Data Transmission

How data travels as packets, the five methods of transmission, how errors are detected, and how encryption keeps data private.

2.1 Types and methods of data transmission

Data Packets

Why data is broken down into packets

- **Efficiency:** Large files are divided into smaller packets to manage network traffic efficiently. Smaller packets can be routed through various paths in the network, leading to better use of bandwidth.
- **Reliability:** If a packet is lost or corrupted during transmission, only that packet needs to be resent, not the entire file. This increases the reliability of data transfer.
- **Speed:** Smaller packets can be processed and transmitted faster by network devices, improving overall transmission speed.

A packet has three parts: a header, a payload and a trailer.

1. **Packet header:** The packet header contains essential information required for **routing and managing the packet through the network**.
   - **Destination address:** This is the address of the receiving device. It ensures that the packet reaches the correct destination.
   - **Packet number:** Each packet is assigned a unique number. This helps in reassembling the packets in the correct order at the destination.
   - **Originator's address:** This is the address of the sending device. It allows the receiver to know where the packet came from and enables error messages or acknowledgments to be sent back to the sender.

   For example:

   ```
   Destination Address:   192.168.1.1
   Packet Number:         001
   Originator's Address:  192.168.1.2
   ```

2. **Payload:** The payload is the **actual data** being transmitted. It is the portion of the packet that contains the meaningful content, such as parts of a file, a message, or any other data being sent.

Note Academy

3. **Trailer:** The trailer contains **error-checking information** to ensure data integrity. It helps in detecting any corruption or loss of data during transmission. Commonly, it includes a checksum.

Packet Switching

Packet switching is a way of transferring data across networks. Below is a detailed explanation about how it works:

1. When data needs to be sent over a network, it is first **broken down** into smaller, manageable units called packets.
2. Each packet is **sent independently and can take different routes** to reach the destination. This means packets from the same data set may travel through various paths depending on network conditions like traffic load and available bandwidth.
3. Routers are network devices that **determine the shortest/quickest path** for each packet to take to reach its destination.
4. Because packets can take different routes, they **may arrive at the destination in a different order** than they were sent.
5. At the destination, **packets are reassembled into their original order** to reconstruct the complete data. The reassembly process waits until all packets have arrived, including the last one, which indicates the end of the transmission.
6. If any of the packets are missing, it requests the client to send it again.

Advantages

- Optimizes the use of network resources. Since packets can take any available path, the network can avoid congested areas, making the transmission process more efficient.
- If a packet is lost or corrupted, only that packet needs to be resent, not the entire message. This increases the reliability of data transmission.
- Supports a large number of devices and data transmissions. Packet switching can handle varying amounts of data traffic smoothly.
- Makes efficient use of bandwidth, reducing the need for dedicated circuits, which can be more expensive.

Disadvantages

- It may result in time delays when correcting errors.
- Requires complex protocols for delivery.
- Unsuitable for real-time transmission applications.

Note Academy

Methods of Data Transmission

Serial transmission

Serial transmission sends data one bit at a time over a single channel or wire. Data bits are transmitted sequentially, one after another, along the communication channel. This reduces the risk of data corruption and ensures data integrity over long distances. It is ideal for long-distance communication where data integrity is crucial. It is commonly used in USB connections and serial ports.

Advantages

- Cost-effective due to fewer wires needed.
- Less cross-talk and electromagnetic interference.
- Data won't be skewed.

Disadvantages

- Slower data transmission rate compared to parallel transmission.

Parallel transmission

Parallel transmission sends multiple bits simultaneously over multiple channels or wires. This method is typically used for short-distance communication. Multiple bits are transmitted at the same time, with each bit travelling on a separate wire. For example, an 8-bit data byte can be sent over eight wires in parallel. It is suitable for short-distance communication where speed is essential. This type of transmission is used in internal computer data buses and printer connections.

Advantages

- Faster data transmission rate due to simultaneous transmission of multiple bits.

Disadvantages

- More susceptible to cross-talk and electromagnetic interference.
- Requires more wires, making it less cost-effective for long distances.

Simplex transmission

Simplex transmission is a one-way communication method where data flows in a single direction only. Data can only be sent from the sender to the receiver, with no capability for the receiver to send data back to the sender. Appropriate for one-way communication scenarios, such as keyboard inputs to a computer or TV broadcasts.

Advantages

- Simple and easy to implement.

Note Academy

- Less complex and lower cost.

Disadvantages

- Limited functionality due to one-way communication.

Half-duplex transmission

Half-duplex transmission allows data to be sent in both directions, but not simultaneously. Communication can occur in one direction at a time. Devices take turns in sending and receiving data. When one device is sending, the other must wait to receive and vice versa. Suitable for two-way communication that doesn't require simultaneous data exchange, such as walkie-talkies and two-way radios.

Advantages

- More efficient than simplex as it allows two-way communication.
- Less complex than full-duplex systems.

Disadvantages

- Communication is not simultaneous, which can slow down the data exchange process.

Full-duplex transmission

Full-duplex transmission allows data to be sent and received simultaneously in both directions. Both devices can send and receive data at the same time, using separate channels for sending and receiving. It is ideal for scenarios requiring simultaneous two-way communication, such as telephone networks and modern network communication systems.

Advantages

- Maximizes communication efficiency with simultaneous data transmission.
- Faster and more efficient than half-duplex.

Disadvantages

- More complex and can be more expensive to implement.
- Requires more sophisticated hardware to handle simultaneous communication.

The Universal Serial Bus (USB) Interface

The **Universal Serial Bus (USB)** interface is a widely-used standard for connecting peripherals to a computer. It has evolved over the years and has become the go-to interface for data transfer and device connectivity due to its simplicity, speed, and versatility. Below is an explanation of how USB works:

1. Data transmission

Note Academy

- **Serial communication:** USB uses serial communication, meaning data is transmitted one bit at a time over a single channel. This is efficient and reduces the complexity of the cabling.
- **Packet-based transmission:** Data is transmitted in packets, which include control information, addresses, and error-checking data along with the actual payload.

2. USB protocol

- **Host-peripheral model:** USB operates on a host-peripheral model, where the host (usually a computer) controls communication with the connected devices (peripherals).
- **Plug and play:** Devices can be connected and disconnected without rebooting the system, and they are automatically recognized and configured by the host.
- **Power supply:** USB can supply power to connected devices, which is useful for peripherals like mice, keyboards, and external drives.

Benefits of the USB interface

1. **Universality:** USB is compatible with a wide range of devices, including keyboards, mice, printers, external storage, cameras, and mobile devices.
2. **Ease of use:** Devices can be easily connected and recognized by the host system without the need for additional drivers or software. Devices can be connected and disconnected while the computer is running.
3. **Speed:** USB supports multiple speeds, including Low Speed (1.5 Mbps), Full Speed (12 Mbps), High Speed (480 Mbps), SuperSpeed (5 Gbps), and the latest USB 3.2 and USB4 standards which support even higher data rates.
4. **Power supply:** USB ports can provide power to connected devices, eliminating the need for separate power adapters for many peripherals. USB Power Delivery (PD) can provide up to 100W of power.
5. **Cost-effective:** USB cables and connectors are inexpensive and widely available.

Drawbacks of the USB interface

1. **Limited cable length:** USB cables have a limited length, typically up to 5 metres for USB 2.0 and shorter for higher-speed versions, which can be a limitation for some setups.
2. **Speed limitations:** Multiple devices connected to the same USB hub share the available bandwidth, which can reduce data transfer speeds when many devices are used simultaneously.
3. **Power limitations:** While USB can supply power, the amount is limited, which may not be sufficient for power-hungry devices without additional power sources.
4. **Device dependency:** USB requires a host device to manage communication so peripherals cannot communicate directly with each other without a host.

Note Academy

2.2 Methods of error detection

Error Detection

Why check for errors?

- **Maintain data integrity:** Ensuring that the data received is the same as the data sent is crucial for accurate communication.
- **Communicate efficiently:** Detecting and correcting errors prevents misunderstandings and potential malfunctions in systems relying on precise data.

How errors occur

- **Interference:** External electromagnetic fields can disrupt the signal, causing bits to flip from 0 to 1 or vice versa. This is called Electromagnetic Interference (EMI). Radio waves from other electronic devices can interfere with the signal, leading to errors. This is called Radio Frequency Interference (RFI).
- **Data loss:** Some packets of data may be lost during transmission, resulting in incomplete data.
- **Data gain:** Sometimes, packets may be duplicated, causing the receiver to get more data than was originally sent.
- **Data change:** Individual bits may change value due to noise or other interference, leading to corrupted data.

Methods for Error Detection

When data is transmitted, ensuring its integrity is crucial. Various error detection methods are employed to check for errors that may have occurred during transmission. Below are detailed descriptions of the processes involved in parity checks (odd and even), checksum, and echo check, including parity byte and parity block check.

Parity check

Odd parity check

1. Count the number of 1s in the data bits.
2. Add a parity bit such that the total number of 1s (including the parity bit) is odd. If the number of 1s is already odd, add a parity bit of 0. If the number of 1s is even, add a parity bit of 1.
3. Send the data along with the parity bit to the receiver.
4. Count the number of 1s in the received data bits, including the parity bit.
5. If the total number of 1s is not odd, an error is detected.

Note Academy

NOTE

The main difference between the even and odd parity check is that in the odd parity check, a parity bit of 0 is added when the number of 1s are odd, whereas in the even parity check a parity bit of 1 is added when the number of 1s is odd, and vice versa.

Even parity check

1. Count the number of 1s in the data bits.
2. Add a parity bit such that the total number of 1s (including the parity bit) is even. If the number of 1s is already even, add a parity bit of 0. If the number of 1s is odd, add a parity bit of 1.
3. Send the data along with the parity bit to the receiver.
4. **Receiver side:** Count the number of 1s in the received data bits, including the parity bit.
5. **Error detection:** If the total number of 1s is not even, an error is detected.

Parity byte

1. Parity can be set to odd or even.
2. Sender and receiver agree on parity to use.
3. Data image is split into bytes.
4. Sender counts the number of 1s/0s in each byte.
5. Each byte is assigned a parity bit to match the parity.
6. Receiving device recounts the number of 1s/0s in each byte and compares to parity used and if it does not match the parity, an error is reported.

Parity block

1. Calculate parity for each row and each column of a block of data.
2. Send the block of data with row and column parity bits.
3. Check parity for each row and each column.
4. If any row or column parity does not match, an error is detected.

Checksum

1. Value is calculated from the data using an algorithm.
2. Value is transmitted with the data.
3. Value recalculated by receiver using same algorithm.
4. If checksum values are different, there is an error.

Note Academy

Echo check

1. Send data to the receiver.
2. Receive data from the sender. Send the received data back to the sender.
3. Compare the echoed data with the original data. If the data does not match, an error is detected.

Check digits

1. **Calculation:**
   - A check digit is calculated based on the digits in the original number using a predefined algorithm.
   - The algorithm often involves weighting each digit by its position, summing these weighted values, and performing a modulus operation to find the remainder.
   - The remainder (or its complement) becomes the check digit.
2. **Validation:**
   - When the number is entered or transmitted, the check digit is recalculated using the same algorithm.
   - The recalculated check digit is compared to the original check digit.
   - If they match, the data is considered valid. If not, an error is detected.

Examples of check digit usage

1. International Standard Book Number (ISBN)

ISBNs are used to uniquely identify books and other published materials. There are two formats: ISBN-10, which comprises 9 digits and a check digit, and ISBN-13, which comprises 12 digits and a check digit.

For an ISBN-13, the check digit is calculated by taking each digit, multiplying it by 1 or 3 alternately, summing these products, and taking the modulus 10 of the sum. The remainder is subtracted from 10 to find the check digit.

Example: ISBN-13 of "978-0-306-40615-7"

$$(9 \times 1 + 7 \times 3 + 8 \times 1 + 0 \times 3 + 3 \times 1 + 0 \times 3 + 6 \times 1 + 4 \times 3 + 0 \times 1 + 6 \times 3 + 1 \times 1 + 5 \times 3) \% 10 = 3$$

Then $10 - 3 = 7$ (check digit).

2. Bar codes

Bar codes used in retail and inventory management also utilize check digits to ensure accurate scanning and data entry.

- **UPC (Universal Product Code):**
  - Typically comprises 11 digits plus a check digit.

Note Academy

- The check digit is calculated by summing the odd-positioned digits, multiplying the sum by 3, adding the even-positioned digits, and then taking the modulus 10 of the sum. The remainder is subtracted from 10 to find the check digit.

Example: UPC "03600029145" with check digit "2"

$$(0 + 6 + 0 + 2 + 1 + 5) \times 3 + (3 + 0 + 0 + 9 + 4) = 42 + 16 = 58$$

$$58 \ \% \ 10 = 8, \qquad 10 - 8 = 2 \ (\text{check digit})$$

Automatic Repeat Query (ARQ) for error-free data transmission

1. The sender transmits a data packet to the receiver. Each packet is numbered sequentially to keep track of the order.
2. When the receiver successfully receives a packet and verifies it to be error-free, it sends a positive acknowledgment (ACK) back to the sender. If the receiver detects an error in the received packet, it sends a negative acknowledgment (NACK) back to the sender.
3. The sender starts a timer when it sends a packet. If the sender does not receive an ACK or NACK within a specified period (timeout), it assumes that the packet was either lost or corrupted. Upon timeout, the sender retransmits the packet.
4. If a NACK is received or a timeout occurs, the sender retransmits the packet. This process repeats until the sender receives an ACK, indicating successful transmission.

Key components of ARQ

- **Positive acknowledgment (ACK):** Indicates that the packet has been received successfully and without errors. The sender can then send the next packet in sequence.
- **Negative acknowledgment (NACK):** Indicates that the packet has been received but contains errors. The sender retransmits the same packet upon receiving a NACK.
- **Timeout mechanism:** The sender sets a timer when a packet is sent. If an acknowledgment (either ACK or NACK) is not received within the timeout period, the sender assumes the packet was not received correctly. The sender then retransmits the packet.

2.3 Encryption

Encryption

Why encryption is needed

1. **Data security:** Encryption transforms readable data (plaintext) into an unreadable format (ciphertext). Only authorized parties with the correct decryption key can convert the ciphertext back into plaintext. This ensures that even if data is intercepted, it cannot be understood without the key.
2. **Confidentiality:** Encryption protects sensitive information from being accessed by unauthorized users.

Note Academy

3. **Integrity:** Encryption helps maintain data integrity by ensuring that any unauthorized modification of data can be detected. Any tampering with the encrypted data will result in incorrect decryption, indicating that the data has been altered.
4. **Authentication:** Encryption can be used to verify the identity of the sender and the integrity of the message. This ensures that the data is coming from a legitimate source and has not been altered during transmission.
5. **Compliance:** Many industries and regulations require encryption to protect sensitive data. Compliance with standards often necessitates the use of encryption to safeguard data.

Purpose of encryption in data transmission

1. **Protection against eavesdropping:** Encryption prevents unauthorized parties from listening to or intercepting private communications.
2. **Secure online transactions:** Encryption is essential for securing online transactions, including e-commerce purchases and online banking. It ensures that financial and personal information is protected during transmission.
3. **Preventing data breaches:** By encrypting data, organizations can reduce the risk of data breaches. Even if an attacker gains access to the data, they cannot read it without the decryption key.
4. **Safe communication:** Encryption enables safe communication between parties. It ensures that messages remain private and confidential.
5. **Protecting sensitive information:** Encryption is used to protect a wide range of sensitive information, including medical records, intellectual property, and personal data. This ensures that sensitive information remains confidential and secure.

Symmetric Encryption

1. **Encryption:** The sender uses an encryption algorithm and a secret key to convert plaintext (readable data) into ciphertext (unreadable data).
2. **Transmission:** The ciphertext is sent to the recipient over the network. Even if intercepted, the data remains unreadable without the key.
3. **Decryption:** The recipient uses the same secret key and the corresponding decryption algorithm to convert the ciphertext back into plaintext.

Example: If the plaintext message is "HELLO" and the secret key is "12345", using the encryption algorithm, "HELLO" is converted to ciphertext. The recipient uses "12345" to decrypt the ciphertext back into "HELLO".

Advantages

- Faster and more efficient for large data volumes.
- Simpler and less computationally intensive.

Note Academy

Disadvantages

- Key distribution can be challenging since the same key must be securely shared between sender and recipient.
- If the key is compromised, the security of the encrypted data is also compromised.

Asymmetric Encryption

1. **Key pair generation:** Each user generates a public and a private key. The public key is shared with others, while the private key is kept secret.
2. **Encryption:** The sender encrypts the plaintext using the recipient's public key.
3. **Transmission:** The encrypted message (ciphertext) is sent to the recipient.
4. **Decryption:** The recipient uses their private key to decrypt the ciphertext back into plaintext. Only the recipient's private key can decrypt the data encrypted with their public key.

Example: If Alice wants to send a secret message to Bob, Alice uses Bob's public key to encrypt the message. Bob uses his private key to decrypt the message and read it.

Advantages

- No need to share a secret key beforehand.
- Enhanced security since the private key is never shared.

Disadvantages

- Slower and more computationally intensive compared to symmetric encryption.
- Requires more processing power, which can be a limitation for large data volumes.

Note Academy