

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

Need agreed richer structure and interpretation of the raw bytes

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.

Typically, a service is associated with an IP port number -- e.g. the HTTP service uses port 80, since the TCP/IP is the dominant networking standard.

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?

Q: 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 as a proprietary MS-only format

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

Free Public Standards

Any time two computers want to co-operate to get something done, there needs to be a standard (a "protocol") that they both follow so the interaction works.

The Internet is entirely about pairs of computers communicating.

1. Publicly available

The standard needs to be well-defined and publicly available.

There may be test suites that verify that an implementation is true to the standard.

Something controlled by one company, kept secret, and changed without notification is not a standard.

2. Unencumbered -- freely implementable

The technology in the standard, typically, should not be patented, or should have generous licensing terms. Anyone should be free to implement the standard.

The standard may come from a particular vendor, but it only works as a standard when anyone is free to implement it.

3. Well Managed

The standard needs to be managed over time: allowing for improvements without breaking compatibility. Typically, this requires an organization that is not too tied to any one vendor. The interests of a vendor, and the overall interests of the standard participants not the same.

Vendors need to take compatibility seriously -- breaking backward compatibility according to their own interests, not the interests of the community.

Submarine patents

Standards are the key ingredient in the Internet, and vendors know it.

To "tax" the standard, a vendor might obtain a patent, hope that the technology is incorporated into a standard, the standard becomes popular over several years, and then the company could extract payment from everyone using the standard. Usually standards avoid patented technology, but in this case, the patent holder tries to keep the patent low-profile while the standard develops. In any case, the current software patent system is deeply flawed: patents are awarded for very obvious technologies and the patents are then used as a club to avoid competition.

Standards Based Networks -- N^2

Incredible Growth -- further questions

We'll study this more later, but at this point we can notice that well defined standards can bring tremendous growth.

TCP/IP, HTML, HTTP -- these are all standards

Indeed, the incredible value potential of standards based networks is the lesson of the Internet.

Metcalf's law -- Network Effect -- n^2

The usefulness, or utility, of a network equals the square of the number of users. Robert Metcalfe -- inventor of Ethernet, founder of 3Com

The standards allow separately authored components to interoperate with each other thing -- n^2 connections. We may not know the exact mechanism, but recent history shows that standards based n^2 networks create a lot of value.

Co-operative Standards vs. Individual Vendors

Microsoft, IBM, Oracle -- these are all dominant vendors in their domains.

And yet the Internet did not arise using anything controlled only by them. The Internet arose with the collective participation of everyone through

standards. The Internet is about co-operating through standards to create value. Even in the face of a vendor with 90% marketshare, the power of co-operation is greater -- it's an incredible story and an important lesson.

Standards vs. Markets

It's hard for vendors driven by market forces to make good standards (even though in reality, the vendors come out ahead once the standard exists). The TCP/IP, HTTP, email, .. standards ... these were all produced by non-profit groups, often with government funding. I think markets are great for some things, but standards are an interesting and important area they get very wrong.

Participant value vs. Vendor value

Because of the standards, no one vendor gets to monopolize the value. It's not like there's some "owner" of TCP/IP that gets all the value out of it. The TCP/IP **participants** collectively receive the value of TCP/IP.

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 "web server" 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.

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

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, ...

Non Web Applications

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

Example EMAIL

Anatomy of an email message send

Sender and recipient both have "accounts" on computers (source and destination hosts)

Compose message on source computer. Email is text based. Binary files such as GIF must be encoded to fit in an email message.

"Send" == hand off to "sendmail" daemon on source computer. If you're running a command line on the Unix host, you pass off to sendmail directly. If you are running on your PC, your mailer (Eudora, ...) will make and SMTP connection to your designated local SMTP server, and it will handle the mail for you from there.

Source sendmail daemon tries to contact "SMTP" daemon on the destination host to send the message.

When contacted, the destination daemon accepts the email on behalf of the recipient. The destination daemon then copies the email into the user's "mailbox" where they will find the next time they check.

Email "Bouncing" back to sender account on err. It was an important for the social development of email that it feel "reliable" -- so there's real effort to notify the sender if the message does not get through.

Elm/Pine reading (old)

Mail files remain at your (Unix) email account.

You connect to your account for reading/sending via Telnet. No matter where you connect from, all of your old and new email is consistently there.

The leland system uses a slight variant of this where the email gets copied down into the users file space as soon as possible -- otherwise the load on the SMTP server to hold the incoming email would be quite high.

POP reading (current)

Mail only temporarily on your email account host

Eudora/Netscape copies the mail down with the POP protocol to your client PC when you read it

Handy GUI interface, but inconvenient to have your email copied down -- you have to always read your email from the same place.

Standard POP sends the password in the clear -- the APOP and KPOP variants avoid this problem.

IMAP reading (future)

Mail remains on your email account host

The IMAP mail reader presents a GUI which allows you to see and manipulate your email conveniently wherever you are while the mail data remains on the server

The best of both worlds

Email Culture or "voice"

Email fills a niche somewhere between phone and letter writing
 quick and dirty -- casual style. Rebirth of letter writing?
 honest/open
 rudeness problem -- like cars

The Inconvenience/Rarity Theory

(Yet another possible thesis topics)

When something is low effort, people "discount" it as a message. Therefore: email "counts" less than a phone call.

Pt: this effect will come up more and more in an increasingly convenient technological world.

The Subject Line

I get lots of email, so I care a lot about email etiquette. I think someday, everyone will get as much email as I do now.

Have a nice, long searchable subject line. Partly so they can see if they need to deal with it now (or ever). Partly so they can find the message easily a month in the future

Is this something I need to deal with, or is this just advisory? Help the recipient know the context without reading the message.

Some people like to put a context word at the very start like "Funny:" "Question:" "Request:" "Suggestion:" or just "Q:" "Req:" "Sugg:"

Good Subjects

LArray link error -- Sent by student to instructor where presumably the expression "LArray link error" means something

Pizza available in room 101 -- Sent to mailing list for whole building

Recommendation? -- Sent to an instructor. Short, but pretty clear. Has the advantage that it will show up when I search for all messages that mention "recommendation" to see what I need to do.

Reminder: Recommendations are due tomorrow -- Sent to instructor from student who had requested a rec

FYI: Knuth talk this Tues -- Sent to advisees to mention an interesting talk coming up on campus

Party at my house this Friday 7:00, come as your favorite ex President -- Sent to all my friends. They can read the body if they want, but the subject basically says it all.

Bad Subjects

"Question" -- Sent to an instructor. Sent to mailing lists of thousands of people. You always have to read the message to see if its an action item or just advisory.

Help, info, note -- Same as above -- these are quite useless

Email Etiquette

Have a great subject line is a courtesy to your recipients

Make sure the "From:" field of your outgoing mail has your proper name -- your email is **you** in cyberspace.

To one person, CC the others -- if you have multiple To: recipients, how do they know who is supposed to deal with it?

What are they supposed to do with this: Action item? Question? FYI?

Beware of "reply to all"

Don't lambaste in public -- private criticism, public comment

When sending email, consider that it may get forwarded around or retrieved off of an archive by the recipient years later.

Send questionable/angry mail you have composed to yourself for a cooling-off.

If you are unsure if something you want to send is rude - send it to yourself, wait a day, when you read/get it you can think about it

If something is sent to you privately, don't assume they would want it forwarded somewhere else.

Don't reply to *everything* with a "thanks" type message. Save for special occasions.

Consider never deleting any of the mail you send or receive. Hard drives are cheap and search engines are fast.

Don't use relative expressions like "today", "tomorrow" -- this applies to web content in general where the time and space of the reader is quite disconnected from the author.

Don't express things you would not want to stand behind. This seems like a reasonable sentiment in any world, but the Internet culture makes it more of a practical reality. Email has a way of resurfacing. Be a person who says what they mean and is not afraid to stand behind it -- even the dumb things you say.