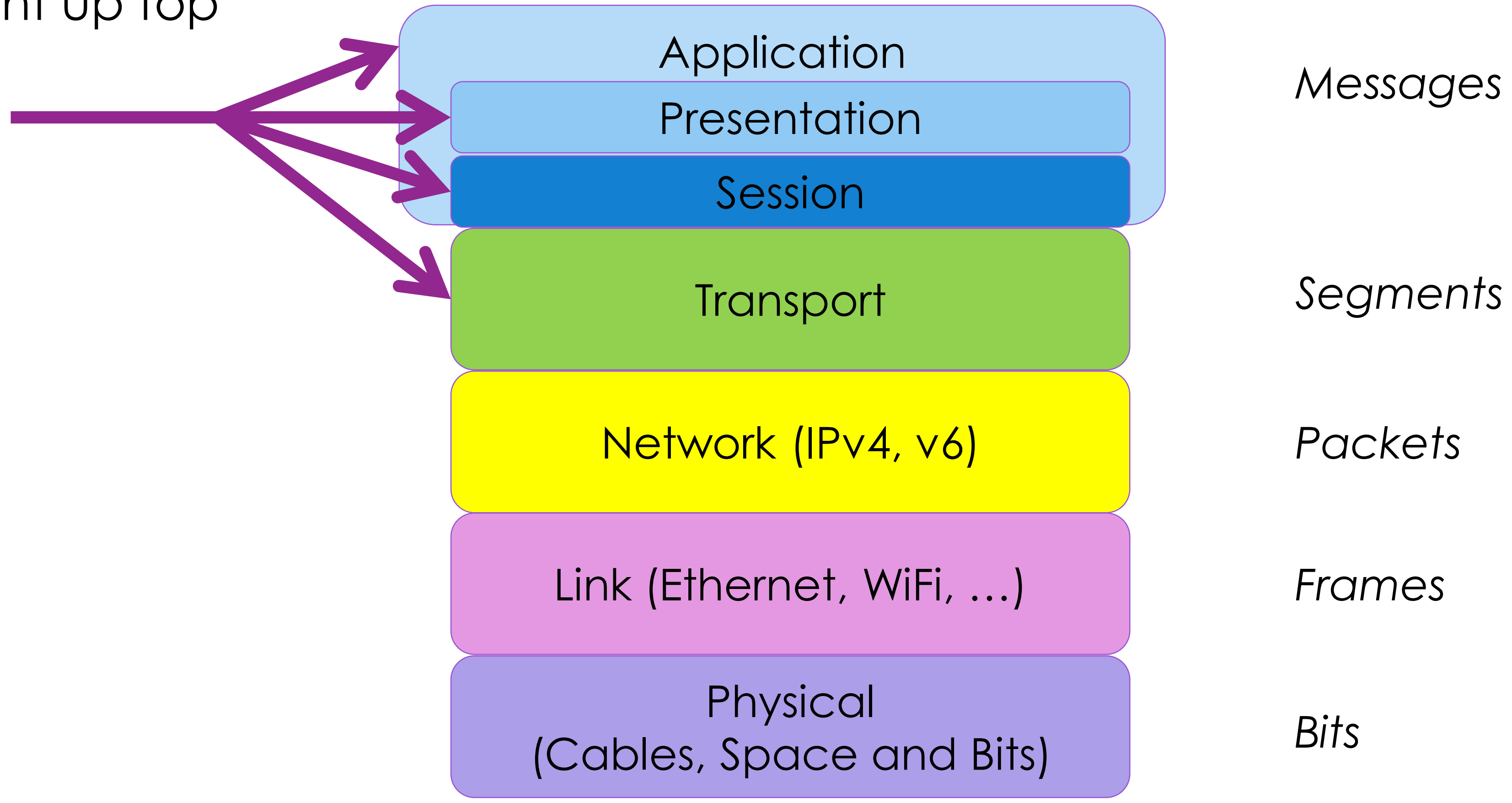
COMP3310/6331 – #16

Realtime communications

Dr Markus Buchhorn: *markus.buchhorn@anu.edu.au*

Where are we?

- Right up top



Applications choose their transport

- UDP-based applications:
  - Short messages
  - Low(er) delays
  - 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 bytestream

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?

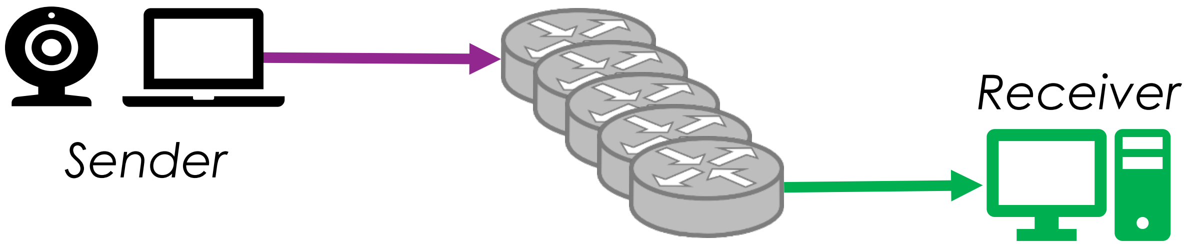
- Very application specific
  - How much can you tolerate a single lost packet?
  - How much can you tolerate a delayed packet?
  - How much can you notice a delay?
    - E.g. streaming vs videoconf
    - And different audio/video delays (synch) are worse

- 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 – **variable** delay is another, "**loss**" is another
  - Various kinds of loss

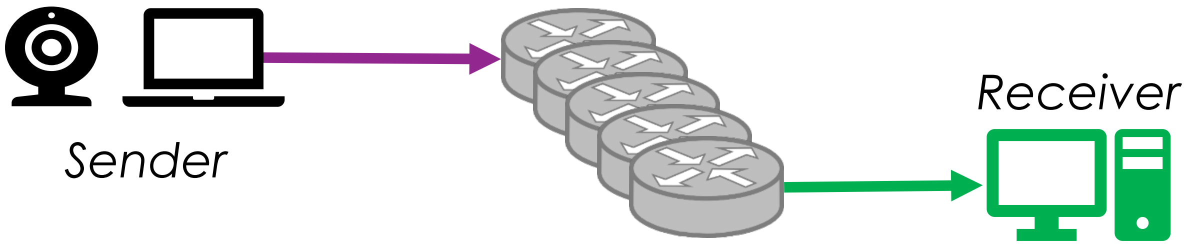


Network delays

- Sender is sending a constant stream of audio/video samples
    - Bandwidth may vary depending on codec
  – Receiver expects to receive a constant stream!

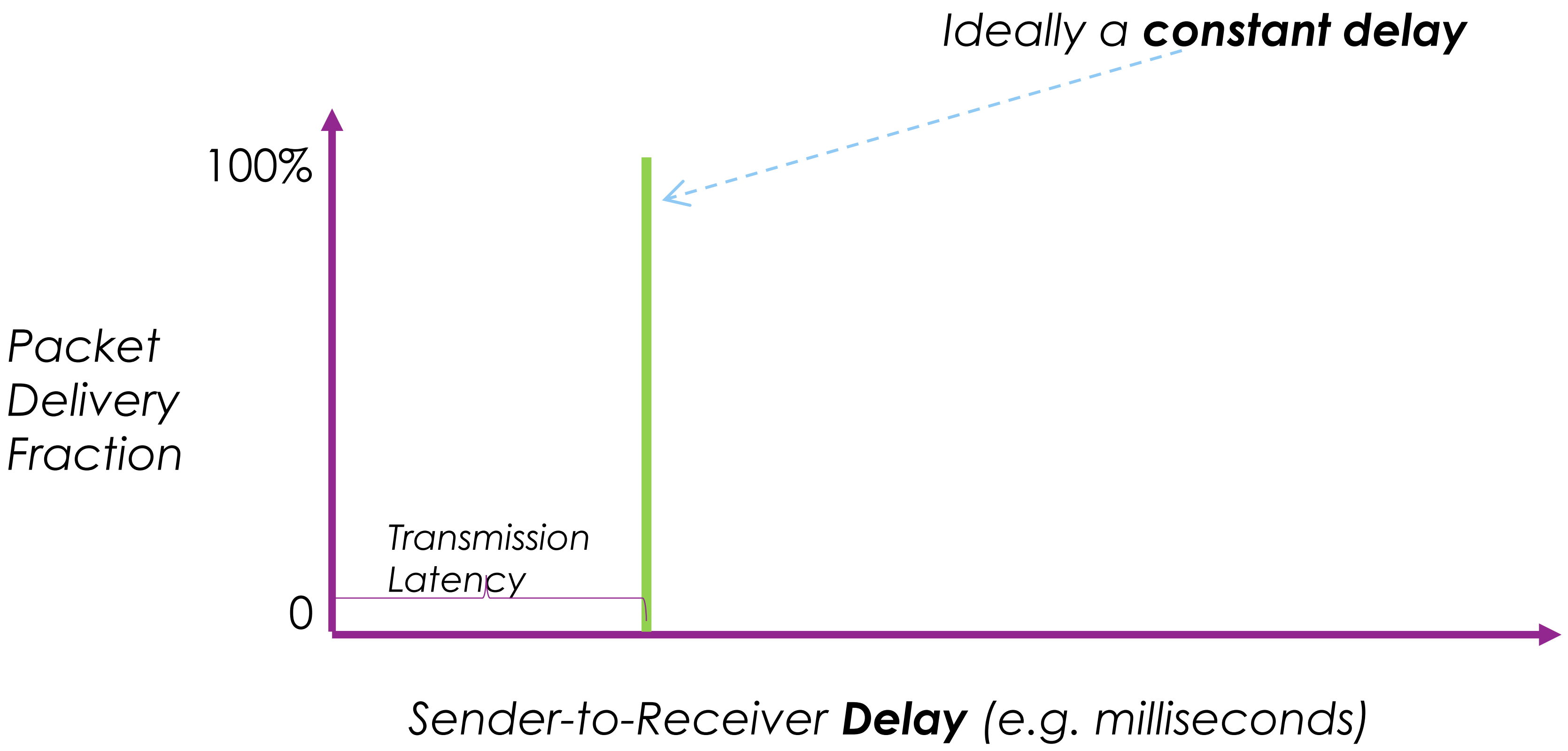
- But all those routers don't belong to us… neither do the paths

- We need the path(s), the bandwidth, and capacity on the routers



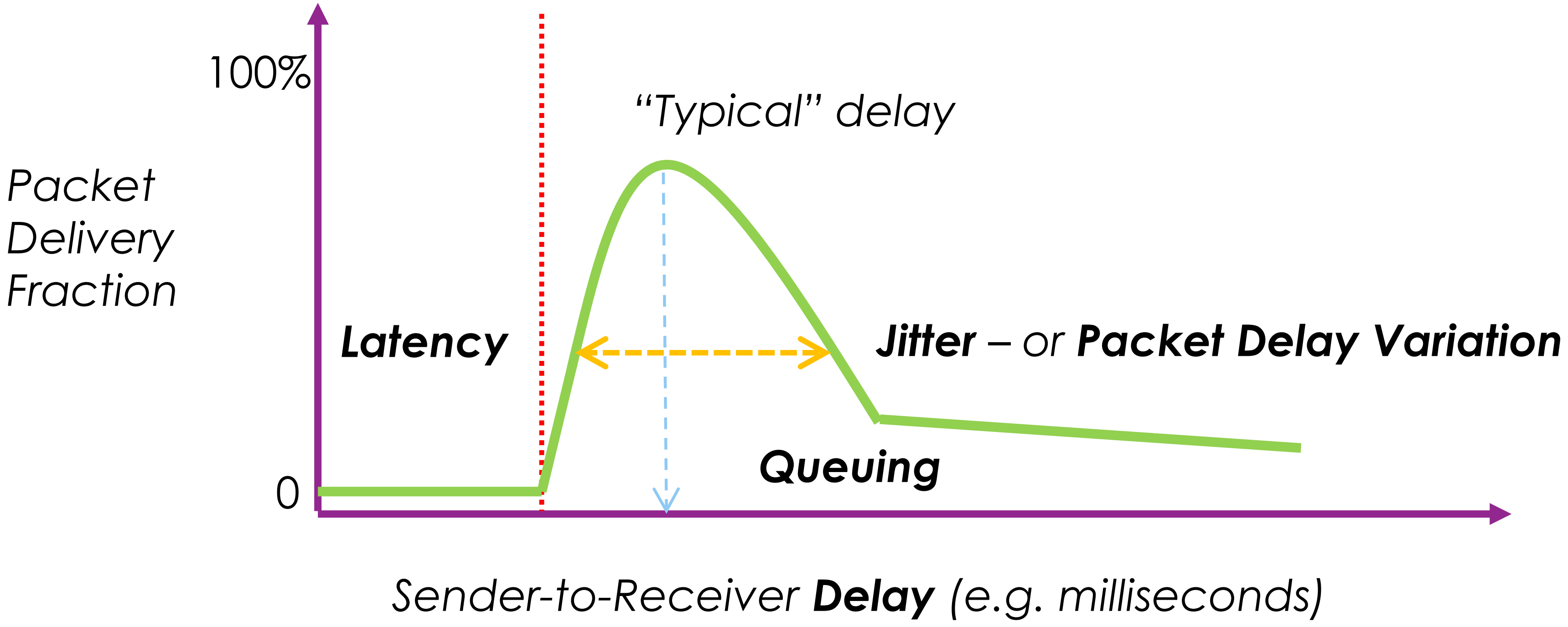
Network delay elements

Tanenbaum and Wetherall

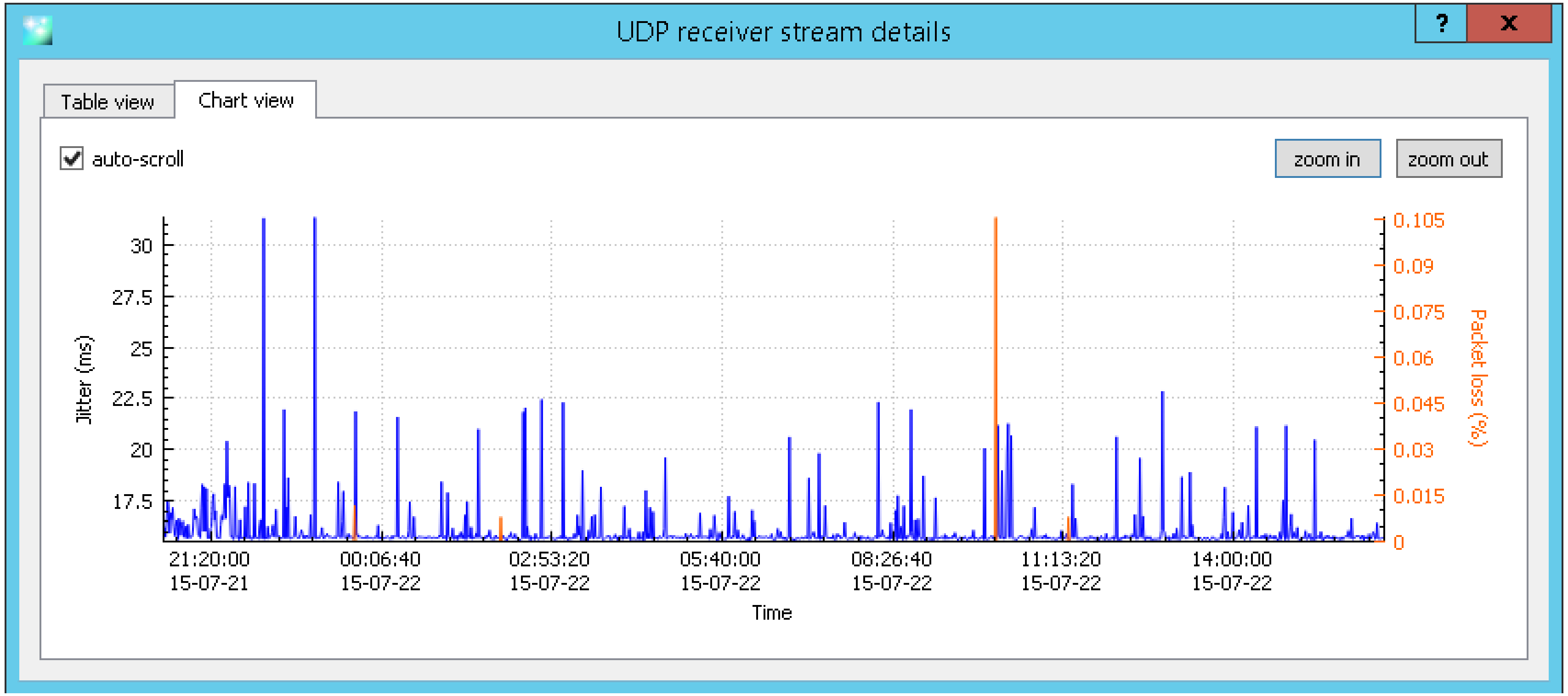


Network delay elements

Tanenbaum and Wetherall

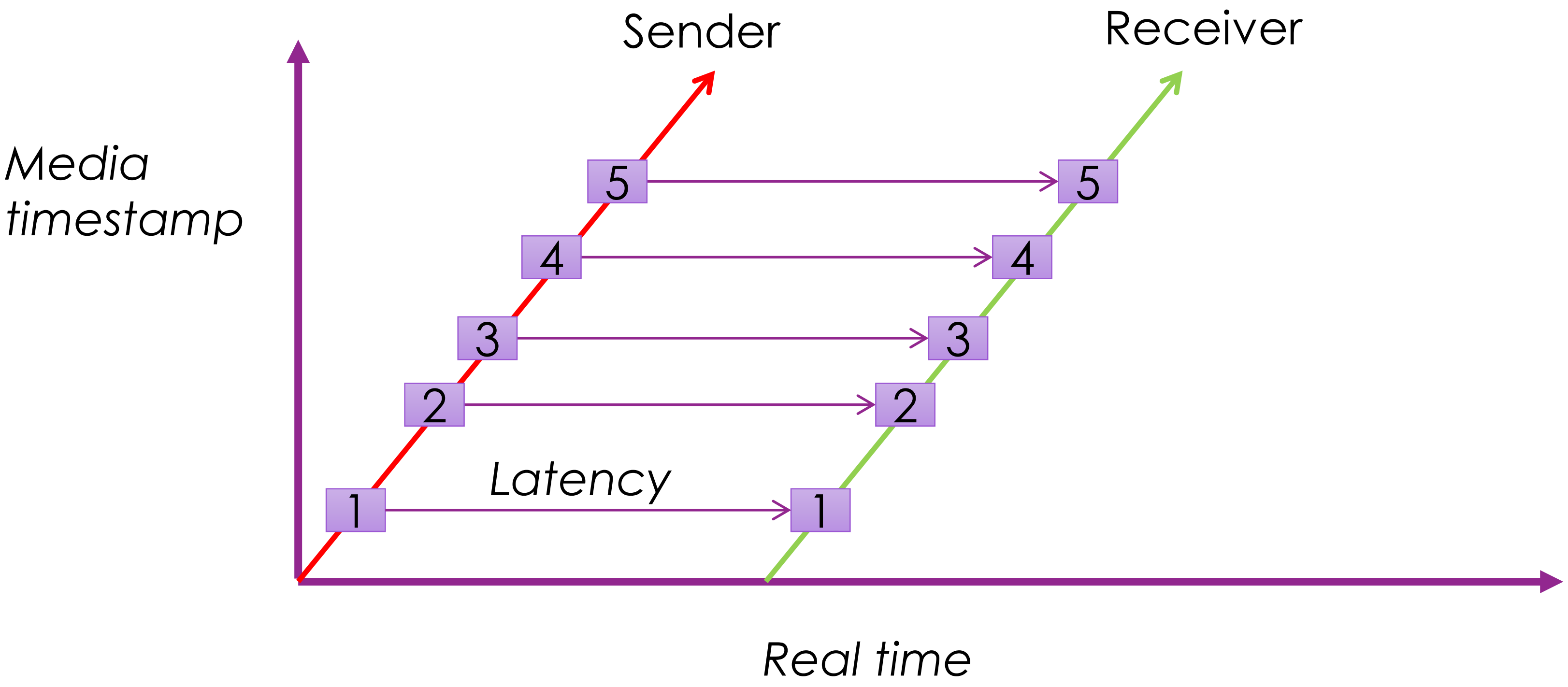


And jitter changes over time



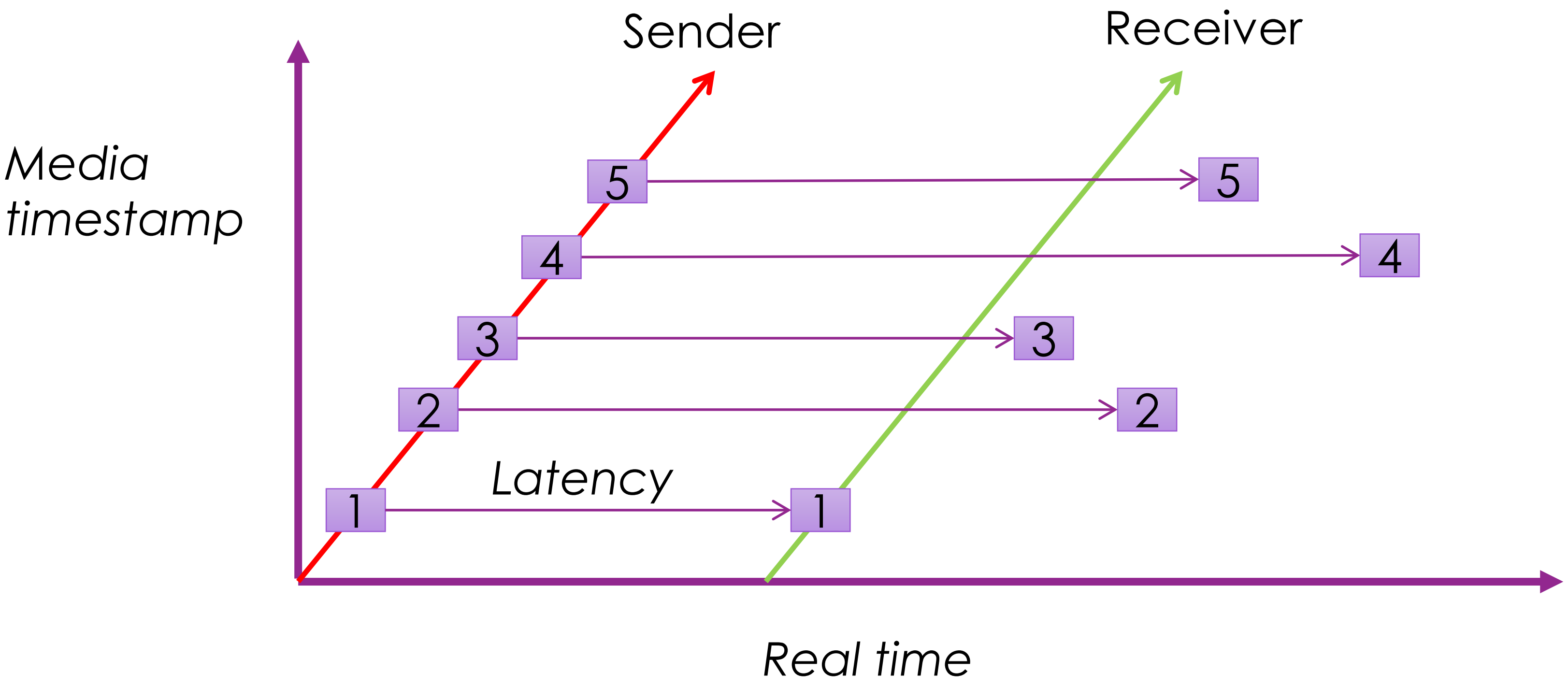
Ideally, sending packets

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



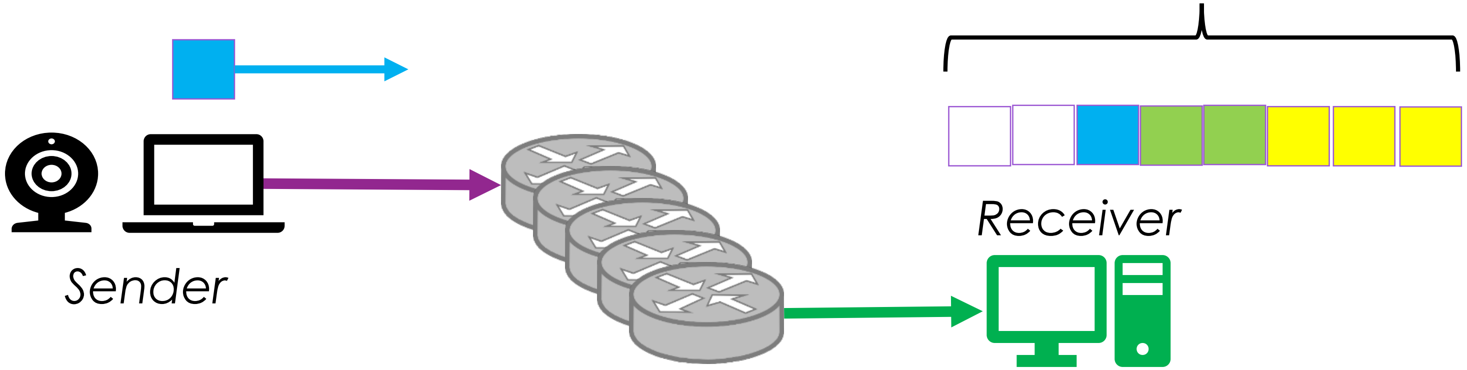
Reality…

Packet delays are random,
and packet order can be messy

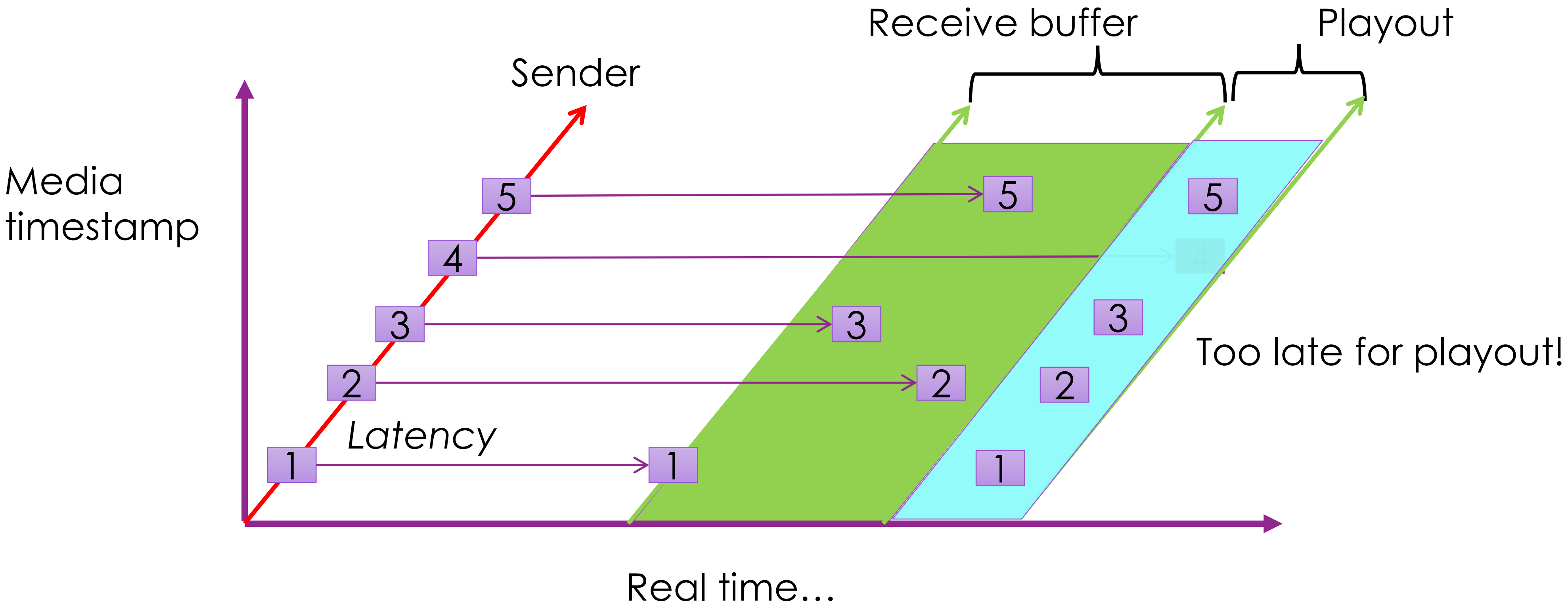


Buffering (TCP-lite)

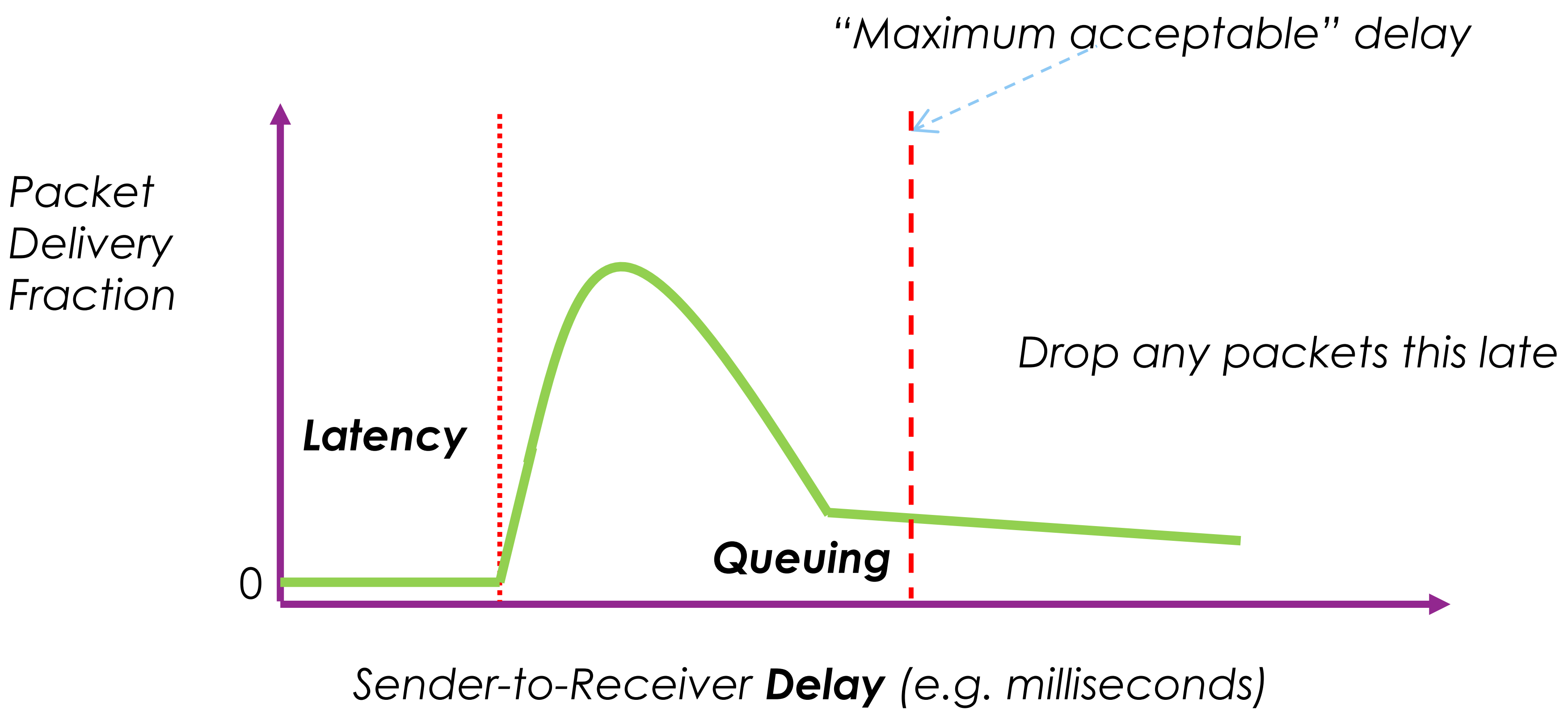
- Sender is sending a constant stream of audio/video samples
  - Receiver expects to receive a constant stream!
- So Receiver makes a playout buffer – smooth out variable delays
  - Measured in bytes, but effectively in time



This solves everything? …



Network delay elements



It’s a trade-off

- **Big** buffer:
  - Fewer packets lost <u>to delays</u> = 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

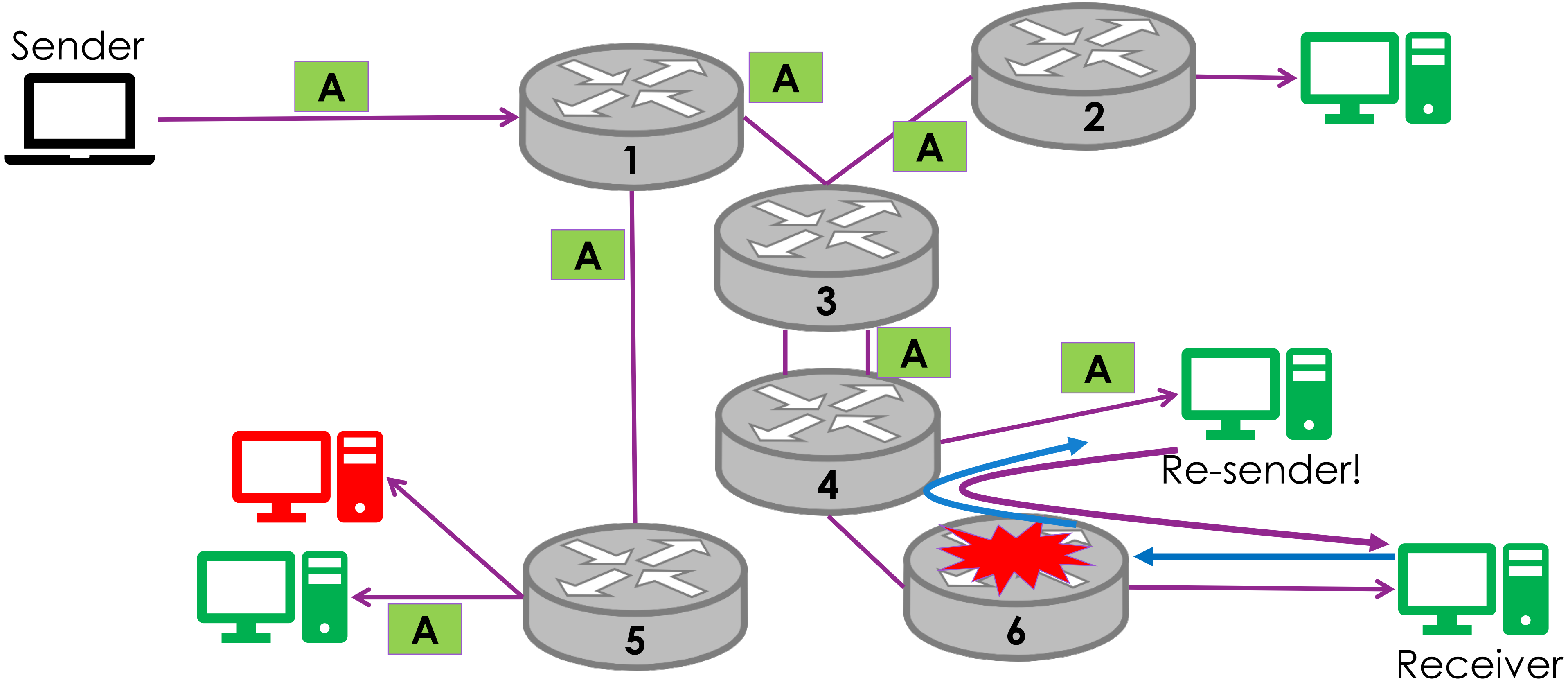
- The smaller the (desired) delay, the harder to deal with loss – so you “glitch”
  - Might be ok for video, less so for audio, but …

Fixes?

- <u>Retransmission</u>
  - You know something is lost
  - Request a resend of the problem packet
    - ARQ -> TCP-like!
  - Takes *round-trip-time* <u>plus</u> *transmission time* <u>plus</u> *queuing delays*
    - Huge buffer/delays implied
  - Generally not done
  - *In multicast, it may not be the sender who retransmits!*

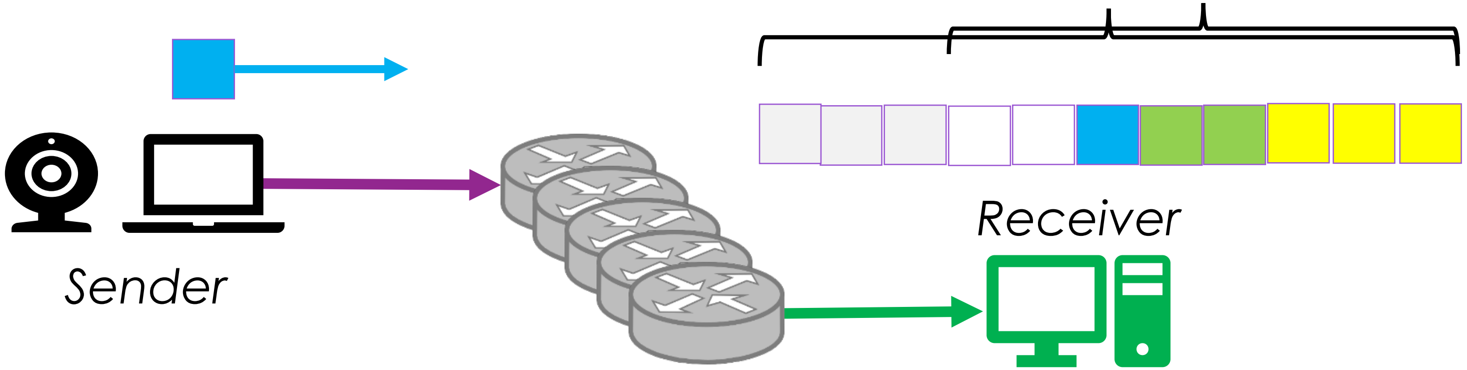
IP Multicast: UDP

Everyone is a listener, and a **sender**



Fixes?

- <u>Elastic buffers</u>
  - 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



Fixes?

- 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)

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Ver.		P	X	**CSRC. Count**				M	**Payload Type**							**Sequence Number**															
Timestamp																															
Synchronisation Source Identifier																															
*Contributing Source Identifier (*N)*																															
Profile Headers, Extension Headers, **Payload** (…)																															

- 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!
  - <u>Sender Reports</u>, <u>Receiver Reports</u>
    - Statistics: Packets received/lost, delay, delay variation,
  - <u>Source Description</u>, <u>Hello/Goodbye</u>, …
  - 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)

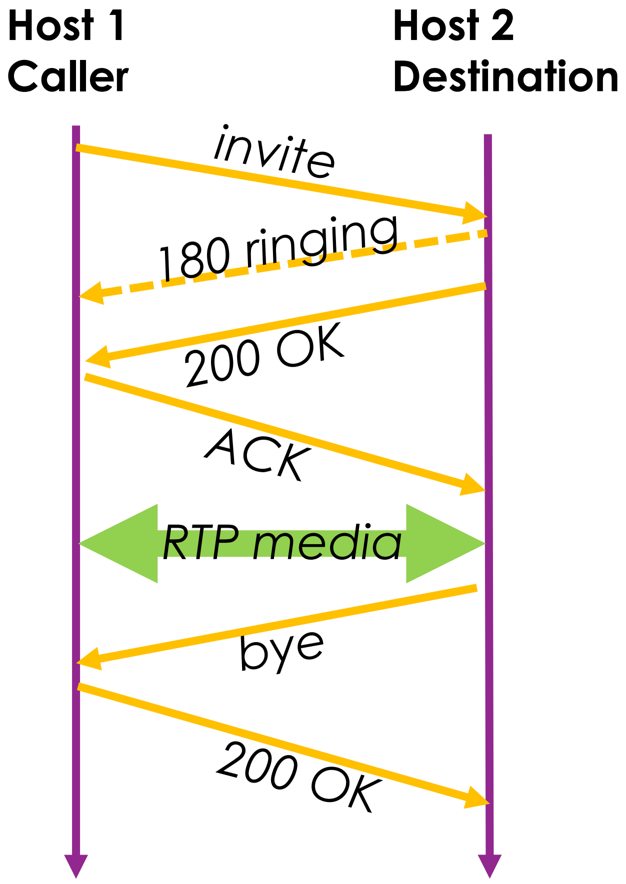
- Open (IETF) protocol to establish and tear-down calls (rfc3261+++)
  - Doesn't care how you transport the media
  - Is not used by Skype, Messenger, WhatsApp, Facetime, Zoom, Lync, …
  - Competes with ITU H.323

- 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 <u>Streaming</u> 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 a dozen 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