TCP/IP Issues

CC BY NC

A TCP/IP Tutorial

What is TCP/IP?

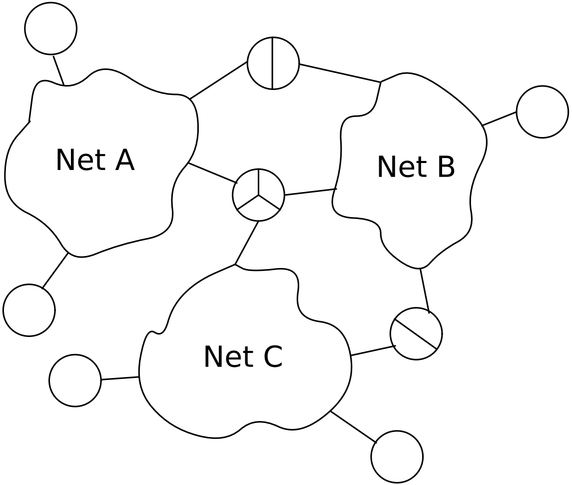
- Fundamental networking protocols of the Internet
- **TCP**: Transmission Control Protocol (RFC 793, September 1981)
- **IP**: Internet Protocol (RFC 791, September 1981)
- There have been improvements, but the fundamental architecture remains
- N.B.: HTTP/HTTPS have an outsize role; more on this later
- We'll start with a brief, simplified overview of TCP, covering only the parts necessary for this class

Design Goals

- Support many applications—most older networks had baked-in (and limited) applications
- Support many kinds of networking hardware—most older networks were tied to particular link types
- Support many different operating systems—most older networks were vendor-proprietary
- No central core network

What's a Network?

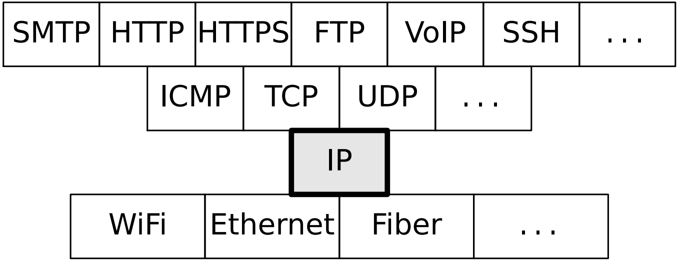
- A *network* is two or more hosts on a common medium
- Example: Ethernet (many hosts), point-to-point fiber (two hosts)
- The Internet is sometimes called a *catenet*: multiple networks joined together
- Networks are connected by *routers*



The Internet Architecture

- Basic principle: “IP over everything and everything over IP”
- Multiple applications use multiple transport protocols—but there’s only one IP
- IP speaks to multiple network devices
- Developed under DARPA (Defense Advance Reseach Projects Agency) sponsorship; replacement for the old ARPANET

The “Hourglass” Model



The Network Stack

7	Application	
6	Presentation	Unused in TCP/IP
5	Session	Unused in TCP/IP
4	Transport: TCP, UDP, etc.	
3	Network: IP	
2	Link: WiFi, Ethernet, etc.	
1	Physical: radio, fiber, etc	

The Real Network Stack?

9	Political	← YOU ARE HERE
8	Financial	
7	Application	
6	Presentation	Unused in TCP/IP
5	Session	
4	Transport: TCP, UDP, etc.	
3	Network: IP	
2	Link: WiFi, Ethernet, etc.	
1	Physical: radio, fiber, etc	

Roughly Speaking

Application	Does what you really want, e.g., web
TCP	Presents a reliable byte stream to the applications
IP	Gets packets from “here” to “there”
Link, Physical	Deals with the hardware

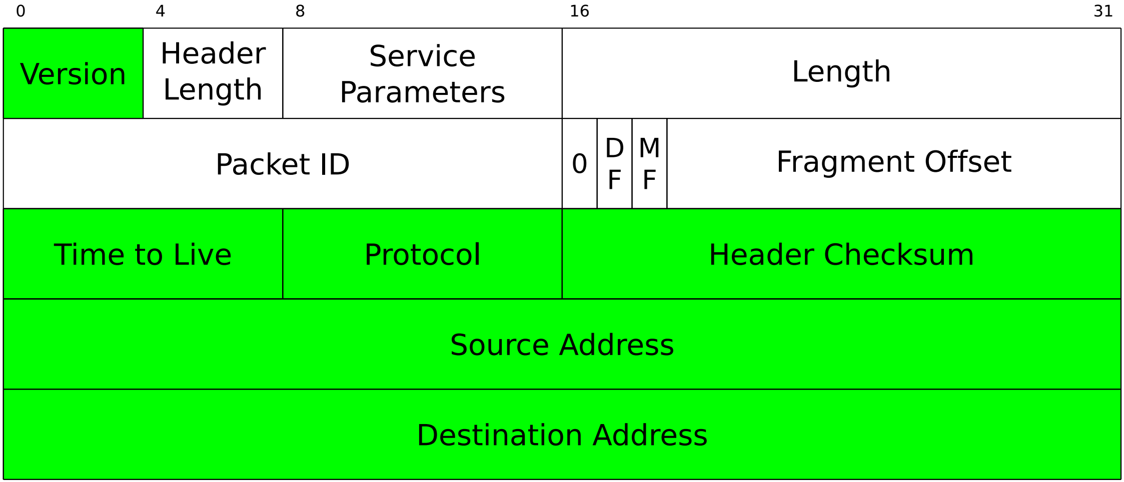
The IP Service Model

- The underlying networks provide a possibly unreliable “datagram” sevice
- Unreliable: packets may be dropped, damaged, duplicated, reordered
- Packets are forwarded to the next hop to their eventual destination—or dropped if not deliverable
- Packets may be dropped because of network congestion
- Very little concern for the correctness of any packet
- *Stateless* forwarding—what happens with a packet does not affect what happens to the next packet
- Note: this is the *service model*—implementations can behave differently to optimize things if they wish

Datagram

- A single message
- No call setup required; every message contains a source and destination
- Note: there are related terms: frame, packet, segment, etc. I'll use packet and datagram interchangeably for all of these concepts, but a networking course would be more precise

IP: The Internet Protocol



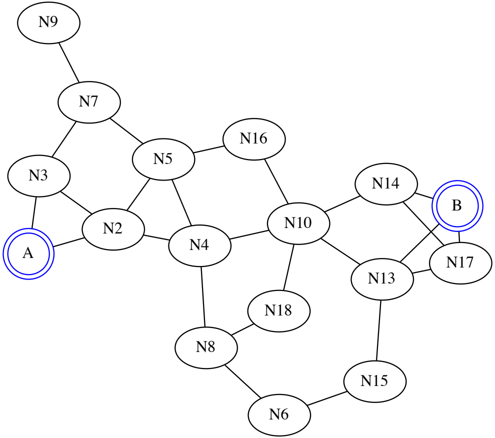
IP: The Internet Protocol

Version	The originally deployed Internet Protocol was Version 4. Version 6 is (very slowly!) being rolled out; I'll say very little more about it
Time to Live	A hop count for packets, to prevent infinite forwarding loops
Protocol	The next protocol: TCP, UDP, etc.
Header Checksum	Validate the correctness of the IP header and only the IP header. Probably a mistake to have included it.
Source Address	Where the packet came from
Destination Address	Where it's going
Other	This isn't a networking class; I won't discuss them. . .

IP Addresses

- IP addresses are 32-bit numbers (for IPv6, 128-bit)
- They in fact have some structure, but that's for a later lecture
- Most IP addresses must be globally unique. Some (so-called "RFC 1918" addresses) are for local use and are translated at the site boundary to a global address

A Network Diagram



Node *A* Talking to Node *B*

- A process on host *A* wants to send a packet to host *B*
- The IP layer on *A* examines the destination address and decides which node (generally a router) is the best next hop
- (How that decision is made will be covered in another lecture)
- It asks its link layer to send the packet there
- The next hop receives the packet via its IP layer. The IP layer either accepts it locally (and sends it to TCP) or forwards it another hop
- Eventually, it arrives

Talking to the Link Layer

- Hosts are trying to send to IP addresses
- Link layers often have their own, very different addresses: for Ethernet and WiFi, these are 48-bit numbers. A mapping is necessary
- The host sends a *broadcast messsage* using the Address Resolution Protocol (ARP)
- All nodes on the local network receive it; the proper receiving machine replies with "here is my link-layer address"

The TCP Service Model

- The purpose of TCP is to convert an unreliable set of packets to a reliable, unstructured byte stream
- That is: IP hands possibly-damaged packets to TCP. TCP handles ordering, damage detection, and retransmission
- TCP also separates IP packets into multiple connections
- In other words: IP deals with unreliable datagrams; TCP produces a set of reliable *connections* (sometimes call *circuits*)

TCP: The Transmission Control Protocol

<table>
<tr><td>0</td><td>8</td><td>16</td><td>24</td><td>31</td></tr>
</table>

<table>
<tr><td colspan="10">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="11">Sequence Number</td></tr>
<tr><td colspan="11">Acknowledgement Number</td></tr>
<tr><td>Data offset</td><td>Reserved</td><td>C W R</td><td>E C E</td><td>U R G</td><td>A C K</td><td>P S H</td><td>R S T</td><td>S Y N</td><td>F I N</td><td>Window</td></tr>
<tr><td colspan="10">Checksum</td><td>Urgent pointer</td></tr>
<tr><td colspan="11">Options, if any. . .</td></tr>
<tr><td colspan="11">User Data (if any). . .</td></tr>
</table>

TCP: The Transmission Control Protocol

Source Port	Part of the connection identifier
Destination Port	Part of the connection identifier
Sequence Number	The sequence number of the first data byte in this packet
Acknowledgment Number	The sequence number of the last byte succesfully received
ACK	The Acknowledgment Number field is valid
RST	Reset this connection
SYN	Technically, "synchronize sequence numbers"; more intuitively, part of creating a connection
FIN	"Finish", i.e., start tearing down this circuit
Checksum	Error detection—is this packet correct?

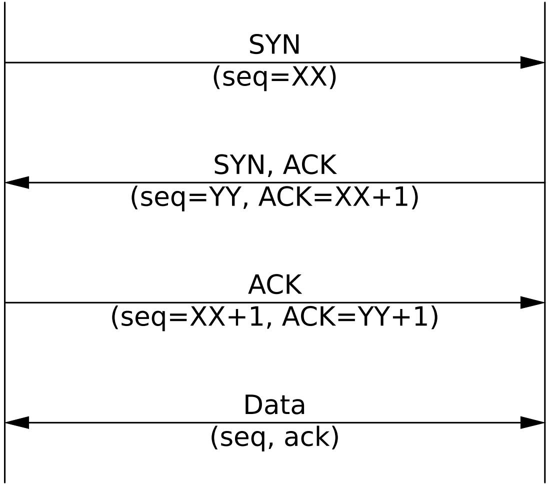
TCP Connections

- A TCP server *listens* on a given *port number*
- Most of the time, this is a “well-known”, i.e., standardized number: HTTP is port 80, SMTP (for email) is port 25, HTTPS is port 443, etc.
- (To a first approximation, you can tell what a connection is used for from its server port number)
- A TCP client is (generally) assigned a random, unused port number by the kernel; it attempts to *connect* to a server by the server’s IP address and port number
- The ⟨source address, source port, destination address, destination port⟩ is the *connection identifier*
- Multiple connections can exist between a single pair of hosts: the client’s port number will be different

Creating a TCP Connection

- The client sends a packet to the server with the *SYN* bit set, saying “this is the sequence number of my first byte”
- This creates client-side state
- The server’s reply has the *SYN* bit set (and “this is the sequence number of my first byte”), the *ACK* bit plus an acknowledgment of the client’s initial sequence number
- Note that at this point, the server has to create state for this half-opened connection
- The client’s reply has only the *ACK* bit set, plus n acknowledgment of the server’s initial sequence number
- This is called the *three-way handshake*, which takes 1.5 round trips
- The connection is not fully open until the server receives this last message

The Three-Way Handshake



Error Handling by TCP

- When TCP receives a packet, it validates the checksum
- If the checksum is invalid, i.e., if the packet is damaged, it is silently dropped
- No, TCP doesn't send negative acknowledgments
- If the sequence number is for the next byte expected, the receiving machine sends an *ACK* packet, probably with no data, but with an acknowledgment number of the next expected byte
- If there's a "hole", it silently saves the received packet until the hole is filled
- Senders wait a certain amount of time for acknowledgments. If they don't get one, the packet is retransmitted.

Error Handling *Only* in TCP

- Why doesn't the IP layer drop damaged packets?

Error Handling *Only* in TCP

- Why doesn't the IP layer drop damaged packets?
- IP could check (and many link layers do check)—but that's redundant
- TCP has to check anyway
- UDP might not want a check–think OFB encryption
- This is the *end-to-end principle*
- Worth noting: because most links are very reliable (and many have their own checksums), very, very few packets are dropped because of TCP checksum issues

Closing a Connection

- When one side is done sending data, it emits a *FIN* packet
- This packet must be acknowledged by the other side—*FIN* bits count as bytes in sequence number space
- The other side must send an appropriate *ACK* packet
- The connection is completely torn down when both sides have received *ACK*s of their *FIN* bits
- Until then, both sides have to retain state

Timeouts and Resets

- When a host sends a packet and doesn't receive an acknowledgment, it waits a while and then retransmits it
- If too long goes by, it can declare the connection dead and notify the application
- If a host receives a packet and there is no matching connection, it immediately replies with a *RST*—reset—packet

Keep-Alives

- Per the last slide, a host can only tell that a connection is dead if it tries to send something that is never acknowledged
- What if a server is waiting for client input but it *never* comes?
- The server has to retain state the entire time!
- Answer: *keep-alives*
- A keep-alive is (effectively) a NOP packet whose sole purpose is to elicit a response. They're sent after a suitable period of silence—generally several minutes to several hours
- If the keep-alive packet does not receive an acknowledgment within the proper interval, the connection can be declared dead and the state discarded

UDP: The User Datagram Protocol

- Some services, e.g., Voice over IP (VoIP) are better suited to a datagram model
- That is, they can tolerate occasional lost or damaged packets, but cannot deal with delays for retransmissions
- UDP provides the same datagram model as IP; the port numbers are used for demultiplexing
- The checksum is optional; if it's received as zero, the validation is skipped
- Note well: no kernel state is created by the receipt of a UDP packet; it's delivered to a listening application, if any, or discarded

UDP: The User Datagram Protocol

0	8	16	24	31
Source Port		Destination Port		
Length		Checksum		
User Data (if any)...				

The HTTPS Hourglass

- For a number of reasons, especially firewalls and encryption, many new protocols are layered on top of HTTPS
- Example: Dropbox, Apple's iMessage, and more
- In other words, HTTPS is not just web servers

TCP in Linux

Client

```
struct sockaddr_in dest;

/* (Somehow) put the IP address
   and port number of the server
   into "dest"
*/
fd = socket(AF_INET, SOCK_STREAM, 0);
connect(fd, &dest, sizeof dest);
```

Server

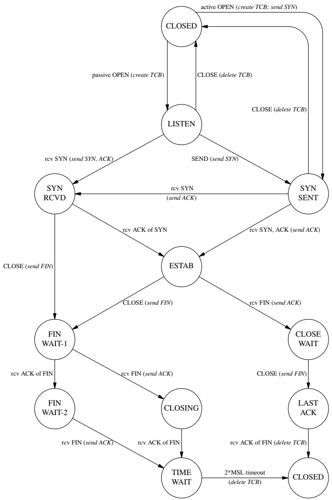
```
struct sockaddr_in us, src;

/* (Somehow) put the service port
   number into "us"; zero the rest
   of it
*/
listenfd = socket(AF_INET, SOCK_STREAM, 0);
bind(listenfd, &us, sizeof us);
listen(listenfd,  5);
newfd = accept(listenfd, &src, sizeof src);
fork();
```

That’s a Lot of Information. . .

- Yes, this is a rapid, deep dive into complex material
- Unfortunately, most of the tutorials I’ve found online go into far more depth on stuff I’ve skipped, e.g., the TCP state diagram
- Most of the rest of the semester depends at least in part on this material

TCP State Transition Diagram



Basic Network Security

Network Security

- The remaining lectures cover what is sometimes called “network security”
- Most of the time, that’s wrong—there’s nothing wrong with the network
- The network is the mechanism by which bad applications are attacked
- (Highway robbery generally does not involve stealing a piece of the road. . .)

Well, Sometimes it Does. . .

Claim: The Designers of the Internet Ignored Security

There are many claims about the Internet being designed without regard to security. Most are incorrect.

Bad design
: **False**. Most of the problems are due to buggy applications, not the design of the Internet

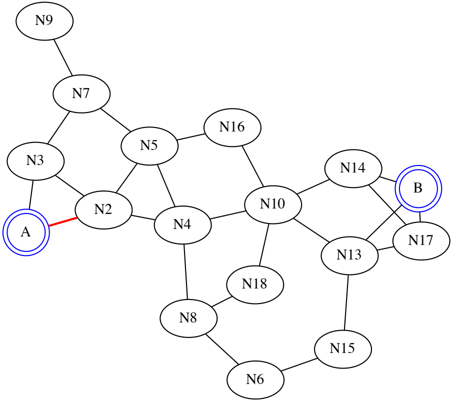
Authentication
: **False**. The Internet couldn't mandate authentication, because different operating systems do things different ways—and why should you trust the sysadmin of a random remote site?

Encryption
: **Partly true**. It was assumed that encryption would be provided outboard to the hosts—but that assumed that link-layer and network-layer encryption was all that was necessary. This is wrong

Ignorance
: **Unknown**. We don't know what attacks were known or thought of back then. Some were definitley known to the NSA; others were known but (wrongly) considered infeasible.

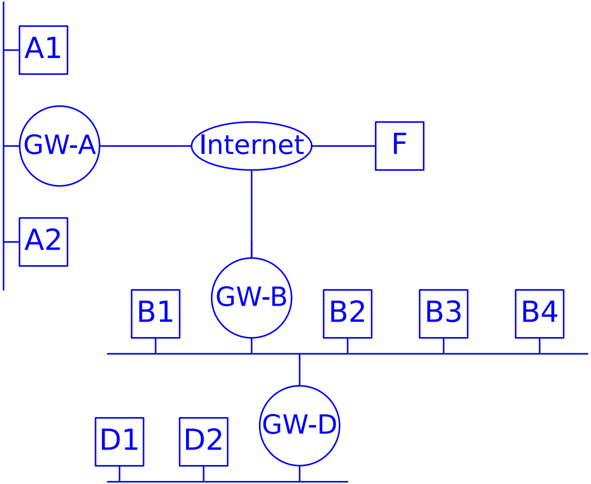
Simple Attack: Eavesdropping

- We're worried about eavesdroppers—let's encrypt the link from *A* to *N2*
- What if the attacker targets a different link, e.g., *N2—N4*?
- Will traffic flow *N10—N14—B* or *N10—N13—B*?
- What if the operator of *N2* is corrupt or the node is hacked?
- We need *end-to-end* encryption, not *link* encryption
- But link encryption is sometimes useful anyway



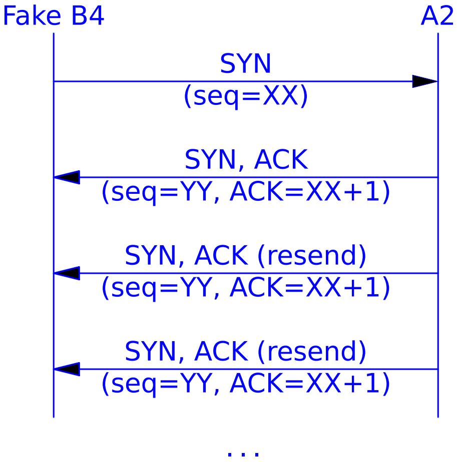
Address Spoofing

- Suppose that host *D1* (somehow) steals *D2*'s IP address
- How will *A1* know if it's talking to the real *D2*?
- Or suppose that B4 answers an ARP query intended for B1? ARP isn't authenticated
- Remember when I talked about the standard cryptographic threat models. . . ?



State Consumption

- Suppose that *B4* is down and *F* impersonates it, sending a *SYN* packet to *A2*
- The *SYN+ACK*—the second message in the three-way handshake—will go *towards* host *B4*, but will be dropped
- *A2* has created state for the half-open connection, but the connection will never fully open
- Eventually, it will time out, but that will take a few minutes
- Suppose that *F* does that repeatedly. . .



Questions?

(Great horned owl, Central Park, November 17, 2019)