

Services 2

Standards Key Points

Standard

Freely available, well defined standard.

Something controlled by one company, kept secret, and changed without notification does not count.

It's possible for a company owned technology to act as a standard, so long as the information is public and well-defined.

Cooperative

The standard allows separate computer systems to cooperate. Each contributes a little by being compatible, and everyone benefits.

Compatible

It's about being compatible. By being compatible with the standard (which may have some costs) your system now interoperates with everyone else who has volunteered to be compatible with the standard.

n^2 Value

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.

Not proprietary

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.

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.

Services on the Internet - Hosts/Servers/Daemons

host = computer which connected to the Internet and which is on all the time

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

"web server" = the host which is running the web daemon.

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

An anthropomorphic "daemon" program listens on the specified port number on the server side and handles incoming client requests.

RFC defines the rules of dialog between the two and the port #

Traditional PC Computing

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).

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.

Cute comparison of CD/DVD disks vs. connectivity.

Example #1 -- DNS

Client program does a DNS lookup to get the IP addr of the DNS name. e.g. "www.yahoo.com" or "elaine23.stanford.edu".

Contacts the local DNS server. It may know the answer right away. More likely, the local DNS server will pass the buck up to its parent DNS server.

The buck-passing follows a path among the DNS servers until it gets to a server that knows about that domain.

The answer tracks back to the local DNS server, which then (finally) answers the original DNS query.

Example #2 EMAIL

Anatomy of an email message send

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

Compose message on source computer.

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

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

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.