

# Networking 2

---

## **Babel**

### TCP – End-to-End Connection

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

TCP has these qualities:

- 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 – bandwidth is not a problem here, and the latency of the network in between dominates.

### TCP "Virtual Circuit"

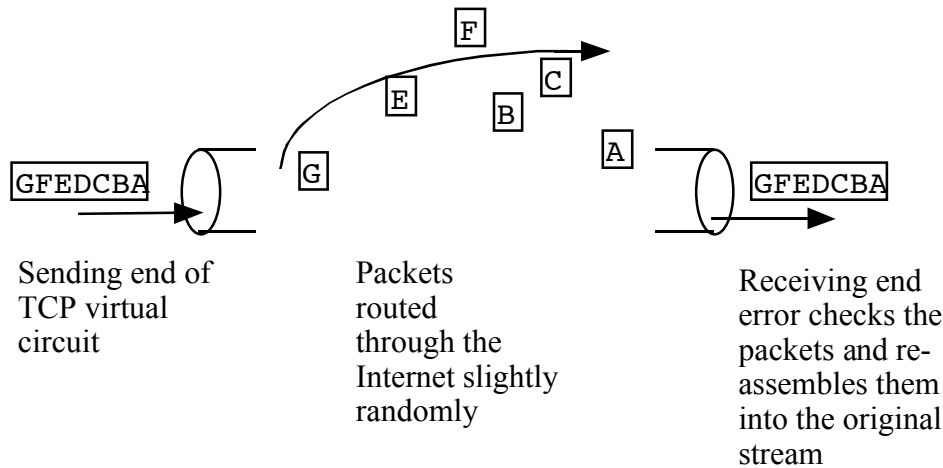
TCP – Transmission Control Protocol  
aka “Stream Oriented”

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 flow of data

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**.

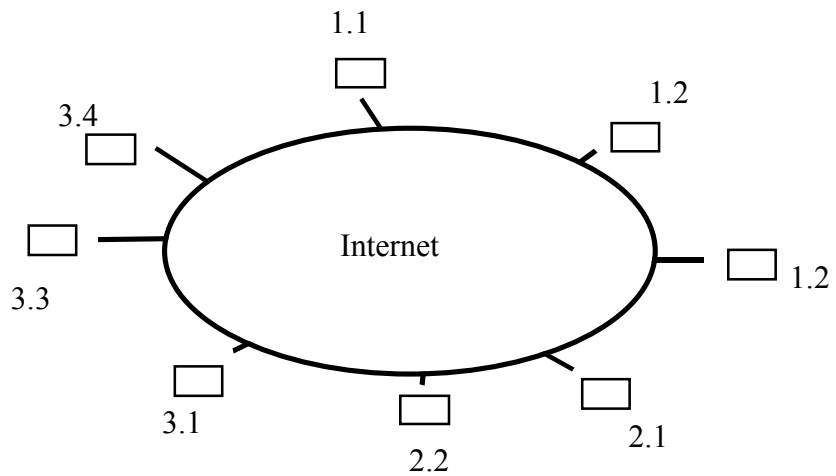
## Phone System

### IP Addr

The TCP/IP Internet is like a phone system 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 address.

### 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.



## Misc TCP/IP Issues

### IP Port Numbers

Each IP addr is logically divided into logical port numbers in the range 1-65535.

Traffic is addressed to a specific port number at the IP – this allows a computer with one IP to be in multiple conversations at once. As we'll see, specific port numbers are reserved for specific purposes – e.g. port 80 is used for HTTP (Web) traffic.

Each end of a connection has a source and destination port number as well as IP addr.

### Broadcast

It's possible to send a packet on the LAN specifically marked as “broadcast,” so everyone reads it. You may be able to see a broadcast packet on your hub if all your “receive” lights blink at once.

TCP/IP also has a "broadcast" notion for sending information to an entire subnet.

### IP to LAN Translation

How does the router know the right LAN addr to use to contact another machine on its LAN? That is, given an IP addr, what is the LAN addr of that machine?

ARP – address resolution protocol

Broadcast “who has IP addr 171.64.64.250” on local LAN

The owner answers back with it's LAN addr (e.g. its ethernet MAC addr if ethernet is the LAN).

Reverse ARP – aka RARP

At boot time, a machine broadcasts its ethernet addr. A RARP server notices the broadcast, and replies with an IP addr to use. DHCP has pretty much replaced RARP.

### Ping

ping – sends a little IP query packet to see if a host responds at all – see if it is up and reachable.

```
ronyeh% ping www.nasa.gov
PING www.nasa.gov.speedera.net (63.210.193.4): 56 data bytes
64 bytes from 63.210.193.4: icmp_seq=0 ttl=54 time=52.151 ms
64 bytes from 63.210.193.4: icmp_seq=1 ttl=54 time=7.7 ms
64 bytes from 63.210.193.4: icmp_seq=2 ttl=54 time=7.385 ms
64 bytes from 63.210.193.4: icmp_seq=3 ttl=54 time=8.078 ms
64 bytes from 63.210.193.4: icmp_seq=4 ttl=54 time=7.825 ms
^C
--- www.nasa.gov.speedera.net ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 7.385/15.194/52.151 ms
```

## Time To Live -- TTL

Neat bit of design – every IP packet has a max - hops counter, so it cannot get stuck in a loop – a robust design, even in the face of router errors. When the packet hits its hop limit, a polite router will send back a “failure” notification, which will identify the IP addr of the router.

## Traceroute

Send out packets with TTL values 1, 2, 3, 4, to probe the path to somewhere.

With firewalls and whatnot attempting to “hide” internal networks, sometimes traceroute won't work and the path appears to go on endlessly.

The times listed are round-trip times in milliseconds

```
elaine40:/usr/class/cs193i/WWW> traceroute www.sjsu.edu
traceroute to rhea.sjsu.edu (130.65.3.13), 30 hops max, 40 byte packets
 1 leland-gateway (171.64.15.97)  1.277 ms  1.371 ms  0.973 ms
 2 Core2-gateway (171.64.1.233)  0.530 ms  0.624 ms  0.513 ms
 3 Core4-gateway-2 (171.64.3.18)  0.897 ms  0.785 ms  0.926 ms
 4 i2-gateway (171.64.1.225)  1.180 ms  0.770 ms  0.777 ms
 5 STAN.POS.calren2.NET (171.64.1.213)  1.125 ms  0.784 ms  1.084 ms
 6 SUNV--STAN.POS.calren2.net (198.32.249.73)  1.562 ms  1.119 ms  1.438 ms
 7 QSV-QSV-C2-GSR-ATM.CSU.net (137.145.203.209)  1.791 ms  2.051 ms  2.078 ms
 8 SJSU-QSV-ATM.CSU.net (137.145.203.106)  286.774 ms  149.992 ms  150.181 ms
 9 firewall.sjsu.edu (130.65.11.6)  138.667 ms  137.488 ms  147.755 ms
10 cclomnicore.sjsu.edu (130.65.11.2)  134.991 ms  134.259 ms  111.274 ms
11 cclomni-a.sjsu.edu (130.65.5.252)  313.109 ms  142.407 ms  146.947 ms
12 rhea.sjsu.edu (130.65.3.13)  137.231 ms * 134.619 ms
```

## Host Configuration

Things that need to be set for a computer to use TCP/IP...

1. IP addr – need to know what IP addr to use
2. Subnet mask – this number encodes what the local set up is for which part of the IP is subnet vs. which part is host addr.
3. Router – the IP addr of the local router to forward traffic to
4. DNS server addr – the IP addr of the DNS server(s) to use.

## DHCP

Dynamic Host Configuration Protocol

Allows a host, at boot time or whatever, to broadcast a query to a local DHCP server. The DHCP server can reply with all of the above configuration values. The DHCP may give out an arbitrary IP addr from its supply, or it may use a specific setup based on the LAN addr of the sender.

This is much more convenient way to do configuration for the end user – just set the use-DHCP checkbox and you're done.

## Standard, Collective, Cooperative

TCP/IP is just a standard -- an agreed format and protocol for things

It's a free, public, open standard

Vendors voluntarily implement TCP/IP

TCP/IP allows each vendor's equipment to, in one step, interoperate with all the other computers which is irresistible

Compare this to the “balkanized” picture where each vendor only works with their own equipment.