

TCP/IP 2

Book

Internet Core Protocols, by Eric Hall. O'Reilly, 2000. \$40. A more detailed, low-level discussion of TCP/IP for the curious.

Routers

"Pass the buck"

A host on a LAN doesn't know much about routing -- it often just forwards all its traffic to its local router.

"Next Hop" table

Routers look at the IP net # of the destination and figure which next router is one closer to the destination -- one hop on the way.

A routing table giving the next hop for various net numbers might 50,000 rows.

No Global Picture

No router has a global, end-to-end picture of the route a datagram should take.

Router Protocol

The routers are constantly talking to each other to collectively decide which routes are best. They can dynamically adjust things as congestion appears or if a link or router goes down.

It's a fascinating area, but 193i will focus on higher level services -- take CS244 for more detailed TCP/IP implementation.

IP Key Points

IP Addresses

Standard namespace for computers participating in TCP/IP

IP Datagram

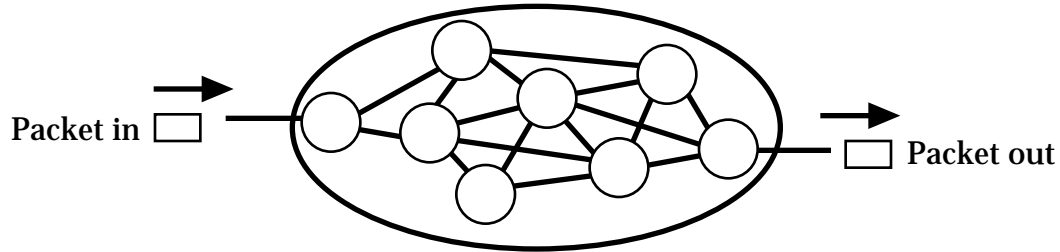
Standard packet format for TCP/IP communication

Uses IP addresses for sender and recipient

IP "Best Effort Delivery"

For the most part, IP is about hopping a single IP datagram to the destination -- more complex communication is built with TCP (below).

IP Datagram Routing -- Hops



Routing of the packet
 -- router by router --
 hop by hop

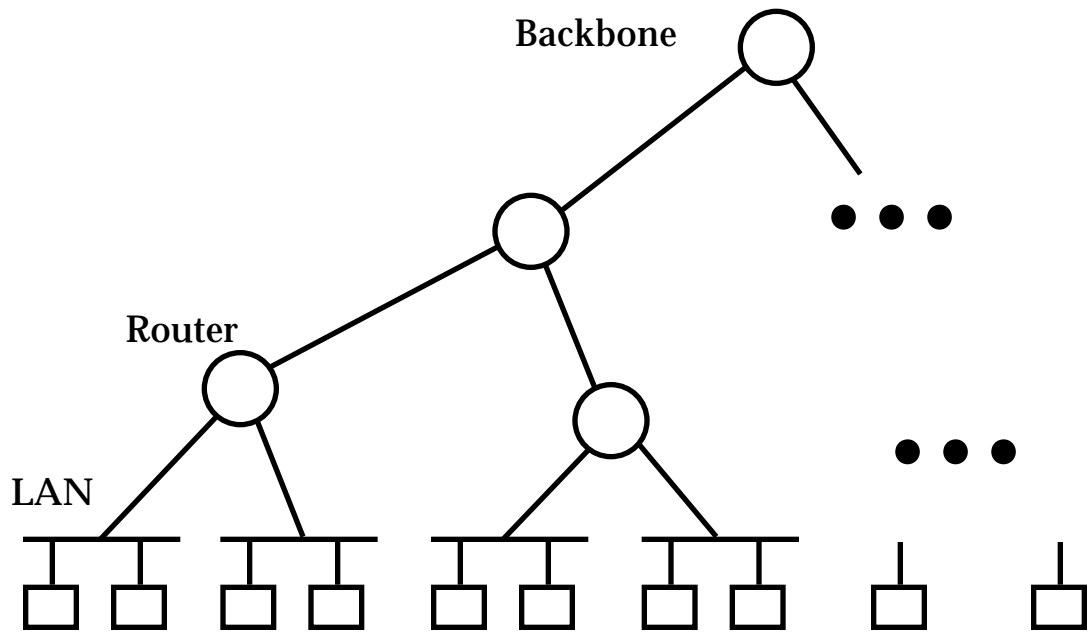
IP Routing Hierarchy

Local vs. Backbone

There are two vague directions -- "downstream" or "local" towards networks smaller than the ones connected to the router, and "upstream" or "backbone" towards larger networks

Up Then Down

In this model, traffic hops upstream until it is high enough that it can get to its destination by hopping down.



Not Really That Simple

The Internet does not look like the tidy hierarchy above. It was grown ad-hoc, and is much more connected, redundant, and ambiguous -- but there are approximate upstream and downstream directions from any point.

TCP "Virtual Circuit"

TCP -- Transport Control Protocol Stream Oriented

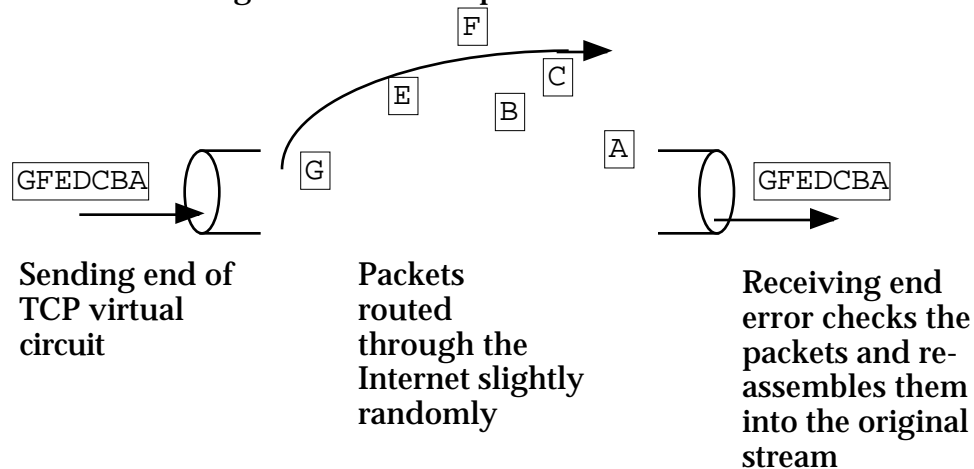
The writer gets to write bytes to a stream -- like a file.

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 Re-assemble them on receipt

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.

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.

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

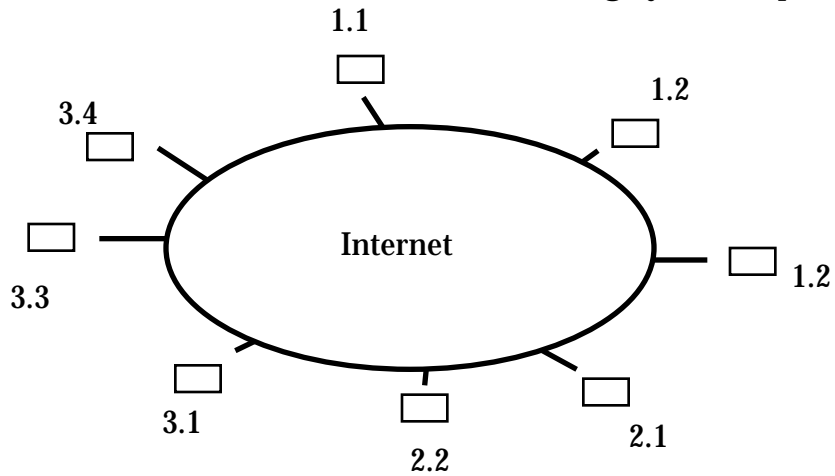
IP Addr

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

Connect to a particular port

So to contact the HTTP service on computer 1.2.3, we will establish a stream to port #80 on computer 1.2.3.

Introduction to Sockets

Socket Interface

The "socket" interface is a standard way for a program to use TCP/IP. It was originally developed as part of Berkeley unix, but it's been widely copied.

It's available in many, many OSs and languages.

We'll look at the functions below in more detail later, but here's an introduction...

1. Socket

Allocates a socket, but does not connect it to anything yet.

2. Connect

IP addr + Port

Given an IP address, and a port #, try to "call" that address -- establish a TCP stream to that addr. The connection may fail for any number of reasons.

Connect == stream

If the connection succeeds, there is now a two-way TCP stream connecting the two parties. Writing and reading on one end exchange bytes with Reading and Writing on the other.

3. Write

Write bytes to the stream. The bytes will, eventually, show up FIFO on the reading end of the other party.

4. Read

Try to read bytes arriving from the other party.

Blocking

Often, there will not be any bytes available at the moment read is called.

Generally, this will cause the reading process to "block" (go to sleep) until some bytes have arrived.

Irregular

The bytes that arrive may be irregularly timed, and in irregularly sized chunks. They will be FIFO however.

5. Close / Shutdown

Close() takes down the whole stream at one end.

"shutdown" can be used to just close, say, writing but not reading.

A close of the writing channel on one end shows up as an EOF on the reading channel on the other end.