Why is the internet so slow?

...

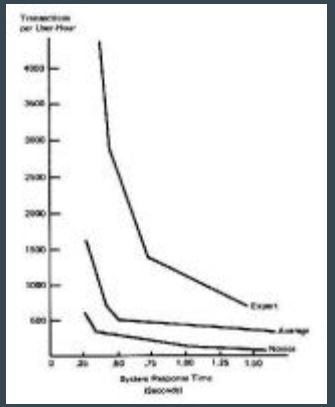
Perry Lorier

But I have gigabit fibre!

- You may have gigabit fibre! Hundreds of MB/s!
- With Google's Stadia I could press a button, and have a character on screen jump within 1/60th of a second!
- So why does everything still feel so slow?

The Economic Value of Rapid Response Time

- This is a paper published in IBM Systems Journal, Nov 1982 by Walter J. Doherty and Ahrvind J. Thadani.
- *"[...] with system response of three seconds, Thadhani found that a programmer executes about 180 transactions per hour. But, bring system response time down to 0.3 seconds and the number of transactions the programmer can execute in an hour jumps to 371, an increase of 106 percent. Put another way,* ***a reduction of 2.7 seconds in system response saves 10.3 seconds of the user's time****. This seemingly insignificant time saving is the springboard for* ***sizable increases in productivity****."*



- This was later called the "Doherty Threshold", assumed you should target below 400ms.

Trying to IM on the tube.

- I was trying to IM to a friend on the tube.
  - There's Wifi at the stations, but not in the tunnels between stations.
- I have ~30s before I lose signal again as the train departs
- 30s should be heaps of time to receive new text messages and send a reply.
- But... it's not. Why?

Wifi Scanning

- First we need to find the WIFI access point.
- This involves “scanning”.
  - Change to each channel in turn
  - Send a “Probe Request” frame
  - Wait for a timeout to see if you hear any Probe Response replies.
  - Then move on to the next channel.
- How frequently does the device start new scans?
  - Trick: Android will scan more often if the Wifi settings is open.
- How long does the device wait on each channel before moving on?

Connecting to Wifi

- (Optionally) Send a Probe Request to get any AP information.
- Send a series of Authentication frames back and forth to setup encryption keys.
- (Optionally) perform an 802.1x authentication.
- Send an Association Request frame.

It typically takes a few seconds to connect to the Wi-Fi service at a Tube station (this delay is due to a security feature). If you're trying to connect at a peak time in a busy station, it may take slightly longer

— https://tfl.gov.uk/campaign/station-wifi

Address Assignment

IPv4

- Send DHCPv4 DISCOVER.
- Receive one or more OFFERs.
- Send a REQUEST.
- Get an ACK.
- (Optionally) Send an ARP to make sure no one else is using the address.
    - And wait for a reply. (1s)

IPv6

- Send IGMP Group Joins.
- Wait up to MAX_RTR_SOLICITATION_DELAY (1s).
- Send Neighbour Discovery for LL address.
- Send a Router Solicitation.
- Get back a Router Advertisement.
- Send Neighbour Discovery for SLAAC Address.
    - Wait for a reply (1s)

Address Allocation - IPv4

- DNSMasq verifies the address it's about to give you is free.
- It sends an ICMP Request to the address.
- Then **waits 3,000ms** (by default) to see if it gets an ICMP Reply.
- During this time it can continue to handle DNS and Router Solicitations.
    - But not other DHCP Requests.

- This also causes other problems (https://bugs.openwrt.org/index.php?do=details&task_id=4189)

Address Allocation - IPv6

- Wait a random 0s...1s delay at startup before sending router solicitation.
- Duplicate Address Detection
  - Waits for 1s.
  - Optimistic DAD - assumes it’s going to work, so you don’t have to wait for the 1s delay.

DNS Lookup

- Send a DNS question to a DNS server learnt by DHCP or RDNSS.
- DNS is typically sent over (unreliable) UDP.

DNS Lookup: Timeouts

- DNS Protocol doesn't have a "Ack, I've received your query and I'm working on it"
- DNS recursive might need to do lots of lookups to get the answer you need.
- So how long do you wait until you retry?
  - RFC 1035 Section 4.2.1: (November 1987)
    - "Depending on how well connected the client is to its expected servers, the minimum retransmission interval should be **2-5** seconds."
  - RFC 1536 Section 6.1.3.3 (October 1989):
    - "When a DNS server or resolver retries a UDP query, the retry interval SHOULD be constrained by an **exponential backoff algorithm**"
    - "A measured RTT and variance (if available) should be used to calculate an initial retransmission interval. If this information is not available, a default of no less than **5 seconds** should be used."
- So Linux (to this day) uses 5s, 10s, 15s as it's default retransmit schedule for DNS.
  - Windows uses 1s 1s 2s 2s 4s
- add "`options timeout:1`" to `/etc/resolv.conf` to make your internet go faster.

First IPv6 Packet Problems

1. A host can receive a Router Announcement that includes SLAAC information.
2. The host can form up it's IPv6 address.
3. The host then sends a packet to it's default router (from the RA above)
4. Packet goes out to the Internet, and a reply comes back to the default router.
5. The default router doesn't have the host in its neighbour cache, so does Neighbour Discovery.
    - While doing ND it only caches a ***single*** packet, with taildrop.
    - So second and subsequent packets are dropped.

- https://datatracker.ietf.org/doc/html/draft-linkova-v6ops-nd-cache-init

Aside: ICMP Redirects

- To avoid all hosts on a subnet to hav to have full knowledge of routing all the time, IP (both v4 and v6) support ICMP Redirects.
- A host sends to the default gateway.
- If there is a better gateway, the default gateway sends a “ICMP Redirect” to host telling it about the direct gateway.
- The host caches the direct gateway in the host’s routing table.

Aside: ICMP Redirects: Error Recovery

- If the direct gateway fails, then the host should give up and go back to the default gateway and see if it can get a better route.
- A gateway is considered failed if we start getting “network” ICMP Errors.
  - eg: ICMP Destination Network Unreachable
  - but not if we get non-network ICMP Errors: eg: “ICMP Destination Port Unreachable”

Network Unreachable
Host Unreachable
Fragmentation Required
Source Route Failed
ToS Network Unreachable
ToS Host Unreachable

Protocol Unreachable
Port Unreachable
Network Unknown
Host Unknown
Host Isolated
Network Administratively Prohibited
Host Administratively Prohibited
Communication Administratively Prohibited
Host Precedence Violation
Precedence cutoff in effect

Aside: ICMP Redirects: Linux Bug

- Linux implements ICMP Redirects
- But has a bug, where it doesn't report "Network" ICMP Errors back to userspace.
- Unless you set `SO_RECVERR` to receive extra error information.
- Bug is from Linux 0.98 with the original import of networking into Linux (and support for up to massive 32MB of memory)!
- https://bugzilla.kernel.org/show_bug.cgi?id=202355
  - "This is just how Linux works..."

DNS Lookup: Linux Bug

- Linux doesn't report Destination Port Unreachable by default forcing a timeout.
- When combined with default glacial DNS timeouts this is terrible!
- Fixed:
  - systemd-resolved: https://github.com/systemd/systemd/issues/10345
  - glibc libresolv: https://sourceware.org/bugzilla/show_bug.cgi?id=24047
  - bind: https://gitlab.isc.org/isc-projects/bind9/-/issues/835
    - libuv: https://github.com/libuv/libuv/pull/2872
  - powerdns: https://github.com/PowerDNS/pdns/pull/7540
  - unbound: https://github.com/NLnetLabs/unbound/issues/781 (unfixed)
  - dnsmasq: Doesn't respond to ICMP errors - doesn't even store the query.

And then there's this whole thing:

SO_BSDCOMPAT

> Enable BSD bug-to-bug compatibility. This is used by the UDP protocol module in Linux 2.0 and 2.2. If enabled, ICMP errors received for a UDP socket will not be passed to the user program. In later kernel versions, support for this option has been phased out: Linux 2.4 silently ignores it, and Linux 2.6 generates a kernel warning (printk()) if a program uses this option. Linux 2.0 also enabled BSD bug-to-bug compatibility options (random header changing, skipping of the broadcast flag) for raw sockets with this option, but that was removed in Linux 2.2.

DNS Timeout: Measurement

- So how long *should* we wait?
- Capture every DNS Packet going in/out of my home network for 18 months.
- What's the longest it takes for a *successful* reply to come back?
  - 134 seconds (!!!)

Then there's all the other stuff:

- TCP 3-Way Handshake
- TLS Handshake
- Nagle adding extra delays
  - QUIC 0-RTT can avoid much of this
- HTTP Request + Response
- Tail drop queues
  - CoDel
- Bufferbloat