Error Detection Mechanisms

Monday, 5 December 2022 8:40 PM

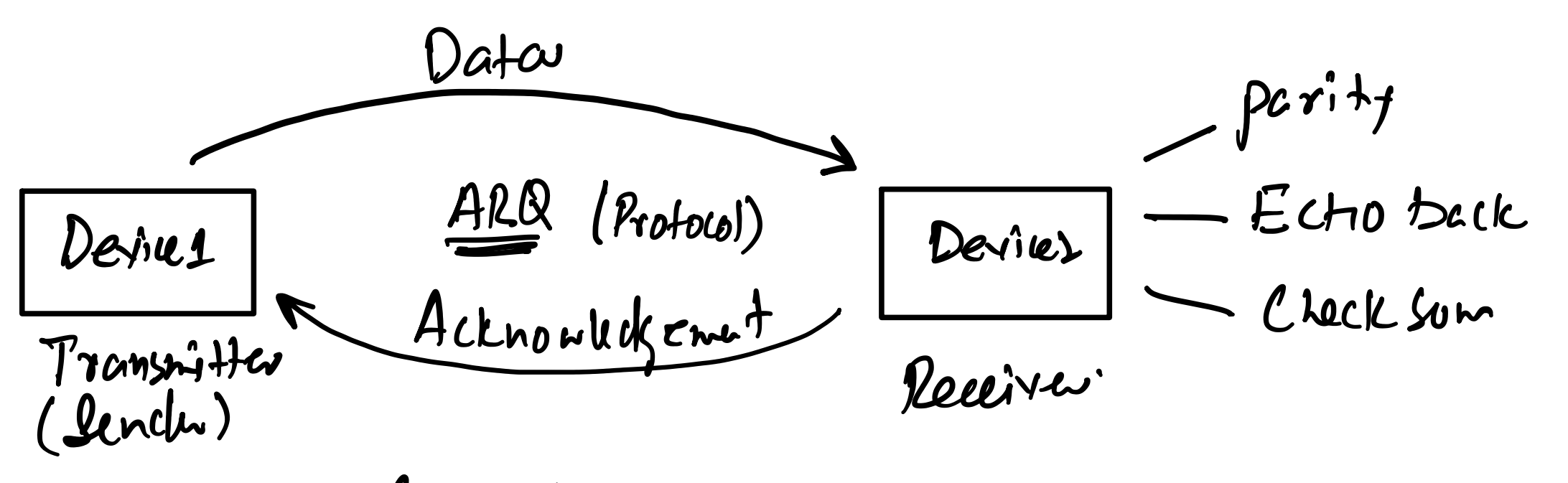
- Verification
  - Manual Transmission
    - Double Entry
    - Visual Check
  - Digital Transmission
    - ARQ
    - Parity
    - Checksum
    - Echo back

Verification is checking of data against the source. If data is transferred manually to computer then a double entry check or visual check is used.
If data is transferred digitally then ARQ, parity, checksum and echo back is used.

Double Entry: Complete data is entered twice and computer compares both entries to find if there was any discrepancy while data was being entered.

Visual Check: Once the data is entered, it is checked visually either on monitor or printout is matched with the source form.

Automatic Repeat Request (ARQ):



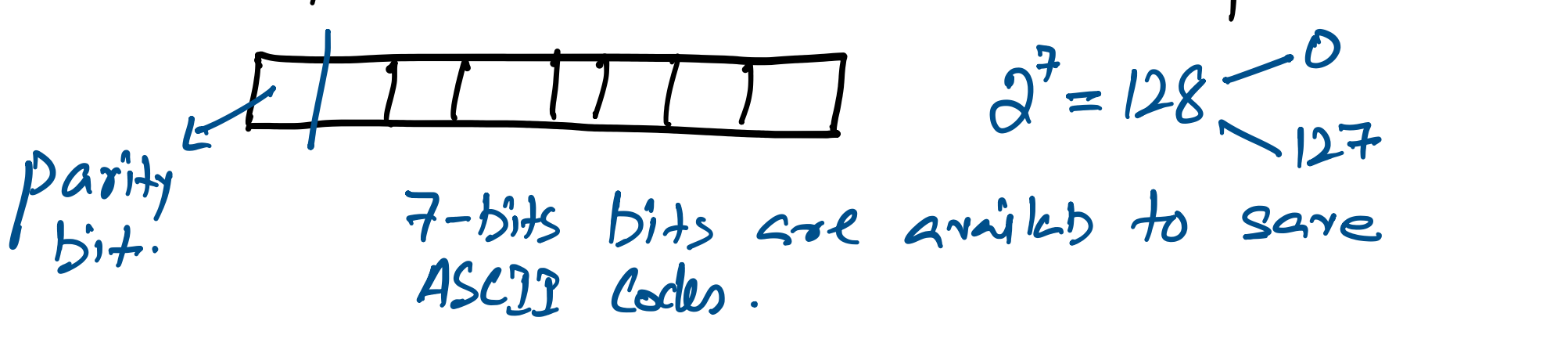
Protocol: Rules set forth between devices before actual data transfer occur. Eg: http, www, ARQ etc.

ARQ: It is a protocol (method) used to check whether data transferred is correct or not.

Acknowledgement: Receiver of the message sends an ack. to the sender when a msg is received.
The ack. is sent for both correct and incorrect receipt of data.
If the data received is "corrupted" the sender when receives negative ack., resends the data.
If the data received is "correct data" then next due data is sent.

Timeout: When two devices start data transfer, they set an agreed time (as part of protocol) for the receipt of ack.
If ack. is not received for that predefined/preset time, timeout occurs and sender resends the data.
This may go-on until ack finally arrives at sender's end.

Parity Check: It is a method used to identify data corruption upon receival of data/msg.
It is mainly used with text data in close proximity.



There are two parity mechanisms at work:
1. Byte parity.
2. Block parity.

They both use either "even" or "odd" parity types.

Example: Parity is the count of "1s"

CHAR	ASCII DENARY	ASCII BINARY	
A	65	01000001	"Even Parity"
B	66	01000010	
C	67	11000011	

CHAR	ASCII DENARY	ASCII BINARY	
A	65	11000001	"Odd parity"
B	66	11000010	
C	67	01000011	

(weakness)
There is a flaw in byte parity. When two bits are changed (corrupted) receiver cannot sense the corruption.
- If two bits are dropped (1→0 drop)
- If two bits are gained (0→1 gain)
- If two bits exchange position (transposition)

01000011
00000001 two bits are dropped.

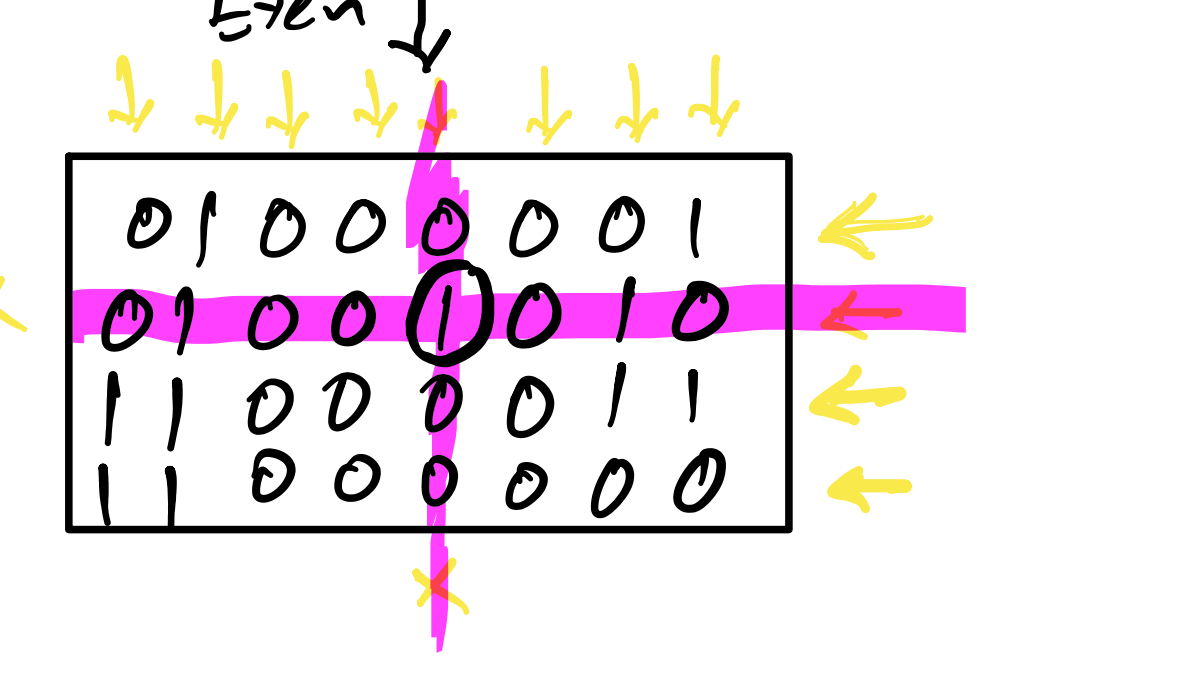
01000001
01110001 two bits are gained

11000011
11000101 two bits exchage their positions.

To overcome this weakness "BLOCK Parity" is used:
In block parity whole bunch of bytes with byte & column parity is sent.

A	01000001	"Even Parity Block"
B	01000010	
C	11000011	
Parity Byte→	11000000	

Sent Even



In block parity, receiver check every byte and column for predecided parity type. Whenever an error is identified, it is marked. At the cross-section of columns and rows of identified error, bits are flipped (0→1, 1→0) to correct the data.

Block parity helps in:
- Avoiding resending of data.
- Uses less time for data correction
- Uses less Internet bandwidth.

Checksum:

Another data checking mechanism used by receiver node to make sure data arrived is not corrupted.

Checksum is a small-sized datum derived from a block of digital data (binary) for the purpose of detecting errors that may have been introduced during its transmission or storage. Checksum, unlike parity, is applied to all types of data.

A = 65	0 1 0 0 0 0 0 1	
B = 66	0 1 0 0 0 0 1 0	← Block of data
C = 67	0 1 0 0 0 0 1 1	
Checksum	1 1 0 0 0 1 1 0	← Small-sized datum

Transferable block.

* While the data is transferred it may be changed for certain reasons. Changed data is called "corrupted data".
* For checksum small-sized datum is found using basic sum operation.
* Checksum:
  - Data is sent in blocks.
  - An additional value called "checksum" is sent at the end of block
  - Checksum is based on the number of bytes in the block.
  - If the checksum at the receiver's end doesn't match as receiver recalcutes the checksum, it is taken as the data is corrupted.

Algorithm used when checksum is calculated:
a. In the transfer block each byte's denary/binary worth is added together.
b. If the sum is within range of one byte, i.e 0–255 then it is treated as 'checksum' and sent along the block.
c. If the sum is greater than the limit of the byte then following additional steps are followed.
   - Divide the sum, X, of the bytes in block by 256
   - Round down the answer to the nearest whole number, Y.
   - Multiply Y with 256, Z.
   - Find the difference b/w X and Z
   - The difference is checksum.

X = 1185
1) 1185/256 ⇒ 4.629
2) Y = INT(4.629)
   Y = 4
3) Z = 4×256
   = 1024
4) Diff = X − Z
   1185 − 1024
   = 161 Checksum.

CHECK DIGIT VALIDATION:

Original		Types
	3529638	
	352638	Skip
	3539738	Change
	3592638	Swap (Transposition)

a b c d e ← Check Digit
×3 ×2 ×3 ×2
0 1 2 3 4 ← Entered CD.
0 + 2 + 6 + 6
= 14

10) 14 → 1
10
(4) our found CD.

1. Find Check Digit
2. check check Digit (Validation)

For calculating check digit we use **Modulo-11** method.

E.g: 4241508(n) — check digit

Digits	4	2	4	1	5	0	8	?
	×	×	×	×	×	×	×	
positions	1	2	3	4	5	6	7	8

4 + 4 + 12 + 4 + 25 + 0 + 56 = 105

11) 105 → 9
99
(6) Check Digit.

Sideline

2) 15 → (7) DIV
14
(1) MOD

Absolute
15/2 = 7.5
15 MOD 2 = 1
15 DIV 2 = 7.

42415086

Considerations: Exceptions:

- Remainder in Modulo-11 method will be from 0–10.
- 0–9 can be accomodated in single digit space but 10 can't be
- Two spaces holding remainder 10 will be replaced by roman 'X' digit.

Number entered: 3240045(X) — ignore the check digit and find your own. Then check both; if they are same then entered number is corrected otherwise corrupted/incorrect.