MANCHESTER 1824
The University of Manchester

COMP28411 Computer Networks

Nick Filer

Multimedia – 2

Some material from:

Kurose & Rose – Chapter 7 + Slides

Halsall – Multimedia Communications

Multimedia 2 - Summary

- TCP or UDP?
- Transport layer 1 to ???? connections.
  - Unicast, broadcast, multicast, anycast.
- Quality of Service
  - Separating classes of media, policing traffic, scheduling
- Error /Loss recovery

MANCHESTER 1824
The University of Manchester

Problem ????

STREAMING?

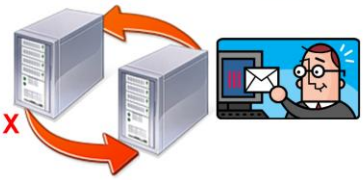
- Listening to pre-recorded music.
- Watching TV/Video/Films.

- We've already introduced some of the basics:
  - Download,
  - Buffering,
  - Jitter,
  - Playback

MANCHESTER 1824
The University of Manchester

TCP – Issues (TCP pros and cons?)

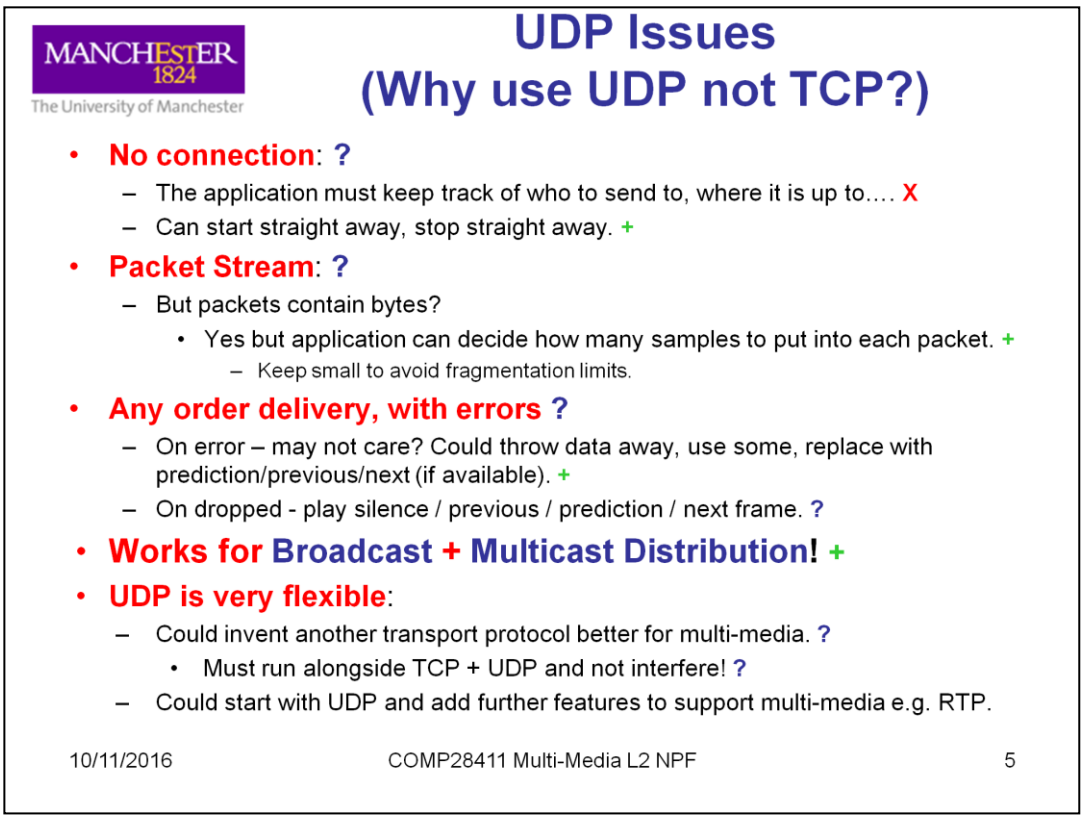
- **Ideal for data for Web, Database, Email, FTP, …!**
- **Connection**: ?
  - Everything is set up before use, torn down afterwards. ?
    - Must wait before starting to send data. X
  - But – used to maintain state. +
- **Byte stream**:
  - Data fragmented into packets not samples. X
  - TCP not application decides on packing. X



- **In order, error free delivery**: +
  - On error – must wait for successful re-transmission – long delay? X
  - On loss (dropped) – must wait for successful re-transmission – long delay? ?
  - Limited number of sequence number bits – not often an issue! ?
- **TCP Congestion Control X**
  - **Traffic delayed, thrown away, slowed down, stopped to control congestion at each router** (window size set to zero).

10/11/2016 COMP28411 Multi-Media L2 NPF 4

This should all be in your general knowledge by now? There are clearly pluses and minuses to using TCP for media downloads (good), streaming (problematic) and live viewing (not advised except in low loaded networks).



Many people think UDP is the ideal protocol to use because TCP has major problems whenever there is a late or non arriving packet. Other issues are with the throughput over high latency paths (e.g. from distant sources or from slow device transfers). Because UDP is so simple it lacks almost all the facilities applications need in order to operate effectively. Therefore, it is almost never used for media transfers on its own. Instead it is used in conjunction with other protocols such as RTP (Real Time Protocol) . UDP is also frequently used with multiple adjacent similar streams or different ones. For example, RTP is almost always used adjacent to RTCP (Real Time Control Protocol) on adjacent ports.

Because UDP packets can be any size up to 64K the packets can be arranged so they always contain a complete set of samples, samples are not split between packets. However, like all IP protocols UDP packets can be fragmented for sending over other networks. The Internet after all is a network of networks! Ethernets typically carry 1,500 byte of data in its frames (being a data link Layer protocol, Ethernet does not transmit packets but frames). Other networks use different frame sizes, ATM used for last miles links in the UK by many ISPs has XX byte frames so almost all data has to be fragmented for transport over ATM.

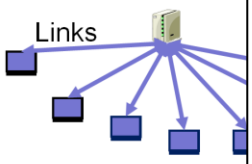
UDP packets are fired off into the network and instantly forgotten by the network stack on the source. The network will transport them perhaps over varying routes for delivery, if possible, to the destination. Packets can arrive in any order. Quite often packets will be dropped by busy intermediate routers. TCP deals with this by buffering packets in the transport layer, UDP users can restore order at the application layer.

Having no connection there is no built-in control so applications using UDP must do all the control that is built-in to TCP. Of course, this means applications can choose what features they want but at the cost of extra coding and that code possibly varying as it is not almost statically integrated into a network stack.

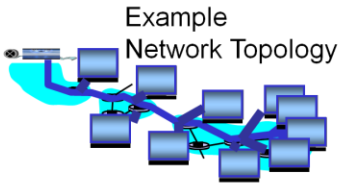
Other missing or optional features of UDP are a Cyclic Redundancy Check (CRC) which is optional and often not used and there is no sequence number or timing information.

MANCHESTER 1824
The University of Manchester

Streaming Issues?



- 1 to 1
  - Point to Point download and play.
  - Unicast.
  - Protocol options: TCP, UDP
- 1 to Many
  - Unicast: Running many concurrent 1:1 connections server to many clients can never be sufficient. Can it?
  - Multicast: Send just one copy. Let routers replicate so each user gets 1 copy. BUT needs synchronization. Complex to allow seeking (jump back/forward) for watchers. Great for live streaming.
- Many to Many
  - Video/Music on demand. Big players Amazon, Netflix, BBC … all successfully offering services. How?



10/11/2016 COMP28411 Multi-Media L2 NPF 6

Streaming: A server sends continuous data to a number of clients simultaneously.

1to1 (see top right). Clients perceive that they are the only ones using the stream from the server. The server sends packets individually to each client. This can be TCP but has the pros and cons of TCP especially slow down in send rate in response to congestion. Or, could use unicast UDP which at the server provides a continuous fixed bit rate being sent. UDP packets during congestion, like TCP's are dropped but there is no detection of this by the server, no repeat sending, no rate adjustment. Hence, UDP is seldom used on its own. UDP is used with extra information and/or controls to provide sequencing order, exact timing data, semantic information about types of media streams. UDP media supply then used with other protocols and their separate streams to provide media control and client quality of service / experience information feedback to server.

The server may be sending many identical packets almost at the same time over the same communication links (at least close to or adjacent to the server). This is inefficient and easily overloads the network resources.

1 to Many: If lots of clients wish to receive the media. Network loading can be vastly reduced if the server just sends one packet which is delivered to many clients. The problem is when/where to replicate the packet. This is clearly an issue for routers (network layer) and switches (data-link layer). Broadcast packets are delivered globally (this can be scoped to not mean across the whole Internet) to everybody whether they want them or not. Multicast packets provide an opt-in service where clients request the multicast delivery stream and the routers decide how to replicate the packets so that a copy of each transmitted packets arrives at each client but minimizing network load by only replicating when packets must follow different paths.

The big problem is that often clients don't want to render the media simultaneously. Clients start, pause, stop, restart. This leads to either multiple streams leaving the server as in 1 to 1 via unicast or the streamed data must be stored (cached) either by the clients or in the network before reaching the clients.

Many to Many: These services provide many servers! They also tend to move the servers close to the clients when they can predict media demand.

MANCHESTER 1824
The University of Manchester

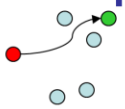
Problem ????

TRANSPORT LINK TYPES?

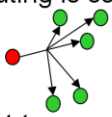
- Unicast - 1:1
- Broadcast - 1:ALL
- Multicast - 1:Many
- Anycast - 1:One of(Many)

MANCHESTER 1824
The University of Manchester

From Unicast to Broadcast



- Unicast
  - Packet sent from one host to one destination.
    - TCP does this, nothing else (why?)! UDP normally does this.
    - Most routing is configured for unicast.

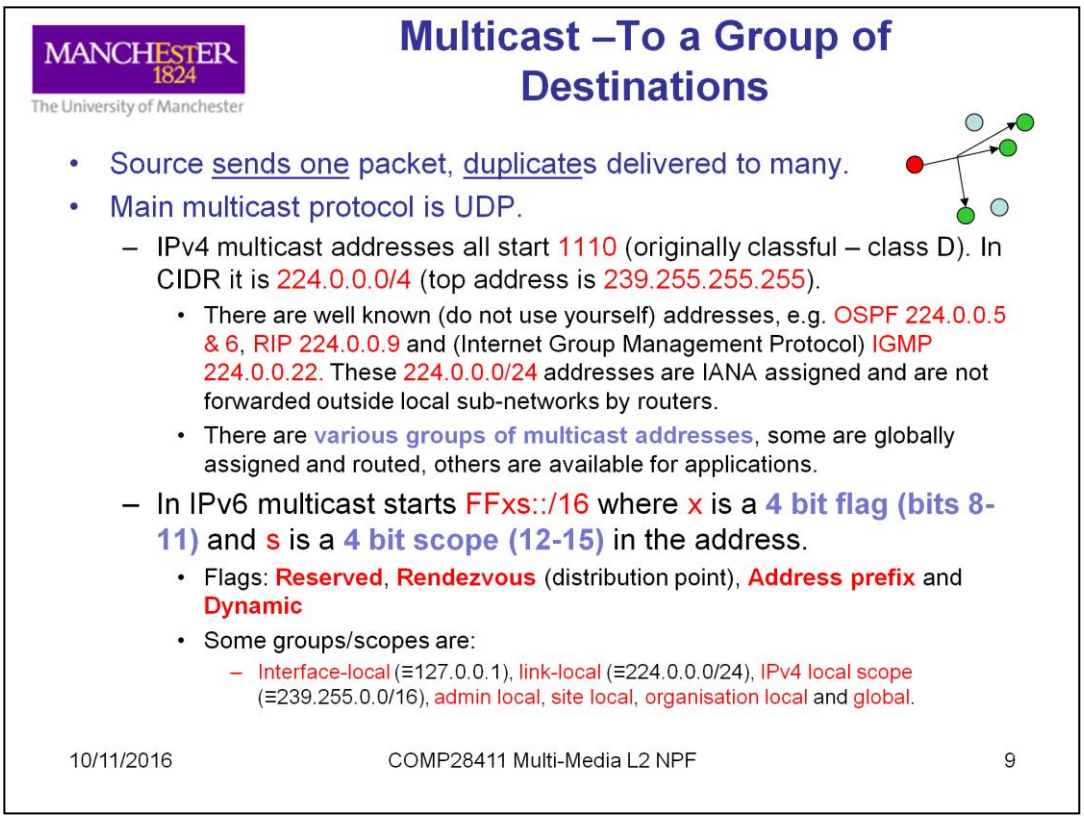


- Broadcast
  - Packet sent to everybody on IPv4, Ethernet etc. but not IPv6!
  - Address is the OR of IP subnet address and bitwise complement of netmask. 192.168.1.0/24 ➔ 192.168.254.255. The address 255.255.255.255 is the broadcast address for the 0.0.0.0 network = This Network.
  - Ethernet (other IEEE 802.x networks – WiFi is 802.11) uses the address FF:FF:FF:FF:FF:FF. IP broadcasts normally sent to this link-layer MAC address.
  - Broadcasts normally blocked at router = local boundary.

Unicast: Is for host to host over a single link. We normally think about connections as being unicast. TCP connections are always unicast! UDP and other stateless/stateful protocols can be unicast but may be able to support other modes.

Broadcast: Almost all networks have a broadcast address for traffic which goes to everybody! But IPv6 has done away with this idea because in 2^{128} hosts it does not seem to make any sense and would take forever and cause massive congestion. To avoid congestion it is very unusual to be able to broadcast from one Autonomous Service (AS) to others outside of the LAN. Routers block this traffic!

IPv6 uses multicast for broadcast like traffic but adds several scope controls to set just how widely the multicasts are sent.



CIDR – Classless Inter-Domain Routing

OSPF – Open Shortest Path First

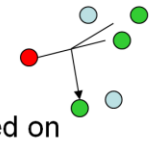
RIP – Routing Information Protocol

IANA – Internet Assigned Numbers Authority

An interesting point is a change between IPv4 and IPv6 for finding and keeping information about nearby neighbours. IPv4 uses local broadcasts to find neighbours with a local IP that are not in their known neighbour table or have not been conversed with recently. This is the Address Resolution Protocol (ARP) you have already seen in the laboratory and will see briefly again when we discuss the link-layer. In IPv6 ARP is replaced with the extended Neighbour Detection Protocol (NDP) which does the same task but has several extra functions which are mainly about monitoring mobile hosts. In IPv6 a LAN could be 2^{64} hosts! This is a lot of machines to bother when looking for just one machine! Therefore, what happens is that small groups of hosts form “Solicited Node” multicast groups. These multicast groups share the same final 24 bits of their unicast address but append these to the prefix FF02::1:FF00:0/104. We remove 104 bits. Because the last 24 bits are used, until we have LANs with 16 million machines each multicast group is in fact a 1:1 mapping. This 1:1 mapping gives much better efficiency than the IPv4 broadcast which every receiving machine has to process but most will not respond to. See: http://tinyurl.com/z5hmbjf

MANCHESTER 1824
The University of Manchester

Anycast



- One source talks to an advertised service hosted on multiple servers with the same IP address.
- Normally, layer 3 routing chooses the "nearest" target based on topology.
- Used for:
  - Reduced latency.
  - Reliability (the 1st choice server dies).
  - Attack resilience.
  - Load balancing & performance.
  - Simplified client configuration.

Who do you think uses "anycast"? Lets guess.

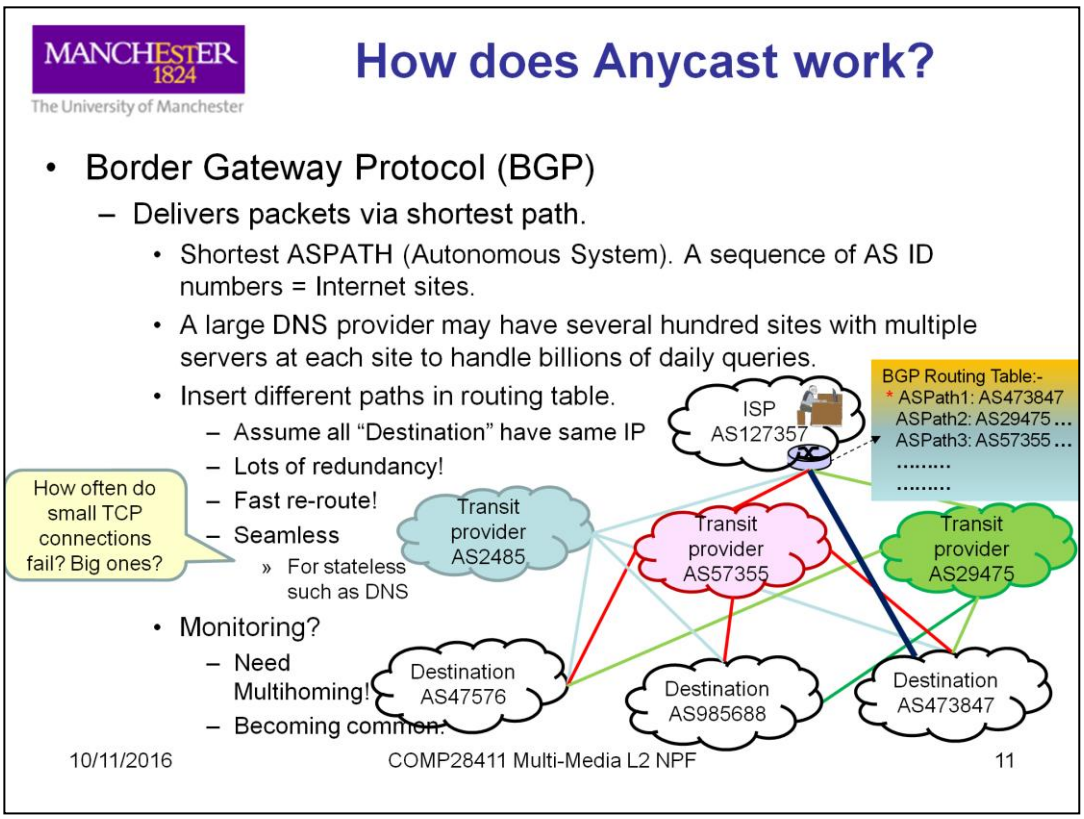
- How?
  - Change the DNS resolver. Instead of having a list & timeout between resolver options in list order. Have a group of servers sharing 1 IP address, return them round-robin style. See: http://tinyurl.com/nsyhvwl and http://tinyurl.com/zy6vf2a
  - Get the routers to return the "closest" route to the "anycast" IP.

10/11/2016 COMP28411 Multi-Media L2 NPF 10

Most big client-server setups – Google, Microsoft .....

Anycast provides important and now quite frequently used ways to distribute heavily used IP addresses to several locations either local or distant. Its implementation is such that almost always the server with the shortest path is used in preference to others. But, if this server fails then another will take on the role of having the shortest path. So, we now know how services such as Google can provide search to billions of users many times per day. They use vast numbers of servers distributed all around the world but sharing a single IP address. The server with the shortest Internet path normally responds.

In fact, the server that responds may be a bit more subtle to select. External connecting routers do most of the work between Internet sites. Within Internet sites, local routers and DNS arrange to load balance work between large farms of servers all sharing the same external IP address.



The BGP works between major Internet connected sites. At each site, it interfaces to an Autonomous System (AS) which has a unique ID number. So, for example, a very large (dynamic?) DNS supplier may map DNS for many millions of sites and handle many thousands of DNS queries per second. Even though the amount of work per query is not massive there is a limit to how many DNS queries a single server can handle in a given time. Therefore, the task of responding to the queries is normally spread between a number of servers. In some cases all the servers are located at the same AS site but in others they may be distributed all over the world. Users simply access the server using its published IP address. This IP address however maps to one or many servers all of which appear to share the same IP address.

Imagine that for the ISP at AS 12735 in the example, DNS is served by a server with the a shared IP address located at one of destinations AS47576, AS985688 or AS473847. Associated with the given IP address in the router tables will be several records giving the shortest ASPATH to DNS. In out example ASPATH1 has a length of 1, ASPATH2 a length of 2 to either of 2 different destination sites (random ordered?) and a 3rd ASPATH which also has, for simple unrealistic example, several path lengths of 2. BGP for the ISP will therefore always return a route to AS473847 provided this route and the server at the end is active.

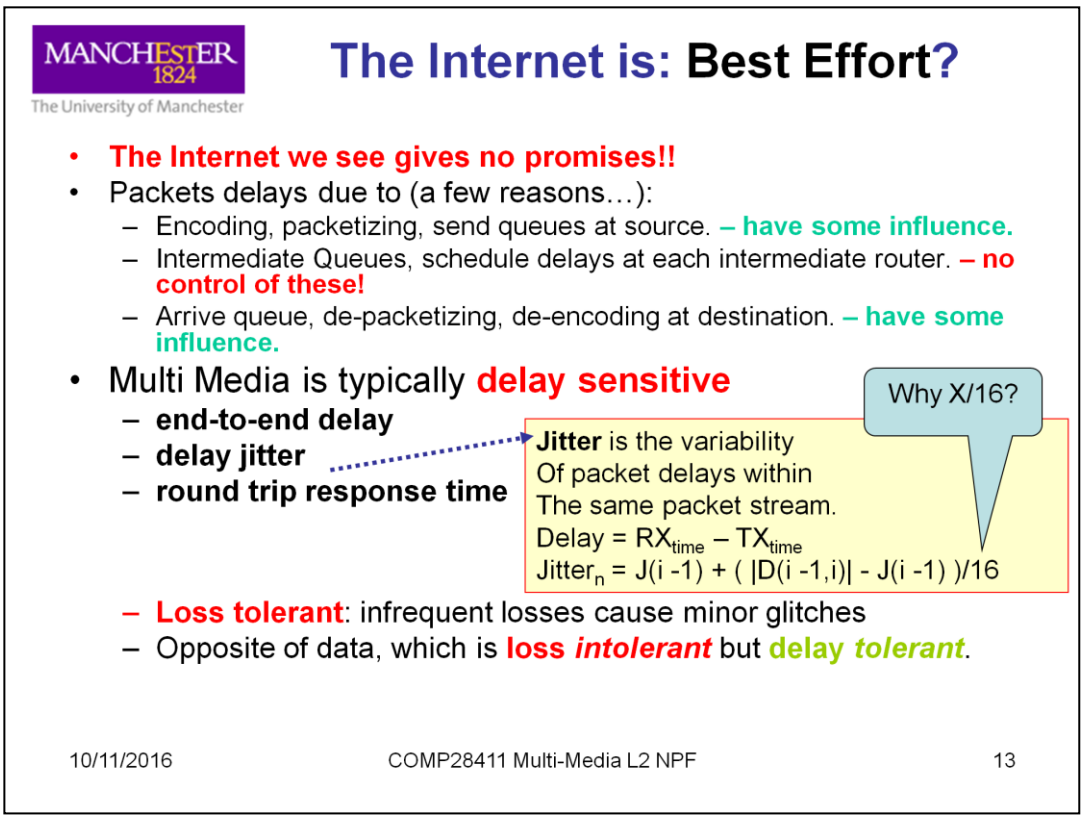
If the server dies or the path disappears then one of the alternatives can be returned instead. Switching paths from one to another is transparent for this simple DNS query which is stateless, a simple query plus response. TCP connections to anycast servers can also be used. But now there is state in the connection which may last for any length time. Because the Internet to mainly reliable, having state in the connection to a particular serving anycast server is not often an issue! However, if the server were to disappear then the application should be transferred to another server, hopefully seamlessly. This, however, is not always that easy to achieve so TCP is less often used. One example where anycast using TCP might work very well is to web servers. An HTTP query is sent, and a web page is returned. Where a session ID is included it is best if the session always goes to the same server. If the server disappears occasionally during a session, provided this is rare, it probably does not matter much?

MANCHESTER 1824
The University of Manchester

Problem ????

QUALITY OF SERVICE?

- Identify and separate media classes?
- Policing the amounts of traffic?
- Scheduling what to send next?



Divide 16 – Cancels noise, reduces randomness.

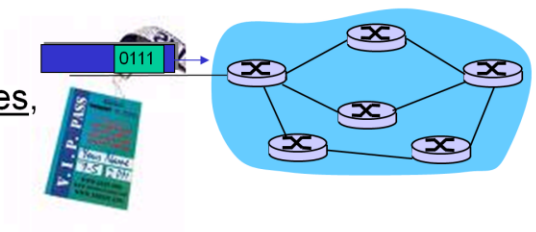
Every process that the data to be communicated has to undergo as part of its communication takes some time. Processing will happen on the source node to get the data ready. On each intermediate router to queue the arriving data then decide which output to send it to after traffic shaping plus to output the data. Finally at the destination to pass the data into the application that will use it and then to actually use it. Each link between machines is also subject to the transit time for each bit from being written to the link until it reaches the other end of the link, this must be summed with the delay between the first bit arriving at the destination (or being sent from the source) until the final bit arrives and the block of data representing the packet can be processed by the destination.

Unlike, FTP, email and similar traffic which me may want instantly, multimedia data is often time sensitive audio or video. This often means the delay constraints are real-time. We want to view the data as soon as possible. When there is a two way conversation, we prehistoric creatures still think we are adjacent to one another whereas we might be a long way apart. Because we use, silence and facial expression to determine whose turn it is to talk and when to swap speakers, prolonged delays can cause conversation to break down.

For, multimedia, the data stream normally has more information than we can process in the time available. This means that short breaks due to late or lost data can frequently be tolerated. Whereas a single binary bit missing from your financial data could easily halve its value!

Providing Multiple Classes of Service - QoS

- Thus far: making the best of **best effort** service
  - **One-size fits all** service model
- Alternative: **multiple classes of service**
  - Partition traffic into classes
  - Network treats different classes of traffic differently (**analogy**: VIP service vs regular service)
- Granularity:
  - Differential service among multiple classes, not among individual connections



10/11/2016 COMP28411 Multi-Media L2 NPF 14

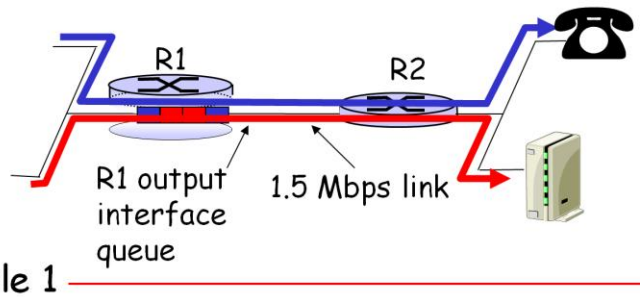
The Internet was designed and built as a one size fits all system in which all users and all packets were treated equally badly. The service was model was defined to be unreliable knowing that reliable services can be made from error resilient unreliable services. Hence, if you wanted you could do all your Internet transport using UDP (send and forget) but add to it all the reliability, buffering, retransmissions etc. of TCP by writing code that replicates how TCP behaves. In fact, this is almost exactly what the implementation of TCP does!

When Internet usage started to expand beyond a few laboratories and the military the obvious connection suppliers were the telephone networks. But telephone networks were all designed for circuit switched connections rather than packet switched. Convergence has taken a very long time but now we have almost reached the stage where ALL data traffic can flow over the same (now largely virtual) circuit providing wires.

But, telephony requires a good Quality of Experience (QoE). Therefore, resources were allocated for conversations and initially separately for all other data traffic. Faster processors and greater memory has allowed more sophisticated adaptive sharing which is discussed in the following slides.

Scenario: Mixed FTP and Audio

- Example: 1Mbps IP phone + FTP share 1.5 Mbps link.
  - Bursts of FTP can congest router, cause audio loss.
  - Want to give priority to audio over FTP.



Principle 1

Packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

10/11/2016 COMP28411 Multi-Media L2 NPF 15

Imagine we have a 1.5Mbps link. This can send no more than 1,500,000 bits in 1 second. The link is shared between telephony requiring 1Mbps and an unknown amount of FTP traffic. Telephony is usually close to being constant bit rate. Other traffic is more complex to model.

We know traffic is diurnal, it triples (or more) between 06:00 and 12:00 noon. The pattern repeats almost the same daily but decreases by about 25% at weekends. This is demonstrated in the images below taken from an1997 study using |MCI's data links in the USA. http://tinyurl.com/z7esro5.

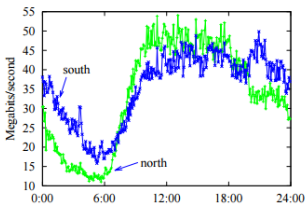


Figure 3a: Byte volume for 1 day on domestic link.

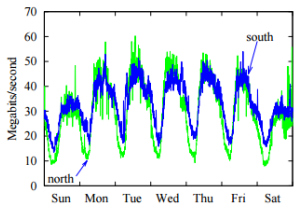


Figure 3b: Byte volume for 7 days on domestic link.

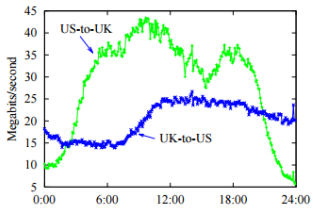


Figure 5a: Byte volume for 1 day on international link.

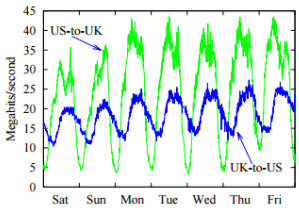
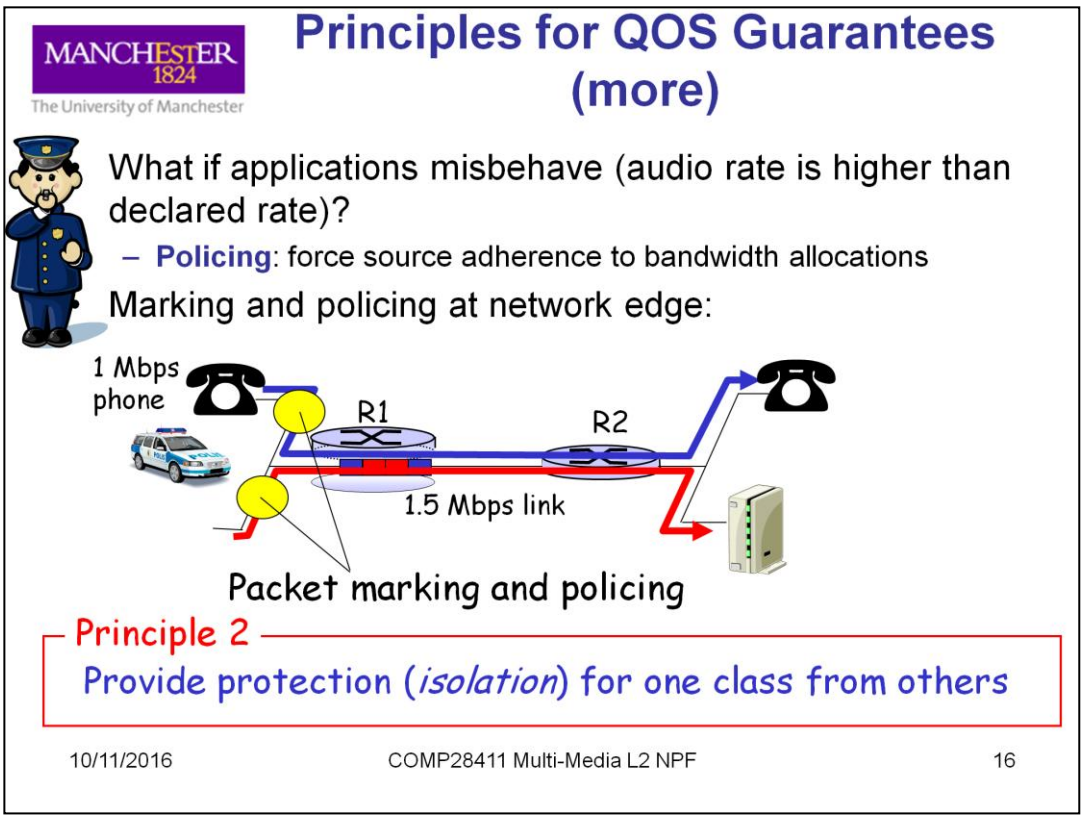


Figure 5b: Byte volume for 7 days on international link.

But be aware, proportions of traffic of each type will have changed since 1997! Most non telephony traffic follows a Long Tailed Stochastic Self Similar distribution which is bursty by nature. There are many models but the Poisson Pareto Burst Process (PPBP) seems to be the most used. This means that when provisioning the Internet, routers, servers etc. we must take account of peak demands in order to maintain sufficient services. See: http://tinyurl.com/jllogz4

If our FTP service has a peak demand less than 0.5Mbps the design above will be OK. If its peak is above 0.5Mbps then sharing in a best effort world will fail resulting in audio loss. Therefore, it is necessary to either reserve capacity for enough audio or limit the other traffic somehow. The first step is to distinguish at the router between the different classes.



Applications and users, even big companies hiring resources, cannot be trusted to only use the resources they purchase. Hence quotas and methods to restrict users to them! The same applies to the Internet.

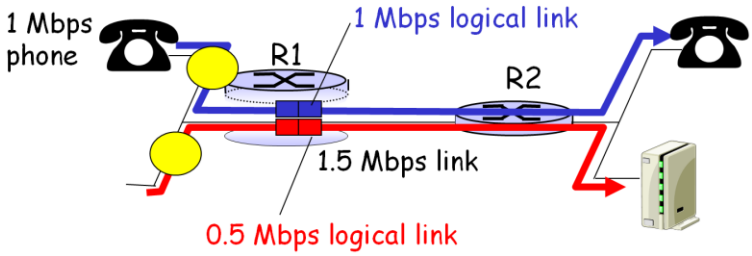
In our example, the audio application was stated to use 1Mbps. What if it misbehaves and uses more. There will be less resources for other services, the FTP file moving service in our example. Also, what happens if the FTP demand exceeds its stated capacity? Some method to force users to stay within their allocated resource limits is required, we call this “policing”.

Firstly by marking the different classes of traffic with labels (remember Multi Protocol Label Switching (MPLS) is a popular routing method that you should know a little about), the labels can then be used to allocate and police the use of network resources by the network traffic based on assigned labels representing the class of traffic.

Secondly, the labels can be used to block excess traffic somehow? But how do we do this in a fair way. Fairness is difficult to achieve as almost any ordering or priority scheme will appear fair in many circumstances except for some special cases.

Principles for QOS Guarantees (more)

- Allocating *fixed* (non-sharable) bandwidth to a flow can be *inefficient* in use of bandwidth if flows do not use their allocation! e.g. At night.



Principle 3

While providing isolation, it is desirable to use resources as efficiently as possible

10/11/2016 COMP28411 Multi-Media L2 NPF 17

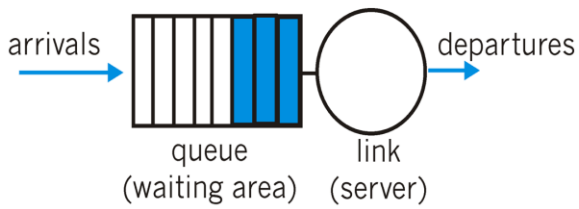
One idea which seems fair is to allocate exactly the resources purchased to each user. Provided the total allocated is <= to the capacity of the network then it should all work as wanted! But, this means that all users must purchase capacity based on their maximum ever usage of resources. Most of the time, some of the resources will be unused. Its fair but its very inefficient.

If there are under utilized resources, then why not allow other users to temporarily use these unused resources. Then, users will purchase sensible amounts of resource but do not need to allow for the almost never going to happen events. Is it fair, it can be. When I need to exceed my allocated resources for a time, provided you are not using all of yours I can use the unused portion. And when the same happens to you, I'll let you use my unused fraction. But, if my allocation is much bigger/smaller than yours then there is still a possibility for a massive imbalance. But, allowing use of unused resources is much more efficient so its worth trying?

To make things fairer, instead of making you purchases resources based on your maximum ever requirements which probably seldom happen you can purchase based on your average usage. Bigger user pay more than smaller users just as before. Also, we'll impose a maximum limit on how much of any spare resources you can use and for how long. Now, for instance, a small user can purchase a large peak usage for occasional situations paying appropriately compared to larger users. Its all a bit complex but it should work quite well if the mechanisms are simple enough.

Policing And Scheduling Mechanisms

- Policing: Action when resources (**total** OR **allocated**?) exceeded.
- Scheduling: Choose next packet to send on link
- FIFO (first in first out) scheduling: Send in order of arrival to queue
  - Discard policy: If packet arrives to full queue: who to discard?
    - Tail drop: Drop arriving packet ← **Default on Internet**
    - Priority: Drop/remove on priority basis
    - Random: Drop/remove randomly
    - Front drop: Good option for real-time media?



What if: Intermittent audio mixed with many large file transfer bursts?
Is it fair?
Jumbo packets up to 9K bytes not a problem but above 15K could delay Gigabit Ethernet for longer than 1500 bytes on100Mbps Ethernet

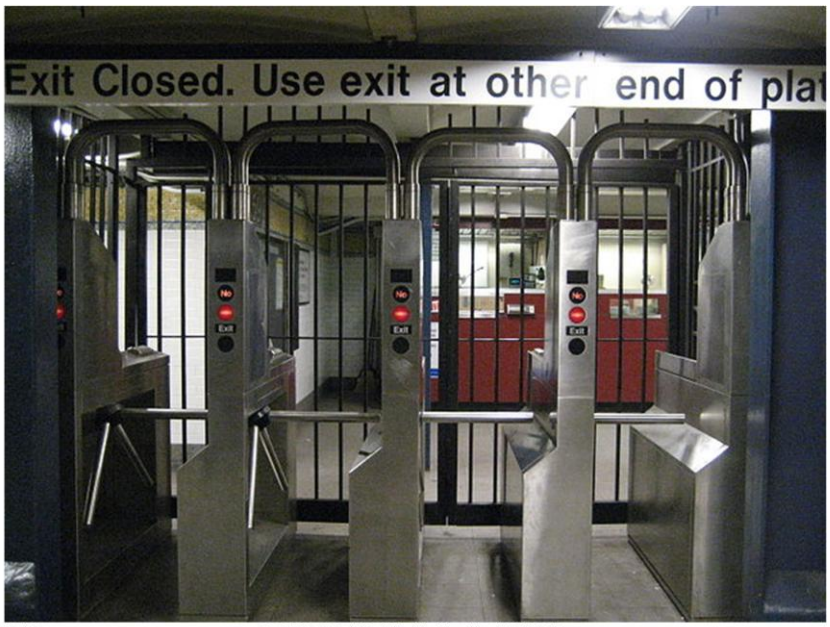
10/11/2016 COMP28411 Multi-Media L2 NPF 18

So, now we start to look at the implementation.

We need to support many different users who will appear to use the whole network according to the resources they have been allocated. This means sharing but in an appropriate and fair way. A simple mechanism to support multiple input streams being re-directed to multiple output streams needs to be implemented in all our switches and routers.

At an Internet device, traffic will appear to arrive randomly distributed. Fairness suggests that the first to arrive should be the first to leave. If the outgoing stream resources does not exceed the demand from incoming traffic (normally I'd say this the other way around!) then incoming traffic will have to sometimes wait to be processed and sent to the appropriate output stream. A sensible data structure to wait inside is a queue or LIFO data structure. The LIFO order is a simple scheduling mechanism. The queue data structure can of course become full (run out of memory) at which point a decision has to be made about what to do. We cannot sensibly expect the queue to just give up and go into a sulk (crash). So we must decide which packets in the queue or the one causing the overflow we will throw away. Something has to be disposed of! As listed above there are many alternative policies for choosing what to do. Depending on the traffic pattern, different policies will impinge on fairness in different ways. None of us users want our data thrown away, but somebodies data has to have this done to it!

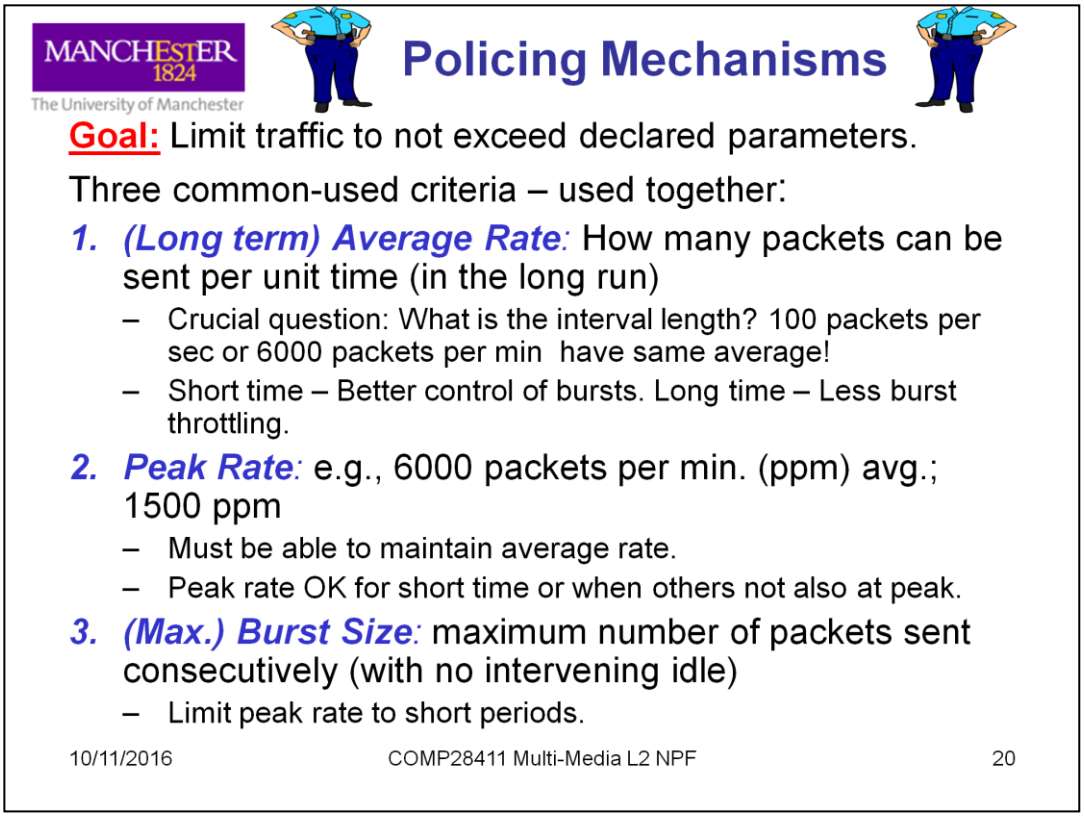
Traffic Flow Control - Policing



The picture above is from the New York subway. Note that people must queue at each entry point to use the turnstile. They can only proceed when the light allows. The turnstile, takes time to rotate. Lock and then release for the next person. This has the effect of breaking up the arriving stream of people into a slower stream. It acts like a dam does to water.

By adjusting the number of usable turnstiles (traffic streams) and the rate at which people can use them, the amount of people flowing on the far side can be controlled.

You might imagine that some people could pay to use a faster queue and turnstile in order to get some advantage over slower moving people. This metaphor is carried across into how the Internet can manage Quality of Service for packet based traffic streams.

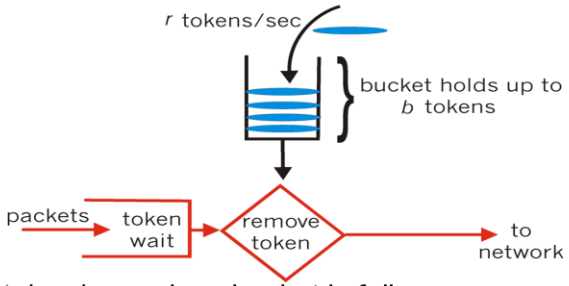


The goal of Quality of Service is to control the rate of flow for a given traffic stream/class/flow. A given router and each of its outgoing links has a maximum amount of traffic it can process in a given time. Incoming streams of traffic are often bursty in nature. There is an underlying rate which is often fairly constant for prolonged periods of time but which can drop and rise from time to time. We describe such a flow based on its average traffic rate. To this we add a constraint representing an absolute maximum amount that should be handle able when there is a peak burst of traffic. Burst are constrained by how much traffic per unit time is processed and by a maximum size (length of time?).

Together, the three criteria are enough to be able with one or two further constraints to guarantee that provided the peak rate and maximum burst size are not exceeded then the processor (a router) will not need to drop any traffic. The main constraint is that we must not exceed the total maximum capacity of the router in terms of traffic arrival rate and waiting queue sizes must always be less than or equal to the rate at which the router can forward the traffic. If this constraint is met then everything should work well.

Policing Mechanisms 2

Token Bucket: Limit input to specified Burst Size and Average Rate.



Bucket can hold b tokens.

- Tokens generated at rate *r token/sec* unless bucket is full.
  - Either put in low priority (e.g. best-effort) queue or dropped.
- **Over an interval of length** *t:*
  - Number of packets admitted *less than or equal to (r t + b).*
- If arrival rate traffic <= token rate (r) then
  - agreed bandwidth and delay/jitter constraints can be met.
  - Stored extra tokens used to clear bursts.

The token bucket is used to control the rate of forwarding traffic for output from the input/arriving traffic queue for a specific class of traffic. Tokens are added to the token bucket at the average rate the traffic should be processed. If the bucket is full then the new tokens must be thrown away. If the bucket is empty, no traffic can be forwarded through the system.

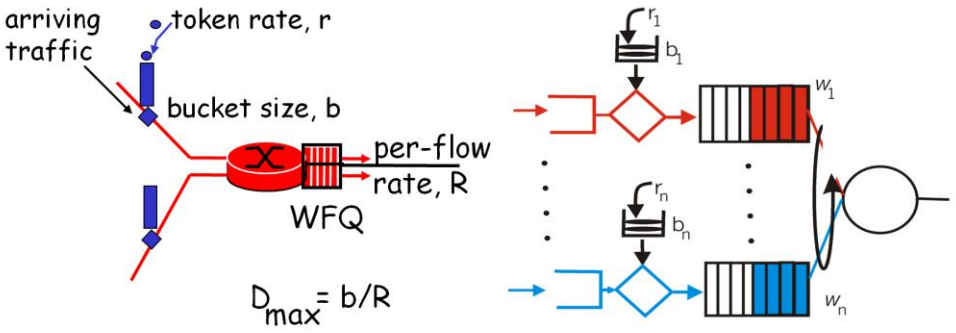
Two types of token bucket are often discussed. The pure token bucket as in this slide which stores some unused tokens which can be used, if available, to handle peak rate bursts of traffic and so called “leaky” buckets where unused tokens drain away so that a sudden burst of traffic after a quiet period will be forced to wait and is straight away policed down to the permitted rate.

With a token bucket we can imagine that when there is lower than the token rate traffic arrivals the bucket fills, when the rate exceeds the token rate it empties.

MANCHESTER 1824
The University of Manchester

Policing Mechanisms 3

- Token bucket and WFQ combine to provide guaranteed upper bound on delay, i.e., *QoS guarantee*!
  - **Provided peak rate not exceeded for too long** – buffer overflow. And
  - **Total allocation does not exceed resources!**



10/11/2016 COMP28411 Multi-Media L2 NPF 22

On the arrival side of a router the arriving traffic can be split into different classes. Each class is assigned a token bucket stream to flow through. Each token bucket has its own token rate and size. Thus, the different classes are allowed to forward different amounts of traffic to the next stage. The next stage is to merge the outputs from the queues ready for forwarding from the router to their next destination.

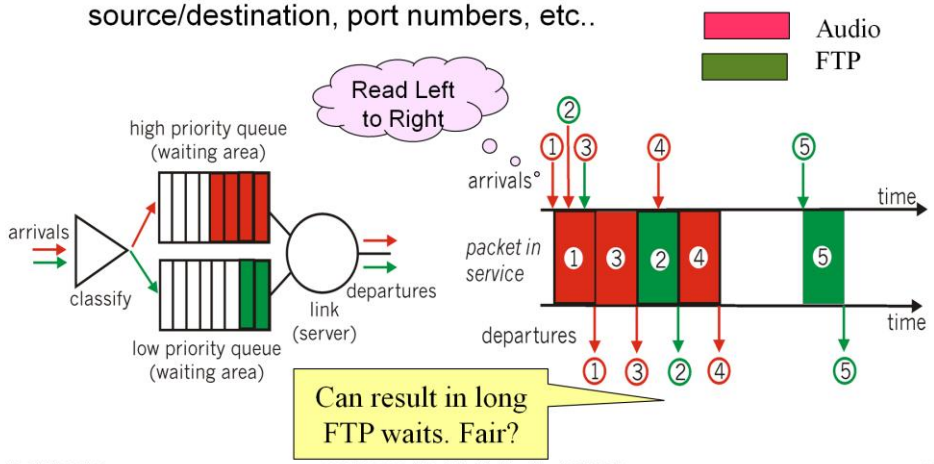
Various ways to select the next items for forwarding from the set of per traffic class queues can be imagined as discussed in the section of traffic scheduling. Commonly in today's Internet the streams are merged using a Weighted Fair Queue (WFQ).

MANCHESTER 1824
The University of Manchester

Scheduling Policies: more 2

Priority scheduling: Transmit highest priority queued packet

- Multiple *classes*, with different priorities
  - Class may depend on marking or other header info, e.g. IP source/destination, port numbers, etc..



10/11/2016 COMP28411 Multi-Media L2 NPF 23

Policies for scheduling that ignore priority are never going to be fair. For example, a large IP packet carrying file like data might take quite a long time to send. If there are. Normally small, real-time audio packets in the queue then delaying them while the bigger file data is sent may mean they arrive to late to be used. By giving the audio data a higher priority, we can ensure that it never gets stuck behind larger (hence slower resource occupying) large file data.

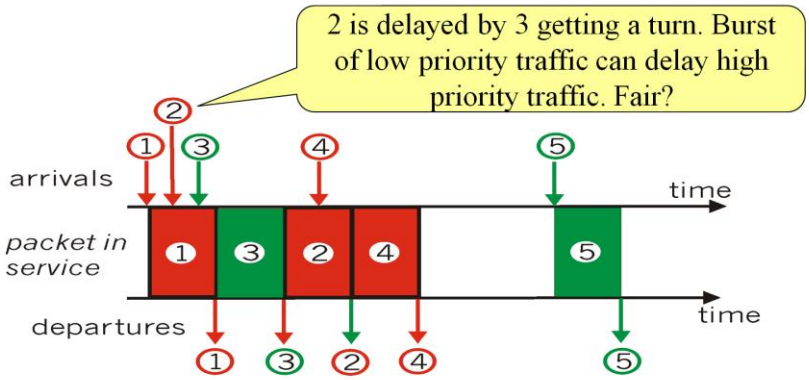
Problem solved!

Well no. A continuous stream of high priority audio data will result in the file data being delayed for long periods. Its not real-time constrained but someone is probably waiting for it. Priority schemes on their own are seldom fair but some form of priority ordering is needed.

Scheduling Policies: still more 3

Round Robin Scheduling:

- Multiple classes
- Cyclically scan class queues, serving one from each class (if available)



10/11/2016 COMP28411 Multi-Media L2 NPF 24

The opposite of priority ordering is to apply a simple round robin scheme. In this scheme every class of data is given a turn cyclically provided there is some data to send during the turn.

This ignored priority so suffers from the problem of, for example, a small real-time audio packet having to wait whilst a large lower priority packet is sent. But, on the other hand, it ensures that every different class of traffic gets a turn without a very long wait .The length of wait being determined by the number of classes and the size of each data packet of each class currently in the queue ahead of you.

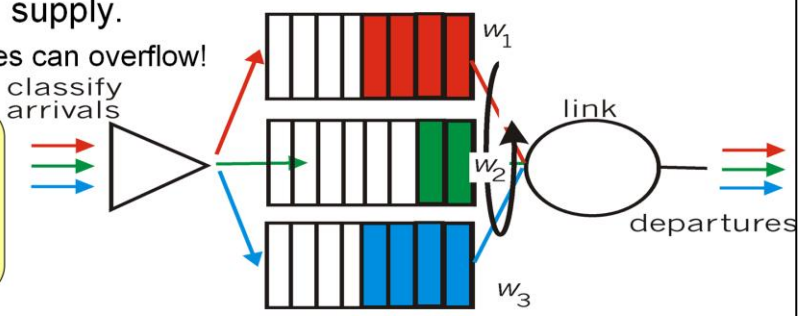
MANCHESTER 1824
The University of Manchester

Scheduling Policies: still more 4

Weighted Fair Queuing (WFQ):

- Generalized Round Robin.
- Each class gets weighted amount (fraction) of service in each cycle.
- At queue exit ensures demand does not exceed supply.
  - But queues can overflow!

Each gets fair share. What happens to unused resource? Bursts when demand is low/high?



10/11/2016 COMP28411 Multi-Media L2 NPF 25

The solution to maintaining a fair scheduling of who to process next is to combine the best features of both priority based and round robin scheduling into the method called Weighted Fair Queueing (WFQ).

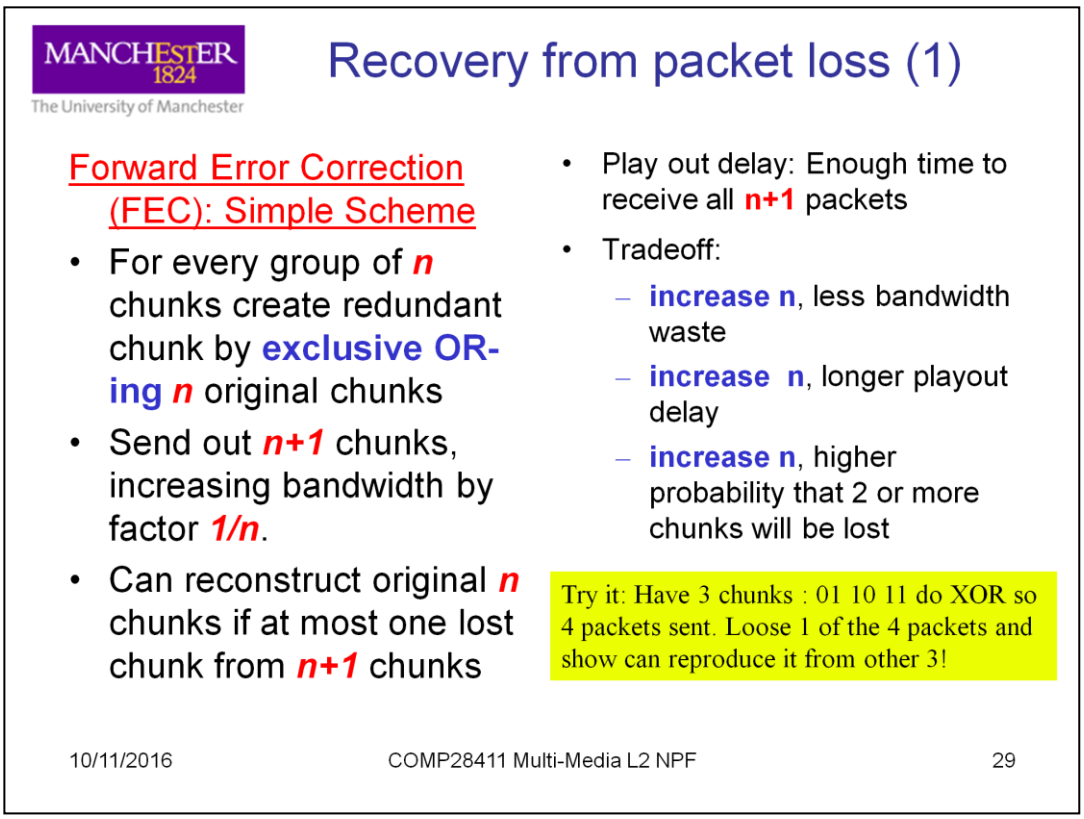
On arrival, packets are identified by class and sent to separate queues, one per class. Packets with the same class can fairly be processed in LIFO order in most situations. One exception is for real-time data when it is known that the packet cannot arrive on time to be played when it should be. In this case, simply throwing the packet away makes sense as it reduces congestion on the route ahead.

MANCHESTER 1824
The University of Manchester

Problem ????

ERROR RECOVERY?

- Acknowledgments
- Forward Error Correction
- Interleaving



On Internet many losses are groups of packets not individual.

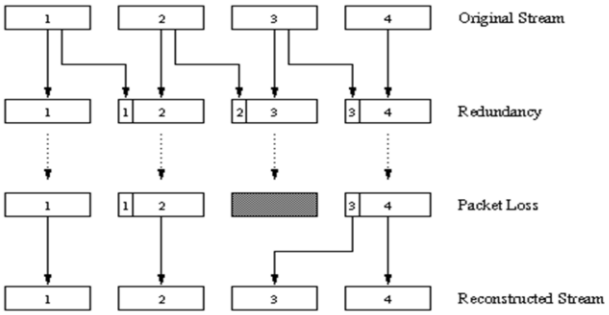
Applies also to point-2-point connections.

Resending data adds big delays so we try to avoid doing this.

Almost all existing Internet systems use FEC. FEC adds extra redundant data which in effect lowers the data rate in return for better ability to detect and correct errors. The most common system on the Internet uses a process called “convolution” which we show here with a vastly simplified example. When convolved data arrives at a receiver the bit sequence is hard to read and interpret. This is especially true if the bit sequence has changed during the transfer from the source to the destination. For convolution a decoding algorithm is used to undo the convolution. This algorithm is very well known but also quite complex. Its called “Viterbi” after its inventor. You will probably study it elsewhere but what it does is read the incoming data bit by bit and builds a lattice of possible interpretations of the incoming bit sequence into possible sequences of the original data bits. Hard decoders return only the best solution found in the incoming data. Soft decoders return a probability that each returned bit is correct. The soft Viterbi probabilities can be very useful when the CRC applied to the decided block of bits is applied and fails indicating one or more errors. The lowest probability bits might be swapped from 0 to 1 or 1 to 0 and retested via the CRC tester. This might correct the sequence in which case the data can be used rather than having to be retransmitted.

Recovery From Packet Loss (2)

- 2nd FEC scheme
  - "piggyback lower quality stream"
  - Send lower resolutic audio stream as redundant informati
  - e.g. nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps.



- Whenever there is non-consecutive loss, receiver can conceal the loss.
- Can also append (n-1)st and (n-2)nd low-bit rate chunk

10/11/2016 COMP28411 Multi-Media L2 NPF 31

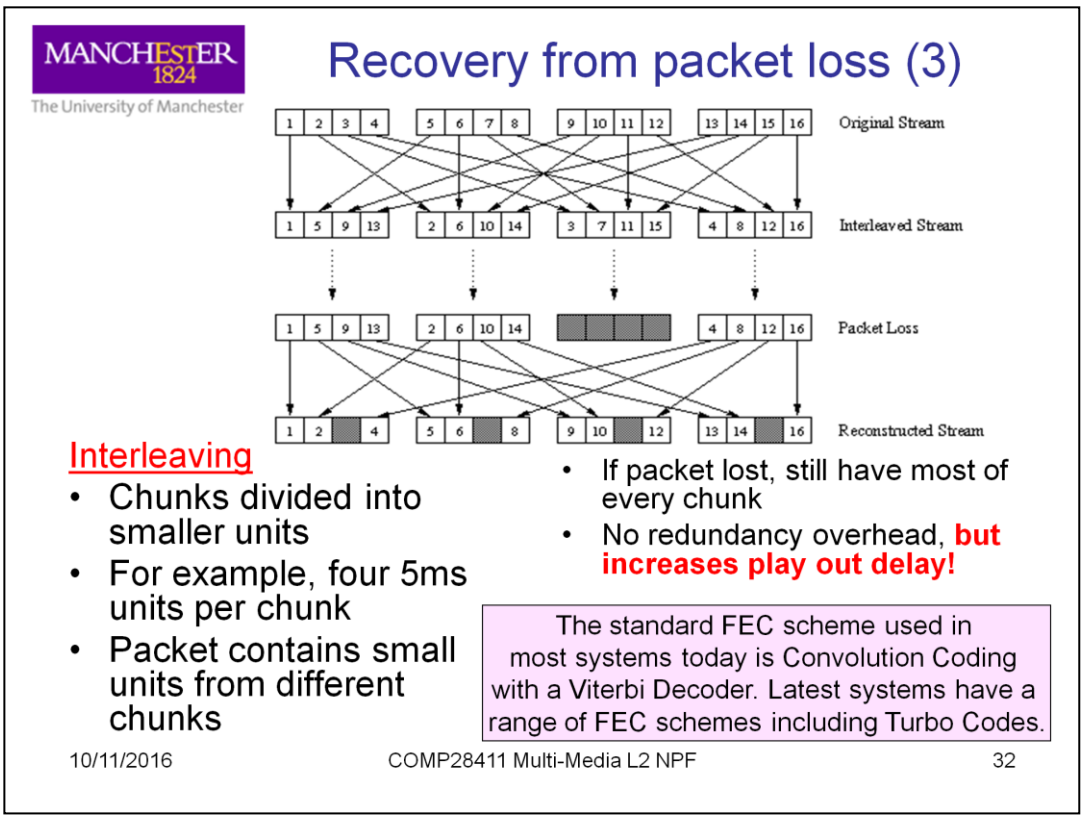
This 2nd scheme generates a low quality representation of the multi-media data which is interleaved alongside the high quality stream but offset in time by a small amount. In my example, I've sent the low resolution stream later so that if the high resolution packet is lost or late then at least the lower resolution data can be rendered instead. Some systems will do the opposite and send the low resolution data earlier than the high resolution stuff as it may be quicker to generate the low resolution version . The size ratios between high and low resolution and their content can be varied to suit the application and the noisiness of the communications channel.

Similar scheme used in some digital video broadcasting – B/W sent low quality stream.

Colour in High Quality. Weak signal – loose colour but not everything.

Actually using different modulations.

Send SD image and HD image.



This scheme relies entirely on interleaving. No extra or redundant data is generated. Payment for improved error correction is by extra delay or latency. Instead of being able to play the whole of item 1 when packet 1 arrives the system must wait until the 4th packet arrives before playing any of the items. This is clearly far from a perfect scheme but it is simple and demonstrates the principle of interleaving quite well.

Mobile phone systems and especially 2G GSM make very heavy use of interleaving alongside redundancy to ensure the core data necessary to the systems correct operation is very well protected against wireless channel errors. This is of course done at the expense of the data rate. For wireless systems subject to the inverse square law for local connections and the even worse inverse 4th power law for longer connections tend to only use high data rates for nearby connections. As the length of the connection grows the data rate must drop because the signal strength drops rapidly and more and more FEC data is needed in order to fix changes in the sent data that happen in the wireless channel. High data rates need high signal power to utilize complex modulation schemes which hare harder to decode but pack more bits into each symbol and so as to reduce the amount of FEC redundancy that is needed.

MANCHESTER 1824
The University of Manchester

Summary

- We've covered quite a bit of ground today! **Hopefully…**
- Types of transport connection: unicast, broadcast, …
- QoS and how its implemented in routers.
- I may have skipped due to lack of time :-
  - Error Detection and Correction
    - But you have the notes ….. There is a workshop on this!
- **Next time:**
  - Setting up Voice over IP (VoIP) telephone calls and other media exchanges.
  - RTP + RTCP
  - Making TCP usable for media streaming and sometimes for 2 way conversations.