1 2001 Paper 6 Question 3

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2001p6q3.pdf

(a) Define the terms *circuit* and *packet* in the context of communications systems.

Packet networks are much cheaper to install – you can send packets other ways

A packet is a small ($\sim$ 1kb) collection of bytes which are transmitted together. Packets consist of headers and payload. There will be many headers appended onto the packet – one for each layer of the stack.

A circuit is an dedicated link from one node to another on a network. This link is only used by the sender. If the sender does not send any data then the link remains idle.

(b) What sort of guarantee does circuit switching provide?

In circuit switching; one fault on a node fails all users using it – one node failing affects 1m users.

Circuit switching guarantees performance – since the circuit is dedicated to a given conversation, no other nodes will attempt to communicate over it. So if a circuit capable of sending n bytes per second is setup, then there is guaranteed to be sufficient bandwidth to send n bytes per second even if other nodes are trying to send across the network. Furthermore, since the circuit is dedicated to a given conversation, there will be no packet collisions. Expensive to install!

Circuits aren't resilient to failure – you need hardware to give this property! V. Expensive!

(c) What advantages does packet switching provide over circuit switching?

Multicast support!

Unlimited Number of simultaneous Users

Same or similar levels of resilience for far less cost.

Packet switching allows better bandwidth utilisation via statistical multiplexing. Often, communication is bursty. Using circuit switching, nodes must request their maximum possible usage (which is often significantly above their average usage), while under packet switching the network makes more realistic assumptions allowing for more than $\frac{\text{bandwidth}}{\text{max node usage}}$ nodes to use the network.

Packet switching also better resilience to failure – packets can take multiple routes and easily be re-routed in the case of failure. However, if a circuit fails then the sender and receiver must notice and establish a new link before transmitting any more data.

Packet switching has no setup cost – circuit switching requires the sender and receiver to setup a circuit, which takes time. Packet switching does not suffer from this initial overhead. However, note there is higher continued overhead with routing packets.

(d) Which of *frequency division multiplexing*, *time division multiplexing* and *code division multiplexing* lend themselves to circuit switching? Which to packet switching? Explain why or why not in each case.

- **Frequency Division Multiplexing**

  In FDM, signals are sent at different frequencies. Each conversation has a frequency band on which messages are transmitted. If a conversation does not transmit, then the frequency band (capacity) is wasted.

  FDM lends itself to circuit switching – it allows multiple conversations to take place simultaneously without impeding each other. This is highly beneficial. It complies perfectly to the circuit switching paradigm.

- **Time Division Multiplexing**

  The bandwidth is split into timeslots. Each conversation is allocated a specific timeslot which it can use. If a timeslot is unused, then that capacity is wasted. Timeslots are allocated during circuit setup and deallocated during circuit teardown.

  TDM lends itself to circuit switching since it allows multiple simultaneous conversations without overhead.

- **Code Division Multiplexing**

  In Code Division Multiplexing, each sender/receiver pair is allocated a unique code n-bit code. In this application, we view 0s as -1. The sender then XORs their code with the bit they wish to transmit and adds this onto the signal it is

The difference between ATM and packet switching is:

- ATM uses FIXED SIZE -- STDM without reserving slots
  In ATM you establish the connection first then use this route

- Packets do not have a fixed size
  Packet routing calculations are done at each router/switch

ATM doesn't have a guarantee of constant bandwidth:
- So this isn't Circuit Switching -- "cells" can arrive in any order at all

"A" in ATM is because there is fixed contention for fixed-size slots

ATM defines what to do on layer 2 and 3.

Layer 2 is Local Link
Layer 3 is Global Link

Optical Wires can use 6 frequencies

FDM and TDM are designed to divide a guaranteed service between n senders
- if you use a circuit, you'll get a circuit at the end
- if you use packets, you'll get packets

You can use any combination of TDM, FDM, CDM that you like at the same time.

Use FDM for legal reasons -- allows other people to talk!

FDM with packets to send multiple packets at the same time -- has the same throughput as sending parts of the packets at the same time but allows multiple packets to transmit at once.

Range of frequencies on which the signal is being sent in Hertz -- also known as bandwidth (N.B. this is an overloaded use of the word) IE I'm sending across a 50Hz range

Power with which the signal is being sent in Watts

$$C = B \lg\left(1 + \frac{S}{N}\right)$$

"Information Capacity": What we think of as Bandwidth

Power of noise in Watts ie CBMR + noise from next door + ...

VLANs add one 16-bit integer to the start of an ethernet frame -- each integer corresponds to a different network.
IE The switch forwards the packets without the VLAN header.
When the switch receives a packet, it adds the VLAN header
This is therefore transparent to the user!

Baud rate is the rate at which symbols are sent -- the number of "different states of the medium you can detect".
A symbol can be 2 bits or 8 bits or 64 bits.
If you're able to sense the medium at 10x per second, the baud rate is 10Hz

The bit rate is the "number of bits which the receiver sees per second"

Asynchronous:
Messages are split into frames.
Synchronise clocks on a long sequence sent at the start of each frame
ie ethernet sends 64 alternating 1s and 0s before each frame -- you can work out drift rate and account for this
work out all constants in myclock = yourclock + c + d * time + e * time^2 to high accuracy

Synchronous:
Have lots of transitions and synchronise as you go
sync myclock = yourclock + c lots of times

In circuit switching; the network is in control -- users ask the network for a signal but the network can decline.
You can be very sure about the service you're giving people.
The circuit network knows when its overloading or when you request something above its capabilities and can say "no"
Once a circuit network gives you a service, you are guaranteed to get this service.
The network won't listen to you if you send something outside your timeslot/frequency etc

Circuit Switching guarantees latency, bandwidth, ZERO jitter and a CONSTANT ERROR MODEL

Constant error model: all packets take the same path -- so the type of errors data will experience is the same
The network can "train" to the line and find out which frequencies work well and minimise errors.

A circuit is a pre-established path which has the property of constant bandwidth; constant delay
We use "Virtual Circuits"; "Physical Circuits" are phone networks.

The packet network is "dumb":
It doesn't know if its overloaded or underloaded
It does the best with whatever the end nodes give it

Each packet is "self-addressed":
The end-point does not need to know what's coming next -- all the information is in the header

When a link fails in packet switching; we rebuild a new Spanning Tree.
The link isn't more resilient -- the errors just happen further down the stack so the user doesn't see it!

FDM:
Both Receivers and Senders must be able to operate on many frequencies.
Until they're assigned a frequency; the receiver / transmitter doesn't know which frequency they will send on so must be capable of sending on many frequencyes

Frequency Hopping Spread Spectrum (FHSS):
Sender and receiver have a shared seed.
They use this seed to generate random numbers and transmit a small number of bits at this frequency
Implement FEC over the top of this
You have distributed FDM without coordination!
Data can be recovered by FEC
Probability of generating the exact same is very low

You can use Frequency Modulation even when you use Frequency Division Multiplexing:
By moving frequency slightly around a central value.

You can use any combination of Frequency Division Multiplexing and Amplitude Multiplexing at the same time.

Compression should be above ARQ/FEC to ensure it doesn't compress and mess up with safety guarantees in the lower levels

Principle of layering:
Every layer does something fast and simple to get a high bitrate network and together they do something complicated.

forwarding. The receiver takes the dot-product between its code and the signal, then divides by n to get the original bit. This allows multiple senders to transmit over the same wire.

These codes can be pseudorandom, but are more commonly Walsh codes. Walsh codes are the columns of the set of matrices generated inductively by $H_1 = (0)$, $H_{2n} \begin{pmatrix} H_n & H_n \\ H_n & \overline{H_n} \end{pmatrix}$.

The (+) in the diagram in the lectures is SUPERPOSITION.

Does CDMA waste bandwidth? YES! CDMA wastes "at least 95%" of the bandwidth

It's used only when you have multiple transmitters to send independently without coordination!

Given n-bit codes and n-senders, we need $n+1$ states. To represent $n+1$ possible states, we need $\lceil \lg(n+1) \rceil$ bits per state. So to transmit n bits, we have to send $n\lceil \lg(n+1) \rceil$ symbols. If this is correct, CDMA would waste $100 - \frac{100}{\lceil \lg(n+1) \rceil}$% of the bandwidth – in real applications this translates to around 90% of the bandwidth. I'd think this would be a massive drawback, however it's not mentioned anywhere. The only brief mentions I've found are ambiguous as to whether they mean "symbols per receiver" is high or "symbols per bit in transit" is high.

CPS satellites use this – you send 1023 chips for every 1 real bit – wasting 99.9% of the bandwidth.

CDMA is also good for error correction! If some bits flip, the probability of most of them still being correct is high!

Chipping codes are CYCLICALLY ORTHOGONAL – this means they are orthogonal starting anywhere – so you don't have to sync!

You can sync by trying to compare each bit until you get a very high match.

CDM is bad for circuit switching – for the reason explained above, it's highly inefficient and wastes vast quantities of bandwidth. It is always beneficial to use any other form of multiplexing, rather than one which burns 90% of bandwidth.

None of these multiplexing methods are useful for packet switching – the premise of packet switching is to allow more communication by exploiting statistical multiplexing and the bursty nature of data transmission without ever setting up a circuit. Therefore, there is no need to allow multiple circuits to multiplex over the same wire – the concept of a circuit does not exist in packet switching.

2 2001 Paper 5 Question 3

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2001p5q3.pdf

Information is to be conveyed from A to B using automatic repeat request (ARQ), forward error correction (FEC), and lossless compression.

(a) Explain the terms ARQ, FEC and *lossless compression*.

- **Forward Error Correction**

  Forward Error Correction is a type of error correction wherein an Error Correcting Code (ECC) as appended to the end of each packet. The ECC is "redundant" data and a function (often a simple hash) of the rest of the data. On receipt, the receiver will apply the function used to calculate the ECC and compare the value to the ECC. If they are different, there has been an error. In FEC, the receiver then tries to establish the most likely error that occurred. This is often nontrivial since bit flips are unlikely to be independent. If the recipient is unable to determine the error with high certainty, it will send a NACK to the sender, requesting retransmission of the corrupted packet.

- **Automatic Repeat Request**

  Automatic Repeat Request is the strategy of using acknowledgements and time-outs to implement reliable delivery. Sliding Window ARQ is one of the most common implementations.

  In the sliding window protocol:

  – The sender maintains three integers: sws (send window size – a constant), lar (last acknowledgement received) and lfs (last frame sent). The invariant is maintained that $lfs - lar \leq sws$. The sender will send frames i for $lar <$

$i \leq lar + sws$. If a frame times out then it will retransmit it. On receiving an acknowledgement for frame i, it sets $lar = i$ – all frames with id $< i$ have been seen by the receiver.

– The receiver maintains three integers: laf (largest acceptable frame), lfr (last frame received) and rws (receive window size). All frames $\leq$ last frame received have been received and acknowledged. The receiver buffers any frames i such that $lfr < i \leq lfr + rws$. On receiving all frames $\leq lfr + z$ for $1 \leq z \leq rws$, the receiver sends an acknowledgement. If the receiver receives frames smaller than its last acknowledged frame, it should retransmit acknowledgements for the last acknowledged frame.

Frames in ARQ should have a sequence number – this sequence number should be $> \lceil \lg(rws+sws) \rceil$ bits to ensure the receiver does not mix up the frames. This also aids with the order in which frames are delivered – unlike in C&Ds, packets do not need monotonically increasing integers.

Furthermore, ARQ with a well-chosen sliding window size and timeout will implement flow control. If the receiver gets overwhelmed, then it will not acknowledge frames. So the senders window will not slide, the sender will not be able to send new frames until the timeout for the original frame triggers.

- **Lossless Compression**

  Lossless compression is the strategy of changing the encoding of data into a compressed format without losing any data. In good situations, lossless compression can decrease the size of data transmitted by a factor of 10. However, lossless compression adds overhead with encoding and decoding the messages. This means that in many situations, best performance is not gained by maximal compression.

  The best type of Lossless compression to use is highly application-specific – text is well-suited to encodings such as huffman coding (which compresses symbols to the entropy limit), code is well-suited to dictionary encoding (which creates compressed special representations for common phrases), while FAXs benefit from run-length-encoding (which records the number of consecutive symbols – FAXs have a lot of whitespace).

(b) If we consider ech of these functions to be operating at different protocol layers, what would be the most sensible ordering of the layers, and why?

FEC is implemented on the physical layer. The types of errors which occur (bit errors vs burst errors) and error rates depend on the physical medium. It therefore makes the most sense to determine and implement error correction on the physical layer, where appropriate ECCs can be chosen to deal with the types of errors which occur.

ARQ is implemented on the Data-Link Layer. The Data-Link Layer transmits data between nodes over the physical layer. ARQ resolves transmission issues and manages flow control (flow control being a data-link protocol). It's rational to implement ARQ on the Data-Link Layer.

Lossless compression is highly application-specific – the right choice can yield 10x compression, while the wrong choice can increase the size of the message. It is therefore rational for lossless compression to be implemented on the application layer.

(c) Suppose:

- The underlying bit channel has a capacity of B, a delay τ and error rate ϵ_0.
- The compression rate is $C < 1$
- The FEC has rate $R < 1$ and given an error rate ϵ_0 provides an error rate ϵ_1 (which is detected).
- The ARQ protocol has a window size of W.

At what rate can information be conveyed?

I assume that the variant of ARQ we use is

The FEC rate is given by $\frac{\text{useful data per packet}}{\text{packet size}}$.

The ARQ window size is the number of packets the sender will try to send at once.

The compression rate is given by $\frac{\text{Compressed size}}{\text{Uncompressed size}}$.

Therefore, the information rate is given by:

$$\text{InfoRate} = \frac{R \cdot \text{DataRate}}{C}$$

The data rate is the number of packets received per second.

For the purposes of this question, I assume that the delay before ARQ will resend the message is 2τ – ARQ will only retransmit if the data has been corrupted and will do so at the instant it would have expected to see the data were it not corrupted.

The expected delay for a packet is given by:

$$\begin{aligned}
\text{Delay} &= 2\tau \cdot (1-\epsilon_1) + 4\tau \cdot (1-\epsilon_1)\epsilon_1 + \cdots \\
&= 2\tau \cdot (1-\epsilon_1) \cdot \sum_{i=0}^{\infty} (i+1) \cdot \epsilon^i \\
&= 2\tau \cdot (1-\epsilon_1) \cdot \left(\sum_{i=0}^{\infty} \epsilon_1^i + \sum_{i=0}^{\infty} i \cdot \epsilon^i \right) \\
&= 2\tau \cdot (1-\epsilon_1) \cdot \left(\frac{1}{1-\epsilon_1} + \frac{\epsilon_1}{(1-\epsilon_1)^2} \right) \\
&= \frac{2\tau}{1-\epsilon_1}
\end{aligned}$$

The number of packets which can be sent serially in one second is the inverse of this:

$$\frac{1-\epsilon_1}{2\tau}$$

Since the window size is W, we can send W packets at once and so the total throughput is W times this.

$$\frac{W(1-\epsilon_1)}{2\tau}$$

However, this is capped by the bandwidth of the channel. The number of messages sent per packet is given by:

$$\begin{aligned}
&1 + \epsilon_1 + \epsilon_1^2 \\
=&\frac{1}{1-\epsilon_1}
\end{aligned}$$

So the maximum number of packets which can be correctly sent along the channel is given by:

$$B \cdot (1-\epsilon_1)$$

Therefore the data rate is given by the minimum of these two quantities. Substituting this into the equation for the information rate gives:

$$\text{InfoRate} \approx \frac{R \cdot \min\left(B \cdot (1-\epsilon_1), \frac{W(1-\epsilon_1)}{2\tau}\right)}{C}$$

To work out the information rate; work out the network characteristics for each layer:

FEC	sees Bandwidth=B,	Delay=τ,	Error Rate=ε0
ARQ	sees Bandwidth=BR,	Delay=τ+δ0,	Error Rate=ε1
Compression	sees Bandwidth=BR,	Delay=τ+δ1,	Error Rate=0
User	sees Bandwidth=BR/C,	Delay=τ+δ2,	Error rate=0

--- N.B. δ is negligible

FEC trades off bandwidth for a better error rate!
For a modern FEC, the error rate will be about 10^-8.
After ARQ, the error rate is about 10^-11.
We tradeoff end-to-end delay to get a much lower error rate.

Information Rate =

```
if (8PC/BR) + 2τ <= (8WC/BR):
    BR/C
else:
    8W/((8PC/BR) + 2τ)
```