Integrated services

- Reading:
  - S. Keshav, “An Engineering Approach to Computer Networking”, chapters 6, 9 and 14

DigiComm II

Module objectives

Learn and understand about:

- Support for real-time applications:
  - network-layer and transport-layer
- **Quality of service** (**QoS**):
  - the needs of real-time applications
  - the provision of QoS support in the network
- Many-to-many communication - **multicast**
- **Integrated Services Network (ISN)**

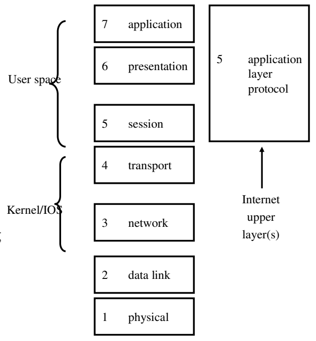
DigiComm II

During the 1990’s, applications have become increasingly reliant on the use of the Internet protocols to provide data communications facilities. The use of the Internet protocols seems likely to increase at an extremely rapid rate and the Internet Protocol (IP) will be the dominant data communications protocol in the next decade. IP is being used for a huge variety of “traditional” applications, including e-mail, file transfer and other general non-real-time communication. However, IP is now being used for real-time applications that have **quality of service** (**QoS**) sensitive data **flows**. A **flow** is a stream of semantically related packets which may have special QoS requirements, e.g. an audio stream or a video stream. Applications such as conferencing (many-to-many communication based on **IP multicast**), telephony – **voice-over-IP** (**VoIP**) – as well as streaming audio and video are being developed using Internet protocols. The Internet and IP was never designed to handle such traffic and so the Internet community must evolve the network and enhance the Internet protocols in order to cater for the needs of these new and demanding applications. Users wish to have access to a whole plethora of telecommunication and data communication services via the Internet; users wish to access an **Integrated Services Network** (**ISN**).

In this set of lectures, we try to understand about QoS for IP-based applications and how the network must be changed to support these new applications.

Support for real-time applications

- Support in the network:
  - routers, routing
- Support at the end-systems:
  - transport protocols
- Support at the application level:
  - user-network signalling
  - application-level signalling and control
- (Link & physical layers?)



DigiComm II

To provide support for real-time applications, we need to introduce mechanisms at many different parts of the communication stack.

At the network layer, we need to modify router behaviour so that packets belonging to QoS sensitive flows receive some kind of preferential treatment, compared to “normal” data packets. We also need to modify the behaviour of routing protocols in order to support multicast communication and QoS-based routing metrics.

At the transport layer, recall that we only have two general protocols: TCP for traditional applications that require an ordered by-stream delivery, and UDP for applications that build in specific control mechanisms at the application layer. For real-time flows, we can identify some general requirements, which we will see can be implemented by extending UDP as in the **Real-time Transport Protocol (RTP)**.

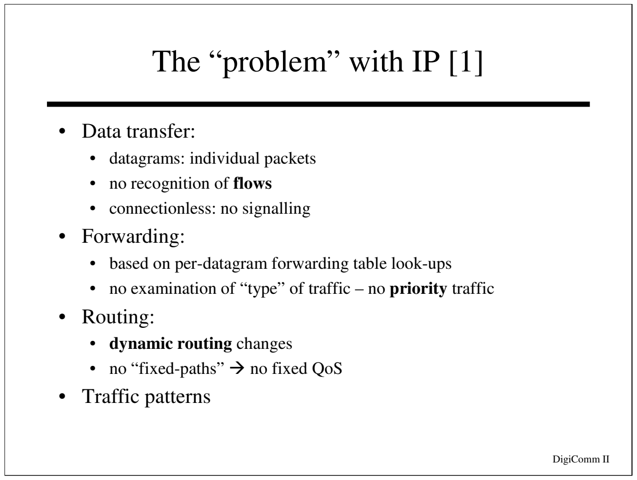
At the application layer, we may identify other mechanisms that are required for specific real-time applications: **floor control** for conference applications; **transcoding** for audio and video flows; **security mechanisms** such as authentication. Although it is possible to identify some general requirements, such higher-layer mechanisms tend to specific to particular applications.

In this set of lectures, we consider the support that we have in the network and at the transport layer, as well as some general issues concerning the interface between the application and the network.

Why do we not consider the link layer and physical layer? Surely these have a fairly vital role in QoS as they provide the transmission capability? Remember that IP tries to hide the lower layers, so although we will se there are important issues concerning the lower layers, we concentrate on the network layer and transport layer.

Real-time flows and the current Internet protocols

DigiComm II



The “problem” with IP [1]

- Data transfer:
  - datagrams: individual packets
  - no recognition of **flows**
  - connectionless: no signalling
- Forwarding:
  - based on per-datagram forwarding table look-ups
  - no examination of “type” of traffic – no **priority** traffic
- Routing:
  - **dynamic routing** changes
  - no “fixed-paths” $\rightarrow$ no fixed QoS
- Traffic patterns

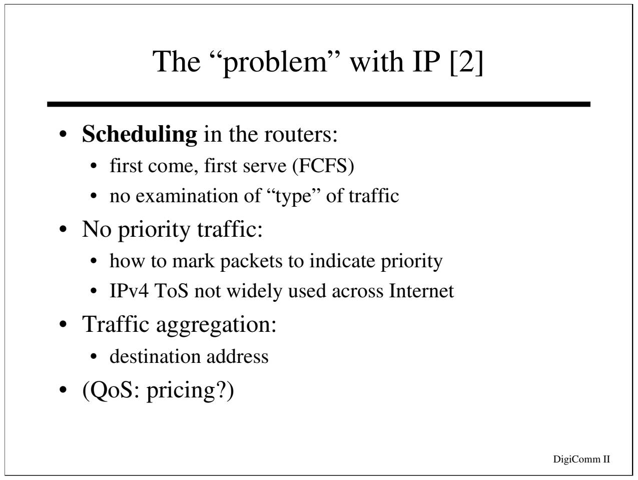
DigiComm II

Let us first examine the service that IP offers. IP offers a **connectionless** datagram service, giving no guarantees with respect to delivery of data: no assumptions can be made about the delay, jitter or loss that any individual IP datagrams may experience. As IP is a connectionless, datagram service, it does not have the notion of **flows** of datagrams, where many datagrams form a sequence that has some meaning to an applications. For example, an audio application may take 40ms “time-slices” of audio and send them in individual datagrams. The correct sequence and timeliness of datagrams has meaning to the application, but the IP network treats them as individual datagrams with no relationship between them. There is no **signalling** at the IP-level: there is no way to inform the network that it is about to receive traffic with particular handling requirements and no way for IP to tell signal users to back-off when there is congestion.

At IP routers, the forwarding of individual datagrams is based on forwarding tables using simple metrics and (network) destination addresses. There is no examination of the type of traffic that each datagram may contain – all data is treated with equal **priority**. There is no recognition of datagrams that may be carrying data that is sensitive to delay or loss, such as audio and video.

One of the goals of IP was to be robust to network failure. That is why it is a datagram-based system that uses dynamic routing to change network paths in event of router overloads or router failures. This means that there are no fixed paths through the network. It is possible that during a communication session, the IP packets for that session may traverse different network paths. The absence of a fixed path for traffic means that, in practice, it can not be guaranteed that the QoS offered through the network will remain consistent during a communication session.

Even if the path does remain stable, because IP is a totally connectionless datagram traffic, there is no protection of the packets of one flow, from the packets of another. So, the traffic patterns of a particular user’s traffic affects traffic of other users that share some or all of the same network path (and perhaps even traffic that does not share the same network path!).



The “problem” with IP [2]

- **Scheduling** in the routers:
  - first come, first serve (FCFS)
  - no examination of “type” of traffic
- No priority traffic:
  - how to mark packets to indicate priority
  - IPv4 ToS not widely used across Internet
- Traffic aggregation:
  - destination address
- (QoS: pricing?)

DigiComm II

At the individual routers, the process of forwarding a packet involves, taking an incoming packet, evaluating its forwarding path, and then sending it to the correct output queue. Packets in output queues are serviced in a simple first-come first-serve (FCFS) order, i.e. the packet at the front of the queue is transmitted first. The ordering of packets for transmission takes the general term on **scheduling**, and we can see FCFS is a very simple scheduling mechanism.

FCFS assumes that all packets have equal priority. However, there is a strong case to instruct the router to give some traffic higher priority than other traffic. For example, it would be useful to give priority to traffic carrying real-time video or voice. How do we distinguish such **priority** traffic from non-priority traffic, such as, say e-mail traffic. The IPv4 **type of service (ToS)** do offer a very rudimentary form of marking traffic, but the semantics of the ToS markings are not very well defined. Subsequently, the ToS field is not widely used across the Internet. However, it can be used effectively across corporate Intranets.

Also, in dealing with traffic that has been marked, we need to be wary of the extra processing a marking scheme may introduce in the core of the network where there is very high **aggregation** of network traffic. With FCFGS, effectively, aggregation is by use of the destination IP address.

Even if we could offer some sort of QoS control mechanism, with **prioritisation** or **traffic differentiation**, there is then the issue of **pricing**. How do we charge for use of network resources for a particular treatment of traffic for a particular customer?

Questions

- Can we do better than **best-effort**?
- What support do real-time flows need in the network?
- What support can we provide in the network?
- Alternatives to FCFS?
- Many-to-many communication?
- Application-level interfaces?
- **Scalability?**

So we can ask ourselves several questions.

Firstly, can we provide a better service that that which IP currently provides – the so-called best-effort?

The answer to this is actually, “yes”, but we need to find out what it is we really want to provide! We have to establish which parameters of a real-time packet flow are important and how we might control them. Once we have established our requirements, we must look at new mechanisms to provide support for these needs in the network itself. We are essentially asking trying to establish alternatives of FCFS for providing better control of packet handling in the network as well as trying to support **multi-party (many-to-many) communication**.

We also need to consider how the applications gain access to such mechanisms, so we must consider any **application-level interface** issues, e.g. is there any interaction between the application and the network and if so, how will this be achieved.

In all our considerations, one of the key points is that of **scalability** – how would our proposals affect (and be affected by) use on a global scale across the Internet as a whole.

Requirements for an ISN [1]

Today’s Internet

- IPv4: QoS not specified
- TCP: elastic applications
- Many network technologies:
  - different capabilities
  - no common layer 2
- No support for QoS:
  - ToS in IPv4 – limited use
- QoS requirements:
  - not well understood

Integrated Services Packet Network (ISPN)

- QoS service-level:
  - service type descriptions
- Service interface:
  - signalling
- Admission control:
  - access to resources
- Scheduling:
  - prioritisation and differentiation of traffic

The Internet was never designed to cope with such a sophisticated demand for services [Cla88] [RFC1958]. Today’s Internet is built upon many different underlying network technologies, of different age, capability and complexity. Most of these technologies are unable to cope with such QoS demands. Also, the Internet protocols themselves are not designed to support the wide range of QoS profiles required by the huge plethora of current (and future) applications.

In [CSZ92], the authors speak of the Internet evolving to an **integrated services packet network (ISPN)**, and identify four key components for an Integrated Services architecture for the Internet:

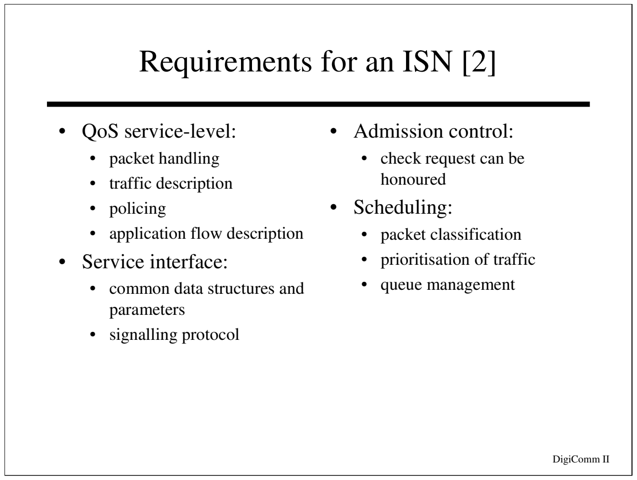
• **service-level:** the nature of the commitment made, e.g. the INTSERV WG has defined **guaranteed** and **controlled-load** service-levels (these are discussed later) and a set of control parameters to describe traffic patterns, which we examine later

• **service interface:** a set of parameters passed between the application and the network in order to invoke a particular QoS service-level, i.e. some sort of **signalling protocol** plus a set of parameter definitions

• **admission control:** for establishing whether or not a service commitment can be honoured before allowing the flow to proceed

• **scheduling mechanisms within the network:** the network must be able to handle packets in accordance with the QoS service requested

A key component that is required here is signalling – talking to the network. Signalling is essential in connection-oriented networks (used for connection control), but datagram network typically need no signalling. No signalling mechanism exists in the IP world – it is not possible to talk to the network, one simply uses the service it provides. The signalling part of a CO network communication offers a natural point at which information about the particular requirements of a connection can be transmitted to the network. As IP is connectionless, any signalling mechanism should that is compatible with current operation of the Internet and should not constrain or change the operation of existing applications and services.



Requirements for an ISN [2]

- QoS service-level:
  - packet handling
  - traffic description
  - policing
  - application flow description
- Service interface:
  - common data structures and parameters
  - signalling protocol
- Admission control:
  - check request can be honoured
- Scheduling:
  - packet classification
  - prioritisation of traffic
  - queue management

DigiComm II

The simple description of the interactions between these components is as follows:

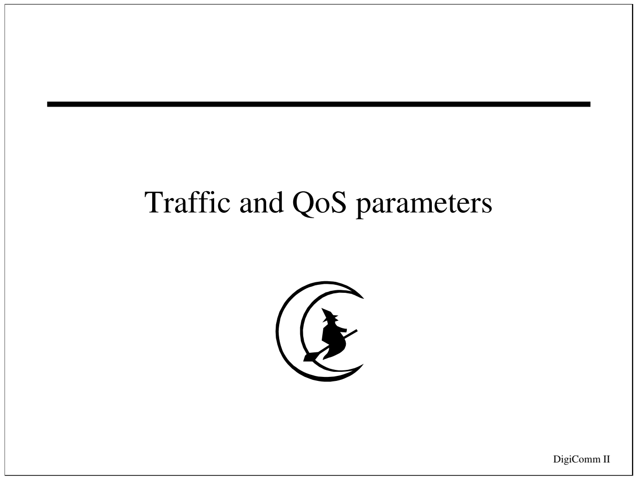
• a **service-level** is defined (e.g. within an administrative domain or, with global scope, by the Internet community). The definition of the service-level includes all the service semantics; descriptions of how packets should be treated within the network, how the application should inject traffic into the network as well as how the service should be policed. Knowledge of the service semantics must be available within routers and within applications

• an application makes a request for service invocation using the **service interface** and a **signalling protocol**. The invocation information includes specific information about the traffic characteristics required for the flow, e.g. data rate. The network will indicate if the service invocation was successful or not, and may also inform the application if there is a service violation, either by the application's use of the service, or if there is a network failure

• before the service invocation can succeed, the network must determine if it has enough resources to accept the service invocation. This is the job of **admission control** that uses the information in the service invocation, plus knowledge about the other service requests it is currently supporting, and determines if it can accept the new request. The admission control function will also be responsible for policing the use of the service, making sure that applications do not use more resources than they have requested. This will typically be implemented within the routers

• once a service invocation has been accepted, the network must employ mechanisms that ensure that the packets within the flow receive the service that has been requested for that flow. This requires the use of **scheduling mechanisms** and **queue management** for flows within the routers

Imagine a video application. The application or user would select a service level and note the traffic characteristics required for the video flow. Information such as required data rate would be encapsulated in a data structure (service interface) that is passed to the network (signalling). The network would make an assessment of the request made by the application and consider if the requirements of the flow can be met (admission control). If they can be met routers would ensure that the flow receives the correct handling in the network (scheduling and queue management).



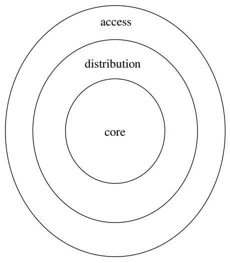
Traffic and QoS parameters

DigiComm II

Network structure [1]

Network hierarchy

- Access network:
  - low multiplexing
  - low volume of traffic
- Distribution network:
  - interconnectivity at local level
  - medium volume of traffic
  - low multiplexing
- Core network – backbone:
  - high volume of traffic
  - high multiplexing



DigiComm II

Let us first examine the problem of traffic. The network consists of several layers of hierarchy with respect to traffic volume and traffic multiplexing.

The outer layer is the **access** network, and can also be considered the edge of the network – closest to the users (applications) which generate the traffic. Here, there is typically a dedicated link to the end-system (for example Ethernet over UTP or residential dial-up). On these links, the level of multiplexing is low and the volume of traffic is comparatively low. If we consider a corporate network, with respect to the Internet, the site network and the corporate's ISP is seen as the access network. For residential users, the ISP network is seen as the access network.

The access network passes on traffic to a **distribution** network. The distribution network may also be called a **transit** network. This network has the job of connecting the access network to the main **core** or **backbone** networks. Typically, end users do no transmit traffic directly onto the distribution network. The distribution network has higher levels of multiplexing and higher volumes of traffic than the access network. However, the level of multiplexing may not be much more than the access network. As an example, several ISPs may use the same transit network (to access the same backbone provider, perhaps). Distribution networks may have a scale that covers a large geographical region, may be as much as a whole country.

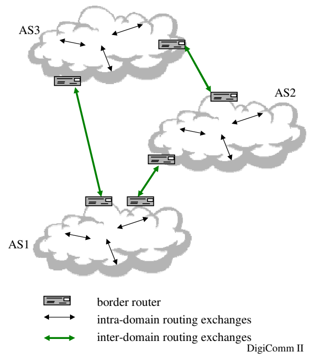
The **core** or **backbone** network is the part of the network that provides international/global interconnectivity. There are in fact many backbone network providers, and the have **peering agreements** to interconnect their networks. Here, there are very high volumes of traffic and very high levels of multiplexing – traffic from millions of users.

This diversity in the volume of traffic and its level of multiplexing has serious implications for any kind of traffic control mechanisms we may wish to apply. We have to remember that if we choose mechanisms that act on a per-packet basis to control traffic, these mechanisms must be scalable in order to maintain performance.

Network structure [2]

Administrative boundaries

- Autonomous system (AS):
  - **intra-domain routing**
  - internal policy
  - routing metric?
  - protocols: RIPv2, OSPFv2
- Interconnection of ASs:
  - **inter-domain routing**
  - interconnectivity information
  - protocols: BGP


AS3
AS2
AS1
border router
intra-domain routing exchanges
inter-domain routing exchanges

DigiComm II

This is not the only way of viewing the network. The Internet was formed from the interconnection of many other networks. Today it consists of many network technologies and many network providers all using a standard set of protocols and mechanisms for internetworking. However, each network operator wishes to have control of the construction, operation, management and maintenance of their own network. Indeed the way that routing is organised for the Internet acknowledges these **administrative boundaries** between networks that run as **autonomous systems**.

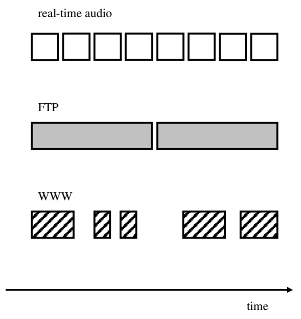
Within an AS, an operator can choose to run their network how they like. They will use an **intra-domain routing** protocol such as RIPv2 or OSPFv2. They will have their own policy for administering the day to day operation of the network, and this policy may well be confidential. The routing metrics that they use may not make sense outside the AS boundary.

Between ASs, different routing protocols – **inter-domain routing** protocols - are used, such as BGP. In fact, no routing metrics are passed between ASs using BGP. rather, **interconnectivity** information is passed,using (effectively) discovery messages to determine connectivity paths with respect to ASs.

Given this autonomy of network operators, and the usage of diverse routing information and routing protocols, it is not realistic to agree on common routing metrics and polices for handling traffic on a global basis.

Mixed traffic in the network [1]

- Different applications:
  - traffic (generation) profiles
  - traffic timing constraints
- Routers use FCFS queues:
  - no knowledge of application
  - no knowledge of traffic patterns
- Different traffic types share same network path
- Consider three different applications …



DigiComm II

Different applications generate different kinds of traffic. The traffic has different requirements with respect to a series or **flow** of chunks of data; the size of data chunks that are created, the inter-chunk spacing with respect to time, and the timeliness of delivery of the chunks across the network. These different traffic types are mixed together when transmitted across the network. Remember that routers traditionally use **first-come-first-served (FCFS)** queuing mechanisms. That is, the packets are forwarded by the router in an order determined only by their arrival at router's input lines, and no consideration is made about the type of traffic or the requirements of the application. Remember that datagram forwarding considers each datagram (IPv4 packet) as a separate entity and there is no notion of a **flow** of packets in IPv4. This means that where an application does have a flow of packets, they will not be treated in any special way by the network. We will have a look at some simple (fictional) example applications to demonstrate what happens when the traffic traverses a network.

Firstly, consider a real-time audio application. This may produce a stream of evenly spaced packets, each of which represents, say, a 40ms time slice of audio. The spacing of the audio packets should be maintained at the receiver so that the playout of the audio stream is not "jerky". Also, we would prefer packets not to be reordered or lost. The whole of the IP packet may be no more than about 100bytes in size.

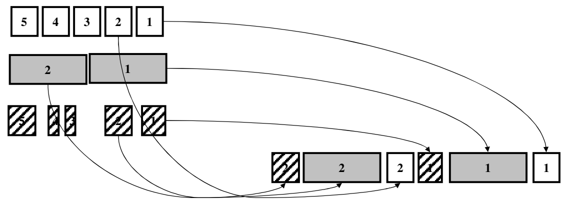
Secondly, let us consider a large file transfer. This will typically create large packets, up to the size of the path MTU (e.g. 1500bytes),in order to make most efficient use of the available network capacity. The packets will be generated as a steady stream by the sender of the file. Although loss of packets could be a problem to the application, reordering and disruption of the packet spacing will not have nay adverse affects to the application.

Thirdly, let us consider someone browsing the web. The traffic generated by the server will depend on the actions of the users of the clients (browsers). There may be relatively large silent periods (no transmissions) while the user is reading a page and the bursts of activities during browsing and download of pages from server to client. The chunks of data are of very variable size, from a few 10's of bytes to several hundred bytes.

A representation of the traffic profiles of these three types of applications is given above. We see that, pictorially, they look very different. Let use consider what happens as these flows pass along parts of the same network path.

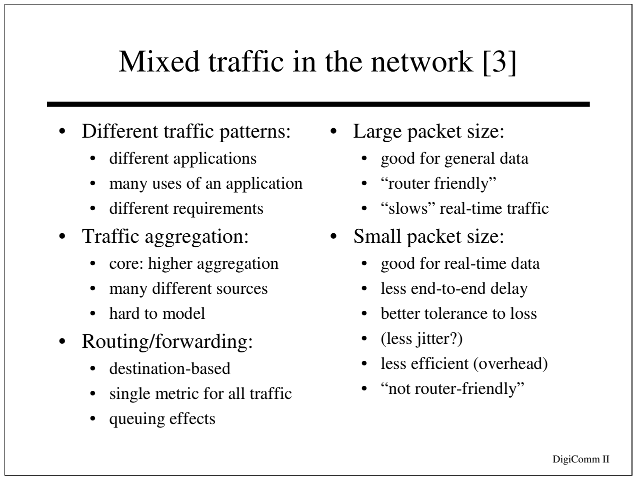
Mixed traffic in the network [2]

- Router:
  - 3 input lines: serviced round-robin at router
  - 1 output line (1 output buffer)



DigiComm II

Let us examine the processing of our example flows in a very simple router. Let us assume that these three flows arrive at the router on three separate interfaces and will all be forwarded by the route onto the same outgoing link. We assume that the router performs a very simple round-robin servicing of the input queues, taking on packet from each input queue and sending it to the output. We can see how the traffic patterns of our flows are disrupted when the flows are aggregated. This picture may be slightly misleading in that we do not show the relative speeds of the incoming lines and outgoing lines, but if we assume that they all run at the same speed, we see that there disruption to the traffic flow.



Mixed traffic in the network [3]

- Different traffic patterns:
  - different applications
  - many uses of an application
  - different requirements
- Traffic aggregation:
  - core: higher aggregation
  - many different sources
  - hard to model
- Routing/forwarding:
  - destination-based
  - single metric for all traffic
  - queuing effects
- Large packet size:
  - good for general data
  - “router friendly”
  - “slows” real-time traffic
- Small packet size:
  - good for real-time data
  - less end-to-end delay
  - better tolerance to loss
  - (less jitter?)
  - less efficient (overhead)
  - “not router-friendly”

DigiComm II

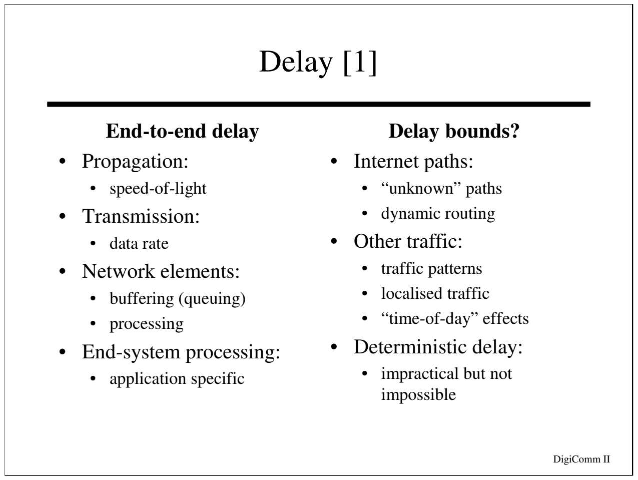
The mix of traffic is inevitable. There are many different types of applications, all with different requirements for how they might like their packets to be treated within the network. Indeed, a single application may be used differently by different users, so different instances of the same application may produce different traffic flows.

As traffic moves towards the core of the network, there is much higher multiplexing and so a much higher mix and aggregation of traffic. The huge number of sources and the extremely varied patterns of traffic, not only due to the sources but also due to the accumulated queuing effects due to upstream routers, makes it very hard to model traffic on the Internet. (See [PF97] and [WP98].) Remember in the traditional IP-based network, all traffic is treated in the same way, destination-based forwarding, all packets for the same destination pretty much sharing the same forwarding path and all routing information using a single metric.

One thing that does seem evident from the previous slide is that the differences in packet size of the different traffic types play an important role in how mixing of traffic affects individual flows. Large packets are efficient for non-real-time applications. For example, file transfer applications are happy with larger packet sizes, e.g. 1500bytes. However, large packets add delay to smaller packets from other sources (such as real-time audio flows) that they share queues with inside the network.

Smaller packets are much more suited to real-time applications, e.g. a real-time audio application may generate a packet that is of size 100bytes or less. This means that the the packet should have smaller transmission delay and so that flow as a whole has a smaller end-to-end delay. Also, losing a small packet – a small amount of data – is generally better than losing a large packet for a real-time application.

Also recall that the main “cost” of forwarding within an IP-based network is the per-packet cost of making a forwarding decision and then queuing a packet to the output at each router hop, so large packets that transfer lots of data per packet can, in this context, be considered more “router-friendly” than many small packets.



Delay [1]

End-to-end delay

- Propagation:
  - speed-of-light
- Transmission:
  - data rate
- Network elements:
  - buffering (queuing)
  - processing
- End-system processing:
  - application specific

Delay bounds?

- Internet paths:
  - “unknown” paths
  - dynamic routing
- Other traffic:
  - traffic patterns
  - localised traffic
  - “time-of-day” effects
- Deterministic delay:
  - impractical but not impossible

DigiComm II

On factor of great importance to interactive applications is end-to-end delay. This is certainly true for real-time applications using voice or video streams. Across a network such as the Internet, the end-to-end delay is made up of many components.

• **propagation delay:** this is also called “speed-of-light” delay, and is a physical constraint that is linked to the propagation of a physical signal. In general, the speed of light, *c*, is taken to be approximately 3.0×10^{8}m/s, and this is often used in calculations. However, it should be noted that in copper the propagation speed of an electrical signal is nearer 2.5×10^{8}m/s, whilst in fibre the propagation speed of an optical signal is nearer 2.0×10^{8}m/s.

•**transmission delay:** this is the delay in getting bits onto the wire due to the speed of the link. Do not confuse this with propagation delay. A 1000byte packet is transmitted “faster” on a 10Mb/s line than on a 64Kb/s link, i.e. the bits are placed onto the wire quicker, but the electrical signal is subject to the same propagation delay in both cases.

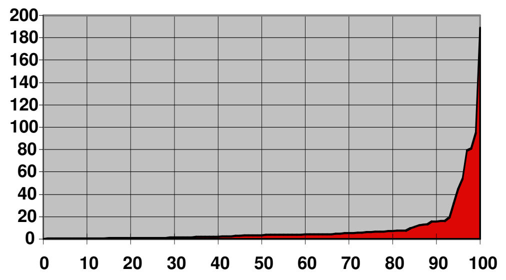
• **network element processing delay:** a packet arriving at a network element may be queued a an input buffer, then it will be read and processed (e.g. forwarding decisions made at routers), and finally queued to an output buffer while it waits to be transmitted.

• **end-system delay:** delay may be introduced at the sender or receiver for various reasons. The input may not be processed immediately or transmitted immediately at the sender, for example due to end-system load. At the receiver, delay may be introduced in order to compensate for network affects, e.g. the use of de-jitter buffers in real-time audio tools.

The end-to-end path that traffic follows across the Internet is never fixed for any application. In general, the application neither knows or cares about the actual end-to-end path. Changes to the the path occur due to the dynamic nature of the IP routing protocols, and do not forget that paths may not be symmetric delay may be asymmetric. Traffic patterns may be observed to have effects that are localised, e.g. localised congestion, as well as “time-of-day” effects.

(See [PF97a] and [PF97b] for a discussion of observed effects of routing and packet dynamics.)

Delay [2] #picture#



DigiComm II

Jitter (delay jitter) [1]

End-to-end jitter

- Variation in delay:
  - per-packet delay changes
- Effects at receiver:
  - variable packet arrival rate
  - variable data rate for flow
- Non-real-time:
  - no problem
- Real-time:
  - need jitter compensation

Causes of jitter

- Media access (LAN)
- FIFO queuing:
  - no notion of a flow
  - (non-FIFO queuing)
- Traffic aggregation:
  - different applications
- Load on routers:
  - busy routers
  - localised load/congestion
- Routing:
  - dynamic path changes

DigiComm II

Jitter is the **delay variation** observed at the receiver. Packets do not arrive with constant delay so the timing of packet generation at the sender is perturbed and timing needs to be re-constructed at the receiver – this is called synchronisation. The effects at the receiver are application dependent, but what is visible is a variable packet arrival rate, and therefore a variable data rate for the flow. This is not suitable for application such as audio which produce flows that may have a constant data rate. For non-real-time applications, jitter is typically not an issue. For real-time applications, jitter compensation needs to be applied at the receiver.

Jitter is caused by a number of factors. At the sender, use of LAN technology like Ethernet may lead to packet transmissions being unevenly spaced. In routers, the use of FIFO queuing and traffic aggregation may lead to packet spacing being perturbed. Some routers may also use non-FIFO techniques, perhaps prioritising certain traffic (e.g. policy-based, using source address), and so disrupting the normal spacing of other flows. Traffic aggregation, with many different size packets sharing the same buffers/output-link may also cause jitter.

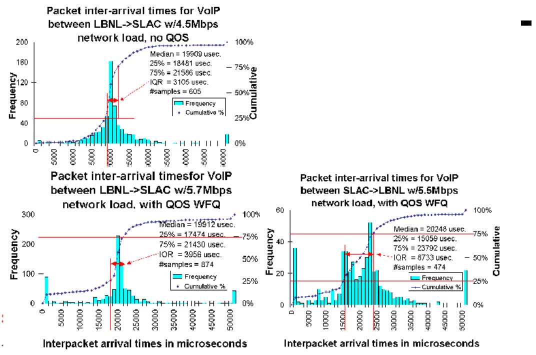
As router load increases, buffers/queues in routers may start to fill up, adding queuing delay. Very busy routers may lead to (localised) congestion, and this may lead to further delay. Where congestion or router failure leads to dynamic routing changes, packets may find themselves traversing different network paths between the same source and destination. This causes delay variation. Congestion in the network can lead to routing instability, and **route flapping** – dynamic changes routes from A → B → A – may occur.

Jitter (delay jitter) [2] #picture#

- Easiest measure is the variance in inter-packet delay
- Can use lots of other metrics (e.g. other moments of the inter-arrival time distribution
- Not critical to protocols like TCP unless jitter is 1st order (I.e. not 2nd order) effect
- Critical to voice
- (e.g. playout buffer to make sure D2A device or display is not starved…)
- VOIP, Video over IP
- Critical for interactive

DigiComm II

VoIP jitter LBNL<=>SLAC with load



DigiComm II

Loss [1]

End-to-end loss

- Non-real-time:
  - re-transmission, e.g.: TCP
- Real-time:
  - forward error correction and redundant encoding
  - media specific “fill-in” at receiver
- Adaptive applications:
  - adjust flow construction

Causes of loss

- Packet-drop at routers:
  - congestion
- Traffic violations:
  - mis-behaving sources
  - source synchronisation
- Excessive load due to:
  - failure in another part of the network
  - abnormal traffic patterns, e.g. “new download”
- Packet re-ordering may be seen as loss

DigiComm II

Loss generally occurs because of congestion somewhere in the network. Congestion at an individual router occurs when it does not have enough resources to deal with the number if incoming packets and so has to discard packets. For non-real-time applications, loss is normally not a problem. For example, applications using TCP rely on TCP’s re-transmission mechanisms and congestion control to ensure that data does get through. However, retransmission schemes are generally unsuitable for real-time traffic, and receiver-side mechanisms (such as forward error correction, redundant encoding and “fill-in” schemes) are used.

Loss is caused by packet drop ate routers, Router do not have enough input-buffer space or output-buffer space to deal with the number of packets they have received and some must discard some packets. Such congestion can occur at at traffic aggregation points (perhaps due to insufficient provisioning) or at the ingress to “busy” network/server sites. Traffic violations from sources may cause congestion, if traffic ingress is not policed. In some cases, even with well behaving, policed sources, there may still be congestion if many sources go to “peak” load at the same time – source synchronisation.

Excessive load at a point in the network maybe due to under-provisioning, traffic being re-routed because of a failure elsewhere in the network or sometimes due to abnormal traffic patterns, for example due to many people downloading the latest piece of software from a popular server.

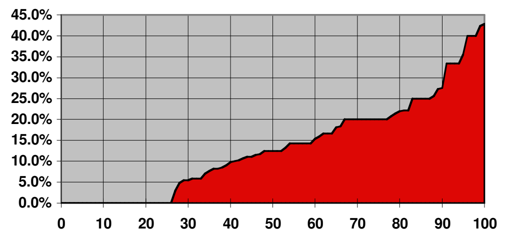
Another important effect to note is that, as viewed from the receiver, certain modes of packet re-ordering may seem like loss to the user. For example, consider an application that is waiting to receive packets A, B and C. A and C arrive, and B has not been lost but just re-ordered and will arrive shortly. However, the application can not wait and so perhaps will “fill-in” or correct fro the absence of B in an application-specific manner.

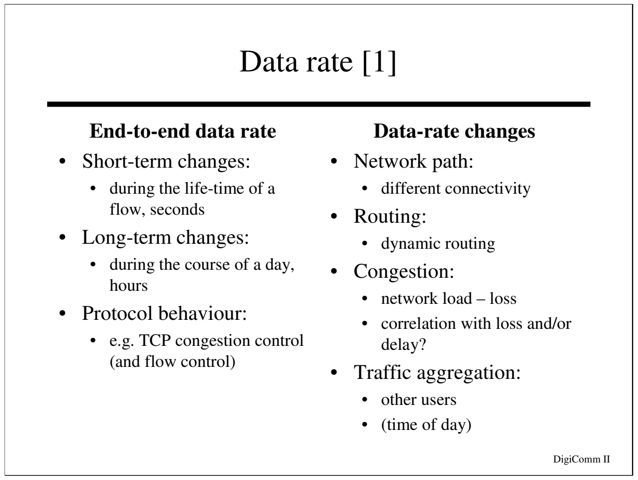
Loss [2] #picture#

- Varies with route
- Depends on link quality
- Depends on router quality
- Varies with time
- Depends on other load
- And interference

DigiComm II

Error rates





Data rate [1]

End-to-end data rate

- Short-term changes:
  - during the life-time of a flow, seconds
- Long-term changes:
  - during the course of a day, hours
- Protocol behaviour:
  - e.g. TCP congestion control (and flow control)

Data-rate changes

- Network path:
  - different connectivity
- Routing:
  - dynamic routing
- Congestion:
  - network load – loss
  - correlation with loss and/or delay?
- Traffic aggregation:
  - other users
  - (time of day)

DigiComm II

The end-to-end data rate across the Internet varies tremendously. Changes are observable at just about every timescale, from second to hours. There are absolutely no guarantees with respect to data rate, and fluctuations can occur at any time. These fluctuations are typically as the result of the complex interaction of network elements and traffic. At the application-level they may also be seen due to the action of transport protocols (or application-specific control), for example TCP congestion control. (note that in TCP, although flow control also affects the end-to-end data rate, this is due to the action of the receiver and not the network, so we do not consider it here.)

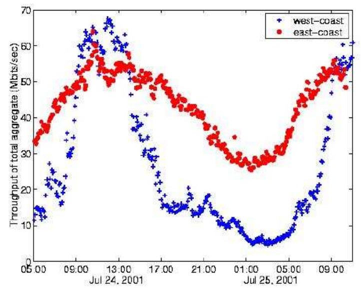
Data rate changes are due to similar reasons as for delay and loss: network connectivity, dynamic changes in routing resulting in a change of the end-to-end path, changes network traffic leading to generally higher network load and congestion. The change in value of the end-to-end data rate may often be closely correlated with the changes in end-to-end delay and/or observed loss, but this is not a general rule.

Data rate changes can be highly visible as time-of-day effects due the the network usage of other users. For example, from the UK try browsing a WWW server on the west coast of the US early on a Sunday morning and you should see a good data rate. Try the same server again at 3.00pm on a Monday afternoon and you will notice a huge difference in the end-to-end data rate.

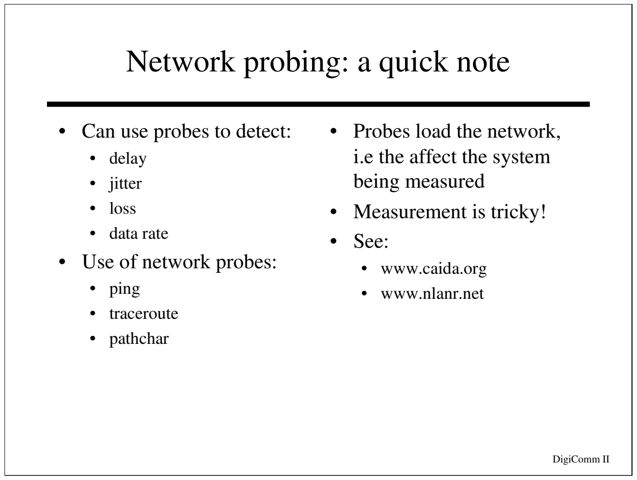
Data rate [2] #picture#

- Link is multiplexed so rate is not just link speed
- Varies with overall load depending on the link sharing rules
- Note latency (RTT) contributed to throughput (e.g. if you have to wait for acks…)
- Lots of different possibile basic link speeds depending on media, modulation, and protocol stack overheads
- "Goodput" is often used for the residual throughput after you allow for all overheads (including retransmissions)

DigiComm II



DigiComm II



Network probing: a quick note

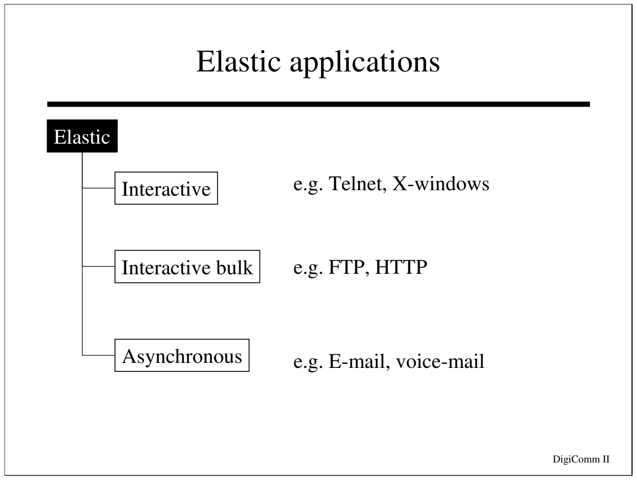
- Can use probes to detect:
  - delay
  - jitter
  - loss
  - data rate
- Use of network probes:
  - ping
  - traceroute
  - pathchar
- Probes load the network, i.e the affect the system being measured
- Measurement is tricky!
- See:
  - www.caida.org
  - www.nlanr.net

DigiComm II

It is possible to using probe techniques to determine the value of certain parameters on an end-to-end path. Probing techniques work by an application generating, transmitting and receiving specially formatted packets, e.g. ICMP packets as used by ping, traceroute and pathchar. These specially formatted packets are sometimes called probes. They travel along the network path and the application can determine the values of certain parameters by looking the the response to the probe from the network. Note that in order to do this, the application is actually loading the network – it introduces additional traffic onto the network – in order to determine what is happening. Measuring performance and QoS in IP-based networks and on the Internet as a whole is still a research issue. Many other tools and techniques are available, and you can find information about some of them at:

• Cooperative Association for Internet Data Analysis: http://www.caida.org/

• National Laboratory for Applied Network Research: http://www.nlanr.net/



Elastic applications

DigiComm II

Elastic applications are those applications that can tolerate relatively large delay variance – essentially the traditional data applications. They work best with low delays but they do not become unusable when delays increase or vary somewhat. Throughput may or may not be an issue. The basic requirement for these applications generally is that that they receive a reliable, ordered end-to-end data delivery service.

Interactive applications require near real-time, human interaction and ideally would like to have delays of 200ms or less, but could possibly tolerate slightly more.

Interactive bulk applications will also have similar requirements for delay, but they may also have high throughput requirements. Generally, the user is willing to tolerate a delay that is roughly proportional to the volume of data being transferred.

Asynchronous applications may or may not involve bulk data (e.g. e-mail), but they are prepared to tolerate much greater delay as they are generally store-and-forward type applications and the user does not expect anything close to real-time interaction.

Examples of elastic applications

- E-mail:
  - asynchronous
  - message is not real-time
  - delivery in several minutes is acceptable
- File transfer:
  - interactive service
  - require "quick" transfer
  - "slow" transfer acceptable
- Network file service:
  - interactive service
  - similar to file transfer
  - fast response required
  - (usually over LAN)
- WWW:
  - interactive
  - file access mechanism(!)
  - fast response required
  - QoS sensitive content on WWW pages

DigiComm II

Examples of "traditional" datacomms services are e-mail, file transfer, network file services and WWW.

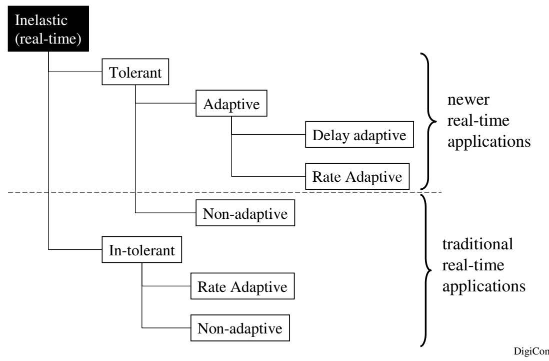
E-mail is the least demanding of these services in terms of timeliness and interaction. E-mail messages can be read and prepared off-line (without a network connection). The network connection is required only when sending or receiving messages. When sending messages, it is sufficient that the e-mail service manages to delver your mail "sometime soon". Typically this is the order of several minutes (or less). However, in many cases, even a delivery time of several hours is sufficient. Asynchronous means (literally) that there is no synchronisation between the sender and receiver of an e-mail message.

With file transfer, what is required is that you can retrieve or send files to a remote server. Generally, the user must somehow interact with the server so some timeliness of response to commands is required. When a file is transferred, the user would like to have the transfer take place "as quickly as possible" but the user's value in use of the service is not severely depreciated if the transfer happens to be slightly "slower".

Network file service can be though of as an enhanced version of file transfer. Here, files that are on a remote server are manipulated as if they were local to the end-system at the which the user is physically situated. Of course, a fast response to commands and fast file transfer are vital to this service. This is typically not a problem as network file services are normally run within an office environment where there is a high data rate available. Again, "slower" transfers may be acceptable but the service would still perform a useful function.

The WWW is essentially an interactive, non-real-time service. Web pages are documents that can be access from a remote server. In some ways, there are elements similar to file transfer and network file service here. However, the interaction is very important here – web pages should appear quickly and hyperlinks that are selected ("clicked") should be responded to quickly also. Web pages often now contain links to other content such as streaming audio and video. This is not really interactive but may have fairly strict requirements with respect to QoS (e.g. loss, available data rates, etc.).

Inelastic applications



DigiComm II

Inelastic applications are comparatively intolerant to delay, delay variance, throughput variance and errors. If QoS is not sufficiently well controlled, the applications become unusable.

Tolerant applications can be adaptive or non-adaptive. Tolerant adaptive applications may be able to:

• adapt to delay variation: a voice application might adapt to a decrease in the average delay by dropping a few packets to allow the transmitter to catch up with the receiver.

• adapt to data rate variation: a video application may be able to adjust the encoding and trade quality against throughput.

Tolerant non-adaptive applications cannot adapt – but can still tolerate – some QoS variation. A voice application may still be usable with a measure of packet loss.

Intolerant applications cannot tolerate QoS variation – for example real-time control of a robot arm. Some may be rate-adaptive being able to adjust to detected changes in throughput, but others are totally non-adaptive.

Examples of inelastic applications

- Streaming voice:
  - not interactive
  - end-to-end delay not important
  - end-to-end jitter not important
  - data rate and loss very important
- Real-time voice:
  - person-to-person
  - interactive
- Important to control:
  - end-to-end delay
  - end-to-end jitter
  - end-to-end loss
  - end-to-end data rate

DigiComm II

Inelastic applications are generally those that involve some sort of QoS sensitive media, such as voice.

If we have a look first at non-real-time voice, e.g. streaming audio. Here, large end-to-end delay and delay variation – **jitter** – does not matter greatly as the media flow is not interactive. There are well-known mechanisms that can adjust for jitter at the application-level. What is important is that a certain data rate is maintained that there is relatively low packet loss.

For real-time audio, e.g. person-to-person, the data rate and packet loss remain important but the end-to-end delay and end-to-end jitter are now quite important. There should be a constant and flowing dialogue in order for human voice interaction to work. If the delay is large (e.g. over half a second), conversation becomes difficult. (You may have experienced this sometimes on very-long distance phone calls, e.g. to Asia or Australasia from the UK.)

In fact, there are similar requirements for streaming and real-time video, respectively, as the are for streaming and real-time voice. However, humans tend to be much less tolerant to packet loss and variations in data rate in video traffic than in voice traffic.

Notice also that the requirements for **real-time** voice and video are different from those of **streaming** voice/video. For real-time voice and video, we assume that the media streams are being created in real-time and are being used in an interactive manner, e.g. a **conversation**. Applications that are providing conversational services have more stringent QoS constraints than applications that are processing streaming media (e.g. voice/video playback). The latter can cope (to some extent) with variations in data rate, packet loss rates and delay.

QoS parameters for the Internet [1]

Delay

- Not possible to request maximum delay value
- No control over end-to-end network path
- Possible to find actual values for:
  - maximum end-to-end delay, D_{MAX}
  - minimum end-to-end delay, D_{MIN}

Jitter

- Not possible to request end-to-end jitter value
- Approximate maximum jitter:
  - $D_{MAX} – D_{MIN}$
  - evaluate D_{MIN} dynamically
  - D_{MAX}? 99th percentile?
- Jitter value:
  - transport-level info
  - application-level info

DigiComm II

In general, requests for particular delay and jitter constraints by an application are not parameters that can be honoured across the Internet. That is not to say they could not be honoured by an IP-based network, e.g. it might be possible to arrange for such mechanisms in an IP-based corporate VPN. As we have noted, there are many autonomous systems that are linked together to form the Internet and it is not possible implement a homogenous environment across them. So, typically, it may be possible for an application to request "low delay" and "low jitter" and it may be possible for the network to give an indication of what the delay/jitter is along a particular network path, e.g. a value for maximum end-to-end delay can be found with the use of RSVP/INTSERV as we will see later.

So, it may be possible to find the maximum possible delay bound by querying network elements along a certain end-to-end path, and it should be possible to evaluate current (and mean) delay by using network probes (a "ping"-type of scheme). However, jitter is very much harder to evaluate. This, once again, is because the end-to-end path may consist of concatenation of various network technologies and routers with different behaviour. However, with an upper bound for the delay, D_{MAX}, and a lower bound for delay, D_{MIN}, it could be argued that a reasonable estimate for the jitter is $D_{MAX} – D_{MIN}$. D_{MIN} can be evaluated dynamically during the operation of the flow, perhaps from application-level header information or from protocol headers as in RTP/RTCP (we look at RTP/RTCP later).

D_{MAX} may also be evaluated dynamically, form the same measurements used to determine D_{MIN}, however, it may be that such an estimate for D_{MAX} results in an estimate that is too conservative. Other summaries may be used, e.g. the 99th percentile for the measured delay values. Evaluations of both D_{MIN} and D_{MAX} are likely to be application specific.

QoS parameters for the Internet [2]

Loss

- Not really a QoS parameter for IP networks
- How does router honour request?
- Linked to data rate:
  - hard guarantee?
  - probabilistic?
  - best effort?
- (Traffic management and congestion control)

Packet size

- Restriction: path MTU
- May be used by routers:
  - buffer allocation
  - delay evaluation

DigiComm II

It should be possible to specify loss characteristics as QoS parameters. However, it may not be practical to ask for a particular numerical value for loss, e.g. asking for 10^{-4} packet loss rate. Note that packet loss decreases the end-to-end data rate. So, specification of loss may be linked to the specification of how a data rate quest if honoured. For example, one could ask for a data rate to be guaranteed (no loss), have "low" loss (low probability of packet loss) or best effort. One reason for not using loss rates is that monitoring per-flow loss rates is taken to be a computationally expensive exercise for routers to perform in practise. So, in IP-based networks such as the Internet, packet loss rate is not usually specified numerically. (In ATM networks, or Frame Relay networks, numerical values for cell loss rates and frame loss rates may be specified, respectively.)

Although packet size may not be considered specifically as a QoS parameter, the maximum (and perhaps the minimum) packet size that an application will generate may be specified. Of course the maximum packet size will be restricted by the path MTU, and real-time applications generally do not exceed the path MTU as IP-packet fragmentation is normally considered harmful for real-time applications. Routers may find the specification of maximum and minimum packet size useful, e.g. for the Guaranteed Service in INTSERV as we will see later.

QoS parameters for the Internet [3]

- Data rate:
  - how to specify?
- Data applications are bursty:

$$\frac{\text{peak data rate}}{\text{mean data rate}} >> 1$$

- Specify mean data rate?
  - peak traffic?
- Specify peak data rate?
  - waste resources?
- Real-time flows:
  - may be constant bit rate
  - can be variable bit rate
- Application-level flow:
  - **application data unit (ADU)**
- Data rate specification:
  - application-friendly
  - technology neutral

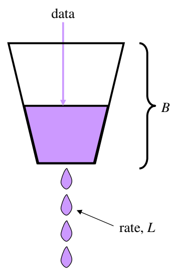
DigiComm II

For many IP-based application, the most precious resource is data rate. So, specification of data rate as a QoS parameter is essential. However, how is data rate to be specified? remember that traditional data applications may produce a very varied traffic profile. Also, while we have generally considered real-time applications as producing constant bit-rate flows, more sophisticated audio and video coding techniques produce variable bit rates. So do we specify mean rates, and then lose data when there are peaks above the mean? Or do we specify peak rates, and then make inefficient use of the network resources when we operate below the peak rate?

Multimedia applications generate a lump of data that can be considered as an **application data unit** (**ADU**). We need to have a specification mechanism can reflect the way that the application may generate data. It should also be something that is technology neutral, as IP itself is. That is, the specification should not rely on any particular mechanisms or functions in levels 1 (physical) and 2 (data link).

Leaky bucket

- Two parameters:
  - *B*: bucket size [Bytes]
  - *L*: leak rate [B/s or b/s]
- Data pours into the bucket and is leaked out
- *B*/*L* is maximum latency at transmission
- Traffic always constrained to rate *L*



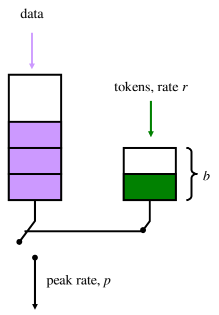
A traffic control mechanism used by ATM is leaky bucket. Here, there is a fixed data rate, effectively the peak rate of transmission, which is *L*. However, to cope with bursts of data, a bucket size, *B*, is also defined. The model is that data arrives in the bucket and is drained from the bucket at the rate *L*. Bursts of traffic that result in an overflow of the bucket are discarded.

Note that the bucket may be filled in ADU sized lumps, so it is probably sensible to arrange for the *B* to be a a multiple of the transmitted ADU size. The maximum latency introduced by the bucket is *B*/*L*. Note that the traffic is always constrained to rate *L* by the leaky bucket, even if greater capacity is available, i.e. a higher peak rate may be possible at a given time. Leaky bucket is used in ATM, where *B* and *L* be specified in units of cells and cells per second, respectively.

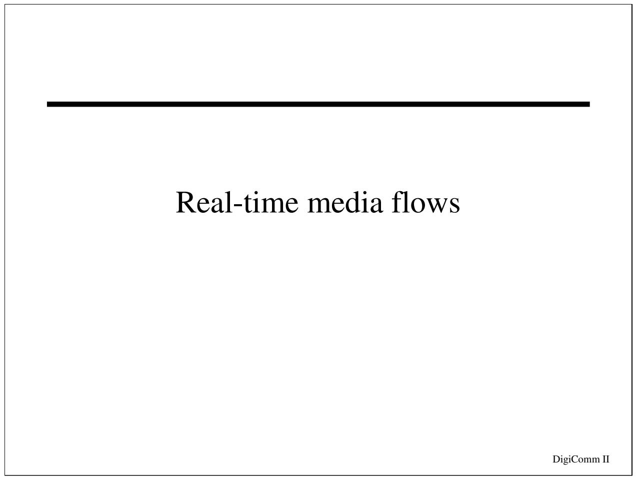
Token bucket

Token bucket

- Three parameters:
  - *b*: bucket size [B]
  - *r*: bucket rate [B/s or b/s]
  - *p*: peak rate [B/s or b/s]
- Bucket fills with tokens at rate *r*, starts full
- Presence of tokens allow data transmission
- Burst allowed at rate *p*
- data sent < *r*t + *b*

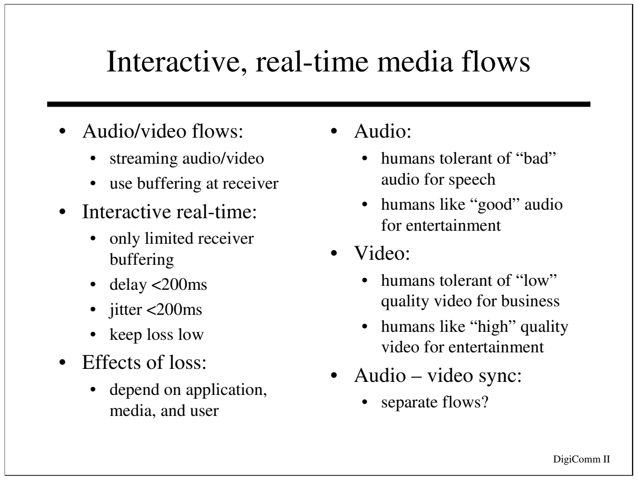


There is a subtle, and very important, difference between a leaky bucket and a token bucket. The token bucket also has a bucket size, *b*, and a bucket rate, *r*, but it allows traffic bursts to be transmitted at peak rate, *p*, under certain conditions. In this case, the bucket does not “fill with data” as it does in the leaky bucket, but it fills with tokens, that that allow data to be transmitted. Data can only be transmitted when there are enough token to allow transmission to take place. Transmission can then take place at a peak rate of *p*. Nominally, data is transmitted at a rate *r*, the same rate at which the bucket is filled with tokens. However, it can be seen that bursts of traffic, up to the bucket size, can be transmitted at the peak rate, *p*.



Real-time media flows

DigiComm II



Interactive, real-time media flows

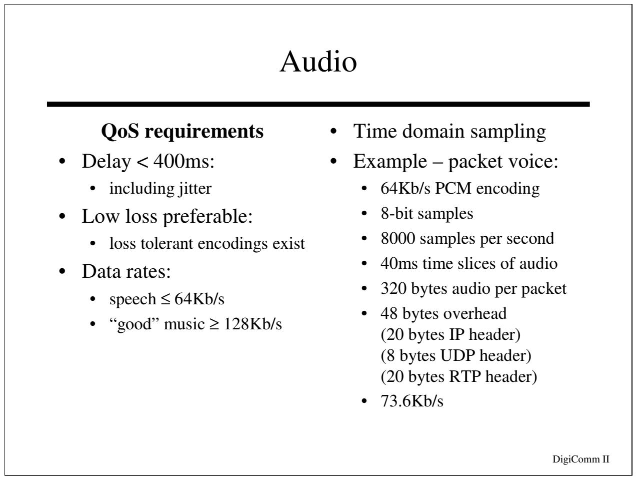
- Audio/video flows:
  - streaming audio/video
  - use buffering at receiver
- Interactive real-time:
  - only limited receiver buffering
  - delay <200ms
  - jitter <200ms
  - keep loss low
- Effects of loss:
  - depend on application, media, and user
- Audio:
  - humans tolerant of “bad” audio for speech
  - humans like “good” audio for entertainment
- Video:
  - humans tolerant of “low” quality video for business
  - humans like “high” quality video for entertainment
- Audio – video sync:
  - separate flows?

DigiComm II

People often think of real-time flows as “audio and video”. We must make a distinction between **streamed** audio and video flows and **real-time**, interactive audio and video flows. For streamed audio and video, while low delay, low jitter, low loss and high data rate are desirable for playback of the flow, under normal circumstances, only the latter (data rate) is significantly important that it might be considered a QoS issue. We do require large amounts of capacity to transmit high quality streamed audio and video. However, end-to-end delay is not an issue, even if this may be in the order of a few seconds but for one –way transmission - playback - it should not matter. Jitter and loss can be compensated having a large buffer at the receiver for delaying playback at the receiver to allow time for smoothing unevenly spaced packet arrival as well as dealing with re-transmissions of lost data as required. This scenario may be complicated in a multicast scenario (reliable multicast is a tricky issue as we will see later).

Real-time audio and video normally involves humans as the subjects of audio/video flows. Humans use applications that generate audio and video flows to interact across a network. Small “timeslices” of audio/video are placed into packets and then sent across a network. Generally, for human interaction to continue in a “conversational” manner, delay and jitter must be kept to around 200ms each, i.e. the maximum end delay should be no more than around 400ms. The rules for loss are less easy to specify so easily, as loss effects tend to be application-specific, media-specific and user-specific. For example, let us consider a fictitious video-telephone (VT) application being used by a human on a laptop machine, ad the user is travelling away from home. That user may find it appropriate to use only low quality audio when talking to someone back at the office to check on deliveries, then use high quality audio (and perhaps some video) when giving a report to his/her boss, and use high quality and use high quality video and audio when talking to his family. Also, hu,mans are generally tolerant of low quality audio and video for “normal” (business or domestic) usage, but for entertainment, they generally want high quality audio and video. Some other applications, e.g. medical applications, may require high quality audio and video at all times.

Also, if audio and video flows are transmitted separately, then the receiver may need to re-synchronise the flows for playback.



Audio

QoS requirements

- Delay < 400ms:
  - including jitter
- Low loss preferable:
  - loss tolerant encodings exist
- Data rates:
  - speech ≤ 64Kb/s
  - “good” music ≥ 128Kb/s

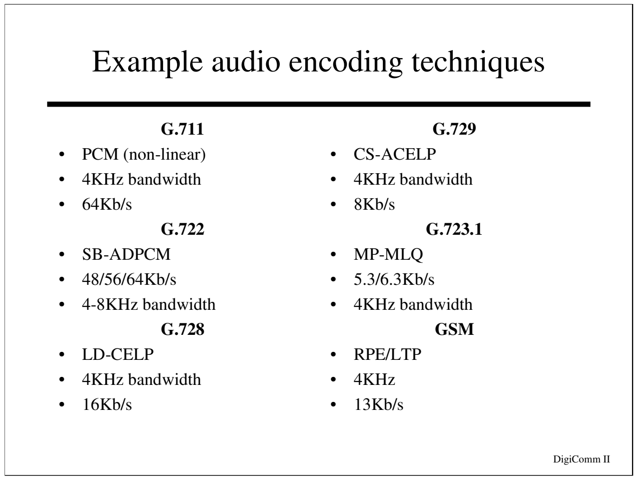
- Time domain sampling
- Example – packet voice:
  - 64Kb/s PCM encoding
  - 8-bit samples
  - 8000 samples per second
  - 40ms time slices of audio
  - 320 bytes audio per packet
  - 48 bytes overhead
    (20 bytes IP header)
    (8 bytes UDP header)
    (20 bytes RTP header)
  - 73.6Kb/s

DigiComm II

We discuss briefly the QoS requirements for real-time audio. For audio, we normally require that delay is less than 400ms end-to-end for any packet, and this includes any jitter. It is preferable to have low loss, but there are mechanisms to compensate for loss, either by use of special audio encoding techniques, or by “fill-in” techniques at the receiver. Telephone quality speech is achievable in audio encodings of 64Kb/s or less. Reasonable quality music (for entertainment) may require at least twice this rate.

Audio is taken to be a single dimensioned variable, and is normally sampled in the time domain.

When it is transmitted as packet voice over IP, a typical dat rate required for the application may be approximately 74Kb/s.



Example audio encoding techniques

G.711

- PCM (non-linear)
- 4KHz bandwidth
- 64Kb/s

G.722

- SB-ADPCM
- 48/56/64Kb/s
- 4-8KHz bandwidth

G.728

- LD-CELP
- 4KHz bandwidth
- 16Kb/s

G.729

- CS-ACELP
- 4KHz bandwidth
- 8Kb/s

G.723.1

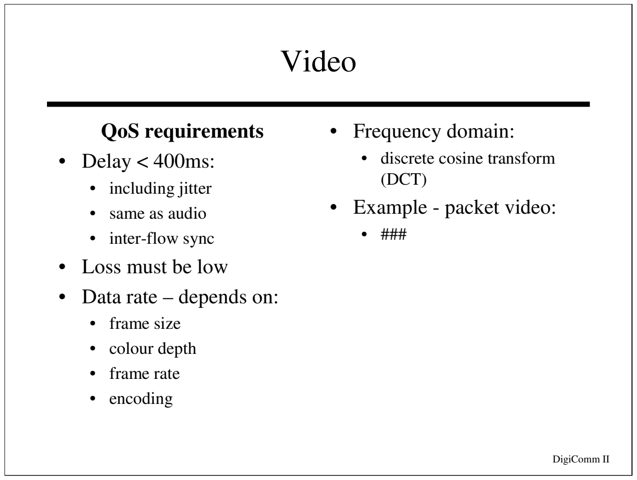
- MP-MLQ
- 5.3/6.3Kb/s
- 4KHz bandwidth

GSM

- RPE/LTP
- 4KHz
- 13Kb/s

DigiComm II

LD-CELP:	low delay code excited linear prediction
CS-ACELP:	conjugate structure algebraic excited linear prediction
ML-MLQ:	multi-pulse maximum likelihood quantisation
PCM:	pulse code modulation
RPE/LTP:	residual pulse excitation/long term prediction
SB-ADPCM:	sub-band adaptive differential pulse code modulation



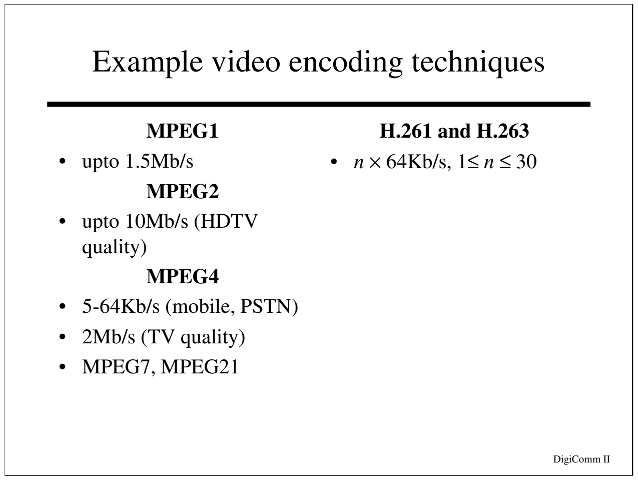
Video

QoS requirements

- Delay < 400ms:
  - including jitter
  - same as audio
  - inter-flow sync
- Loss must be low
- Data rate – depends on:
  - frame size
  - colour depth
  - frame rate
  - encoding
- Frequency domain:
  - discrete cosine transform (DCT)
- Example - packet video:
  - ###

DigiComm II

We discuss briefly the QoS requirements for video. Loss and delay on video has a much more significant affect than on audio, especially if there is significant compression. Remember that compression removes redundancy, the very thing that we need for robustness in the face of loss. Typically, a whole screen of data may not fit into a single packet and so multiple packets may be required to make up a single frame of video. This means that it is possible for part of a picture to go missing. However, burst errors (loss of several packets close in sequence) may mean that several parts of the same picture could go missing.



Example video encoding techniques

MPEG1

- upto 1.5Mb/s

MPEG2

- upto 10Mb/s (HDTV quality)

MPEG4

- 5-64Kb/s (mobile, PSTN)
- 2Mb/s (TV quality)
- MPEG7, MPEG21

H.261 and H.263

- $n \times 64$Kb/s, $1 \leq n \leq 30$

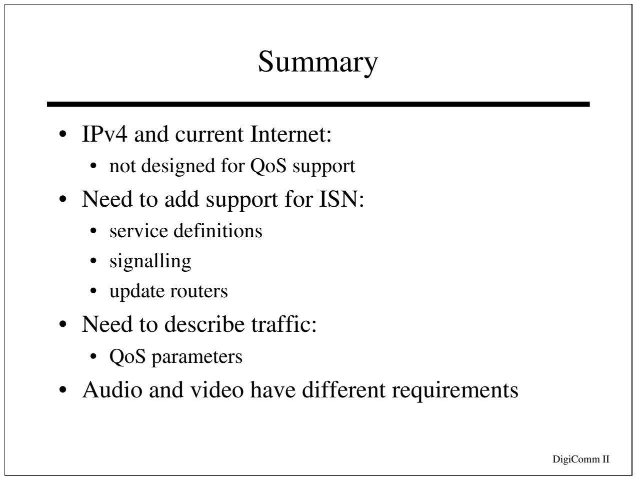
DigiComm II

MPEG moving pictures expert group

Commonly used picture sizes are given below.

Picture format	**Image size**	
sub-QCIF	128x96	
QCIF	176x144	
CIF	352x288	
4CIF	702x576	(full screen)
16CIF	1408x1152	

CIF	common intermediate format
QCIF	quarter CIF



Summary

- IPv4 and current Internet:
  - not designed for QoS support
- Need to add support for ISN:
  - service definitions
  - signalling
  - update routers
- Need to describe traffic:
  - QoS parameters
- Audio and video have different requirements

DigiComm II