

# Networking 5

---

## Recall

IP addresses  
IP datagrams -- best effort service

## TCP "Virtual Circuit"

TCP -- Transport Control Protocol  
Stream Oriented

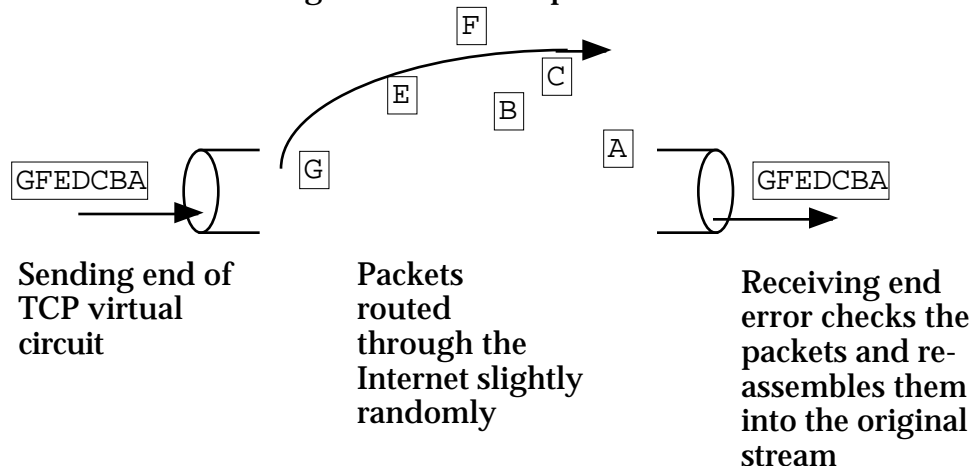
The writer gets to write bytes to a stream -- like a file.  
The writer can just "print" to the stream as if they were writing to a file.  
TCP breaks the stream up into IP datagrams for transmission, but this detail is hidden from the writer.

Number the packets as they go out

Include a checksum with each packet

Error check and re-assemble the packets at the destination

Re-build the original stream and present it to the reader



## Errors

Corrupted packets

Missing packets

Duplicated packets

Out-of-order Packets

TCP Solution

TCP detects all of these cases and fixes them on its own

# TCP "ACK" Strategy

## ACK

TCP uses "acknowledgment" traffic back from the destination to tell the destination which packets have been received correctly.

## Re-Send

The sender can re-send a packet that did not get through

## "In flight" window

The sender will only send so many packets out before some of them are acknowledged. This is the number of "in flight" packets allowed at one time. In TCP/IP terminology this is known as the "window size".

## Flow-Control

The flow of ACK packets also serve to slow the sender down to a rate that the recipient is capable of accepting. If the sender is putting out packets faster than the recipient (or a router on the way) can process, it's a real problem. Error detection is the obvious function of TCP, but actually flow control or (or "flow optimization") is just as important.

# TCP Stream Service

## Stream

TCP gives the appearance of a stream all the way from the sender to the recipient. It hides the underlying datagram, routing, ack, re-assembly details -- it just looks like stream-in and stream-out to the client.

Stream = FIFO

The writer writes a linear sequence, and the reader will see that same linear sequence.

## Bursty Timing

However, TCP cannot hide the irregular "bursty" timing of the traffic.

## Irregular Cadence

The bytes may **look** like a stream, but the cadence (timing) that they arrive will be irregular.

# TCP 3-Way Handshake

## How a TCP stream connection is set up

Each party sends a SYN (request) which is acknowledged (ACK) by the other end. Each party picks a random sequence number (like an id number), that is used by both parties to identify the conversation.

1. Client sends a SYN to the server, including the sequence number the client would like to use
2. Server ACK's the client SYN, and sends its own SYN and sequence number
3. Client ACKs the server SYN (The ACK can be piggybacked on data traffic the client is sending to the server)

## Latency

Notice it takes 3 trips minimum for the client to send anything to the server. Therefore, in TCP, we'll have at least 3x single trip latency. The packets involved are tiny -- the latency of the system dominates.

# Blocking

## Reader Code

Suppose the reading code looks something like...

```
while (defined($line = read())) {
  ## do something with $line
}
```

## Reading

The recipient calls their "read" operation on the socket, but there are no bytes available. The read "blocks" until some bytes are available.

Modern operating systems handle this efficiently -- a process can be "blocked" on a read without using many resources, and when the needed bytes show up, the process gets scheduled for a wake-up.

## Writing

Writing can also block, but it's less common. Writing will block if the recipient cannot accept data that fast, and the local buffers are full.

## The Point

You are used to writing programs where when you call read() or write(), it basically happens instantly. That is no longer true.

Operations can block for significant periods of time. You can use multiple threads in your program so the blocking does not cause your whole GUI to freeze (take CS108 or CS193k for examples).

# Client Abstraction of TCP/IP

## Ignoring Implementation Details

Think about how the Internet looks to a simple computer connected to it -- not thinking about how the routers etc. work, but just what they accomplish.

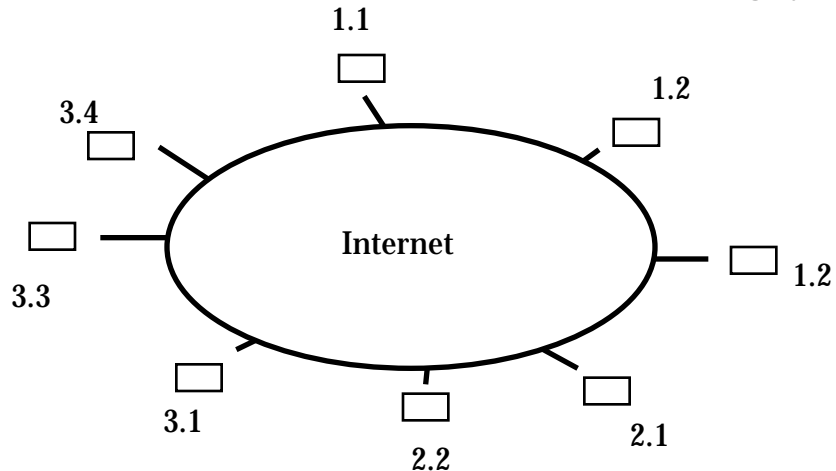
# Phone System

## IP Addr

The TCP/IP Internet is like a phone system that connects all the computers. Each computer has a phone-number. Any computer can place a call to any other computer. Each computer has its own IP addr

## TCP Stream

Through TCP, the sender can write a stream of bytes on their end, and TCP will re-create that stream for reading by the recipient.



# IP Port Numbers

Each IP address is actually subdivided with port numbers in the range 0-65000. Trusted parts of the OS generally use port numbers in the range 1-1024, leaving numbers higher than that up for grabs.

# Services

## Service -- Port Number

By convention, each "service" that a computer wants to implement uses a fixed port number.

HTTP service uses port 80 (web serving)

FTP service uses port 21

## Server

A "server" roughly speaking is a computer that is always on, has a fixed IP addr, and is listening for incoming calls on a particular port #.

## Connect to a particular port #

So to contact the HTTP service on computer 1.2.3.4, we will establish a stream to port #80 on computer 1.2.3.4