

Services

TCP/IP -- Phone Sys Infrastructure

The TCP/IP standard allows computers to identify and exchange bytes with each other.

IP addresses

IP datagrams

TCP connections / sockets

What are you going to build on top of this basic infrastructure?

Service

A service is a protocol used by two or more computers to interact with each other. The term "service" refers to the overall solution, while the term "protocol" suggests more the nuts and bolts of how the dialog works.

e.g. email, FTP, HTTP, ... all the useful services that are built on TCP/IP

Typically, a service is associated with an IP port number -- e.g. the HTTP service uses port 80.

RFC

Request for Comments -- the term for the document typically used to define an Internet service standard.

Each RFC gets a number, and you can look at them all at www.ietf.org

Behind every useful two-way dialog between computers, there's a humble old RFC document which allowed it to happen.

Many RFCs use the term "octet" instead of "byte". Apparently, "byte" is a naughty word in French.

Internet Standards

The secret of the Internet

Q: How did the Internet come about? A: Standards

Networking applications are about multiple computers talking to each other

Standards are key, compared to say, a desktop application

e.g. MS Excel file format -- works fine as a proprietary MS-only format. It is not necessary that other vendors are able to "speak" Excel format.

e.g. Email -- only has value if other parties, vendors etc. understand it.

N^2 Network Effect -- Metcalfe's law.

Example of Standardized Services

All the interesting things you can accomplish on the Internet are actually services built on the TCP/IP structure...

DNS

Domain Name Service -- the service computers on the Internet use to look up names like "www.yahoo.com" to find their IP addresses. Implemented as a distributed database.

FTP

File transfer protocol -- move a named file from one computer to another (does not encrypt the password)

SCP

Secure Copy -- like the unix CP command, but over the Internet. Encrypted, so this is safe (unlike FTP).

Ping

Contact a host just to see if it's up and reachable

Finger

Ask a host if it knows about a particular username

Telnet, SSH

Establish a command line interface to a login on the server. (SSH is a newer service that encrypts the password and session data.)

SMTP

Simple Internet Mail Transfer Protocol -- Send/forward Internet email

POP

Post Office Protocol -- The owner of an email box retrieves their arrived mail.

IMAP

Internet Mail Access Protocol -- An alternative to POP where the email remains up on the server -- it is not copied down.

HTTP

Hypertext Transfer Protocol -- The "world wide web" service -- how a client uses a URL like `http://www.stanford.edu/class/cs193i/` to contact a server, make a request, get back a web page.

Client Server Service

Services on the Internet - Servers/Daemons

"Server" (hardware) = computer on the Internet, running all the time

"Server" or "daemon" (software) = a software program which is running on a server. The daemon sits around waiting on a particular port #, receives incoming calls, processes them, and may make outgoing calls.

"web server" = (hardware) the machine, (software) the web daemon running on the machine, processing web requests

Typically, the client program contacts the server (using the defined port #) and initiates the dialog.

RFC defines the rules of dialog between the client and the server and the port #

1. Traditional PC Applications

Everything local: PC has brains, the nice display, the pointing device, the software and the data.

Operations are responsive.

e.g. word processing, spreadsheets -- still the best solution for many applications
 Pro: CPU/memory near screen makes for much more responsiveness/activity
 Con: local copies of things, the local PC requires more admin than the dumb terminal. Harder to share with others (but that's what the Internet is for).

2. Client/Server

Client program runs on "PC" with user -- provides responsive human interface (HI) facilities. Elements of the computation or data are centralized on the server where appropriate

Client is good at being responsive and graphical to the person physically right there. It's hard for the server to be responsive because of the network latencies.

Server is good to centralize things -- where you want a single, synchronized copy of a resource, or the resource is so large or otherwise expensive that you only want to have one copy. If something is centralized on the server, then only one person needs to do a good job of administering it. Paying people for their time is one of the few things in the Internet which costs a significant amount (remember: bits are cheap), so reducing the amount of administration which needs to happen is interesting.

Pro: responsive/facile computer at user end

Pro: centralize things which should be centralized

- lot of data, and you don't want to have a local copy (local copy gets out of synch easily) -- the "never have two copies of anything" software rule.

- particular type of computer necessary (parallelism)

- reduce the amount of administration work

e.g. Google

e.g. Encyclopedia -- don't want your own local copy, you want easy access to a centralized copy which is always up to date.

Con: more complex to build, networking latency may make client unresponsive, when the network is down, it doesn't work.

Cute comparison of CD/DVD disks vs. connectivity.

3. Web Application -- Thin vs. Thick

Web application -- a client/server application expressed through a browser with HTML and HTTP

e.g. Amazon -- book research and buying service, expressed through HTTP

e.g. Google, Yahoo, ...

This is also called "thin client", since the real processing is on the server side. The client just does presentation.

Thin client is easy on the client side, since it's just a standard browser (i.e. nothing special to install -- yay!)

Thin client can be awkward to implement and have a limited GUI, since all the real logic is moved to the server side.

Thick Client

AIM, Napster

These are networked apps, but not through the web. They have custom clients.

This may be the trend of the future, since HTML/HTTP are limiting in some ways. On the other hand, the web browser/thin client is standard, so it may dominate forever.

Telnet Trick

Text

Many services use a textual dialog between the client and server -- short phrases sent back and forth as lines of text.

Telnet

Telnet to the appropriate port number, and just type the commands by hand...

`% telnet host port`

"Transparency"

Systems are easier to debug if their state is visible and accessible. This is an advantage of text-based protocols.

ASCII standard

Using plain ASCII characters also provides portability. Every system in the world understands ASCII, and because each letter is a single byte, it does not depend on big-endian little-endian issues that make binary data harder to port between systems.

Telnet - POP

(things typed by me are in bold)

`[localhost:~] nick% telnet example.stanford.edu 110`

Trying 171.64.14.86...

Connected to example.stanford.edu.

Escape character is '^'.

+OK example.Stanford.EDU Cyrus POP3 v2.0.12 server ready

USER nick

+OK Name is a valid mailbox

PASS foobar

+OK Maildrop locked and ready

STAT

+OK 2 1242

+OK

QUIT

Telnet - HTTP

`[localhost:~] nick% telnet www.stanford.edu 80`

Trying 171.64.14.237...

Connected to www.lb-a.stanford.edu.

Escape character is '^'.

HEAD / HTTP/1.0

HTTP/1.1 200 OK

Date: Mon, 23 Apr 2001 18:24:04 GMT

Server: Stronghold/2.4.2 Apache/1.3.6 C2NetEU/2412 (Unix) mod_fastcgi/2.2.4

Connection: close

Content-Type: text/html