2. DATA TRANSMISSION

Data packets

- Data is broken down into packets to be transmitted.
- After transmission, data packets are reassembled to form the original message/data.

Structure of data packet

Packet header	- Contains packet number - Contains originator's/sender's IP address - Contains destination/receiver's IP address - Contains error detection method
Payload	The actual data being carried/sent in the packet.
Trailer	- Identifies end of packet// end of packet notification - Additional error checks to ensure packet not corrupted

NOTE: Corruption is when packet data is changed/ lost in some way // data is gained that originally was not in the packet

How packets are sent across the internet - packet switching

- Data is broken/split/divided into packets
- Each packet could take a different route
- A router controls the route/path a packet takes
- ... selecting the shortest/fastest available route/path
- Packets may arrive out of order
- Once the last packet has arrived, packets are reordered
- If a packet is missing/corrupted, it is requested to be resent

Benefits of splitting data into packets // benefits of packet switching

- Data packets are quite small, making it easier to control.
- Each packet can be sent along a different route to its destination; important when a particular transmission route is out of action / busy.
- Quicker transmission of data
- Only individual packets should be resent if lost/ damaged: minimises corruption & saves time & internet bandwidth
- Harder to hack: each packet contains minimal data, & travels through network separately

Drawback of splitting data into packets

Data needs to be reassembled when it reaches destination.

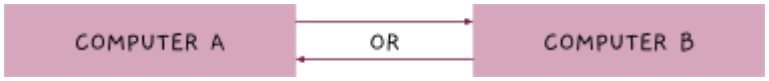
Annotate diagram to demonstrate how packet switching is used to transmit data across a network, including the use of routers, from Device A to Device B.

- Packets sent through several routers
- … taking different routes from device A to device B
- Packets arrive out of order
- Packets being reordered when all arrived at device B

Modes of data transmission

Mode	Description	Applications	Benefits	Drawbacks
Simplex	Data can be transmitted in one direction only; unidirectional	- Mic to comp - Sensor to comp - Comp to printer - Comp to speaker - Comp to monitor - Webcam to comp	cheap: only one wire is used	- Slow: data travels one bit at a time in one direction - Expensive: Requires 2 sets of wires for bidirectional transmission
Half-duplex	- Data can be transmitted in both directions - not simult-aneously; bidirectional	- Phone conversation - Walkie-talkie	cheaper than simplex for bidirectional transmission: requires fewer wires	Slow: data travels in one direction at time
Full-duplex	- Data is transmitted in both directions - simultaneou sly; bidirectional	- Broadband internet connection - Phone conversation - Video conferencing - Instant messaging - Wireless technology - Computer to modem - Social networking - Multiplayer games	Fast: data can travel in both directions simultaneously. Receiver doesn't have to wait for sender to stop before transmitting data	Expensive: wire technology to transmit in both directions is more expensive

Simplex



Half-duplex

Full-duplex

<u>Why data transmission between computer & printer needs to be half-duplex rather than simplex</u>

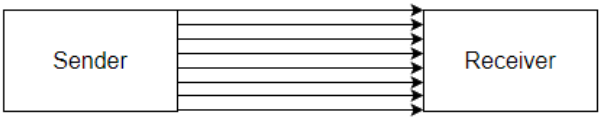
- Simplex only sends data in one direction
- … so, printer may not be able to tell computer an error has occurred, and computer may not be able to send printer the document to be printed

Mode	Description	Applications	Benefits	Drawbacks
Serial	- Data is sent one bit at a time - A single wire is used	- Connecting external hard drive to comp: USB - Transmit data over telephone line - Wi-Fi	Single wire - cheaper to buy/ manufacture/ install - less interference/ corruption/ crosstalk - more reliable over longer distances One bit at a time - less chance of data being skewed - bits synchronised after transmission/ will arrive in order - reduce rate of errors - Transmission speed is adequate	- Slower - Additional data may need to be sent - Bits need to be organised before they are sent over the channel - Expensive over long distances
Parallel	- Data is sent multiple bits a time	- Integrated circuits - CPU buses - In RAM	- Faster - Bits do not need to be organised before they are sent over	- Expensive to setup/manufacture: more wires - Limited

	- multiple wires used		the channel.	distance - More prone to interference - Bits may arrive skewed

Serial

Parallel



Why serial transmission is used over long distances

- Bits remain synchronised
- « reducing data errors
- Only single wire is require
- « more cost effective to install/manufacture

Data is transmitted to a central computer 30 kilometres away using serial transmission.
Why serial transmission is more appropriate than parallel transmission:

- More accurate/reliable/efficient over long distances
- Less chance of interference / cross talk
- ...that will skew / distort the data // less likely to get errors
- Data will arrive in order
- Serial is cheaper to purchase/install/maintain

Why serial may be used when computer is connected to router in different room

- Data arrives in order sent // does not need reordering
- Less likely to experience interference
- Less likely to have errors
- Can transmit over a longer distance (i.e. another room)
- Still fast transmission..
- ..sufficient for this purpose

Why internal circuits in a computer use parallel data transmission

- The distance travelled between the components is very short.
- High-speed transmission is essential.

Why parallel may be used when computer is connected to router in different room

- Faster transmission speed than serial
- ...faster response to requests
- Very long connection not needed
- ...next room is (likely) within distance for parallel
- ... unlikely to error/arrive out of sequence/skew

Company uses parallel half-duplex data transmission to transmit data for new videos to the web server, for users to stream. Explain why parallel half-duplex data transmission is the most appropriate method.

- Parallel would allow fastest transmission
- ... of the large amount of data
- Data can be uploaded and downloaded ...
- ... but this does not have to be at the same time
- Data is not required to travel a long distance
- ... therefore skewing is not a problem

Serial simplex data transmission

- Data is sent one bit at a time
- Data is sent using a single wire
- Data is sent in one direction only

Serial duplex data transmission

- Data is sent one bit at a time
- Data is sent using a single wire
- Data is sent in both directions ...
- ... at the same time

Serial half-duplex data transmission

- Data sent one bit at a time
- ... down a single wire
- Data sent in both directions ...
- ... but only one direction at a time

Parallel simplex data transmission

- Data is sent multiple bits at a time
- ... down multiple wires
- Data is sent in one direction only

Parallel half-duplex data transmission

- Data is sent multiple bits at a time
- Uses multiple wires
- Data is sent in both directions …
- … but only one direction at a time

Parallel-Full-duplex data transmission

- Data is sent multiple bits at a time
- Uses multiple wires
- Data is sent in both directions …
- … at the same time

Universal serial bus (USB)

How data is transmitted using USB connection

- Using serial transmission
- Data is sent one bit at a time
- Data is sent down a single wire

How it works

- Device is plugged into computer using USB ports.
- Computer automatically detects the presence of device.
- Device is automatically recognised.
- Computer will look for device driver that matches the device.
- Device driver software is installed/downloaded so computer and device can communicate

Benefits of using USB connection

- Universal connection: likely to be compatible with the computer
- Backward compatible: supports earlier versions; no additional technology required
- Cable can only be plugged in one way: cannot be inserted incorrectly
- Uses serial transmission: data less likely to be skewed / corrupted & is cheaper to manufacture/buy
- High speed transmission
- Supports different transmission speeds
- Automatically detected // automatically downloads drivers: no need to find them online / install them manually
- Powers the device // can charge/power mobile device at same time: no separate source of power is needed
- Doesn’t require a wireless network: can be used if a network is down

Drawbacks of using USB connection

- Has a maximum cable length: cannot be used over long distances
- Data transfer is not as fast as other wired systems; Older versions have limited transmission rate
- Very old USB standards aren't always compatible with the latest computers.

USB-C

- Can fit into the port either way round.
- Smaller and thinner.
- Faster data transmission rate.
- Offers more power.

Error Checking

Why data has to be checked for errors after transmission

- Computers require data in specific formats
- Computers carry out processes & calculations on data, which go wrong if data not in specific format
- Computers process/ represent information in binary: 1s and 0s
- Computers carry out processes & calculations on data, which go wrong if order bits are changed

How errors occur during data transmission

- Due to interference
    - Data loss - data is lost in transmission
    - Data gain - additional data is received
    - Data change - some bits have been changed or flipped
- Wireless technology uses electromagnetic signals (radio signals) to transmit data
    - Signals blocked by physical barriers (buildings etc)
    - Interference caused by bad weather (rain/ clouds)
    - Interference from other wireless signals
- Wired technology
    - Physical components damaged/ degrade
    - Interference from outside signals

Error detection methods

- Parity check
- Checksum
- Echo check
- Check digit
- Automatic repeat request // ARQ

Parity Check

Parity bit: A bit (0 or 1) added to a byte of data in the most significant bit position (left-most bit) to ensure that byte follows the correct parity bit protocol (even / odd).

Working

- Parity can be set to odd or even
- Sender and receiver agree on parity to use
- Data/email/image is split into bytes // blocks of 7 bytes
- Sender counts the number of 1s/0s in each group/byte
- Each group/byte is assigned a parity bit to match the parity/odd/even
- Receiving device/server recounts the number of 1s/0s in each group/byte
- … and compares to parity used/odd/even and if it does not match the parity, an error is reported/identified
- In block check, the location of the error(s) can be identified/estimated at the intersection

How odd parity check detects errors

- The number of 1 s/0 s are counted
- A parity bit is added to each byte/7 bits before transmission
- … to make the sum of the bits/1 or 0 s in each byte odd
- After transmission, if the number is odd no error is detected
- After transmission, if the number is even an error is detected

Why an error may not be detected in parity check

- There is a transposition of bits
- It does not check the order of the bits (just the sum of 1s/0s)
- Even number of errors occurred / even number of bits have changed
- Incorrect bits still add up to correct parity

NOTE: Parity checks only check that an error has occurred, they do not reveal where the error(s) occurred

Parity bytes & parity blocks

- Parity block consists of block of data with number of 1's totalled horizontally and vertically
- Parity byte is also sent with the data; it contains the parity bits from the vertical parity calculation

ODD	Parity bit	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Byte 1	0	1	1	0	1	0	1	1
Byte 2	0	0	0	0	1	0	0	0

Byte 3	1	0	1	0	1	1	1	1
Byte 4	1	0	1	1	1	0	0	1
Byte 5	1	1	0	1	0	1	0	1
Byte 6	1	1	0	0	1	1	1	0
Byte 7	0	0	1	1	1	1	1	0
Byte 8	0	1	0	1	1	0	0	0
Parity byte	0	1	1	1	1	1	1	1

- The above table uses odd parity
- Each byte row calculates the horizontal parity as a parity bit as normal
- Each bit column calculates the vertical parity for each row: parity byte.
- Parity byte is calculated before transmission and sent with the parity block
- Each parity bit tracks if error occurred in a byte; while parity byte calculates if error occurred in a bit column
- By cross referencing both horizontal and vertical parity values the error can be pinpointed

Explain how a parity block check might detect an error in transmission that would not be detected by a parity byte check.

- In parity check, interchange of bits will not be detected // Parity check cannot detect even number of changes // Parity check cannot detect error if parity stays correct ...
- ...the (possible) position of all changes will be highlighted // will identify the horizontal and vertical position of all differences/changes

Checksum

- Checksum value is calculated from the data ...
- ...using an algorithm
- Value is transmitted with the bits/ block of data
- Value is recalculated by the receiver using the same algorithm
- The recalculated value is compared to the transmitted value
- If checksum values are same, then data was transmitted without any error
- If checksum values are different, there is an error and a request is sent for the data to be re-transmitted

Echo check

- Copy of received data transmitted back to the sender.
- Sender compares returned to data to sent data to check for errors.

- If it does not match, error detected.
- If error does occur, sender will retransmit the data

Drawback: If 2 sets of data are different, it isn't known whether error occurred when sending data in the first place, or when sending data back for checking.

Check Digit

- Validation method
- Data entry check, not data transmission check: mis-typing, mis-scanning error, etc.
- Ensures data entered is correct

How it works

- Check digit is calculated from inputted data, using some mathematical calculation
- Check digit is appended to data & input
- Digit is recalculated after data has been input
- Calculated digit is compared to stored value
- If it matches, data entered is correct
- If it does not match, the data entered is incorrect

Examples of check digits

- International standard book numbers (ISBN); read how it works on SME
- Bar codes

Automatic Repeat Request (ARQ)

- Uses positive & negative acknowledgement + timeout
- Sender starts a timer when data is transmitted
- Receiver uses error checking method to check whether data has been received accurately
- If no error detected, a positive acknowledgement is returned to sender
- If error detected, negative acknowledgement is returned to sender
- If sender gets no acknowledgement within the set time, it resends the data
- Until acknowledgement is received / resend limit is reached

How Checksum and ARQ operate together to detect & correct errors

- Checksum used to detect errors (during transmission)
- ... using a calculated value
- ARQ checks if data is received
- ... uses acknowledgement and timeout
- ... requests data be sent again if (checksum) detects error / not received

How parity checks and ARQ operate together to detect and correct errors.

- Odd or even parity is set/agreed for the data
- A parity bit is added to each byte of data
- … to make the number of 1s match parity
- Data is checked after transmission to see if parity is correct
- ARQ uses acknowledgement and timeout
- If no error is found, a positive acknowledgement is sent to the sender / no acknowledgement is sent to the sender
- If an error is found, a negative acknowledgement is sent to the sender …
- … that triggers the data to be resent
- When the data is sent, a timer is started
- If an acknowledgement is not received within the time set, the data is resent …
- … until an acknowledgement is received / resend limit is reached

How ARQ operates using positive acknowledgement

- Timer is started when sending device transmits a data packet to receiver
- Receiving device checks the data packet for errors
- Once the receiving device knows the packet is error free it sends an acknowledgement back to the sending device …
- … and the next packet is sent
- If the sending device does not receive an acknowledgement before the timer ends …
- … a timeout occurs
- … the data packet is resent …
- … until acknowledgement received // until max number of attempts reached

Encryption

- Scrambles/ encodes data
- Making it meaningless/unintelligible
- Uses an encryption algorithm / key
- Data / plain text is changed to cipher text
- Data can be decrypted: turns the encrypted data into data that can be understood again

Purpose of encryption: Makes data meaningless if intercepted.

Plaintext: data before it is put encrypted using encryption algorithm.
Ciphertext: data after it has been encrypted using encryption algorithm.

Symmetric Encryption

Uses the same key to encrypt and decrypt message

Working

- Data is encrypted using encryption key/algorithm
- Data before encryption is known as plain text
- Data after encryption is known as cypher text
- Same key/algorithm is sent to receiver to allow data to be decrypted

Drawback: Issue of security/ distributing the key

Asymmetric Encryption

Uses public keys and private keys to ensure data is secure.

➢ **Public key:** Encryption key that is known to all users; used to encrypt data

➢ **Private key:** Encryption key that is known only to receiver; used to decrypt data.

Working

- Person A uses public key to encrypt data
- Person A sends data over the network/ internet
- Person B decrypts data using secret private key

Advantage: Only one private key can be used to decrypt the message and it is not sent over the internet.

How strength of encryption can be improved

- Increase the length of the key // use more than 128 bits: will generate more possibilities for key / less chance of decryption by brute force method
- Use a more complex encryption algorithm

How encryption keys are created

Key: binary string of certain length; when applied to encryption algorithm, can encrypt plaintext and decrypt ciphertext.

- Can be created manually, randomly or via an algorithm.
- Strong encryption keys are created using hashing algorithm.
- Hashing algorithm: non-reversible mathematical algorithm that converts a given input into an output. Once output has been generated it cannot be converted back to original input.
- Hashed encryption key can be sent symmetrically or kept secret as part of an asymmetric private key.
- Both sender and receiver need a copy of the key to decrypt information regardless of using symmetric or asymmetric encryption