

# Networking 2

---

## TCP -- End-to-End Connection

Built on top of the basic IP best-effort datagram service.

End-end reliability -- reassemble and error check the packets at the destination.

Re-send corrupted packets as needed.

Provide the illusion of a continuous, error-free byte stream from the sender to the recipient.

The connection is two-way --at each end, software can write bytes to the connection and the data will show up at the other end.

Flow control -- slow down the sender to a packet rate the receiver and the network in between can cope with. The receiver sends "ACK" acknowledgment packets back to the sender to signal which packets have been successfully received.

## 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.

We'll say that the "client" initiates the connection...

1. Client sends a SYN to the server, including the sequence number the client would like to use
2. Server sends an ACK's in response to the client's SYN, and sends its own SYN and sequence number
3. The client ACKs the server SYN -- the two-way connection is now up. (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 network in between dominates.

## TCP "Virtual Circuit"

TCP -- Transport Control Protocol

aka "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.

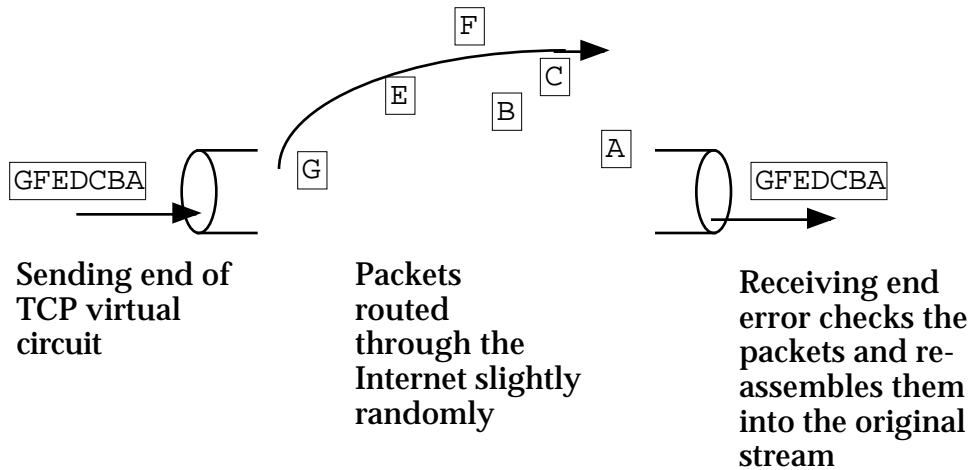
TCP breaks the stream up into IP datagrams for transmission, but this detail is hidden from the writer.

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



## Possible Packet 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, requesting that the source re-send packets as needed.

## TCP "ACK" / Re-send Strategy

ACK

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

Re-Send

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

"In flight" window

The source 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

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.

## TCP Bursty / Irregular

### 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.

### Irregular Sizes

It's hard to say how the data will get divided, re-sent, etc., into IP datagrams.

Therefore, the data may not arrive in the same size chunks as it was sent.

e.g. the Sender writes three 1000 byte buffers. The destination may get 800 bytes, then 500 bytes, then a big pause, and then 1700 bytes

## 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 the 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.

