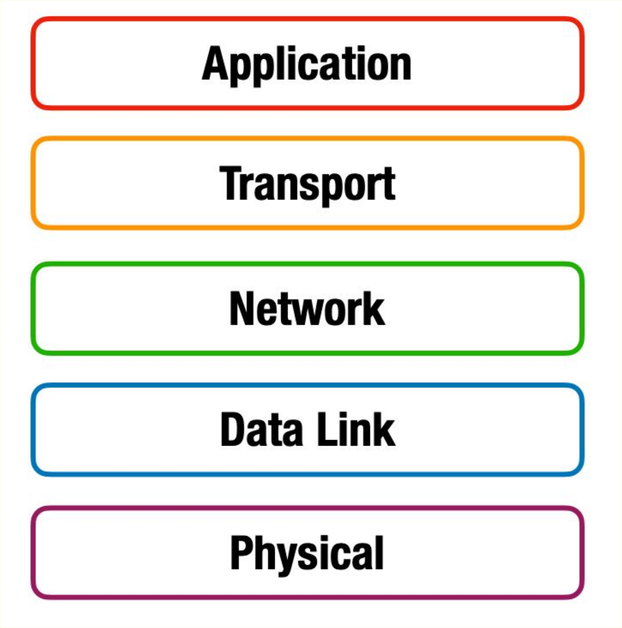
Internet Protocols and Network Security

Review: Internet Protocols

The layers of the internet



← **HTTP, HTTPS, DNS, SSH** (how applications communicate)

← **TCP, UDP** (reliable communication - connections)

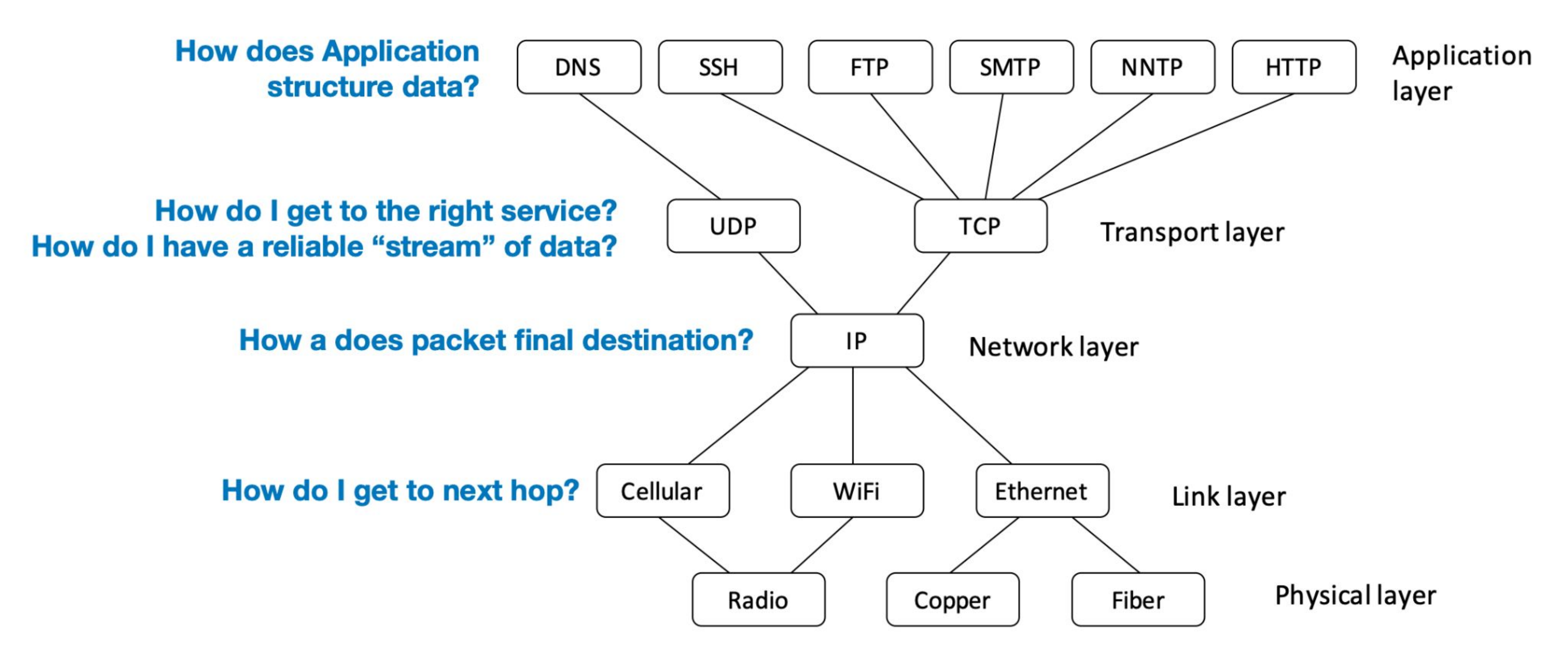
← **IP layer** (packet forwarding, getting from src to dst)

← **Ethernet layer, ARP** (next hop transmission, ethernet)

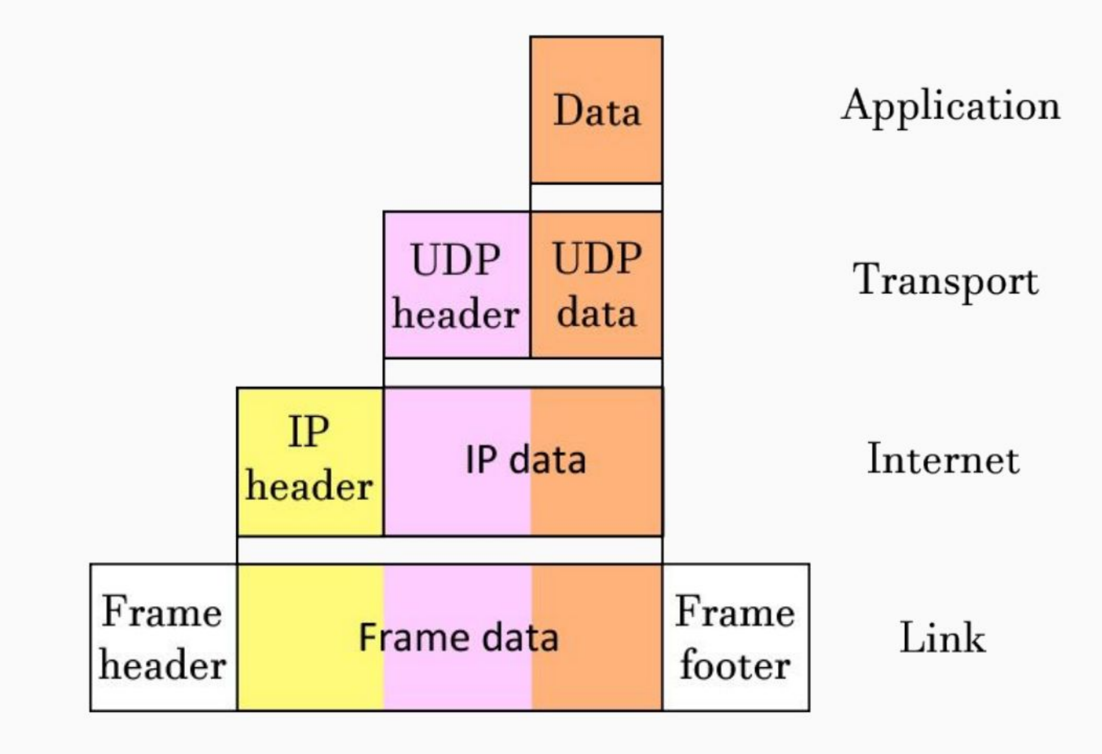
← **Bits on a wire** (electrical signals, often on an actual wire)

Packets on the internet are sent with one header for each layer to unwrap.

A map of internet protocols



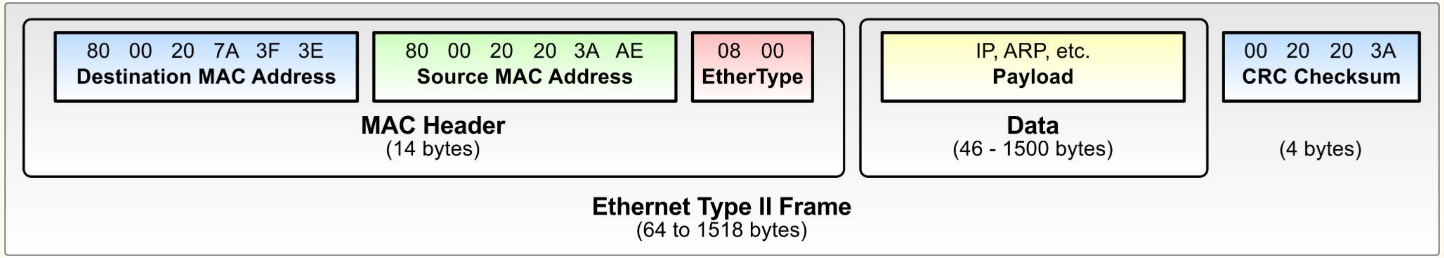
Packet encapsulation



Credit: Alex Stamos, INTLPOL 268 *Hack Lab*, Fall 2022

Ethernet

- **Link layer** protocol for sending data on a *Local Area Network* (LAN)



- **Host abstraction**: MAC (hardware) address (e.g. `60:3e:5f:37:d1:71`)
  - To send a message to a host, you must be directly connected to it

Internet-layer protocols

Internet Protocol (IP)

- **Motivation**: Ethernet only allows sending packets within a local network
    - But we want to send packets that traverse the internet
- **Host abstraction**: IP address (independent of the local network)

Internet Protocol (IP)

- **Motivation**: Ethernet only allows sending packets within a local network
  - But we want to send packets that traverse the internet

- **Host abstraction**: IP address (independent of the local network)

IPv4 address

- 32-bit address; written as 8 octets
  - e.g. 171.64.64.64 (Stanford CS)
- Subnet notation: prefix and # *defined bits*
  - e.g. 171.64.0.0/14 (Stanford)
- Private IPv4 subnets:
  - 192.168.0.0/16
  - 172.16.0.0/12
  - 10.0.0.0/8

IPv6 address

- 128-bit address, written as 8 groups of 4 hexadecimal digits (blanks denote 0)
  - e.g. 2606:4700:3036::ac43:9b70
- Introduced in 1995, but still not well-supported

IP spoofing

- **Security issue with IP**: any host can pretend to be any other host!
  - A malicious host can send packets with a different source IP address

- **Defense**: perform ingress filtering to block packets coming from outside the network that have a private IP address

- **Defense in depth**: don't give special privileges to a machine just because it's inside the network
  - Add authentication and authorization for all sensitive actions (*Zero Trust*)

Address Resolution Protocol (ARP)

- **Motivation**: On a LAN, given a host where you know its IP address, how to get its MAC address?

- **ARP request**: a *broadcast* question "Who has IP address 192.168.170.100? Tell 192.168.170.102"
- **ARP response**: a *unicast* message "00:0c:29:74:4c:4d has IP address 192.168.170.100"

- Machines can also send a *broadcast* **ARP announcement** with the same body as an ARP response

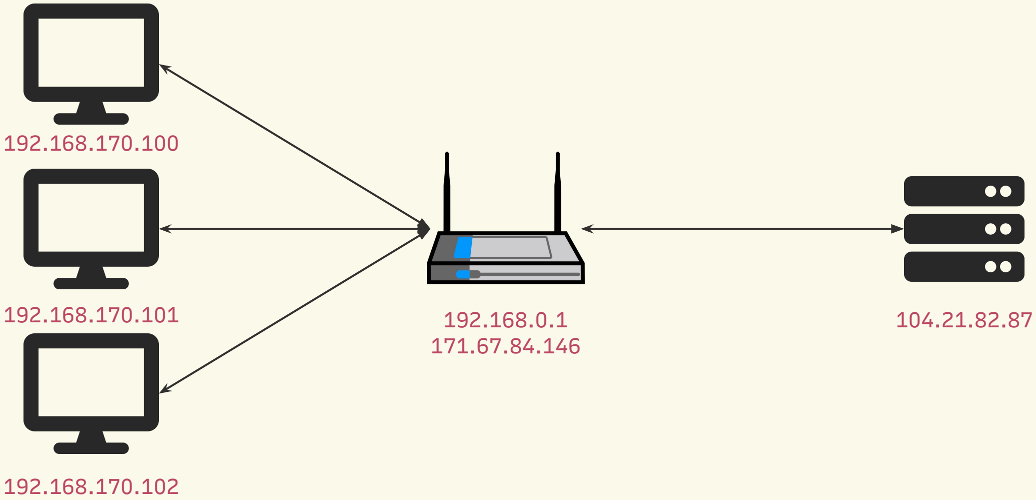
ARP spoofing

- **Security issue with ARP**: any host can pretend to be any other host!
  - A malicious host can *broadcast* that it has a given IP address, even if a different host on the network has that IP address

- **Defense**: ignore unsolicited ARP announcements, or detection software on the upstream switch

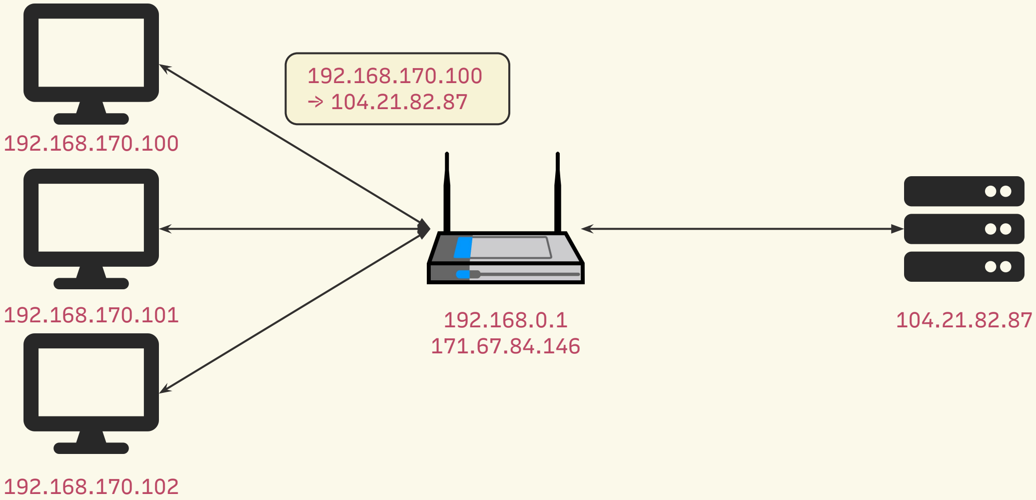
Network Address Translation (NAT)

- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
    - But there are far more than 4 billion internet-connected devices!
- NAT lets multiple hosts share a single public IP address while having different private addresses



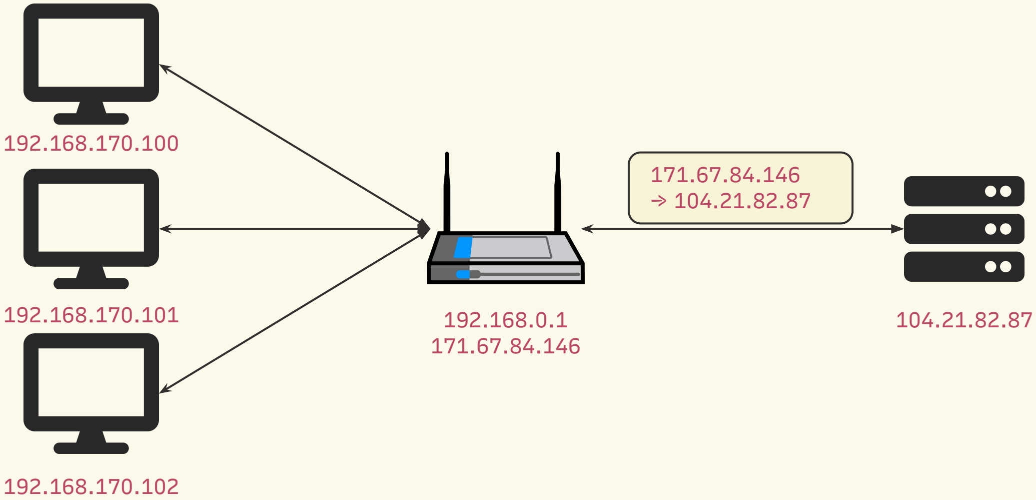
Network Address Translation (NAT)

- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
  - But there are far more than 4 billion internet-connected devices!
- NAT lets multiple hosts share a single public IP address while having different private addresses



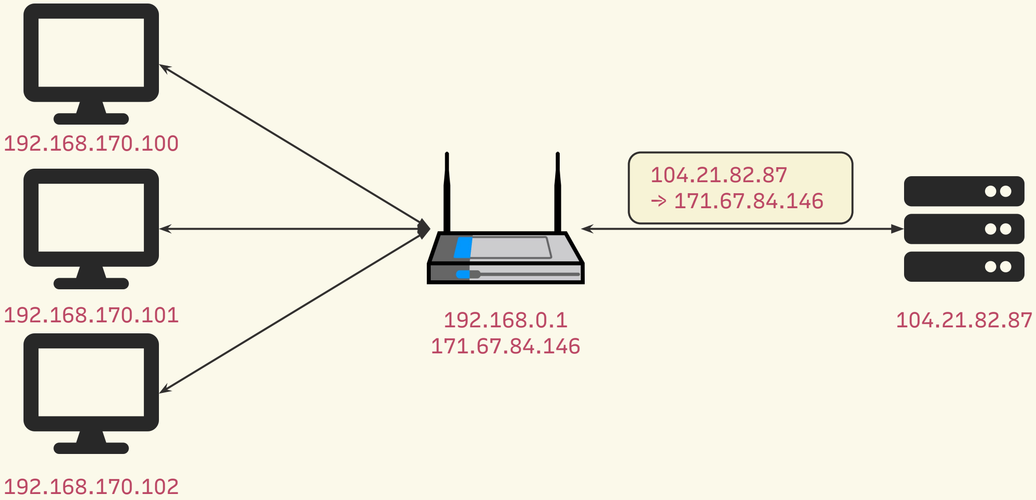
Network Address Translation (NAT)

- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
  - But there are far more than 4 billion internet-connected devices!
- NAT lets multiple hosts share a single public IP address while having different private addresses



Network Address Translation (NAT)

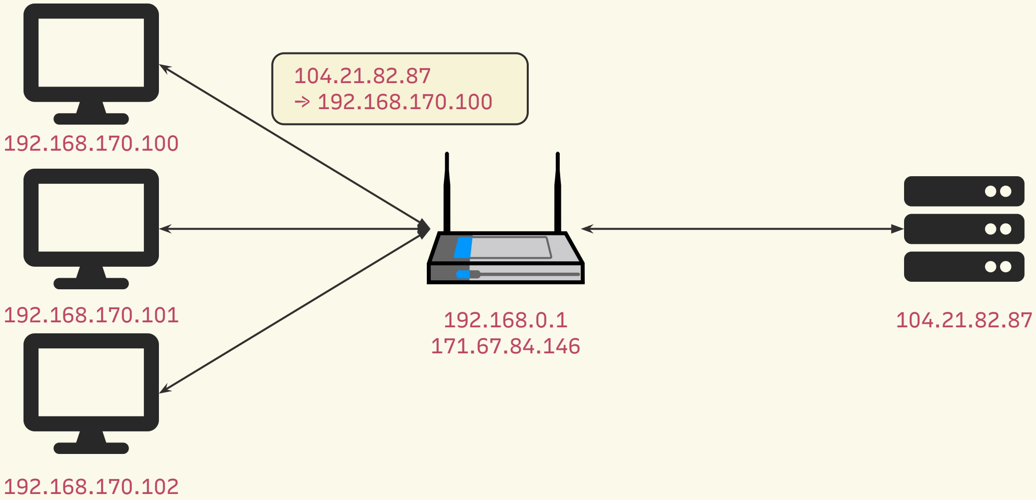
- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
  - But there are far more than 4 billion internet-connected devices!
- NAT lets multiple hosts share a single public IP address while having different private addresses



Network Address Translation (NAT)

- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
  - But there are far more than 4 billion internet-connected devices!

- NAT lets multiple hosts share a single public IP address while having different private addresses



Network Address Translation (NAT)

- **Motivation**: 32-bit addresses mean there are only 2^{32} ~= 4 billion IPv4 addresses available
  - But there are far more than 4 billion internet-connected devices!
- NAT lets multiple hosts share a single public IP address while having different private addresses
- **Security benefit**: if a host needs to receive inbound traffic behind a NAT, the NAT must explicitly be configured to allow this
  - Inbound traffic is opt-in: the NAT is effectively an implicit firewall

Routing

- **Motivation**: With IP addresses, how to make a *best effort* to get packets to their eventual destination?

- *Routers* store *routing tables* that inform how to forward packets towards their destination, often stored as tuples of (**CIDR prefix**, **next hop**)

```
vyos@team-router188:~$ ip route
default nhid 14 via 192.168.0.1 dev eth0 proto static metric 20
10.10.188.0/24 dev eth1 proto kernel scope link src 10.10.188.1
192.168.0.0/16 dev eth0 proto kernel scope link src 192.168.188.1
vyos@team-router188:~$
```

Routing

- **Motivation**: With IP addresses, how to make a *best effort* to get packets to their eventual destination?

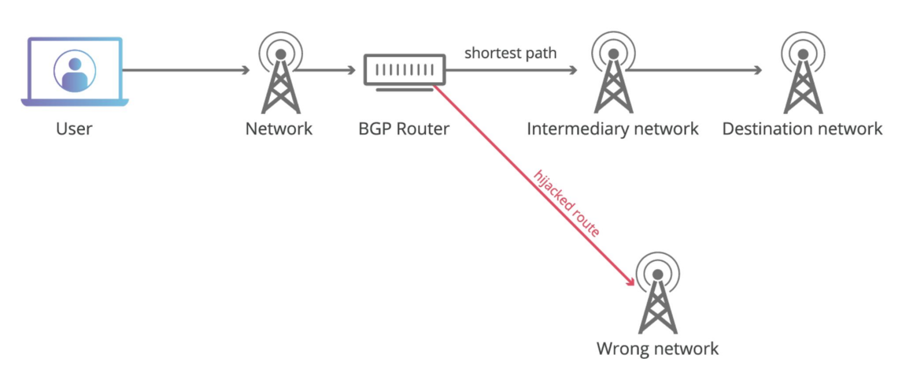
- *Routers* store *routing tables* that inform how to forward packets towards their destination, often stored as tuples of (**CIDR prefix**, **next hop**)

```
vyos@team-router188:~$ ip route
default nhid 14 via 192.168.0.1 dev eth0 proto static metric 20
10.10.188.0/24 dev eth1 proto kernel scope link src 10.10.188.1
192.168.0.0/16 dev eth0 proto kernel scope link src 192.168.188.1
vyos@team-router188:~$
```

- *Border Gateway Protocol* (BGP): a protocol for routers to exchange (*announce*) routing table information with each other
  - *Autonomous Systems* (AS): Groups of networks managed by a single entity that announce routes to neighboring routers via BGP

BGP hijacking

- **Security issue with BGP**: any AS can announce routes for IP subnets that belong to other ASes!
  - e.g. in 2018 a Russian ISP advertised a route for **myetherwallet.com**'s IP address that directed to malicious servers, allowing attackers to steal >$150k of cryptocurrency



- **Defense**: *Resource Public Key Infrastructure* (RPKI), which provides a cryptographically auditable chain of trust for BGP route announcements

Transport protocols

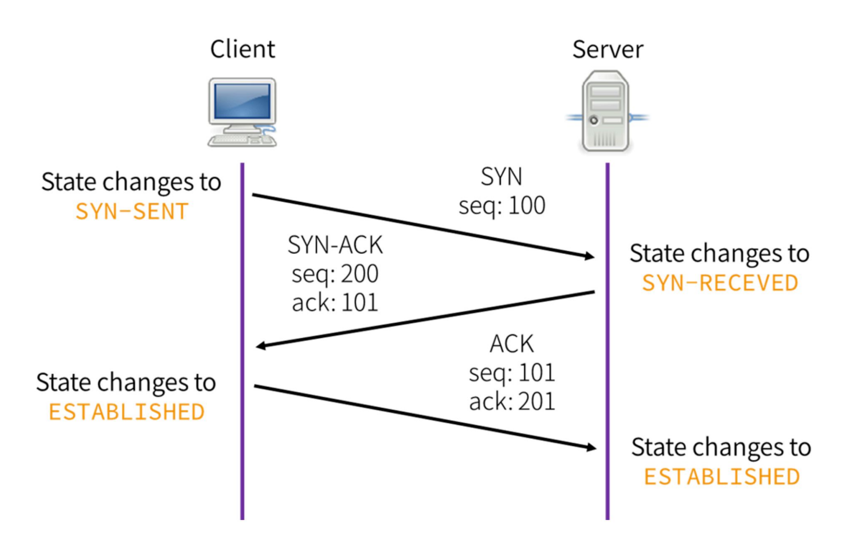
User Datagram Protocol (UDP)

- **Motivation**: *Multiplex* traffic on the same host (single IP address) across different *ports*
  - This allows multiple applications to send and receive traffic from the same host

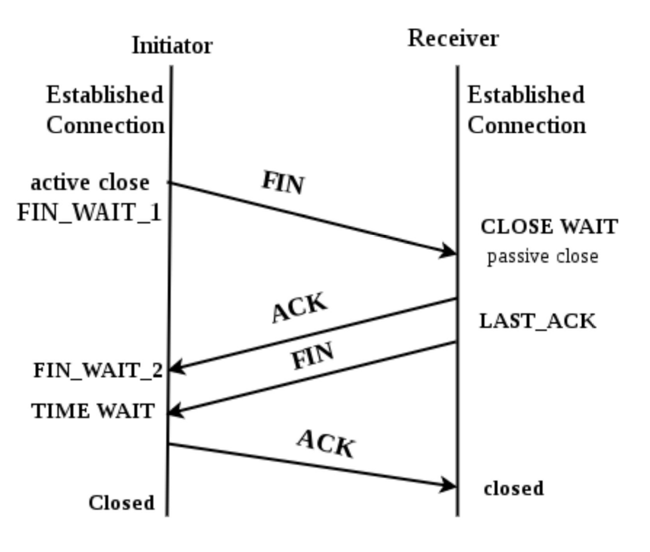
Transmission Control Protocol (TCP)

- **Motivation**: Guarantee *acknowledged (reliable) delivery* and *data stream ordering* for internet packets
  - Implemented directly on top of IP, but can be conceptualized as extending UDP

Establishing a TCP connection



Ending a TCP connection

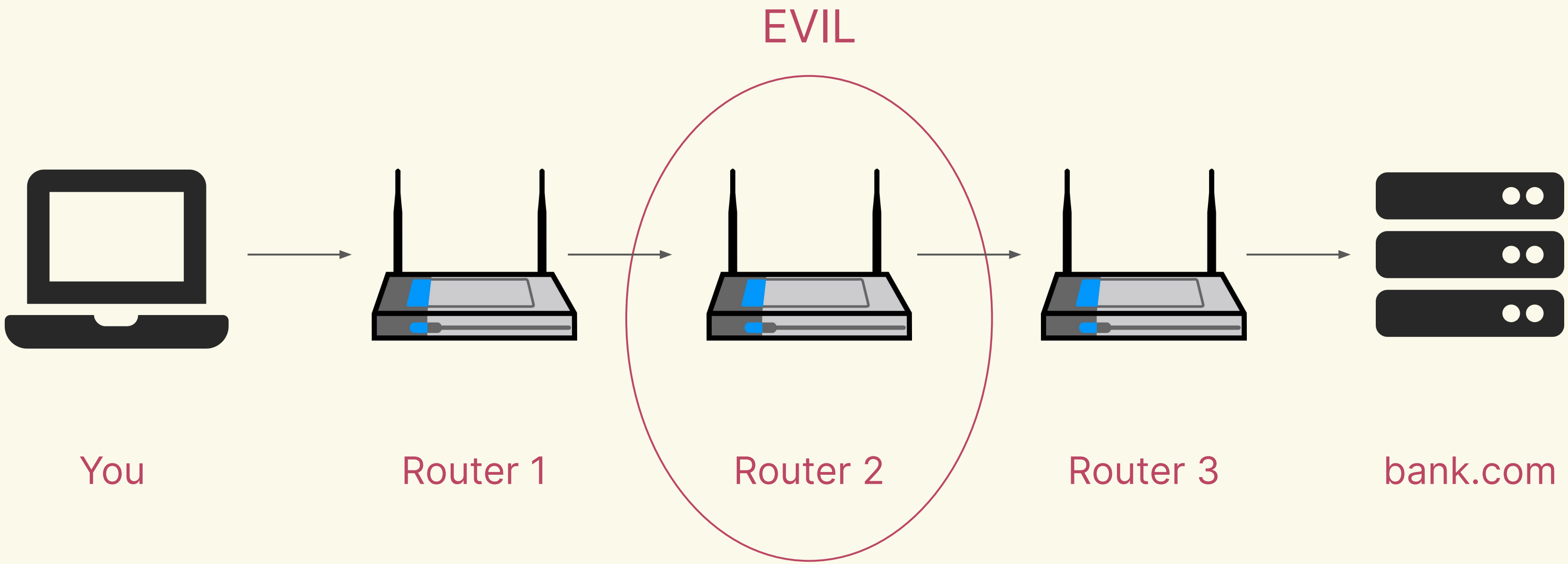


Attacks on TCP

- **TCP connection spoofing:** Pretend to be the host (send back a **`SYN`**-**`ACK`** with guessed sequence number)
- **TCP reset attack**: Send **`RST`** to correct port
- **TCP SYN flooding attack:** Send **`SYN`** packets to all the ports of a server → denial of service

Transport Layer Security (TLS)

- **Motivation**: Encrypt data in transit, such that any intermediary that routes your connection cannot see the underlying data



Asymmetric cryptography for TLS

- Server sends you a *public key* that can **only be used to encrypt data**, such that only the corresponding *private key* can decrypt the data
  - Because the public key can't decrypt the data, an attacker has no way of viewing the plaintext data

- This helps create a **secure channel** that can be used to communicate
  - Server public and private key used to **agree on a shared key** used to encrypt all traffic

Asymmetric cryptography for TLS

- Server sends you a *public key* that can **only be used to encrypt data**, such that only the corresponding *private key* can decrypt the data
  - Because the public key can't decrypt the data, an attacker has no way of viewing the plaintext data

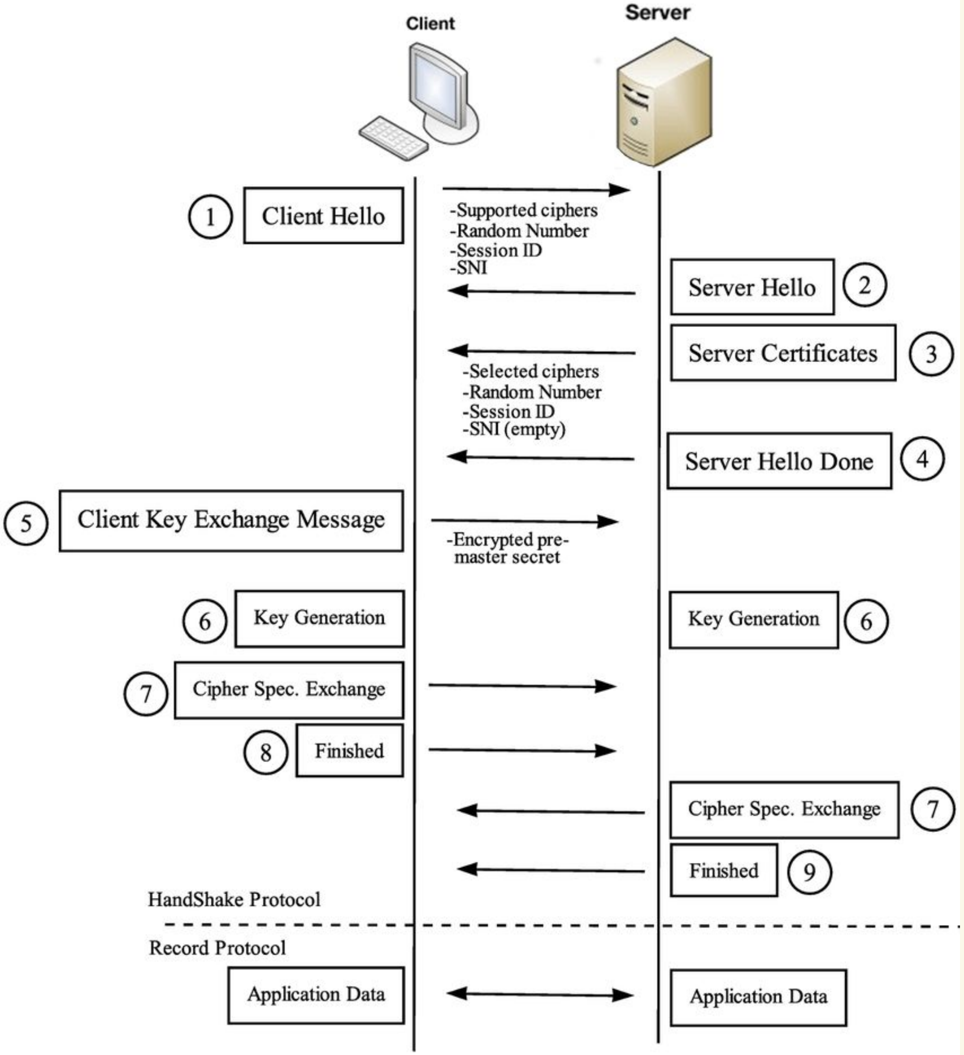
- This helps create a **secure channel** that can be used to communicate
  - Server public and private key used to **agree on a shared key** used to encrypt all traffic

- **Issue**: How do you verify the public key is **legitimately owned** by the website you are trying to connect to?
  - **Solution**: have some way of **choosing which keys to trust**
    - Designate a number of **trusted providers** (*certificate authorities*) which can verify other keys
    - Only accept a key if someone you already trust has verified it's legitimate

Public Key Infrastructure (PKI) for TLS

- PKI refers to a global hierarchical tree structure of trusted certificates
  - Roots of the tree are distributed with OSes and some applications (e.g. Python, Chrome)
  - Any certificate with a chain of trust ending at one of these certificates is trusted
  - Certificates are given out by *certificate authorities* (e.g. Let's Encrypt)

- TLS certificates use the X.509 format, and contain:
  - Public key
  - *Signature* (to be widely trusted, must come from a trusted CA and not self-signed)
  - Domain name(s) the certificate is valid for
  - Expiry



Establishing a TLS session

1) **Client**: Say hi and propose ciphers
2) **Server**: Pick a cipher
3) **Server**: Send certs
4) **Client**: Verify certs
5) **Both**: Diffie-Hellman Key Exchange
6) **Both**: Send packets

TLS attacks and security issues

- **Forged certificates**: e.g. self-signed or expired certificate
  - Defense: check the certificate is still valid

- **SSL strip attack**: prevent upgrade to HTTPS (man-in-the-middle)
  - Defense: *HTTP Strict Transport Security* (HSTS) – always connect over HTTPS
  - Either in the Content Security Policy for a site, or in the *HSTS Preload List*

- **Fraudulently or mistakenly issued certificates**: by rogue or poorly secured and hacked CAs
  - Defense: *Certificate Transparency* (all CAs must declare all certs issued)

Common application protocols

Domain Name System (DNS)

- **Motivation**: Host addressing and routing works using IP addresses, but humans don’t want to type in IP addresses to load web sites

- **Solution**: DNS provides a way to *resolve* human readable names (e.g., `cs155.stanford.edu`) to IP addresses

- Runs over *both* UDP and TCP, conventionally port **53**

DNS architecture

- DNS is a hierarchical, delegatable namespace
  - *Root DNS servers* are maintained by ICANN

- Home routers (usually) host a *local DNS server*, which provides DNS for the internal network
  - Individual machines can host their own internal DNS records: `/etc/hosts` on Unix, `C:\Windows\System32\drivers\etc\hosts` on Windows

- Router specifies an *authoritative DNS server* for machines outside of the domain
  - Performs a *reverse DNS lookup* for all queries it cannot resolve internally

Well-known DNS servers

- Cloudflare: **1.1.1.1**
- Google: **8.8.8.8**, **8.8.4.4**
- Others: **9.9.9.9** (Quad9), various ISP DNS servers
  - e.g. Stanford: **171.67.64.53**, **171.64.69.53**

DNS record types

- `A` (alias): simplest record type; maps domain name to IPv4 address
  - e.g. `applied-cyber.stanford.edu` → `171.67.84.46`
  - `AAAA`: Like `A`, but returns IPv6 addresses

DNS record types

- **A** (alias): simplest record type; maps domain name to IPv4 address
  - e.g. **applied-cyber.stanford.edu** → **171.67.84.46**
  - **AAAA**: Like **A**, but returns IPv6 addresses

- **CNAME** (canonical name): used to create aliases, mapping domain names to domain names
  - e.g. **cs155.stanford.edu** → **securitylab.github.io**

DNS record types

- `A` (alias): simplest record type; maps domain name to IPv4 address
  - e.g. `applied-cyber.stanford.edu` → `171.67.84.46`
  - `AAAA`: Like `A`, but returns IPv6 addresses
- `CNAME` (canonical name): used to create aliases, mapping domain names to domain names
  - e.g. `cs155.stanford.edu` → `securitylab.github.io`
- `NS` (nameserver): used to designate an authoritative nameserver
  - Output: another DNS server to query the domain name against
  - e.g. `sad.singles` → `dara.ns.cloudflare.com`

DNS hijacking

- **Security issue with DNS**: any man-in-the-middle attacker can return the client a different domain than the correct one
    - DNS is vulnerable because it is unauthenticated and unencrypted by default

DNS hijacking

- **Security issue with DNS**: any man-in-the-middle attacker can return the client a different domain than the correct one
    - DNS is vulnerable because it is unauthenticated and unencrypted by default

- **Defense**: *DNSSEC* authenticates DNS responses using public-key cryptography

DNS hijacking

- **Security issue with DNS**: any man-in-the-middle attacker can return the client a different domain than the correct one
  - DNS is vulnerable because it is unauthenticated and unencrypted by default
- **Defense**: *DNSSEC* authenticates DNS responses using public-key cryptography
- **Second defense**: *DNS over TLS* (RFC 7858, 2016) and *DNS over HTTPS* (RFC 8484, 2018) encrypt DNS requests and responses so that they cannot be easily tampered with
  - Also helps with privacy!
  - Enabled by default in Firefox and Chrome since 2020

A quick note on HTTP

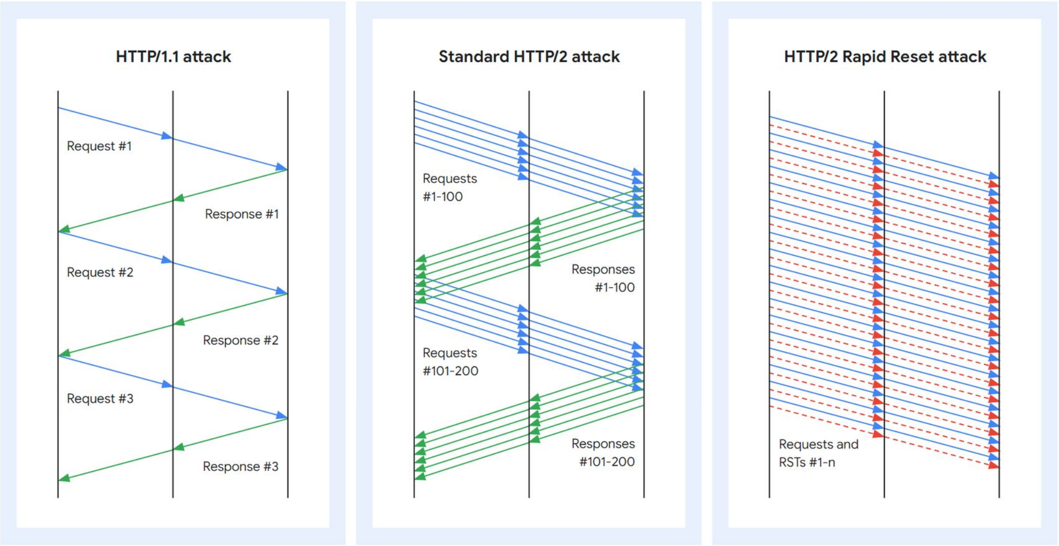
- Recall: *Hypertext Transfer Protocol* (HTTP) transports **unencrypted web content**, conventionally over TCP port **80**
  - You've already seen HTTP request and response structure in lecture

- HTTP itself can *multiplex*: a single HTTP server (one IP address) on port 80 can host web sites on different domains
  - Client needs to specify a **Host** header in a HTTP request to indicate the site it wants
  - e.g. the Cloudflare server **104.21.82.87** serves both **saligrama.io** and **infracourse.cloud**
  - This allows large hosting platforms to serve web sites from *any* of their servers

HTTP/2 Rapid Reset

- **HTTP/2** (RFC 9113, 2015) allows *multiplexing* of multiple HTTP streams over a single TCP connection

- **HTTP/2 Rapid Reset attack** (October 2023) involves creating large numbers of HTTP/2 streams to DoS a server and immediately closing them with `RST_STREAM`



HTTPS

- HTTPS is HTTP over TLS, and conventionally listens on TCP port **443**
    - All HTTP content transported via HTTPS is encrypted in transit, so no intermediate router can see plaintext content

- However: *Server Name Indication* (domain name of site being connected to – like a **Host** header) is unencrypted
    - Because a single server can multiplex sites on different domains, the client needs to provide SNI in plaintext, to tell the server which certificate to send
    - *Encrypted SNI* (2018) and *Encrypted Client Hello* (2020) alleviate this, but are not widely deployed yet – only enabled by default in Firefox and Chrome in late 2023

More Networks and Security at Stanford

- CS 144 *Introduction to Computer Networking*
- CS 244 *Advanced Topics in Networking*
- CS 249I *The Modern Internet*
- **Stanford Applied Cyber**