Real-Time Communication on IP Networks

Real-Time and Embedded Systems (M)

Lecture 16

UNIVERSITY
of
GLASGOW

Lecture Outline

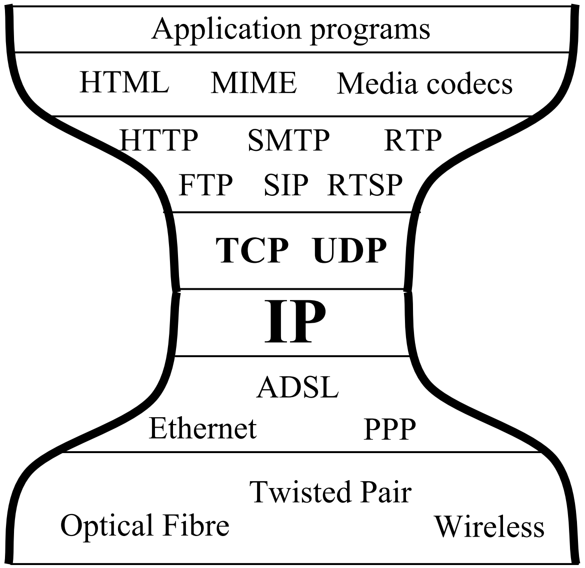
- Timing properties of IP networks
  - Examples of network behaviour
- Transport protocols
  - TCP/IP
  - UDP/IP
- Building real-time applications on IP
  - Use of RTP

Real-Time Communication on the Internet

- Primary focus has been on networked multimedia
  - First audio experiments on the ARPANET in 1973
    - RFC 741, "Network Voice Protocol", 1977
    - Predates the development of TCP/IP
  - First video experiments in the early 1980s
  - Modern standards development began in 1992
    - Developing from teleconferencing systems
    - Precursors to RTP and the present standards
    - Initial standards completed in 1996
  - Widespread availability of suitable networks in the last few years
- Starting to see experiments with sensor networks, data streaming
  - E.g. eVLBI radio astronomy

- How do the properties of IP networks impact real-time traffic?

The IP Protocol Stack



- IP provides an abstraction layer
  - Applications, transport protocols above
  - Assorted link technologies below
- Applications can't see the link layers
  - Just see IP layer performance
  - The IP routers can provide enhanced packet delivery service, but often don't
  - Assume lowest common denominator behaviour, unless you control the entire system
- Link layer can't tell the needs of the application
  - Just see a series of packets
  - Optimisations for particular traffic classes are risky (e.g. 802.11 retransmit)
  - Is the traffic really what you think?
- Real-time on IP ⇒ decoupling applications from the network

The IP Protocol Stack

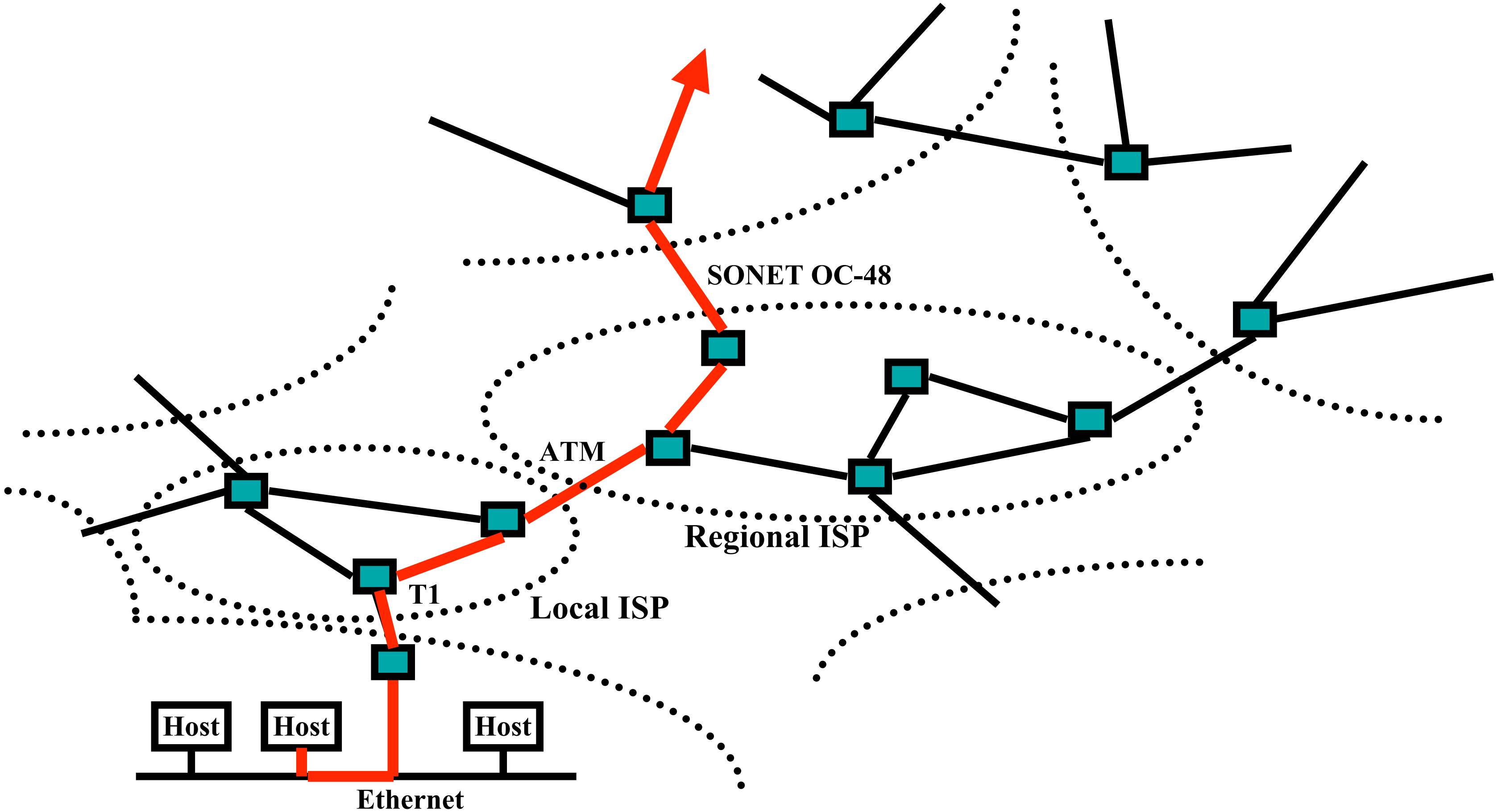
- Performance not guaranteed
- Packets can be…
  - lost
  - delayed
  - reordered
  - duplicated
  - corrupted

…and the transport protocol must compensate

- Many causes of problems:
  - Congestion ⇒loss and queuing
  - Packet corruption ⇒ loss
  - Route change ⇒ loss; change in latency
  - Multi-path routing ⇒ reorder
  - Link-layer striping ⇒ reorder
  - Spurious retransmissions ⇒ duplication

Assumption: significant packet loss, latency and jitter can be observed on a best effort IP network

Structure of the Internet

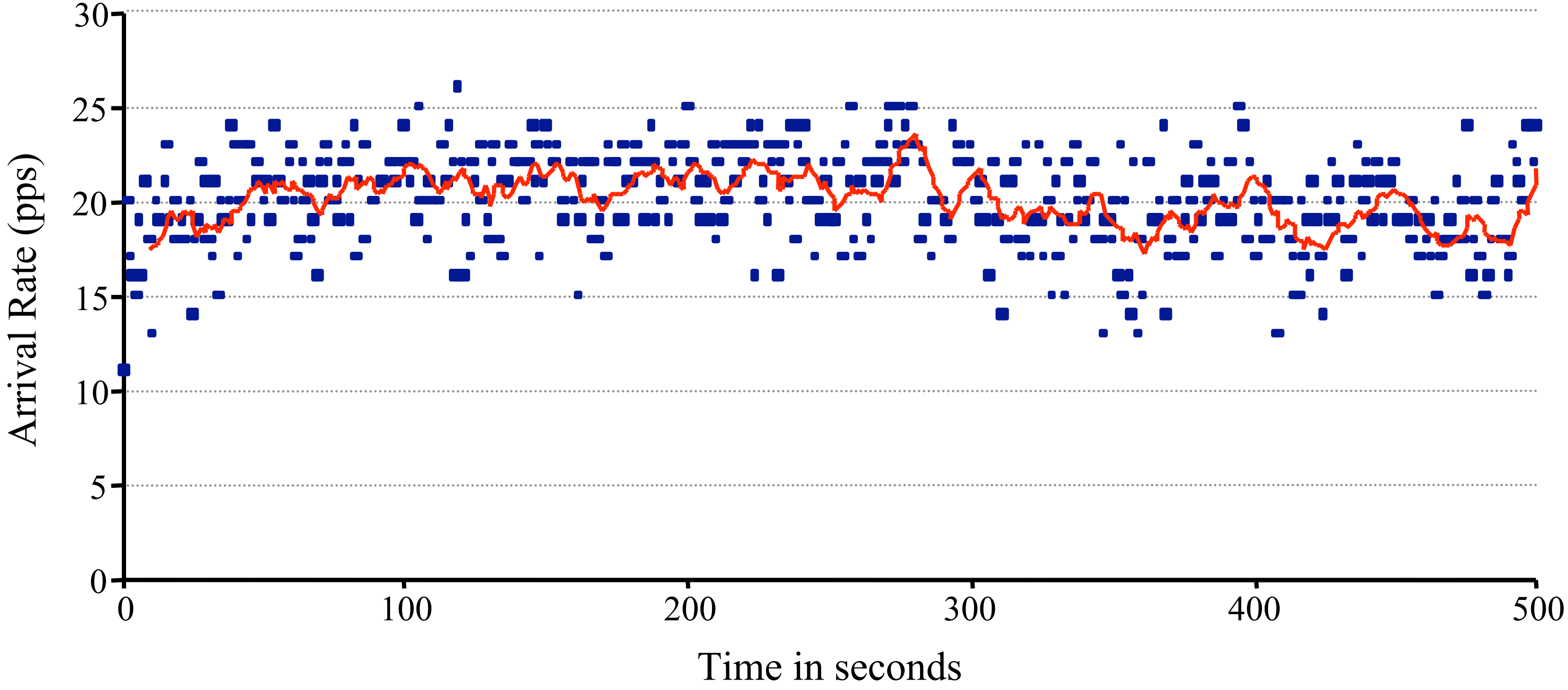


- Traffic passes through many hops, which can be maintained by different ISPs. How is the packet timing affected?
- Do you have an SLA with each?

Sample Internet Measurements

- Tests using a streaming audio application running between a site on the west coast of the US to the UK
- Observed the audio traffic at the IP layer
  - Constant rate (isochronous) traffic source
  - Packets generated by a periodic task: constant packet size, inter-packet gap
  - Desired behaviour is constant arrival rate, no jitter and no clock skew

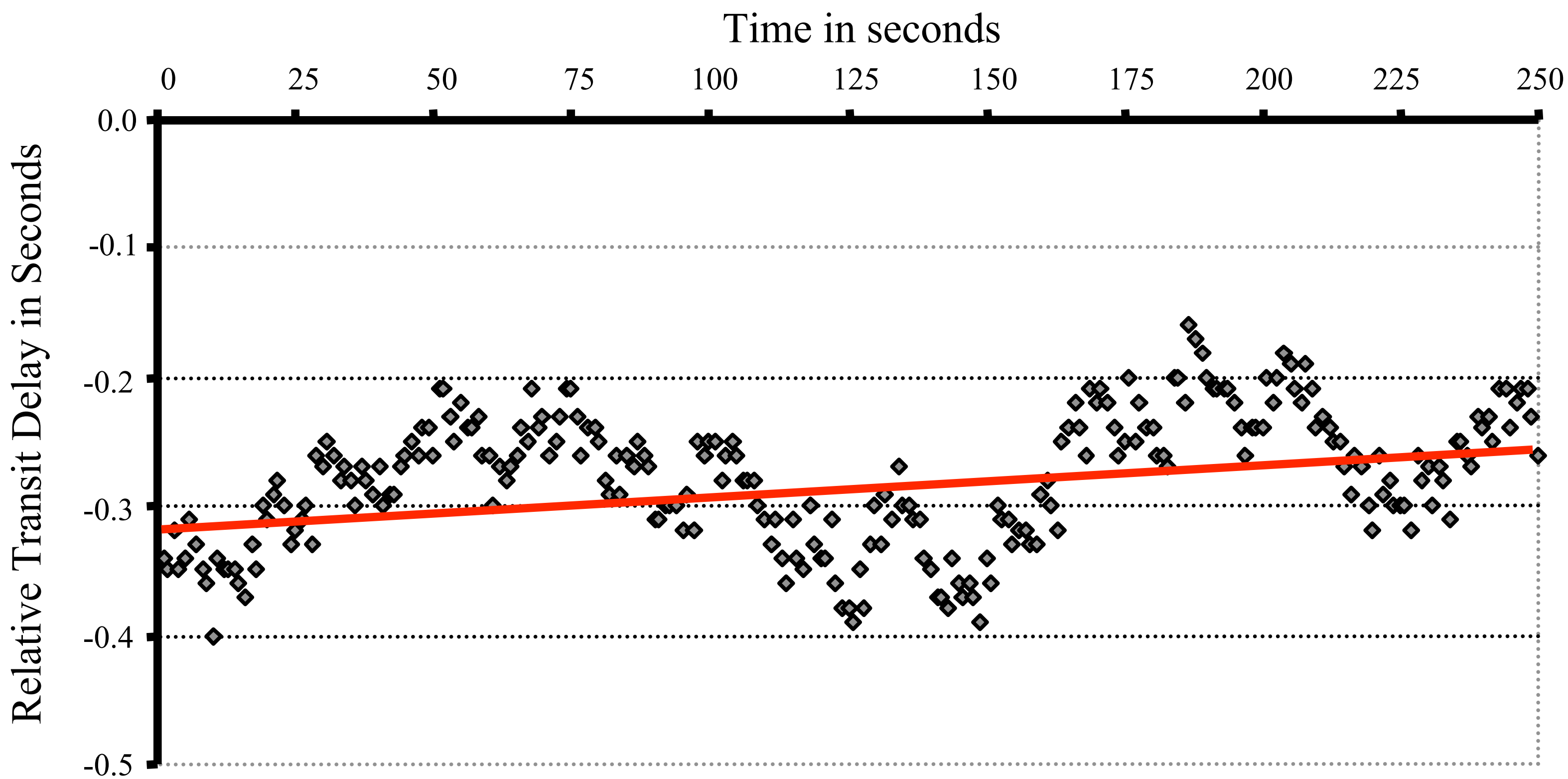
Throughput Variation in an IP Network



Blue points show a 1 second average

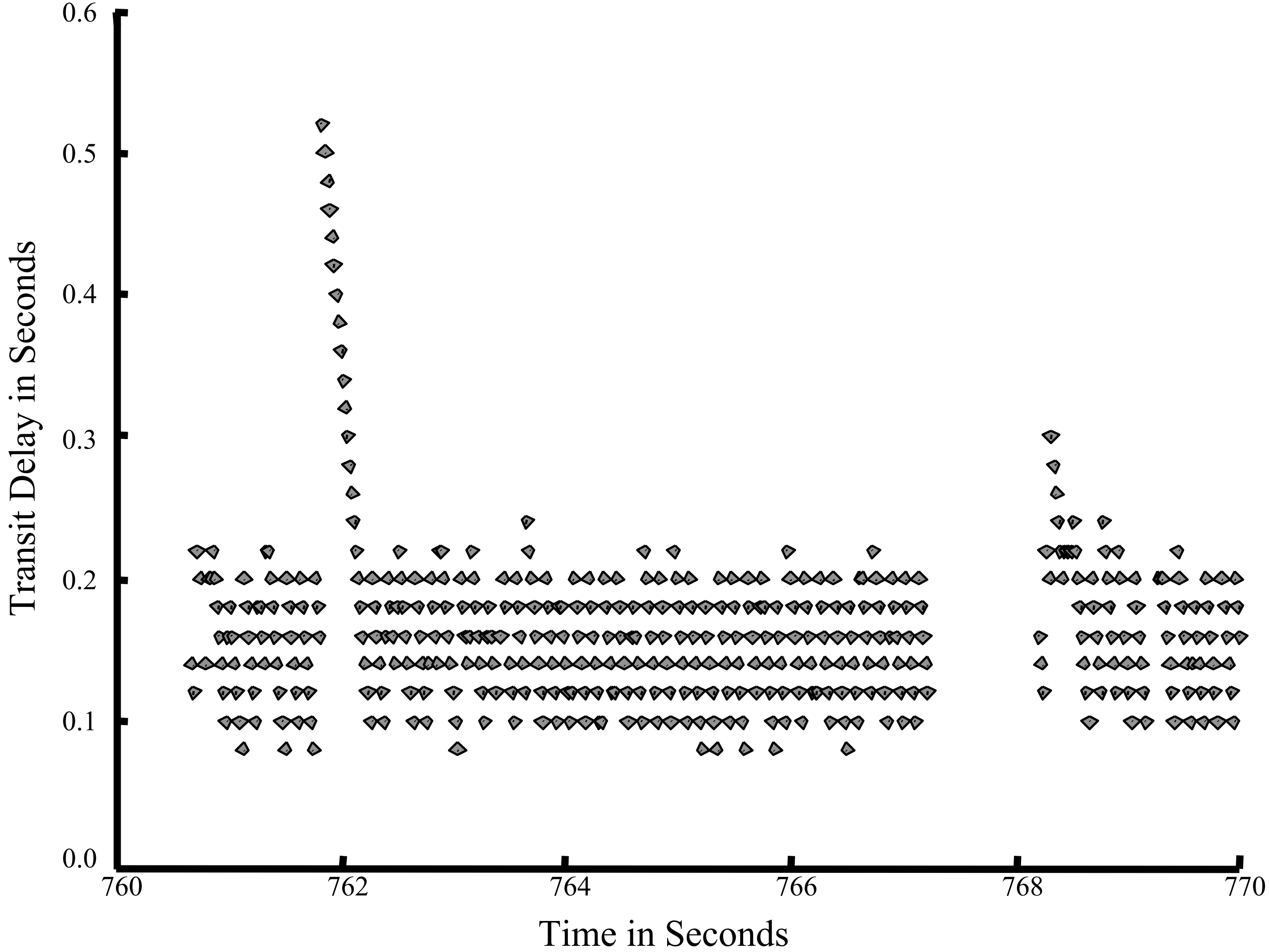
Red line shows a 10 second moving average

Jitter in an IP network



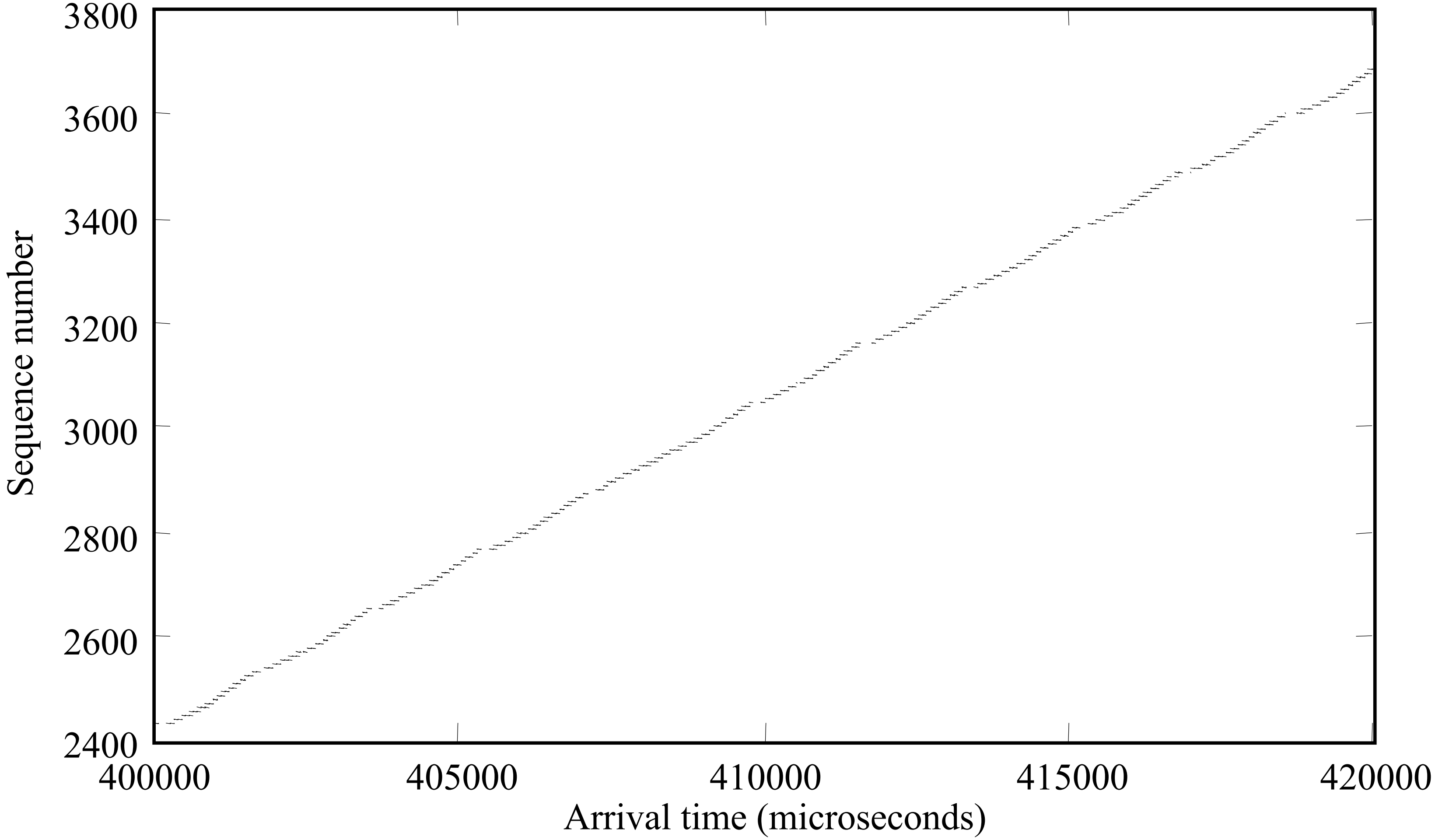
Subset of a longer plot, which shows clock skew indicated by red line

Jitter in an IP network



From: S. B. Moon, J. Kurose and D. Towsley, “Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms”, ACM/Springer Multimedia Systems, January 1998

Jitter in an IP Network



1Gbps video between Washington DC and Los Angeles offices of USC/ISI across a commercial ISP's backbone network

Real-Time on IP

- Performance *can* be bad
  – Applications should be prepared to compensate, isolating their timing behaviour and reliability from that of the network

- Packet loss, latency and jitter can be kept small through careful engineering and over-provisioning
  – Most backbone networks have very good performance
    - Essentially no loss
    - Very little queuing delay
  – Interconnects and customer LANs are currently the main trouble spots
  – Enhanced service networks can be used, if necessary
- Good enough for soft real-time, in many cases

Transport Protocols

- The IP service, by itself, is very limited
  - Just (tries to) deliver packets

- Always augmented by a transport protocol
  - UDP/IP
  - TCP/IP
  - (others in development)
- The transport protocol will impact perceived timing performance

UDP/IP

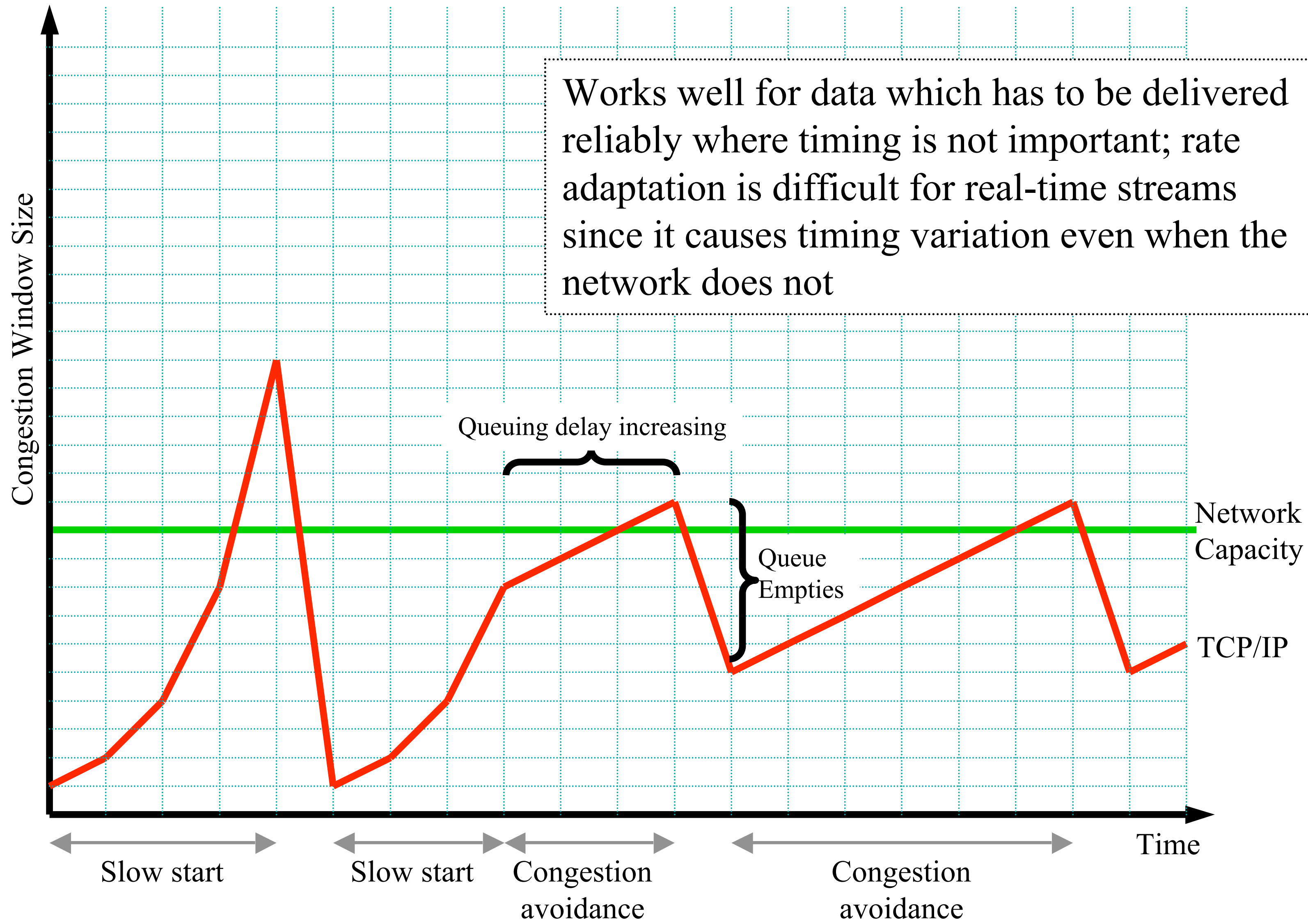
- Exposes the IP datagram service to applications
  - Best effort (unreliable) packet delivery
  - Connectionless
  - Unicast and multicast

- Can have all the problems we discussed in lecture 15:
  - Packet loss
  - Variable throughput
  - Jitter
- Uncontrolled timing, unless running on an enhanced service network, but no worse than the timing of IP

TCP/IP

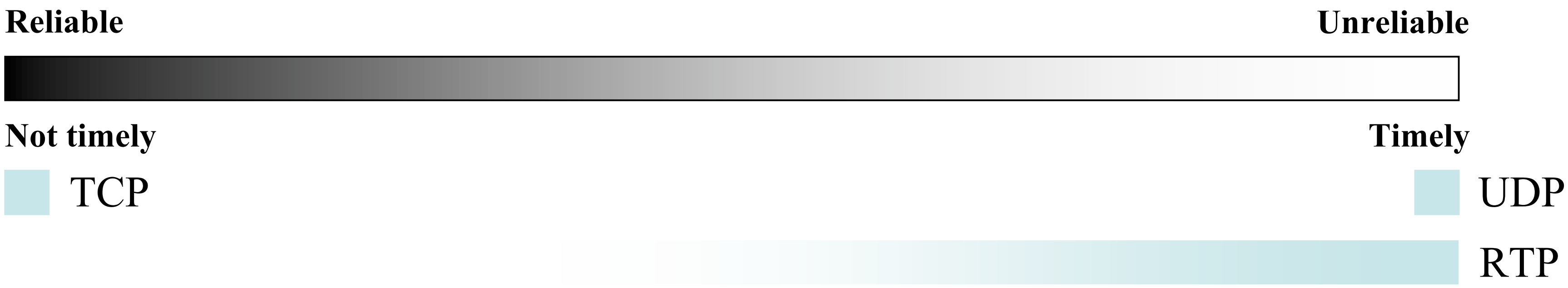
- Connection oriented, reliable, rate adaptive protocol built on IP
  - Each packet contains a sequence number
  - Acknowledgements sent as packets arrive
  - Sender retransmits any lost packets
  - Receiver buffers data until all preceding packets have arrived, and presents to the application in order
- Adapts transmission rate to match network capacity
  - High link utilization
  - Approximately fair share between flows
    - No prioritisation
- Combination of retransmission and rate adaptation result in significant timing variation
  - Affected by network dynamics, not controlled by application
  - Largely unusable by real-time traffic

TCP/IP Rate Adaptation



Works well for data which has to be delivered reliably where timing is not important; rate adaptation is difficult for real-time streams since it causes timing variation even when the network does not

Reliability/Timeliness Trade-off



- Protocols built on uncontrolled packet networks must make a fundamental trade-off:
  - Unreliable, accepting (mostly) timely behaviour of the network
  - Reliable, accepting that error correction will worsen the timing
- TCP is at one extreme, UDP the other
  - Application level protocols can blur the boundary
- Real-time systems choose their transport carefully:
  - TCP for control
  - UDP for data, aided by the application

Real-Time on UDP/IP Networks

- The challenge:
  - Build a mechanism for robust, real-time media delivery above an unreliable and unpredictable transport layer
  - Without changing the transport layer
    - If you can change the transport layer, would just use an enhanced service network, and avoid these problems

Push responsibility for media delivery onto the end-points where possible	Make the system robust to network problems; media data should be loss tolerant
⇓	⇓
The end-to-end argument	Application level framing

The End-to-End Argument

- Two options for ensuring reliability
  - Pass responsibility hop-by-hop, along with the data
    - E.g. Email
  - Responsibility remains with the end points, which ensure delivery even if the intermediate steps are unreliable
  - Most Internet protocols take the second approach

- Consequences:
  - Intelligence tends to "bubble-up" the protocol stack to the end points
  - The intermediate systems can be simple, and need not be robust
    - They can simply discard data they cannot deliver, since it will be recovered end-to-end

- The network is dumb, but end-points are smart

Application Level Framing

- Only the application has sufficient knowledge of its data to make an informed decision about how that data should be transported
- Implications:
  – The transport protocol should accept data in meaningful chunks (“ADUs”)
    - The application must understand the data,
    - The application must be able to process ADUs independently, in arbitrary order, and in the presence of loss
  – The transport protocol should expose details of delivery, allowing the applications to react intelligently if there are problems
    - The application can monitor delivery times, and adjust data use rates to match
    - Blind retransmission is not always appropriate
    - Maybe the data is stale, and an updated version can be sent
    - Maybe the data is obsolete, and doesn't need to be resent
    - Maybe an alternate representation of the data can be sent

Real-Time on IP Networks

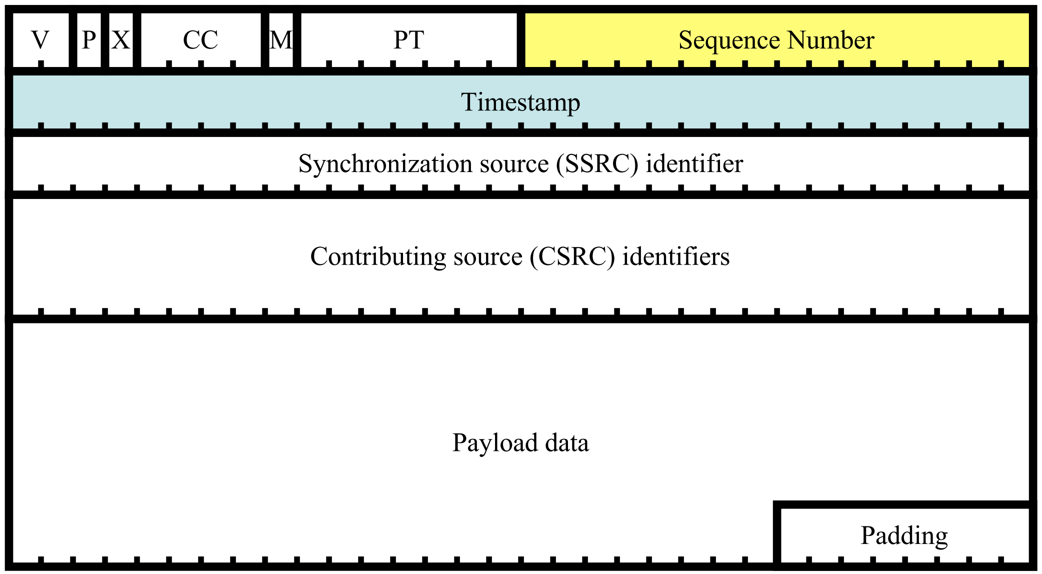
- This philosophy implies smart, network-aware, applications that are capable of reacting to problems end-to-end.
  - Both sender and receiver are intelligent
  - The network is dumb and can be unreliable
- Use a network protocol designed to work with applications, and to expose timing and reliability of the network

- Fits well with the IP service
- Contrast with traditional real-time networked applications:
  - Telephone network is smart, end-points are dumb
  - TV distribution: MPEG sender is smart, receiver relatively dumb

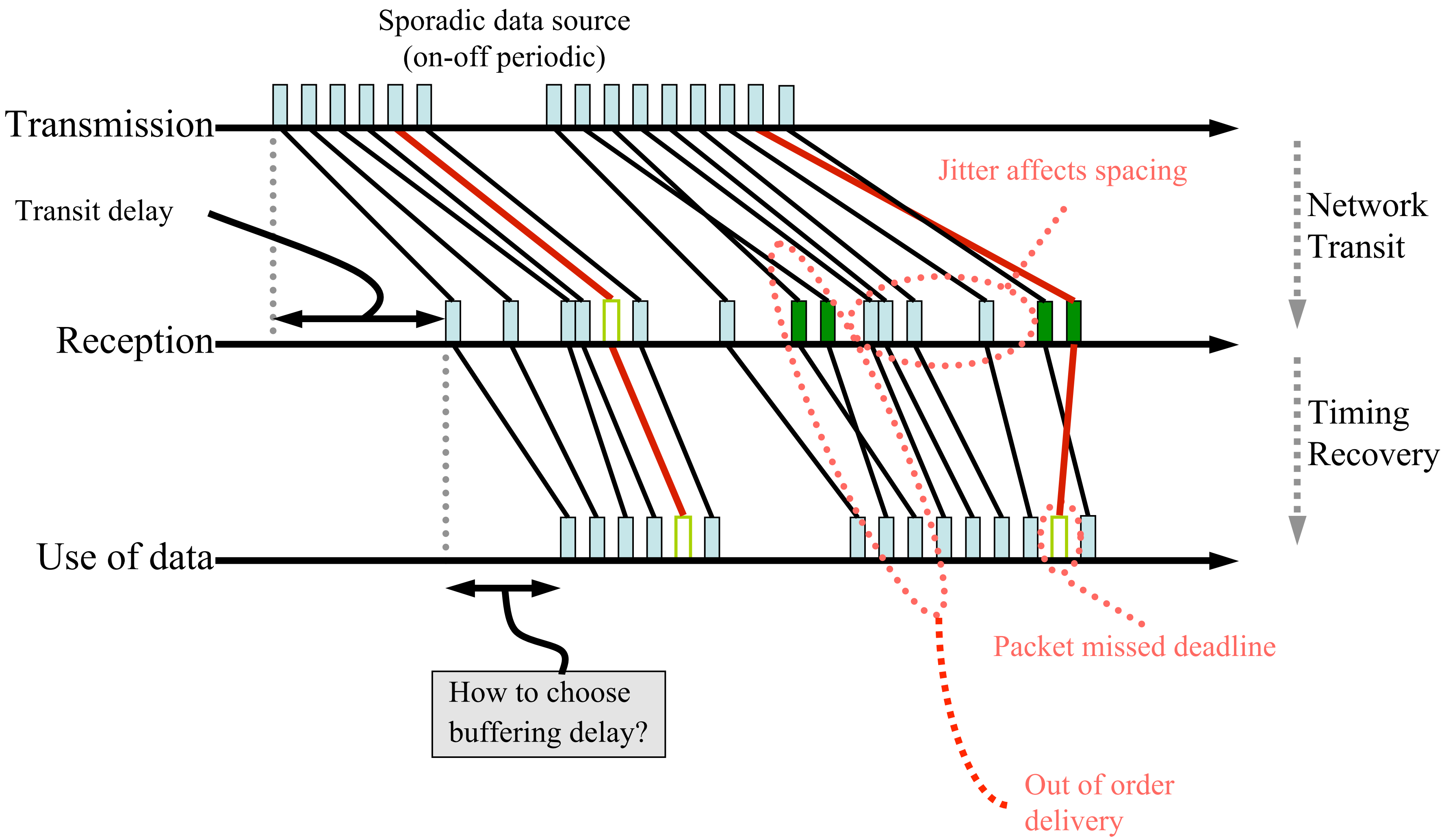
RTP: Real-time Transport Protocol

- The standard for real-time transport over IP networks
  - Streaming audio and video
  - Voice over IP
  - Sensor data
- Implemented as part of application, exposing the underlying timing of the network to allow us to build real-time systems

Sequence numbers and timestamps allow the application to recover timing and ordering



Buffering for Timing Recovery

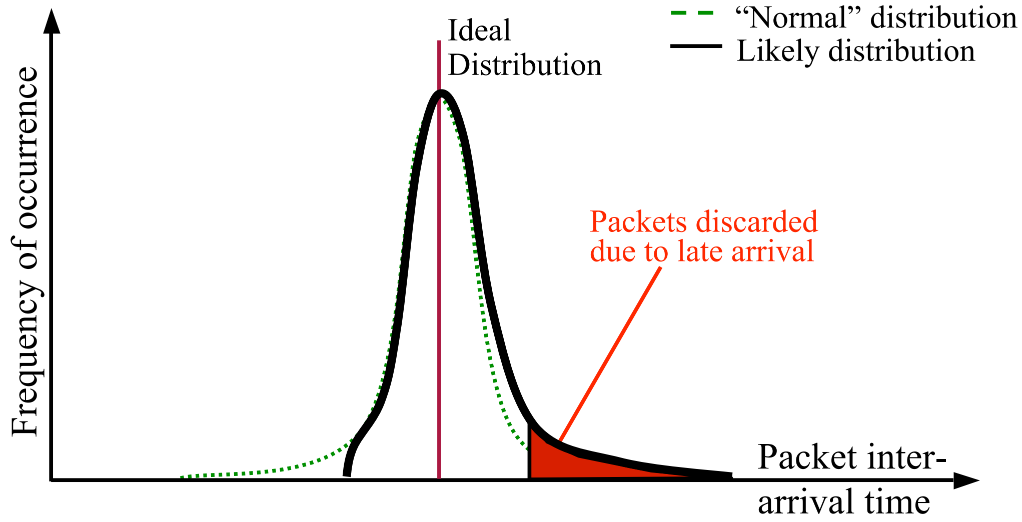


Receiver must buffer, to smooth network timing variations

How Much Buffering Delay?

- Depends on jitter statistics
- Assume a normal distribution, and calculate standard deviation σ of inter-arrival times

⇒ 99.7% within 3σ of the mean



- Buffer for 3 times the standard deviation of the inter-arrival times and hope this missing ~0.3% of deadline is acceptable

- Is a normal distribution a valid assumption?
- Absolutely not!
  - But close enough for many soft real-time applications
    - Engineering rule of thumb: assume, approximate, test
  - The Internet is clearly *not* suitable for hard real-time applications anyway…

Timing Recovery

- RTP does not specify standard buffering and timing recovery algorithms
  - The necessary information is provided
  - Implementations choose how to recovery timing, based on their needed accuracy

- Many trade-offs to consider:
  - latency versus quality
  - speed of reaction to change
  - buffering ability

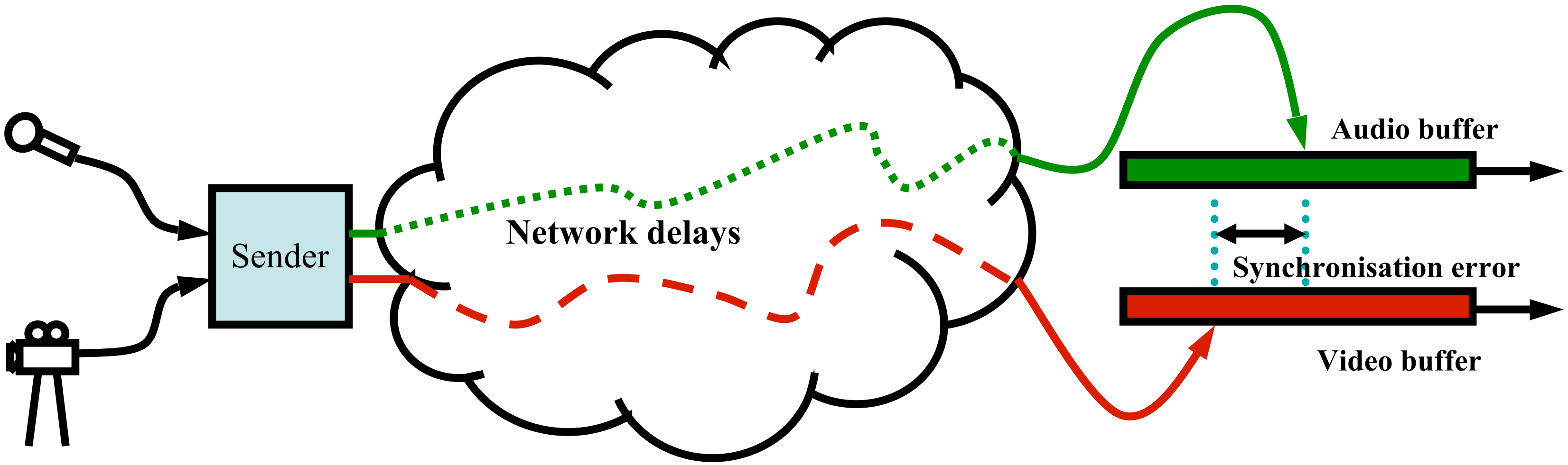
- Typical design:
  - Streaming applications use large delay (several seconds)
  - Interactive applications try to keep delay low (tens of milliseconds)

RTP Control Protocol (RTCP)

- Each RTP data flow has an associated control flow
- The control flow provides:
  - Time-base management and information for synchronization
  - Quality of service feedback
  - Member identification and management

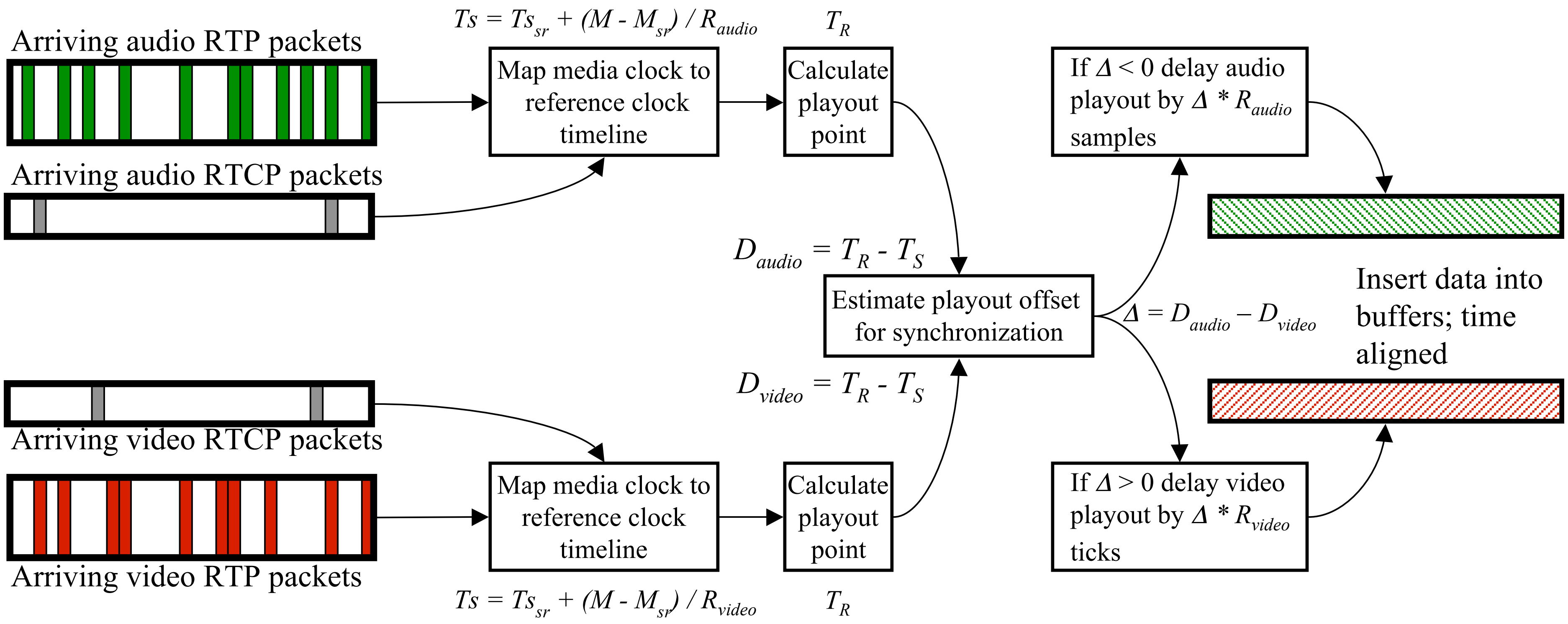
- Low-rate periodic status report packets
  - 5 seconds ±50% for point-to-points sessions
  - Scales with group size for multicast sessions

Synchronization and Time Management



- RTCP packets contain timestamps to map between the RTP timeline and NTP “wall-clock” time
  - Provides the information needed for a receiver to synchronize data sent as different flows, with different clocks
- Also allows receivers to estimate data/packet rate and possibly clock skew

Synchronization and Time Management



- Use RTCP packets to map data clocks to a common timeline
- Estimate offset and skew between clocks
- Delay use of one set of data to align with the other set

Reception Quality Reporting

- Quality of service feedback from each receiver:
  - Loss fraction
  - Cumulative number of packets lost
  - Highest sequence number received
  - Inter-arrival jitter
  - Round-trip time
- Many uses:
  - Loss rate can be used to select amount of FEC to employ
  - Jitter gives estimate of play out buffer delay at receiver

Summary of RTP

- RTP provides:
  - Flexible and extensible real time data transfer protocol
    - Supports a range of data type
    - Allows detection of network problems
    - Allows recovery of media timing
  - Associated, low rate, reporting of reception quality, time-base, and presence information

- The building blocks to let *soft real-time* applications adapt to the vagaries of an IP network
  - Follows end-to-end argument and principles of application level framing; applications required to be intelligent

Summary

By now, you should know…

- Timing properties of IP networks
- Use of TCP/IP and UDP/IP for real-time traffic
- Overview of RTP
- Understanding that real-time on IP networks is limited to soft real-time, with flexible applications