LAN

- Ignore the
- LAN is where everything starts
    - You connect devices to a LAN
    - First networks were
    - First useful network had many devices, connected to a common network – Started from shared topology
- Leads to 2 better concepts:
    - domain – one standard (media, modulation, encoding, ...)
    - domain – my device can talk to anybody on the LAN (without help)

Guarantees

- A good LAN design is really simple
- Get a message from here to there, as fast/efficiently as you can
    - Over a particular 'cable'
- **No**
- **No**
- **No**
- Leave all those hard things to the software – in general

Multiplexing

- Multiplexing:
- Everyone gets
- But not everybody needs that much – It's effectively a circuit – Wasting capacity

Statistical Multiplexing

- Demand for capacity varies with time

– Random across all devices

– Statistically:

- Don't need all of the bandwidth, all of the time
- Share the bandwidth fairly and easily

– Whoever needs to, can send, when they want – Probably need to control that...

– However much you want to send, you can send it all – Definitely need to control that

Circuits, Cells and Frames

- **Circuits** – other end
- **Cells** – other end
- **Frames** –

Designing a Frame

- Need to specify where it's going = destination address
- Need to specify where it's come from = source address
- Need to specify where it's assemblage of bits starts and stops
- Need to agree how long a frame could be

– Infinitely long is not acceptable

– Don't

-
- Need to agree how to access the network fairly

Simple Frame

Could compress this down to just Given framelength, don't need trailer You just need to be/stay in [redacted]; Easy to make a mistake

Better Frame

A frame starts and ends with a 'Flag'

- – The frame length is what's between (including) some 'flag' bytes
  - Regain sync by [redacted]
- – Slight wrinkle: 'flag' byte inside the payload
  - Put an [redacted] any [redacted] bytes inside the payload
- – New wrinkle: 'escape' byte or 'escape/flag' byte pair in the payload • Need to [redacted] the [redacted] – and so on. • [redacted] stuffing...

MAC and sharing

- M[redacted] A[redacted] C[redacted]
- Needs an [redacted] scheme
  - – 'MAC address' – hardwired (sort of) to your network interface – [redacted] not the computer
  - – Listen to (receive) all traffic, [redacted] stuff sent to you
- Needs an [redacted] scheme for multiple devices ("multiple access"...)
  - – No one is in charge. Think 'party atmosphere'
- Two common models
  - – [redacted] – try your luck
  - – [redacted] – stricter rules Or Channel access

Randomised access

- [redacted] (1960s!)
  1. If you have data to send, send it
  2. If the other end doesn't ACKnowledge it, or you hear another device transmitting while sending: **We've had a COLLISION!**
  3. If Collision, wait a random time(back-off),and try again
- Very simple. Very effective. If the [redacted] isn't too high
  - – Statistical performance up to 18%-36%.
  - – Performance depends on the back-off scheme, and better designs exist

CSMA

- [redacted]
  - – Good for [redacted] networks, needs more work with wireless
- Like ALOHA, just check if somebody is sending first (sense for carrier) – As soon as line is clear, send the whole frame
- Much better – no collisions?
  - – Delay on [redacted] cables. Two senders can [redacted] – As [redacted] gets bigger, problem gets bigger
  - – Sets [redacted], and [redacted]
    - Need enough time to detect a collision,
    - Whole frame could be out, you start next one...
    - Minimum frame time is [redacted]

CSMA/C*

- [redacted] (CSMA/CA)
  - – Listen for carrier – Once it's clear, [redacted]
    - Avoid all the waiting terminals [redacted]
  - – Then send the whole frame

- ____ (CSMA/CD)
  - – Listen for carrier
  - – Send, and ____ while sending.
  - – If yes, stop sending immediately, and “jam”... (‘hey, everyone back off’) – Then ____ before retrying

Backing off...

- Need some limits on how long to back-off – Can’t be too short – Can’t be too long
- Ideally back-off depending on the ____ – Send with ____ probability. How do you know N?
- ____ (BEB)
  - Counting the number of collisions you experience
  - <u>First</u> time, wait 0-1 frames. <u>Second</u> time wait 0-3 frames. <u>Third</u> time wait 0-7 frames.

Wireless is harder

____ – i.e. may be no single carrier to sense

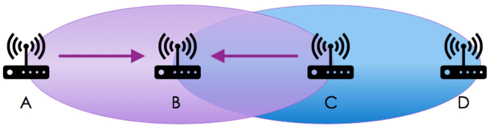
– Can be part of a single network but can’t see all the nodes (____)

In wireless, ____ – i.e. can’t listen for collisions

– Can’t respond quickly to collisions, lots of wasted time

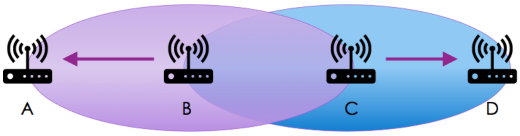
____ (stations)

A and C are “____” from each other: Both can talk to B – and collide



____

B and C are ____ to each other:
Both could talk to an outer neighbour – and NOT interfere Want to avoid ____



MACA

- ____

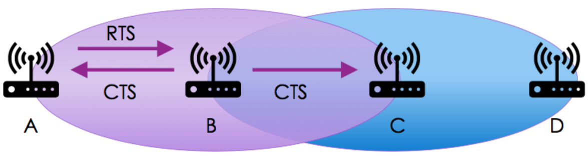
A quick handshake before yelling:

- **Sender**: ____(RTS) + N bytes
- **Receiver**: ____(CTS) + N bytes
- Sender transmits

– and any nodes that heard the ____ N bytes – and any nodes that heard ____ stay silent for the CTS

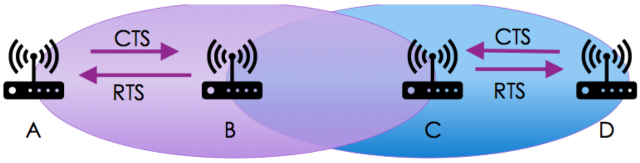
Fixing hidden terminal problem

Node C knows something is going on, and waits N bytes



Fixing exposed terminal problem

B can’t hear D/C-CTS, C can’t hear A/B-CTS – All Clear!



Contention-free access

- Take turns! • Identify an order of the devices on a network
- E.g. : uses a special frame, passed around – You can't send if you haven't got the token – Which can make it fragile to errors, or engineer around it

Broadcasting

- LAN broadcasts
- I have an announcement – That all should know & That somebody should know, but I don't know who
  - I'm asking a question
  - Implemented through a special ' ' address (all-1's)

Topologies

- Most wired LANs have moved away from bus topologies
  - – Eventually doesn't scale
  - – Make a bus (cable) longer:
    - Needs a
    - Or , which links two LANs, and learns on each side
- Today nearly all are ' '
  - – Crossbar devices that learn addresses from the
  - – Makes every link (computer to)
  - – Greater & Greater

-LAN-

Acronym overload

- General "LANs": ?
  - – Cheap, mass-produced, reasonably well-behaved, scalable – and simple!
- Carrier-grade: SONET/SDH, ATM, FrameRelay, GPON, ... – Expensive, robust, service-level guarantees
- Data-centre: FibreChannel, Infiniband, FDDI, ... – High-speed, low-latency, specific-purpose
- Wireless: RF, LiFi, whitespace, Zigbee, Z-wave, HaLow, 6LoWPAN, ...
  - – Regulated in some frequencies, free-for-all in others
  - – Device-oriented: Low power (long battery life), long range, low datarates

Standards

- IEEE: community standards, active research, publications
  - – Different to ISO, ITU, IEC – government-recognised standards bodies
  - – Physical engineering,
- Also naming, best practices, software/hardware architecture, ...
- IEEE 802: standards and committee
  - – LAN/MAN networks carrying variable-sized frames (not cells)
- 802.1: . • 802.3: • 802.11:
- 802.15 • 802.16:

– Note 802.15.2: WG on "bluetooth and wifi co-existence" – is 'hibernating'.

802.3 Ethernet

- 1983 – coax cables, vampire taps, 10Mb/s
- It's evolved a lot since then: faster, further, more robust, more functionality
- Lots of individual standards, sometimes superseding or merging earlier versions.
- Lots of backwards compatibility
- 802.3a - 802.3z (1985-1998) / 802.3aa - az (1998-2010)
- 802.3ba - bz (2010-2016) / 802.3ca - cs (2016-today)

Which Ethernet?

- **10Base2** vs **1000Base-LX** ??
- Naming:
  - – Speeds: 10, 100, 1000 (Mb/s), 2.5G, 10G, 25G, 40G, 50G, 100G, 200G, 400G
  - – Signal: BASE, PASS, BROAD
  - – -? = media. T=Twisted Pair, S=short 850nm, L=long 1300nm, E=extra-long 1550nm
    - C (or blank) =coax. Mostly. F=fibre. B=bidirectional (single fibre).
  - – Last **letter** = encoding (8b/10b, ...), or ignored
  - – Last **number** = channel count (wavelengths, copper pairs).
    - Or reach (2,5,36 * 100m, or 10,20,30*1km). Or …

1000BASE-LX:

– [redacted]

– 1310nm over MMF (500m) or SMF (10km)

– 8b/10b NRZ encoding

- **1000BASE-T**

– [redacted]

– [redacted] with 8P8C (RJ45) connectors, Cat5e or better

– [redacted] pairs, in [redacted]-directions simultaneously

– 4b/5b, PAM-5 encoding

Ethernet

- Started over shared medium coax – CSMA-CD, Manchester
  - – 10Base2, 10Base
  - – Moved to [redacted] 10BaseT, using 1-4 pairs
  - – A plethora of encodings, differential signalling, and ultimately fibre
  - – Moved from shared media to fully-switched
- Half-duplex (bus) to full-duplex (SDM) to full-duplex (FDM)
- If not all 4 copper pairs in use, can run power, telephone over the others
  - – And with FDM – power over data wires. 802.3af, at, bt, and bu (for cars)
- Very good [redacted] – Well designed to cope with network changes
- Very good [redacted] – Link negotiation on connection

Auto-negotiation

- When plugging in an Ether net device to a switch, need to agree:
  - – Speed
  - – Duplex
  - – Cross-over(which wire does what)
- Need to detect a plug-in/disconnect.
- **Heartbeat**=NormalLinkPulses(NLP)
- **Capability**=FastLinkPulses(FLP) – Encodes messages in16 bit words
- Allows both ends to negotiate and agree – (if they're allowed to!)

An Ethernet frame

1 byte = 8bits

- Every device that can listen will receive the frame
- If the frame destination is not yours, drop it, otherwise inform Operating System

- UNLESS you are in ‘ mode’, and listening to everything

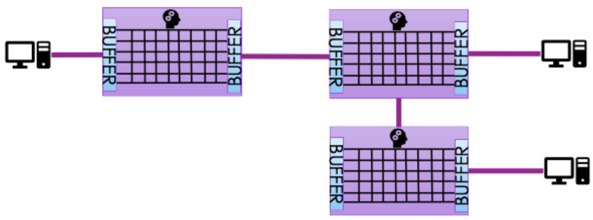
Addressing
- (802) MAC addresses
- Globally unique address (EUI-48)
- Various allocations of 48 bits to a NIC
- Written in hex: *38:10:d5:bc:be:99*
- ‘All ones’ ff:ff:ff:ff:ff:ff = broadcast frame • *Some addresses are special messages*

Ethernet hubs
Shared media, CSMA and collisions – through a hub/repeater

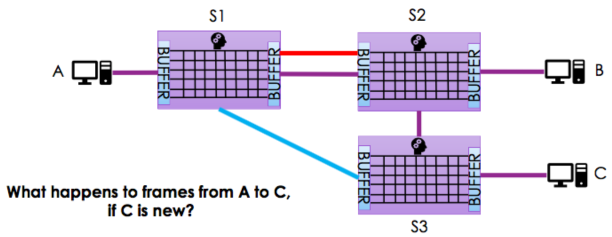
Ethernet Switching
More scalable, more reliable

What goes where?
- Need to know which MAC address(es) is(are) on which port – Without being manually told
- Switches learn on the fly – Using source addresses, most trustworthy.
- **First** – listen to what’s coming in, and record the source MAC address
- **Second** – if it’s a new MAC destination:
  - – Send it to all ports(*) (unicast port flooding)
  - – Hope somebody replies, and then record their port.
- Broadcasting is now the switch’s responsibility – ***not*** the cable.

Hierarchy of switches
- This works well for any loop-free topology :And now a ‘broadcast’ domain can be defined as wide as you want



What about if there are loops?
- Redundant links. Parallel links. Short cuts. Made a mistake. Evil intent.



Spanning Tree
- Broadcast storms, MAC table updates, duplicate frames – BAD!
- Develop an overlay view that **spans** the network with a **loop-free** **tree** – Effectively: disable some links (“block ports”)

– Switches need to work this out themselves, and adapt, in real-time
– And then forward frames accordingly

panning Tree (Protocol) rules – 802.1d, w, ...
- Before doing anything else - or on any change – or a timer - block all but STP traffic
- Elect a root node (lowest address wins), and at the same time
- Grow the shortest tree, using distance (hop count) and value (speed) from root
  - – Tie-breaker: lowest address
  - – Record the ports that are on the tree towards the root
- Initially everyone thinks they are the root – and tells their neighbours so.
  - – Some switches get disappointed quickly
  - – They tell their neighbours
  - – Everyone updates
- Once converged: turn off ports (paths) that aren’t on the tree – But remember they are there, if/ when something changes

Casting

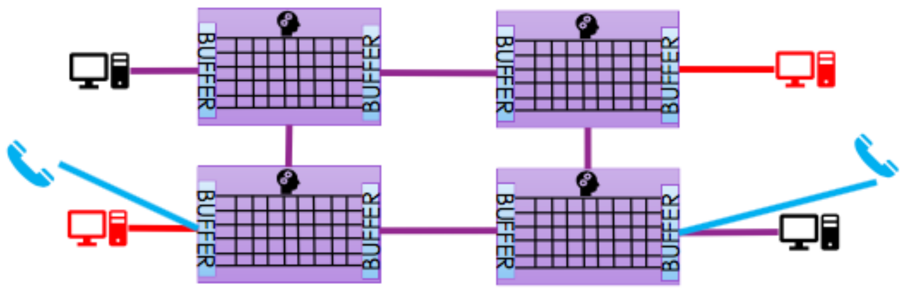
- **Broad-cast**: Everyone gets it. MAC destination = all 1's (ff:ff:ff:ff:ff:ff)
- **Uni-cast**: Only the intended recipient (should) get it. **MAC** destination
- **Multi-cast**: Everyone who is interested gets it
  - – Special bit-flag in MAC address
  - – Devices can 'subscribe' their NIC

Special features

- Link aggregation/ '
  - – When a single 1Gb/s or 10Gb/s won't do – Performance – Failover
- 802.1AX (was 802.3ad) – up to 8 (identical) links
- (LACP)
  - – Send a 'do you do LACP?' every second
  - – If yes, identify other common links and aggregate
- Various modes: round-robin, active-backup, random-alloc, …
  - – _Must not_ cause mis-ordered or partial frames, nor duplicates

Virtual LANs (VLANs)

- 802.1Q – add a 4-byte "tag" to the Ethernet frame
- Now have 2+ 'broadcast domains' on the same network
  - **Separation** of traffic
  - **Prioritisation** of traffic (was 802.1p)



- Standard maximum frame:
  - 1500 bytes data, 26-30(+12) bytes 'overhead' – at **best**
  - At 10Gb/s, one maximum frame every 1.5µs
  - Buffers overflow, congestion, drops
- What happens if we make the payload bigger?
  - **Jumbo Frames** – 9000 byte payload
  - Lower overhead, lower cpu load – but need a different checksum

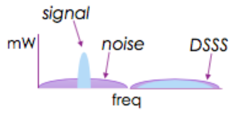
WiFi! (aka WLAN)

- WiFi is not wireless Ethernet
  - – But has inherited a lot from it
  - – Access points like 802.3 repeaters
- Much more challenging communications environment, clients
  - – More work to be robust and perform well
  - – Rate and power adaptation
- Based on CSMA/CA – with optional RTS/CTS (MACA) – Single access point for many clients...
- Along with OFDM, DSSS and MIMO

Acronym soup

- OFDM – Orthogonal Frequency Division Multiplexing

• MIMO – Multiple input, multiple output
– Multiple antennas, beamforming (RX and TX)
– Multiple paths, deconstructed interference, voodoo magic



• DSSS – Direct Sequence Spread Spectrum
  – Related to Frequency Hopping Spread Spectrum
  – Uses codes across a frequency band (CDMA)
  – Encoded 1b/10b - or even 1b/10,000b

Standards – IEEE 802.11

802.11	Hz	Bandwidth MHz	Datarate Mb/s	Range (m)	
a	5G	20	6-54	30-150(*)	
b	2.4G	22	1-11	30-150	
g	2.4G	20	6-54	30-150	
n	2.4G/5G	20-40	<600	70-250	
ac, ax	5G (2.4G)	20-160	<3500	30	
ad, ay	60G	2 or 160	<6600-10,000	3, 10	*WiGig*
af, ah	0.05-0.9	1-16	30-300	100-1000's	*TV, HaLow*

Cantennas • Assume need area-coverage • more directional and longer range

Channels @2.4GHz
• TV channels – tune to central frequency – Not all channels in all countries
• (most)Channels overlap
  • Channels taper at edges
  • Different802.11interoperate–by stopping!

802.11 WiFi Channels @5GHz
DFS Channels< Weather Radar

802.11 Frame-types
• All those addresses... src, dest, AP and 'other'
• Frame Control:
  – Control Frames: Control the communication with the Access Point
  – Management Frames: Manage the relationship with the Access Point
  – Data Frames: Send data...

802.11 Control Frames
• Request To Send (RTS) • Clear To Send (CTS). • Acknowledge (ACK)
• Request for RTS (RRTS) • Data Sending (DS)

Reliability
• LANs should be simple
  – LANs should not do overly-smart things
• But: LANs should perform efficiently, effectively
• Who takes care of errors?
  – Defined as 'failure to get through correctly' – for multiple reasons
• Three approaches:
  – Detect errors and drop frames (something else will take care of it – 802.3)
  – Detect errors and fix frames at receiver (forward error correction)
  – Detect errors and sender sends again (Automated Repeat reQuest – ARQ – 802.11)

ARQ by ACK
• For every frame I send, receiver should ACKnowledge receipt
  – As long as it arrives correct!
  – If they don't ACK, within a timeout, send it again
• What happens if the ACK is lost?
  – Send again, but flag it's a resent frame
• What happens if timeout is too short? – Send again, but flag it's a resent frame
• *Stop-and-wait* ARQ
  – Helps with high delays
  – Single-bit sequence number (alternate 0,1)
  – ACK includes that sequence number

- Robust, but **throttles** performance as (bandwidth*delay) goes up

Client by association

• Need to know what's available:
  1. SSID (service set identifier) – a wireless (V)LAN
  2. Access Points that accept connections for that SSID

• So either – Listen for AP's offering services, or – Call out for AP's offering services
• Identify who you are (authentication)
• Associate with an AP (resource allocation)
• And then keep it running, while everything changes...

802.11 Management Frames

• **Beacon:** (broadcast) I'm an AP and can offer these SSIDs at these rates in these frequencies with these standards and ...
• **Probe Request (**(broadcast) I'm a client, what can you offer? **)**
  - Probe Response
  - Can also **Probe request** 'do you offer SSID X?'
• **Authentication** (open or shared-key)
  – Deauthentication ((targeted): I can offer these SSIDs)
  - Can also have username/pw, MAC filters,WirelessProtectedSetup(WPS),…
• **Association Request** and **Association Response**
  – Disassociation
  – Reassociation Request and Reassocation Response

Pass-through authentication

• **802.1X** – Uses Extensible Authentication Protocol (EAP) – Can be used on 802.11, 802.3, etc. – Typically RADIUS back-end • Keep PW's off AP's

Encryption – everything is sniffable

• WiredEquivalentPrivacyWEP
  – Don't go there. Single key, easily calculated from traffic sniffing.
• WiFiProtectedAccessWPA
  – WithPre-shared-key(PSK)="personal"or802.1X="enterprise",
  – TemporalKeyIntegrityProtocol(TKIP)–per-frame-key
  – Better integrity checks than simple CRC
  – Heaps better. Still broken, largely throughWPS("easy-to-join"feature)
• WPA2
  – Lots of additional measures. Much stronger encryption and other protections.
KRACKed(2017)
• WPA3 – Warm off the press (Jan 2018)
• Question around what in a frame is encrypted/protected, and when–some info leaks.
  – My neighbour is talking to an interesting site?

——————————————————Ethernet & Wifi——————————————————

Wide-Area Networking

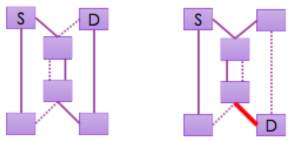
Scaling

• Address tables and management don't scale globally
  – Billions of devices – and every switch needs to store them all?
• Updates take a long time to propagate
  – Long delays, depending on the paths
  – Topology changes happen a lot
• Broadcasts have to be sent globally
  – E.g. whenever a new device connects
  – Spanning Tree won't converge in time

Traffic control

• LANs organise themselves for simplicity – not optimisation
• Spanning Tree doesn't guarantee the optimal topology

– Sometimes people do know better
• Network traffic costs money, needs to consider politics



Which LAN?

• Many LAN choices
– 802.3 Ethernet, 802.11 WiFi, xDSL, 4G, ... each fit-for-purpose (wired/wireless)
• *<u>Different</u>* ***LANs*** don't mix. Mismatched behaviours with different – Address schemes
– Service models (frames, cells, circuits)
– Security models
– Frame sizes
– Performance
– Prioritisation mechanisms
• Don't want to write applications tuned to different LAN types
• Don't want to buy boxes that translate between every possible combination – (Many, expensive) single points of failure

Solving these

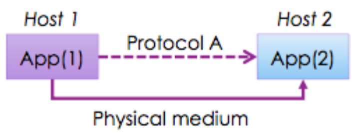
• Want to communicate across networks: aka Inter-network • Take the **LAN** to the **WAN**
• *<u>Scaling problems</u>* – Use a hierarchy of connections, addresses and aggregate/group
• *<u>Traffic control</u>* – Optimise routing with more information, and support prioritisation
• *<u>Which LAN?</u>* – All of them. Put a common **layer** across the top
• *<u>Just one layer?</u>*

Why stop at networks?

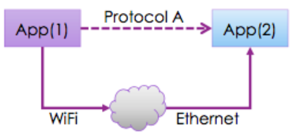
• Applications need functionality too
– Find/advertise resources
– Connect to other machines • One or many
– Exchange application-specific messages
– Adapt to capabilities : Devices, software, ...
– Maintain state of a connection
– Expect sufficient reliability
– Expect trustworthiness
– Maximise performance, minimise delays
• Simplify: let's modularise/layer things...

Layers upon layers

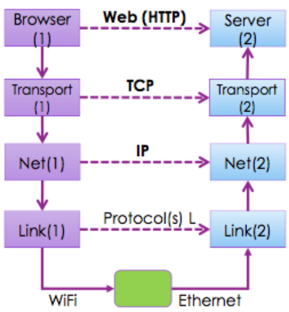
• Use **layers** to divide (allocate) functionality
• Use **protocols** to exchange information within a layer • User **services** from lower layers to build on



Copper medium and Ethernet protocol



Protocol is not Ethernet nor Wifi?
Offload more: reliability, app-muxing, security, performance, ...



Applications offload the networking details
Network layer provides a service, takes care of **everything?**
<u>Network layer:</u> transmission, topology**, ...**
Each protocol exchanges Protocol Data Units (PDUs) Each layer provides services through an API, exchanging Service Data Units (SDUs)

An OSI *(ISO+ITU-T/CCITT)* reference – great standard – almost never used

Layer	Name	Function
7	Application	Deliver functionality
6	Presentation	Convert information for application needs
5	Session	Combine diverse communications, maintain state
4	Transport	Ensure end-to-end performance
3	Network	Send packets over multiple links
2	Link	Transmit Frames
1	Physical	Modulation and encoding of bits

Internet "Reference" protocol Stack

Layer	Name	Function
7	Application	Deliver functionality (and the presentation/session)
4	Transport	Ensure end-to-end performance
3	Network	Send packets over multiple links
1,2	Physical, Link	Transmit Frames

Layer	What it transports (Protocol Data Unit)	How they connect
Application	Messages/Data	Proxy, gateway
Transport	Segments/Datagrams	
Network	Packets (!!)	Router
Link	Frames (cells, circuits)	Switch, Bridge
Physical	Bits	Hub (repeater)

In reality

• There are protocols that span layers

– For internet working, some knowledge has to move up and down layers

• There are layers that have sublayers

– E.g. Ethernet has MAC and LLC

• Logical Link Control; adds **payload-muxing**, flow-control and extra error-handling

• There are people who think about this classification too much – It's not a rule

• But as a concept, and to guide protocol design – it's very useful

– Think about what functionality you need/offer, and where it should be

– Success: Internet end-to-end principle – smart edges, dumb core (rules, not state)

• Every protocol has a dictionary for what it sends – Every layer has a 'payload' to transmit

• Every payload is passed to a lower layer and is encapsulated – Think of a letter in an envelope

– Add headers (and trailers), maybe encrypt, compress, segment

– Repeat till it's sent

• Ideally don't hold up the traffic

– Minimal packet inspection

– Only read the layer (headers) they are responsible for forwarding • Link, network

• Sometimes we need more, e.g. security, priority

– *Deep Packet Inspection*, in real-time

• Right down to the inner payload • Significant load, delays

"Demuxing" the "encapsulated"

• Host/Receiver ultimately gets the (whole) message • Need to pass it to the process that needs it – Which one???

• Every layer has a 'key' about its payload (in its header)

– **Ethernet** has an address, and Ether-type

– **IP** has an address and Transport-type (tcp vs udp)

– **TCP** has a port number

– **Applications** have message keys (e.g. http url's)

Security, Priority

• Having DPI is considered harmful. • Lack of DPI can be considered negligent

• Good design works out how much to unpack, and when

• Firewalls and DMZs and VLANs and VPNs and ...

Circuit
- POTS/PSTN
- Set up (and tear-down)
- Guaranteed channel

– But inefficient

– Solid block-out for duration of a conversation

After Circuit switched...
• **MESSAGE switched:**

– Postal Service:

  • Put message in container,

  • Address it

  • Put in the network

– Network (hopefully) takes care of delivery

– "Store-and-forward" – hold entire message

  • Examine container at each hop before forwarding.

– Message loss is a large problem.

  • Less than a circuit. Failure along the path is flagged from that point

– Potentially High Latency

  • Each hop takes time, especially for large messages

• **PACKET switched**

– Put fragments of message in multiple containers

– Address them all, and put them on the network

– Network (hopefully) takes care of delivery

  • Examine each container at each hop before forwarding. [Acknowledgement happen sooner, more frequently]

  • Packet loss is more tolerable[Recovery is a smaller effort]

– Still high latency

  • Each hop takes time, plus overheads

  • But less than for large messages[Not waiting for each hop]

– And much better sharing of capacity

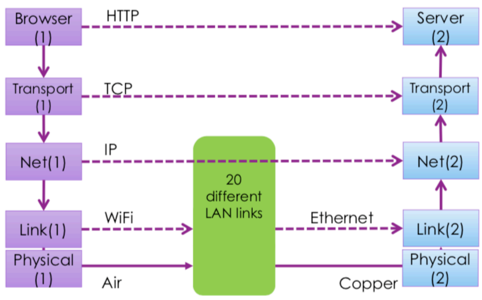
Role of the Network layer
- Each consumes services (functionality) from the layer below
- Each offers services (functionality) to layer above

Role of the (IP) Network layer
• Simplest common functions – Across many/all link types – Just a glue layer

1. End-to-end delivery of packets
2. Global addressing
3. Cope with evolving network topology

• Consume little from lower layer • Offer little to higher layers

Hiding **path** complexity



Applications don't know nor care. Unless there is a performance question.

Guiding principles
• Simplicity and end-to-end design

  – Keep connection/conversation state at the edge, not in the network

• Provide '**best-effort**' delivery

  – Minimal Service Level Agreement (SLA)

• Ensure '**reliability**' is delivered (only) where it is needed

  – For specific application needs: file transfers, audio calls, ...

  – Can be done at different layers

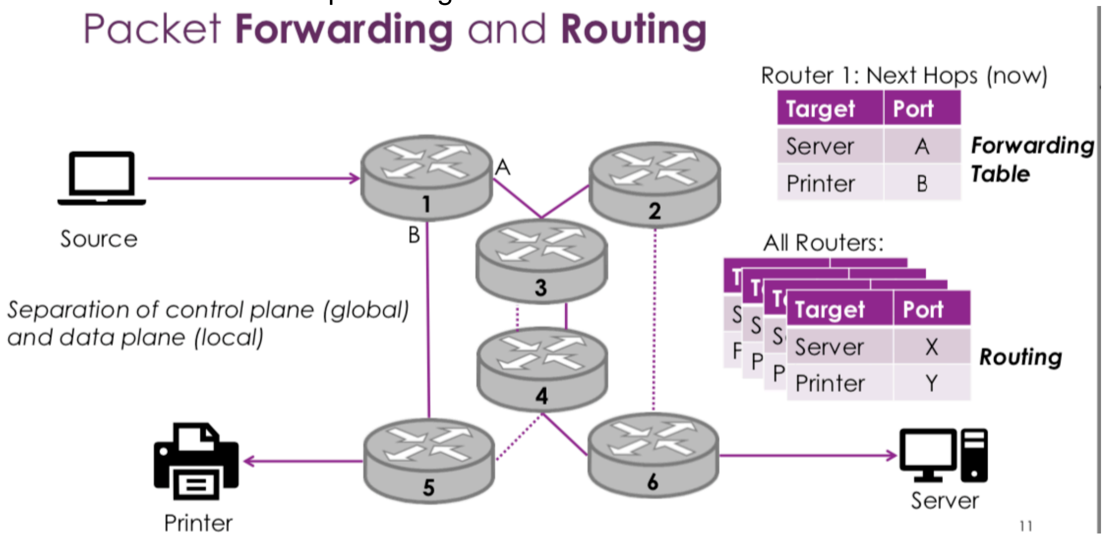
    • Link (vlan, ...), Transport (tcp, ...),

    • Network layer? (mpls, ...)

Connectionless vs Connection-oriented
• What guarantees does your application need? • Which layer provides it?

• Connectionless, packets – Go where the network chooses, in realtime

• Connection-oriented, circuits – In a packet-switched world – ‘virtual’ circuits.
– Go where I configured the network to send them
• Need to understand how packets get sent towards their destination



Circuit ID	Payload

Circuit	Port
X	A
Y	B

Circuit Forwarding Table

Source	Server	Payload

Target	Port
Server	A
Printer	B

Packet Forwarding Table

Router 1 Forwarding table

Target	Next Hop
150.99.10.0/24	Router 5
150.203.0.0/16	**Router 4**

Multi-path packet forwarding

- Statistical multiplexing
- Unpredictable ordering
- Variable delays
- No guarantees
- **Receiver’s** problem to deal with order, loss, jitter

Circuits over packets

• **Why?**– Guaranteed path – Guaranteed (maybe) bandwidth/performance
• **How?** – Circuit set-up and tear-down: manual, or on-demand
– Packets: More encapsulation – IPinIP, Multi-Protocol Label Switching (MPLS)

	Packets	**Circuits**
Path router control	Not needed	Required
Prior Setup	Nothing needed	Required
Router State	Per destination	Per circuit configuration
Addressing	Packets carry full src/dest	Packets carry short label
Forwarding	Per packet	Per circuit
Router failure	Packets lost, reroute	Circuit fails completely
Quality of service	Difficult	Easy(*)
Security	Per-packet, other layers	Maybe...

(The) Internet design

- Standards: the **Internet Engineering Task Force** (IETF.org)
  - – Just a bunch of people, arguing. Not a company, no board, no members
  - – Lots of Working Groups, under Areas
  - – Work revolves around 'drafts' and 'request for comments' (RFC)
- **RFC**
  - – Strict rules about structure, references, and language (MUST/SHOULD/MAY)
  - – Standards Track or
    - • Best Current Practice, Informational, Experimental, Historic (Lost interest or Detrimental)
  - – Locked down on publication. Regularly Obsoleted or Updated
  - – RFC-0001: April 1969, RFC-8571: March 2019 • Watch out for 1 April RFCs...

Taming the crowds

- IETF needs some structure:
  - – ISOC: Internet Society – international, non-profit, legal entity
  - – IESG: Internet Engineering Steering Group – oversees IETF processes, signs off.
  - – IAB: Internet Architecture Board – Big picture view, identify/review issues
  - – IRTF: Internet Research Task Force – Researches issues... • Overseen by IRSG: Internet Research Steering Group
  - – IANA: Internet Assigned Number Authority – Directory keeper
    - • Contracted to ICANN: Internet Corporation for Assigned Names and Numbers
  - – RFC Editor

IETF "principles"

- End-to-end...
- "Rough consensus and running code"
- "Be conservative in what you send, liberal in what you accept"
- Simplicity, clarity,
  - – Fight feature creep, use other layers, offer just one way to do something
  - – Don't hardwire too much, let it be negotiated
  - – Aim for good, broad design; let others deal with edge-cases
- Think about scalability, non-linearity, heterogeneity, cost – Law of Unintended Consequences

IP addressing

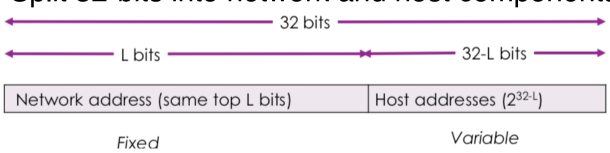
- 32-bits = 2^32 hosts = ~4billion - in theory • Written in 'dotted-quad' notation

– i.e. four numbers, separated by dots

- 11010101|11110000|10101010|00001111
- 213.240.170.15
- Not a host, but an interface – 1 *IP* = 1 MAC most of the time…

IP prefixes

- Aggregate 'nearby' addresses into a *block* for routing (tables)
- A block of addresses is described by its *prefix*
- Split 32-bits into network and host components using upper **L** bits:



- Use a '/' ('slash') notation:
- Network address/prefixlength:
  - – Network address is the first 'host' in the host-range

• For example: 150.203.0.0/16

– From 150.203.0.0 up to 150.203.255.255

– 32 bits, using 16 for the prefix: 232-16 = 216 = 65,536 addresses

- A “/24” has 256 addresses [e.g. 150.203.0.0/24 = 150.203.0.0-150.203.0.255]
- A “/30” has 4 addresses

IP subnets

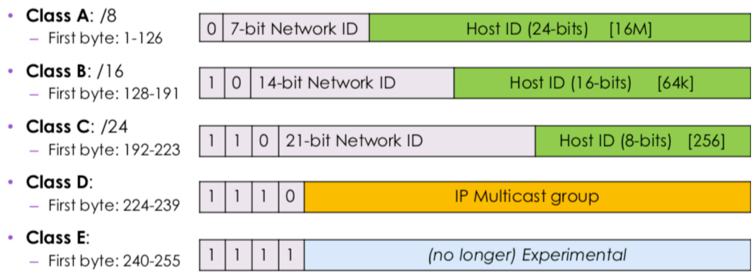
• Network address/prefixlength •Prefix length = a subnet mask

- Network address = a subnet (a block of contiguous host addresses)• For example: 150.203.10.0/24

– The 150.203.10.0 subnet

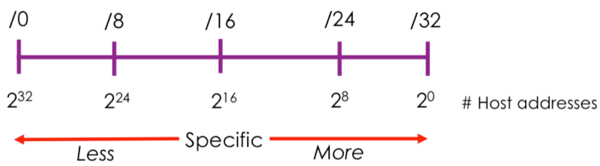
– /24 = 24-bit network id so mask= 24 1’s and 8 0’s: 255.255.255.0

IP address classes



More or Less?

- A “More-specific” prefix = longer prefix = fewer hosts
- A “Less-specific” prefix = shorter prefix = more hosts



Forwarding by longest matching prefix

- Prefixes in a forwarding table are allowed to overlap

– For good reasons!

– Aggregation benefit of hierarchical addressing (e.g. a /20 holds sixteen /24’s)

– As well as flexibility to direct some specific traffic

• Longest matching prefix **rule**:

1. For each packet, identify all subnet prefixes that apply
2. Select the one with the longest matching prefix • The ‘most specific’
3. Forward accordingly to the next hop

Why?

- Provide default behaviour with shorter (less-specific) prefixes

– Catches more host-addresses in a single block

• Support specialised behaviour with longer (more-specific) prefixes

– Key services to be reached via • Higher performance paths

• Lower cost paths • More secure paths • ... (policy reasons)

- Hierarchy generates more compact forwarding tables on routers – Cost of lookups vs simple tables is largely optimised away now

Hosts as routers?

• How does a host decide how to send a packet?

– Assume it has learnt the destination IP address from somewhere

• Hosts are not good routers – keep it simple, let routers route!
• Two types of destination
  – On my LAN – **use LAN services**
  – Beyond the LAN – **use a router**

Host forwarding table
• How to decide?
• Longest matching prefix plus a catch-all address: • 0.0.0.0/0 = 'the whole internet'
• Host knows its IP address and its prefix (subnet mask): – "I'm 150.203.56.99 and I'm on a /24"
– So my network is 150.203.56.0/24

Target	Next Hop
150.203.56.0/24	Direct on my LAN
0.0.0.0/0	My (default) Router *150.203.56.1*

Longest matching prefix
...which is also on my LAN

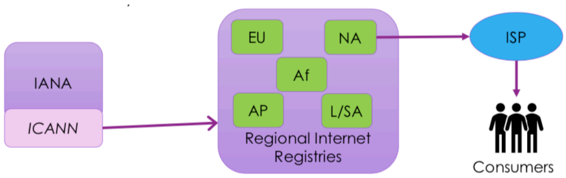
Home on the LAN

Source MAC	**Dest MAC?**	Source IP	Dest IP	Payload

• Network-Layer:
  – Hey, I'm on the same Ethernet as my target, I can just send this packet directly
  – Link layer, send this to IP-address 150.203.56.99
• Link-Layer:
  – What's an IP address????– I need a MAC (Link Layer) address – Network-layer won't help me
• Need to cross layers. Need some kind of Address Resolution Protocol

The Address Resolution Protocol (ARP)
• RFC 826 (and updates)
• Mapping IP addresses to Ethernet/etc. hardware addresses – Not an IP packet – Link Layer
• A wants to send IP packet to C – Send a Link layer broadcast
  • Src MAC = AA:AA:AA:AA:AA:AA
  • Dest MAC = FF:FF:FF:FF:FF:FF
  • I am/Tell 150.203.56.88
  • Who has 150.203.56.99?
• Some optimisations:
  – Caching, with timeouts
  – Catch passing ARP information [B doesn't ignore the broadcast]
  – Tell everyone of your changes
• Gratuitous ARP – "look for yourself" • Also helps find duplicate IPs
• Also applies to packets going to/through the router – Need MAC address of R
ARP is a simple <u>Discovery Protocol</u>

Getting addresses (blocks) for your network
• Need to consider
– Globally-unique allocation
– Routing aggregation opportunities – Politics
• Need an authority, which scales



Addresses are not equal
• IPv4 – 2^32 addresses – can't use all of them.
• Special allocations
  – **Private Networks:** 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
    • Can be used on networks (that are/are NOT) connected directly to the Internet
  – **IP Multicasting:** 224.0.0.0/4 [old class-D] • Distribute packets to groups of subscribers
    • Requires additional services on the network
  – **Experimental**: 240.0.0.0/4 [old class-E, 200M addresses!]
    • Still waiting for an experiment• Most OS will drop such packets

- Special networks:
  - – **This host** on this network: 0.0.0.0/8
    - Only used as a source address • Used for ‘any interface’ or ‘I do not know’
  - – **Local interface**: 127.0.0.0/8
    - Loopback interface: 127.0.0.1
  - – **Link-local**: 169.254.0.0/16 • My LAN when all else fails
  - – **Broadcast**: 255.255.255.255 • Specific address for a global broadcast (In theory…)

Address conventions

- Subnet broadcast: A.B.C.255
  - – All ones in the host field (.255 for /24, .255.255 for /16)
- The subnet: A.B.C.0
  - – Aka “the wire”, usually followed by /n – All zeroes in the host field
- Router/gateway: A.B.C.1
  - – A convention. Makes it easy to find
- Note: Host field is shorter than 8 bits, for prefixes >24
  - – 150.203.56.0/28: *(Sub)Netmask=255.255.255.240, Broadcast=150.203.56.15*

Getting feedback from the Internet

- Sometimes things happen to packets
  - – Along the route
  - – Loss, corruption, mistakes, ...
- Wrong addresses, nobody home, packet malformed, ...
- Sender needs to know what happened
  - – With little/no feedback from receiver
  - – Retransmission may be wrong• Don’t keep making the same mistake: “Internet says NO!”
- Internet Protocol needs some Control Messages

Internet Control Message Protocol (ICMP)

- IP packet – protocol #1
- Designed mainly for routers to inform senders (including routers)
  - – Senders listen, but don’t (usually) send
- ‘Type’: category. ‘Code’: actual problem/question/answer
- Data=
  - – Header of packet that caused the problem, and 8+ bytes of payload
  - – Other information related to the problem/request

0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Type	Code	Checksum	
Data			

ICMP Types

- Type 0,8 – ICMP Echo
- Type 3 – Destination Unreachable – Many reasons, at intermediate routers, final router, host
- Type 4 – Source Quench – Please Slow Down! (deprecated)
- Type 5 – Redirect – Looks elsewhere
- Type 9,10 – Router discovery
- Type 11 – Time exceeded E.g. TTL has hit zero
- Type 12 – Bad header
- Type 13,14 – Timestamp
- Type 15+ - Deprecated, Experimental, Unallocated and Reserved

ICMP use by hosts?

- **Ping**
- Send an ICMP Echo-request (Type 8/Code 0) to an IP address
- If received, receiver sends back an ICMP Echo-reply (Type 0/0)
- Useful for testing (many options) – and for probing...

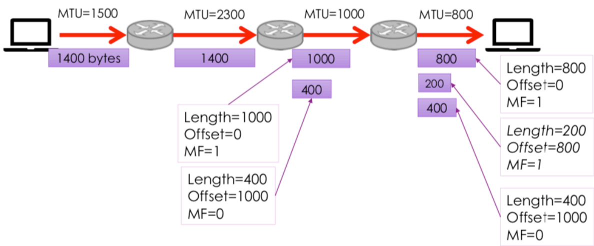
Traceroute

- Identify all the routers through which your packets are going (now)
- Use ‘TTL decrement’ and ‘ICMP Time Exceeded’ (Type 11/0)
  - Replies include IP of router that hit zero
- Really useful to identify path, intermediate devices and distances
- And to probe internal networks...

3 attempts each hop, RTT increases in jumps (within variations)

Big packets

• Bigger packets are more efficient, but can be ‘too big’.
• What’s a ‘big’ packet?
• Something bigger than the payload of your LAN
  – **Maximum Transmission Unit** (MTU)
  – Ethernet: 1500bytes, WiFi: 2300bytes
  – Leads to **Fragmentation**



Router Fragmentation Process

- Incoming packet of size > outbound MTU
- Split packet into (large) new packets
- Copy IP Header to each new packet – including the Identification
- Adjust Length field for each packet – And Checksum, and TTL
- Set Offset to identify location within overall packet
- Set MF flag on all packets, except the last one
- Receiver collects all fragment-packets and reassembles

It works, but...

• Has been used since the beginning of IP, and works well
• Creates performance issues
  – More work for routers and receivers
  – Increased probability of (total) packet loss
• No retransmission of fragments – Security issues
• Easier to hide malicious traffic
• Harder for Deep Packet Inspection

Better approach

- Test the network and send the smallest big-packet you can • **Path MTU Discovery**
- Looks like traceroute – but use packet sizes and DF=1

ICMP Destination unreachable (Type 3) Fragmentation required, and DF flag set (Code 4) Data = next-hop MTU

IPv6!

- When IPv4 just won’t do it anymore
- IPv4 designed in a smaller, more scalable and way more trusting world
- Never considered planetary wide participation, major infrastructure role, IoT, smart devices, mobility,... bad people
- We now have a problem – Several – But especially the need for more than 4 billion devices

• New effort from ~1994
  – Address exhaustion was long predicted – Note around the rise of WWW
• Standardised around 1998, OS support from 2000
• And till recently, limited effort
  – 1983.1.1 Internet flag day: Comply or disappear. Can’t do that now!
  – Hampered by deployment issues
  – Lacking incentives • Nobody does homework till there’s a deadline
• What do we get? – Bigger addresses – And other stuff...

IPv6 addressing

- 128-bits = 2^96 more than IPv4
  - – 6x10^23 per square meter on Earth – A few thousand for every atom on the surface
- 'Coloned hex-quad (with compression)' – Instead of 'dotted quad'
- 8 groups of four hexadecimals (8*16bits)
- For **visuals**, compress
  1. Drop leading zeroes
  2. Drop consecutive zero blocks

1. 3018:0ae8:0000:0000:0000:ae00:0098:8ac2
2. 3018:ae8:0000:0000:0000:ae00:98:8ac2
3. 3018:ae8::ae00:98:8ac2

IPv6 address types and scopes

- **Types**:
  - – Unicast – to one
  - – Multicast – to a group
  - – Anycast – to the nearest in a group • Note – no broadcast!
- **Scopes**: (*except multicast*)
  - – Link-local – my subnet
  - – Site-local – my organisation/site
  - – Global - everywhere

IPv4 is over

- A common catchphrase
- However... Addresses are largely exhausted
  - – RIR's ran out 2011-2015
  - – Lots of wasted address space
  - – Re-allocating ever-smaller chunks (/30) • With tighter rules
  - – Can't aggregate address blocks for routing • Forwarding tables are immense

The routing problem

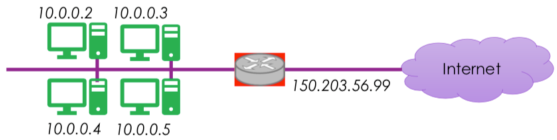
- Every router has a forwarding table – Fortunately only 10-100 interfaces
- Across the whole internet
  - – Lookup tables of ~1M entries– Fuzzy matching on 32bit address – In under 5ns (100Gb/s)
- And 170k updates/day – 2 per second
- Address blocks are getting much harder to aggregate

Moving to IPv6

- Will probably have both for another decade or more – around 10-20% now
- Transitioning is a large problem...
  - – Bottom-up, top-down challenges – leaves islands of addressing
- Dual stack (run both)
- Translate – convert IPv6 <-> IPv4. – But how do you handle those addresses
- Tunneling – IPv6 inside IPv4 – V4 is everywhere

IPv6 killer – NAT – Network Address Translation

- Use of private address spaces inside: – Homes – Mobile networks – Organisations
- All 'hiding' behind a single public IP address



Getting into the transport layer

- Leave all the packet to-and-fro to the network layer

• Everything here is a payload for IP packets – A Segment
• Offers rich functionality (or not) to Applications
  – Reliability, performance, security, and other quality measures – on unreliable IP
• Routers and other network devices do not get in the way
  – They (should) only look at 'the envelope' of a message, not the messages – This is pure host-to-host.

Simple client/server model
• Servers offer something,
• Clients connect – Send a request – Server replies
• Servers can handle multiple clients
  • Model breaks in p2p applications – everyone is both.

Transport Services
• What common application needs are there?
• Main decision:
  – **Reliable** - everything has to arrive bit-perfect.
    • Transport layer repairs packet loss, mis-ordering (and other damage) • I can wait!
  – **Unreliable**
    • Don't care about eventual perfection,
    • Do care about performance, simplicity, ...
    • Two types of communication
  – **Messages**: self-contained command and response (post office)
  – **Byte-stream**: generic flow of bytes, chunked into segments (conversation)

Which does what?
- UDP is an enhanced IP packet ;TCP is a lifestyle choice – many features

	Unreliable	**Reliable**
Messages	UDP (datagrams)	
Byte-stream		TCP (Streams)

- Could have reliable messages - but can build that on top of TCP
- Could have unreliable byte-streams - but that looks like UDP
  Transmission Control Protocol: TCP = IP Protocol 6 *User Datagram Protocol:* UDP = IP Protocol 17
  ICMP = 1, IGMP = 2, IPv6 encapsulation = 41, 130+ more

TCP	**UDP**
Connection-oriented (significant state in transport layer @host)	Connectionless (minimal state in transport layer)
Delivers BYTES: once, reliably, in order (to your process)	Delivers MESSAGES: 0-n times, any order
Any number of bytes (in a stream)	Fixed message size
Flow control (sender/receiver negotiate)	Don't care
Congestion control (sender/network negotiate)	

IP Multicast: UDP
Connectionless, maybe time-sensitive, Replica packets are fine!

Ports
IP is "host-to-host" ; Applications are process-to-process
Port: 16bit identifier(s) for a process, on a host, on an interface, at each end

• Opening ports below 1024 requires extra privileges

20,21	ftp	File transfer
22	ssh	Secure shell
25	smtp	Email – outbound
80	http	Web
110	pop3	Email – inbound
143	imap	Email – inbound
443	https	Secure-Web

A Port is just a start

• Inetd/xinetd
  – Don't continually run every server-service somebody may eventually talk to
  – Single service, launch appropriate service on demand
  – Listens to all (registered) ports and protocols (tcp, udp)
  – Spawns the service to have the conversation

• Port mapping
  – (e.g. remote procedure calls, bittorrent, ...) – Listen on a well-known port
  – Accept connections
  – Redirect them to a spawned service on another port [Services can register with the portmapper)

NAT is actually NAPT

• NAT has everyone 'hiding' behind a single public IP address
• But everyone wants access to/from the Internet at the same time
• <u>So translate **addresses** and **ports**</u>
• Router maintains a table – Dynamically for outbound. Can be static for inbound.
  "150.203.56.99:7880 = 10.0.0.2:80"
  "150.203.56.99:7881 = 10.0.0.4:80"

Byte-streams

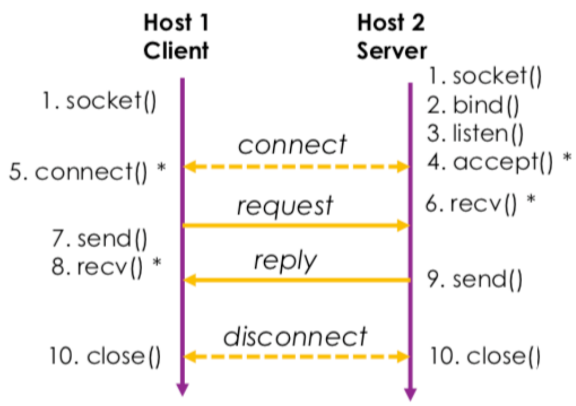
• TCP segments carry chunks of a byte-stream – "Message" boundaries are not preserved
• Sender packetises (eventually) on <u>*write()*</u> – Multiple writes can be one packet and vice-versa – buffer dependent
• Receiver unpacks – Applications <u>*read()*</u> a stream of bytes
• Hence: Segments

TCP Options

• These actually get used...
• Maximum Segment Size: how much each end is willing to take
• Window Scale: When 64kB is not enough – multiply
• Timestamp: For computing rtt and expanding sequence number space • Selective Acknowledgement: Like ACK, but better.

Programming connections

• "Socket" programming – an address, a port, and a need to communicate
• *Connections are identified in the Operating System by a '5-tuple'*
– source/destination ip, source/destination port, protocol
• Server needs to be prepared for connections
• Client initiates the connection



Socket API

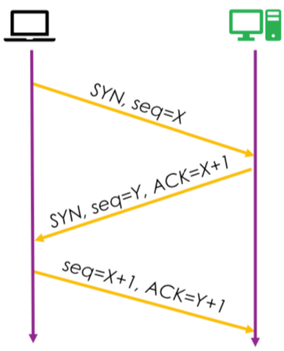
Primitive (function)	What it does
SOCKET	Create an object/descriptor
BIND	Attach a local address and port
LISTEN (tcp)	Tell network layer to get ready
ACCEPT (tcp)	Be ready!
CONNECT (tcp)	... Connect ...
SEND(tcp) or SENDTO(udp)	... Send ...
RECEIVE(tcp) or RECEIVEFROM(udp)	... Receive ...
CLOSE	Release the connection/socket

TCP and reliability

- TCP is a reliable, bidirectional byte-stream
  - Uses Sequence Numbers and Acknowledgements to provide reliability
  - Piggybacks control information on data segments in reverse direction
    - If there's no data, just sends feedback
- **Sequence numbers:** N-bit counter that wraps (e.g. ...,253, 254, 255, 0, 1, 2...)
  - Byte count (pointer) in a stream – a cumulative ACK
  - Can wrap quickly on high-speed links (2^32 = 4GB) – can use timestamps too
  - Does not start from zero (for security)
- **Acknowledgements**: Which bytes have been received/is expected

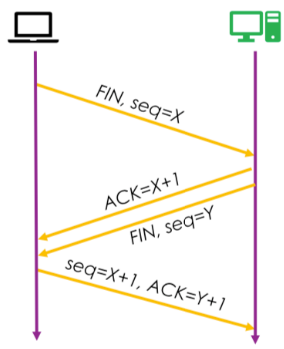
Getting connected – 3 way handshake

• TCP is full-duplex = two simplex paths – Both need to start together(*)
  • Synchronise Sequence numbers in both directions
• Connecting – Receiving transport stack decides:
• anybody *listen()* ing on that port?– If not, ReSeT– If yes, passed to receiving process listen()ing,
– Transport stack ACKnowledges
– Originator ACKs that SYN/ACK and off they go



Hanging up

- Both need to end together – Ideally...
  - Time to flush buffers
- Disconnecting
  - One side initiates *close()*
  - Triggers a FIN(alise)
  - Other side ACKs and FINs too
- And if FIN is lost? Resend...



Socket states:

State	Description
LISTEN	Accepting connections
ESTABLISHED	Connection up and passing data
SYN_SENT	Waiting for reply from remote endpoint
SYN_RECV	Session requested by remote, for a listen()ing socket
LAST_ACK	Closed; remote shut down; waiting for a final ACK
CLOSE_WAIT	Remote shut down; kernel waiting for application to close() socket
TIME_WAIT	Socket is waiting after close() for any packets left on the network
CLOSED	Socket is being cleared
CLOSING	Our socket shut; remote shut; not all data has been ACK'ed
FIN_WAIT1	We sent FIN, waiting on ACK
FIN_WAIT2	We sent FIN, got ACK, waiting on their FIN

TCP Sliding Windows

- Want reliability **and** throughput (of course!)
- Start with ARQ – stop-and-wait
  - – Single segment outstanding = problem on high bandwidth*delay networks
- Say one-way-delay=50ms so round-trip-time (RTT)=2d=100ms
- Single segment per RTT = 10 packets/s
  - Typical packet ? Say 1000 bytes = ~10,000 bits -> 100kb/s
- Even if bandwidth goes up, throughput doesn't!
- Allow W segments to be 'outstanding' (unACKed) per RTT
  - – Fill a pipeline/conveyor-belt with segments
- Set up a 'window' of W segments
- W=2*Bandwidth*delay
- At 100Mb/s, delay=50ms means W=10Mb
  - – Assuming same 10kb segments, W=1000 segments – 500 are out there somewhere!

If(lost) then: ARQ – "Go Back N"

- **Receiver** buffers just a single segment
- If it's the next one in sequence, ACK it, everyone happy • If it's not, drop it,
- Let sender retransmit what I'm actually waiting for
- **Sender** has a single timer. After timeout, resend (all) from (first) ACK-less.
- Really simple, but somewhat inefficient

ARQ – "Selective Repeat"

- Receiver buffers many segments – Reduce retransmissions
- ACK what has been received in order
- And also ACK received segments that aren't
  - – Any gaps indicates missing segment!
  - – SelectiveACK(SACK)
  - – TCP header has an ACKflag(1bit),andaSACKOption(32bits...)
    - – **3 duplicate ACKs (plus SACKs) trigger resend**
- **Sender** has a timer per unACKed-segment – As each timer expires, resend that segment
- Cope with (some) misordering. Way more efficient, now widespread

Everybody runs the same TCP...?

- No. There is no single TCP stack
- Many years of various optimisations, experiments, algorithms, ...
  - – Suited to various circumstances
  - – And as vulnerabilities have been found and mitigated (and found and ...)
- Doesn't impact the network, only hosts, so you can do what you want...

Application space
• Build <u>sessions</u> (a series of interactions)

– E.g. a web page with multiple resources, multiple sources – A videoconference between particular endpoints

• Build on top of TCP (reliable byte-stream) or UDP (unreliable messages) – And add whatever functionality they require – e.g. reliable UDP sessions?

• Applications have one or more application-layer protocols – E.g. http/https for webpages

• Also handle <u>Presentation</u>

• Manage:

– Content-types (images, video, audio, text,...)

– Content-encodings (compression, uuencode, mime, ...)

– Content-packaging (file formats, message types, ...)

– Content-selection (receiver capability negotiation)

• Deal with command and control between two endpoints – “I want X”– “You are about to receiveY”

• Often see plain-English application protocols

– Efficiency is for geeks, debugging is much easier

– Overheads are low(command headers vs data and lower-layers)

Helper protocols (are applications too!)
• ARP – translate between layer 3 (IP) and layer 2 (MAC)

• ICMP, IGMP – network control and feedback

• So (1) how do I get my IP address?– I need a routable/forwardable address to participate

• And(2)how do I get my name?

– **150.203.56.47** or **3018:ae8::ae00:98:8ac2** are not memorable, nor guessable

– www.anu.edu.au is

Dynamic Host Configuration Protocol...
• Problem: node wakes up, knows nothing.

• “What’s my IP, mask, router/gateway?”

– Needed to join the internet!– At least I have my MAC address.

• Solution 1: Manual configuration. Depends on local needs. Doesn’t scale.

• Solution 2: Automatic configuration, service from IT

• DHCP (1993 – ex BOOTP) – gives/leases you your IP address

DHCP application
• Client/server application,

• UDP, client port:68, server port:67 – just ARQ if no reply

• Bootstrap:

– How to send IP packets before IP is configured?

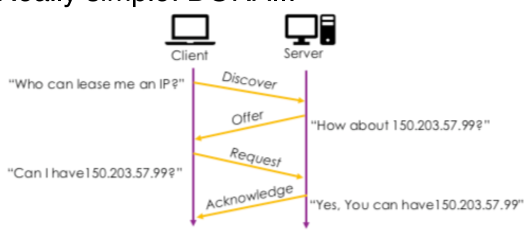
– How to send them to DHCP server when you don’t know where it is?

– Broadcast to the rescue! IP:255.255.255.255 => Ethernet ff:ff:ff:ff:ff:ff

– Source = 0.0.0.0

– DHCP server should be on the same LAN (broadcast domain) 【Or somebody needs to do some more work... 】

DHCP messages
• Really simple: DORA...

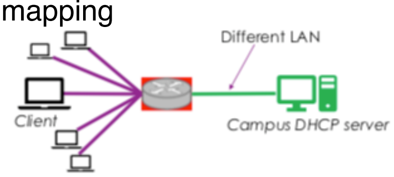


• Lease renewal:
  – Just REQUEST (can I please have) and ACK (yes you can)
    • unicast
  – If server disagrees:
    • Rejected (authoritative)
    • Ignored (passive) and timeout
• With new IP address, clients SHOULD (gratuitous) ARP to make sure it's ok...
  – Two DHCP servers; A manual/dynamic overlap;
• Actually a little more complex, due to BOOTP inheritance
  – Transition from BOOTP to DHCP with backwards compatibility
  – Packet format was kept, but purposes shuffled

DHCP does more

• DHCP relays
• Multiple DHCP servers (failover, performance)
• DHCP release – tell server to free up the address (optional) (*)
• 50+ features/records
– Subnet mask, router, time server, dns server, log server, boot files, smtp, ...

• Also allow for fixed ('static') MAC<->IP mapping



How does the DHCP server know?

• Manually configured, or
• Built off reasonable defaults
• Maintains database of who has what for when
• E.g. Home modem/router acting as DHCP server:
  – 192.168.x.y/24 subnet
  – DHCP server is the Default Route (to the Internet) – DHCP server is the DNS server

Domain Name System (DNS)

• Memorable, or guessable, names
  – www.anu.edu.au instead of 32-128 bits of addresses
  – A fixed name, rather than a variable address
• And a whole lot more!
  – Key service endpoints
  – Redirection, load balancing, dynamic allocation
  – Service metadata (priority)
  – Trust – somebody is in charge • Trust the device, if not the application, or the other user
• IP addresses and service endpoints change
• Why does an IP address change?
  – At home – ISP reallocation of your router
  – Organisational renumbering
    • Sold their block of IP addresses,
    • Relocating equipment, new server, ...
  – Mobile devices
• Having multiple devices that failover/share a service as needed – Web servers, email servers, directory servers, file servers, ...

Definitions

• <u>Names</u> (for humans) – not just devices/services, e.g. email address, social-media accounts, ...
• <u>Addresses</u> (for protocols) – not just TCP or IP or MAC, e.g. URLs

• **Resolution** maps between them– Definitively/unambiguously– Mostly downwards, but lookups can also be 'reversed'
• Note – a Name can have multiple Addresses – an Address can have multiple Names

DNS Design

• Provide a Resolution Service
  – Mostly to convert names to IP addresses (www.anu.edu.au = 130.56.66.152)
• Need to be
  – Easy to manage: many parties may be involved
  – Efficient: high data volumes, low-delays, low-load
• Build it:
  1. DistributedDirectory (no central database)
  2. HierarchicalNamespace (delegate to authorities)
  3. Automated protocol/processes for running it (set and forget(!))

DNS Namespace

• Everything starts from '.' – the ROOT
• Add a 'TOP LEVEL DOMAIN' (TLD)
  – Which may be 'generic' (gTLD) = com, edu, org, net, mil, gov, ...
  – Or a Country Code (ccTLD) = au, uk, us, it, fm, tv, to, ...
• And keep building up from there towards your hostname • A Fully Qualified Domain Name
• Like www.anu.edu.au. Or www.google.com.(orgoo.gl.)

How many TLDs?

• TLDs carry a lot of politics, and money, and culture, and ...
• Defined by IANA, implemented by ICANN
• 6 originals, notionally for defined purposes (com = commercial, ...)
• 7 new in 2000, .museum, .aero, .coop, .name, .info, .pro, .biz
  – Anger and confusion with .com and .biz!!
• 8 more from 2004-2012
• In 2008 new rules: No rules! Ok, some rules.
  – Financial model ($US185k),
  – Policies for each domain
  – Support for internationalisation (e.g. Chinese, Arabic, Cyrillic, ...)
  – Sponsored TLDs (industry sectors, like .aero)
  – Geographic TLDs that aren't countries (.kiwi, .asia, .paris, ...)
• In March 2018 – **1200 gTLDs**!
  – Lots of competition for the same names
  – Some very/too close •hotels and .hoteis .unicorn and .unicom

This creates jobs (for lawyers and marketers) but little extra value

ccTLDs

• Based on ISO 3166 two-letter country codes
  – Yet more politics!
  – "Country" can be a disputed topic...
  – Countries come and go too...
• Own sub-domain rules within ccTLDs
  – .edu.au (like US, and added .asn.au and .id.au) – .ac.jp – .uniX.de

Delegations = relationships = ownership

• Domains are what gets delegated - through legal entities
  – start from ICANN

– AU Registrar (auda.org.au) administers second-level-domains in **.au**
– Education Services Australia administers domains in **.edu.au**
– ANU administers domains (and hosts) in **.anu.edu.au**
– Colleges can have sub-domains, etc.

- Zones are shared pieces of the DNS database – through technology – Each zone identifies an authoritative nameserver – Each zone records delegations and their nameservers

What's in a zone?

- Information about
  – The zone, responsibilities – Further relationships (delegations) – And lots of addresses, services, etc. – And metadata about records (timeouts, etc.) – Through 'resource records'

RR Type	What it carries
SOA	Start of Authority – who's the boss
A	IPv4 address of a host
AAAA	IPv6 address of a host
CNAME	Canonical name, an alias
MX	eMail exchange for domain
NS	Nameserver of this or delegated domain

DNS resolution

- Depends on the query...
- Let's start with "What is the IP address of host X?"
- Without anything to go by, go to the root! – It knows everything? – It knows who might know more

DNS root servers

- https://www.iana.org/domains/root/servers
- 13 important (and tempting) boxes on the Internet (a..m.root-servers.org)
  – Actually, several hundred replicas
- Every nameserver knows about them
  – Default route is the root
- Reachable via 'anycast'
  – (advertise the same IP address)

Recursive and Iterative

- Iterative: "Hey NS, who is next in the tree?", then repeat
  – High performance, low delay – Provides a service
- Recursive: "Hey NS, you work it out, just give me the answer!"
  – Low performance, low impact
  – Good for the end client

Caching

- Performance of this doesn't scale
  – A web page can have hundreds of resources from unique servers
  – Client needs to contact all of them.
  – Many lookups for a single session!
  – Need a shortcut – only need the last one/two?
- Nameservers can cache iterative-query results
  – **.au** won't change often
  – **.edu.au** won't change often
  – **.anu.edu.au** won't change often
- But they will – so need a Time-to-live (*)

DNS Messages

- Simple, lightweight, UDP, port 53

– ARQ – stateless servers

– UDP: Need high-performance, minimise (TCP) load on the server 【However, there is a TCP option... (for really large responses) 】

• Same packet structure for queries and answers – Just flags are changed

- *Query or answer*
- *Recursion desired*
- *Recursion available*
- *Reply is authoritative*

• Messages carry a 16-bit ID

32bits

Identifier	*Flags*
# of questions	*# of answer RRs*
# of authority RRs	*# of additional RRs*
Question(s) {some number}	
Answer(s) {some number of RRs}	
Authority(ies) {some number of RRs}	
Additional info {some number of RRs}	

Of course this is secure. Right?

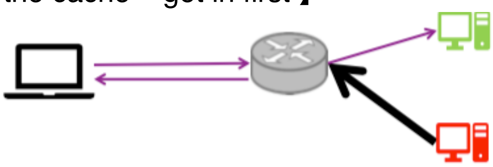
• Uhm – no.

• Villain-in-the-middle can corrupt/tamper/interfere with DNS queries

• Can redirect anybody, e.g. your connection to your bank's server...

– Hack the authoritative nameserver?

– "Hack" the caches/intermediary nameservers? 【 Actually spoofing - poison the cache – get in first 】



DNS (in)security

• Must be tricky?

1. How does villain know what to send?
2. How does villain make it look real?
3. What happens when real reply turns up?

• Actually, not as hard as we'd like – Not that it's "easy"

• Don't try this at home, or anywhere, ok?

• What to send? – Make the query yourself! Villain is just another client...

• Make it real? Circumvent DNS checks.

– Nameserver just checks headers:

1. Is it from a known server?
2. Does ID match?
3. Does it help an outstanding-query?

– but not the content

1. Make source-IP the IP of an authority
2. Sends lots of replies with guessed/snoopedID(16-bit)
3. Send(flood!) the reply immediately after a query

And third?

• What happens when the real response arrives?

– Remember: Nameserver just checks

• Is it from a known server? • Does ID match?• Does it help an outstanding-query?

– But there's no longer an outstanding query...

– And so that response gets ignored

– And the DNS server is now caching your poisoned record...

Bring on DNS Security!

• Easy? DNSSEC...

– Integrity and **authenticity** – it just adds authentication – Not about confidentiality (quite the opposite!)

• Extend DNS with new resource records
• Been discussed since 1997,
• Reasonably final by 2005,
• Root servers upgraded in 2010, • but the rest, and the clients...?

New RRs

• RRSIG
  – Digital signatures of a set of domain records • Clusters of all your A, AAAA, MX, ...
• DNSKEY
  – Public key for RRSIG signatures
  – Actually, two – Zone Signing Key (ZSK) and Key Signing Key (KSK).
    • KSK >> ZSK, reduces load on nameservers for key-validation. Need to trust the key!
• DS
  – Delegation Server key – for delegated zones
  – And CDNSKEY and CDS for delegated zone servers to propagate upwards

DNSSEC needs

• Try to minimise encryption overheads
  – DNS is a very popular transactional protocol – every transaction begins here!
  – Delays are bad.
  – Allow for new encryption techniques to be swapped in • And keys to be rolled-over
• Other RRs such as NSEC/NSEC3 – authenticated "no such name"
  – Unfortunately, this leaks zone information. People like to probe networks...
  – Quote: "Either lie, or don't trust DNS to hold your secrets." 【Avoid highlighting interesting endpoints.】

So what changes?

• Query Nameservers as before, AND
• Validate replies for authenticit
  – From the top down, PKI chain of trust
  – Anchor is the root public key
  – Every reply carries the necessary keys

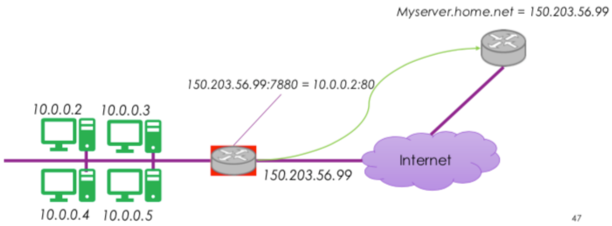
1. Use **key(root)** to check real-NS(.au)
2. Use **key(.au)** to check real-NS(.edu.au)
3. Use **key(.edu.au)** to check real-NS(.anu.edu.au)
4. Use **key(.anu.edu.au)** to confirm-IP(www.anu.edu.au)

Today?

• DNSSEC requires both clients and servers to update • gTLDs (common ones) approaching 90%
• ccTLDs approaching 50%
• Lower domain levels from 2-90%
• Applications... maybe 10-15%?
• Don't even think about 'smart devices'
  – Web-cameras, baby monitors, home-security systems, ...

“Dynamic DNS”

• Remember your NAT box at home?
– With its changing IP address?
– And that webserver running behind it?



Other DNS features

- Multiple names can point to one IP
  - One physical server hosting multiple virtual webservers
- One name can point to multiple IPs – Failover/load-balance
- Reverse lookups
  - Ensure connection from IP is from a domain, e.g. email spoofing, site validation
  - Uses a PTR record, in the .in-addr.arpa domain
  - Query for the PTR of D.C.B.A.in-addr.arpa points to the A record (the forward)

- Sort-list:
  - Can prioritise from a list of response – e.g. ‘in your prefix’ vs ‘not’
  - Useful for e.g. ‘nearest’ server, or for multi-interface servers
- Geopolitical-sensitivities – split DNS
  - What you get back depends on *where* you ask from
    - E.g. within some countries you can’t get to some domains...
- Round-robin/”load-balancing”
  - Send a list, in different order each time
  - Broken a little by caching, and not knowing the actual load
- LOC records
  - Latitude, longitude
  - and Altitude - from -100km up to +42000km
  - along with ‘precision’ of 1cm to 90000km
- SRV records
  - Identify service endpoints
    - That aren’t email (MX)
  - by Protocol Name and Type, and priority and weight – e.g. SIP, XMPP, STUN, Minecraft, ...

UDP-based applications:

- Short messages
- Simple request/response transactions
- Light server touch
- ARQ suffices

TCP-based applications:

- Larger content transfers
- Longer, and more complex, sessions
- Reliability matters
- Packaging and presentation becomes important – TCP is a byte stream

World Wide Web

Core idea: HTML to link “stuff”; need a protocol HTTP(IETF); Now: W3C.org
HTTP underpins the web
to deliver html and (many) associated content items
Request(s)/response(s) from multiple resources/sites
Port 0, TCP, A few versions

Aggregating and linking resources need IDENTIFIERS
(URI)
Or is that a (URN)?
Or a U (URL)?
• Stick with URLs here scheme:[//[
e.g: https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml
there are 280 schemes. Others:
Callto://<phone-number>
Tel://<phone-number>
mailto://<email-address>
File://<path-to-file-on-my-system>
ftp://<some-host>/some-file
http:// and https://
e.g **http**://user:password@host:port [/path][?query][#fragment]
You can provide authentication inline. If you want. In plain text...
Host = something you find in the DNS (or an IP address)
Port=ifit'snot80,tellme
Path identifies (absolute-path-to) resource on the host
– **#fragment** goes to a point within that resource
– http://en.wikipedia.org/wiki/IEEE_802#See_also
Query passes information to that resource

8 Steps to HTTP happiness

1.
2.
3.
4.
5.
6.
7.
8.

HTTP requests – RFC1945 (HTTP 1.0)

• Request/response, text based, start with the **method**
GET <path> HTTP/1.0 :Get the resource at <path>

HEAD <path> HTTP/1.0 :Get the headers about the resource at <path>
POST <path> HTTP/1.0 : Append my contribution to the resource at <path >
Requests indicate the protocol version – Servers provide backwards compatibility
Server returns headers, and a body (entity)

HTTP Responses

Code	Category	Example
1XX		No longer used; could be used
2XX		**200** ; 201 Created;
3XX		**301** **302**
4XX		400 Bad request; 403 Forbidden; 404
5XX		400 Bad request; 403 Forbidden; **404 Not Found**

Headers (both directions)

• Provide information about the resource
– Or additional information about HTTP codes – Or other hints about the server/client

Function	Examples
( )	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
( )	Date, Last-Modified, Expires, Cache-Control, Etag, If-Modified-Since, If-None-Match
( )	Cookie, Referer, Authorization, Host, Range
( )	Content-Encoding, -Length, -Type, -Language, -Range, Set-Cookie, **Location**

HTTP is stateless

•
– Server shouldn't
– How do I stay logged-in?

•Encode in a .
•Encode a in
– Set by the server, held by the client – and returned whenever "relevant" (same domain).
– Include various tags/types/flags, and the domain that set them. Sort of.

•*Session Cookie* –
•*Persistent Cookie* –
•*Secure Cookie* – only over secure channels
•*And more….*

Protocol Performance

Measured in www by user experience –
Depends on:

– Browser
– Content structure and complexity, processing
– Protocols:
– Network path, bandwidth and round-trip-time

Typical web page
– Core html
– Plus scripts, css, images, frames/divs, ...
  • Each is their own 'object' for GETting

HTTP 1.0
connection for each page resource
Sequential request/response
  – to the one server
  – TCP overhead on
  – Network and endpoints idle for significant periods => Only delivering for a small fraction of time
• Easy – but slow
• Worse with many small resources (and TCP throughput has performance limits too...)

Improvements to "Page Load Times"
• Adjust content to suit client–
• – Avoid getting the same thing multiple times
• – Be smarter with its connections

Smarter (http) connections
•
– Instead of one http GET, just do 8+ at the same time...! • No needed
  Take advantage of
  Creates bursts of CPU/NIC load, traffic and loss
•
– sequential requests (HTTP 1.1) – Open one TCP connection
– And use it for multiple requests in order
•
– Make all your requests at once
– Responses come back in order

In real world: images cost the most on performance, then is .js

More performance: Caching
• In the browser
  – Don't download what you grabbed earlier
  – Populated on demand
• Along the path
  – Same idea, bigger and better and SHARED – win for you and your
  – – on your behalf
• (CDNs)
  –

The art of caching
How do you know cache is good?
  Expires header (HTTP 1.0) Should...
  Last-modified header (HTTP 1.0) • If have it–take a guess, If no have it – ask for it (**HEAD** method)

E(ntity)**Tag** (HTTP [redacted]) Like a checksum, a small HEAD request

- ‘[redacted] **GET**’ (HTTP [redacted])

  Header: If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match

- ‘[redacted] **GET**’ (HTTP [redacted])

  Range header - only part of the entity is transferred

Proxies

- Somebody else does the work for you – Hide network internals, protect clients, ...
- **Proxying cache** – or – **Caching Proxy**
  - Put cache out further on the network and share it
  - – Win: Performance
  - – Win: Network traffic reduction
  - – Win: Security checking
  - – Win: Organisation Access Policies!!
  - Lose: Not for secured content
  - Lose: Not for dynamic content
  - Lose: Gets filled with lots of ‘fluff’

Content Distribution Network

- Invert the picture:
- **Push** [redacted] to caches [redacted] the request
- Use [redacted] – Html encodes [redacted]
- Take [redacted] popular sites – And the [redacted] host them

– Win:Win:Win

- Akamai (~1996) pioneered this
  - – It’s a commercial service (benefits clients)
  - – They see a lot of network behaviours (benefits Akamai)

Ever faster/better

- HTTP 2.0 (newest)

  Better [redacted] requests

  Client can [redacted] server responses

  [redacted] compression

  Server push “You’ll probably want this too”

  Slowly appearing, some contentious elements – expect to see HTTP 2.1? \

- [redacted] improvements?

  Some Apache modules rewrite/repackage your page (and code) on the fly…

HTTP as a ‘transport’ protocol?

- It carries real-time audio/video!?!

  Various web-conferencing apps – vs RTP, RTSP, ...

- SOAP and REST

  Simple Object Access Protocol

  Representational State Transfer

  Remote Procedure Calls (RPC) over HTTP

- Used as a [redacted]

  Everything (else) gets blocked!

  So let’s use HTTP...