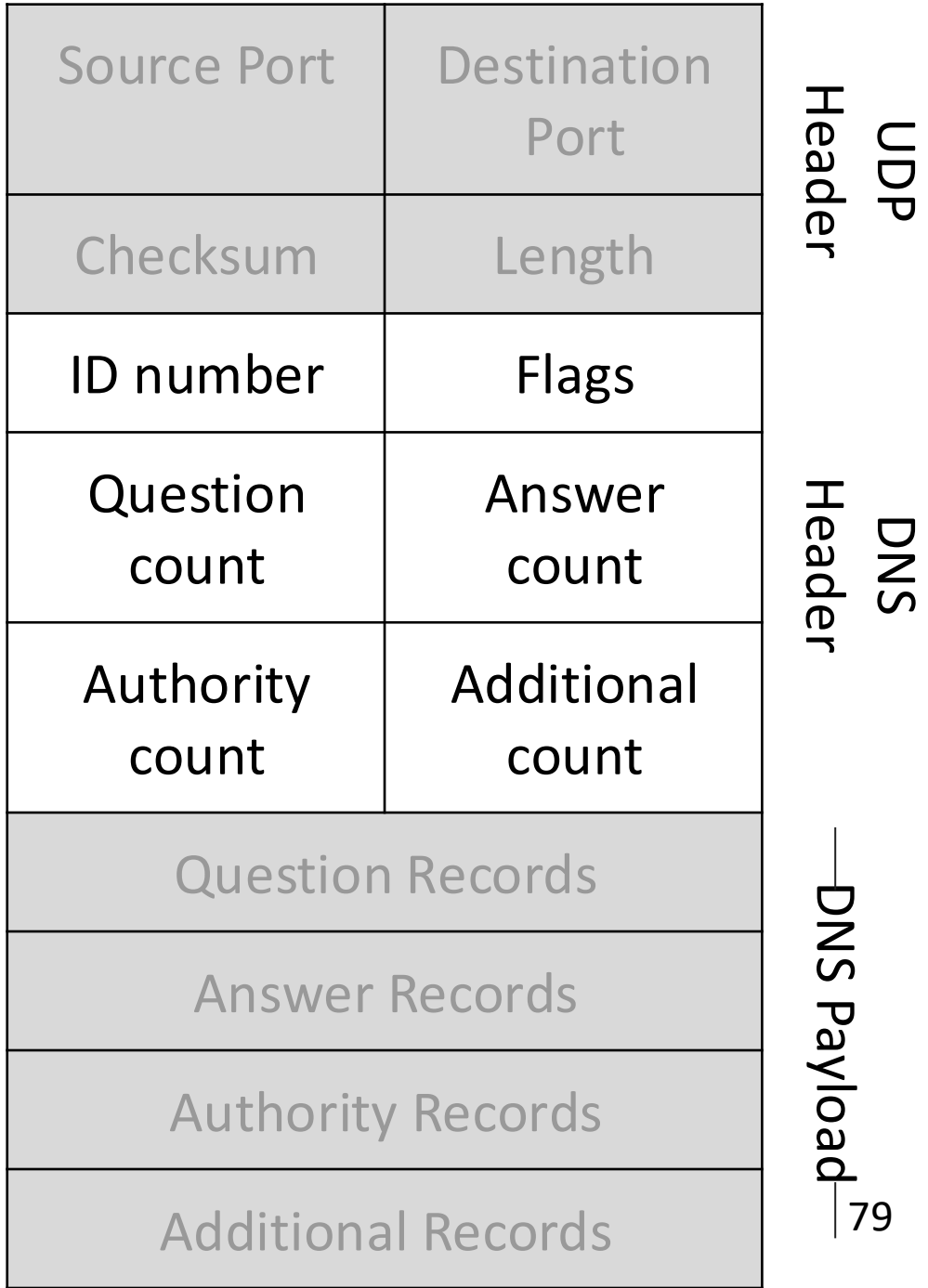
Destination port (16 bits): Usually 53

- DNS name servers answer requests on port 53



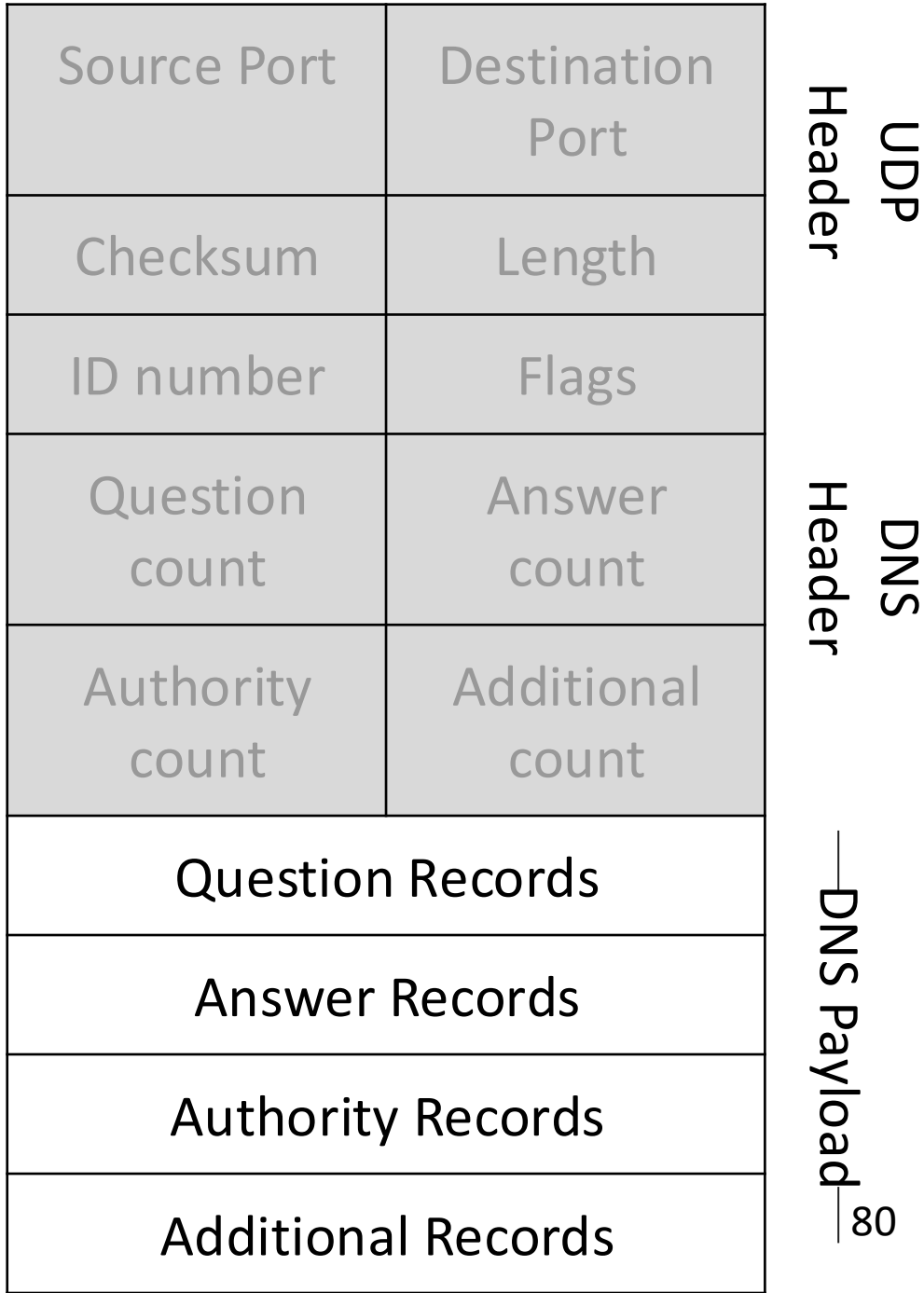
DNS Packet Format: DNS Header

- **ID number** (16 bits): Used to associate queries with responses
  - Client picks an ID number in the query
  - Name server uses the same ID number in the response
  - Should be random for security, as we'll see later



DNS Packet Format: DNS Payload

- The DNS payload contains a variable number of **resource records** (**RRs**)
- RRs are sorted into four sections
  - Question section
    (What is the query?)
  - Answer section
    (What is the final answer?)
  - Authority section
    (Which name servers should I ask for more info?
  - Additional section
    (What are the IP addresses of name servers I should ask?)



Example: DNS Request Packet

Lookup for "www.unixwiz.net"

Request to the ".net" TLD nameserver

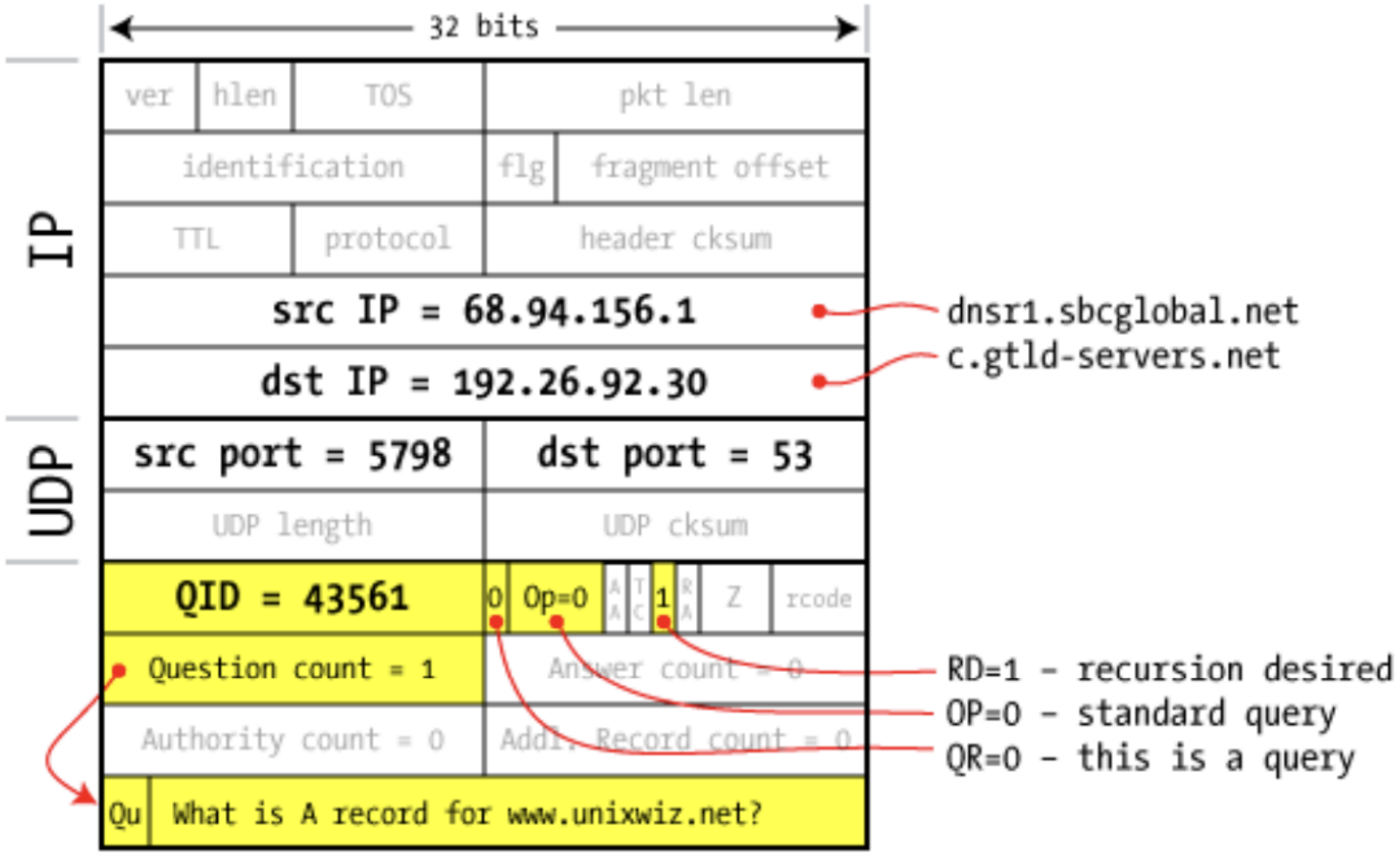


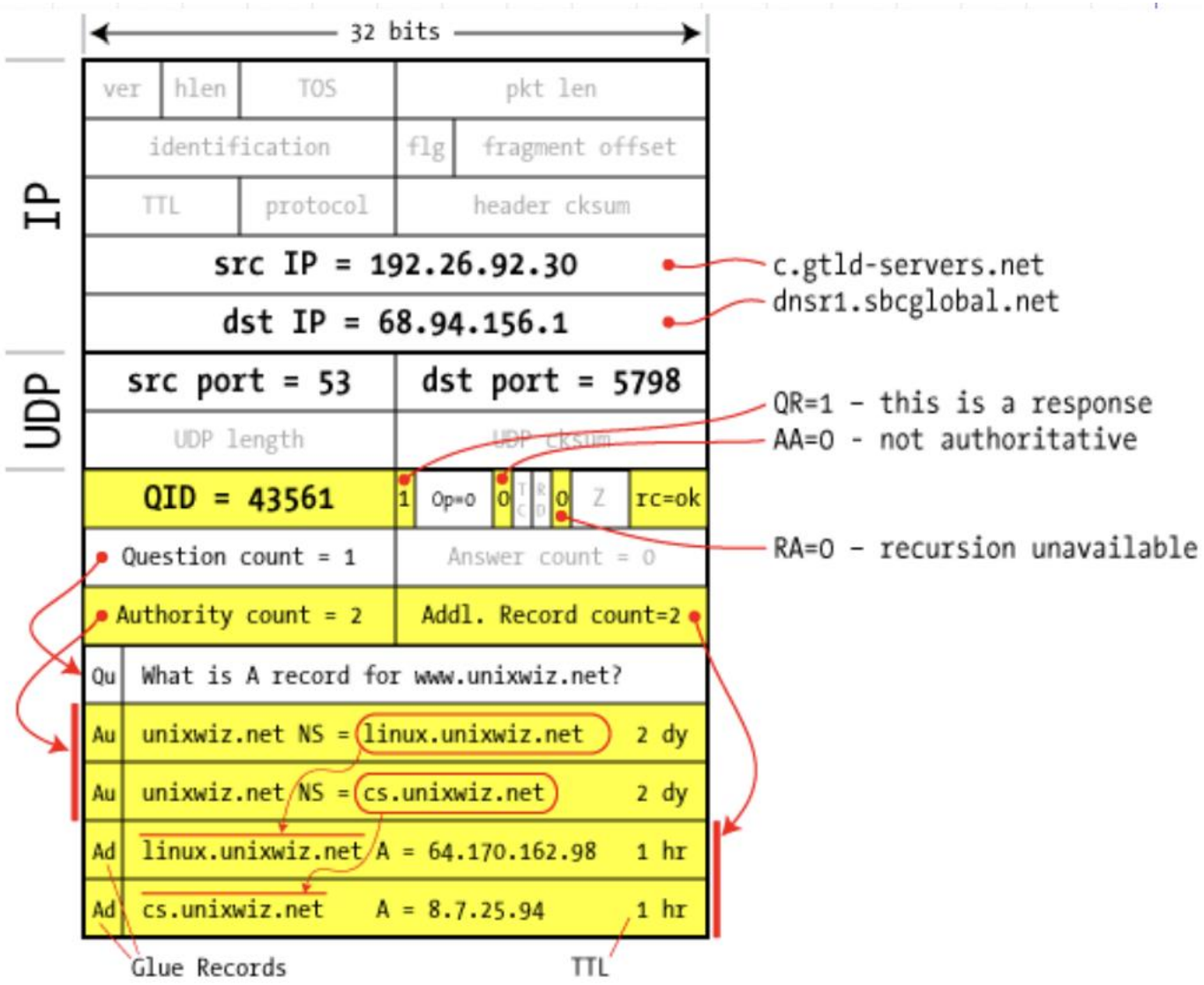
Image from http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Example: Response Packet

Response by the ".net" TLD nameserver to our local DNS resolver

Authority Section: Who are the name servers you should talk to next?

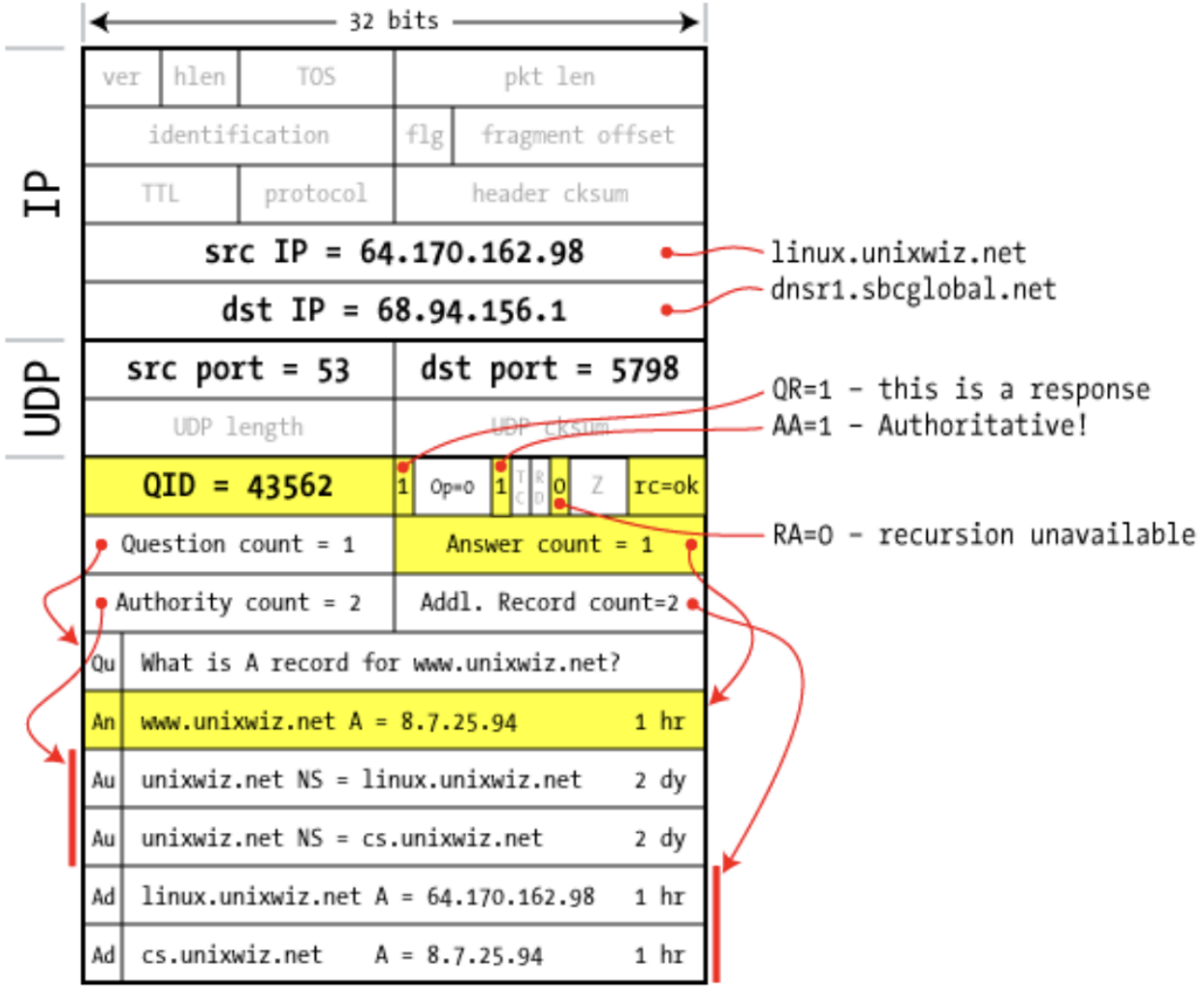
Additional Section [Glue Records]: What are their IP Addresses so you can go ask them?



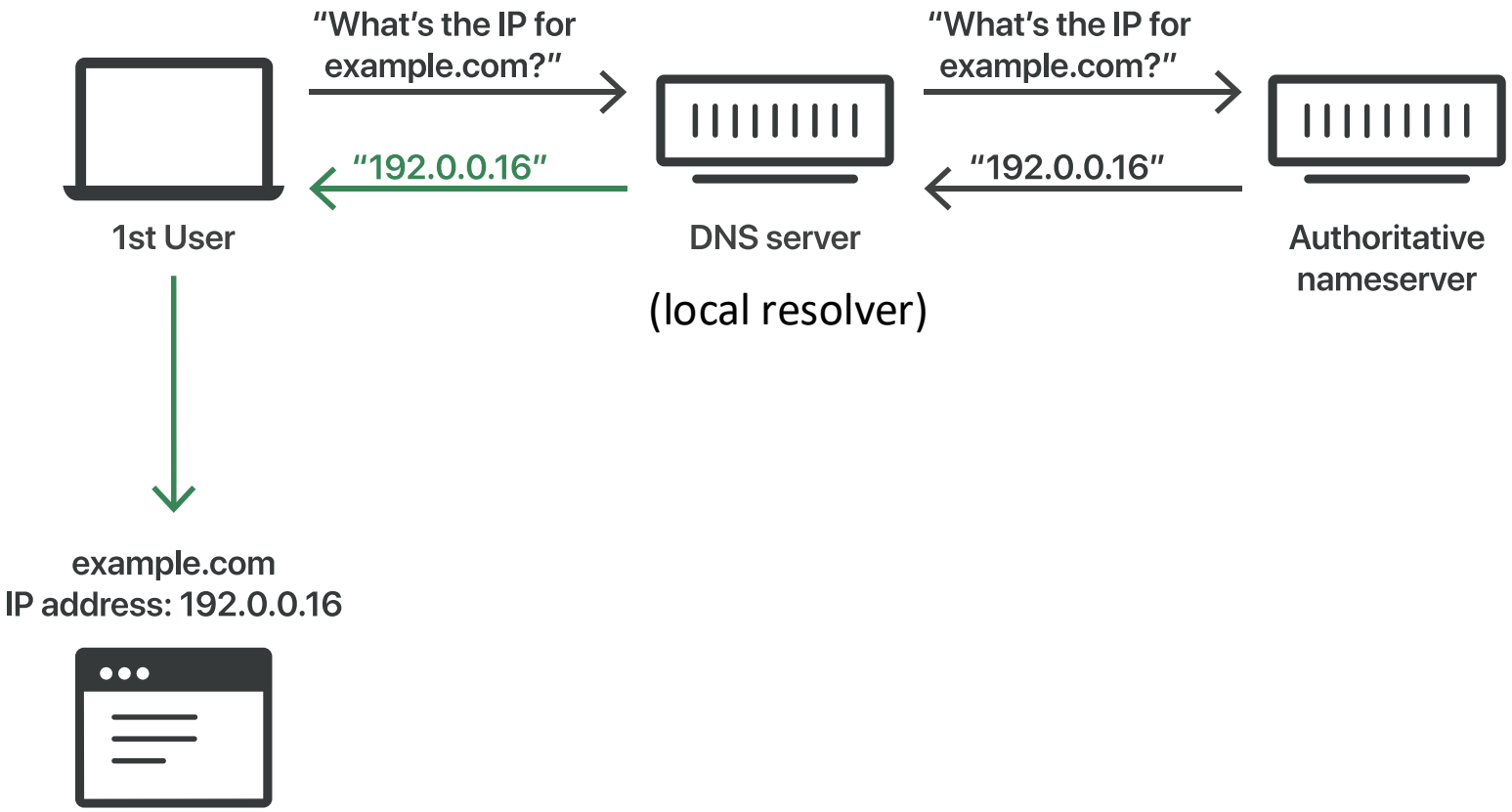
Example: Authoritative Response

Response by the "unixwiz.net" nameservers to our local DNS resolver

Answer Section: What is the IP address for the queried domain: "www.unixwiz.net"?

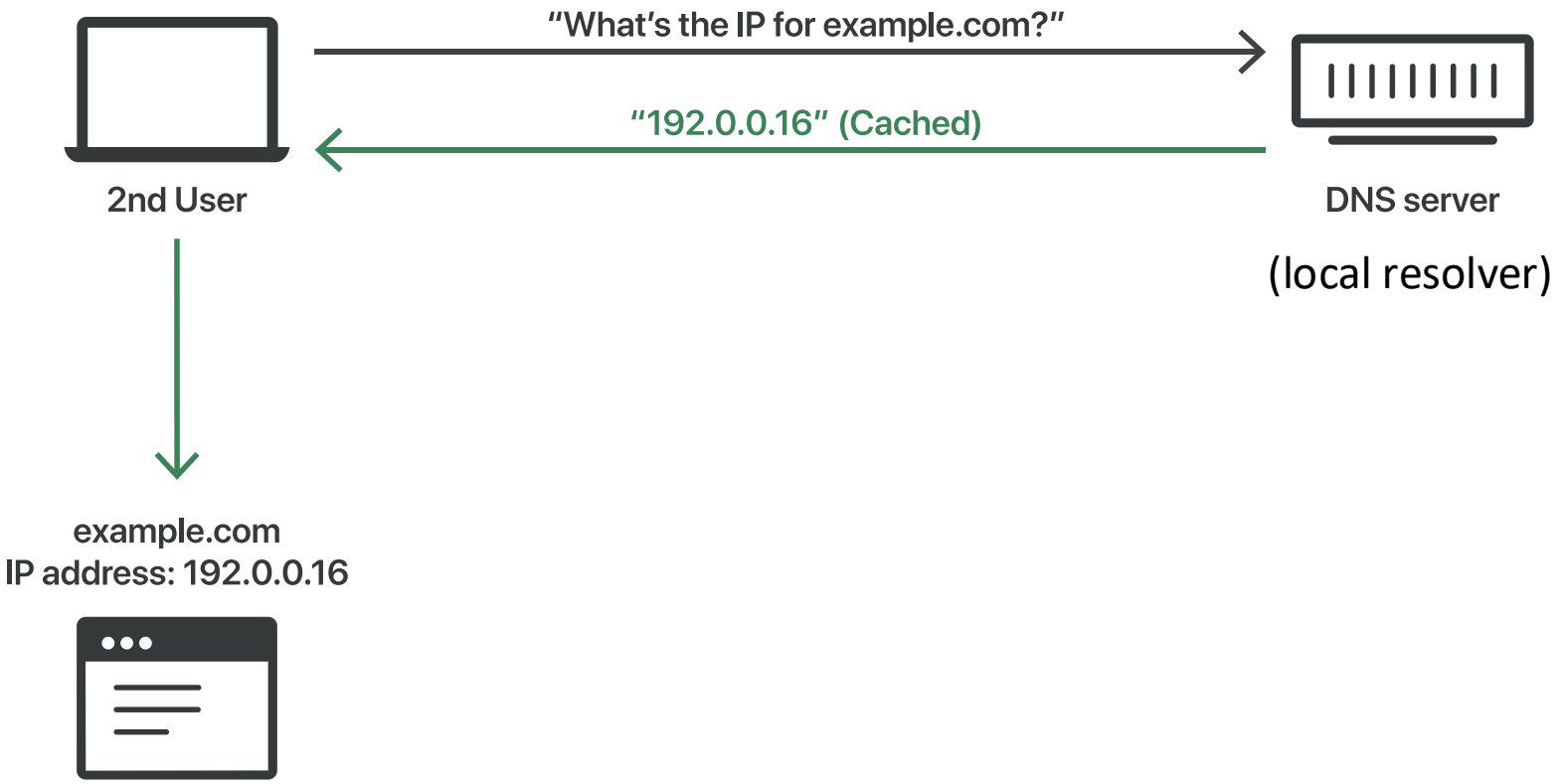


DNS (Uncached)



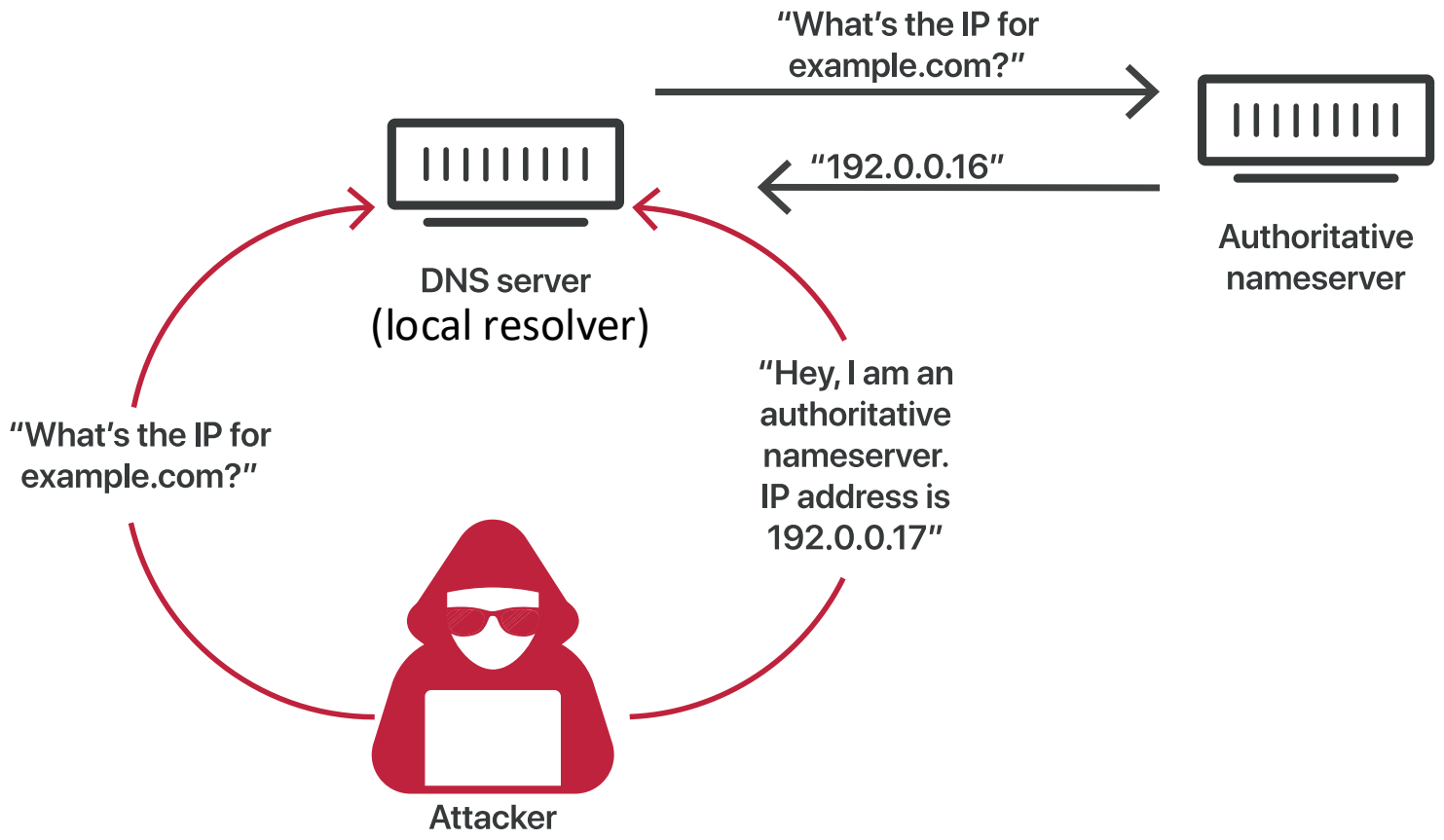
Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS (Cached, Benign)



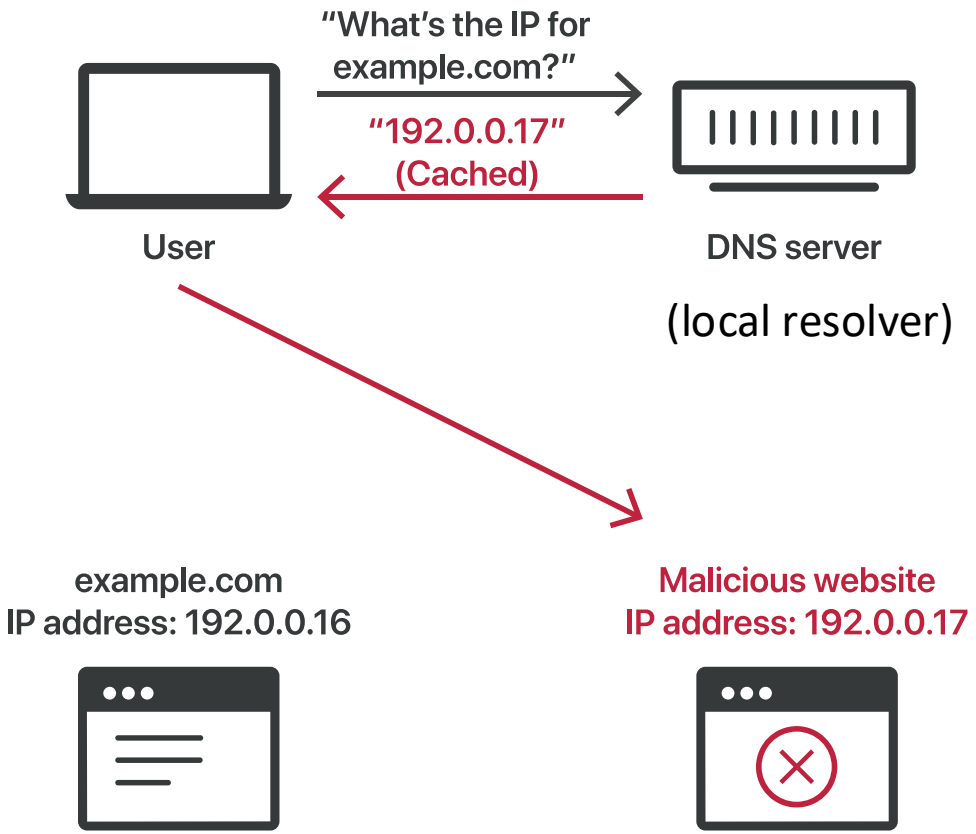
Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning Attack



Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning Result



Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/