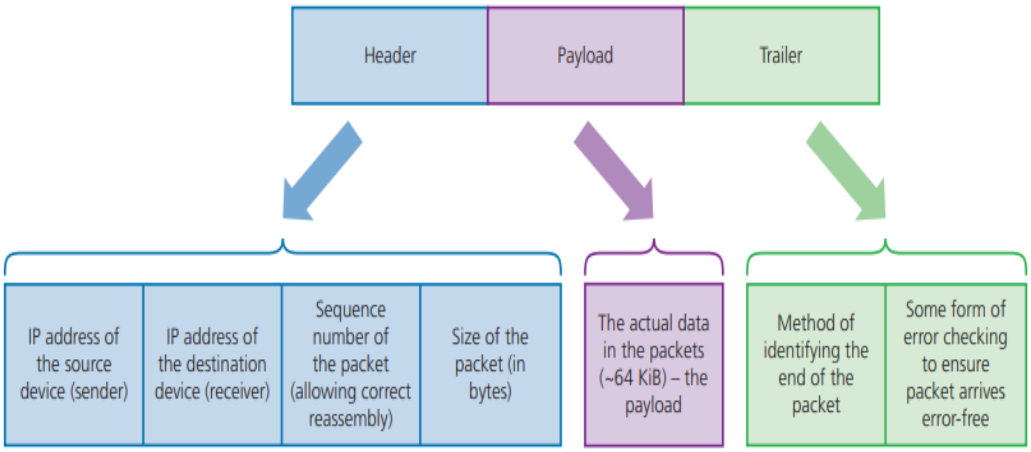
Data Transmission

Types and Methods of Data Transmission

Data Packets

- **Packet Structure -**
    - **Header**
        - Contains the IP address of the sender and the receiver
        - The sequence number of the packet
        - Size of the packet
    - **Payload**
        - Contains the actual data
    - **Trailer**
        - Includes a method of identifying the end of the packet
        - Error-Checking methods
        - 

- Packet Switching - Method of data transmission where the data is broken into multiple packets. Packets are then sent independently from start to end and reassembled at the receiver's computer.

Advantages	Disadvantages
There is no need to create a single line of communication	Packets may be lost
Possible to overcome failed or busy nodes	More prone to errors in real-time streaming
High data transmission speed	Delay at the receiver while the packets are being re-ordered
Easy to expand package usage	

Data Transmission

- ***Simplex data transmission*** is in one direction only (e.g. computer to printer)
- ***Half-duplex data transmission*** is in both directions but not at the same time (e.g., in a phone conversation where only one person speaks)
- ***Full-duplex data transmission*** is in both directions simultaneously (e.g. broadband connection on the phone line)
- ***Serial data transmission*** is when data is sent one bit at a time over a single wire
- ***Parallel data transmission*** is when data of several bits (1 byte) are sent down several wires at the same time.

Comparison of Serial and Parallel Data Transmission

Serial	Parallel
Better for longer distances (Telephone Lines)	Better for short distances (Internal circuits)
Cheaper Option	Expensive (More hardware required)
Used when the size of data transmitted is small	Used when speed is necessary
Slower Option	Faster than Serial

Serial	Parallel
less risk of external interference than with parallel (due to fewer wires)	faster rate of data transmission than serial
more reliable transmission over longer distances	works well over shorter distances (for example, used in internal pathways on computer circuit boards)
transmitted bits won't have the risk of being skewed (that is, out of synchronisation)	since several channels/wires used to transmit data, the bits can arrive out of synchronisation (skewed)
used if the amount of data being sent is relatively small since transmission rate is slower than parallel (for example, USB uses this method of data transmission)	preferred method when speed is important
used to send data over long distances (for example, telephone lines)	if data is time-sensitive, parallel is the most appropriate transmission method
less expensive than parallel due to fewer hardware requirements	parallel ports require more hardware, making them more expensive to implement than serial ports
	easier to program input/output operations when parallel used

Universal Serial Bus (USB)

- USB is an asynchronous serial data transmission method
- USB consists of:
    - Four-wire shielded cable
    - Two wires are used for power(+5V) and Earth (Ground)
    - Two wires are used in data transmission

Advantages	Disadvantages
Automatically detected	Transmission rate is less than 120 MB/sec
Only fit one way, prevents incorrect connections	Maximum cable length is about 5 metres
Different data transmission rates	
Backwards compatible	
Industry-standard	

Benefits	Drawbacks
devices plugged into the computer are automatically detected and device drivers are automatically loaded up	standard USB only supports a maximum cable length of 5 m; beyond that, USB hubs are needed to extend the cable length
connections can only fit one way preventing incorrect connections being made	
it has become an industry standard, which means considerable support is available	even though USB is backward compatible, very early USB standards (V1) may not always be supported by the latest computers
can support different data transmission rates (from 1.5 Mbps to 5 Gbps)	
no need for external power source since cable supplies +5 V power	even the latest version 3 (V3) and version 4 (V4) USB-C systems have a data transfer rate which is slow compared to, for example, Ethernet connections (Note: USB V2 has a maximum data transfer rate of 480 Mbps.)
USB protocol notifies the transmitter to re- transmit data if any errors are detected; this leads to error-free data transmission	
it is relatively easy to add more USB ports if necessary, by using USB hubs	
USB is backward compatible (that is, older versions are still supported)	

Methods of Error Detection

Parity Checks

- It uses the number of 1-bits in a byte
- Type Types -
    - Even - Even number of 1-bits
    - Odd - Odd numbers of 1-bits
- Example (Even Parity) -

0	**1**	**0**	**1**	**1**	**0**	**1**	**0**

- The **LMB (Left-Most Bit)** is the parity bit. As the number of 1s is even, the parity bit would be set to even.

Limitations with Parity Checks

- Two bits may change during transmission; therefore, error is not found

- Even though the parity checks would reveal the errors, the bit(s) changed wouldn't be identified

Parity Blocks

- To overcome the limitations of parity bits, Parity blocks would be used.

	1	2	3	4	5	6	7	
A	1	0	1	1	1	1	1	0
B	1	1	0	1	0	0	0	1
C	1	0	1	1	0	1	1	1
D	1	1	0	1	0	0	1	1
E	1	1	1	0	0	1	0	0
	1	1	1	0	0	1	1	1

- Any changes in bits would be identified through the rows and columns

Checksum

- Whenever a block of data needs to be sent, the sender would calculate the checksum value using a specific algorithm.
- Once the data has been sent, The receiver would calculate the checksum again with the same set of data and the same algorithm used before.
- The receiver would then compare the value received and the newly calculated value. If they aren't matched, A request is made to re-send the data.

Echo Check

- Once the data has been sent, the receiver will send the data back to the sender for verification.
- The sender would compare the received and original data for errors.

- The only downside is that we wouldn't know if the error occurred when sending the data or sending the data back for verification.

Check Digits

- Check digits are calculated from all the other digits in the data (ex-codes). The check digit would be the last digit of the code.
- These are used to identify mistyping errors such as -
    - 6372 typed as 6379
    - 8432 typed as 842
- The two examples of check digit are
        - ISBN-13
        - Modulo-11

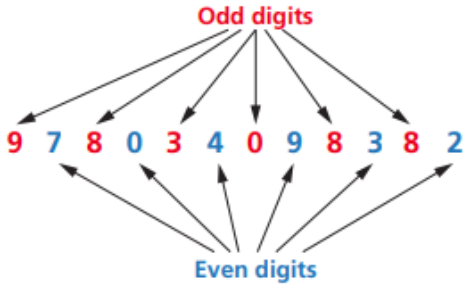
ISBN-13

Calculation 1 – Generation of the check digit from the other 12 digits in a number

The following algorithm generates the check digit from the 12 other digits:

1 add all the odd numbered digits together
2 add all the even numbered digits together and multiply the result by 3
3 add the results from 1 and 2 together and divide by 10
4 take the remainder, if it is zero then use this value, otherwise subtract the remainder from 10 to find the check digit.

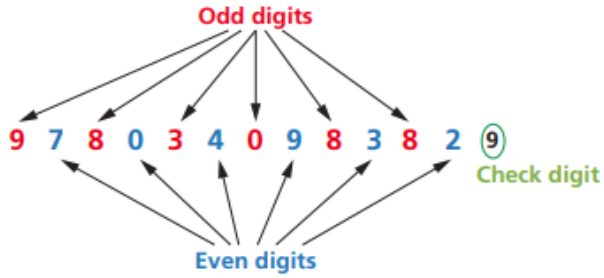
Using the ISBN **9 7 8 0 3 4 0 9 8 3 8 2** (note this is the same ISBN as in Figure 2.15):



▲ **Figure 2.16** ISBN (no check digit)

1 **9 + 8 + 3 + 0 + 8 + 8 = 36**
2 **3 × (7 + 0 + 4 + 9 + 3 + 2) = 75**
3 (**36** + **75**)/10 = 111/10 = 11 remainder 1
4 10 – 1 = **9** the check digit

So we end up with the following thirteen-digit number (which matches the number shown in Figure 2.15):



▲ **Figure 2.17** ISBN (including the check digit)

Calculation 2 – Re-calculation of the check digit from the thirteen-digit number (which now includes the check digit)

To check that an ISBN 13-digit code is correct, including its check digit, a similar process is followed:

1 add all the odd numbered digits together, **including** the check digit
2 add all the even number of digits together and multiply the result by 3
3 add the results from 1 and 2 together and divide by 10
4 the number is correct if the remainder is zero.

Using the ISBN **9 7 8 0 3 4 0 9 8 3 8 2 9** (including its check digit) from Figure 2.17:

1 **9 + 8 + 3 + 0 + 8 + 8 + 9 = 45**
2 **3 × (7 + 0 + 4 + 9 + 3 + 2) = 75**
3 (45 + 75)/10 = 120/10 = 12 remainder **0**
4 remainder is 0, therefore number is correct.

Modulo-11

Here are the steps to calculate a modulo 11 check digits:

1. Assign weights to each digit of the code, starting from the rightmost digit. The weight sequence usually starts with 2 and increases by 1 for each subsequent digit. For example, if the code has 7 digits, the weights would be 2, 3, 4, 5, 6, 7, and 8.
2. Multiply each digit of the code by its corresponding weight.
3. Sum up all the products obtained in step 2.
4. Calculate the remainder when the sum from step 3 is divided by 11.
5. If the remainder is 0 or 1, the check digit is 0. Otherwise, subtract the remainder from 11 to obtain the check digit.

Example:

The example to be used has the following seven-digit number:

1 7-digit number: 4 1 5 6 7 1 0

weighting values: 8 7 6 5 4 3 2

2 sum: (8 × 4) + (7 × 1) + (6 × 5) + (5 × 6) + (4 × 7) + (3 × 1) + (2 × 0)
= 32 + 7 + 30 + 30 + 28 + 3 + 0
total = 130

3 divide total by 11: 130/11 = 11 remainder 9

4 subtract remainder from 11: 11 – 9 = 2 (check digit)

So we end up with the following eight-digit: 4 1 5 6 7 1 0 2

Calculation 2 – Re-calculation of the check digit from the eight-digit number (which now includes the check digit)

To check that the eight-digit number is correct, including its check digit, a similar process is followed:

1 each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3, 2 or 1 starting from the left

2 the digit is multiplied by its weighting and then each value is added to make a total

3 the total is divided by 11

4 the number is correct if the remainder is zero

Using the 8-digit number: 4 1 5 6 7 1 0 2

1 weighting values: 8 7 6 5 4 3 2 1

2 sum: (8 × 4) + (7 × 1) + (6 × 5) + (5 × 6) + (4 × 7) + (3 × 1) + (2 × 0) + (1 × 2)
= 32 + 7 + 30 + 30 + 28 + 3 + 0 + 2
total = 132

3 divide total by 11: 132/11 = 12 remainder 0

4 remainder is 0, therefore number is correct

Automatic Repeat Requests (ARQs)

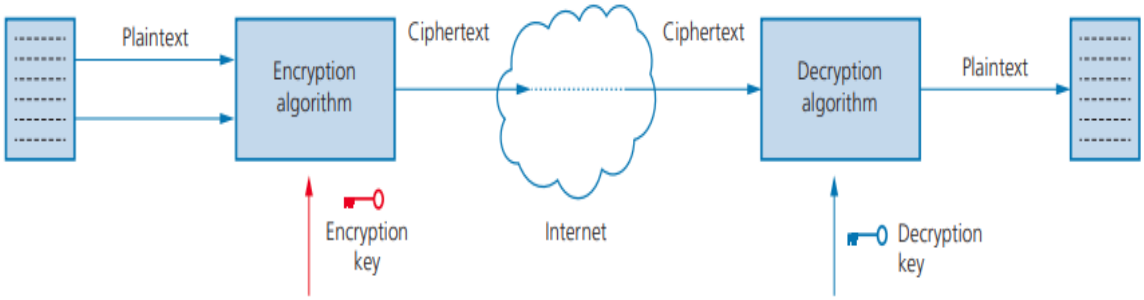
- Uses acknowledgements and timeouts to make sure the user received the data
- The receiver would check the data for any errors; if none are found, a positive acknowledgement is sent to the sender. However, if errors are found, a negative acknowledgement will be sent, and the data will be sent again.
- The sender uses timeouts to wait for a pre-determined amount for the acknowledgement.
- If no acknowledgements are received after the timeout, the data will be sent again to the receiver.

Encryption

- Encryption is a process of turning the data into an unreadable form so it doesn't make sense to hackers and other attackers.

Plaintext and Ciphertext

- Plaintext is the original data that is being sent
- Ciphertext is the text produced after encryption



Symmetric and Asymmetric Encryption

- Symmetric Encryption:
  - It uses an encryption key for the encryption process; the same key is used for encrypting and decrypting the data.
- Asymmetric Encryption:
  - Uses a public key and a private key. The public key is available to everyone, whereas the private key is only available to the user.
  - The receiver would have the private key, and they would send the public key to the sender. The sender can encrypt the message with the public key, and the data can be decrypted using the private key.