LAN

• Ignore the arbitrary ‘distance’ classification
• LAN is where everything starts
– You connect devices to a LAN
– First networks were point-to-point links (2 devices)
– First useful network had many devices, connected to a common network – Started from shared medium/bus topology
• Leads to 2 better concepts:
– Administrative domain – one standard (media, modulation, encoding, ...)
– Broadcast 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** guaranteed delivery
• **No** built-in error-detection/correction
• **No** specialised features (bulk-transfers, realtime, ...)
• Leave all those hard things to the software – in general

Multiplexing

• Multiplexing: – By time – By space– By frequency
• Everyone gets an equal share
• But not everybody needs that much – It’s effectively a circuit – Wasting capacity

Statistical Multiplexing

• Demand for capacity varies with time
– Random access-priority 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** – a fixed pipe, just for you, tied to both endpoints – Send bits whenever you want, to the other end
• **Cells** – Time Division Multiplexing – Send a specific number of bits, when it’s your turn, to the other end
• **Frames** – arbitrary length, targeted messages – Send bits as and/when you want ?!?

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 • hog,
• use up all the memory at the receiver, • let errors pile up
• 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 synch ; 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 waiting for next frame to start

– Slight wrinkle: 'flag' byte inside the payload
  • Put an 'escape' byte before any 'flag' bytes inside the payload

– New wrinkle: 'escape' byte or 'escape/flag' byte pair in the payload • Need to 'escape' the 'escape' – and so on. • Byte stuffing...

MAC and sharing

• Media Access Control

• Needs an address scheme

– 'MAC address' – hardwired (sort of) to your network interface – Identity of the interface, not the computer

– Listen to (receive) all traffic, respond to stuff sent to you

• Needs an access scheme for multiple devices ("multiple access"...)

– No one is in charge. Think 'party atmosphere'

• Two common models

– Randomised access – try your luck

– Contention-free access – stricter rules Or Channel access

Randomised access

• ALOHA (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 load isn't too high

– Statistical performance up to 18%-36%.

– Performance depends on the back-off scheme, and better designs exist

CSMA

• Carrier-Sense Multiple Access

– Good for wired (copper) 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 long cables. Two senders can start at the same time without realising – As bandwidth*delay gets bigger, problem gets bigger

– Sets upper limits on delay, and minimum frame size
  • Need enough time to detect a collision,
  • Whole frame could be out, you start next one...
  • Minimum frame time is 2*delay

CSMA/C*

• Collision Avoidance (CSMA/CA)

– Listen for carrier – Once it's clear, wait a random time
  • Avoid all the waiting terminals starting at the same time

– Then send the whole frame

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

Backing off...

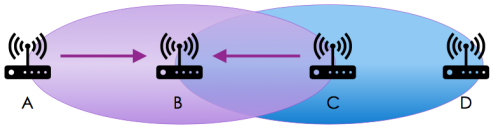
• Need some limits on how long to back-off – Can’t be too short – Can’t be too long
• Ideally back-off depending on the number of terminals on the LAN – Send with 1/N probability. How do you know N?
• Binary Exponential Back-off (BEB)
  • Counting the number of collisions you experience
  • First time,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

Wireless is harder

Each node has ‘coverage’ – i.e. may be no single carrier to sense
– Can be part of a single network but can’t see all the nodes (power limitation)
In wireless, TX can be a million times louder than RX – i.e. can’t listen for collisions
– Can’t respond quickly to collisions, lots of wasted time

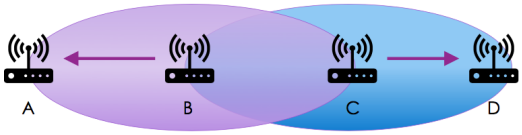
Hidden terminals (stations)

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



Exposed terminals

B and C are “exposed terminals” to each other:
Both could talk to an outer neighbour – and NOT interfere Want to avoid wasting bandwidth

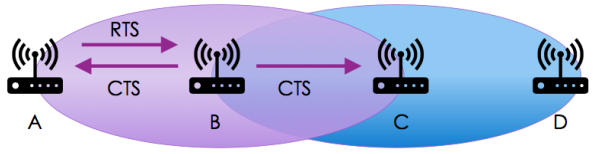


MACA

• Multiple Access, Collision Avoidance
  A quick handshake before yelling:
• **Sender**: Request to Send (RTS) + N bytes
• **Receiver**: Clear to Send (CTS) + N bytes
• Sender transmits
– and any nodes that heard the CTS stay silent for N bytes – and any nodes that heard the RTS stay silent for the CTS

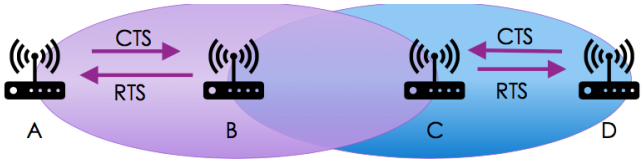
Fixing hidden terminal problem

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. Token Ring: 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 'destination' 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 repeater or hub (regen)
    • Or a bridge, which links two LANs, and learns addresses on each side
• Today nearly all are 'switched'
  – Crossbar devices that learn source/destination addresses from the traffic
  – Makes every link point-to-point (computer to switch)
  – Greater scalability & Greater performance

———————————————————-LAN———————————————————

Acronym overload

• General "LANs": Ethernet, WiFi, Bluetooth, 4G?
  – 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: MAC details. • 802.3: Ethernet • 802.11: Wireless LAN & Mesh
• 802.15: Wireless PAN • 802.16: Broadband Wireless Access (WiMAX)
– 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:

– 1Gb/s, Baseband – Fibre optic pairs,
– 1310nm over MMF (500m) or SMF (10km)
– 8b/10b NRZ encoding

- **1000BASE-T**

– 1Gb/s, Baseband
– Twisted Pair copper cables with 8P8C (RJ45) connectors, Cat5e or better
– Uses all four pairs, in both-directions simultaneously
– 4b/5b, PAM-5 encoding

Ethernet

- Started over shared medium coax – CSMA-CD, Manchester
  - – 10Base2, 10Base
  - – Moved to UTP, 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 *plug-and-play* – Well designed to cope with network changes
- Very good *backwards compatibility* – 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 ‘promiscuous 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

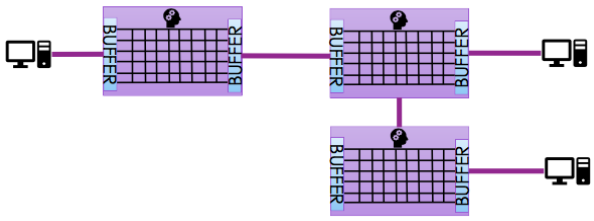
More scalable, more reliable

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.

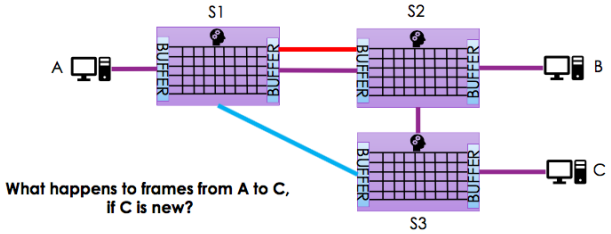
Hierarchy of switches

• 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

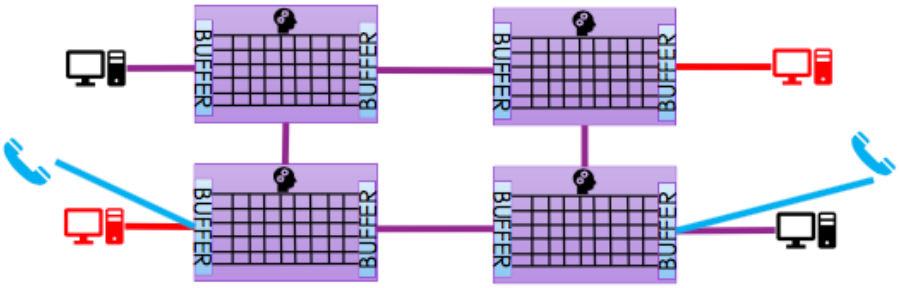
- **Broad-cast**: Everyone gets it. MAC destination = all 1's (ff:ff:ff:ff:ff:ff)
- **Uni-cast**: Only the intended recipient (should) get it. **MAC** destination
- **Multi-cast**: Everyone who is interested gets it
  - – Special bit-flag in MAC address
  - – Devices can 'subscribe' their NIC

Special features

- Link aggregation/ "trunking"
  - – When a single 1Gb/s or 10Gb/s won't do – Performance – Failover
- 802.1AX (was 802.3ad) – up to 8 (identical) links
- Link Aggregation Control Protocol (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

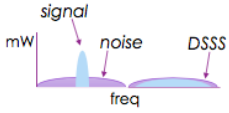
WiFi! (aka WLAN)

- 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

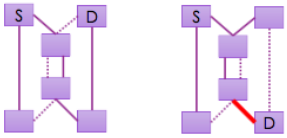
Scaling

• 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)
• *Different* ***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

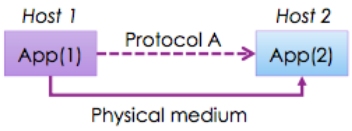
• Want to communicate across networks: aka Inter-network • Take the **LAN** to the **WAN**
• *Scaling problems* – Use a hierarchy of connections, addresses and aggregate/group
• *Traffic control* – Optimise routing with more information, and support prioritisation
• *Which LAN?* – All of them. Put a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

Why stop at networks?

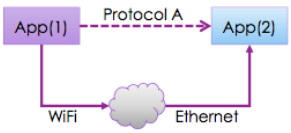
• Applications need functionality too
– Find/advertise resources
– Connect to other machines • One or many
– Exchange application-specific messages
– Adapt to capabilities : Devices, software, ...
– Maintain state of a connection
– Expect sufficient reliability
– Expect trustworthiness
– Maximise performance,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

Layers upon layers

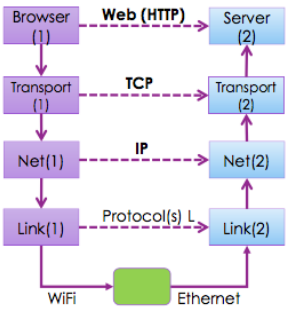
• Use **layers** to divide (allocate) functionality
• Use **protocols** to exchange information within a layer • User **services** from lower layers to build on



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?**
Network layer: 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

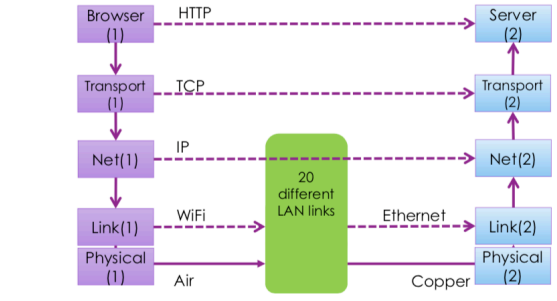
Role of the (IP) Network layer

- 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.

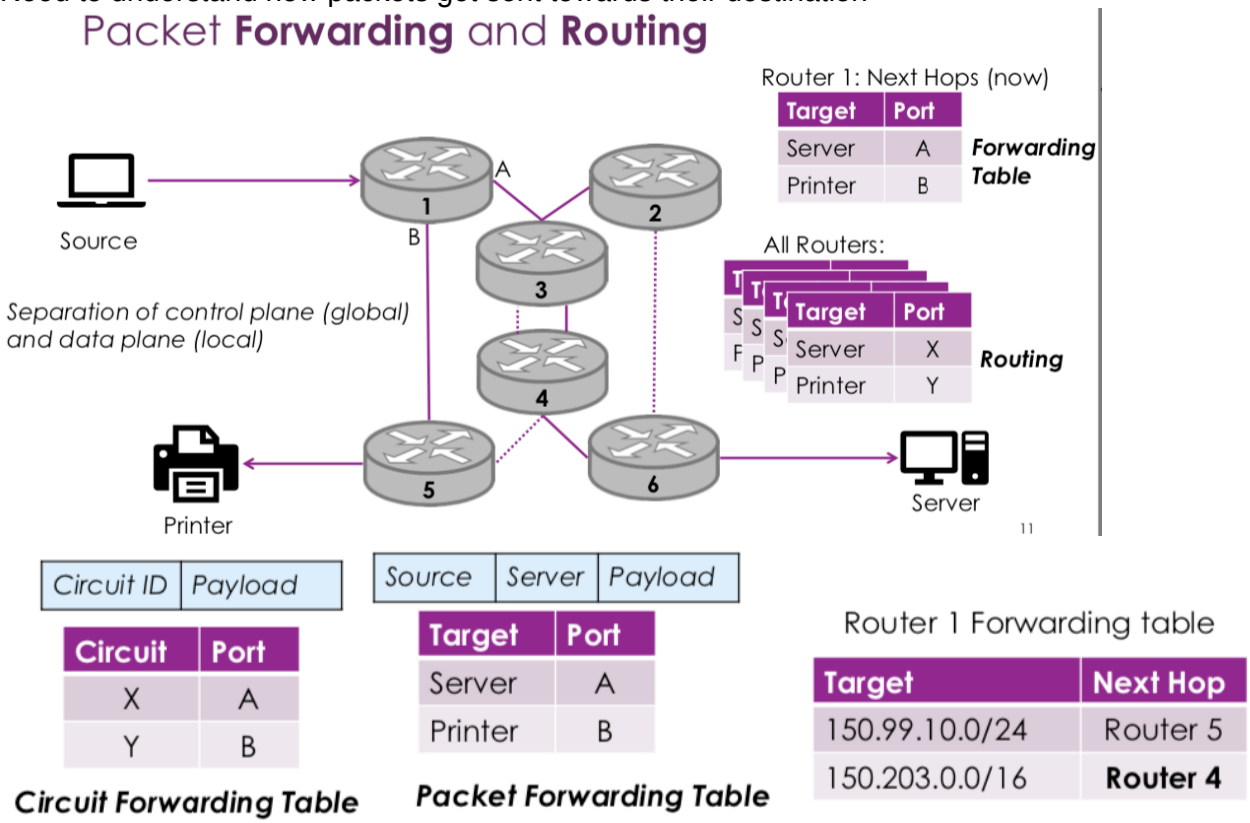
Guiding principles

- 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



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

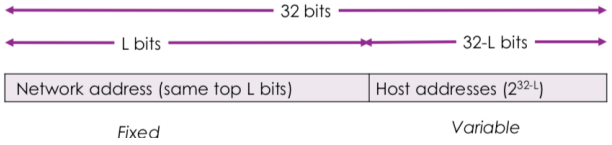
- 32-bits = 2^32 hosts = ~4billion - in theory • Written in 'dotted-quad' notation

– 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:



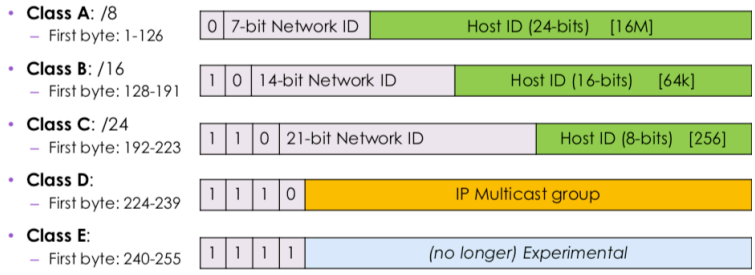
- Use a '/' ('slash') notation:
- Network address/prefixlength:
  - Network address is the first 'host' in the host-range

• 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

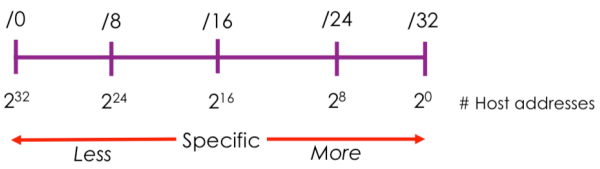
• Network address/prefixlength •Prefix length = a subnet mask
• Network address = a subnet (a block of contiguous host addresses)• For example: 150.203.10.0/24
  – The 150.203.10.0 subnet
  – /24 = 24-bit network id so mask= 24 1's and 8 0's: 255.255.255.0

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	*Longest matching prefix*
0.0.0.0/0	My (default) Router *150.203.56.1*	*…which is also on my LAN*

Home on the LAN

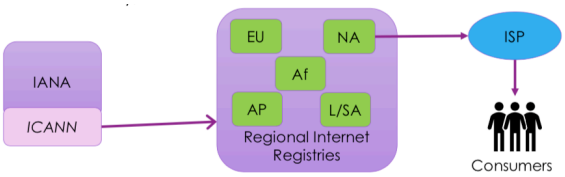
• Network-Layer:
  – Hey, I'm on the same Ethernet as my target, I can just send this packet directly
  – Link layer, send this to IP-address 150.203.56.99
• Link-Layer:
  – What's an IP address????– I need a MAC (Link Layer) address – Network-layer won't help me
• Need to cross layers. Need some kind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

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

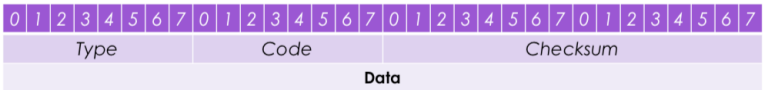
• Subnet broadcast: A.B.C.255
  – All ones in the host field (.255 for /24, .255.255 for /16)
• The subnet: A.B.C.0
  – Aka “the wire”, usually followed by /n – All zeroes in the host field
• Router/gateway: A.B.C.1
  – A convention. Makes it easy to find
• Note: Host field is shorter than 8 bits, for 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



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...

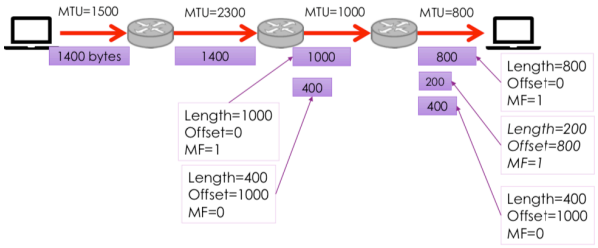
Traceroute

- 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

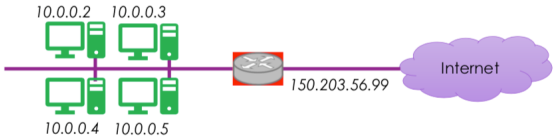
- Every router has a forwarding table – Fortunately only 10-100 interfaces
- Across the whole internet
  – Lookup tables of ~1M entries– Fuzzy matching on 32bit address – In under 5ns (100Gb/s)
- And 170k updates/day – 2 per second
- Address blocks are getting much harder to aggregate

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
• So translate **addresses** and **ports**
• 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

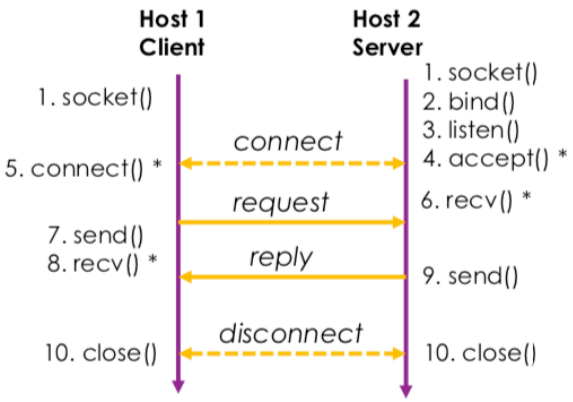
• TCP segments carry chunks of a byte-stream – "Message" boundaries are not preserved
• Sender packetises (eventually) on *write()* – Multiple writes can be one packet and vice-versa – buffer dependent
• Receiver unpacks – Applications *read()* a stream of bytes
• Hence: Segments

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

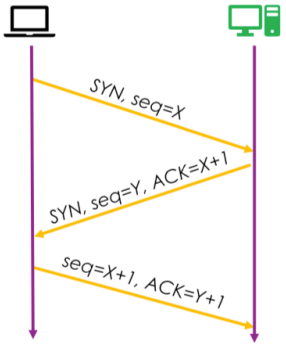
- TCP is a reliable, bidirectional byte-stream
  - – Uses Sequence Numbers and Acknowledgements to provide reliability
  - – Piggybacks control information on data segments in reverse direction
    - If there's no data, just sends feedback
- **Sequence numbers:** N-bit counter that wraps (e.g. ...,253, 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,

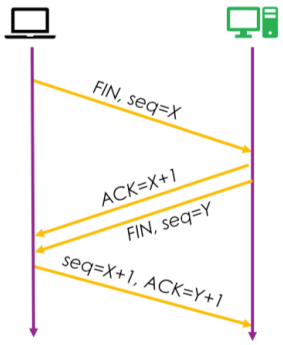
– Transport stack ACKnowledges

– 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 sessions (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 Presentation
• 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...

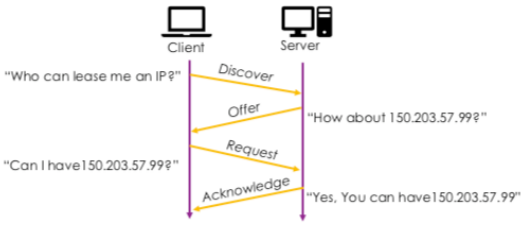
• Problem: node wakes up, knows nothing.
• “What’s my IP, mask, router/gateway?”
– Needed to join the internet!– At least I have my MAC address.
• Solution 1: Manual configuration. Depends on local needs. Doesn’t scale.
• Solution 2: Automatic configuration, service from IT
• DHCP (1993 – ex BOOTP) – 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...

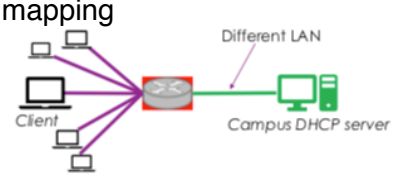


• Lease renewal:
  – Just REQUEST (can I please have) and ACK (yes you can)
    • unicast
  – If server disagrees:
    • Rejected (authoritative)
    • Ignored (passive) and timeout
• With new IP address, clients SHOULD (gratuitous) ARP to make sure it's ok...
  – Two DHCP servers; A manual/dynamic overlap;
• 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

• Names (for humans) – not just devices/services, e.g. email address, social-media accounts, ...
• Addresses (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?

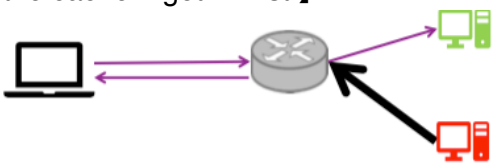
• Uhm – no.

• 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

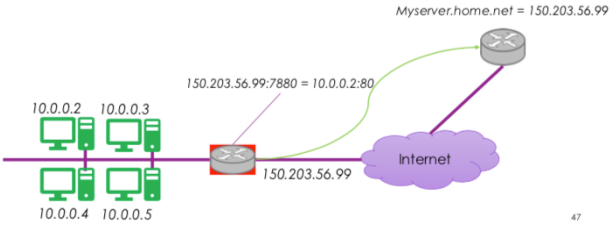
1. Use **key(root)** to check real-NS(.au)
2. Use **key(.au)** to check real-NS(.edu.au)
3. Use **key(.edu.au)** to check real-NS(.anu.edu.au)
4. Use **key(.anu.edu.au)** to confirm-IP(www.anu.edu.au)

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 80, TCP, A few versions

Aggregating and linking resources need IDENTIFIERS
    Uniform Resource Identifiers (URI)
        Or is that a Uniform Resource Name (URN)?
        Or a Uniform Resource Locator (URL)?
• Stick with URLs here scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
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. Parse URL
2. Resolve DNS
3. Connect to host:port via TCP
4. Make HTTP request
5. Receive content
6. Close TCP connections
7. Unpack content
8. Render

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	Information	No longer used; could be used
2XX	Successful	**200 OK**; 201 Created;
3XX	Redirection	**301 Moved Permanently;** **302 Moved Temporarily**
4XX	Client Error	400 Bad request; 403 Forbidden; 404 Not Found
5XX	Server Error	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
Browser Capabilities (c->s)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching (both)	Date, Last-Modified, Expires, Cache-Control, Etag, If-Modified-Since, If-None-Match
Browser sates(c->s)	Cookie, Referer, Authorization, Host, Range
Content delivery(s->c)	Content-Encoding, -Length, -Type, -Language, -Range, Set-Cookie, **Location**

HTTP is stateless

- Every request stands alone
  - Server shouldn't hold state for everything
  - How do I stay logged-in?
- Encode a session key in a URL
- Encode a session key in Cookies!
  - 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* – deleted when browser closes
- *Persistent Cookie* – kept till expiry
- *Secure Cookie* – only over secure channels
- *And more….*

Protocol Performance

Measured in www by user experience – *Page Load Times*
Depends on:

– Browser
– Content structure and complexity, processing
– Protocols: HTTP, TCP, IP
– 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

One TCP connection for each page resource
Sequential request/response
    – Multiple TCP connections to the one server
    – TCP overhead on set-up/tear-down
    – 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– Small screen, small images, Large screen, large images
• Add **caching**– Avoid getting the same thing multiple times
• Change **http** – Be smarter with its connections

Smarter (http) connections

• **Parallelism**
– Instead of one http GET, just do 8+ at the same time...! • No server change needed
    Take advantage of idle bandwidth
    Creates bursts of CPU/NIC load, traffic and loss
• **Persistence**
– sequential requests (HTTP 1.1) – Open one TCP connection
– And use it for multiple requests in order
• **Persistence** + **pipelining**
– 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 ISP/org
    – Proxy caches – on your behalf
• Content Distribution Networks (CDNs)
    – Replicate the site somewhere closer to you – And then act like a cache

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 1.1) Like a checksum, a small HEAD request
• '**conditional GET**' (HTTP 1.0)
Header: If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match
• '**partial GET'** (HTTP 1.1)
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** content to caches **before** the request
• Use **DNS** to send clients to nearest cache – Html encodes cache locations
• Take traffic load off popular sites – And the ISPs that 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 pipelining of requests
Client can prioritise server responses
Header compression
Server push "You'll probably want this too"
Slowly appearing, some contentious elements – expect to see HTTP 2.1? \
• Server-side application-level 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 firewall-traversal-protocol
Everything (else) gets blocked!
So let's use HTTP...

What about real-time traffic?

Audio/Video applications are obvious – And exponentially hard (N sites)^2
But what about robotics?What about tele-medicine??
What about broader system control – (open a gate, close a valve, ...)?
When do you trade off: timeliness (delays) for reliability (loss)?

How real is real-time?

Very application specific

– How much can you tolerate a single lost packet?
– How much can you tolerate a delayed packet?
– How much can you notice a delay? E.g. streaming vs videoconf; And different audio/video delays (synch) are worse

• TCP for streaming (**one**-way), with CDNs – Delays less crucial, win on reliability
• UDP for (**two**-way) interactive, real-time – low-delay, low-overheads

Why is it hard?

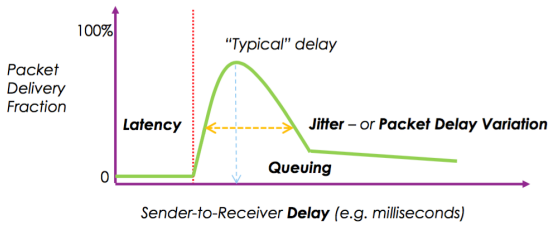
- Real-time media – e.g. videoconference
- Internet is 'best-effort' – *Unless you have circuits*
- Any **delay** is a problem – e.g **variable** delay,**"loss"** delay is another –-<cause>-> Various loss

Network delays

- Sender is sending a constant stream of audio/video samples
  - Bandwidth may vary depending on codec
  – Receiver expects to receive a constant stream!
- But all those routers don't belong to us... neither do the paths
- We need the **path**(s), the **bandwidth**, and capacity on the **routers**

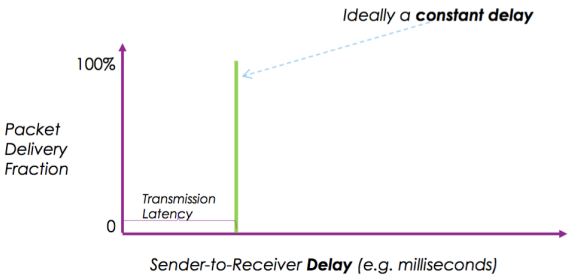
Network delay elements

Tanenbaum and Wetherall



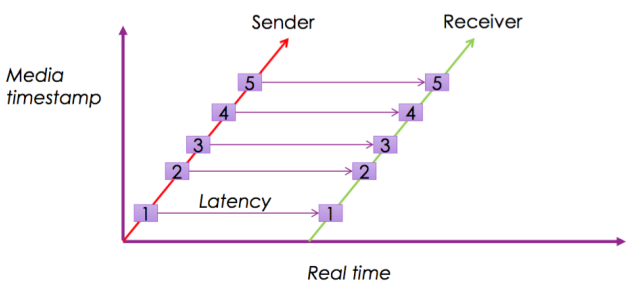
Network delay elements

Tanenbaum and Wetherall



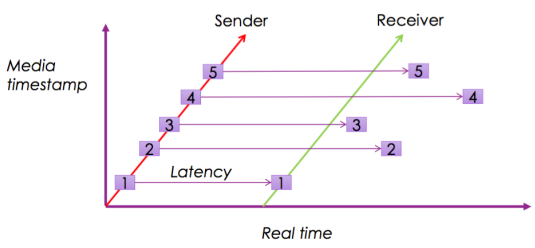
Ideally, sending packets

Ideally, network delay is constant (PDV=0) – like a telephone circuit

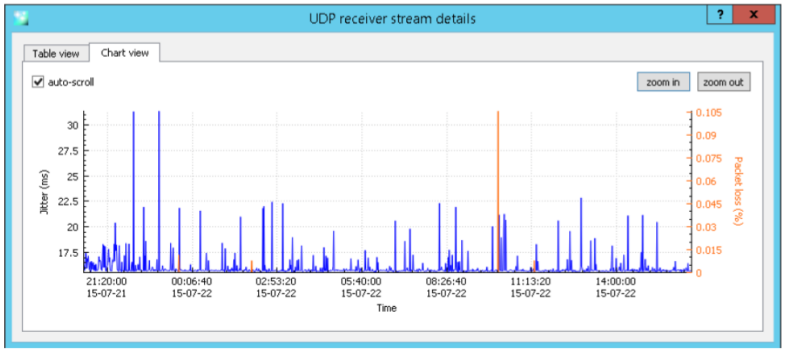


Reality...

Packet delays are random, and packet order can be messy



And jitter changes over time

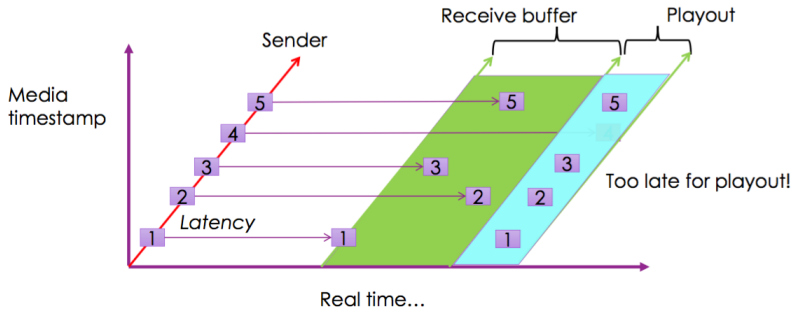


Buffering (TCP-lite)

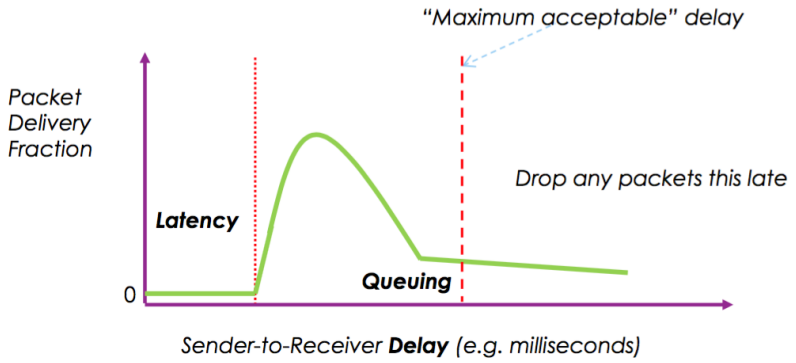
• Sender is sending a constant stream of audio/video samples – Receiver expects to receive a constant stream!

• So Receiver makes a playout buffer – smooth out variable delays – Measured in **bytes**, but effectively **in time**

This solves everything? …



Network delay elements



It's a trade-off

• **Big** buffer:
  – Fewer packets lost to delays = More tolerant of jitter/PDV
  – Greater delay between transmission and playout

• **Small** buffer:
  – More packets may be lost to delays = Less tolerant of jitter/PDV

– Smaller delay between transmission and playout
- The smaller the (desired) delay, the harder to deal with loss – so you “glitch”
  – Might be ok for video, less so for audio, but ...

Fixes?

- **Retransmission**
  – You know something is lost
  – Request a resend of the problem packet • ARQ -> TCP-like!
  – Takes *round-trip-time* + *transmission time* + *queuing delays =>* Huge buffer/delays implied
  – Generally not done
  – In multicast, it may not be the sender who retransmits!
- **Elastic buffers**
  - When things go bad, you adapt – stretch the buffer
  – And playout *slower*: Easy to stretch/squeeze audio and video...
  – Shrink the buffer when things get better: Want to keep as close to real-time as possible
- **Error-correction**
  – Encode media for interpolation between packets
    • Anything missing, you have an approximation .
    • Similar to compression algorithms, progressive-display images, ...
  – (Adaptive) Forward Error Correction
    • Anything broken, you can reconstruct
    • Useful for (control) reliability, or where retransmission is too expensive
- **Parallel-transmission**
  – Send several copies at the same time
    • At (multiple) different (lower) qualities
      Low quality audio/video still better than no audio/video!
      And also useful for control reliability

Realtime Transport Protocol (RTP)

- UDP payload (or TCP)
- Allows for any media (stream) encodings
- Allows for multiple sources to be merged, identified, and synchronised – Lip-synch audio to video

RTP Feedback

- As **sender**, would be useful to know:
  – How well is my stream getting through?
    • Should I adjust rate, encoding, error-correction, retransmission, ...
  – How many endpoints are receiving it? (multicast) • And identify them
- RTP Control Protocol (RTCP)
  – Bidirectional, out-of-band signalling • But need to limit bandwidth usage!
  – Sender Reports, Receiver Reports • Statistics: Packets received/lost, delay, delay variation,
  – Source Description, Hello/Goodbye, ...
  – Also provides heartbeat, distance-measure, retransmission-request channel, ...

All together: a videoconference

- Multiple sites, multiple media streams
- Need to:
  – Establish a call: *Session Initiation Protocol (SIP)*
  – Negotiate the details: *Session Description Protocol (SDP)*
  – Deliver the media: *RTP and RTCP*
  – Playout the media, as reliably and quickly as you can: buffers etc

Session Initiation Protocol (SIP)

Open (IETF) protocol to establish and tear-down calls (rfc3261+++)

- – Doesn't care how you transport the media
- – Is not used by Skype, Messenger, WhatsApp, Facetime, Zoom, Lync, ...
- – Competes with ITU H.323

• Widely used for Voice-over-IP (VoIP) = Internet Telephony

• Includes proxies, registrars, redirectors, border controllers, and gateways – Useful for "mobile" users, large directories, NATs, PSTN connections, etc.

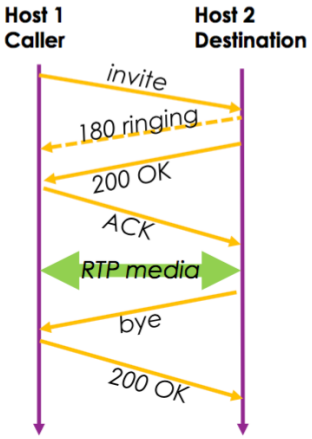
SIP signalling

• Looks a lot like a phone call • Looks a lot like HTTP

– Commands, Responses, Code-classes

– Runs over TCP and UDP

– (and not – no XYZ codes?)



Non-realtime realtime?

• One-way media transmission: Streaming

- – Less interactive
- – Play out from a file, not necessarily a capture-device
- – Less sensitive to delays ("almost live")
- – Still have issues with bandwidth and jitter

• Now manage playout buffers through sliding windows

- – Receiver 'pulls' content
- – Fills its buffers as content is played out
  - • (Progress bars on video clients)
  - • If bandwidth is not sufficient, client pauses – or other **magic** happens

Real Time Streaming Protocol (RTSP)

• Establishes a streaming session and negotiates media transport – rtp/rtcp – http

• Looks a lot like HTTP and SIP.

- – OPTIONS (what can you do?)
- – DESCRIBE (what can this file give me?)
- – SETUP (get ready to send one or more streams, over protocol P)
- – PLAY (play, from time T1 to time T2)
- – …and more...

• Over RTP/RTCP can adapt bandwidth, encoding, ...

Or *sigh* use HTTP

Because it gets through **firewalls**

Because it has so many extensions – It's an application and a transport protocol

Use HEAD requests to learn about media and options

Use GET with Range requests

- Download pieces for the playout buffer
- Can ask server to encode adaptively

Note: No server state required! && Inherit HTTP proxies, caches and CDNs!

HTML5 has video player built-in (to kill Flash)

In closing

• Perfect, low-delay, real-time over a best-effort network is really hard

- – Very application-specific.– Brains are adaptive, robots less so.

• All of the smarts is in the end-points– Unless you build circuits (RSVP, QoS, ...)

• Audio/video works 'ok' – In your context...

• Increasing use of real-time traffic for device control – But maybe the Internet isn't the best choice • But maybe it's the only choice

• HTTP - TCP application – (more) reliable bytestreams
– Exchange messages, command/responses, strong client/server relationship
• RTP - UDP applications – short messages, ARQ, low-delays
– Send messages, hope they arrive, weak client/server relationship
• Useful when everything is a sizable computer, with good bandwidth

What is IoT/IoE

• Internet of Things, Internet of Everything
• Independent devices
  – acting on their own,
  – acting collectively
  – Sensors, controllers, ...
  – Smart homes, smart cities
• From the large – Appliances – Vehicles
• To the small – Cover farms/battlefields, distribute across factories
  – Insert into bulk cargo, pipelines
  – Inject into people

Measuring, monitoring, detecting, reacting, ...

• Focus on Machine-to-machine (M2M)
• Engines, industrial equipment, predict failures – Temperatures, pressures, fluid levels
– Noise – and what kind, compare to normal
• Weather, microclimates • Ecological
– Plant health, water quality, chemical/biological agents • Traffic flows
• Presence of (bad) people in a space – face, gait, ... • Presence of (bad) cells in people
• Militaryapplications...

Why is IoT different?

1. Scale: Number of devices
2. Power: Ever smaller devices, doing smart/expensive things
3. Networking: Low power, remote locations, widely distributed
4. Timeliness: May need quick commands, responses
5. Reliability: Challenging – small devices

What's needed?

1. **Scale**
   – No limits on number of devices (addresses) and relationships (connections)
     • N^2 relationships, all storing state? Edges and routers
   – Limit messages to avoid swamping networks
     • 1 billion devices at just 1 byte/sec...
2. **Power**
   – Focus on minimal power needs – solar, batteries(!), RF
     •Reduce transmission power
     •Turn off transmitters, ...
   – Do smart things elsewhere • CPUs = power drain
3. **Networking**

– Limit transmission needs
  Reduce bandwidth/distance/# targets
    Application and Protocol design, compression, heartbeats, Which devices/reports are crucial?
  Take advantage of neighbourly assistance
    Ad hoc mesh networks – needs better protocols, routing, transmission technologies,
  Trade-off: staying awake just to help neighbours?

4. **Timeliness**

– Design accordingly

- Exceptions vs Regular Reports
- Transmitter vs Receiver requirements
- Short messages, prioritised messages

5. **Reliability**
   – Add only where needed
   – Make it lightweight • ARQ (push/pull) – or delegate it

Design: PubSub: Publication/Subscription

• Separate the 'announcement' of data/state from 'consumption'
   – Announcements: really easy
   – Consumption: as flexible as needed • Ask the server: what do you have, ...
   – Allow for any type/number of consumers to subscribe – Allow for any type/number of sources to publish
   – Avoid 'connections'
• Needs a broker (or server)
   – Lightweight, fast, flexible, open, ... • i.e. not a webserver !!!

MQTT

• Was Message-Queueing Telemetry Transport (1990s, IBM) – No longer queues (sort of) and less Telemetry-specific
• Runs over TCP (v3.1)– and (v5, May'18) *UDP*, and ZigBee and ... as **MQTT-SN**
• More scalable, more flexible, more lightweight, better error-reporting •
• A "database" of key/value pairs – That deletes data as fast as it can.
• Standardised by OASIS, not IETF
• Organisation for the Advancement of Structured Information Standards
• Global industry association
• Lots of business-related standards, markup languages, XACML/SAML, PKI, BPN, ...

MQTT for information sharing

• Shared whiteboard for information exchange
   – Publish key=value by writing • No arithmetic operations
   – Subscribe for reading
• HTTP: monitoring by asking, often, for any given X
Various consumers
• MQTT uses messages
   – As information
   – As 'triggers' (by listening clients)
   – So server/broker **pushes** messages
• Concept of 'topics'– Build your own database structure, on the fly!

PubSub - Pub

• You (the sensor) **publish** a **message** to an MQTT broker
   – any (value) type (number, string, file, JSON, …) Can include your own keywords, userId, timestamps, ...
   – to a specific (key) "topic/subtopic/sub-sub-topic/..." – as you want
• "Sensors/Paddock-A/Moisture/Sensor-1" = 93%
• "Sensors/Paddock-B/Temperature/Sensor-3" = 28C
• "PizzaPreferences/HouseMate/Malcolm" = "vegetarian, but no olives"
• You (the consumer) subscribe to a particular topic – No guarantee it exists! It may exist later...
• or to a filtered-set (wildcard) of topics – (using # and +) – All temperature readings– All sensors from a given area
• Sensors/Paddock-A/Moisture/Sensor-1 (sensors/location/type/ID)
• Sensors/# (all sensors, anywhere)
• Sensors/+/Moisture/+ (all moisture sensors, anywhere) "+" means From given area
• Sensors/Paddock-A/# (all sensors, in Paddock-A)

Magic Topic(s)

• $SYS/#
• Holds very useful system information about the broker itself – Only broker publishes here
• Load, clients, bandwidth, storage, etc.
• Unfortunately:– Fields are not standardised (across brokers) – Resolution can be **low** (implementation-specific – 60sec?)
• Unwise to use MQTT to monitor health of your MQTT server?

MQTT packets

Runs over TCP (up to v3.x), can also run over UDP and others (v5 onwards)
• Very bit-oriented
– Very condensed messages, no plain English – Minimise load on publisher and network

MQTT Messages

• 16 Messages types
  – Connect and Disconnect (and Ack)
    • Establish a channel and server state, and (you) identify yourself (lightweight security)
  – Ping request, and response • Server level, not ICMP !!!!
  – Publish, and Subscribe, Unsubscribe
    • Publish actually used both source->server and server->subscriber
– Publish-Ack/Received/Released/Complete (various QoS guarantees)
– Subscribe-Ack, Unsubscribe-Ack

Main MQTT rule: minimalism

• Server wants to maintain the minimal possible amount of state – Subscribers: “Temporarily unreachable” or “no-longer interested”?
• Server does not ‘queue’ messages
  – Once messages are pushed to all subscribers, they are deleted (*)
  – Published messages for topics with no subscribers are deleted (*)
• Give the server every chance to clean up its database
• (*) means mostly…

QoS

• “Quality of Service”
• What guarantees can you give me about this service? • Is it timely, reliable, accurate, trustworthy, ...
• The Internet can have QoS features – MPLS, DiffServ, RSVP, CoS, ...
• MQTT has QoS at the application level
  – Because some subscribers need to be sure
  – Because subscribers can join at any time
  – Is that power off, is that gate open, how full is the tank now, ...

MQTT Quality of Service (QoS)

• Three levels (0, 1, 2)
– Each with more load, storage, bandwidth, energy implications
• *Level 0*: (default)
– “**Fire-and-Forget**”
– Client/Server pushes a message out, then deletes it.
• *Level 1*:
– “**At least once**”
– Guaranteed delivery, requires confirmation, but could transmit duplicates
• *Level 2*:
– “**Exactly once**”

– Guaranteed delivery, 4-step handshake, with no duplicates • Lots of energy, time, storage, ...

MQTT QoS is part of the SUBSCRIBE/PUBLISH set up – Which
is used source->server and server->subscriber
• Can have reliable publishing turn to unreliable delivery – And vice versa?
this is for every single subscriber (PUBLISH QOS0)

MQTT – "Last Known Good"

• Sensors that rarely report,
  – E.g. state changes: door open/close
  – E.g. very remote, low-power, "expensive" sensors
• Need to give new subscribers something to start from – Could be waiting a long time...
• "**Retain**" flag
  – Retained by server
  – Even across reboots
  – Sent on first subscription request by client

MQTT – "Last Will and Testament"

• Sources can publish a default message for any topic(s) (e.g. client/status) – A **retained** message, but not sent immediately
• If server e.g.
  – Does not receive a published message after <KeepAlive> period, or
  – Sees (TCP) connection dropped without (MQTT) disconnect, then
  – Assumes connection is lost, source has failed, ...
• Can then inform subscribers (new and old)
– E.g. "Service temporarily/permanently down/redirected; contact x@y.com"
• QoS considerations. KeepAlive considerations.

MQTT – "Clean Session"?

• Flagged on connection
• **Clean**: Pretend I'm brand new
• **Not clean:** Ask server to remember you – aka Persistent session – In case you drop off
• Pain for the server:
  – Store all your subscribed topics
  – Store all messages (with QoS 1 and 2)
  – Push in burst when reconnected – give me everything I've missed

MQTT in a smart home

• Attach sensors
  – Brightness, temperature, humidity, movement, voice, locks, ...
  – Each publishes to a state topic
• Sensors/Temperature/Lounge = 18 • Sensors/LightLevel/Lounge = 10%
• Allows you to monitor the environment – Lots of charts over time

MQTT in a smart home

• Attach **controllable** devices
  – Lights, heaters, coolers, curtains, locks, AV system, ...
  – **Each device subscribes to a command/state topic that you write to**

• Lights/Lounge/Light-27 = Off [On, 10%, 50%] • Heater/Lounge = Off [On]

• Attach **controllers**
  – Physical switches, web-client, app, Alexa/Google/Siri, ... (at the same time!)
  – Each publishes to a command/state topic
    • Switches/Lounge/Switch-19 = On [Off] • Thermostat/Lounge = 22
• Note: **controllers** are not directly publishing to the controlled
  – Gives you way more flexibility
  – Able to modify behaviours all in software, no hardware changes needed
• Connect topics by a **Rules Engine**
  – Given X (is published), do Y (publish something) [e.g. NodeRed, IFTTT.com]
• Overseen by a **State Machine**
  – Store state, Note changes, Combine rules, Create scenes [e.g. OpenHAB] – Bring in extra information (time of day, weather forecast, ...)

MQTT Security?

• **It has some!** (if enabled...) • Username/password
• Client identifier (64kB!)
• Role-based Access Control • X509 Certificates
• Payload signatures, integrity checking • **Encrypted** connections

IoT security?

• **Ha!**
  • **Firmware v1.0** – 5 years later?
  • Standard admin login?
  • Standard access URLs? (google-able)
  • Cloud gateways?
• Web-cameras, baby monitors, powerboards, ...
• Take over for
  – local attack (home and home network)
  – external attack (Denial-of-Service/flooding) DoS

Routing: Packet Forwarding and Routing
Separation of control plane and data plane - globally

Back into the network layer

- Distinction between forwarding and routing – Local decisions vs global decisions – Given a network with multiple paths/interfaces, which one do you send to?
- Focus on unicast routing – Opposed to broadcast, multicast, any-cast routing

Spanning Tree = routing?

An wide-area view that spans a network
– Removes loops, establishes reliable paths
– But only runs at layer-2 (single-technology), and doesn't scale
– Wastes paths • No measure of 'quality' of a path • Doesn't use redundant paths when beneficial

From local to global

• Locally find devices via ARP, but that doesn't scale.
• But then what on the WAN?
– LAN to WAN, a single default route? • Can have backup paths
– Routers advertise a prefix (subnet) – aggregation!
– Need to go from 'enterprise' networks up to global scale

Global routing is hard

- "Routing table" sizes – growing (1M+)
- Updates – growing (170k/day)
- Computing forwarding tables – growing
- Routers – used to be simple computers
  – 100Gbps = one small IP packet every 5 nanoseconds.
  – "Performing a lookup into a data structure of around one million entries for an imprecise match of a 32 -bit value within 5 nanoseconds represents an **extremely challenging** silicon design problem.

"Routing" at different timescales

- Routing = sending some traffic over some path at some time
  – It's allocating bandwidth across the network
  – While adapting to requirements and changing conditions

What you are doing	How quickly	Because
Forwarding/ "load-sensitive" routing	Seconds	Bursts, Congestion
Routing	Minutes	Changes, failures
Traffic Engineering	Hours	Long-term load
Provisioning (Provide)	Month	Customer demand

Expectations

Routing has to work "right", all the time

Expect	Because
Correctness	It has to get packets from A to B
Efficiency	Use available bandwidth well
Fairness	Don't ignore capable network elements

Convergence	Recover quickly from any disturbances
Scalability	Copes with increasingly large and complex networks

Routing context – ideally:

• Decentralised, no controller, no hub – Up to a point...
• All nodes (routers) are alike
– Speak the same language, run the same algorithm, at the same time
• Learn through message exchanges with “neighbours”
• Need to deal with router, link and message failures

What is the ‘best’ route?

• depends on – Latency (delay = distance) – Bandwidth (slow)– Cost (money) – Hops (forwarding delays)
• For a fixed topology: – Ignores link congestion & router load

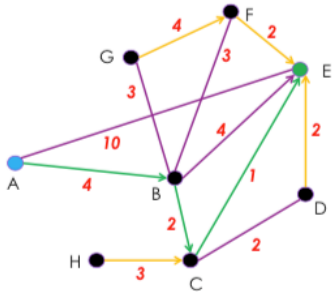
Shortest-path routing(“lowest-cost” path routing)

Associate some **cost** with each link
– You choose: $$, ms, hops, bps, ...– In each direction – can be asymmetric
Add up the total end-to-end And try to minimise the total – If tied, pick one
Optimality property:Sub-paths of the shortest path are themselves shortest paths
Dijkstra’ algorithm

Sink (and source) trees

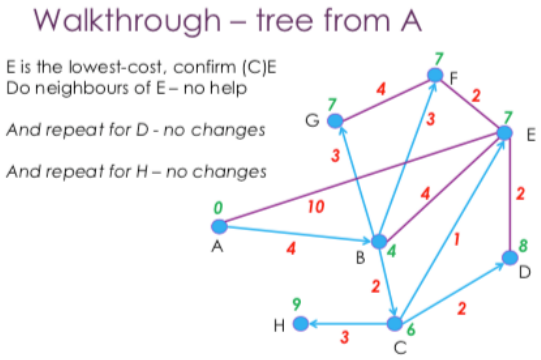
• Union of all shortest paths towards a node from each source
• Consider E’s sink tree. – ABCE = shortest for A, B, C – Work out the rest
Source tree = sink tree <Usually, but doesn’t have to>
• Asymmetric costs => different trees • What happens if BC=2, CB=5?
Regardless of where you start, routing decisions only considers **destination**
– Source is irrelevant (*) A,B,H get to C–and then to E

• Each node only needs to know A **next hop on the optimal path** => forwarding table
• Forwarding table = Next hop for every destination



Algorithm outline:

– Start at source node, mark all others as ‘tentative’
– Give source “0” node-cost, everyone else is “infinite” cost
– Loop: while (tentative nodes)
• Identify lowest-cost node, confirm it
• Add link to source tree
• Modify (‘relax’) other costs by distances you now know



Dijkstra

• Works out from the source
– And you need to repeat for every source
• Leverages optimality property
– Use sub-paths to build longer shortest-paths

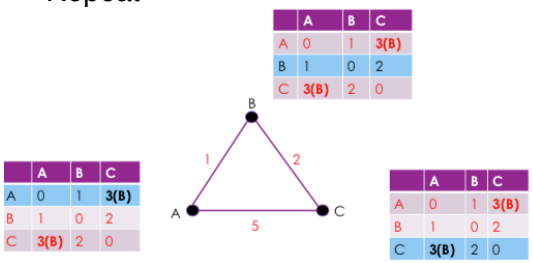
• Has some scaling issues in complex networks
– Imagine 1000+ nodes, 1000’s links
– Imagine changes at any single point
– Lots of research...
• Needs complete topology – At each node/ source

Distance Vector routing

• When you don't know the topology...
  – Calculate Source tree in a distributed fashion – Looks a bit like Spanning Tree
• Nodes only know costs to neighbours
• Nodes only talk to neighbours
  • Nodes all run the same algorithm
  • Nodes/links may fail or lose messages

DV algorithm: Distributed Bellman-Ford

One of two major approaches for routing protocols – Link-state is the other one
Early routing approach (Routing Information Protocol (RIP), 1988)
  – Simple to use, but slow to converge, and somewhat fragile – still improving
Each node stores a vector of distances, and next hops, to all destinations
  – Initially vector has 0 cost to self, infinity to all others
  – Send vector to neighbours
  – Update for each destination with lowest cost heard, adding cost of link
  – Repeat



Adding routes– One hop wider awareness for every message exchange
Deleting routes:– Deliberately or due to failures & Drop out of vectors, other nodes delete
One small problem– Count to infinity...– When a particular piece of the network falls off.

Count to infinity

• Good news travels quickly, Bad news travels slowly
Normally everyone else (C, D) sees a path to A **through B**
  – When **A** falls off, **B** sees a path to **A** through **C**, which sees a path through B...
• Deal with problem via "poison reverse", "split horizon"
  – Don't advertise route to the node you learnt it from
  – These don't scale well in certain circumstances

Link state routing

• The other, more common routing algorithm • More computation, but better behaviours
  – Scales well to enterprise networks, though not globally
• Used in
  – Open Shortest Path First (OSPF),
  – Intermediate System to Intermediate System (IS-IS)
• Same baseline rules for a federated environment
  – Only talk to neighbours
  – Only know their costs
  – don't know the topology (to start with)
  – Deal with node/link/message failures

Link state routing

• 3 parts:
  – **Flood** the network
  – **Learn** the topology
  – **Compute** tables with Dijkstra
  – And repeat every time there's a change.

Flooding?

• Broadcast incoming (broadcast) message to all (other) outbound interfaces
  – Yes, it's bad. – Unless it isn't.

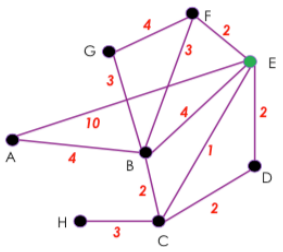
- Just keep track so you don't repeat yourself, or cause storms
  – Use incrementing sequence numbers – May get multiple copies, deal with it

- Ironically, if you miss a message, use ARQ

Link State flooding

- EACH NODE FLOODS
  link-state-packet (LSP)
  – to the entire
  reachable network

Node E LSP (+ some Seq #)	
A	10
B	4
C	1
D	2
F	2



Link-state topology analysis

Listen for LSPs, learn the topology
Then run Dijkstra locally – On each node! – Wasteful
  replicated communication/computation,
  Lots of CPU grunt needed
But it's effective.
On changes (node/link failures) the neighbours: – Detect a link down, or lack of heartbeat packets
– flood updated LSP
– and everybody recomputes
Various (rare) failure modes (flooding fails, node flaps, seq# errors, races, ...) – Manage by ageing LSPs and timeouts

Expectation	Distance Vector	Link State
Correctness	Distributed Bellman-Ford	Replicated Dijkstra
Efficiency	Reasonable – shortest path	Reasonable – shortest path
Fairness	Reasonable – shortest path	Reasonable – shortest path
Convergence	Slow... many exchanges	Fast – flood and compute
Scalability	Excellent – storage/compute	Ok – storage/compute

Equal-cost multipath routing (ECMP)

- Not a protocol/algorithm, but an extension for flexibility
- Allow for multiple paths for packets between source and destination
  – Greater redundancy – Improve performance • Capacity increase• Load balance
- Need to– Detect them, and Forward traffic along them

ECMP detection

One approach: don't tiebreak
Allow shortest path to be a set
  – Rather than a single choice
Not a tree now but a directed acyclic graph

ECMP forwarding

Forwarding tables now have a set of *interfaces* for each destination

(1)Allocate each packet randomly?
  – **Good** for load balancing,
  – **Bad** for jitter (packet delay variation)
(2)Allocate by 'relationship'
  – Use the **destination** and **source IP#**
- E.g. E chooses: F-H goes EC, E-H goes EDC
- Equal cost, and consistent performance

(3)Allocate by 'flow'
  – Using flow identifiers (IPv6)
Less balanced, but more predictable

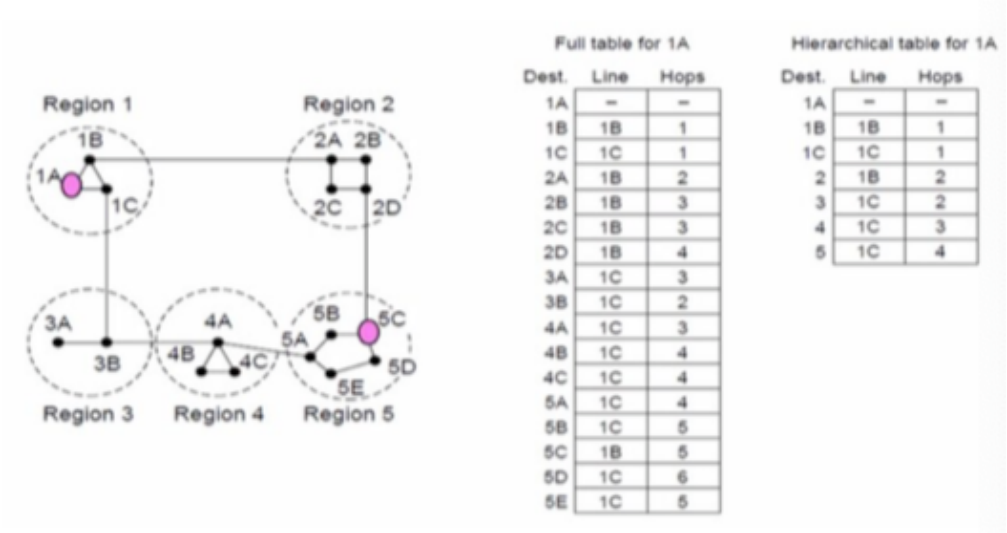
Hierarchical routing

- Scaling problems as identified earlier
  - – Routing tables growing
  - – Routing computing growing
  - – Forwarding tables growing
- Network aggregation
  - – Already have LAN prefixes aggregating a whole subnet
    - • Don't need to advertise every single host on your LAN
  - – Can treat a group of subnets as a larger subnet • E.g. adjacent /24s within a /16 (150.203.aaa.bbb) • But not all subnets now are 'adjacent'
  - – What about geographical aggregation?

Routing to a region

• Aggregate nodes/subnets – Hide internal complexity
– Shorter tables
• Downside:
– Less optimal paths
– Full 1A to 5C[1B] = 5hops – Hier.1A to 5C[1C] = 6hops

• Outside of a region, routers get told one route to get to that region – All hosts are aggregated into a smaller table,
– Reducing communication and computation
• There can still be more than one route into or out of a region
– It's still a local router decision how to get in/out for a particular region – A region sets a context, and designates some border routers
– Within a region, we make our own (excellent) arrangements



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Policy-based routing – and routing policies

• At the heart of the Internet
– Multiple ISPs, interconnecting via Internet Exchange Points (IXP) – All running a business. Or a country.

Policy routing

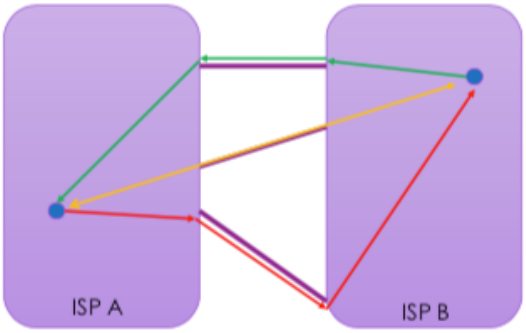
• Already have dynamic, complex, large routing databases and algorithms
• Now Introduce human needs: Layers 8+ – Money – Politics – Security – Religion

• i.e. we have POLICIES to add to our protocols
– E.g. National Research and Education Networks have an R&E traffic policy
• Wholesale purchase **AND** don't compete with commercial providers **AND** social good

Shortest path is a local priority...

• E.g. each ISP policy: offload as quickly as possible
Technical Term: • **Hot Potato Routing**

- Sub-optimal shortest path
- Asymmetric paths!
- Hierarchy is (consciously) broken, for good business reasons



Most common policies: Transiting ISPs

- Take your traffic and pass it through their network to the Internet
- They take the Internet traffic and pass it through to you
- And you pay them.

Common policies: Peering ISPs

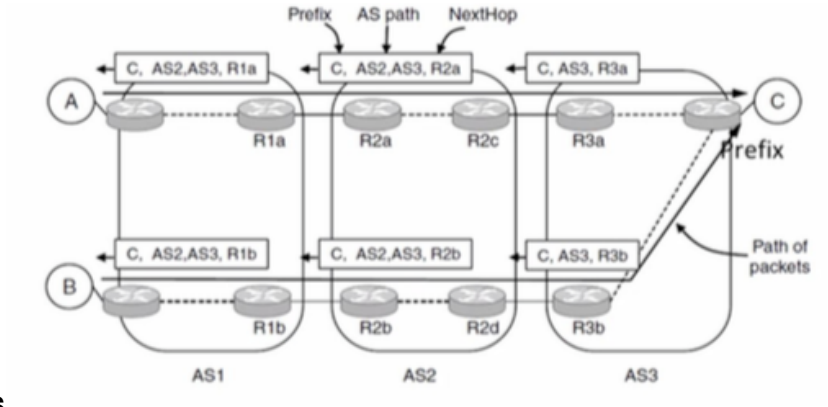
- Take your traffic and pass it through to the other network
- They take the other networks traffic and pass it through to you
- You cannot reach the Internet through them.
- Mutual benefit
  • No money exchanged
  • a CDN, or a cloud provider, or an NREN, or..

Border Gateway Protocol (BGP)

• The main Internet routing protocol today
• Key concepts:
Separation of interior routing protocols and exterior routing protocols – Intradomain vs Interdomain (Enterprise vs International)
• Identifies Border Routers (or Gateways) which run BGP
– Creates an edge between interior and exterior routing
• Aggregates nodes within an 'Autonomous System' (AS) – Think a region, a business, an ISP

BGP is more DV than LS

Instead of Distance Vector it is a **Path Vector**
• Announcements:
– IP Prefix, Next Hop
– Path: list of AS's to transit:allow loops to be detected and removed • No distance indications



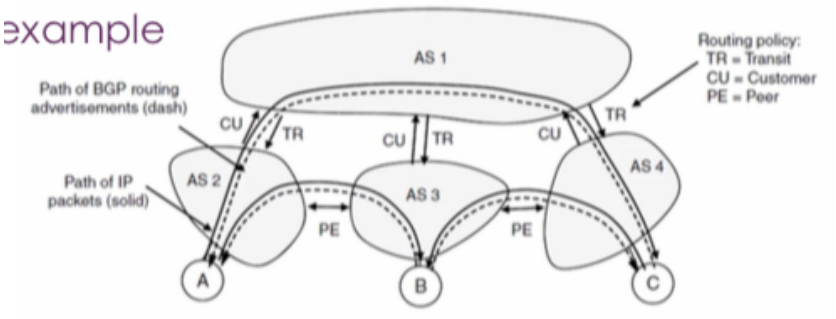
BGP route advertisements

Policy implementation

BGP allows you to configure your route “advertisements”
Border routers advertise available paths
– with policy constraints
– Only to those AS’s that may use them
– And filter out those they cannot use
– E.g. offer transit to some, peering to others – E.g. offer a faster path to some, slower to others ($$$)
Border routers listen for available paths
– And (given a choice) pick the one that suits them – for any reason!
• Shortest, cheapest, friendliest, safest, politically/contractually-suitable, ...
• Human rather than ‘technical’ optimisation

BGP example

Various businesses here: – AS1 is selling transit to AS2,3,4– AS2 and AS3 are peering (ditto AS3 and AS4) – AS2 is selling transit to customer A



Various advertisements here:
– Customer: **[A, (AS2), router2U]** is sent by AS2 to AS1
– Transit: **[B,(AS1,AS3), router1L]**and **[C,(AS1,AS4), router1L]** is sent by AS1 to AS2
– Peer: **[B,(AS3), router3L]** is sent by AS3 to AS2, **[A,(AS2), router2R]** is sent by AS2 to AS3

So AS2 (and hence customer A)
– Hears one option for reaching **C**: (AS1, AS4)
– Hears two options for reaching **B**: Transit(AS1,AS3) and Peer(AS3) • And peering traffic is usually free...

In closing
• Routing is complicated and hard – this has been a very high-level view! – DV, LS and BGP are very important

• Internet is large and complex
• Policies are an important factor – the internet is also a business
• **Connecting** interior and exterior routing/gateway protocols – Literally an edge case. Haven't even discussed it
• Performance is challenging
– Scalability, convergence, reliability, trustworthiness, optimisation, ... – All in a (globally) distributed system

Remember (TCP) Sliding Windows?

- Want reliability and throughput – and fill pipes!
- Start with ARQ – stop-and-wait
  – Single segment outstanding = problem on high bandwidth*delay networks
- Say one way delay=50ms so r**ound-trip-time (RTT)**=2d=100ms
- **Single segment per RTT = 10 packets/s**
  Typical packet on Ethernet? Say 1000 bytes = ~10,000 bits -> 100kb/s or 10% of link • Even if bandwidth goes up, throughput doesn't!

Sliding Windows

• Allow W segments to be 'outstanding' (unACKed) per RTT – Fill a pipeline with segments
Set up a 'window' of W segments – **per connection**
• W=2*Bandwidth*delay
• At 100Mb/s, delay=50ms means W=10Mb
– and assuming same 10kb segments, W=1000 segments – 500 are on their way out there!
Sender buffers up W segments until they are ACKed
Window not full, so send a packet; Packet ACKed, so Window not full.

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, *I just don't care*
• Let sender retransmit what I'm actually waiting for
• **Sender** has a single timer. After timeout, resend • Really simple, but somewhat inefficient

ARQ – "Selective Repeat"

• **Receiver** buffers many segments – Reduce retransmissions
• ACK what has been received in order
• And also ACK segments that haven't – Any gaps indicates missing segment!
– SelectiveACK(SACK)
• **Sender** has a timer **per unACKed-segment** – As each timer expires,resend that segment
• Way more efficient,now widespread

Very sender/network oriented

- **Sender** manages the transmission
  – UDP – send-and-forget, no control
  – TCP - Slows down waiting for ACKs
  – Optimised to keep network full
  – What about the receiver application?
- Consider **Receiver** being **swamped**
  – HD video streaming to small device – it(receiver) needs to control the flow

Flow Control:Sliding Windows on the Receiver side

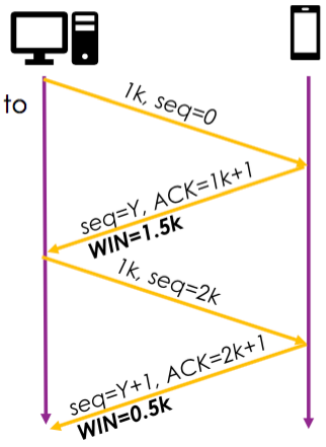
• Transport layer:
  – receives the segment from the network – and adds it to application buffer
• Application calls recv(N-bytes) to read from buffer
  – But what happens if the application is slow?
More segments arrive, fill (*TCP*) buffer – and eventually application recv()s
TCP Sender Sliding Window (**W**) – Both sides know
• Receiver Sliding Window = Flow Control Window (WIN) – Number of "ACCEPTABLE" segments to be sent
Sender gets told **WIN** and uses lower of **W** + **WIN** as the 'effective' window

Sequence numbers identify where sender is up to

Acknowledgements where receiver is up to

But receiver can also report **buffer** available

Simple Flow Control



Congestion

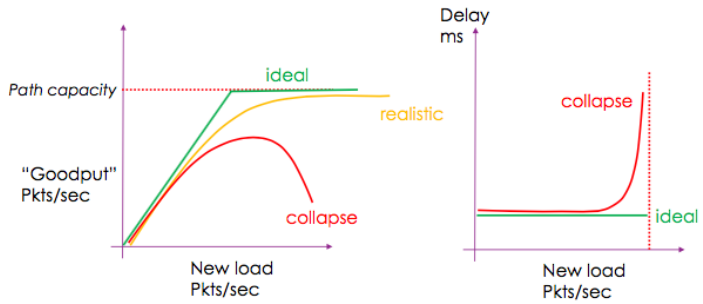
- A traffic jam – something filled up and is holding up the rest
  - A dynamic condition
  - Somewhere (unknown) along the (unknown) path
- Senders keep sending
  - – Makes it worse, for themselves, and everybody else – Congestion -> loss
- It is not the links that "cause" loss (by being congested) – They run at a specific clock, bits in/bits out.
  - – They set the limits

Too many inputs for the one output

Router Buffers: Queues...

- FIFO (First in, First out) queues on every interface
- Great for absorbing (short) bursts of traffic
  - – Data-rate in > data-rate out
- For a while... then queue overflows, and packets get dropped
- Largely driven by traffic patterns
  - – Multiple conversations randomly sending to the same path at the same time
  - – Assuming similar bandwidth links in/out

Congestion Effects



Why???

- Rising load fills buffers – Delays go up
- Overflowing buffers drop packets – Loss rises
- What do the receivers do? – ASK FOR RETRANSMISSION
- What do the senders to? – RETRANSMIT
- Network fills with retransmitted packets, new packets are held back – Goodput goes to zero

Managing capacity

• Want to operate (just) below congestion damage – Use the network to nearly “capacity”
• Need to allocate total capacity:
• **Efficiently**: get as much as I can, without causing congestion and
• **Fairly**: everyone gets a reasonable share

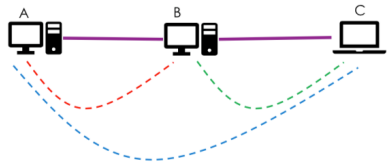
Who handles that?

• To be effective, both **Transport** and **Network** layers have a role
• Network layer (IP) **sees** congestion – It’s happening in the routers’ buffers – And it could provide feedback
• Transport layer causes congestion! – But can’t see where.– It can back off on transmissions

Isn’t it statistical multiplexing?

• Could allow all senders just to fight it out – eventually it’s even?
  – The very big and the very small
  – Problem: in congestion everybody loses.
• This is hard:
  – Different applications have different behaviours : cat video Vs. security sensor
  – Load is constantly changing(time)
  – Congestion may be happening in multiple, different, places (space)
  – There is no central view (everyone’s blind)
• Need to find solution(s) where:

– Senders adapt concurrently and continuously? – We can make it *efficient* and *fair*?



Efficient...	
AB	1
BC	1
AC	0
Total: 2.0	

Fair...	
AB	0.5
BC	0.5
AC	0.5
Total: 1.5	

“Equal per flow” fairness?

• AC uses twice the network of AB, BC – is that fair?
• Exact fairness is hard. Avoiding full starvation (AC=0) is more important
– Some starvation might be ok...

Network bottlenecks – unequal paths

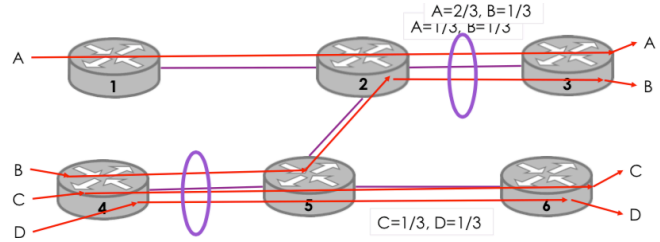
• AC is choked by A-B link. BC is choked by B-C link • So now what’s fair?

Max-Min Fairness

• Allocating bandwidth such that :
• “increasing the rate of one flow will decrease the rate of a smaller flow”
– “Maximising the minimum” – keep adding, and sharing what’s left.

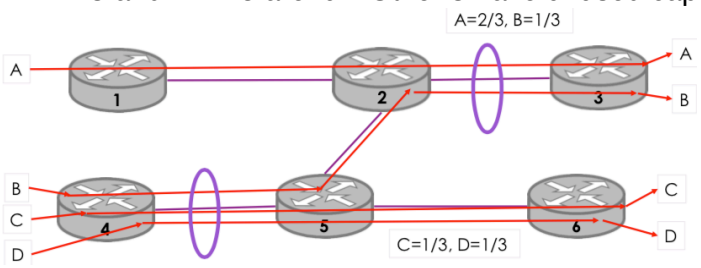
Start from zero. Increase bandwidth of A,B,C,D till something bottlenecks

– R4-R5 fills at 1/3 each for B,C,D. Hold them down. What’s left to raise?– A…can go up to 2/3, on R2-R3



SoA=2/3.B,C,D=1/3

• R2-R3 and R4-R5 are full. Other 3 have unused capacity.



So how to adapt?

Open/Closed loop

– Open: reserve a circuit ahead of time – Closed: adjust on feedback

Host or Network driven

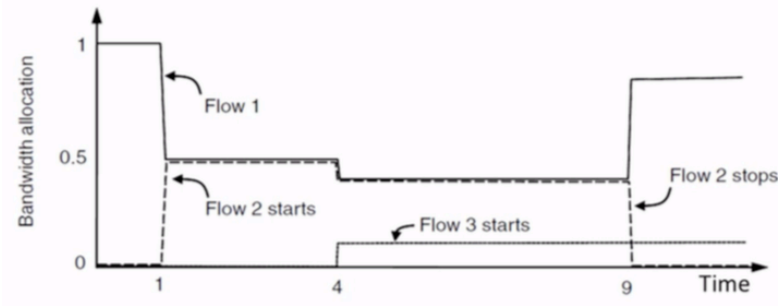
– Host manages the allocation (use)– Network policing is strong, but inflexible

And "allocate" bandwidth: Rate based or Window based

– Tell application to send at a specific rate, – or To watch window sizes

• TCP is Closed-Loop, Host-Driven, Window-Based

And adapt over time



Two layers, working together

• Network layer (IP) provides feedback on allocation? – Actually, it indicates congestion

• Transport layer (TCP) modifies sender behaviour

– TCP window sizes get adjusted – Dynamically, in response – This is a 'control law'

• *Additive Increase, Multiplicative Decrease (AIMD)*

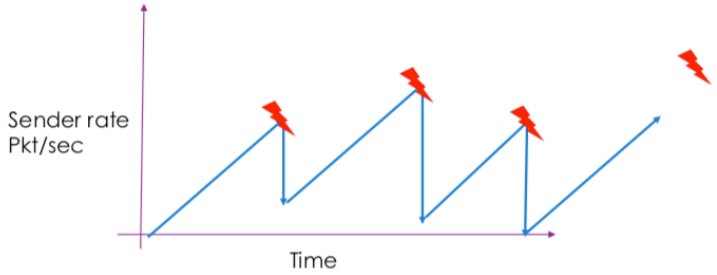
– Senders additively increase rate, while no congestion (gently, gently)

– Senders multiplicatively decrease rate when there is congestion (quickly, quickly!)

AIMD Sawtooth

• Slowly increase to probe the network – Multiple small steps that add to the rate

• Quickly decrease to avoid congestion collapse – Single(+) large percentage decrease



Nice features

• Converges to a fair and efficient allocation when all hosts run it – And everyone(*) does, with some parameter variations– Doesn't care about the topology

• Works effectively compared to other control laws

– Slow decrease=bad, fast increase=bad, both slow=bad, both fast=bad

• Just needs a **single** signal from the "network" (actually, receiver) – Path is congested, or not.
How does the network signal the sender?
Remember – multiple TCP implementations, by OS and date and ...

Signal	Pros/Cons?
Packet loss	• Really obvious • Don't detect congestion till it happens
Packet delays	• Detect congestion earlier • Detection is more inferred than actual
Router signal Explicit Congestion Notification (ECN)	• Detect congestion earlier • Needs the affected router and hosts to support it

Implementing AIMD

• What are the best numbers for increase/decrease?
• Several components in TCP contribute – let's focus on a few. • Start with ACK clocking...

ACK clocking process

• High-speed link, talking to low-speed (or congested) link
• Sender sends a burst of packets to destination (to router with big buffer) – Doesn't know any better!
• Packets get buffered, and Low-speed link takes longer – packets get 'longer'
• ACKs returned at rate of slowest link! • Sender learns to back off
• Sender matches ACK rate. Buffers can drain – congestion avoided
• Bursty traffic has become a smoother stream
• And we get a new measure – the '**Congestion Window**' (CWND) – Smaller than W (=2*B*delay). [and not related to Flow-control Window (WIN)]

Getting started

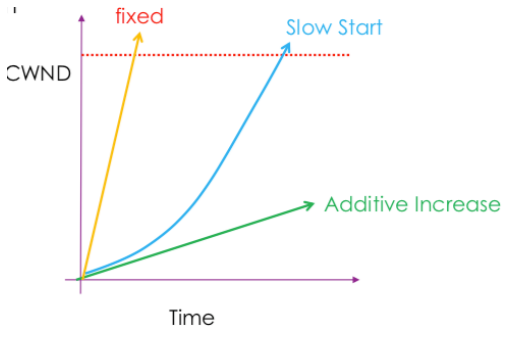
• On initial TCP connection, what is CWND?
– Guess? Too many variables (bandwidth, delay, congestion, ...) – Pick something? Could be way under, or over.
• TCP Additive Increase (on start):
– Start with CWND of N bytes (~1 packet).
– Every round trip without loss, make CWND bigger by 1 packet
• Increase very gently, but it could be a long time to reach the ideal CWND – Whatever it currently is...
• Want an algorithm for TCP CWND growth to **start** a bit **faster** - and it's called...

TCP Slow-Start...

Instead of adding, double CWND every RTT (1,2,4,8,...)
• Start slow, but quickly reach high

Slow-start overshoot

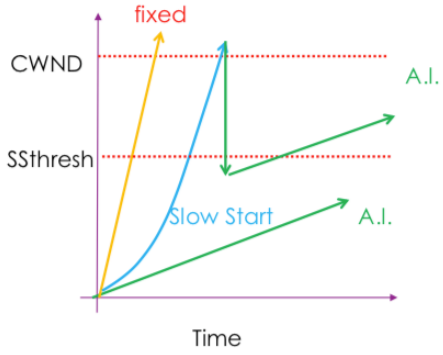
• Get to the right CWND more quickly
• But will still go (suddenly) over it
– Get packet loss/feedback
– Multiplicative decrease (big drop in CWND)
• So combine Slow-Start with Additive Increase:
– Initial connection, get MD'd down. Below right CWND, but still close? – Define a threshold: ss-thresh = 1⁄2 * CWND(@loss)
– Stop doubling, start adding

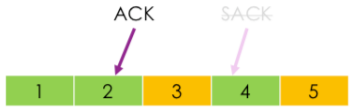


Combined behaviour

- After the first overshoot...
- Start with slow-start
- Move to A.I. phase
- Gets you there quicker
- Keeps you there longer – Within that good Ssthresh/CWND band

– Trying to maximise performance, politely.





Fast Retransmit

• Loss -> timeouts

• If timeout is too long, lose ACK clock

– Start all over again, with a CWND's of packets out. – Slow start (CWND=1) then additive increase - ugh

• Recall ACKs (**Seq#**) are cumulative, sequential

– If packet is lost, but later ones arrive, receiver sends a duplicate ACK

– "New data arrived, but it wasn't the next segment" • Probably the next segment is lost

– *Third duplicate ACK* triggers a resend of **Seq#+1** (lost?) segment: Fast Retransmit • Hopefully repairs the single-segment loss quickly• And ACK Seq# catch up with what's been sent before loss?

Fast Recovery

• Had loss, so still need to multiplicative-decrease the CWND

• Also have to wait for receiver to tell you where it's Seq# is up to.

• Hang on:

– Additional (duplicate) ACKs are arriving = receiver got more segments • Probably the next one(s)!

• And they maintain the ACK clock

– Take a chance: advance the sliding window as if everything is ok (count ACKs)

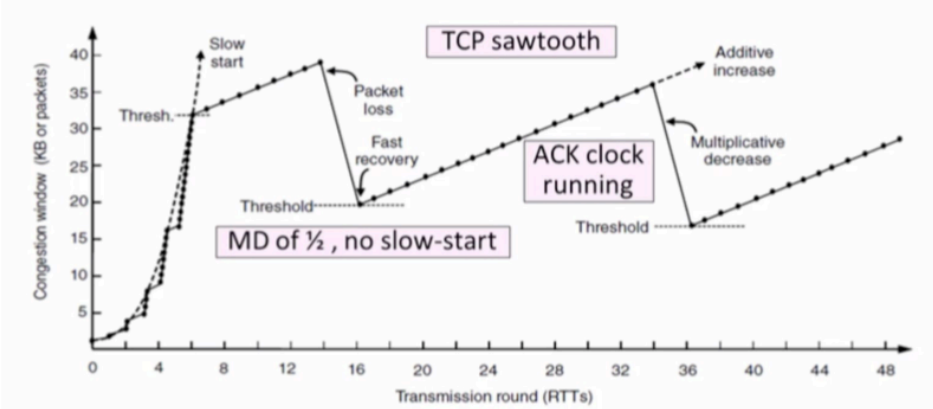
• MD the CWND (1⁄2 it!) and then continue sending (**Fast Recovery**)

– Somewhat slower, but hopefully little loss, and no re-start.

- Receiver will sort things out and let you know (ACK)

TCP Reno

- Fairly common TCP codebase (1990s)



And beyond?

• TCP Reno
  – Can repair one loss per RTT
  – Multiple losses = timeout = (slow) start all over
• TCP NewReno
  – Better ACK analysis
  – Can repair multiple losses per RTT
• TCP SACK
  – Far better!
  – Receiver sends ACK ranges (set) – sender can retransmit without guessing

Can routers help?

• **Explicit Congestion Notification**
  – Still being deployed (routers and hosts) – only standardised in 2001...
• TCP drives network to congestion, then backs off – Prefer to detect congestion (well) before it happens
• Really simple, with in-band signalling
  1. Router notices queues getting full
  2. Marks packets in queue (ECN “congestion looming” – IP header)
  3. Forwards on to receiver
  4. Receiver marks TCP segments sent back to Sender(ACK or normal)
  5. Sender notices, and backs down(MD of CWND)
  6. No additional packets needed!

All “managing” packets randomly running through a network – Non-trivial...

Performance over time : • Capacity planning • Outages • Patterns
Performance at a moment: Network status

Network feedback

• **ECN** – Explicit Congestion Notification
• **ICMP** – Internet Control Management Protocols
– Used passively and actively (ping, traceroute, ...)
• **TCP ACK**nowledgements
• **Application** measures
•...
• No unified view
• No aggregated view, in space or time

Two domains

• Within your administrative domain (interior)
– You have authority
– Get information from everywhere on your network – Put some software on each device
– Probe, measure, scan, ...
• Beyond your administrative domain (exterior)
– No authority (Except maybe a contract?)
– Ask somebody else to put some software on each device, and share

Simple Network Management Protocol (SNMP)

- Design requirements: We want...
- Reach everywhere – All sizes, types of devices
  Switches, routers, access points, printers, servers, ...
  Support devices that are too small, too simple, too hard, too old, ...
- Lightweight– no interference on device
- Operate when things are under stress
  – Identify what is struggling/failing, and when
  – Help to fix/improve things
- Scale to large number of devices and parameters
  – Global naming, delegated, vendor-independent, extensible
- Provide both queries/response and command/control
- And add some trivial security and upgrade it much, much later

SNMP

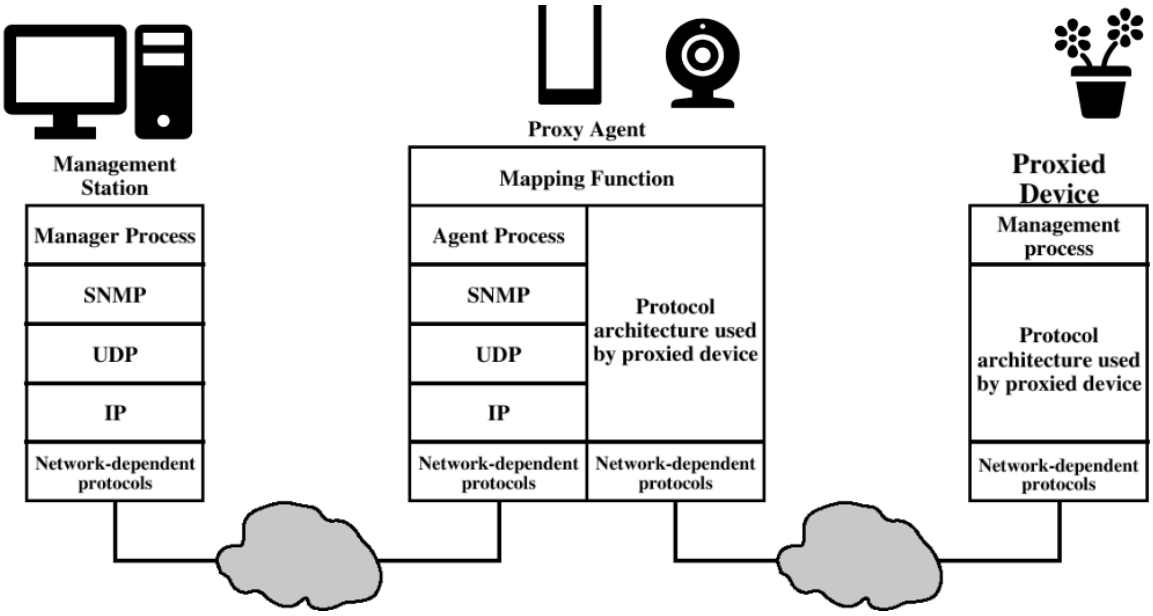
• An application framework
• For managing/monitoring network resources
• Components of SNMP:
– SNMP agents (software on the equipment)
maintains configuration and current state in a database.
Proxies: an agent that talks with non-SNMP devices
– SNMP managers (application that contacts an agent)
to query or modify the database at the agent.
Part of Network Management Systems (NMS)
– Management information bases (MIBs) (describes the database)
MIB, MIB-II (RFC 1213) – and millions more
Structure of Mgmt Info (SMI) defines sets of related objects in a MIB
– SNMP protocol itself
SNMPv1, v2(*), v3
Manager send SNMP protocol messages to all SNMP agents

Information design for lightweight SNMP agents

- No rates, no calculations
- No absolute clocks
- No history
- Just
  – Counters and gauges, – Time since s tart-up – Strings,Identifiers

- "Time ticks",in1/100ths sec.
- Command/control through variable setting

SNMP Proxies



SNMP messages

• SNMP/UDP is connectionless – Use a request ID to maintain a session
• SNMP messages are 'protocol data units' (PDUs) – Different versions of SNMP use the same PDU for different messages (We're still living through that pain..)
• Messages have particular capabilities (SNMPv1):

Get – the value of a object from an agent
Set – the value of a object from an agent
Notify – a manager that the agent has had an event

SNMP(v1) Protocol

On-demand:

Get-request: Request the values of one/several objects
Get-next-request. Requests the value of the "next" object.
Set-request. Modify the value of one or more objects
Get-response. Agent response to a request.

Triggered: **Trap:** A notification from an agent to a manager, some event at the agent.

Traps

• Traps are sent asynchronously by an agent to a manager
• 6 core traps:
linkDown: An interface went down
linkUp: An interface came up
coldStart: Unexpected restart (system crash)
warmStart: Expected restart (manual reboot)
AuthenticationFailure: Somebody tried to query, but ...
egpNeighbourLoss: Link is up but my neighbour has gone
• And ~2^32 others (vendor specific)

SNMP community

• SNMPv1defines"communities"– specify access to specific variable sets – read-write, readonly, none
• Each SNMP message includes community name – Like a password – Unencrypted!!

• Typical values: – Read-only:“Public” – Read-write:“Private”
• Slight enhancement: agent/manager relationship – IP address of permitted managers, stored on agent
• First thing fixed in v2...

SNMP Versions

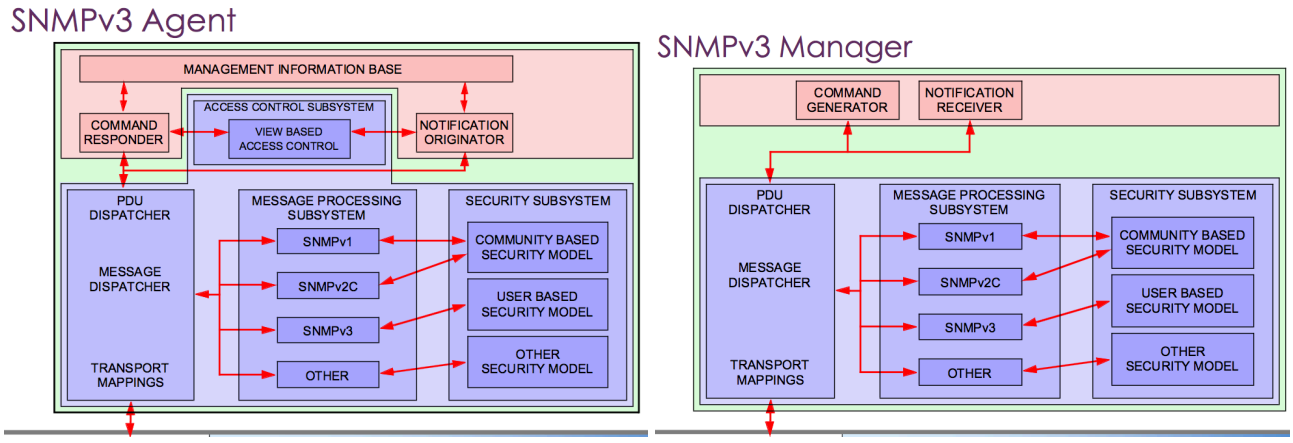
• Three versions in use today:
  – SNMPv1 (1990)
  – SNMPv2c – [and three more] (1996)
    • Adds “GetBulk” function
    • Adds federated monitoring capabilities (manager to manager)
    • Adds TCP transport option
    • Adds 64bit counters
  – SNMPv3 (2002)
    • SNMPv3 started from SNMPv1 (and not SNMPv2c)
    • Addresses security
• All versions are still used today.
• Many SNMP agents and managers support all three versions.

SNMP Security

• SNMPv1 uses “community” strings for authentication – In plain text without encryption
• SNMPv2 was supposed to fix security problems, but effort derailed – The “c” in SNMPv2c stands for “community”??
• SNMPv3 has key security features:
  – Ensure that a packet has not been tampered with (**integrity**)
  – Ensures that a message is from a valid source (**authentication**)
  – Ensures that a message cannot be read (**privacy**)

SNMPv3

• Has three security levels:
  – Depending on how you connect – you get more access rights
• **noAuthNoPriv**: Authentication by matching a user name.
• **authNoPriv**: Authentication with message digests.
• **authPriv**: Authentication with message digests, and encryption



What are we GET/SETting in those packets?

• Values stored in a Management Information Base (MIB)
  – Collected under a Structure for Management Information (SMI)
• Written in a formal language (ASN.1) – A formalism, rather than a language
• Field day for informaticians, logicians and other purists...

Table 24.1 *Data Types*

Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31}-1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from zero to 2^{32}; when it reaches its maximum value it wraps back to zero
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

On Counters and Gauges...

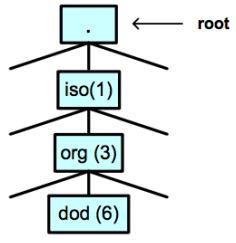
• Reading Counters/Gauges tell you about “now”
  – **Counter** e.g. packets on an interface (can wrap)
  – **Gauge** e.g. memory/disk space (ranges between zero and <maximum>)
• Agents don’t have history, and don’t calculate rates/changes – Agents only have a temporary clock - Time since boot
• Managers have to ask more than once, and make assumptions
  – Counter doesn’t change = World hasn’t changed
  – Gauge doesn’t change = World may have changed, or not, between requests
  – MIB designers might need multiple fields/types for related information

ASN.1 Know it exists and where to look it up...

• Abstract Syntax Notation One (1980’s) – predates XML, etc.
• Formal description of data structures, message formats – Type, length, value (TLV)
• Predefined basic types
  – BOOLEAN, INTEGER, OCTET STRING, BIT STRING, REAL,
  – ENUMERATED, CHARACTER STRING, OBJECT IDENTIFIER
• Constructed types
  – SEQUENCE, SEQUENCE OF, CHOICE
  – Arbitrary nesting of types and sub-types
• Encoding types (10+) = TLV to bytes – we’ll stick with ‘BASIC’

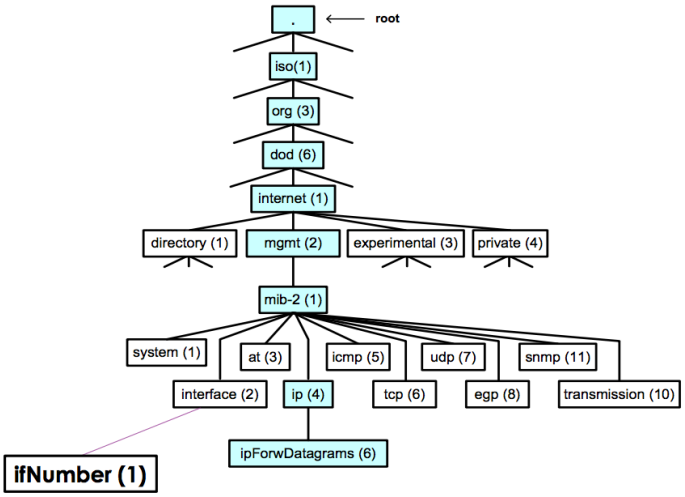
ASN.1 OBJECT IDENTIFIER (MIB)

• Define an information object and reference
• Managed at the international level
  • internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
• Globally unique



OID Organisation

- Tree hierarchy – like DNS
- Each OID is a node in the tree.
- Most internet stuff is 1.3.6.1.2.1.xyz
- Manufacturers can add product specific objects to the <private> hierarchy. 1.3.6.1.4.abc
- SNMP uses OID for reference
- MIBs map OID to readable form – And specify their type, etc.



ASN.1 string examples

```
• Access ::= "read-only"
           | "read-write"
           | "write-only"
           | "not-accessible"
• Status ::= "mandatory"
           | "current"
           | "optional"
           | "obsolete"
```

ASN.1 examples

- **Type** definitions
  - `NumberofStudents ::= INTEGER`
  - `PassOrFail ::= BOOLEAN`
  - `GradeType ::= ENUMERATED {A, B, C, D, E, F}`
  - `PointsScored ::= REAL`
  - `Image ::= BIT STRING`
  - `Data ::= OCTET STRING`
- **Value** definitions and assignments
  - `studentsFridaySession NumberofStudents ::= 9`
  - `passCourse PassOrFail ::= TRUE`
- **Combine** type/value definitions

```
– StudentType ::= INTEGER {
      ugrad  (0)
      ms     (1)
      phd    (2)
  }
```

MIB-2 object counting packets

```
ipForwDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The number of input datagrams for which this
            entity was not their final IP destination, as a
            result of which an attempt was made to find a
            route to forward them to that final destination.
            In entities which do not act as IP Gateways, this
            counter will include only those packets which were
            Source-Routed via this entity, and the Source-
            Route option processing was successful."
    ::= { ip 6 }
```

Aka 1.3.6.1.2.1.4.6

A MIB "object"

```
-- The Interfaces group
-- Implementation of the Interfaces group is mandatory for all systems.

 ifNumber OBJECT-TYPE
             SYNTAX  INTEGER
             ACCESS  read-only
             STATUS  mandatory
             DESCRIPTION
                     "The number of network interfaces (regardless of
                     their current state) present on this system."
             ::= { interfaces 1 }
```

Variable names are aliases for digit strings (defined by MIB)
interfaces *defined in MIB as 1.3.6.1.2.1.2, so* ***ifNumber*** *= 1.3.6.1.2.1.2.1*

More on the interfaces (MIB-II)

Name	Description
ifMTU	Maximum packet size
ifSpeed	Bits/sec
ifPhysAddress	e.g. MAC address
ifOperStatus	Up(1), Down(2), Testing(3)
ifInErrors	# incoming packets discarded due to errors
ifInDiscards	# incoming packets discarded due to buffer overflow
ifOutQLen	# packets in outbound queue
ifInUcastpkts	# incoming packets received

Why?

• OIDs provide global uniqueness – and extensibility
• OIDs provide human-readable-names for tree-position-identifiers
• Also: ASN.1 does not offer tables – But humans need them

Interface #	IP address	State	Packets	Errors	Rate
1	150.203.1.1	Up	1172	5	100Mb/s
2	130.56.3.1	Up	1234	3	100Mb/s
3	197.197.4.1	Down	5678	4	100Mb/s
4	197.197.5.1	Up	8451	197	1000Mb/s
5	8.8.8.1	Up	9191	2	10Mb/s

Tables and GetNext

- Each table cell has a 1.3.6.1.2.x.y.z.abc.label identifier
- Rows in a table get sequential entries based on the index – E.g. Interface number
- Manager doesn't know how many rows (interfaces) there are

Get ("Interface.1.ipAddress") ➔ interface.1.ipAddess = 150.203.1.1
*Get-next ("Interface.1.ipAddress") ➔ interface.**2**.ipAddress = 130.56.3.1*
...
Get-next ("Interface.5.ipAddress") ➔ something else in the MIB

– There is no 'row/column-count'. Don't need it. May change anyway!

This works even if you don't know the names/columns/rows – ***Lexicographical Order*** for OIDs

But!

- Repeated Get-next:
  – Lots of extra there-and-back traffic
  – More state to maintain/evolve in Manager (row#/column#)
- SNMPv2 introduced Get-Bulk request
  – Get-Bulk("interface") ==> every row, every column
  – But only one UDP packet comes back
  – Error response "tooBig" (64kB UDP limit)

SNMP beyond my domain?

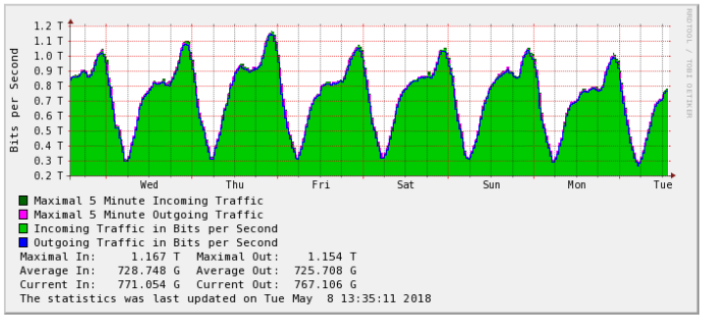
- SNMP in the wide area is ... unwise
  - – SNMPv1/v2 agents should not be visible. Ever.
  - – Lots of traffic
  - – Easy to scan/map network
- Becomes a human problem

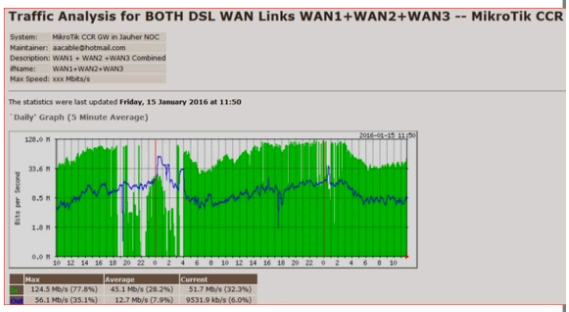
– Need to ask for favours

- **Beacons** (e.g. multicast)
- **perfSONAR**
- **Looking Glass**

- – Remote login
- – Limited (read-only) queries
- – Various ISPs

Review: Performance over time





MRTG, Cacti, Nagios - as monitoring/graphing tool

Security at what layer?

• Spans all layers
• Every layer provides opportunities and risks • From the application down to the physical
• Various security designs, for a variety of threats – It's more than just cryptography

Security?

• In the eye of the beholder
• Need to understand a range of properties
• Threat model:
  – What are dangers? the capabilities of the 'attacker'? the probabilities, impacts and costs?
• All help to assess the risk – and the effort to mitigate it: Remove a risk, or deal with the consequences?

Example threats and levels

• Note: focussing on network security, rather than application security
– But each is a vector into the other!
– The edge is not very clear
– And scale can be tiny/targeted to huge/widespread
• Eavesdropping: Intercept messages, gain content (passive)
• Intrusion: Compromise device, modify messages (active)
• Impersonation: Identity fraud, gain content, modify messages (active)
• Extortion: Disrupt services (active)

Vulnerabilities

• Attack surfaces
– How many places are you vulnerable?– How do you know? – E.g. Use of cloud services for smart devices
• Single points of failure
– Can I knock you out at a single device?– Routers, servers, directories, databases, file

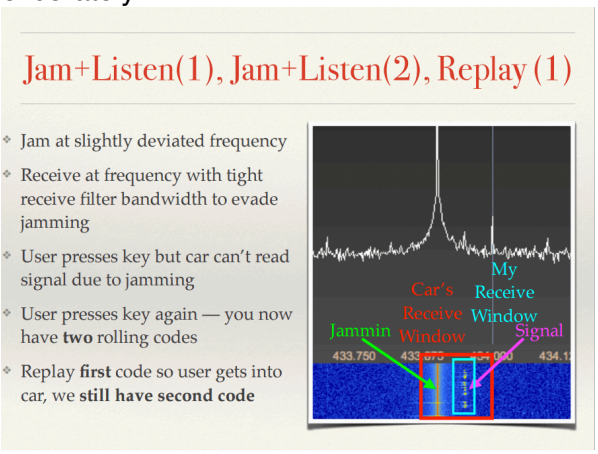
Security = risk management

• Can never be perfect – cannot prove an absence
• Ensure security model to minimise probabilities
• Only as secure as the weakest link:
  – Design flaws
    • Poorly thought through
    • Law of unintended consequences (multi-component systems)
  – The Internet is the world's largest multi-component system... – Code flaws
    • Always one more bug
  – Somebody else's flaws...
    • Human behaviours – Accidentally, deliberately

e.g. Replay attacks

- Samy Kamkar – Hackaday Supercon
- Jam and Listen: Steals codes without alerting car/user

• Pick up short-range car-signal for nearby keyfob: retransmit to friend near car-owner, return
response, car unlocks!
• Find owner: Send hunt signal into room and listen for responses!



e.g. Wifi is ‘more’ secure now...?

• Old WEP Cryptography – really weak – easy to snoop and decrypt
• 802.11i – Nearly impossible to brute-force decrypt on today’s computers
• Removed one threat, but e.g
– Could have configured a guessable SSID/key – gets attacker on the WLAN
– Could have wired LAN access to devices on WLAN – gets attacker onto LAN/WLAN
– Physical access to network devices (access points): People have tapped the chips on AP boards
• All lead to someone accessing the network, and listening to (some) traffic – And sending, possibly

Naïve Internet designs

• Security has been added on over the years
– Many protocols came from a smaller, friendlier network community
• DNS, DHCP, HTTP, ...
• Clients/servers had prior relationships
• No concept of identity in most protocols – And which identity?
• Earlier designs never considered the value attached to the network
• Banks, businesses, factories, power stations, government agencies, control systems, ...
• Significant effort to retrofit security
completely redesign with security in mind / use other mechanisms to provide security

Basic assumptions

• There will always be villains, in various forms and paygrades – And they know nothing initially – so probe. All the time.
• All physical links can be interfered with – snooped, misdirected, cut, added, ...
• You can’t trust packets; headers nor payloads!
• Protocol designs are public and have many holes
• Security ‘on the wire’ can be undermined by security ‘on the host’ • Technology is easy to hack, and people are even easier

Crypto-graphy...(“Hidden writing”)

• A common first point for security
– For data ‘in motion’ (travelling over networks) – For data ‘at rest’ (in applications)
• Cryptography
– Encrypt information
– Make it computationally infeasible (too much time) to decrypt
– But it has an ‘edge’, where it stops
• Cryptanalysis – Try to decrypt information

Encryption

• Not just for preventing eavesdropping (confidentiality)
• Can also
– Confirm messages came from device you expect
– Confirm remote party is who they say they are
– Validate message has not been tampered with
• Remarkably easy to develop poor encryption
– Many half-baked attempts over many years
– Stick with well-known platforms/systems • And try to use them well! • And still be wary

Confidentiality

• Encryption to ensure messages can't be read while travelling
  – Goal: send a private message from A to B
  – Threat: (Passive) villain-in-the-middle reads message along the path
  – "Application" end-points en-/decrypt messages
• Two common approaches:
  – Symmetric (shared key)
  – Asymmetric (public/private key)

Symmetric vs Asymmetric

• Shared secret crypto
  – Both parties have the **same key**, use it to encrypt/decrypt messages
    • Same algorithm used at both ends (e.g. AES)
    • Assume attacker knows the algorithm
    • Sharing the key is a weakness
• Public Key crypto
– **Key pair** (public/private);
– Owner (only) holds private key, anyone can/should have public key
– Encryption through expensive mathematics (e.g. RSA)
• "Only" private key can decrypt public-key-encryption, and v.v.
– Public Key Infrastructure (PKI)
• Rule: Want network to carry ciphertext (encrypted messages) only.

Key distribution and performance

• Trade-off: which approach?
• **Symmetric**
  – Encryption is lightweight/fast – great for high data rates
  – Key sharing is hard:
• Need to get (different) key to everyone who needs/deserves it • For every new conversation
• **Asymmetric**
  – Encryption is heavy/slow – not for general use – Key sharing is easy
  – But also need to know you can trust the public key • Bind an identity to the key
• Needs a directory service (see certificates)

Best of both?

• Use public key encryption to send a shared key
• Use shared key for encrypting further communication – Ensure confidentiality
• This shared key is a **session-key**– Short-term use, encrypt each packet – As big as you need it to be– Can generate a new one each time

Authentication and Integrity

• Confidentiality is great against passive villains – They can't read the packets, and you can authenticate source
• Active villains (intruder) may still tinker with the **message** en-route
  – Incorrect, misleading, broken message gets through
  – Corrupt, replay, reorder packets: WAIT DO NOT STOP – STOP DO NOT WAIT
• Need **message integrity**
  – Use session-key (established through PKI)
  – Calculate a summary of the **message** (signature, message digest, hash) • And encrypt that with session key or private key

Freshness

• Villain can store (encrypted) messages,

• and send them again and again - later
  – E.g. ‘transfer $1000 to X’/ ‘set password = ABCDEFG’
  – No matter how well encrypted, as long as within timeframe of session key
• Replay attack
• Easy fix: include timestamp (or other ‘nonce’) in the message/signature – Before encryption – (Application requirement, not part of crypto)

Applying cryptography

• Each application can now build their own:
1. Confidentiality 2. Authentication 3. Integrity 4. Freshness into their protocols
• We trust every app developer, every language, every OS to get it right? • Why not add it to the network? – Which layer - Link, Network, Transport? • Options for each of them...

Establish a Secure Socket Layer

• SSL (1995/96 SSLv3) – Netscape browser
• Triggered by HTTP ==> HTTP/S (HTTP over SSL) = **https://**
• Led to generalised Transport Layer Security (TLS) [V1.0 1999, V1.1 2006, V1.2 2008, V1.3 2018] DTLS = TLS/UDP
• No longer specific to HTTP
• Sits “between” Transport and Application – encrypts tcp payloads

HTTP, ...	Application
SSL/TLS	
TCP	Transport
IP	Network

What do we get?

• SSL/TLS provides
  – Verification of server by client (padlock icon in www)
  – Message exchange • with confidentiality, integrity, authentication and freshness
• Starts with authentication phase
  – To establish encrypted channel and session key(s)
  – Before any single application message (HTTP) is exchanged
• Client needs to authenticate some random new server
  – Network traffic can be spoofed and misdirected
  – Needs server public key, **for sure**... provided by a certificate

Certificates

• **Bind an identity to a public key** – Server/service, or person
• Requires somebody authoritative to say so
  – Certificate Authorities (CA)
  – X.509 standard: Metadata about identity, plus public key, encrypted with CA private key

Public Key infrastructure

• Can’t have just **one** Certificate Authority – Too busy to deal with the whole Internet
  – Too big to fail
  – Too obvious a target
  – Too easily untrustworthy and a monopoly
• Build a hierarchy
  – One or more competing ‘root’ CA’s
  – That validate/sign delegate CA’s
• That ... – That ...
• That sign your/your web server’s certificate

Anchoring trust

• Browsers hold certificates for root CAs
– Around 70 **root** certs in Chrome today – and cache many more – That's a lot of trust...
• On SSL/TLS initiation, request server certificate
– And check its CA signature • And check its CA signature – And check its CA signature• ... up to the root signature
• Nice, till somebody gets hacked
– Private key is exposed
– Certificate must be revoked
– PKI has a Certificate Revocation List (CRL) – this does not scale well!

That solves everything?

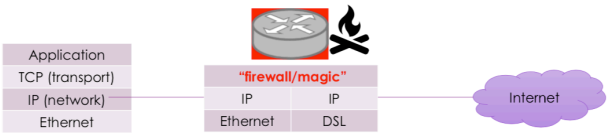
• Not even close
• SSL/TLS: a common security layer between applications and transport
– And has itself been broken a few (12+) times
– Code flaws, crypto flaws, interaction flaws with other protocols, spooks, ...
– Not used by all applications, or other layer protocols (DNS, DHCP, BGP, ...!) • Recall DNSSec discussion earlier
• If you need it, use it–or get a PhD in crypto (and then use it). • What about other layers?

Firewalls

• Edge routers/gateways/processes that (can) explicitly block packets
• Internet: – You can send to any host. – Any host can send to you!
• Only want to let the nice packets in (and out)

The "middlebox"

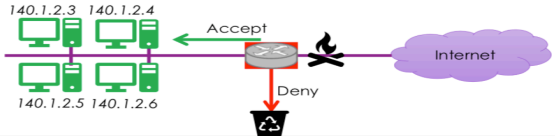
Routers (normally) just look at IP – but...



Policy time

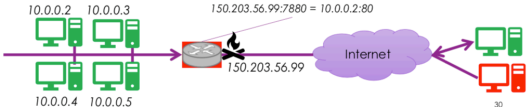
• Establish Accept/Deny rules
– Break applications we don't like – Very high level
• Packet filtering is low-level
– Doesn't look at protocol messages,

encrypted traffic, ...



Firewalls at different layers

• Basic block – "stateless"
– No state from one packet to the next
– Allow/Deny based on IP addresses
– Allow/Deny based on TCP vs UDP
– Allow/Deny based on Port numbers
– E.g. deny port 25 tcp (email) / deny all, allow port 80 tcp (http) • Because... people.



Stateful Firewalls

• Change the rules based on other triggers
– Track packet flows between internal and external hosts
– E.g. NAT: Allow inbound TCP from any outside X to our inside Y that initiated a connection to X
• But timeout after idle...

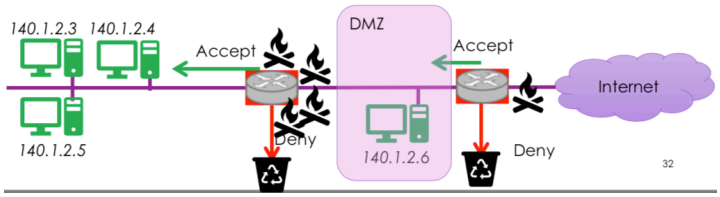
Application firewalls

• ***Deep Packet Inspection***

• Understands application protocols
– Will reassemble messages from packets
– Understands content • E.g. viruses in emails/web pages
• And performance suffers

Firewall deployment

• Sometimes need to protect some devices more than others – Internal finance system versus your company web-server
• Two stage firewall: create a 'demilitarised zone' (DMZ)



Firewall implementation

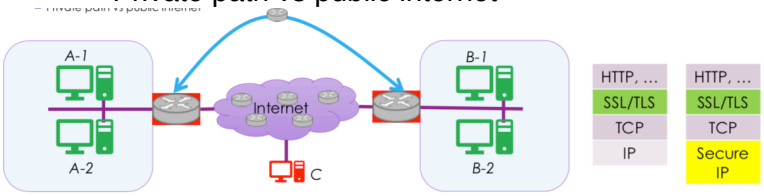
• Dedicated devices – Especially Application/DPI Firewalls
• Routers/Modems
• Wireless Access Points
• On hosts, in the Operating System (e.g. Linux IPTABLES)
• Tradeoffs
– Multiple firewalls = more security **AND** more places to break/slow things
• More places to maintain and coordinate
• Protect hosts from each other

Firewalls = Islands of trust

• SSL doesn't hide everything
– Intermediate routers can see the traffic • Leak information about activities
– What about End-to-End confidentiality/etc? • Host to hos• subnet to subnet

Islands of trust

• Leased lines
– Who needs the Internet? • Effective – sort of (smaller attack population)
– But • Doesn't scale to buy physical links ($$) • Need to manage routing ($$)
– Private path vs public internet

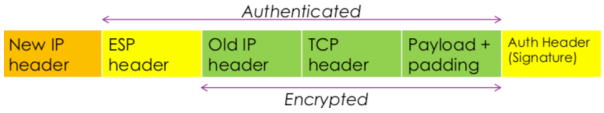


• Can we use the Internet?– Need security at the IP layer
• Make a "virtual" leased line – Virtual Private Network (VPN)
VPNs :• Tunnel (=encapsulate) IP packets across the Internet (IP in IP)

IP tunnelling security

• IP in IP (encapsulation) has **no** protection
– They're still ordinary packets, just an extra header – No confidentiality, authentication, or integrity

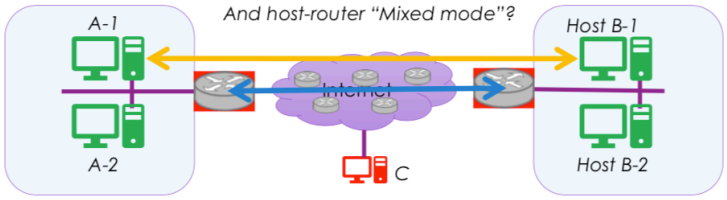
• Use **IP Security (IPsec)** to establish secure VPN connections
– Cryptography at the network layer
– Keys are exchanged between **endpoints** (can be hosts and/or routers) – Packets are encapsulated and encrypted



(ESP = Encapsulating Security Payload)

VPN endpoints

• **Tunnel mode:** router to router (i.e. with forwarding)
– Connects whole subnets transparently – make them look as one – Can be NAT-friendly
• **Transport mode**: host to host (i.e. no forwarding)
– Different format, only encrypt IP payloads, NAT-challenged.

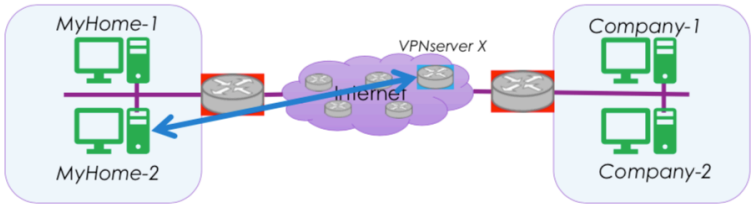


Address transparency

• New packet gets IP address of tunnel endpoints
– Effectively a new layer, IPsec over IP
– Can't identify (original) source/destination • Until you come out of the tunnel
– Good thing? Bad thing?
• Break firewalls (both for simple IP and packet inspection) • Undermine optimal routing
– Sort of...

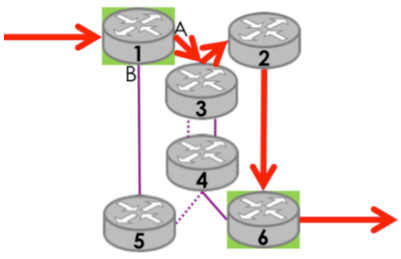
Address opaqueness

• A VPN endpoint is not necessarily the destination
– Just where the packets 'emerge' from the tunnel
– 'source' address is the tunnel endpoint • Responses need to go back there
– IP addresses are allocated to geographical regions • This has some benefits...



Looks a bit like a circuit?

• Well, yes.
• But only the endpoints are tied down
– Internet routing handles everything in between
• Dynamically, politically, ... • Deals with failover, multi-path, ...
– Packets are labelled with IP addr.
– Fails when either endpoint dies
• Compare with MPLS
– Multi-protocol Label Switching (back in T10)
– Pre-establish the entire path
– Packets are labelled with a token
– Fails when any path-element dies



Device security

• Switches/routers have physical and virtual interfaces
• Remote access
  – SNMP agents (http admin)
  – Operating systems (telnet/ssh)
  – Port monitors/mirrors： Reflects traffic to another port • Can't tell from outside
• Physical access
  – Interface rearrangement (denial) or attacks – Cable cutting/interference
  – Chip-pin-level snooping

Copper security

• Easy to tap
• Hard to detect
• Actively cut cable– Denial of service or Impersonate Device
• Splice cable
  – Eavesdrop what is on the wire
  – Impersonate Device
  – Denial of service (noise, energy injected)
• Hands-off tapping
  – Unshielded copper is an antenna
  – As transmitter (leak) and receiver (interfere)
• Some Layer-2 security

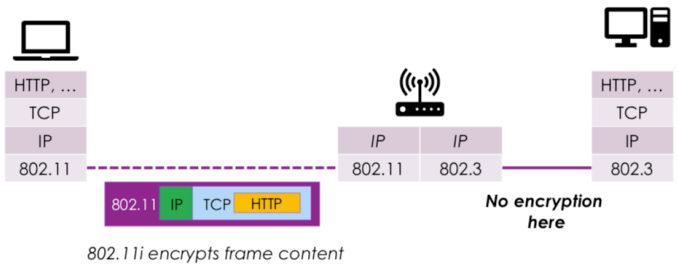
Fibre security

• Also easy to tap
– Some monitors (oc3mon) used a prism
• Can be easier to detect
– Energy loss (attenuation?)
– Quantum entanglement of photons...
• Splicing and cutting (mostly) same as copper – Suggestions NSA have done this on the seafloor
• Hands-off tapping
– Adjacent fibres leak, fibre jackets leak

Wireless security

• By definition, wireless is broadcast – Narrow beam antennas help
– Shorter wavelengths help (optical)
• Take it as read, villains are listening, and can actively intrude
• Each wireless approach is different – And limits what it promises
– Please encrypt at higher layers!
– But in case you don't...

802.11 – and Wifi Protected Access (WPA1-3)



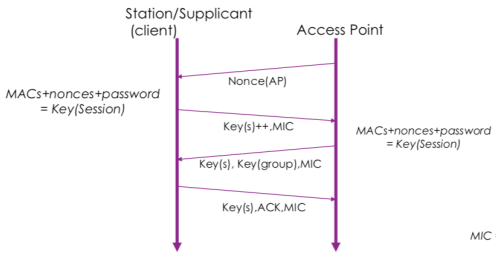
Consider a WiFi Home network
• Typical(un-open)Wifi–pre-shared secret(key)[PSK]

• Client has to prove it(also)knows the(AP)secret
– Ideally without sending it in plain text across the network – AP then grants access

WiFi network keys

• Encryption again - several keys – now at layer 2
• Client authenticates to AP
– Each calculates a shared session key from the SSID password • Pairwise Transient Key (PTK)
– Client tells AP, AP accepts, registers and hands out more keys • AP-to-clients (broadcast/ multicast)
– Additional group (temporal) key (GTK)
• Client and AP encrypt with session key – Confidentiality, integrity, and authenticity

Keys on keys…



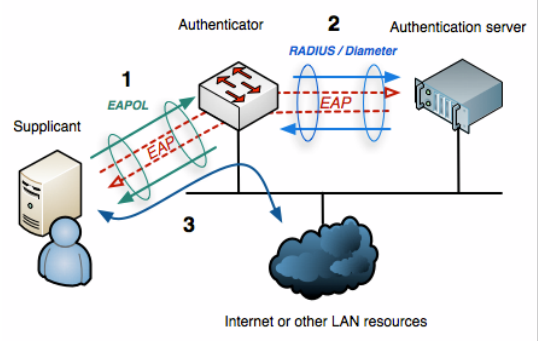
Keys over keys on keys with keys…

- Pairwise Temporal/Transit Key (64 bytes) =
  - Key Confirmation Key (KCK)
  - Key Encryption Key (KEK)
  - Temporal Key (TK)
  - MIC Authenticator (AP) Tx Key
  - MIC Authenticator (AP) Rx Key
- Group Temporal Key (32 bytes) =
  - Group Temporal Encryption Key
  - MIC Authenticator (AP) Tx Key
  - MIC Authenticator (AP) Rx Key

And more:
- *Pairwise Master Key ~ PSK*
- *Group Master Key*
- *Master Session Key*

Pass-through authentication

• 802.1X
– Uses Extensible Authentication Protocol (EAP) • Can be used on 802.11, 802.3, etc.
• Typically RADIUS back-end
– Each client has own credentials • No PSK



WiFi Encryption history

• **Wired Equivalent Privacy WEP**
– Don't go there. Single key, easily calculated from traffic sniffing.
• **WiFi Protected Access WPA**
- With Pre-shared-key (PSK)="personal" or 802.1X="enterprise",
  • Better integrity checks than simple CRC
  • Temporal Key Integrity Protocol (TKIP) – per-frame-key
  • Becomes Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP)
- Heaps better. Still broken
  • largely through WPS ("easy-to-join" feature)

• **WPA2**
- Lots of additional measures. Much stronger encryption and other protections.
- Still KRACKed

• **WPA3** – Still warm of the press (Jan 2018)

Other kinds of attack - Denial of Service

• Not all attacks are about eavesdropping or fraud

• Some (many) prevent your systems from being available to intended users
  – “Take down” your website, booking system, banking portal, government site, …
  – For fun or fortune!
• Based on resource starvation
  – Encryption can’t help you now!
  – Use up bandwidth, router cpu/memory, server cpu/memory/disk

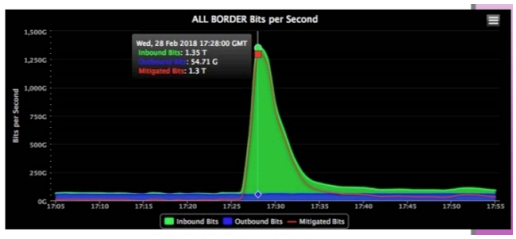
DoS vectors

• Tricky packets at different layers...
• Ping of Death
  – Large ICMP packet, multiple fragments, modified headers
  – Reassembled into >64kB – memory overflow
  – Actually a direct attack on the target host
• SYN flood
  – Open a TCP connection to a server
  – But never follow through on SYN/ACK
  – Server builds connection state for each inbound packet
  – Can be avoided with SYN Cookies
• Server builds connection state only after ACK
• Application layer messages:
• Making small but complex queries – Big database queries – Complex regex
• Memory overflows • Other bugs
• Can starve server

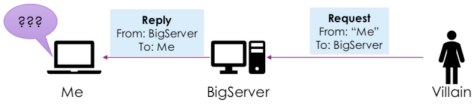
Distributed Denial of Service

• Could flood somebody with a single server
• But hey, it’s the internet, let’s use **millions**
– IoT devices, webcams, NVR, baby monitors, smartphones, ... “botnets” – More common than desktops, laptops

• Scale?
  – 10Million attacks/year, at 1-2Gb/s
  – 2016 – first 1Terabit/s DDoS attack
    • *100,000+ wireless webcams*
  – 2017 – multiple 1Tb/s attacks

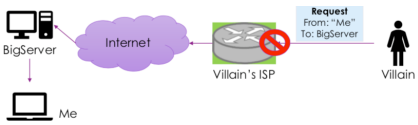


“Spoofing”



• Many DoS attacks spoof
• Sending packets with false source addresses – Get other boxes to work for me
– Very hard to trace

Spoofing vs Ingress Filtering



• Good practice:
  – Check Source IP at their boundary to the Internet • Nobody else can...
  – Obvious? Obvious! And yet not widely enabled (more work, and only benefits others)

Multipliers

- **Host multipliers**: Botnets...
  - – Lots of hacked devices, controlled centrally
  - – Each sends any old kind of packet (e.g. ping flood, udp, ...)
- **Packet multipliers** – send one, get many – often with 'spoofing'
- SMURF (and variations)
  - – Ping the broadcast address (one packet out)
  - – Use the target's IP address as source (spoofing)
  - – Everyone replies to the source (many packets back) – Easy to prevent...

Packet multiplication

- Small requests, big responses
- DNS
  - – 1 packet request "list the hosts inside anu.edu.au" – Multi-megabyte response
- HTTP/FTP/NFS/...
  - – 1 packet request "Send me that 10GB file" – ...10GB later...?
- Memcache servers
  - – Database accelerators – over UDP
  - – Should be only inside your network – and yet there are thousands visible

Mitigation – really hard

- The Internet is designed to shift lots of packets...
- Content Distribution Networks – Don't be a single target
- Edge routers/Attack detection
  - – Efficient packet dropping
  - – Better filtering support (e.g. DoS from a particular country?)
- Upstream provider support
  - – Can re-route most/all traffic elsewhere
  - – E.g. dedicated DDoS processing systems
- Get ingress filtering everywhere! – And fix all those webcams while you're at it...

Wednesday 8 May 2pm: Glenn Archer. Glenn是前澳大利亚联邦政府的首席信息官（在PM Tony Abbott和Malcolm Turnbull时期），在联邦公共服务部门担任过多个非常高级的职位。 在此之前，他曾为Gartner，思科和Apple工作多年，并且是ANU-CS的校友。 最近，Glenn一直在为位于ACT东北边界的Wamboin，Bywong，Sutton地区的1000多个家庭协调区域FTTP计划。 该项目刚刚被新南威尔士州政府授予500万美元，并被列为类似区域项目的典范。 在这个讲座中，格伦将概述这一举措，以及NBN如何与此并列，以及在通过人们的院子讨论沟渠和电缆时，社区参与如何发挥最佳作用。

Wednesday 15 May 2pm: Robin Eckermann AM. Robin是前TransACT项目负责人，最终担任首席架构师。 他还是部长Conroy和特恩布尔的联邦政府宽带顾问，并支持多个州/地区级通信网络基础设施计划及其对社会的影响。 最近，他还参与了智能电网和大规模传感器网络。Robin将主要讨论TransACT的历史和进展，TransACT是在NBN之前20年建立的堪培拉全球FTTN网络，并且仍然很强大，并且他认为未来正在带领我们。

- $250m Network Rollout in Canberra
- Developed under ACT Electricity & Water (ACTEW) • Australia's first Advanced Broadband Network
- Cited as "inspiration" for the NBN
- One of the world's first "triple play" networks
- Fibre-to-the-kerb (FTTK) cabling architecture
- Footprint of ~65,000 premises
- Initial network build completed in 2003
- Continues to operate & evolve as NBN alternative
- Now under iiNet/TPG ownership

•堪培拉的2.5亿美元网络部署
•在ACT Electricity＆Water（ACTEW）下开发•澳大利亚首个高级宽带网络
•被引为NBN的"灵感"
•世界上第一个"三网融合"网络之一
•光纤到路边（FTTK）布线架构
•约65,000处所的足迹
•初始网络构建于2003年完成
•继续作为NBN替代方案运营和发展
•现在属于iiNet / TPG所有权

Why TransACT? Why Canberra?

Vacuum (bypassed by cable TV rollout):In the mid-1990s Telstra & Optus were engaged in a race to establish Pay TV coverage – but Canberra was seen to be difficult and low on the priorities 真空（有线电视推出绕过）：在20世纪90年代中期，Telstra和Optus参与了建立付费电视报道的竞赛 - 但堪培拉被认为是困难和低优先级

Opportunity (great market) Information economy Well educated Well equipped Technically literate Relatively affluent 机会（大市场）信息经济良好的教育装备精良技术文化相对富裕

Capability (ACTEW's utility assets) A multi-network company Poles, rights of way etc Lifeline culture 100% customer base Systems & infrastructure Workforce 能力（ACTEW的公用事业资产）多网络公司Poles，权利等生命线文化100％客户群系统和基础设施劳动力

TransACT's 3-Pronged Vision

Advanced 高级

• Fibre deep into network for high capacity • Optimised for interactive use (Internet etc)•光纤深入网络以实现高容量•针对交互式使用进行优化（Internet等）

Open 开放

• All welcome (no regulatory intervention needed!)• Service providers not threatened by network owner 欢迎所有人（无需监管干预！）•服务提供商不受网络所有者的威胁

Full-Service 全面服务

• Not biased towards any one service • Telephony, video and data•不偏向任何一项服务•电话，视频和数据

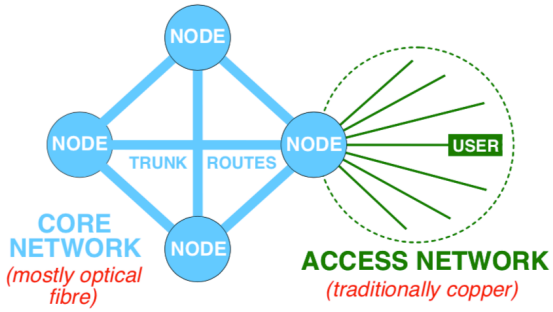
An Open Access Philosophy

- Many services on one infrastructure is efficient
- Efficiency = lower costs for users and service providers (SPs)
- Enhanced service choice is attractive to users
- User uptake critical to viability for network owner
- NOTE: access/service split rather than wholesale/retail split

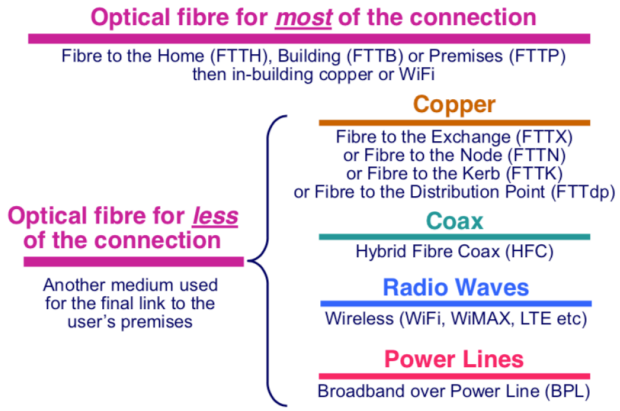
- 一个基础架构上的许多服务都是高效的•效率=降低用户和服务提供商（SPs）的成本•增强的服务选择对用户具有吸引力•用户对网络所有者的可行性至关重要•注意：访问/服务拆分而不是批发/零售 分裂

One single fibre pair can carry billions of simultaneous phone calls between two centres ...
... but think about what's needed at each end to collect and distribute those calls!
一根光纤对可以在两个中心之间同时拨打数十亿电话.....
…但想想收集和分发这些电话的每一端需要什么！
The access network is by far the most challenging & expensive communications infrastructure problem! 接入网络是迄今为止最具挑战性和最昂贵的通信基础设施问题！

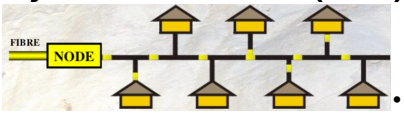
Core and Access Networks



Different media & technologies used in access networks

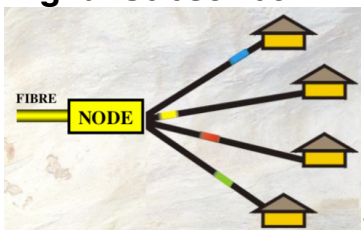


Hybrid Fibre Coax (HFC)

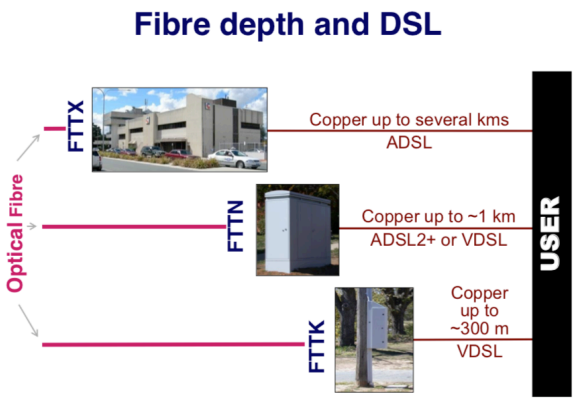


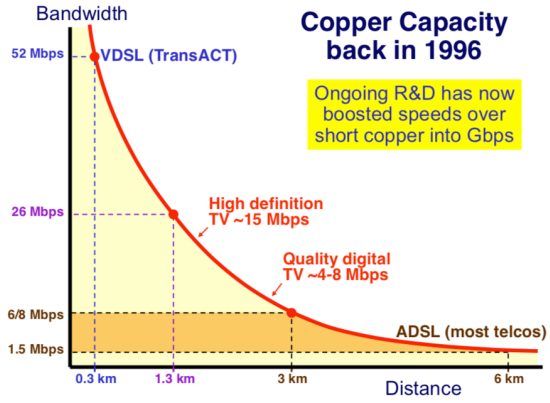
• Uses shared COAX cable from the end of the fibre to the user • Like a "party line" - everyone receives everything • Ideal for broadcast services - like TV - but ...
... with interactive services, everyone wants to do their own thing and this leads to contention for capacity in busy times •使用从光纤末端到用户的共享COAX电缆•就像"派对线" - 每个人都收到一切•非常适合广播服务,如电视. 但通过互动服务，每个人都想做自己的事情会导致在繁忙时期争夺容量

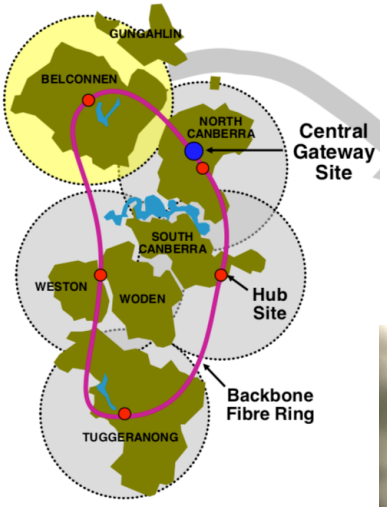
Digital Subscriber Line (xDSL)

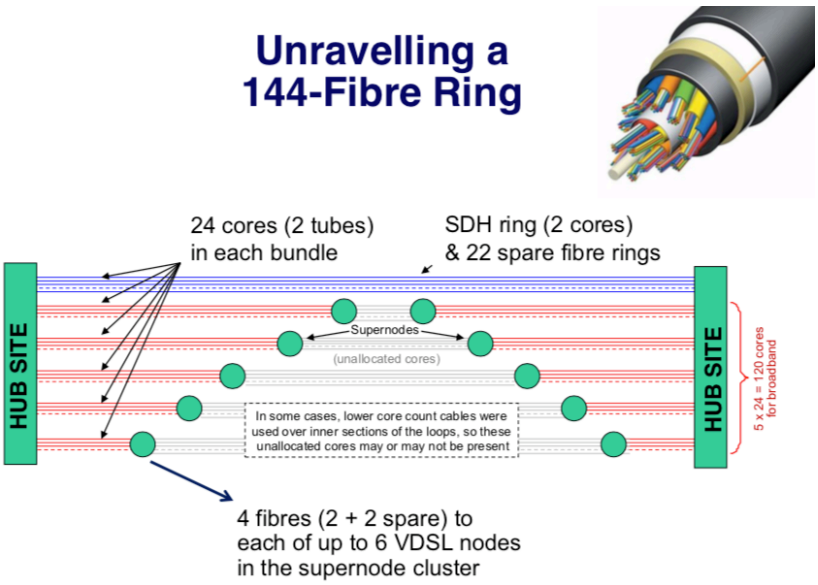


• Uses dedicated copper cable from the end of the fibre to the user • Each user has their own non-shared capacity • Ideal for interactive applications - like Internet - but capacity depends on length and condition of copper •使用从光纤末端到用户的专用铜缆•每个用户都有自己的非共享容量•非常适合交互式应用 - 如互联网 - 但.....容量取决于铜的长度和状况









Gateway Site

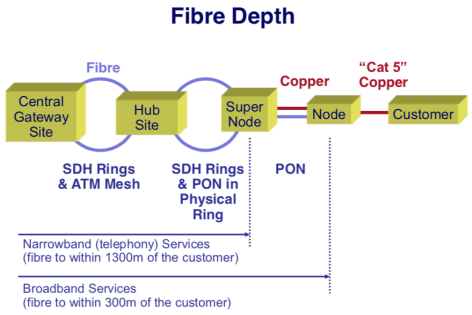
Central Broadband & Narrowband Switching/Routing Primary Point of Interconnect with Carriers & Service Providers Broadcast Video Headend Tele-housing Facility for Content & Service Partner Equipment 中央宽带和窄带交换/路由与运营商和服务提供商的主要互连点广播视频头端内容和服务合作伙伴设备的远程住房设施

"Supernode"

- Plain old telephony (POTS) as well as broadband equipment)
- Serves ~400 homes
- Battery-backed power • Located on fibre ring
- Fibre management
- Max 1.3 kms to user

- "超级节点"
- •普通老式电话（POTS）以及宽带设备）
- •服务约400个家庭
- •电池供电•位于光纤环上•光纤管理
- •最大1.3公里的用户

CableReserve(in case of pole move)& passive cooling



Functional View of Network - Old

- Three service "streams" - voice, image & data • Two "parallel" networks (Broadband Network & Narrowband Network) sharing common infrastructure

Functional View of Network - New

- Narrowband-specific infrastructure is gone
- Telephony & video tend to be carried as "over the top" (OTT) services on the data network

TransACT Services

Telephony:

- POTS, equivalent to Telstra
- "Life-line" grade, battery-backed

Data:

• Raw 52 Mbps download
• Maximum 8 Mbps sold to protect video services from interference
• Business services – fibre to the building (whatever the customer wants)

Video:
• Broadcast services with additional "PayTV-like" channels
• Streaming VOD – a la Netflix

Critical Mass: Why its so important

Consider whats involved grooming a TV signal for delivery through an access network

Relayed to the customer via an access network

The cost of signal grooming is the same whether you deliver to one customer or a 100,000 or a million customers

At the heart of the network lies a big pool of expense that is related to richness of service, not size of market

Takeup: A Critical Success Factor!

If one in five homes-passed connect, that customer must support $5,000 in network infrastructure

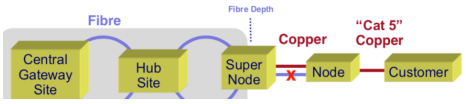
If two in five homes-passed connect, those customers need support only $2,500 each in network infrastructure

No other parameter influences viability so dramatically!

Good Outcomes
Open network philosophy Focus on data, rather than TV Consumer focus
POTS – then a vital service Fibre-rich architecture Community engagement
开放的网络理念专注于数据，而不是电视消费者focusPOTS - 然后是一项重要的服务光纤丰富的架构社区参与

Things that caused grief 引起悲伤的事情
Politics, Predating VDSL standards Demand under-estimation Cashflow pressures
Tech bubble – supplier instability Separate access & services bills 政治，预测VDSL标准需求低估现金流压力技术泡沫 - 供应商不稳定单独的访问和服务账单

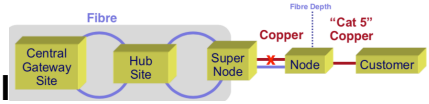
iiNet "Upgrade" ~2013



Result?
Copper distances up to ~1.3 kms
VDSL2 performance from ~25/1 Mbps at longest distances, to ~100/40 Mbps for those in original Supernode serving area
Loosely equivalent to NBN FTTN

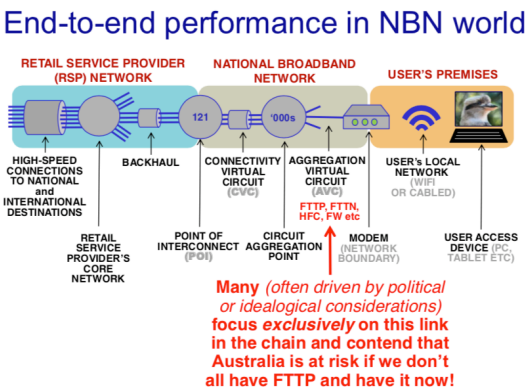
1. DecommissionoriginalVDSL mini-DSLAMs at all Node locations
2. Install VDSL2 DSLAM in Supernode cabinets
3. Conscript copper originally intended for telephony-only for broadband
4. Use Node cabinets just as a cross-connect point
5. Upgrade customer modems

iiNet "Upgrade" ~potential



Result?
VDSL performance to ~500+ Mbps across entire network using G.Fast and probably to ~1+ Gbps as xDSL technologies continue to evolve
Loosely equivalent to NBN FTTK(C)

1. Re-installnextgenerationmini- DSLAMs in all Node locations
2. Optionally power from end-user equipment (over copper lead- ins)
3. Upgrade modems to support latest XG.Fast VDSL protocols
4. Being undertaken selectively already



Some low-speed choices may be due to line limitations. Clearly many more are driven by cost/ affordability considerations 一些低速选择可能是由于生产线的限制，但更多的是由成本/可负担性考虑因素驱动的

Improvement due to discontinuance of ADSL support, greater use of FTTK in final stages of rollout, and continued greenfield FTTP growth 由于ADSL支持中断，在推出的最后阶段更多地使用FTTK以及持续的绿地FTTP增长而改善

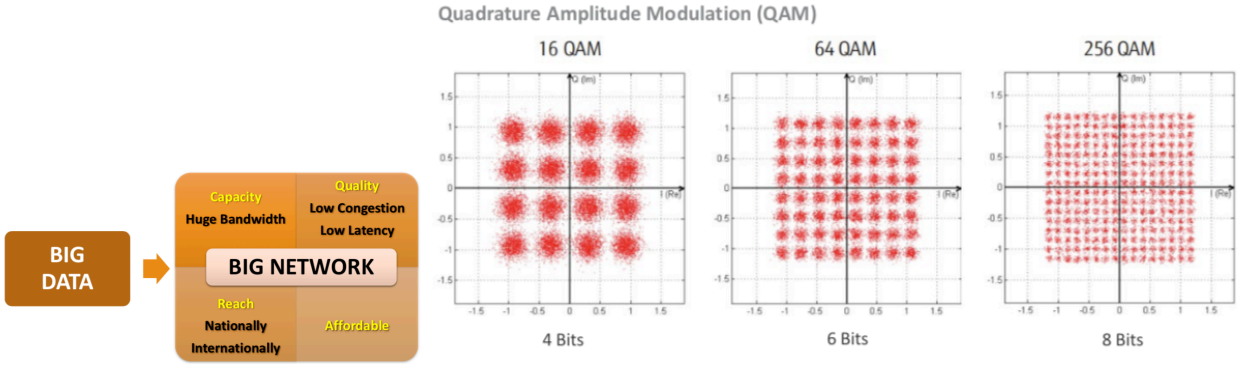
These costs **include** one unit of per-home-passed cost and one unit of per-home-connected cost They **do not include**: • cost of homes passed but not connected• migration fees paid to Telstra & Optus• some $billions in central infrastructure & system costs • NBN operational losses ($20.7b to date)Informed opinion: FTTP93 would have cost ~$100b (approximately double the MTM deployment) 这些成本包括一个单位的每个家庭通过的成本和一个单位的每个家庭连接成本
它们不包括：•房屋成本通过但未连通•支付给Telstra＆Optus的移民费•中央基础设施和系统成本约为数十亿美元•NBN运营亏损（迄今为止为207亿美元）知情观点：FTTP93的成本约为100亿美元大约是MTM部署的两倍

NBN：未来充满挑战
1.导航政治！2.修复可负担性 - 第一要务•目标ARPU（52美元）与消费者承受能力不兼容•5G移动网络是低端用户不断增长的选择•经济实惠的宽带可能支持约250亿美元的网络价值（无论技术如何）
3.将网络升级到2020年以后•推动光纤更深（例如：FTTN到FTTK，降低HFC和FW单元尺寸，卫星覆盖范围内的地面卸载）“你可以实现以下三个目标中的两个，但只有两个：绩效，可负担性和回报”（Bevan Slattery）

22 May 2pm: Peter Elford Peter是AARNet政府和电子研究的现任/前任主管，他们在澳大利亚及其他地区建立了多个100Gb / s的网络，用于研究和教育。 它们连接大学以及医院，学校，文化机构，如博物馆，政府机构以及超级计算机和大型望远镜等主要仪器和设施，在澳大利亚和海底建设自己的光纤（和其他）网络到亚洲部分地区。 彼得也恰好是前ANU CS校友，30年前（几周后）和Geoff Huston（potaroo，ipv6和bgp演讲成名）将互联网带到了澳大利亚。

INDIGO CABLE SYSTEM：海下电缆

AARNET的作用：为澳大利亚研究和教育部门提供电信服务，市场不提供或不以合理的价格提供



将光纤转移到网络中：使用CWDM和DWDM；主要技术从10G过渡到40 / 100GCOHERENT Transmission（2008）80/96通道/光纤@ 100G = 9.6 Tb 长途DWDM设备在CEV中部署了约100公里许多100G服务; 一些200G Trialled 600G；路由器提供L2 / L3（IP）服务

那说什么关于NBN?

经典NBN提供residential服务，而非institutional服务

•在技术和商业上与AARNet服务无关

•带宽有限（大多数AARNet连接为1G，10G ..... 100G）•不对称（100/40）

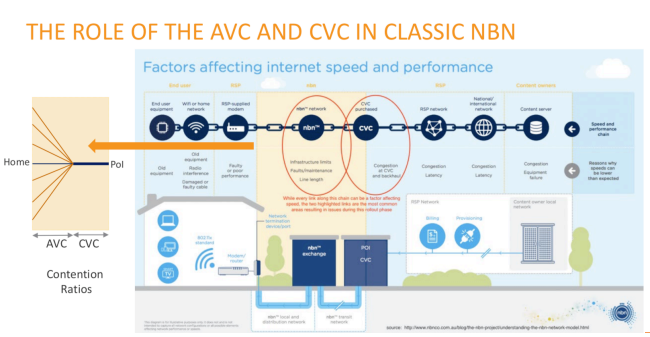
•高度拥挤（假设很多人）

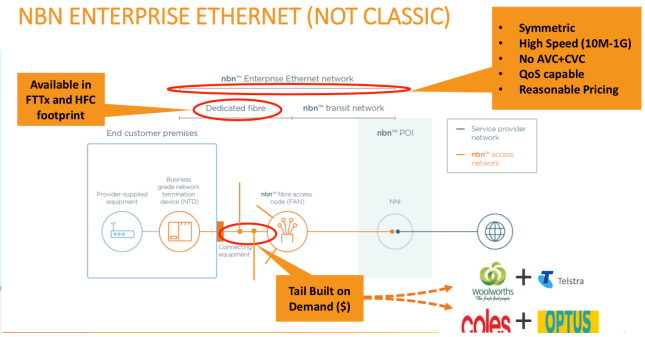
•专用带宽昂贵

在澳大利亚，电信的大部分额外费用是“回程”无法访问

AARNet拥有专门的回程和建立/协商访问权限

•在大多数情况下，将过度建立经典的NBN * NOT A NATIONAL NETWORK ...A bunch of logical connections between locations and Points of Interconnect (POIs)

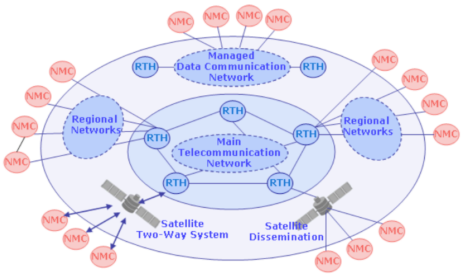


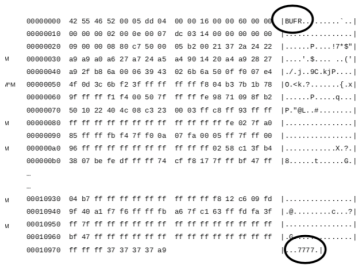


持续的高容量洲际转移证明（400MB TCP缓冲）NREN性能大大超过商业互联网; ISP阻止流量（可能的DoS？;丢包率为0%vs 2-4%;数据传输节点（DTN）仅限DTN内存;今天可实现多10G磁盘到磁盘

Friday 24 May 9am: Okki Lee: 在过去的十年里，Okki一直担任韩国气象局信息和通信技术（ICT）部门的副主任。 他的主要职责包括24x7全天候和国际信息系统和网络的开发和运营。他将介绍天气预报社区的网络需求。 只有随着电信的出现，天气预报才成为现实，因为它允许同时传播的大气条件比天气本身更快。 本讲座介绍了名为GTS / WIS（全球电信系统/ WMO信息系统）的国际气象数据交换框架。 它还将涵盖一个国家气象网络以及它如何与国际网络连接，以标准格式交换标准化观测资料，通常在WMO（世界气象组织）社区的旗帜下全天候进行。
Global Telecommunication Network (**GTS**)

Weather prediction ultimately depends on 1. current conditions of weather on a global scale and 2. weather prediction models running on high-performance computers.





1. Current status of implementation of the GTS circuits

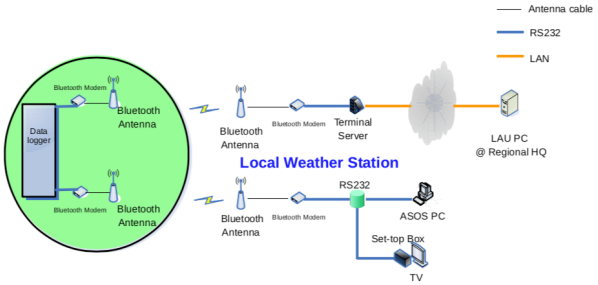
Current status of GTS circuits connected with the RTH Exeter is shown in Table 1.

Connection type	Backup	Centre	Protocol	Daily traffic (Tx/Rx)
Internet		Brasilia	FTP	Under test
		Dar es Salaam	FTP	Under test
		Lusaka	FTP	0.5MB
Megastream (100Mbps)		ECMWF	FTP	6500MB / 20300MB
RMDCN: Internute MPLS 2 x 20Mbps Platinum site		Brussels	FTP	60MB / 30MB
		Copenhagen	IP sockets	105MB / 20MB
		De Bilt	IP sockets	225MB / 470MB
		Dublin	FTP	170MB / 460MB
		Lisbon	FTP	40MB / 42MB
		Madrid	IP sockets & FTP	26MB / 8MB
	Internet	Melbourne	IP sockets & FTP	550MB / 295MB
	Internet	Montreal	FTP	6500MB / 2MB
		Moscow	FTP	5100MB / 25MB
		New Delhi	FTP	572MB / 72MB
	Internet	Offenbach	FTP	1248MB / 10500MB
		Oslo	IP sockets	98MB / 3MB
	Internet	Pretoria	IP sockets & FTP	590MB / 2MB
		Reykjavik	FTP	121MB / 8MB
	Internet	Rome	FTP	50MB / 8MB
		Tokyo	IP sockets	1050MB / 250MB
	Internet	Toulouse	FTP	9300MB / 1390MB
	Internet	Washington	IP sockets & FTP	6000MB / 12000MB

Table 1. Implementation status of GTS circuits connected to RTH Exeter

ASOS Configuration & BoM, Australia Bureau of Meteorology

大多数AWS网站使用Telstra 3G / 4G无线服务作为主要通信（627个站点），宽带全球区域网络卫星服务作为关键或重要站点（373个站点）的辅助通信。
宽带全球区域网络卫星系统用作整个网络的第二个通信服务，通过Inmarsat-4网络提供服务，该网络具有99.9%的卫星和地面网络可用性。



WIS (WMO Information System) OpenWIS software (2)
Based on opensource, modern, and proven technology : Java, Linux, App Server, DBMS, etc.
Operational quality software – solid and working Maintained and managed by partners

Some lessons for 24x7

- Failures are inevitable 失败不可避免
  Save and test configuration – ATO case; Planned shutdowns - Switches/Routers, Firewalls, Storage 保存并测试配置 - ATO案例；计划停机 - 交换机/路由器，防火墙，存储
- Built for failures - duplication 专为失败而设计 - 重复
  Prepare for the unthinkable;Keep it simple – easy remedial action 为不可想象的事做好准;保持简单 - 简单的补救措施
- See the unseen with penetrating insights 通过深入见解看到看不见的东西
  Understand the underlying principle;Learn from what went wrong – blameless postmortem; Spread best practices 理解基本原则; 从错误中学习 - 无可指责的事后传播; 最佳实践

Wednesday 29 May 2pm: Andrew Howard. 除其他事项外，Andrew还是NCI的网络和云管理器 - 国家计算基础设施。 它是澳大利亚驻澳大利亚国立大学的研究高峰超级计算机构。 安德鲁一直致力于网络技术和高性能应用，因为互联网是一个新颖的想法，甚至还运行自己的ISP。 本讲座将介绍NCI在其设施内的网络要求和设计，以及它如何在国内和国际上与外部世界连接。 NCI的许多用户通过气候，天文学，物理学，基因组学，地质学等世界上最大的数据集在全球范围内进行合作，不仅需要高性能计算，还需要高性能网络。

National Computational Infrastructure (Canberra Australian Capitol Territory) • Delivering world-class, high-end computing services for Australian

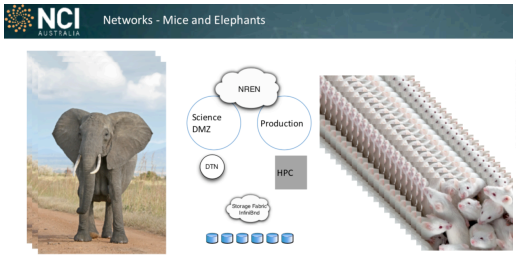
NCI Clouds

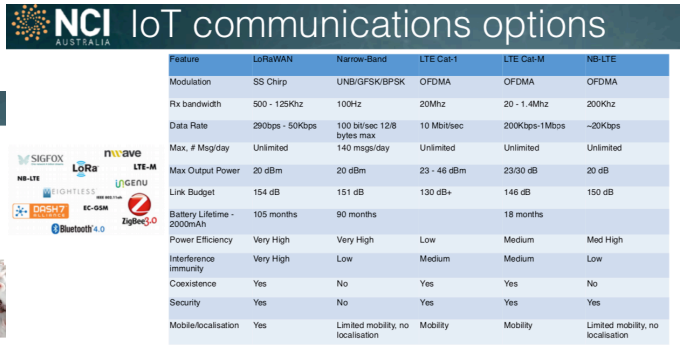
- Supporting a wide variety of research workflows
  - Able to utilise NCI HPC offerings for high throughput computing • IAAS and PAAS
  - National Federated cloud

• Mix of technologies
• InfiniBand for High Performance Computing and High Performance Storage interconnect
• Ethernet for Cloud and external access to Services
•技术组合
•用于高性能计算和高性能存储互连的InfiniBand
•以太网云和外部访问服务

Role of NRENs

• Our National Research and Education Networks are critical • Advanced network services
• 100G
• Anycast • IPV6
• Data sharing services (AARNet Cloudstor) • National service termination point





Friction Free Data movement

- We need to provide our researchers with a friction free data transfer system
  - Easy to use ; Secure using a Federated Access system
- The network and tools should have the data in the right location at the right time
- Able to effectively use different storage tiers
  - SSD
  - Spinning Disk
  - Tape
    - The researcher creates a Data Intent definition
    - Data Source
    - Data Target
      - Transfer priority (High, Medium, Low) • Storage performance (SSD, Disk, Tape)
    - optional Network intersection

By default TCP/IP does not perform well over high bandwidth, high delay circuits.

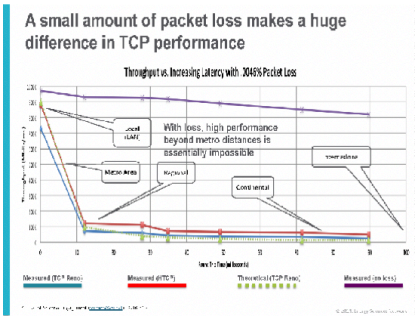
The challenges of the Big Data
如何... ?
manage dynamic and complex workflows for scientific data analysis?
distribute and deploy applications in a flexible way?
Avoid software and vendor lock-in?

Capabilities and Requirements
• Regional connection
• Federated access
• Data capacitor capabilities : Local storage
• Container provisioning : Instantiate toolkit containers
• VM provisioning : Provide VM access on regionally connected DTN

National Research Cloud Overview
• Data movement: • File replication• Object replication • Scheduled and background transfers
• Service endpoints: • Shared capabilities• Distributed data stores integrated into a single metadata namespace
• Build on advanced network capabilities: • IPV6
•数据移动：•文件复制•对象复制•计划和后台传输
•服务端点：•共享功能•集成到单个元数据命名空间中的分布式数据存储
•基于高级网络功能：•IPV6

痛点
•ANU:校园建筑工程中断了冗余
•地理:针对区域距离调整网络协议•小损失对吞吐量有显着影响•需要与沿线组织合作

优化网络的完整路径•与ICM波兰合作，加快ESA数据流



Tidal fluctuation and long gradient causes sewage pipes to break潮汐波动和长梯度导致污水管道破裂

What is LoRaWAN ?
Power, Wide Area (LPWA) networking protocol; Designed to wirelessly connect battery operated ‘things’ to the internet Supports regional, national or global scale networks; Designed to support Internet of Things (IoT) requirements:Bi-directional communication, End-to-end securit, Mobility and Localisation低功耗，广域（LPWA）网络协议;旨在将电池供电的“物品”无线连接到互联网支持区域，国家或全球规模的网络旨在支持物联网（IoT）要求双向通信，端到端安全，移动性和本地化

为何选择LoRaWAN?
远程;免许可证;基于社区的商业模式;低成本;广泛的可用性并支持多个国家频率分配标准

node-red 节点红色
基于开源流程的物联网和数据驱动应用程序编程; 广泛的共享流程库; 支持各种设备，协议和API
将传感器与应用程序，数据库，分析，仪表板和其他服务连接起来

Friday 31 May 9am: Bushra Ismaiel. Bushra是ANU的讲师和导师，也是一系列移动网络技术的研究员。 她特别关注5G等新兴无线网络系统的功能和性能。 本讲座将提供5G的高级概述，其广泛的网络功能，部署 - 以及您可以在何时何地使用它。

4G对5G

•更快的下载和上传速度
- •4G中1 Gbps
- •5G中5-10 Gbps
- •使用mmwave的40 Gbps

•减少延迟
- •4G中大约100 ms•5G大约1 ms
- •延长电池寿命•连接可靠
- •具有成本效益

5G Essential Components

Core Network
- • Software Defined Network (SDN)

Radio Access Network
- • New Spectrum
- • Massive MIMO
- • Heterogeneous Network
- • Internet Of Things (IOT)

Software Defined Network (SDN)

• Centralizes the management of multiple networking devices (routers, switches,etc) into one central control device 将多个网络设备（路由器，交换机等）的管理集中到一个中央控制设备中

- Network Slicing
  • is a type of virtual networking architecture • Separates the control plane and data plane
  • Benefit: It can be configured to support certain use cases. Each use case receives a unique set of optimized resources and network topology 是一种虚拟网络架构•将控制平面和数据平面分开•优点：可以配置为支持某些用例。 每个用例都会收到一组独特的优化资源和网络拓扑
- SDN enforces high Quality of service (QoS), better firewall policies, and seamless user mobility across different technologies. • Scalability SDN实施高质量的服务（QoS），更好的防火墙策略以及跨不同技术的无缝用户移动性。 •可扩展性

光谱

• Support between 400MHz and 90 GHz, often referred as mmWave
• Support full duplex which increases network capacity
• Gives combination of high capacity, wide coverage and high data rates

•支持400MHz至90GHz，通常称为mmWave
•支持全双工，增加网络容量
•提供高容量，广泛覆盖和高数据速率的组合

大规模MIMO波束成形 Massive MIMO Beamforming
- •增强覆盖范围和容量 •通过直接形成减少干扰

波束成形beamforming
- •大规模MIMO可以通过更高的天线增益将覆盖范围提高6-9 dB
- •包括移动电话和设备在内的5G用户设备还将内置MIMO天线技术用于mmWave频率。

异构网络 Heterogeneous Network

•小型蜂窝（Cell）是5G网络的关键
•增加容量，覆盖范围并减少成本

物联网

•设备的大规模连接 •物联网生态系统由支持Web的智能设备组成，这些设备使用嵌入式处理器，传感器和通信硬件来收集，发送和处理从环境中获取的数据。

•应用•M2M •V2V•D2D

5G Wi-Fi共存

•LTE在未经许可的频谱（5Ghz）下运行•未经许可的LTE（LTE-U）•LTE许可协助接入（LTE-LAA）•LTE Wi-Fi聚合（LWA）•MuLTEfire

LTE-U

•LTE信道充当主要信道，而未经许可的信道充当辅助信道
•LTE-U在载波侦听自适应传输（CSAT）机制上工作，其中LTE设备感知信道的时间较长
•当占空比处于ON模式时，LTE设备可以发送，而当占空比处于OFF模式时，LTE设备可以保持静音。
•在OFF模式下，LTE-U允许WiFi设备访问系统。•目前与LTE -Advanced一起使用

LTE许可协助访问（LAA）

•使用LBT（Listen Before Talk）而不是使用CSAT LTE-LAA。•取代LTE-U •它是用单一的全球框架开发的•
缺点：它需要额外安装支持5GHz LTE的硬件
•LTE-LAA最大的问题是与Wi-Fi技术的共存。

LTE Wi-Fi聚合（LWA）

•LTE-U / LTE-LAA的替代技术
•对于LTE流量的传输，LWA使用类似于LTE-U和LTE-LAA的免许可频段，但传输是通过Wi-Fi完成的。
•使用相同的Wi-Fi协议进行传输
•LWA基站可以根据负载和RF条件管理无线资源，从而提高LTE性能
•LWA在LTE频段使用LTE，在Wi-Fi频段使用Wi-Fi，而LTE-U / LTE-LAA则不然。
•它不需要安装额外的硬件，只需要软件升级

MuLTEfire

•MuLTEfire基于3GPP LTE-LAA和LTE-eLAA（LTE增强型许可辅助接入）
•MuLTEfire的表现与LTELAA和LTE-eLAA类似，区别在于它在没有许可锚节点的情况下在未经许可的频段中运行。
•MuLTEfire的主要目标是通过将LTE技术的优势与WiFi部署的简单性相结合，在高密度网络中提供无缝的用户体验。

农村5G

•农村地区的5G容量将减少
•农村地区将获得一种名为“低价”的5G形式
乐队“或”sub-6“5G
•它具有极低的延迟，能够与大规模的工业传感器网络协同工作。
•低频5G可以使用现有的塔和现有的覆盖区域，但其容量的增加可能不足以让家庭使用151GB /月
•大规模的物联网网络将在农业和医疗保健领域发挥重要作用