Packet/Frame, Error Detection

How to send data efficiently & reliably?

- Packet and Packet Communication
  - Shared Network Resource, Fairness, Reliability
- Frame
  - Byte Oriented Frame and Bit Oriented Frame
  - Byte Stuffing in BSC
  - Bit Stuffing in HDLC
- Error Detection and Redundancy Check
  - Parity Check
  - LRC (Longitudinal Redundancy Check)
  - Checksum

Postal System and Telephone System

Postal System

1. No need to make a connection before sending
2. Addresses in each letter/parcel
3. Many Letters/parcels share one link

Letter → Postal Office
Parcel → Postal Office
Postal Office → Dispatch Center
Dispatch Center → (L1~N) → Dispatch Center
Dispatch Center → Postal Office
Postal Office → Destination

Postal Service

Telephone System (~1970)

Phone Operator

Local Center — Exchange Center — Exchange Center — Local Center

1. Need to make a connection before communications
2. There exists a physical link between two telephones
3. The link is occupied by two telephones without share to others
4. Called **circuit switching communication**

Computer Communication and Packet

The data transmission method in computer communication is conceptually similar as the postal system. A large data stream will be divided into relatively small blocks, called packet, before transmission. Each packet is transmitted individually and independently over networks → **Packet switch/based Communication**.

Original data stream: 10011…01 01001…11 … … … … … … … … … 10100…10

10011…01	01001…11	…	10100…10
1st block	2nd block		Nth block

C-data	10011…01	C-data	01001…11	…	C-data	10100…10

1st **packet** 2nd **packet** … Nth **packet**

Circuit & Packet Switch

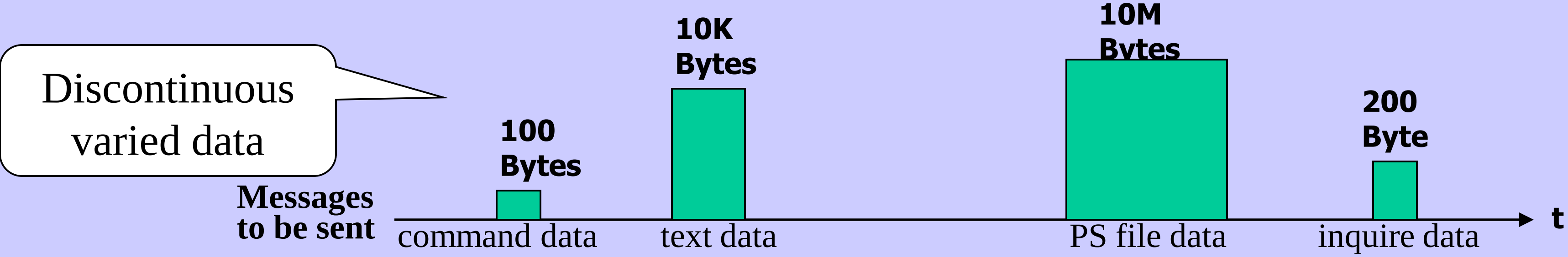
Why packet communication ???

C-data (control data):
- addresses
- packet size & order
- reliability
- synchronization
- others

Letter/parcel:
- addresses
- size & weight
- registered
- fast mail
- EMS, …

Motivations of Packet Communication
– Shared Communication Link/Network –

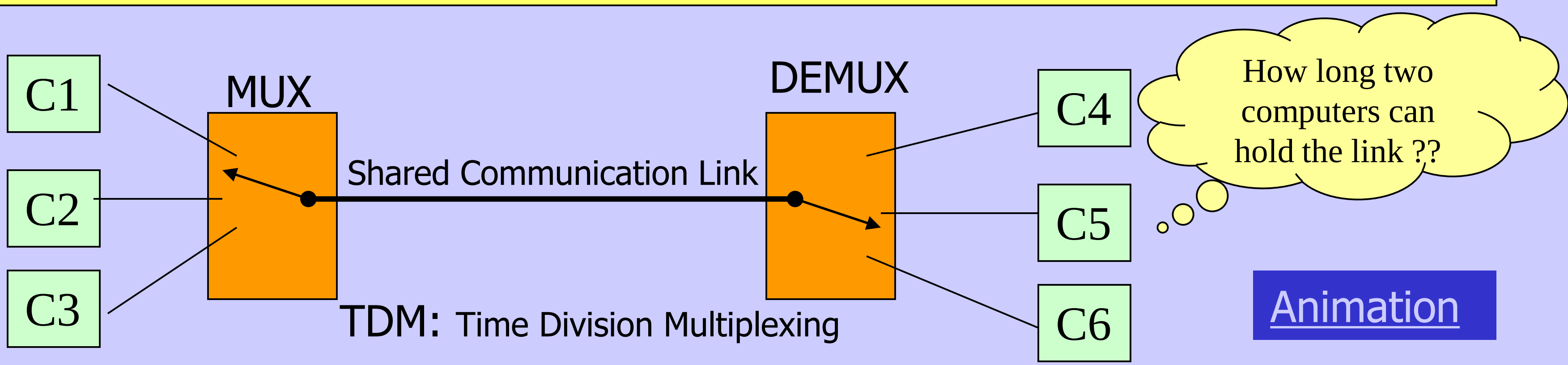
Features of computer data: discontinuous, varied data amount, high reliable



Because of the discontinuous and varied data, resources will be wasted if a pair of computers occupy a communication link/network for long time since they transmit data only in certain time periods, and there is no data transmission in most of time.

→ Many computers can share resources of a communication link/network

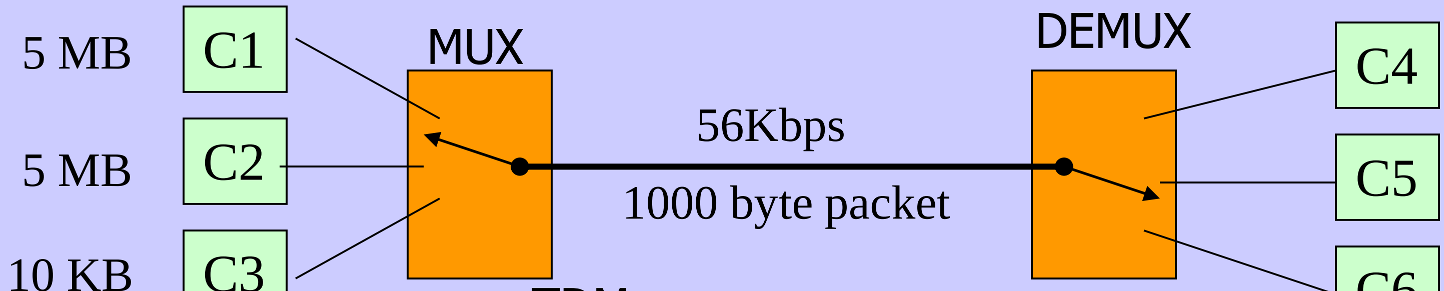
→ To increase efficiency of resource usage and reduce cost of building a network.



Animation

Motivations of Packet Communication – Fairness in Resource Sharing –

Message based multiplexing: after a computer sends a whole message, another can start.



TDM: Time Division Multiplexing

Examples

1. 5 MB file transferred across network with 56Kbps capacity will take about 12 minutes.

$$\frac{5\text{x}10^6 \text{ bytes} * 8 \text{ bits/byte}}{60 \text{ secs/minute} * 56\text{x}10^3 \text{ bits/second}} = 11.9 \text{ minutes}$$

All other computers will be forced to wait at least 12 minutes before initiating transfers.

2. Suppose the above file is broken into 1000 byte packets, each packet will take 0.143s to transmit

$$\frac{1000 \text{ bytes} * 8 \text{ bits/byte}}{56\text{x}10^3 \text{ bits/second}} = 0.143 \text{ seconds}$$ (ignoring packet control data)

The 2nd computer must only wait 0.143 seconds before beginning to transmit.

3. If both files are 5MB long, each takes 24 minutes to transmit when packet size is 1000 bytes.

4. If 3rd file is 10KB, it can transmitted in 4.3 seconds while two 5MB files takes roughly 24 minutes.

Conclusion: For fairness, packet size can't be very large but should be relative small !!!

Motivations of Packet Communication
– Reliability and Error Control–

Transmission errors will occur because of noise.

P_{bit}: probability of a bit error, P_{block}: probability of a block error

$P_{block} = 1-(1-P_{bit})^n \sim nP_{bit}$ when P_{bit} is very small

[10101110…010010110] – n bits, data block

Example: Suppose $P_{bit}=10^{-6}$,

n=1000, $P_{block}= 1000\times 10^{-6} = 10^{-3}$ (0.1% errors)

n=10000, $P_{block}= 10000\times 10^{-6} = 10^{-2}$ (1% errors)

n=100000, $P_{block}= 100000\times 10^{-6} = 10^{-1}$ (10% errors)

n=1000000, $P_{block}\rightarrow 1000000\times 10^{-6} = 1$ (100% errors)

An incorrect data block (packet) is unacceptable, and is needed to retransmit.

Two kinds of error control methods:

- Error Detection: detect if a received packet is correct
  Automatic Retransmission Request → **ARQ**
- Error Correction: correct errors in a received packet → **FEC**

General idea and format:

original data	Redundancy check

Conclusion: To improve reliability via error control technique, packet can't be very large!

Pioneers of Packet-based Communication

Paul Baran
(1926-2011, US)

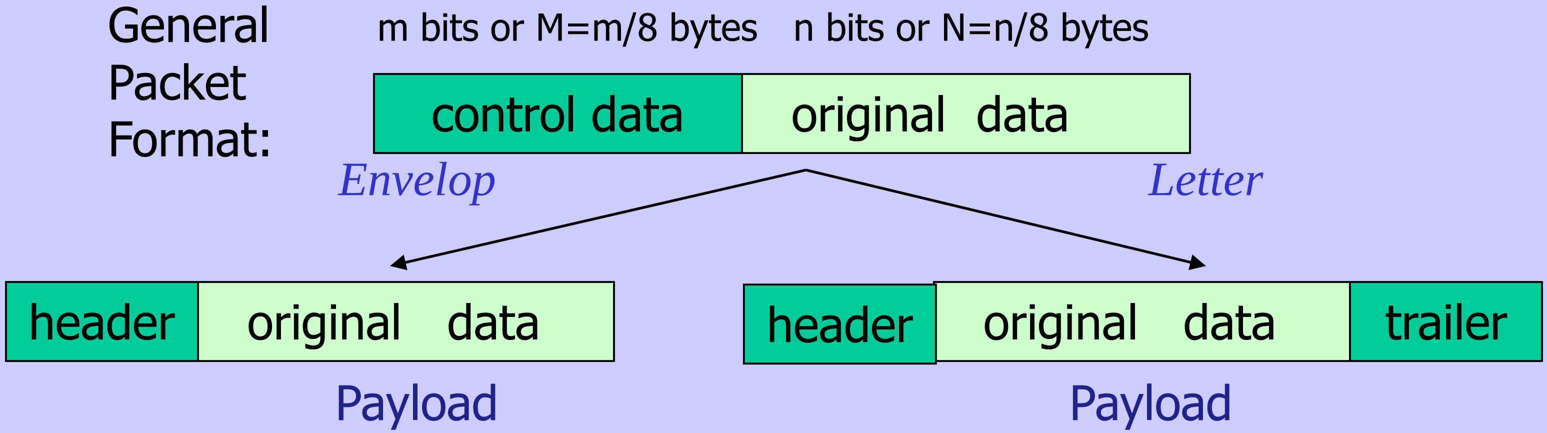
Donald W. Davies
(1924-2011, UK)

A pioneer of computer networks
Inventor of packet switching techniques
- an essential part of the Internet and other modern digital comm.

One of the inventors of packet switched computer networking, originator of the term, and the Internet itself can be traced directly back to his work.

Packet Format and Frame

General Packet Format:



Packet data efficiency: n/(n+m) = N/(N+M). It should not be too small.
Packet data overhead: m/(n+m) = M/(N+M). The smaller, the better.

- **Packet** is "generic" term that refers to a small block of data
- Each hardware/software communication technology uses different packet format
- **Frame** or hardware frame denotes a packet of a specific hardware technology
- Need to define a standard frame for data to indicate the frame beginning and end
- **Header** and trailer used to "frame" the original data - **Payload**.

Byte-Oriented Frame and Bit-Oriented Frame

Byte-oriented frame interprets a transmission frame as succession bytes (8 bits), all control data is in an existing encoding system such as ASCII characters.
Bit-oriented frame interprets a transmission frame as succession individual bits, made meaningful by their placement and values.

A byte-oriented frame of BSC (Binary Synchronous Communication, IBM in 1964)

soh						… data …					eot	CRC

- soh (start of header, 00000001): indicate a start of a frame data
- eot (end of text, 00000100): indicate an end of a frame data
- CRC (cyclic redundancy check): for error control

Animation

BSC Wiki

A bit-oriented frame of HDLC (High-level Data Link Control, ISO in 1979)

Flag	**add**	**cont**	… data …	**FCS**	**flag**

- Flag (01111110): identify both the beginning and end
- add (address): 8 or 16 bits, station id number
- cont (Control): 8 or more bits, frame order, etc.
- FCS (frame check sequence): 16 or 32 bits for error control
- HDLC based LAP, Frame Relay, PPP, LAN’s protocols, …

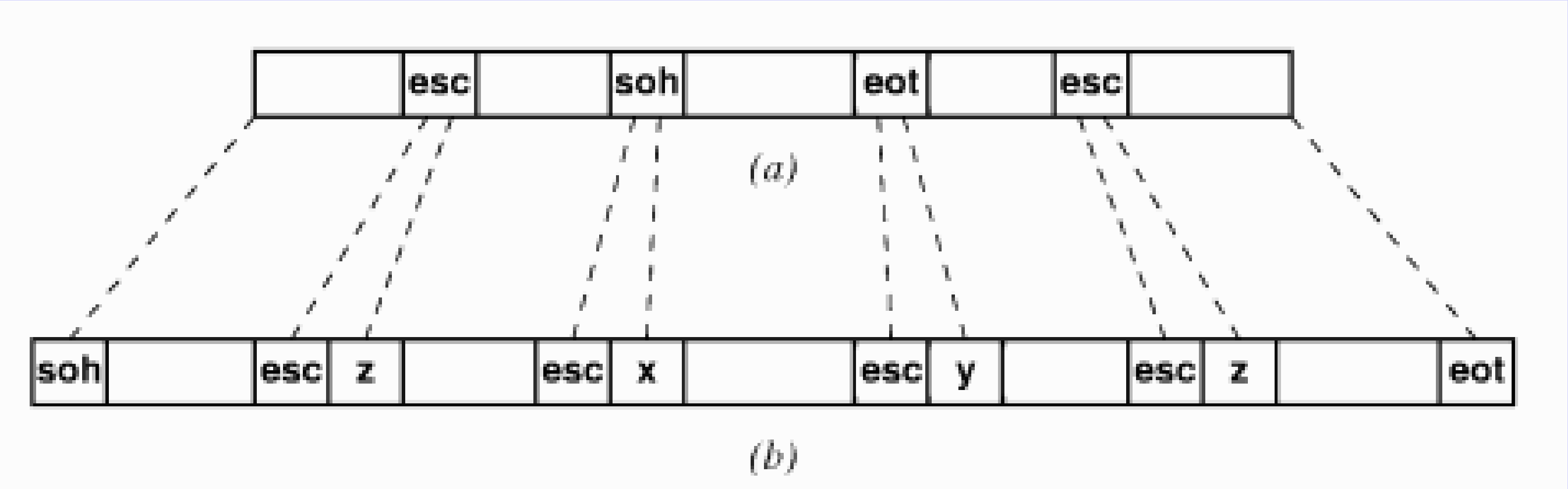
HDLC Wiki

Transmit Arbitrary Data – Byte Stuffing in BSC

- BSC was originally designed to transport only textual information data.
- Most of current data (e.g., program and graphics) that is nontextual data.
- An arbitrary data file may include specially reserved bytes like **soh** and **eot**
- The **soh** and **eot** will be misinterpreted if they exist inside a data file
- Byte stuffing is a technique for inserting extra data to encode reserved bytes
- For example in BSC:

byte in data	bytes sent
soh	esc x
eot	esc y
esc	esc z

Animation



Transmit Arbitrary Data – Bit Stuffing in HDLC

Flag (01111110)

Flag	add	cont	… data …	FCS	flag

- An arbitrary data file may also include the flag field, **01111110**.
- It may cause misinterpretation
- Bit stuffing, similar as byte stuffing, is a technique for inserting extra bit to data
- For example in HDLC: insert extra 0 after 5 consecutive 1s

Original data

… 0001111111001111101000 …

Stuffed data

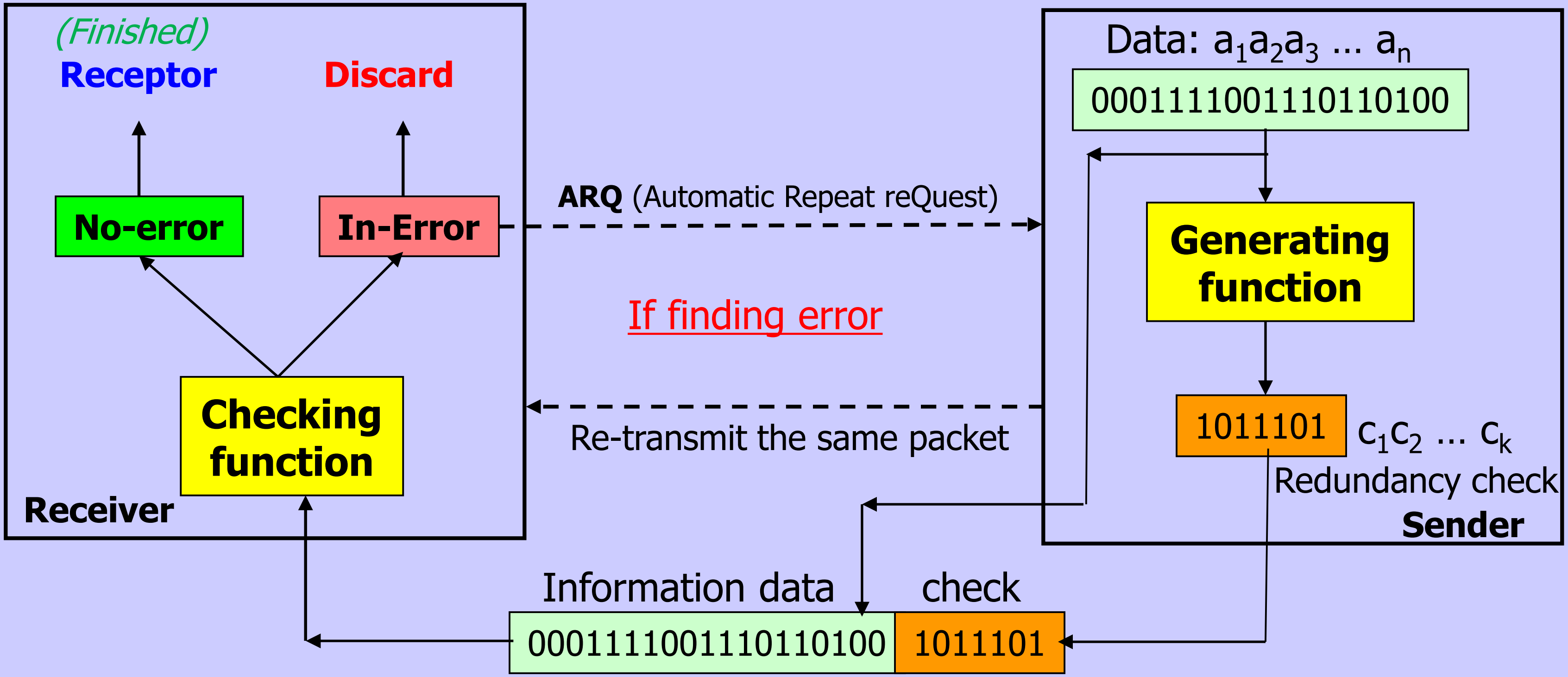
Flag	add	cont	… 000111110110011111001000 …	FCS	flag

… 0001111111001111101000 …

Unstuffed data

Error Detection and Redundancy Check

To detect errors in a received data block, extra data, called redundancy check, must be added to the end of the data block before sending the data.



n: number of information bits, k: number of check bits, k/(n+k): redundancy ration

Error Detection Wiki

Error Detection – Parity Check

Parity checking has only one parity check bit: $a_1a_2a_3 \ldots a_n$ | c_1

- Even parity C1= $a_1+a_2+a_3+\ldots+a_n$ Modulo 2 → $c_1=0$ for even number of 1s in $\{a_i\}$
- Odd parity C1= $a_1+a_2+a_3+\ldots+a_n+1$ Modulo 2 → $c_1=0$ for odd number of 1s in $\{a_i\}$

Example: Suppose to send the word "world" in ASCII code using **even** parity checking,

	w	o	r	l	d
Original data	1110111	1101111	1110010	1101100	1100100
Data with check	1110111**0**	1101111**0**	1110010**0**	1101100**0**	1100100**1**
Received data with bit errors	1110111**0**	1111111**0**	1011000**0**	1101100**0**	1110000**1**
	no error	1 error	3 errors	no error	2 errors
$(a_1+a_2+\ldots+a_7+c_1)\ \%2=?$		detectable	detectable		undetectable

Parity checking can detect bit errors of odd number: 1, 3, 5, …
can't detect bit errors of even number: 2, 4, 6, …
Error detection ability is usually depended upon the redundancy

Error Detection – Longitudinal Redundancy Check (LRC)

In LRC, data length n=u×v organized in a table (u rows & v columns)

$$\begin{array}{ll} & a_{11}\ a_{12}\ a_{13}\ \dots\ a_{1v} \\ & a_{21}\ a_{22}\ a_{23}\ \dots\ a_{2v} \\ & \quad\cdots\cdots \\ + & a_{u1}\ a_{u2}\ a_{u3}\ \dots\ a_{uv} \\ \hline \text{(mod 2)} & c_1\ \ c_2\ \ c_3\ \ \dots\ c_v \end{array}$$

(v-bits checks)

11100111 11011101 00111001 10101001	Data

v=8, u=4

11100111
11011101
00111001
10101001

8-bits LRC 10101010

Data plus check

11100111 11011101 00111001 10101001	10101010

Received data

A) 11101010 01100101 00111001 10101001 10101010 — 6 bit errors, detectable

B) 11101011 11011001 00111001 10101001 10101010 — 2 bit errors, undetectable

LRC can detect bit errors of odd number: 1, 3, 5, …
can detect many bit errors of even number: 2, 4, 6, …
can't detect some bit errors of even number: 2, 4, 6, …

Error Detection – Checksum

In checksum, data length n=uxv organized in a table (**u rows & v columns**)

$$
\begin{array}{ll}
 & a_{11}\ a_{12}\ a_{13}\ \dots\ a_{1v} \\
 & a_{21}\ a_{22}\ a_{23}\ \dots\ a_{2v} \\
 & \ \ \ \ \ \ \ \ \ \ \ \ \dots\dots \\
\textbf{+}\ \text{(carrier)} & a_{u1}\ a_{u2}\ a_{u3}\ \dots\ a_{uv} \\
\hline
\textbf{(mod 2)} & s_1\ \ s_2\ \ s_3\ \ \dots\ s_v\ \text{(sum)} \\
+\ (\textbf{complement}) & 1\ \ 1\ \ 1\ \ \dots\ 1 \\
\hline
\textbf{v-bits checksum} & c_1\ \ c_2\ \ c_3\ \ \dots\ c_v
\end{array}
$$

```
carrier     1 11   1
           00111001
         + 00101001
        ------------
           01100010
```

```
Carrier 11111111
         11100111
       + 01011101
      ------------
         01000100
       +        1
      ------------
         01000101
```

11100111 01011101 00111001 00101001 Data

v=8, u=4

11100111
01011101
00111001
00101001

sum 10100111

8 bits checksum 01011000

Data plus check

11100111 01011101 00111001 00101001	01011000

Checksum can detect bit errors of odd number: 1, 3, 5, …
can detect most bit errors of even number: 2, 4, 6, … (better than LRC)
can't detect few bit errors of even number: 2, 4, 6, …
widely used in Internet where v=16 or 32

Error Detection Video

Exercise 3

1. Describe what are the packet, the packet communication, and motivations using the packet in computer communications.

2. (1) An ASCII coded data block to be sent as the following:

esc	F	r	a	m	e	eot	B	y	t	e	soh

Draw its BSC frame with byte stuffing.

(2) Bit stuff the following data: 0011111111111001011111011

3. (1) Find the even and odd parity check for a 7-bit data: 1001011.

(2) Find the even parity check for each ASCII code of the word “packet”.

4. Find the 8 bit (v=8) LRC of the data: 1001 0011 1001 0011 1001 1000 0100 1101

5. Find the 16 bit (v=16) checksum of the data: 1001 0011 1001 0011 1001 1000 0100 1101