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 [redacted] ([redacted]-way), with [redacted] – Delays less crucial, win on reliability
• UDP for ([redacted]-way) interactive, real-time – low-delay, low-overheads

Why is it hard?

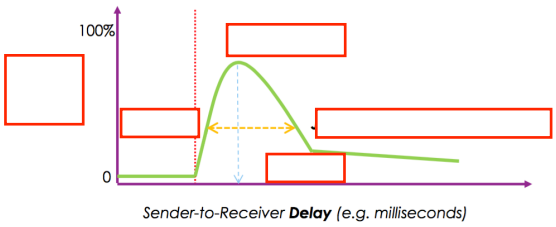
- Real-time media – e.g. videoconference
- Internet is 'best-effort' – *Unless you have circuits*
- Any **delay** is a problem – e.g **variable** delay,**"loss"** delay is another –-<cause>-> Various loss

Network delays

- Sender is sending a constant stream of audio/video samples
  - [redacted] may vary depending on [redacted]
  – Receiver expects to receive a constant stream!
- But all those routers don't belong to us... neither do the paths
- We need the [redacted]), the [redacted], and [redacted] on the [redacted]

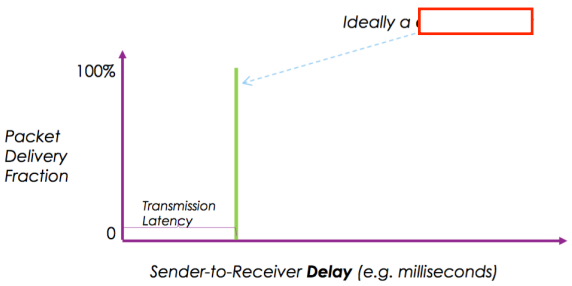
Network delay elements

Tanenbaum and Wetherall



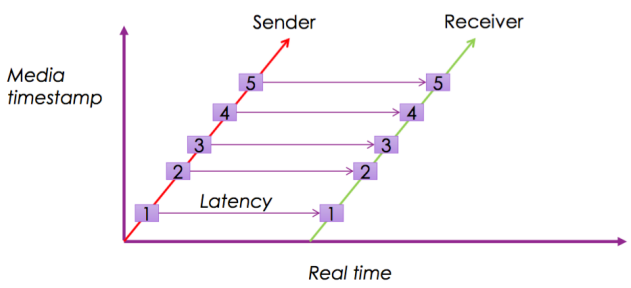
Network delay elements

Tanenbaum and Wetherall



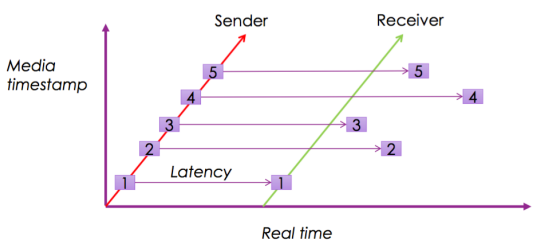
Ideally, sending packets

Ideally, network delay is [redacted] (PDV=0) – like a [redacted]

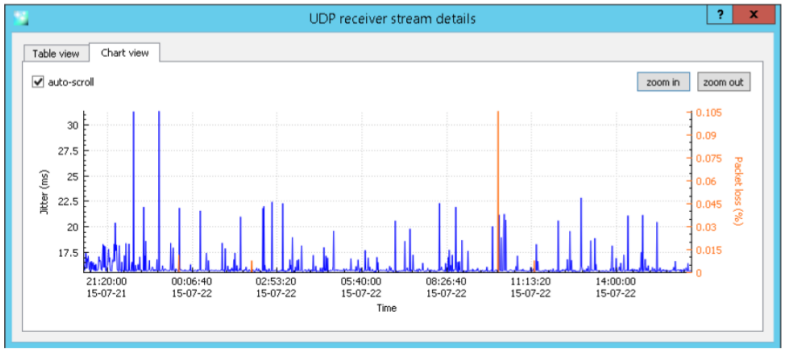


Reality...

Packet delays are [redacted] and [redacted]r can be messy



And jitter changes over time

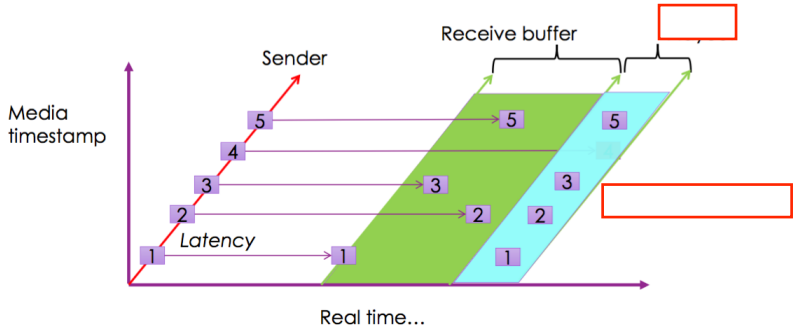


Buffering (TCP-lite)

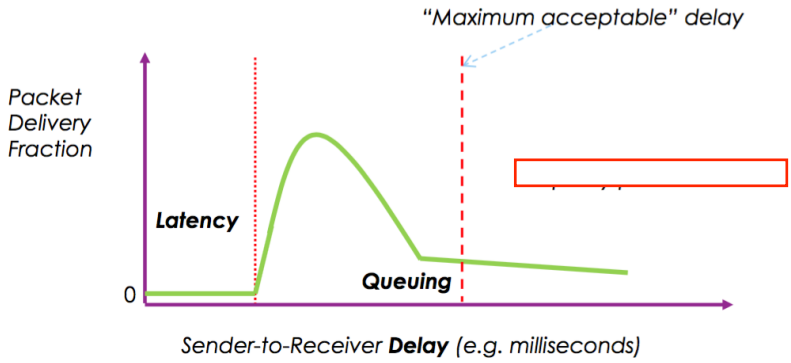
• Sender is sending a constant stream of audio/video samples – Receiver expects to receive a constant stream!

• So Receiver [ ] – smooth out [ ] – Measured in [ ], but [ ]

This solves everything? …



Network delay elements



It's a trade-off

• **Big** buffer:
  – [ ]
  – [ ]

• **Small** buffer:
  – [ ]

– [redacted]
• The smaller the (desired) delay, the harder [redacted]- so you "[redacted]
– Might be ok for [redacted], less so for audio, but ...

Fixes?

• [redacted]
– You know [redacted]
– Request a resend of the problem packet • ARQ -> TCP-like!
– Takes [redacted] + [redacted] + [redacted] => Huge buffer/delays implied
– Generally not done
– In multicast, it may not be the [redacted] who retransmits!
• [redacted]
- When things go bad, you adapt – [redacted]
– And playout [redacted]: Easy to stretch/squeeze audio and video...
– [redacted] buffer when things get better: Want to keep as close to real-time as possible
• [redacted]
–[redacted] for interpolation [redacted]
• Anything missing, you have an approximation .
• Similar to [redacted] algorithms, progressive-display images, ...
– (Adaptive) [redacted]
• Anything broken, you can reconstruct
• Useful for (control) [redacted], or where retransmission is too [redacted]
• [redacted]
– Send several copies at the same time
• At [redacted]) different ([redacted] qualities
Low quality audio/video still better than no audio/video!
And also useful for control [redacted]

Realtime Transport Protocol (RTP)

• UDP payload (or TCP)
• Allows for any [redacted]
• Allows for r[redacted]l, identified, and synchronised – Lip-synch audio to video

RTP Feedback

• As **sender**, would be useful to know:
– How well is [redacted]
• Should I adjust rate, encoding, error-correction, retransmission, ...
– How many endpoints are receiving it? [redacted] • And identify them
• RTP Control Protocol (RTCP)
– [redacted], [redacted] signalling • But need to [redacted] 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 [redacted], multiple [redacted]
• Need to:
– Establish a call: [redacted] *IP)*
– Negotiate the details: [redacted] *P)*
– Deliver the media: [redacted]
– Playout the media, as reliably and quickly as you can: [redacted]

Session Initiation Protocol (SIP)

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

– Doesn't care

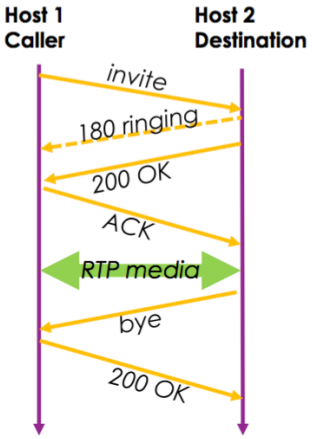
– Is not used by Skype, Messenger, WhatsApp, Facetime, Zoom, Lync, ...

– Competes with ITU H.323

• Widely used for) = 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

– Commands, Responses, Code-classes

– Runs over

– (and not – no XYZ codes?)

Non-realtime realtime?

• One-way media transmission:

– Less interactive

– Play out from a file, not necessarily a capture-device

– (“almost live”)

– Still have issues with bandwidth and jitter

• Now manage playout buffers through

– 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 and – 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 ? can adapt

Or *sigh* use HTTP

Because it gets through

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

Use HEAD requests to learn about

Use GET with requests

Download pieces for the

Can ask server to

Note: No server state required! && Inherit , and

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 - application – (more) reliable bytestreams
– Exchange messages, command/responses, strong client/server relationship
• RTP - 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. Number of devices
2. Ever smaller devices, doing smart/expensive things
3. : Low power, remote locations, widely distributed
4. May need quick commands, responses
5. : 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
    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 [redacted] – That deletes data as fast as it can.
• Standardised by [redacted], 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=v[redacted] • No arithmetic operations
   – Subscribe for reading
• HTTP: monitoring by asking, often, for any given X
Various consumers
• MQTT uses messages
   – As [redacted]
   – As 't[redacted]' (by listening clients)
   – So s[redacted] **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
  – 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…

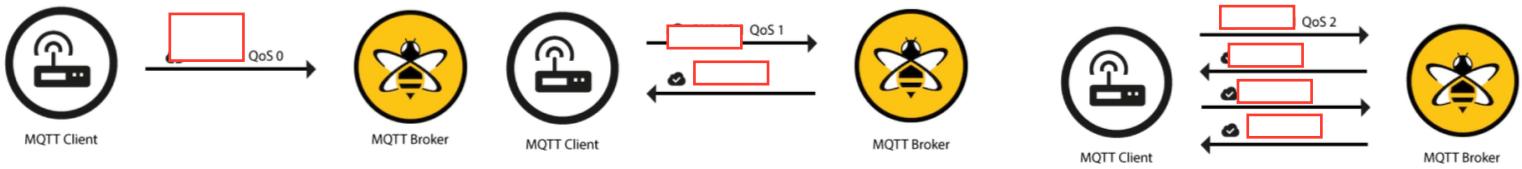
QoS

• “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
  – 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
  – Even across
  – Sent on subscription by

MQTT – "Last Will and Testament"

• Sources can publish a message for any topic(s) (e.g. client/status) – A message,
but not sent
• If server e.g.
  – Does not receive a after <> period, or
  – Sees ( ) connection dropped without (MQTT) disconnect, then
  – Assumes connection is lost, source has failed, ...
• Can then inform (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**:
• **Not clean:** Ask server to remember you – aka session – In case you drop off
• Pain for the server:
  – Store all your topics
  – Store all (with QoS 1 and 2)
  – Push in burst when – 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
  – Able to modify behaviours all in software, no hardware changes needed
• Connect topics by a
  – Given X (is published), do Y (publish something) [e.g. NodeRed, IFTTT.com]
• Overseen by a **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!)
• Access Control • Certificates
• Payload , integrity • **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 (e/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

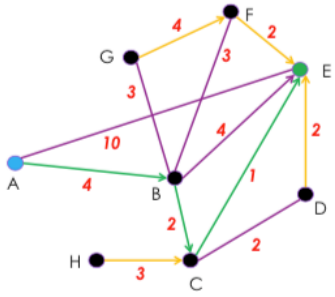
Shortest-path routing(“lowest-cost” path routing)

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

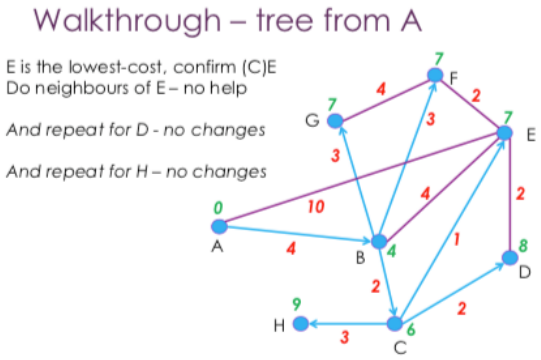
• Union of all shortest paths towards a node from each source
• Consider E’s sink tree. – ABCE = shortest for A, B, C – Work out the rest
Source tree = sink tree <Usually, but doesn’t have to>
• Asymmetric costs => different trees • What happens if BC=2, CB=5?
Regardless of where you start, routing decisions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

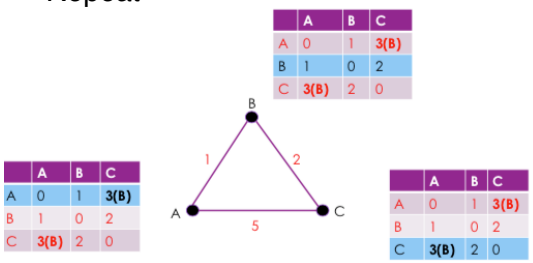
• Has some scaling issues in complex networks
– Imagine 1000+ nodes, 1000’s links
– Imagine changes at any single point
– Lots of research...
• Needs complete topology – At each node/ source

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
  –
  –
• 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:
  – the network
  – the topology
  – 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.

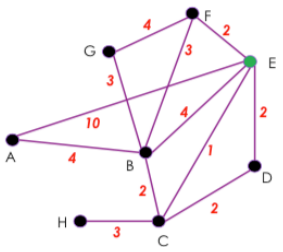
- Just keep track so you don't repeat yourself, or cause storms
  – Use incrementing sequence numbers – May get multiple copies, deal with it

- 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
  – Rather than
Not 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
  – **Bad** for j (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 ( )
Less balanced, but more predictable

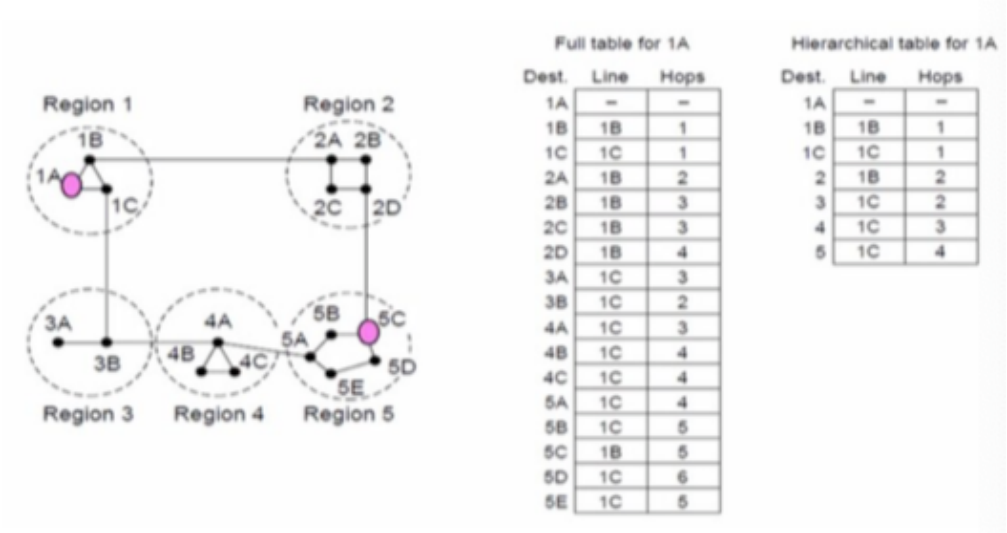
Hierarchical routing

• Scaling problems as identified earlier
  – growing
  – growing
  – growing
• Network aggregation
  – Already have aggregating a
    • 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:
–
– Full 1A to 5C[1B] = hops – Hier.1A to 5C[1C] = hops

• 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 , 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

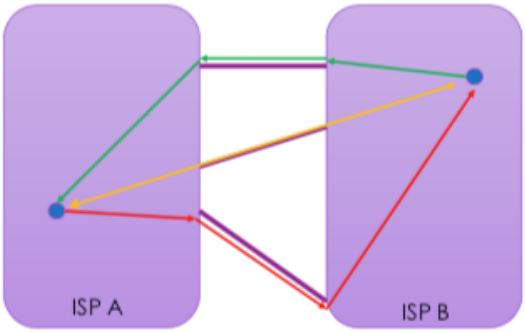
• i.e. we have POLICIES to add to our protocols
  – E.g. National Research and Education Networks have an R&E traffic policy
  • Wholesale purchase **AND** don't compete with commercial providers **AND** social good

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:

- 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: P

- 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..

(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 (or) which run BGP

– Creates an edge between interior and exterior routing

• Aggregates nodes within an ' ' (AS) – Think a region, a business, an ISP

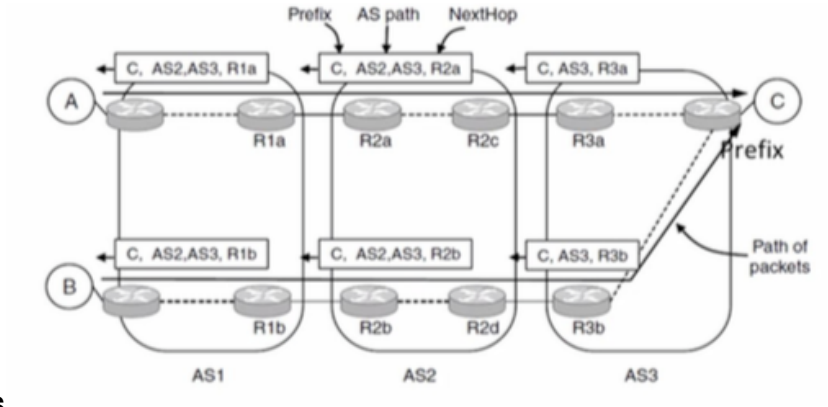
BGP is more DV than LS

Instead of Distance Vector it is a

• Announcements:

– I

– Path: list of to transit:allow loops to be detected and removed • No distance



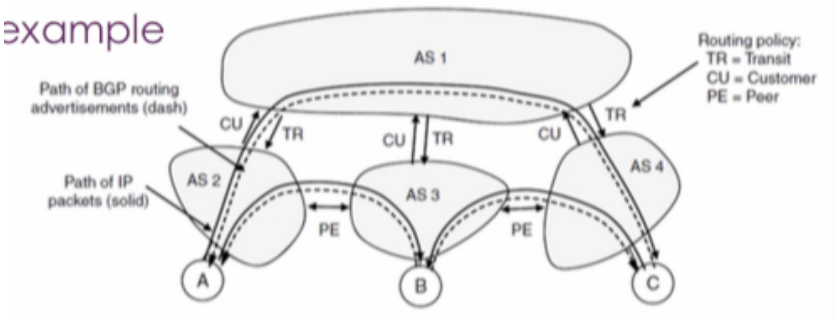
BGP route advertisements

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)=**
- **Single segment per RTT = packets/s**
  Typical packet on Ethernet? Say 1000 bytes = ~10,000 bits -> 100kb/s or 10% of link • Even if bandwidth goes up, doesn't!

Sliding Windows

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

If(lost) then: ARQ – "Go Back N"

• buffers just a
• If it's the , ACK it, everyone happy • If it's not, it, *I just don't care*
• Let sender retransmit what I'm actually waiting for
• **Sender** has a r. After timeout, • Really simple, but somewhat inefficient

ARQ – "Selective Repeat"

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

Very sender/network oriented

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

Flow Control:S on the side

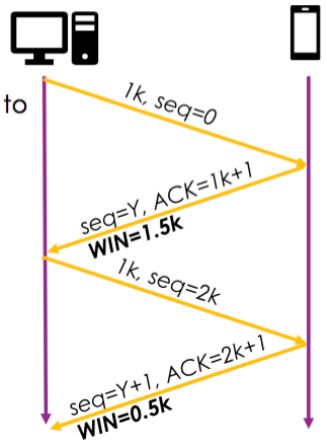
• layer:
  – receives the segment from the network – and adds it to
• calls recv(N-bytes) to read from buffer
  – But what happens if the application is slow?
, fill ( ) buffer – and eventually application recv()s
Sender Sliding Window (**W**) – Both sides know
• Receiver = Window (WIN) – Number of "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

Sequence numbers

Acknowledgements

But

Simple Flow Control



Congestion

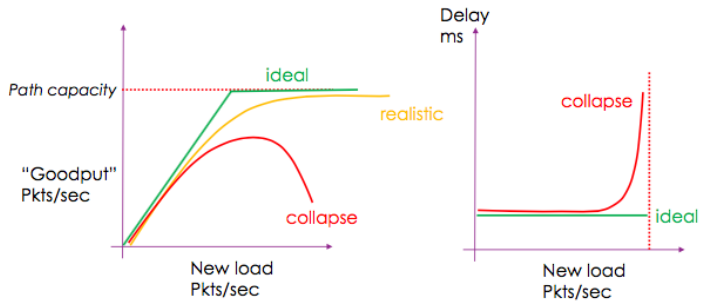
- A traffic jam – something filled up and is holding up the rest
  - A dynamic condition
  - Somewhere (unknown) along the (unknown) path
- Senders keep sending
  - – Makes it worse, for themselves, and everybody else – Congestion -> loss
- It is not the links that "cause" loss (by being congested) –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

Too many inputs for the one output

Router Buffers: Queues...

-) queues on every interface
- Great for absorbing (short) bursts of traffic
  - – > t
- For a while... then queue overflows, and packets
- 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:
• [redacted]: get as much as I can, without causing [redacted] and
• **Fairly**: [redacted]

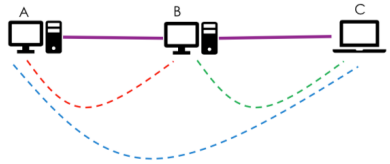
Who handles that?

• 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 [redacted] 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...

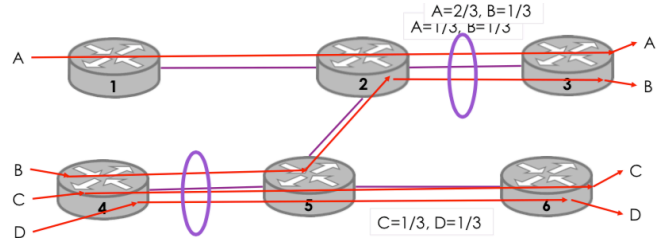
Network bottlenecks – unequal paths

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

Max-Min Fairness

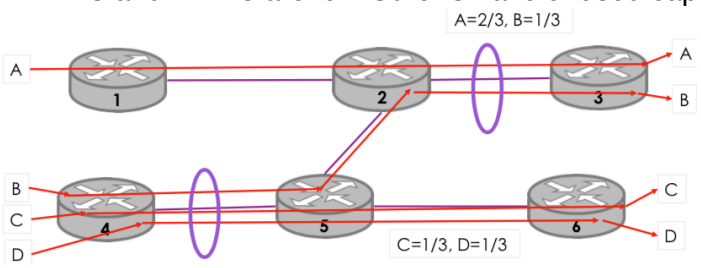
• Allocating bandwidth such that :
• “increasing the rate of one flow will decrease the rate of a smaller flow”
– “Maximising the minimum” – keep adding, and sharing what’s left.

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

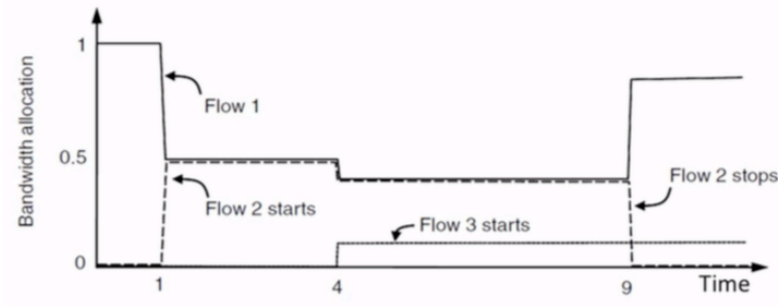
– Host manages the allocation (use)– Network policing is strong, but inflexible

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

• Transport layer (TCP) modifies

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

• *Additive Increase, Multiplicative Decrease (AIMD)*

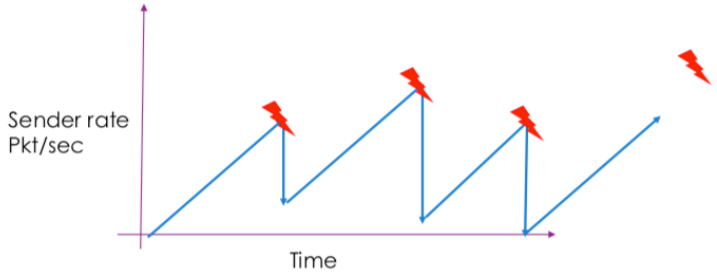
– Senders

– Senders

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 . [and not related to Flow-control Window (WIN)]

Getting started

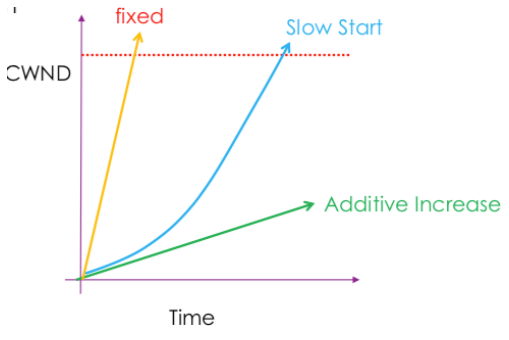
• On initial TCP connection, what is CWND?
– Guess? Too many variables (bandwidth, delay, congestion, ...) – Pick something? Could be way under, or over.
• TCP Additive Increase (on start):
– Start with CWND of N bytes (~1 packet).
– Every round trip without loss, make CWND bigger by 1 packet
• Increase 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

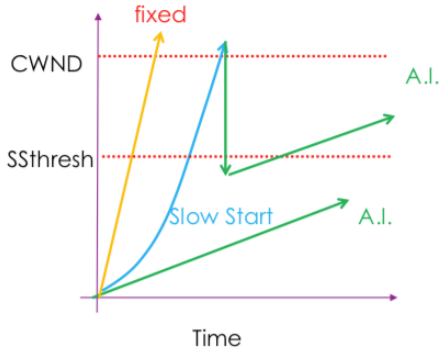
• Get to the right CWND more quickly
• But will still go (suddenly) over it
– Get packet loss/feedback
– Multiplicative decrease (big drop in CWND)
• So combine Slow-Start with Additive Increase:
– Initial connection, get MD'd down. Below right CWND, but still close? – Define a threshold: ss-thresh =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

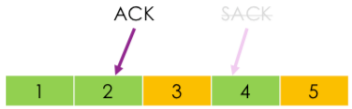


Combined behaviour

- 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)

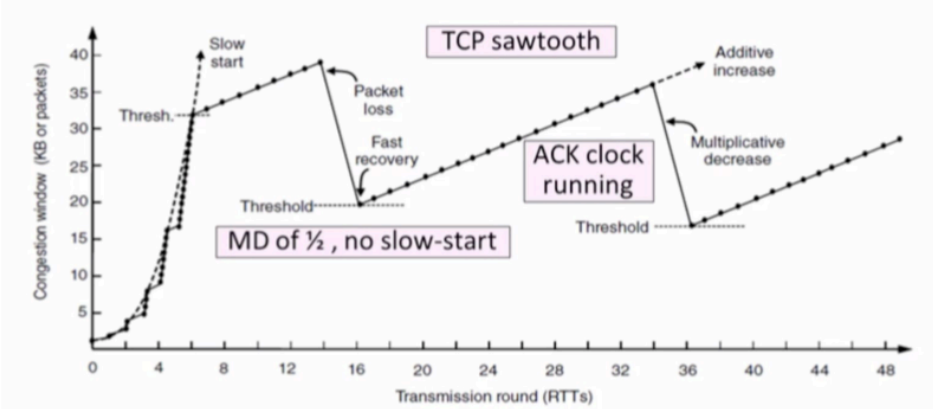
• MD the CWND (1⁄2 it!) and then continue sending (**Fast Recovery**)

– 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

(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

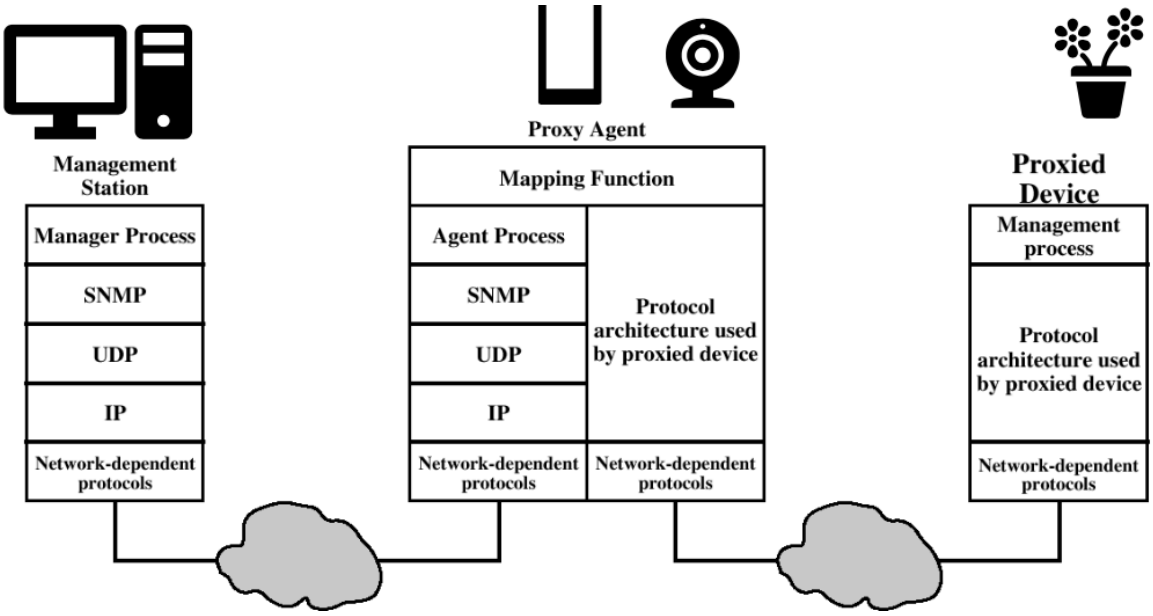
• An application framework
• For managing/monitoring network resources
• Components of SNMP:
–
maintains configuration and current state in a database.
Proxies: an agent that talks with non-SNMP devices
–
to query or modify the database at the agent.
Part of Network Management Systems (NMS)
– s the database)
MIB, MIB-II (RFC 1213) – and millions more
Structure of Mgmt Info (SMI) defines sets of related objects in a MIB
–
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
  – s, – T since s
  tart-up – Strings,

- "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: A
linkUp:
coldStart: system crash)
warmStart: ual 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 –

• 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

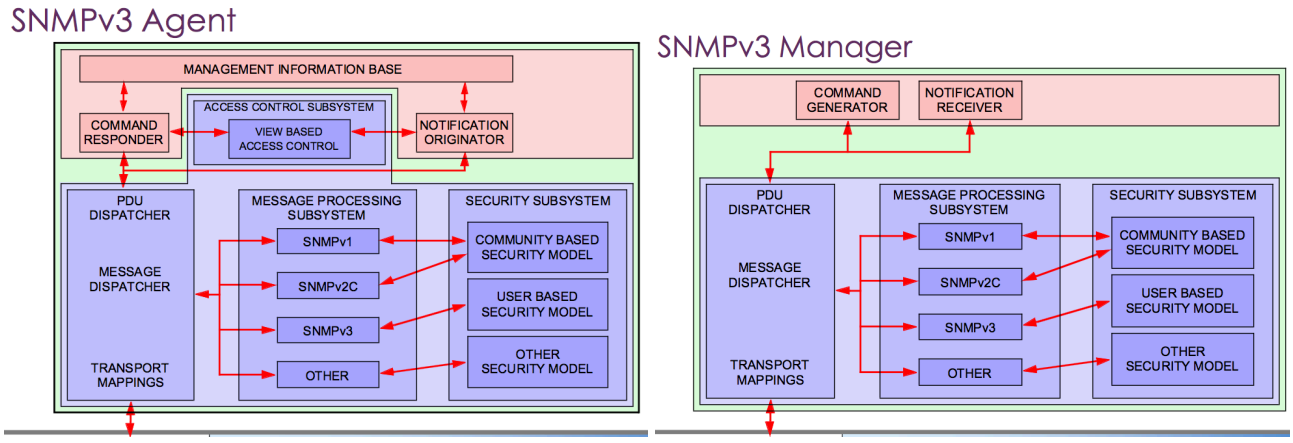
• Three versions in use today:
  – SNMPv1 (1990)
  – SNMPv2c – [and three more] (1996)
    • Adds “GetBulk” function
    • Adds federated monitoring capabilities (manager to manager)
    • Adds TCP transport option
    • Adds 64bit counters
  – SNMPv3 (2002)
    • SNMPv3 started from [redacted] (and not SNMPv2c)
    • [redacted] 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 ([redacted])
  – Ensures that a message is from a valid source ([redacted])
  – Ensures that a message cannot be read ([redacted])

SNMPv3

• Has three security levels:
  – Depending on how you connect – you get more access rights
• [redacted]v: Authentication by matching a [redacted]
• [redacted] Authentication with message digests.
• [redacted]v: Authentication with message digests, and encryption



What are we GET/SETting in those packets?

• Values stored in a [redacted]
  – Collected under a [redacted])
• 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...

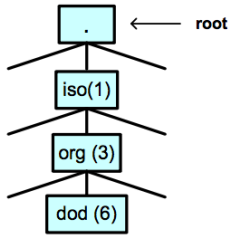
- Reading Counters/Gauges tell you about "now"
  - **Counter** e.g. packets on an interface (can wrap)
  - **Gauge** e.g. memory/disk space (ranges between zero and <maximum>)
- Agents don't have history, and don't – Agents only have a temporary clock - Time since boot
- Managers have to ask , and make 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 – (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'

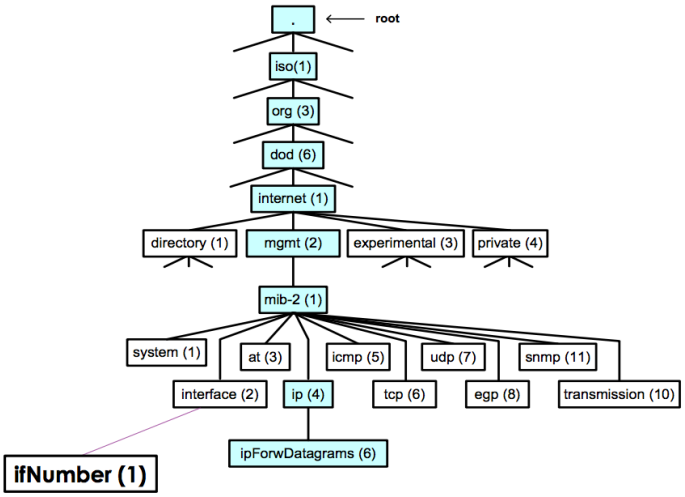
ASN.1 OBJECT IDENTIFIER (MIB)

- 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 [redacted] – and extensibility
• OIDs provide [redacted] for [redacted]
• 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 [redacted]
  – Get-Bulk("interface") ==> every row, every column
  – But only one UDP packet comes back
  – Error response "tooBig" (64kB UDP limit)

SNMP beyond my [redacted]

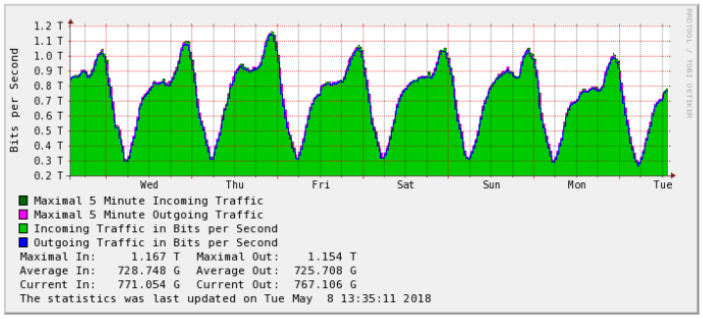
- SNMP in the wide area is ... unwise
  - – SNMPv1/v2 agents should not be visible. Ever.
  - – Lots of traffic
  - – Easy to scan/map network
- Becomes a human problem

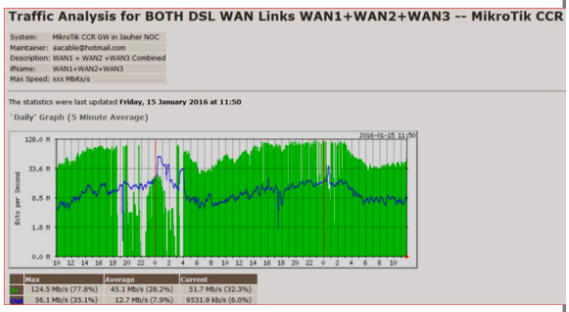
– Need to ask for favours

- **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

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

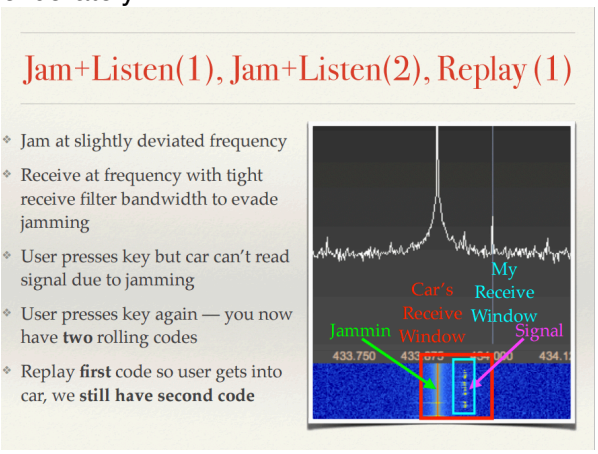
Security = risk management

• 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 [redacted] – really weak – easy to snoop and decrypt
• [redacted] – Nearly impossible to brute-force decrypt on today’s computers
• Removed one threat, but e.g
– Could have [redacted] – gets attacker on the WLAN
– Could have [redacted] access to devices on [redacted] – gets attacker onto LAN/WLAN
– [redacted] 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 ‘[redacted]’ can be undermined by security ‘[redacted]’ • 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 [redacted]
– Confirm [redacted]
– Validate [redacted]
• 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 vs Asymmetric

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

Key distribution and performance

• Trade-off: which approach?
• **Symmetric**
  – Encryption is – great for
  – Key sharing is hard:
• Need to get (different) key to everyone who needs/deserves it • For every new conversation
• **Asymmetric**
  – Encryption is – 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 (see certificates)

Best of both?

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

Authentication and Integrity

• 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
  – Use (established through)
  – Calculate a summary of the (signature, message digest, hash) • And encrypt that with key or

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: (or other ‘nonce’) in the message/signature – Before – (Application requirement, not part of crypto)

Applying cryptography

• Each application can now build their own:
1. 2. 3. 4.
• 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 Layer

• SSL (1995/96 SSLv3) – Netscape browser
• Triggered by HTTP ==> HTTP/S ( ) = **https://**
• Led to generalised Transport Layer Security (TLS) [V1.0 1999, V1.1 2006, V1.2 2008, V1.3 2018]
= 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
  – of (padlock icon in www)
  – exchange • with confidentiality, integrity, authentication and freshness
• Starts with phase
  – To establish and
  – Before any (HTTP) is exchanged
• Client needs to some random new server
  – Network traffic can be spoofed and misdirected
  – Needs server key, **for sure**... provided by a

Certificates

• **Bind an to a** – Server/service, or person
• Requires somebody authoritative to say so
  – (CA)
  – standard: Metadata about identity, plus , encrypted with

Public Key infrastructure

• Can’t have just **one** – Too busy to deal with the whole Internet
  – Too big to fail
  – Too obvious a target
  – Too easily untrustworthy and a monopoly
• Build a
  – 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
– Around 70 **root** certs in Chrome today – and cache many more – That's a lot of trust...
• On initiation, request certificate
– And check its signature • And check its signature – And check its signature• ... up to the signature
• Nice, till somebody gets hacked
– key is exposed
– Certificate must be revoked
– has a (CRL) – this does not scale well!

That solves everything?

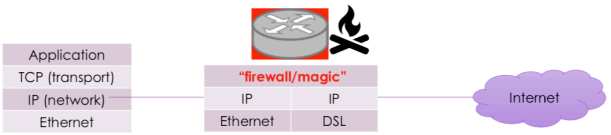
• Not even close
• SSL/TLS:
– And has itself been broken a few (12+) times
– Code flaws, crypto flaws, interaction flaws with other protocols, spooks, ...
– Not used by all applications, or other layer protocols (DNS, DHCP, BGP, ...!) • Recall DNSSec discussion earlier
• If you need it, use it–or get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

Firewalls

• Edge routers/gateways/processes that (can) explicitly block
• 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"

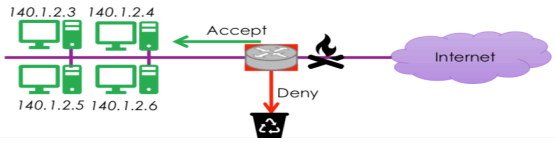
Routers (normally) just look at IP – but...



Policy time

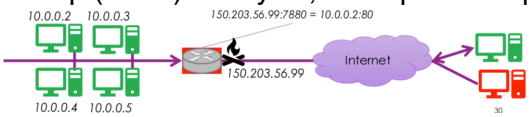
• Establish rules
– Break applications we don't like – Very high level
• Packet filtering is -level
– Doesn't look at

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.



Firewalls

• Change the rules based on other
– Track between
– E.g. NAT: Allow i from to our that
• But timeout after idle...

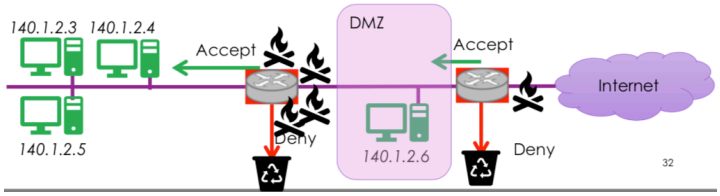
firewalls

•

•
– Will from packets
– Understands content • E.g. viruses in emails/web pages
• And

Firewall deployment

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



Firewall implementation

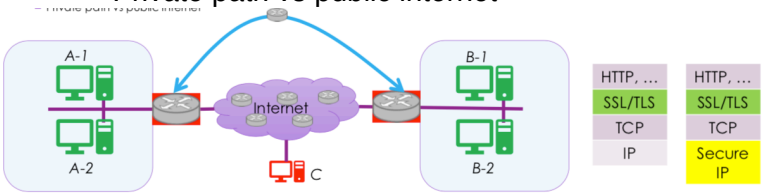
• Dedicated devices – Especially Application/DPI Firewalls
• Routers/Modems
• Wireless Access Points
• On hosts, in the Operating System (e.g. Linux IPTABLES)
• Tradeoffs
– Multiple firewalls = more **AND** more
• More places to maintain and coordinate
• Protect

Firewalls = Islands of trust

• doesn’t hide everything
– 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

•
– 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

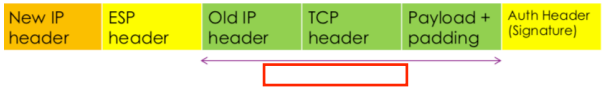


• Can we use the Internet?– Need security at the IP layer
• Make a “virtual” leased line –
VPNs :• Tunnel (=) across the Internet ( )

IP tunnelling security

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

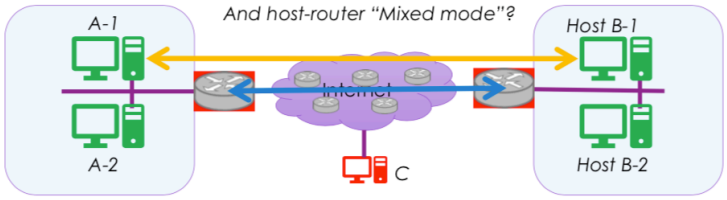
• Use to establish secure VPN connections
– Cryptography at the network layer
– Keys are exchanged between (can be and/or) – Packets are
and



(ESP =)

VPN endpoints

• : router to router (i.e. forwarding)
– Connects whole subnets transparently – make them look as one – Can be NAT-
• : host to host (i.e. forwarding)
– Different format, only encrypt , NAT-

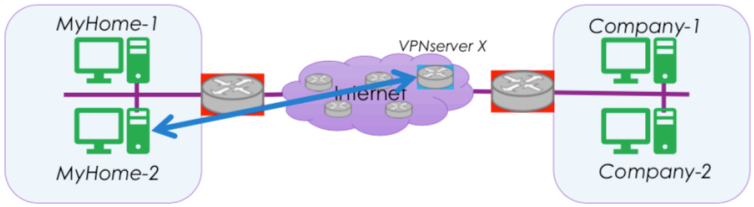


Address transparency

• New packet gets IP address of
– 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

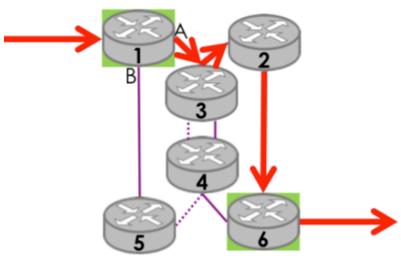
• A VPN is not necessarily the
– Just where the packets 'emerge' from the tunnel
– ' is the tunnel endpoint • Responses need to go back there
– IP addresses are allocated to geographical regions • This has some benefits...



Looks a bit like a circuit?

• Well, yes.
• But only the endpoints are tied down
– handles everything in
• Dynamically, politically, ... • Deals with failover, multi-path, ...
– Packets are labelled with
– when either endpoint dies
• Compare with MPLS
– Multi-protocol Label Switching (back in T10)

–
–
–



Device security

• Switches/routers have physical and virtual interfaces
• Remote access
  – [redacted] ([redacted])
  – Operating systems ([redacted])
  – 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
  – [redacted] copper is [redacted]
  – As transmitter (leak) and receiver (interfere)
• Some Layer-2 security

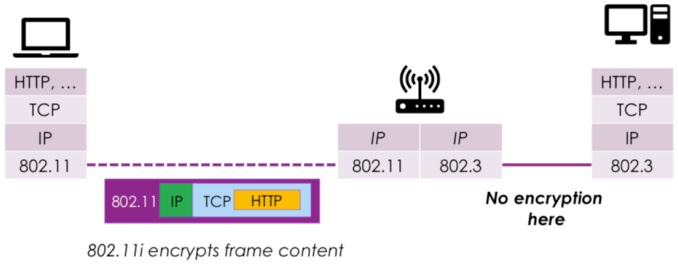
Fibre security

• 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
– [redacted] leak, fibre jackets leak

Wireless security

• By definition, wireless is [redacted] – Narrow beam [redacted] 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 [redacted] (WPA1-3)



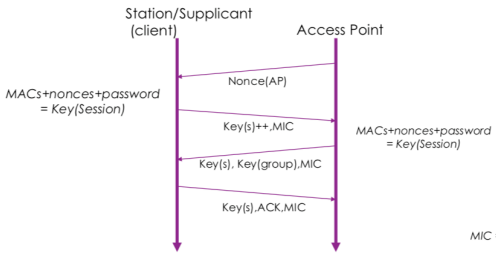
Consider a WiFi Home network
• Typical(un-open)[redacted](key)[PSK]

• 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
• Client to AP
– Each calculates a shared s from the • (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 with – Confidentiality, integrity, and authenticity

Keys on keys…



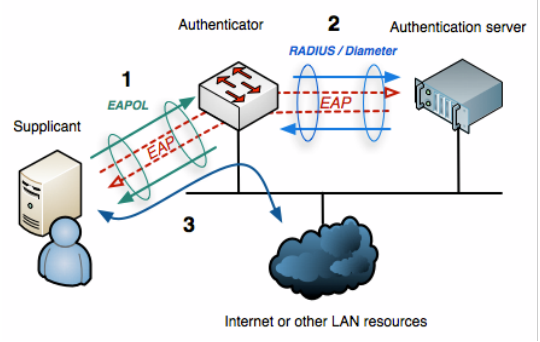
Keys over keys on keys with keys…

- 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

•
– Uses (EAP) • Can be used on 802.11, 802.3, etc.
• Typically RADIUS back-end
– Each client has own • No



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

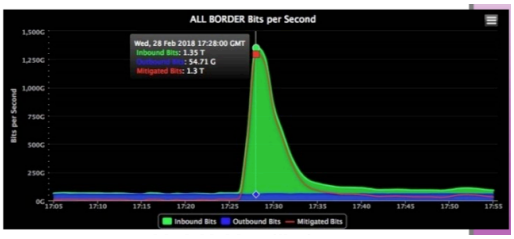
DoS vectors

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

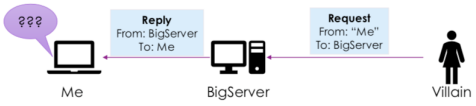
Distributed Denial of Service

• 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

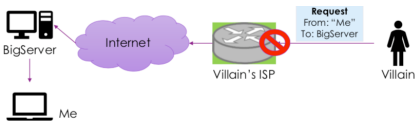


"Spoofing"



• spoof
• Sending packets with – Get other boxes to work for me
– Very hard to

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** – [redacted] – often with [redacted]
- SMURF (and variations)
  - Ping the [redacted] (one packet out)
  - Use the [redacted] as source (spoofing)
  - [redacted] (many packets back) – Easy to prevent...

Packet multiplication

- [redacted]
- 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
  - – [redacted]
  - – Better filtering support (e.g. DoS from a particular country?)
- Upstream provider support
  - – Can re-route most/all traffic elsewhere
  - – E.g. [redacted]
- Get ingress filtering everywhere! – And fix all those webcams while you’re at it...