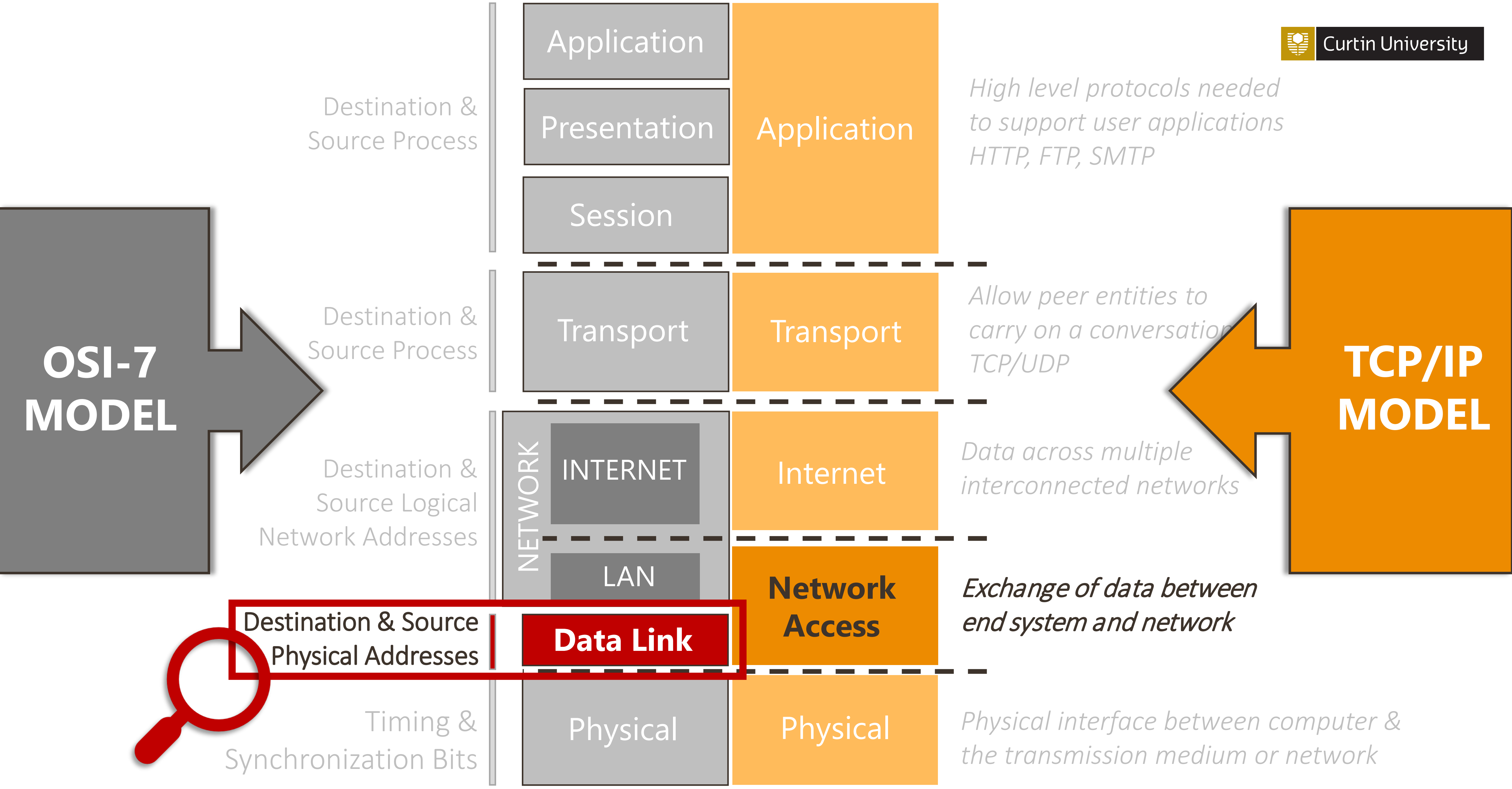
Curtin University

Data Link Layer I

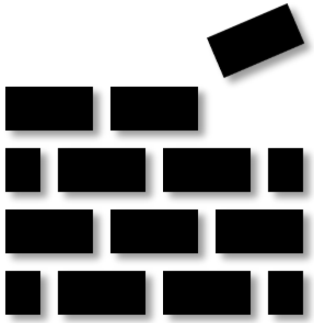
Prof. Ling Li | Dr. Nadith Pathirage | Lecture 03

Semester 1, 2021


OSI-7
MODEL
TCP/IP
MODEL
Destination &
Source Process
Application
Presentation
Session
Application
High level protocols needed
to support user applications
HTTP, FTP, SMTP
Destination &
Source Process
Transport
Transport
Allow peer entities to
carry on a conversation
TCP/UDP
Destination &
Source Logical
Network Addresses
NETWORK
INTERNET
Internet
Data across multiple
interconnected networks
LAN
Destination & Source
Physical Addresses
Data Link
Network
Access
Exchange of data between
end system and network
Timing &
Synchronization Bits
Physical
Physical
Physical interface between computer &
the transmission medium or network

Data Link Layer

Curtin University

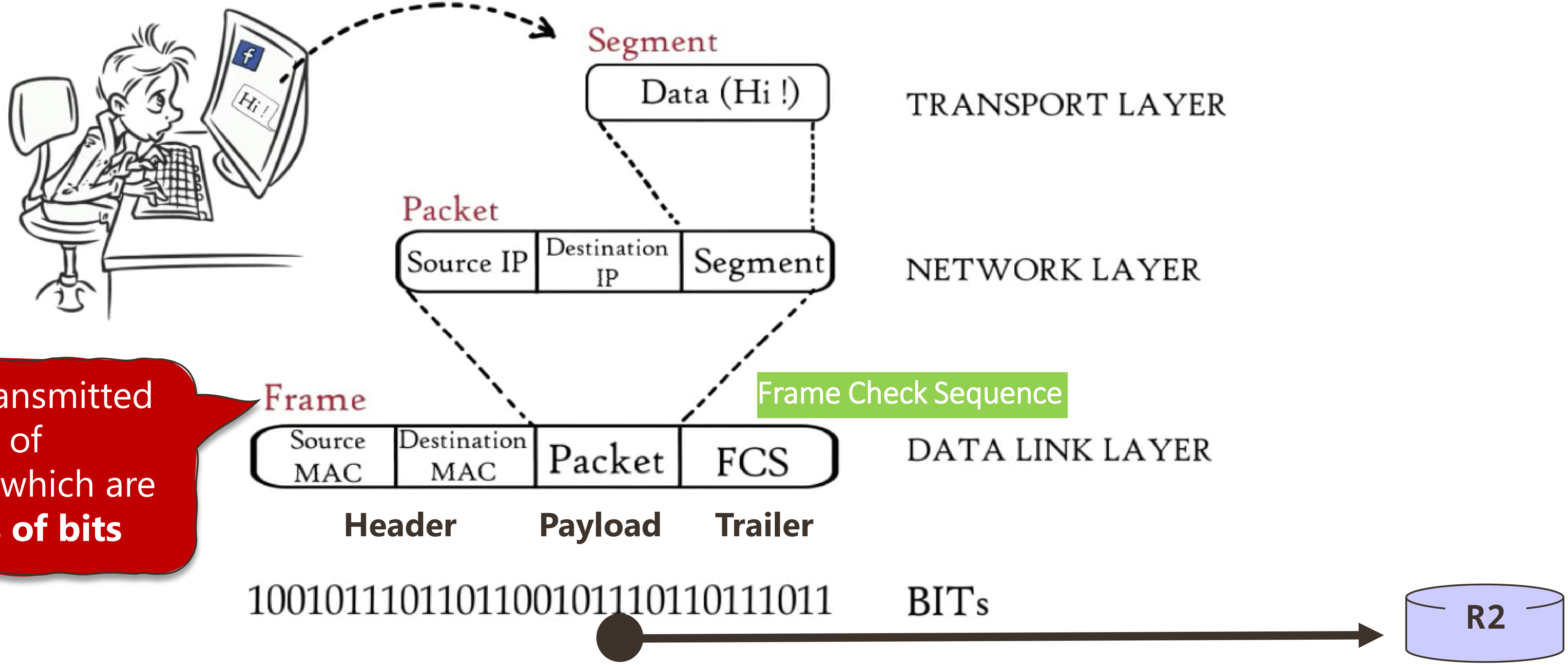


The data link layer provides the **building blocks for communication** across a variety of physical media

Data Link Layer

Curtin University

The data link layer is concerned with local delivery of **frames** between nodes

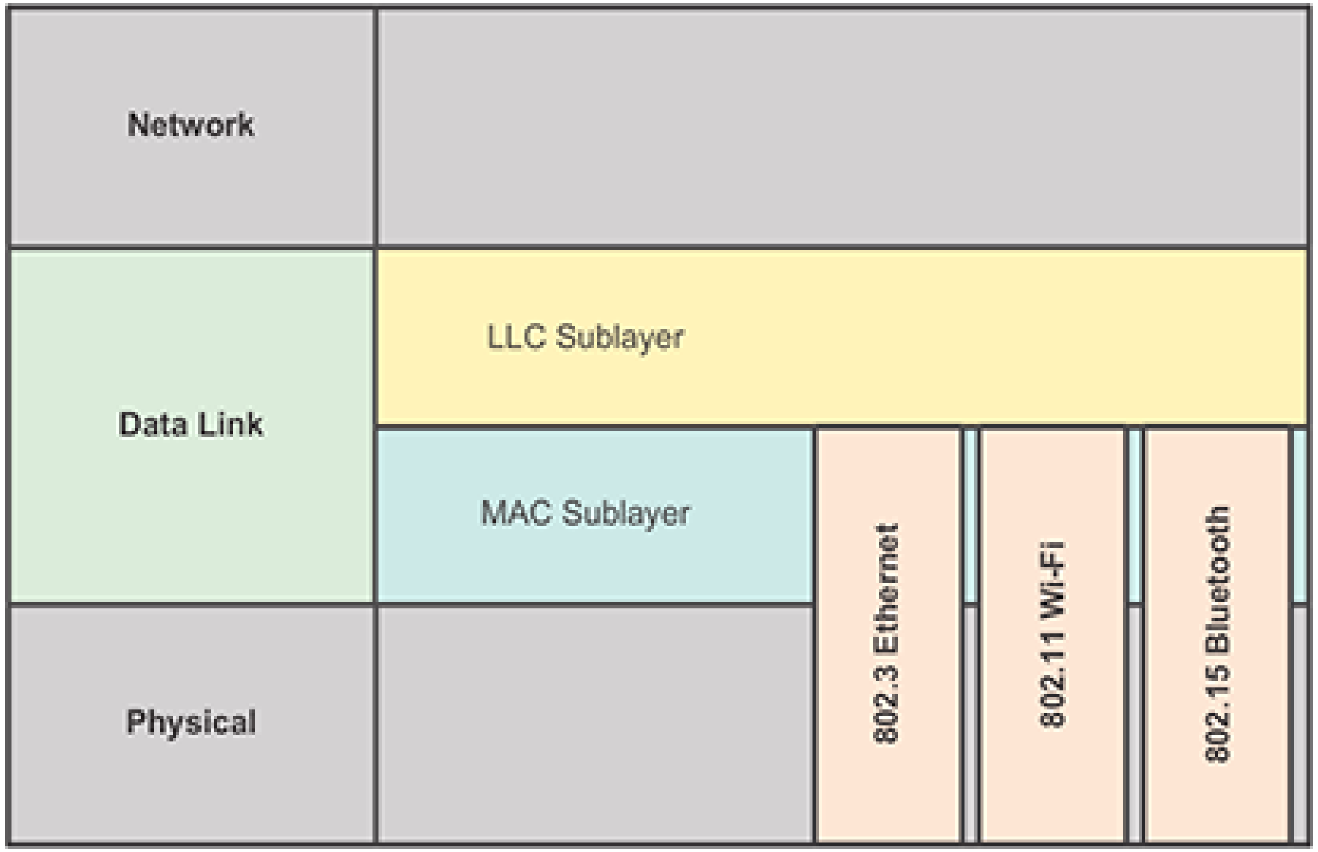


Data transmitted in units of frames which are **groups of bits**

- **Implementation**
  - Primarily implemented in software
  - May be embedded in physical devices such as switches and network adapters (firmware)

- **Data Link Layer Protocols**
  - Point-Point Protocol (PPP)
  - High-Level Data Link Control (HDLC)
  - IEEE 802.3 Ethernet LAN
  - IEEE 802.11 (Wi Fi) LAN

- **Two Sub Layers**
  1. **LLC (Logical Link Control)**
  2. **MAC (Media Access Control)**



Data Link Layer: Services

Curtin University

LLC:

- Provide services to network layer protocols
- **Flow Control**
- **Error Control**

MAC:

- **Framing:** bits to frame (vice versa)
- Physical addressing **(MAC addressing)**
- Multiple access methods for channel-access control **(CSMA/CD, CSMA/CA)**
- LAN switching (packet switching), including MAC filtering, **Spanning Tree Protocol (STP)**
- Data packet queuing or scheduling
- Store-and-forward switching or cut-through switching
- Quality of Service (QoS) control
- **Virtual LANs (VLAN)**

Curtin University

LLC Sub-Layer

Flow Control

- Stop-And-Wait Protocol
- Sliding Window Protocols
  - Go-Back-N ARQ
  - Selective-Reject ARQ

Automatic Repeat Request (ARQ)

- Ensure a sequence of information packets is delivered **in order** and **without errors** or **duplications** despite transmission errors & losses

1. **Stop-and-Wait Protocol/ARQ**
2. **Sliding Windows Protocol**
   - **Go-Back N ARQ**
   - **Selective Reject ARQ**

Basic elements

- ✓ Error-detecting code with high error coverage
- ✓ ACKs (positive ACKs)
- ✓ NAKs (negative ACKs)
- ✓ Timeout mechanism

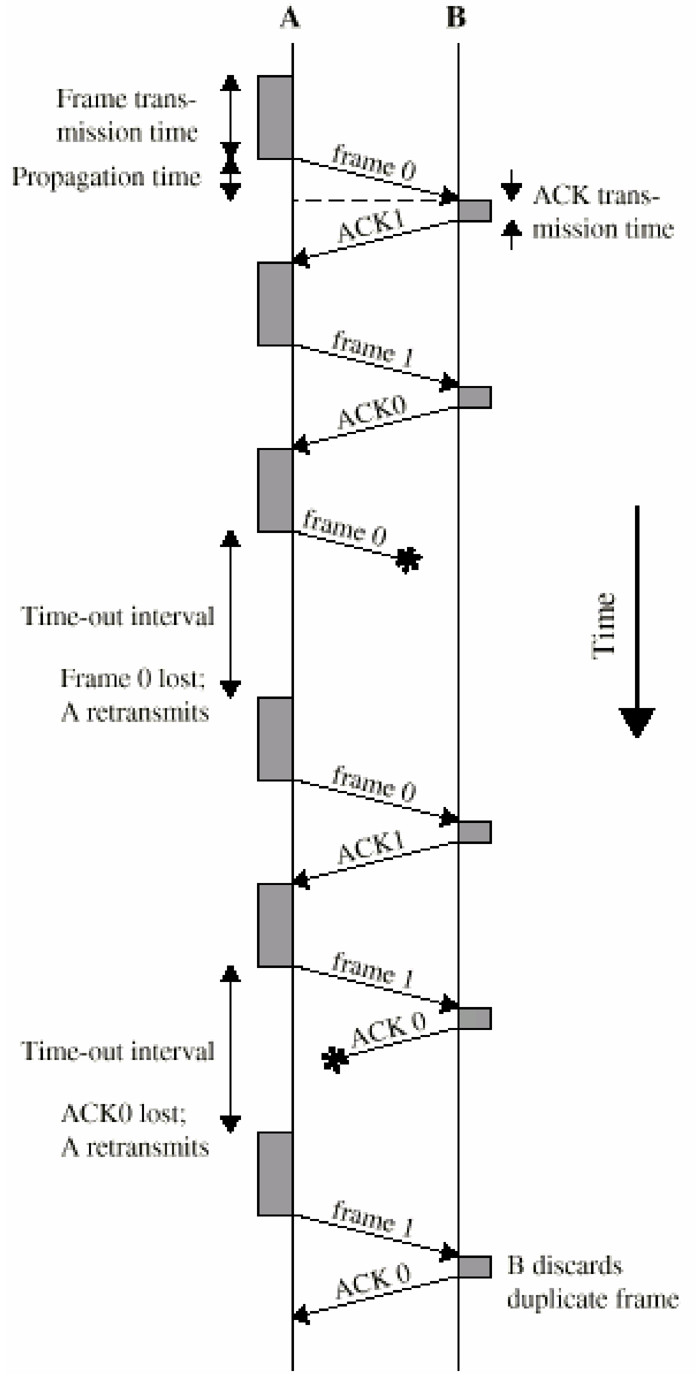
Stop-and-Wait (SW) ARQ

- ✓ **Simplest form** of flow control

1. Source transmits a frame
2. Destination receives the frame and replies with ACK
3. Source waits for ACK before sending next frame
4. Destination can stop flow by not sending ACK

- ✓ **Works well** when a message is sent in a **few <u>large frames</u>**

However, source may break up a large block of data into smaller blocks **(Fragmentation)**



Fragmentation

Curtin University

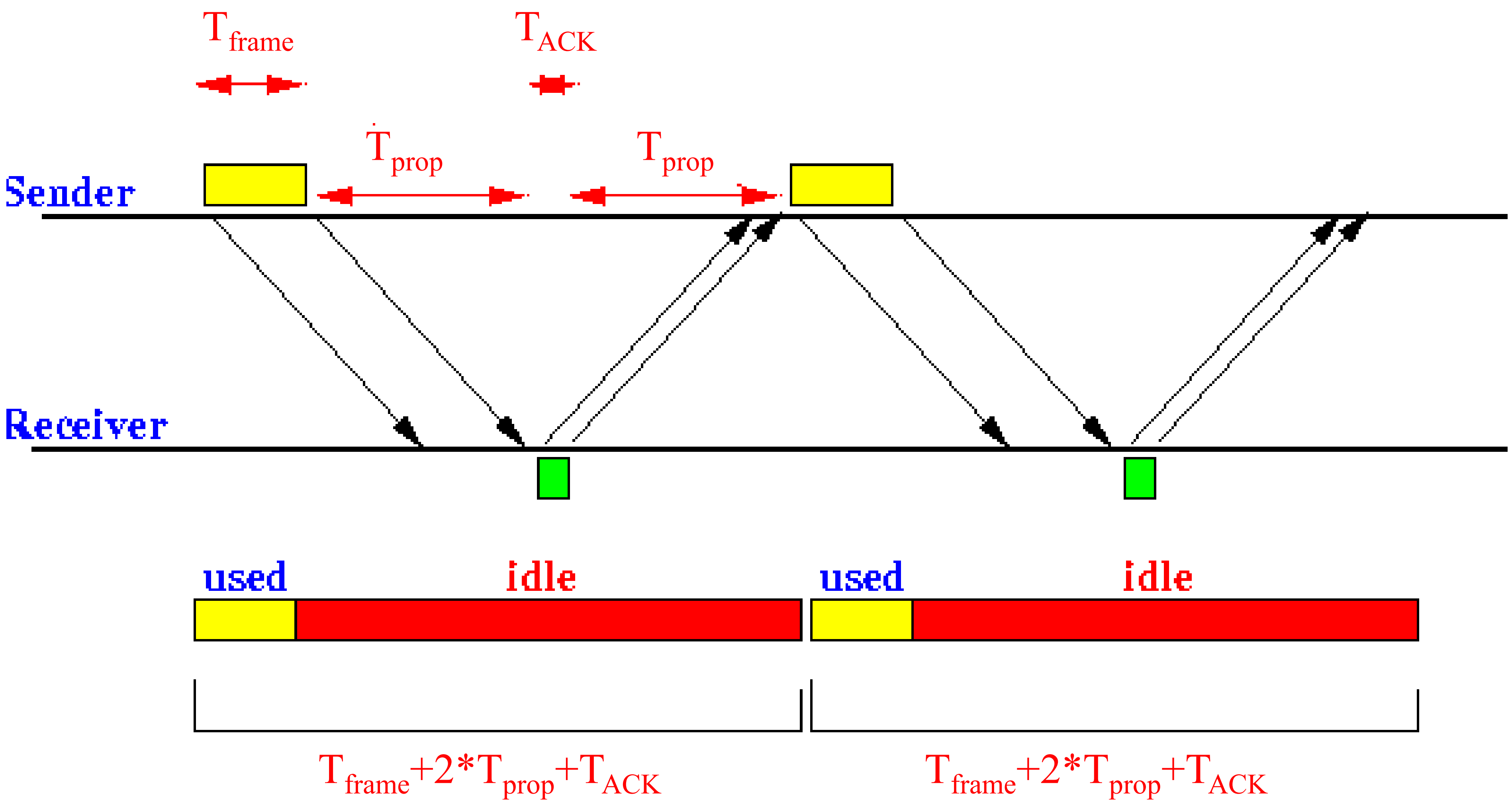
- Large block of data may be split into small frames. Reasons:
  - ✓ Limited buffer size of the receiver
  - ✓ Errors detected sooner with smaller frames (the longer the transmission, the more likely there will be an error)
  - ✓ On error, retransmission of smaller frames is needed
  - ✓ Prevents one station occupying medium for long periods

- **Stop-and-wait** becomes **inadequate** with the use of multiple frames for a single message
  - ➢ *Only one frame at a time can be in transit*

!

(SW) Link Utilization

Curtin University



(SW) Link Utilization

T_{frame} = time to transmit a frame *(time to send out all bits of the frame onto the line)*

T_{prop} = propagation time between A and B *(either direction)*

T_{ack} = time to transmit an acknowledgement

- Time to transmit one frame **T**

$$T = T_{frame} + 2 \times T_{prop} + T_{ack} + T_{processing}$$

$$T = T_{frame} + 2 \times T_{prop} + T_{ack} = 0\ (<< T_{frame}),\quad T_{processing} = 0\ (\text{negligible})$$

- Total time to send **n** frames = **nT**

(SW) Link Utilization

Curtin University

Fraction of possible throughput

$$Throughput = \frac{1}{T} = \frac{1}{T_{frame} + 2 \times T_{prop}}$$

$$Normalised\ throughput, S = \frac{\left(\dfrac{1}{T_{frame} + 2 \times T_{prop}}\right)}{\left(\dfrac{1}{T_{frame}}\right)}$$

Actual frame sending rate

Base frame sending rate

$$Therefore, S = \frac{T_{frame}}{T_{frame} + 2T_{prop}}$$

(SW) Link Utilization

Curtin University

- Let $a = \dfrac{T_{prop}}{T_{frame}}$ then, $\boxed{S = \dfrac{1}{1+2a}}$

Parameter ***a*:**

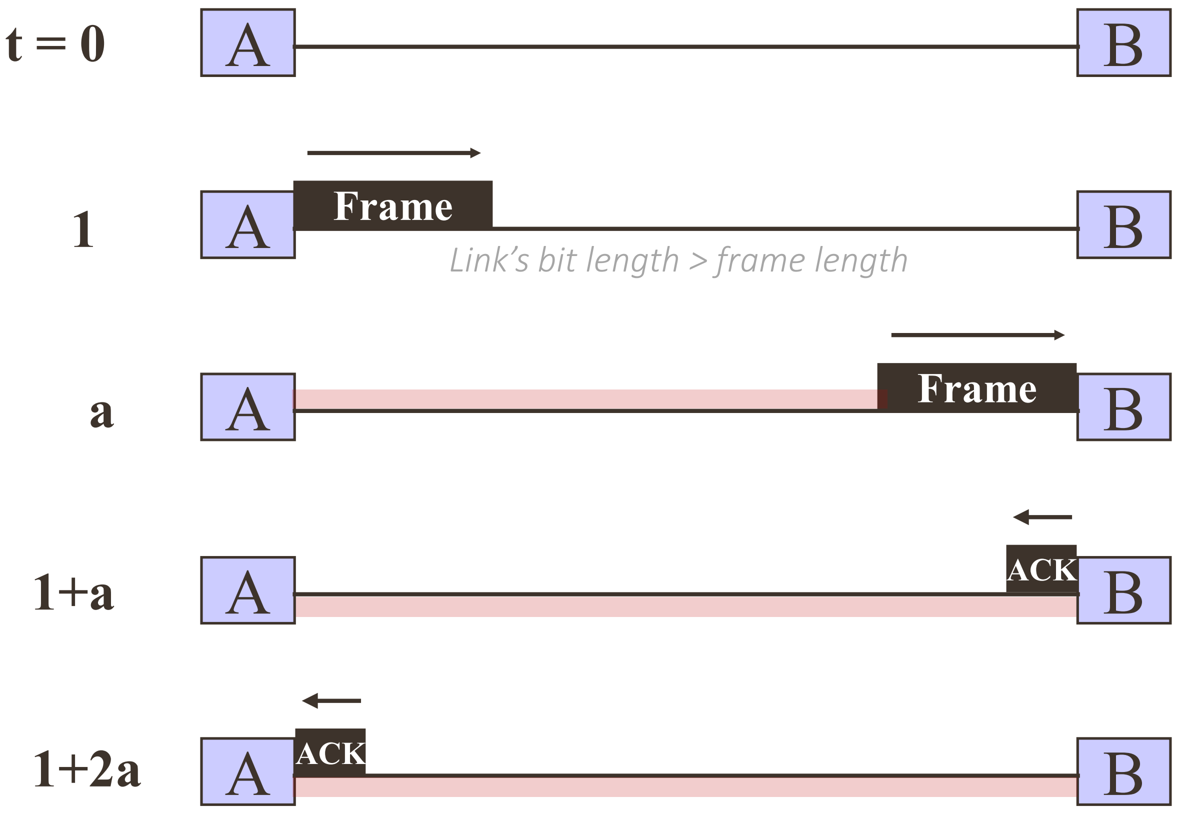
- Useful in **comparing the performance** of different link control schemes

- Provides insight into the factors affecting performance

Stop-And-Wait

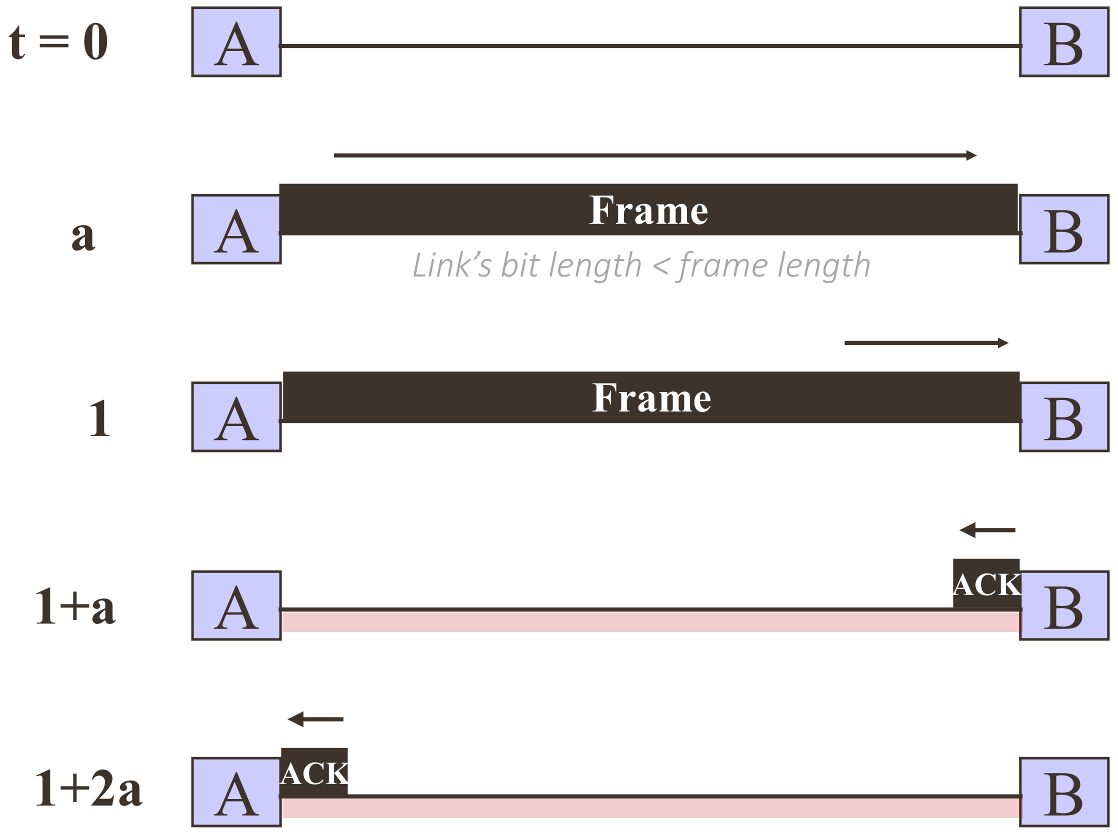
Curtin University

a >1, (T_{prop} > T_{frame})



a>1, Line is **under-utilized**

a <1, (T_{prop} < T_{frame})



a<1, Line is **better-utilized**

Sliding Window Protocols

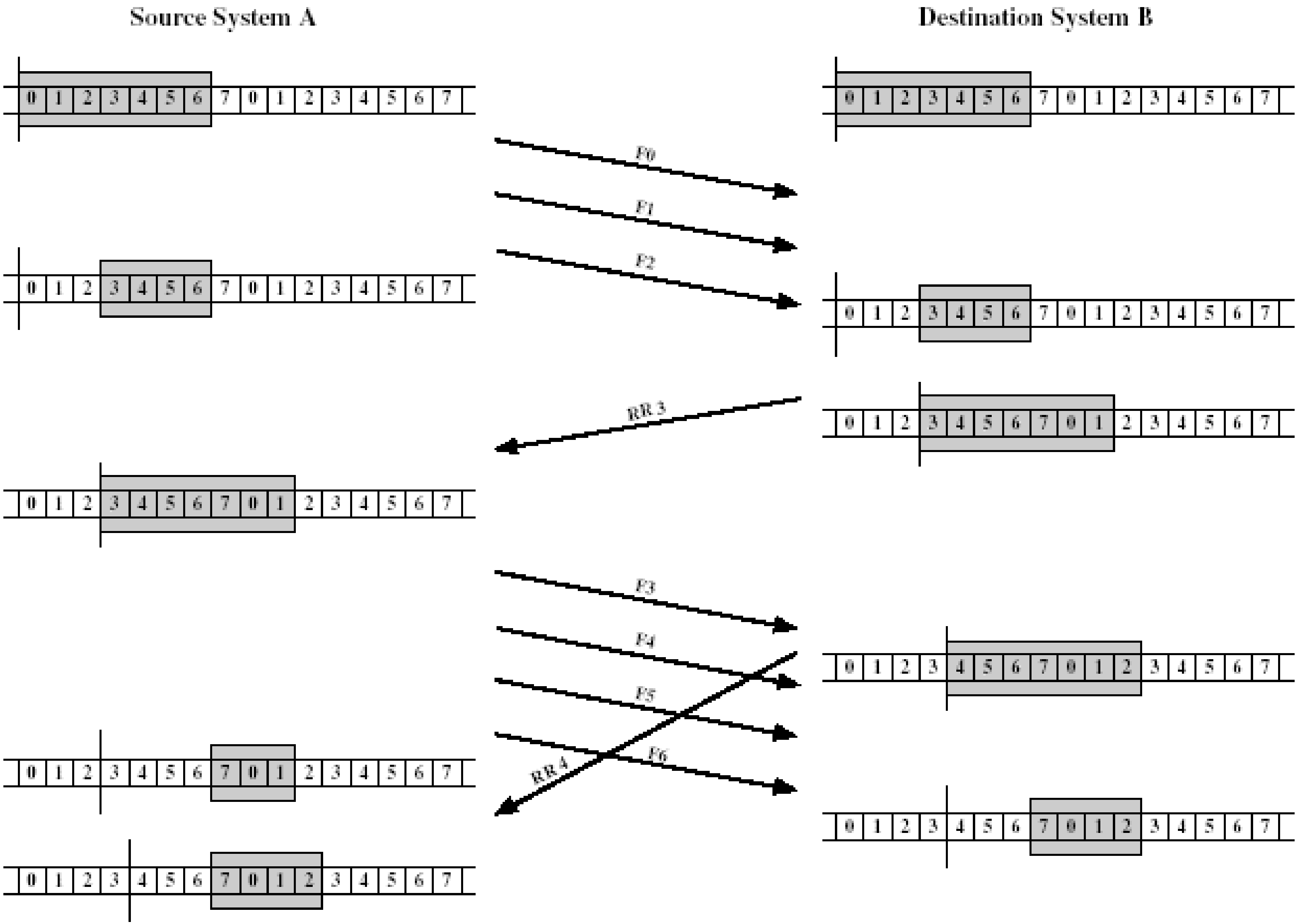
Curtin University

- Stop-and-wait is **inefficient** if $T_{prop} > T_{frame}$
- Sliding Window:
  - ✓ Allowing multiple frames to propagate from A to B *(efficient)*
  - ✓ **A** is allowed to **send n frames** without waiting for ACKs
  - ✓ **A** has a **buffer** of size n
  - ✓ **B** also has a **buffer** of size n, and **accepts n frames**
  - ✓ Each frame is **labeled with a number** modulo some maximum

1. **Go Back N ARQ**
2. **Selective Reject ARQ**

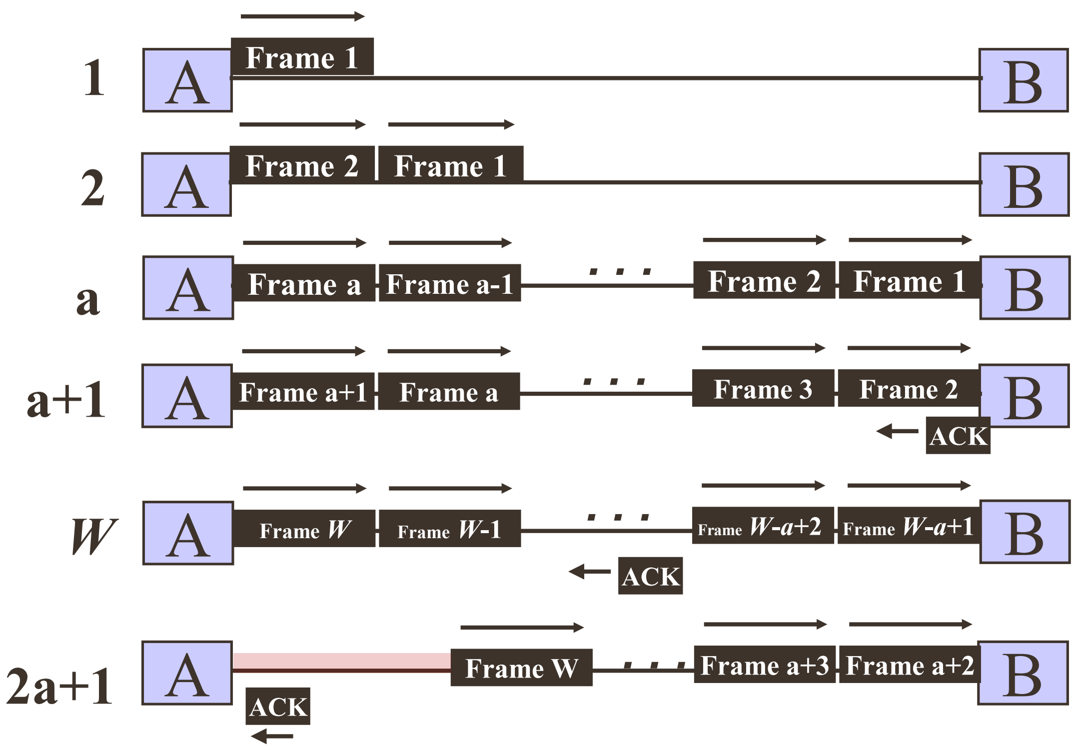
Curtin University

Sliding Window



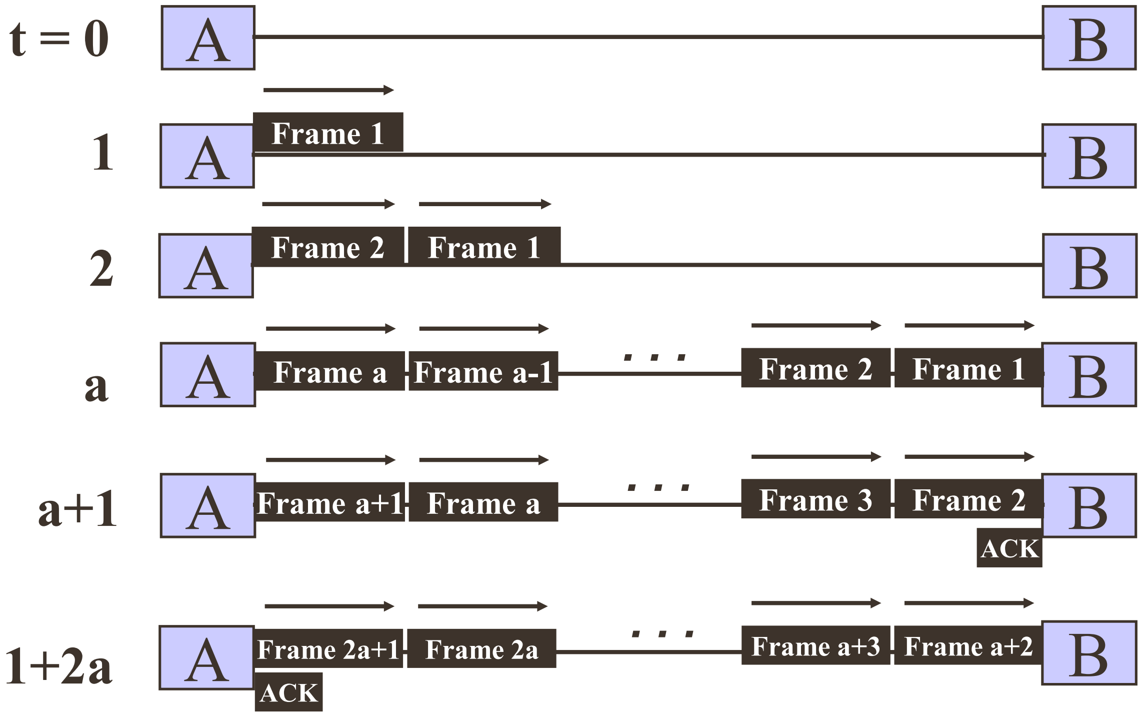
Sliding Window

W < 2a + 1



W < 2a+1, Line is **under-utilized**

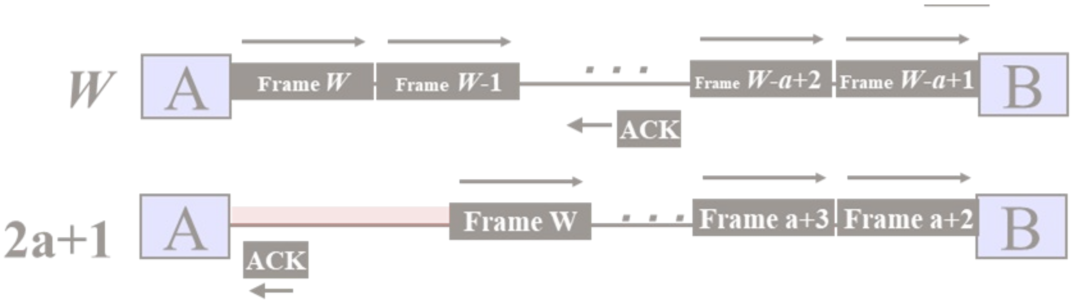
W ≥ 2a + 1



W ≥ 2a+1, Line is **fully-utilized**

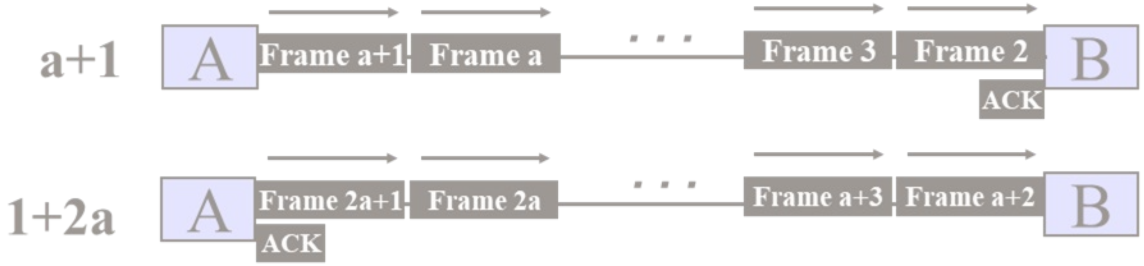
Curtin University

W < 2a + 1



A exhausts its window at **t = W** and cannot send any more frames **until t = 2a+1**

W ≥ 2a + 1



ACK reaches A **before** A has **exhausted** its window. A can transmit continuously

$$S = \begin{cases} 1, & \boldsymbol{W \geq 2a + 1} \\ \dfrac{W}{(2a+1)}, & \boldsymbol{W < 2a + 1} \end{cases}$$

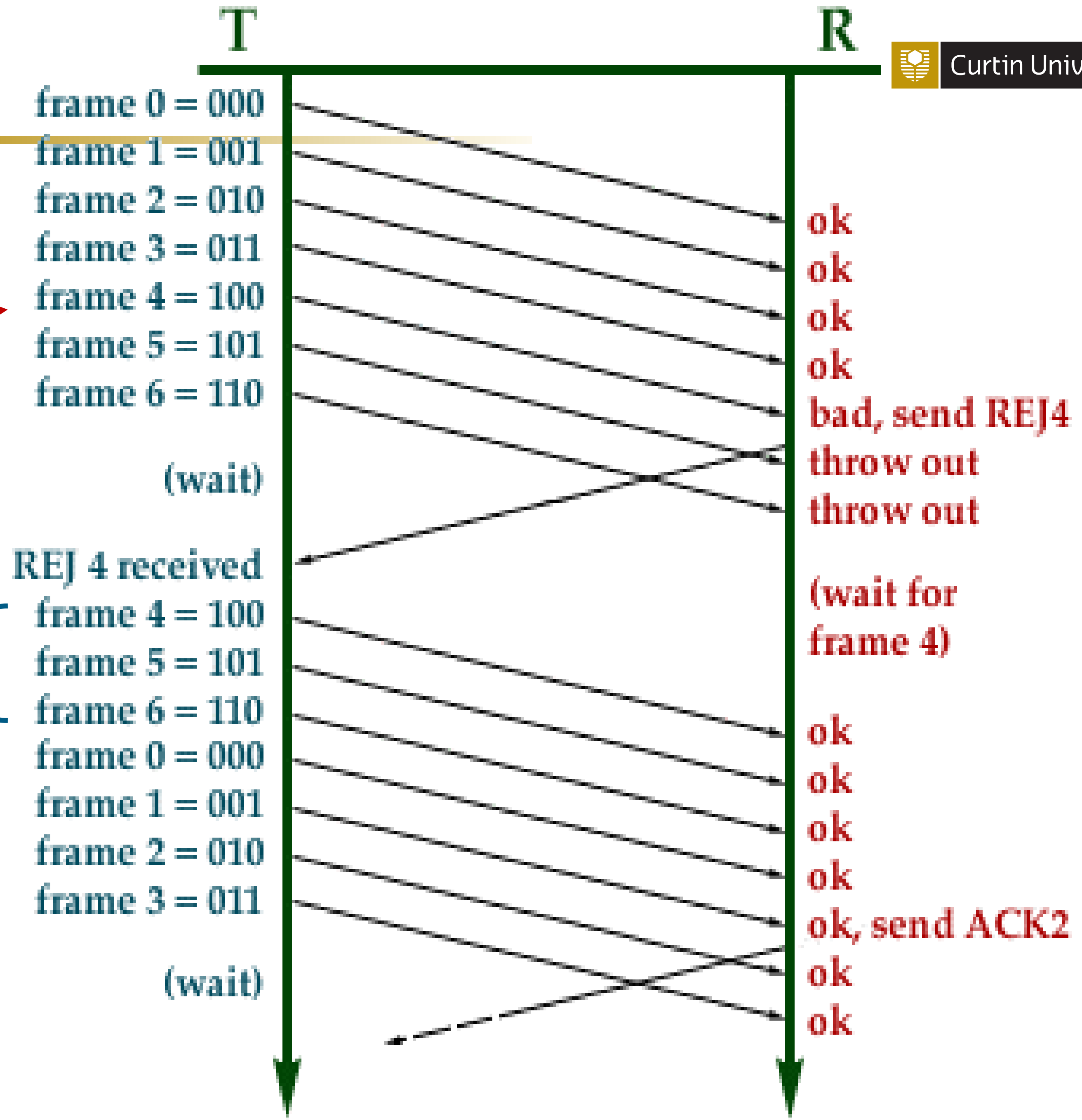
Sliding Window Link-Utilization

Curtin University

Go-Back-N ARQ

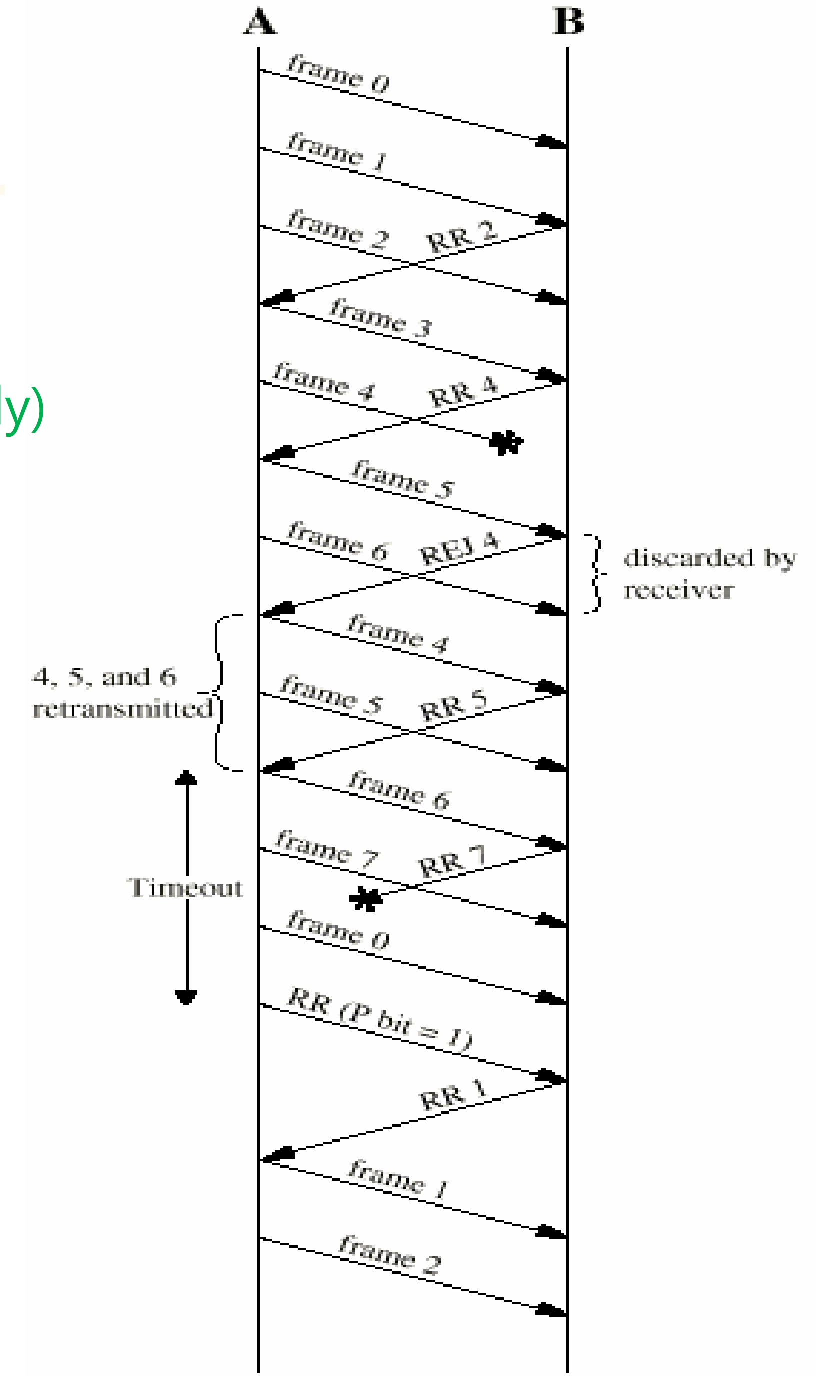
Go-Back-N Frames

Retransmits frame ***k*** (=4) and all **succeeding frames** that were transmitted *(say N)* in the interim period.



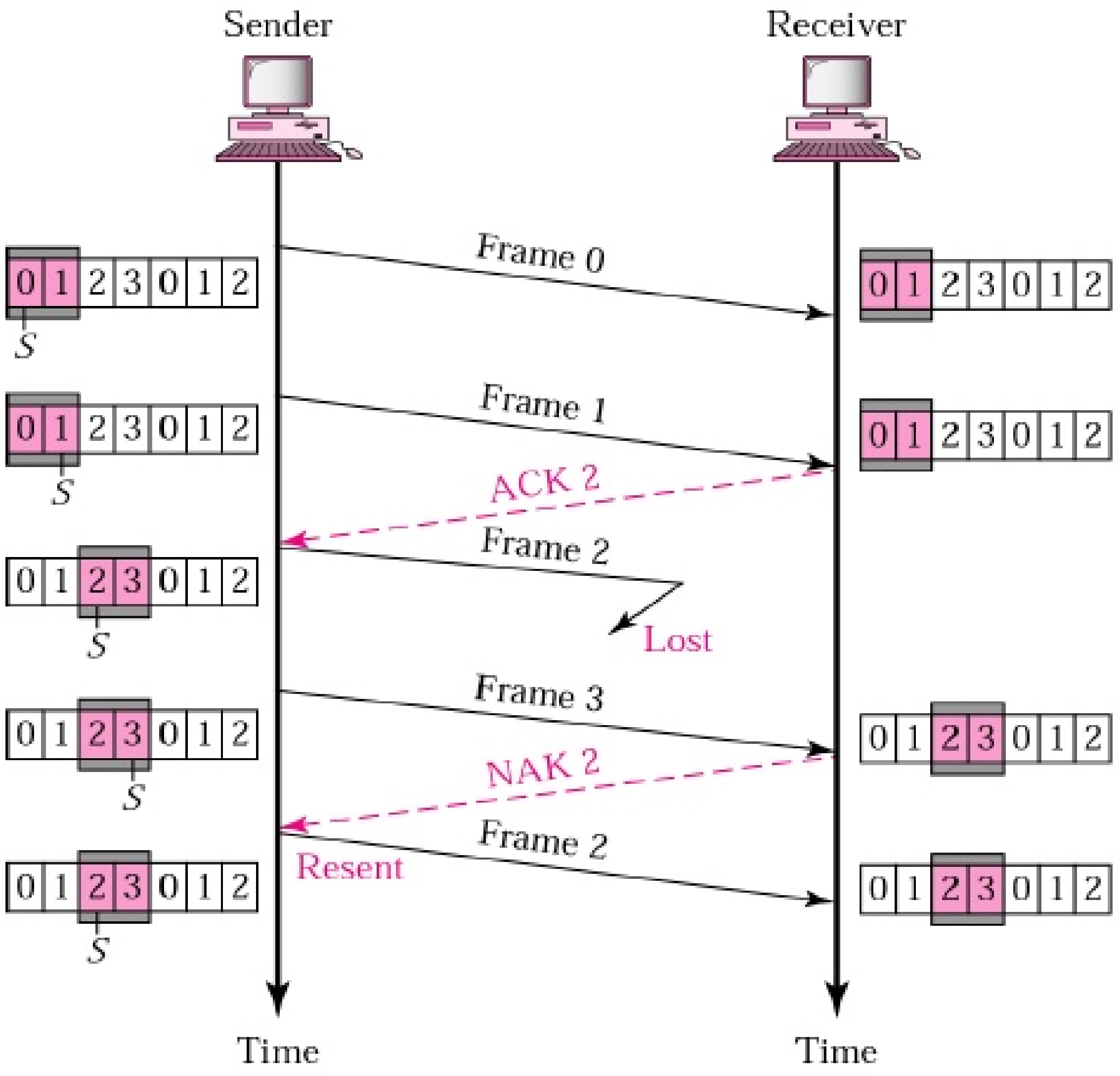
Go-Back-N ARQ – cont.

- Station A sends frames to station B.
- Station B ACKs incoming frames with an **RR** (Receive Ready)
- Suppose Frame 4 is damaged.
- Frames 5 and 6 are received **out-of-order** and are discarded by B
- When frame 5 arrives, B sends a **REJ 4**
- When the **REJ** to frame 4 is received, frames 4, 5 and 6 must be retransmitted
- Transmitter must keep a copy of all un-acked frames



Selective-Reject ARQ

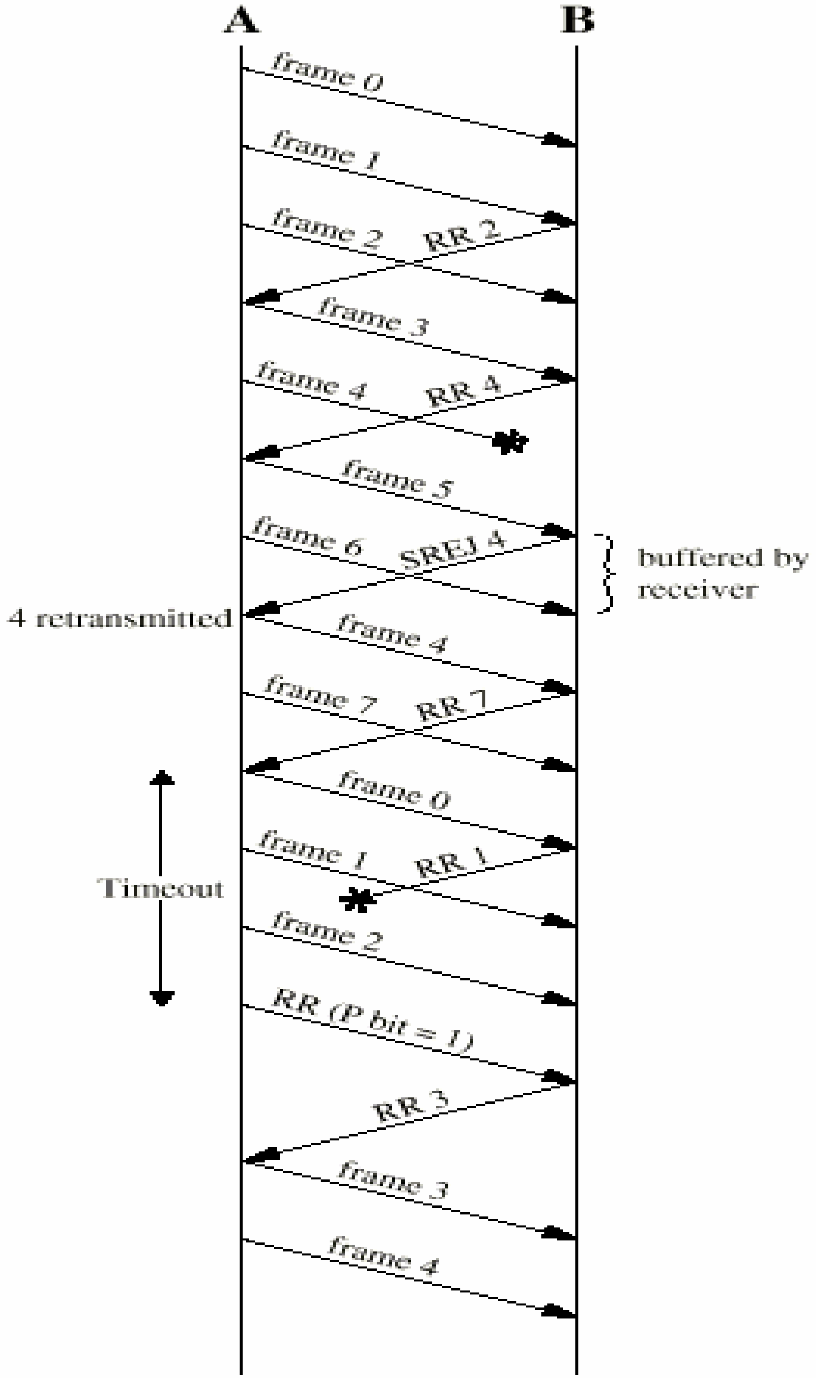
Curtin University



Selective-Reject ARQ

- The only frames retransmitted are those
  - that receive **NAK** (or **SREJ**)
  - that time out
- B sends **NAK *i*** if frame ***i*** is in error or lost
- B is required to **buffer frames** in order **until the correct frame arrives**

Complex logic required to send out-of-sequence frames and insert frames in appropriate places



Curtin University

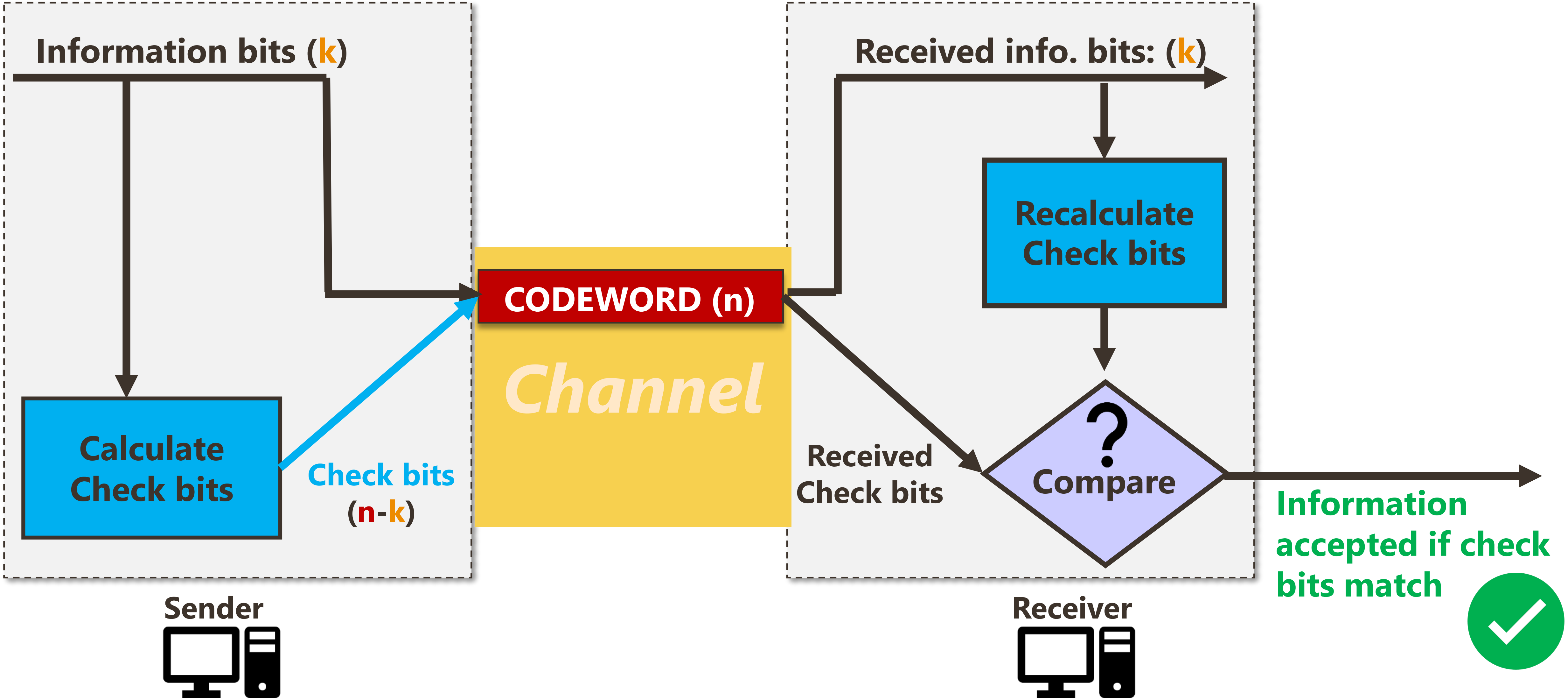
LLC Sub-Layer

Error Detection and Correction

- Parity Check
- 2D Parity
- Checksum
- CRC

Error Detection Pipeline

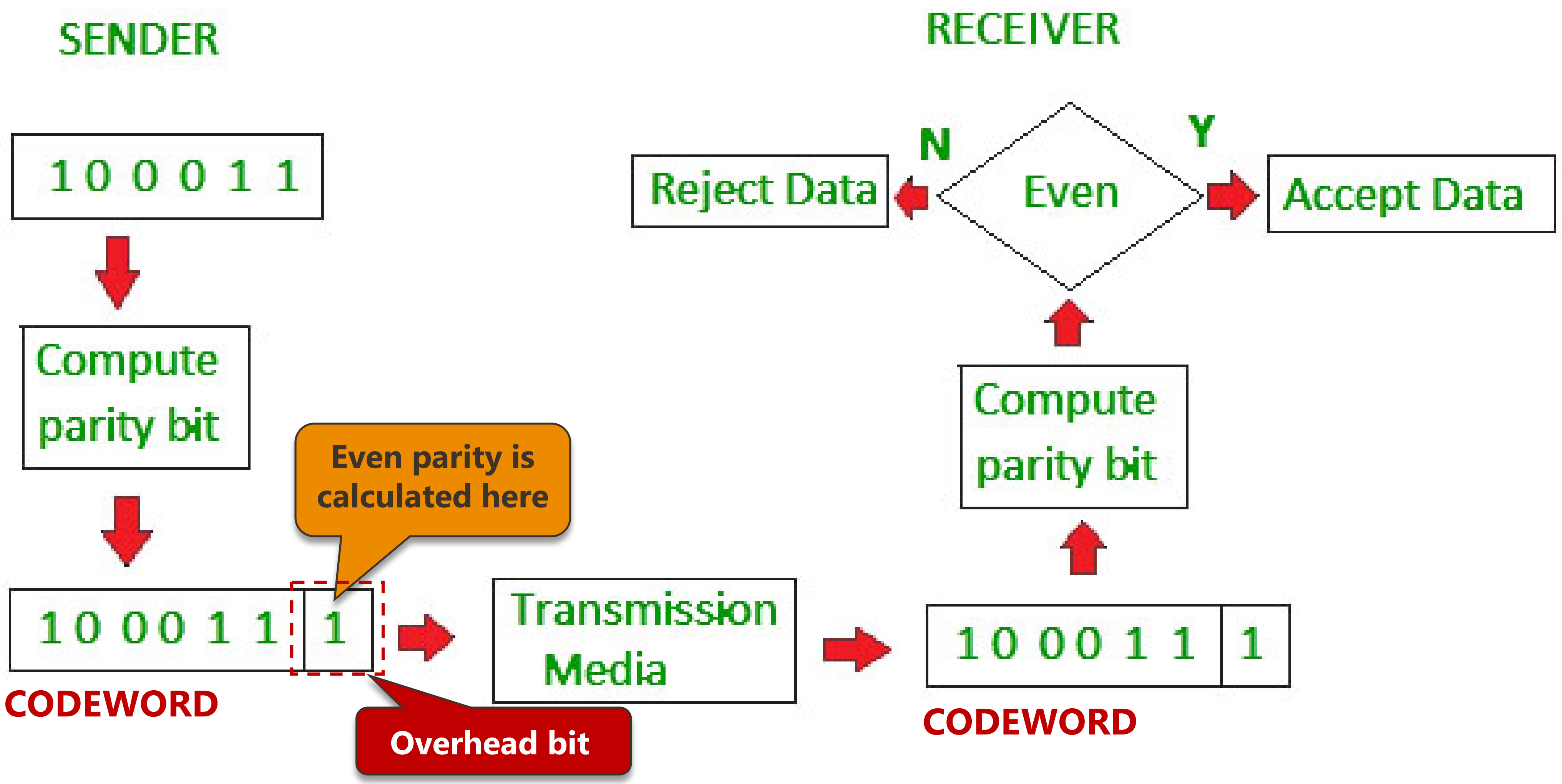
Curtin University



Information accepted if check bits match

Parity Check (Even Bit/Odd Bit)

Curtin University



2D Parity Check

Original Data

10011001	11100010	00100100	10000100

Row parities

1 0 0 1 1 0 0 1	0
1 1 1 0 0 0 1 0	0
0 0 1 0 0 1 0 0	0
1 0 0 0 0 1 0 0	0
1 1 0 1 1 0 1 1	0

Even parity is calculated here

Column parities

Overhead bits

CODEWORD =

100110010	111000100	001001000	100001000	110110110

Data to be sent

Curtin University

Checksum

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

k=4, m=8

Sender

```
1          1 0 0 1 1 0 0 1
2          1 1 1 0 0 0 1 0
          ----------------
          1 0 1 1 1 1 0 1 1
                          1
          ----------------
           0 1 1 1 1 1 0 0
3          0 0 1 0 0 1 0 0
          ----------------
           1 0 1 0 0 0 0 0
4          1 0 0 0 0 1 0 0
          ----------------
          1 0 0 1 0 0 1 0 0
                          1
          ----------------
Sum:       0 0 1 0 0 1 0 1
CheckSum:  1 1 0 1 1 0 1 0
```

Reciever

```
1             1 0 0 1 1 0 0 1
2             1 1 1 0 0 0 1 0
             ----------------
             1 0 1 1 1 1 0 1 1
                             1
             ----------------
              0 1 1 1 1 1 0 0
3             0 0 1 0 0 1 0 0
             ----------------
              1 0 1 0 0 0 0 0
4             1 0 0 0 0 1 0 0
             ----------------
             1 0 0 1 0 0 1 0 0
                             1
             ----------------
              0 0 1 0 0 1 0 1
              1 1 0 1 1 0 1 0
             ----------------
Sum:          1 1 1 1 1 1 1 1
Complement:   0 0 0 0 0 0 0 0
```

Conclusion: Accept Data

CODEWORD = [<original_data><checksum>]

Overhead bits

Checksum

Method 02

▪ Checksum Calculation Steps (Sender)

1. $x = b_0 + b_1 + b_2 + \ldots + b_{L-1} \text{ modulo } (2^{\langle block\_size \rangle} - 1)$
2. **checksum** $= -x \text{ modulo } (2^{\langle block\_size \rangle} - 1)$

▪ Checksum Validation Step (Receiver)

3. $0 = (b_0 + b_1 + b_2 + \ldots + b_{L-1} + \textbf{checksum}) \text{ modulo } (2^{\langle block\_size \rangle} - 1)$

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

i.e. @ Sender

b1 = 10011001 = 153

b2 = 11100010 = 226

b3 = 00100100 = 36

B4 = 10000100 = 132

x = 153 + 226 + 36 + 132 modulo (2^8 -1) = 37

checksum = -x modulo (2^8 -1) = 218 = **11011010**

i.e. @ Receiver

= (153+226+36+132+218) mod (2^8 -1)

= 0

Cyclic Redundancy Check (CRC)

MOST POPULAR

Curtin University

- a.k.a **Polynomial Codes**

- **In this technique,**

1. Treat the entire string of data bits as a **single number D(x)**
2. Divide this number by a pre-defined polynomial, called the **generator polynomial G(x),** and **append the remainder R(x)** to the data string
3. The **receiver performs the same division** and obtain the remainder
4. A **non-zero remainder indicates an error**

CRC – cont.

Curtin University

- **Polynomials** instead of vectors for codewords

$$\texttt{D(x)} = \begin{array}{ccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ x^6 & + \cancel{x^5} & + x^4 & + \cancel{x^3} & + \cancel{x^2} & + \cancel{x^1} & + \cancel{x^0} \end{array} = x^6 + x^4$$

- **Polynomial arithmetic** instead of checksums
  - Follows laws of ordinary algebra, except **addition** is done in modulo 2:

$$\boxed{x^a + x^a = 0\ (similar\ to\ 1\ XOR\ 1 = 0)}$$

- Mostly used for error detection but a basis for error-correction methods

1 $D(x) = 1010000 = x^6 + x^4$

- info bits (k = 7)

2 $G(x) = 1001 = x^3 + 1$

- Codeword length

  n = degree of G(x) + k = 3 + 7 = **10**

3 **Pre-encoding = $x^3 D(x) = D'(x)$**

4 **Perform polynomial (mod 2) division ($D'(x)$ / $G(x)$), obtain remainder R(x)**

5 **Codeword = C(x) = $D'(x) + R(x)$**

1 D(x) = 1010000 original message

@ means X-OR

2 Generator polynomial

x^3+1

$1.x^3+0.x^2+0.x^1+1.x^0$

CRC generator

G(x)= 1001 4-bit

If CRC generator is of n bit then append (n-1) zeros in the end of original message

Sender

```
G(x)= 1001 | 1010000000  =D'(x) 3
          4 @1001
             0011000000
              @1001
               0101000
               @1001
                0011000
                 @1001
                   01010
                   @1001
                    0011 =R(x) 4
```

Message to be transmitted

```
  1010000000
5      +011
Codeword = 1010000011
```

Codeword

```
1001 | 1010000011
      @1001
       0011000011
        @1001
         01010011
         @1001
          0011011
           @1001
            01001
            @1001
             0000
```

Receiver

Zero means data is accepted

Choice of Generator Polynormal G(x)

Curtin University

It has been proven that a strong G(x) can detect:

- All **single-bit errors**
- Almost all **double-bit errors**, if G(x) has a factor with at least three terms
- Any **odd number of errors**, if G(x) has the factor x +1
- All **bursts of** up to m **errors**, if G(x) is of degree m
- **Longer burst errors** with probability $1 - 2^{-m}$, if bursts are randomly distributed

Standard Generator Polynomials

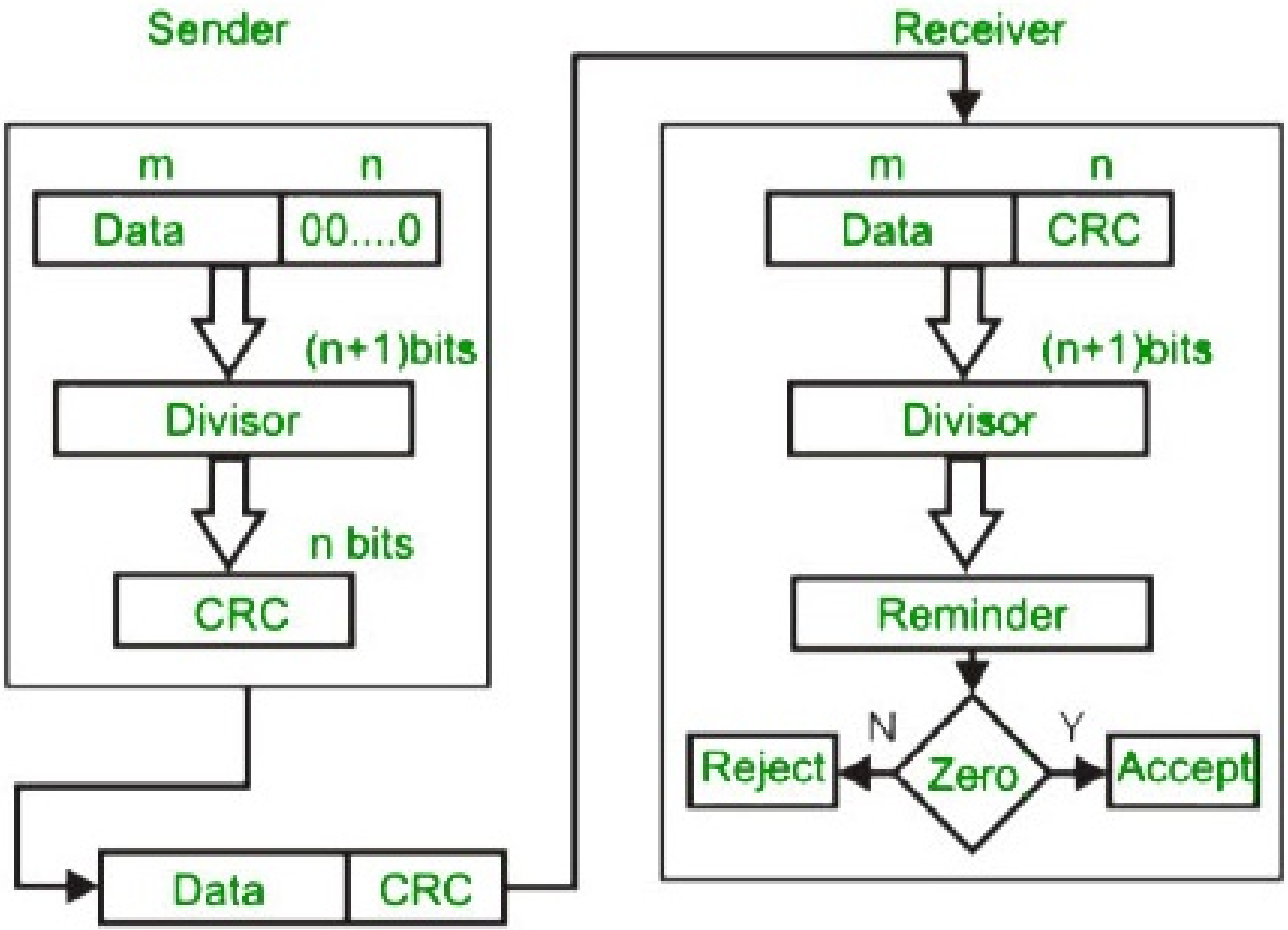
Curtin University

- **CRC-8:** $\boxed{x^8 + x^2 + x + 1}$
  - ✓ ATM
- **CRC-16:** $\boxed{x^{16} + x^{15} + x^2 + 1}$
  - ✓ HDLC, XMODEM, V.41
- **CRC-32:** $\boxed{x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1}$
  - ✓ IEEE 802, TCP/IP Model, **V.42**

V. 42 permits computer modems to work with both digital and analog phone lines

CRC Summary

Curtin University



Curtin University

MAC Sub-Layer

Framing

- Application protocols: **HDLC**, **PPP**
- **HDLC** – Framing with Bit Stuffing
- **PPP** – Framing with Byte Stuffing

Applications of HDLC & PPP

Curtin University

- **HDLC (High-level Data Link Control)**
  - Bit-oriented
  - LAPD (**L**ink **A**ccess **P**rotocol **D**-Channel) in ISDN
  - LAPD (**L**ink **A**ccess **P**rocedure for **M**odems) in cellular telephone signaling

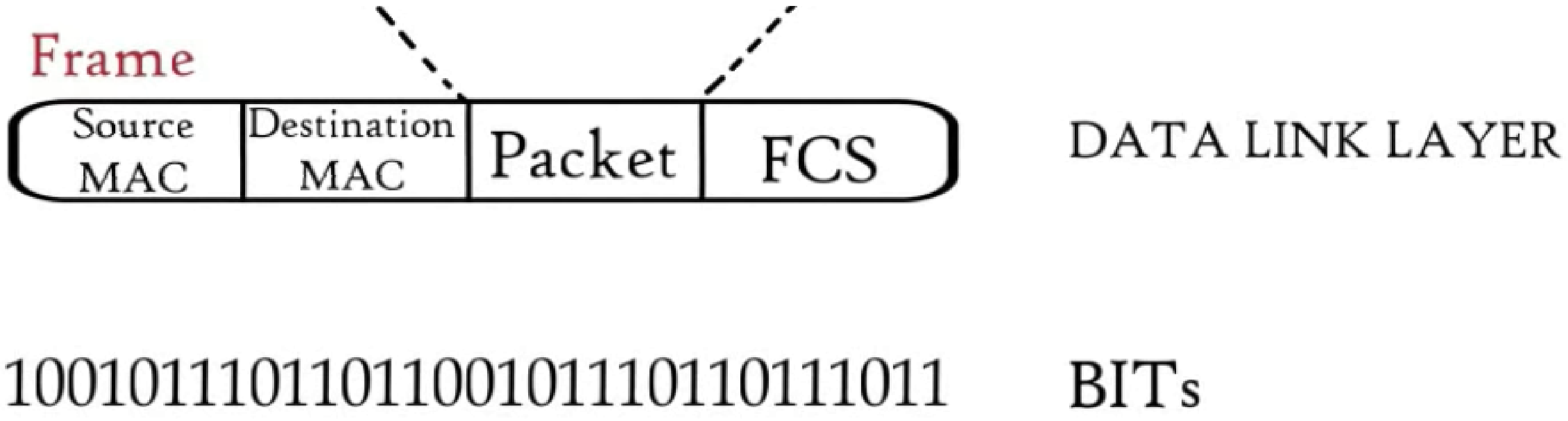
- **PPP (Point to Point Protocol)**
  - Telephone Modem Links (30 Kbps)
  - Packet over SONET (600-10000Mbps) - **S**ynchronous **O**ptical **Net**work
    - IP->PPP->SONETAPD (Link Access Protocol D-Channel) in ISDN
  - PPP over shared links
    - PPP over Ethernet (RFC 2516)
    - Used over DSL

used to transmit a large amount of data over relatively large distances using **optical fibre**

Framing / De-framing

Curtin University

- Map stream of physical layer **bits** into **frames** (vice-versa)
- **Frame boundaries can be determined using:**
  - ✓ Character Counts
  - ✓ Control Characters
  - ✓ **Flags**
  - ✓ CRC Checks



Framing & Bit Stuffing

Curtin University

- **HDLC** uses bit-stuffing

HDLC FRAME

Flag	Address	Control	Information	FCS	Flag

Flag: 01111110

Bit stuffing prevents occurrence of **flag** in information

METHOD

- **Transmitter** inserts extra 0 after each consecutive five 1s inside the frame
- **Receiver** checks for five consecutive 1s
  1. if next bit = 0, it is removed
  2. if next two bits are 10, then flag is detected
  3. If next two bits are 11, then frame has errors

Curtin University

Data to be sent: 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0

After stuffing and framing

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

HDLC Frame Bit-Stuffing

Data received: *0 1 1 1 1 1 1 0* 0 0 0 1 1 1 0 1 1 1 1 1 ~~0~~ 1 1 1 1 1 ~~0~~ 1 1 0 *0 1 1 1 1 1 1 0*

After destuffing and framing

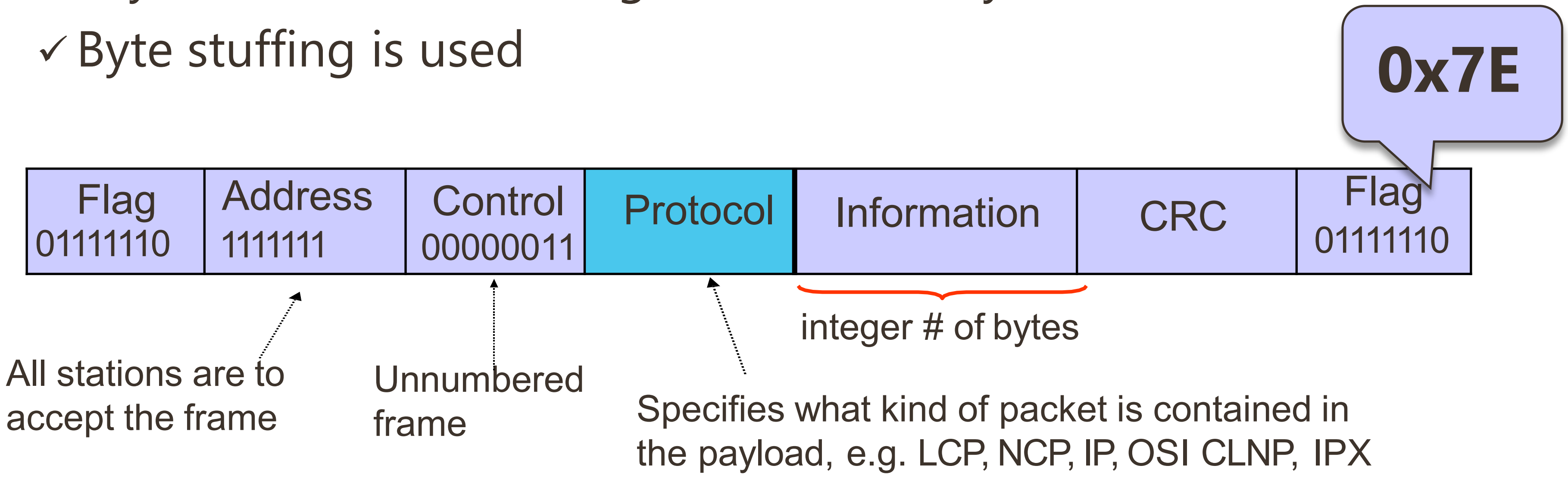
* 0 0 0 1 1 1 0 1 1 1 1 1 - 1 1 1 1 1 - 1 1 0 *

HDLC Frame Bit-destuffing

Framing & Byte Stuffing

Curtin University

- **PPP** (Point-to-Point Protocol) is **character-oriented**
- **PPP** Frame ~= **HDLC** Frame, **except**
  - ✓ Protocol type field
  - ✓ Payload contains an integer number of bytes
  - ✓ Byte stuffing is used



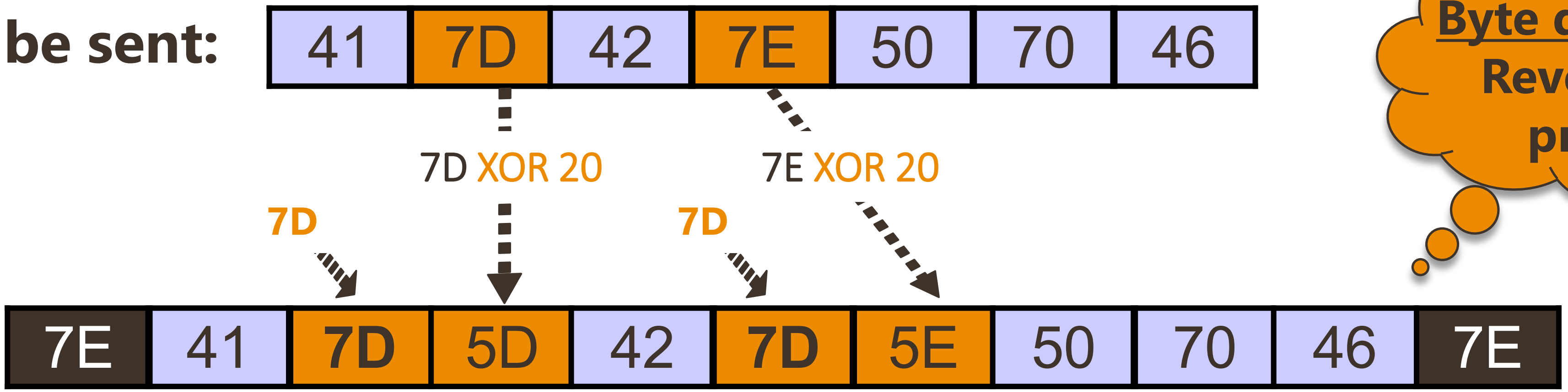
PPP FRAME

Byte Stuffing

Use **control escape (7D)** in front of any 7E in the payload to indicate it's not a flag.

Any occurrence of **flag(7E)** or **control escape (7D)** inside the frame is replaced with 0x7D(01111101) followed by them XORed with 0x20 (00100000)

Data to be sent:

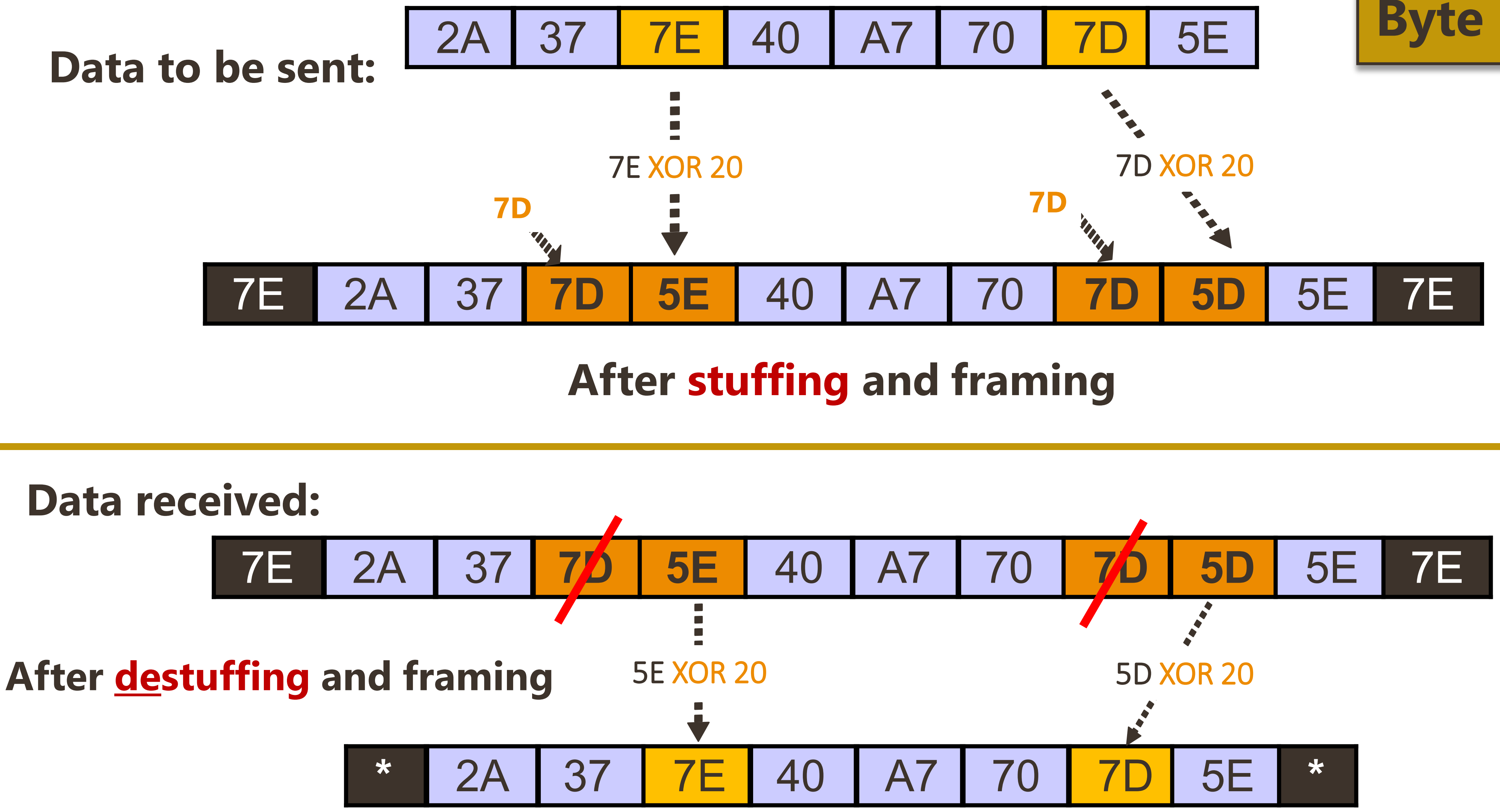


After stuffing and framing

Byte destuffing
Reverse the process

- **Problems with PPP Byte Stuffing !**
  - Size of frame varies unpredictably due to byte insertion
  - Malicious users can inflate bandwidth by inserting 7D & 7E

Byte Stuffing



MAC Sub-Layer

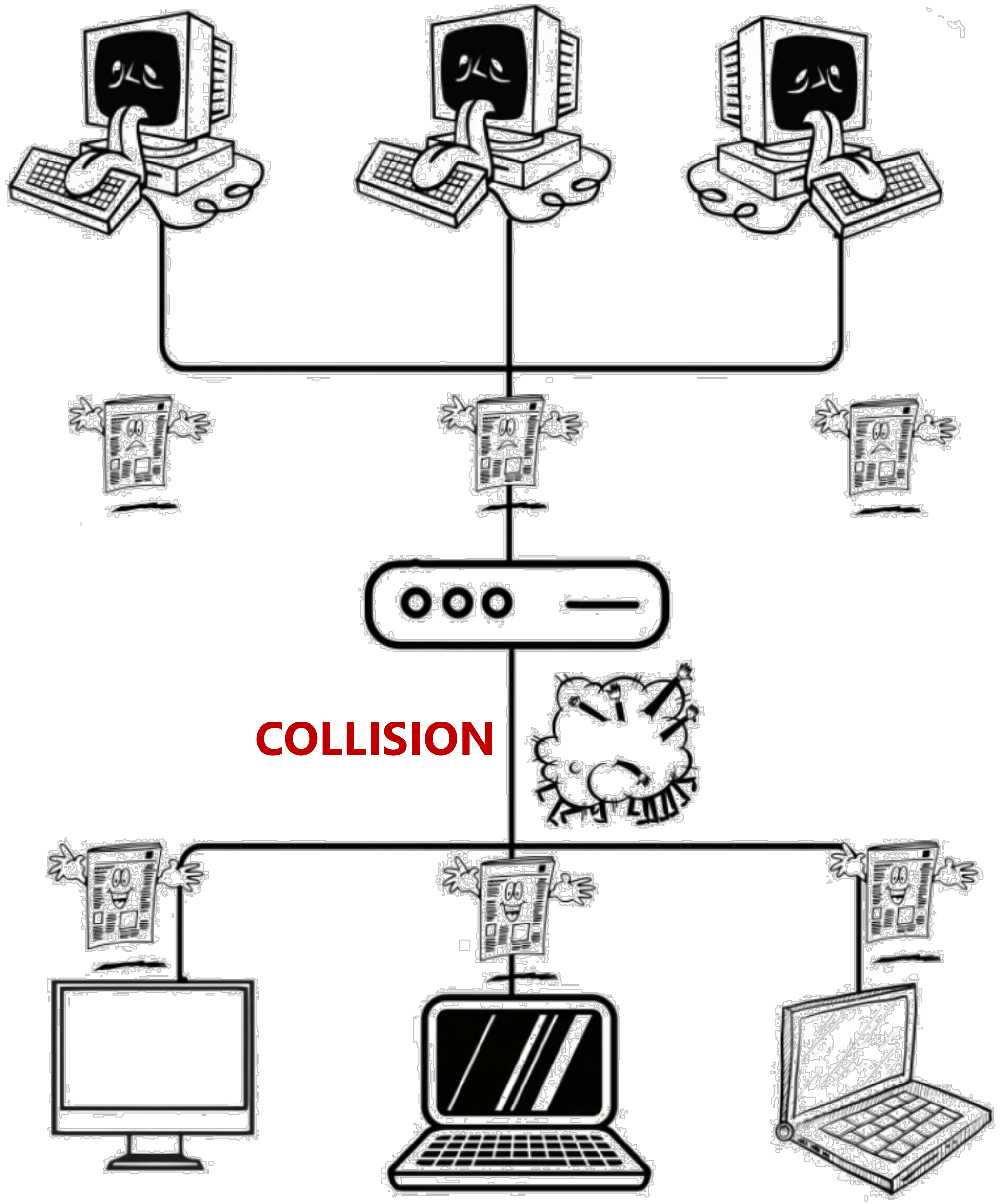
Curtin University

Media Access Control/Sharing

- Scheduling Methods
  - Polling
  - Token-passing
- Random Access Methods
  - ALOHA
  - CSMA, CSMA Options
  - CSMA/CD
- Media Sharing Example: Wireless LAN

Curtin University

What happens when everybody start sending data ?



Medium Access Control

Curtin University

Static Channelization

- Partition Medium
- Dedicated Allocation to users
- Satellite Transmission
- Cellular Telephone

Dynamic Medium Access Control

Scheduling

- Polling: Take turns
- Request for slot in transmission schedule
- Token Ring
- Wireless LANs

Random Access

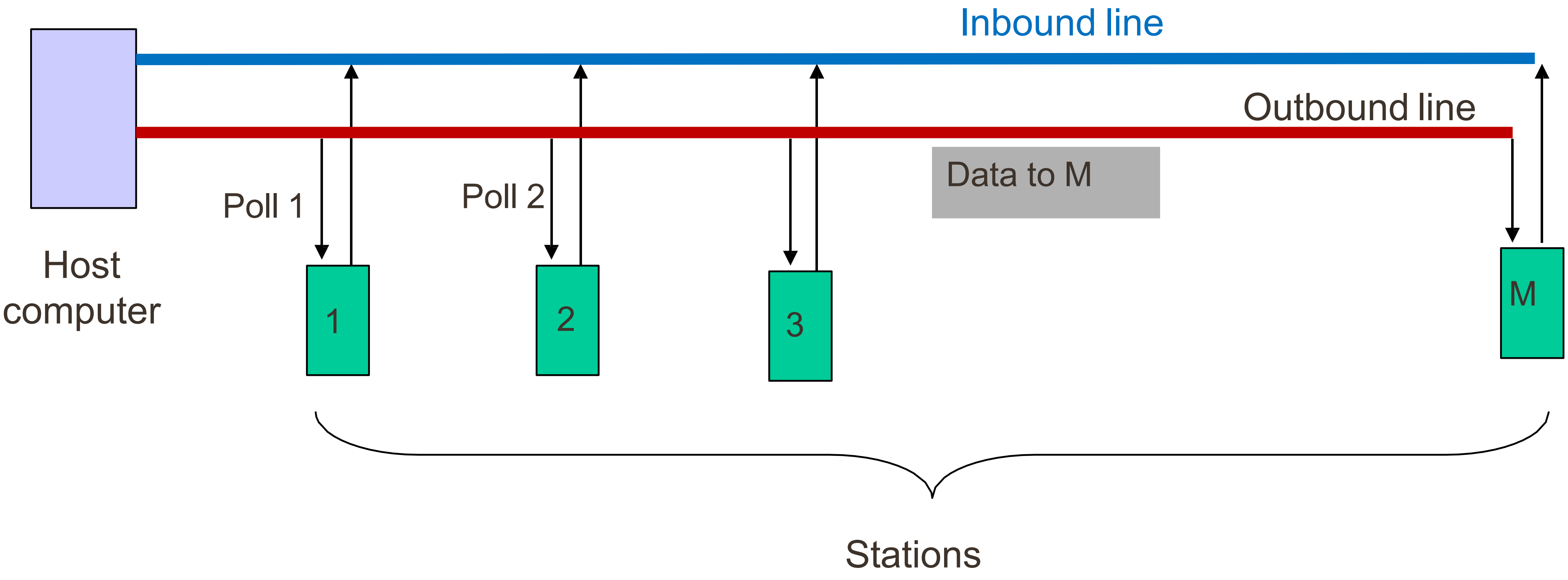
- Loose coordination
- Send, wait, retry if necessary
- Aloha
- Ethernet

Selecting a Medium Access Control Method

- **Applications**
  - What type of traffic?
  - Voice streams? Steady traffic, low delay/jitter
  - Data? Short messages? Web page downloads?
  - Enterprise or Consumer market? Reliability, cost
- **Scale**
  - How much traffic can be carried?
  - How many users can be supported?
- **Current Examples:**
  - Design MAC to provide wireless DSL-equivalent access to rural communities
  - Design MAC to provide Wireless-LAN-equivalent access to mobile users (user in car travelling at 130 km/h)

Scheduling: Polling

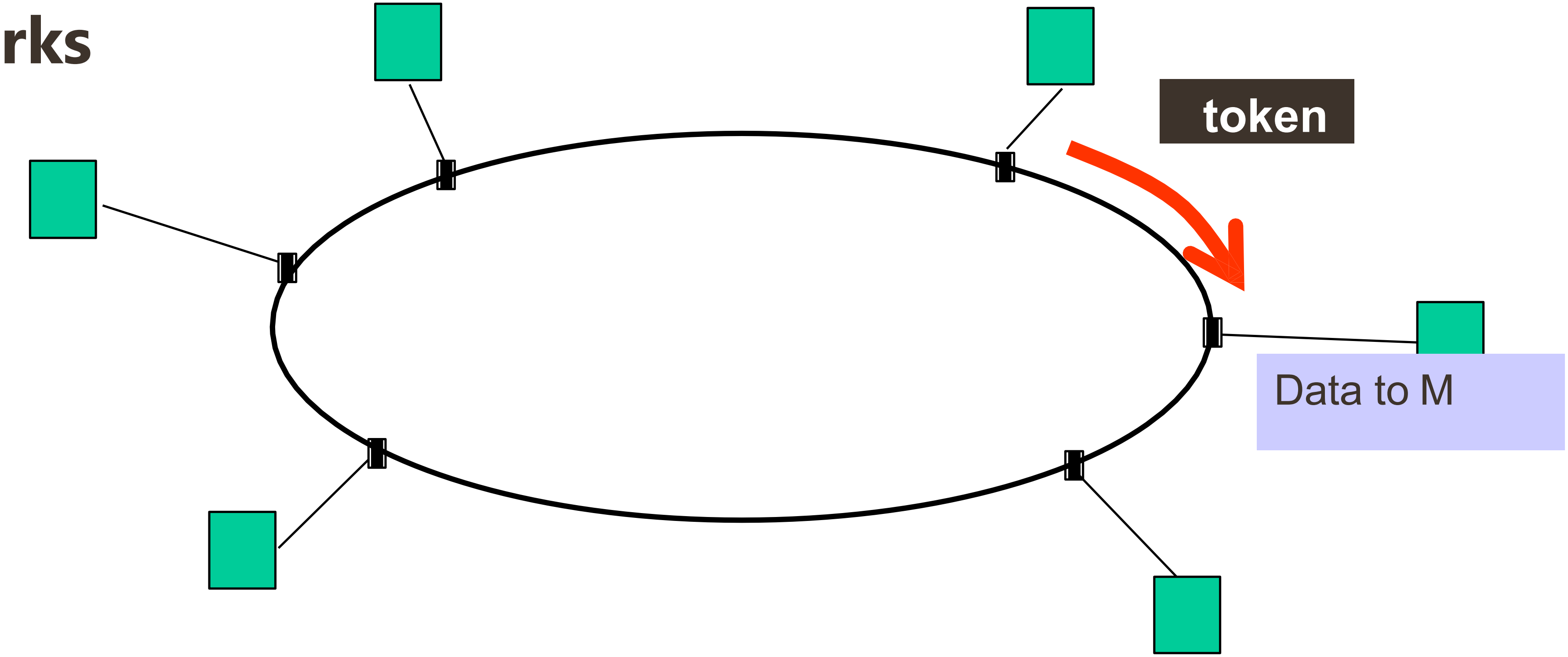
Curtin University



Scheduling: Token-Passing

Curtin University

Ring Networks

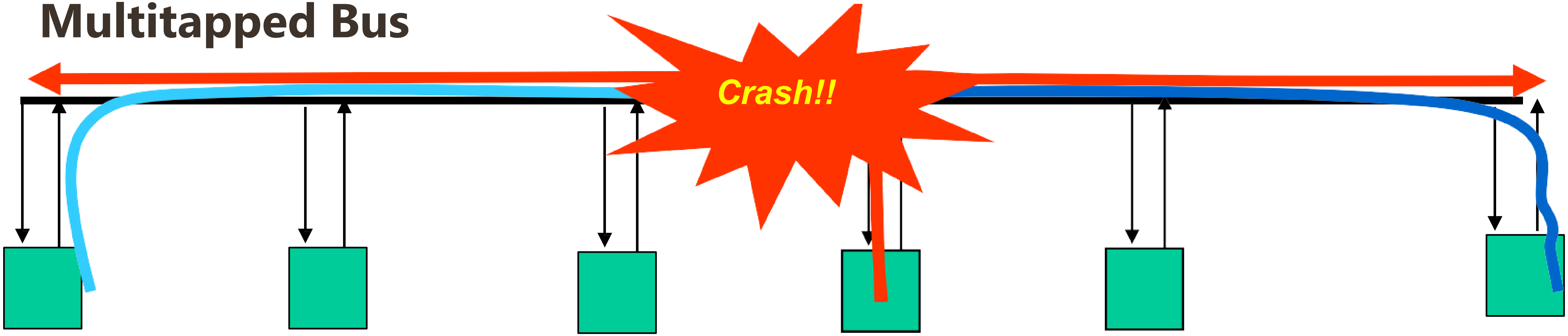


Station that holds token transmits into ring

Random Access

Curtin University

Multitapped Bus



Transmit when ready

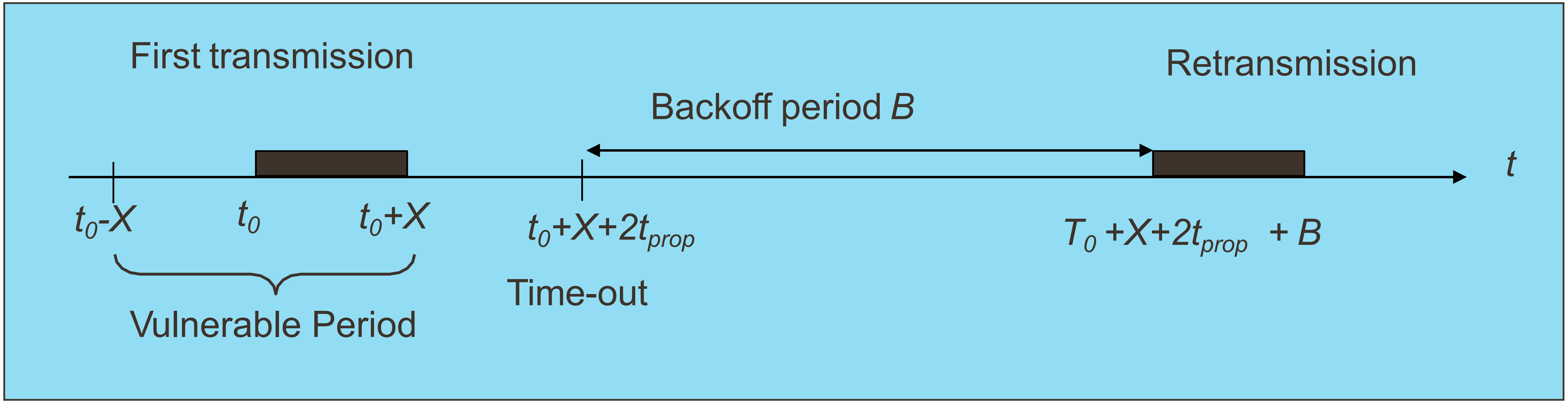
Collisions can occur;

Need a retransmission strategy

Random Access: ALOHA

Curtin University

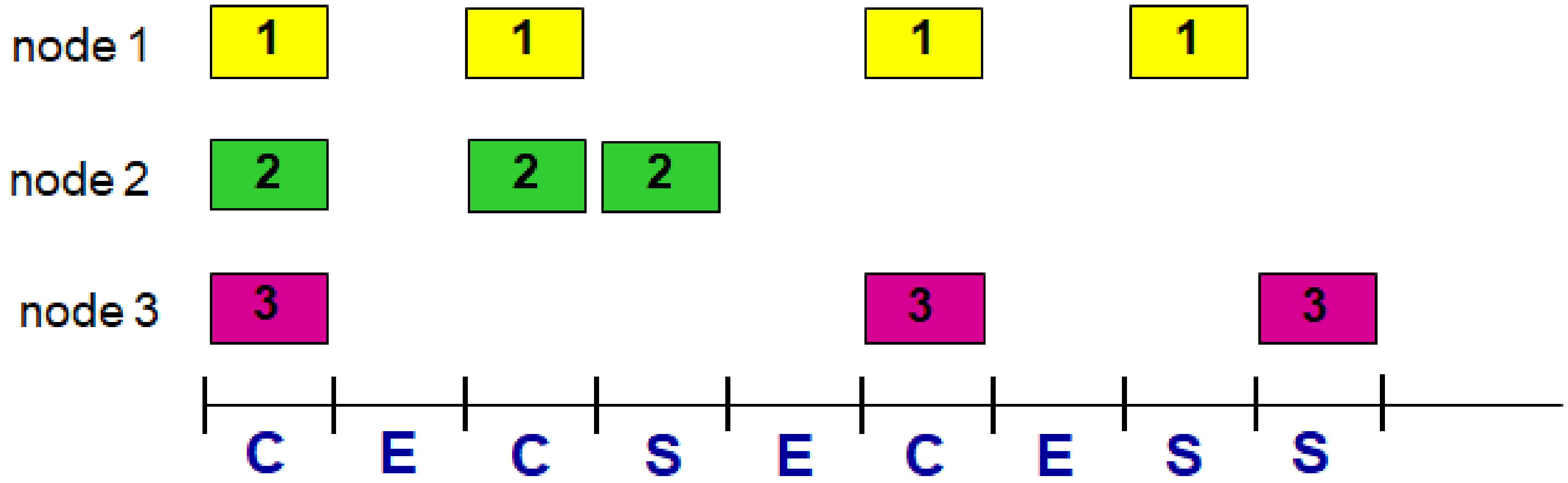
- A station transmits whenever it has data to transmit
- If more than one stations are transmitting **(frame collision) !**
- If ACK not received before timeout, a station picks random backoff time **(to avoid repeated collision)**



Random Access: Slotted ALOHA

Curtin University

- All frames are same size
- Time is divided into equal size slots (time to transmit 1 frame)
- Stations (nodes) are synchronized
- Nodes start to transmit at the beginning of the next slot after data is ready
- if 2 or more nodes transmit in a slot, all nodes detect collision
- *if collision:* node retransmits frame in each subsequent slot with probability p until success



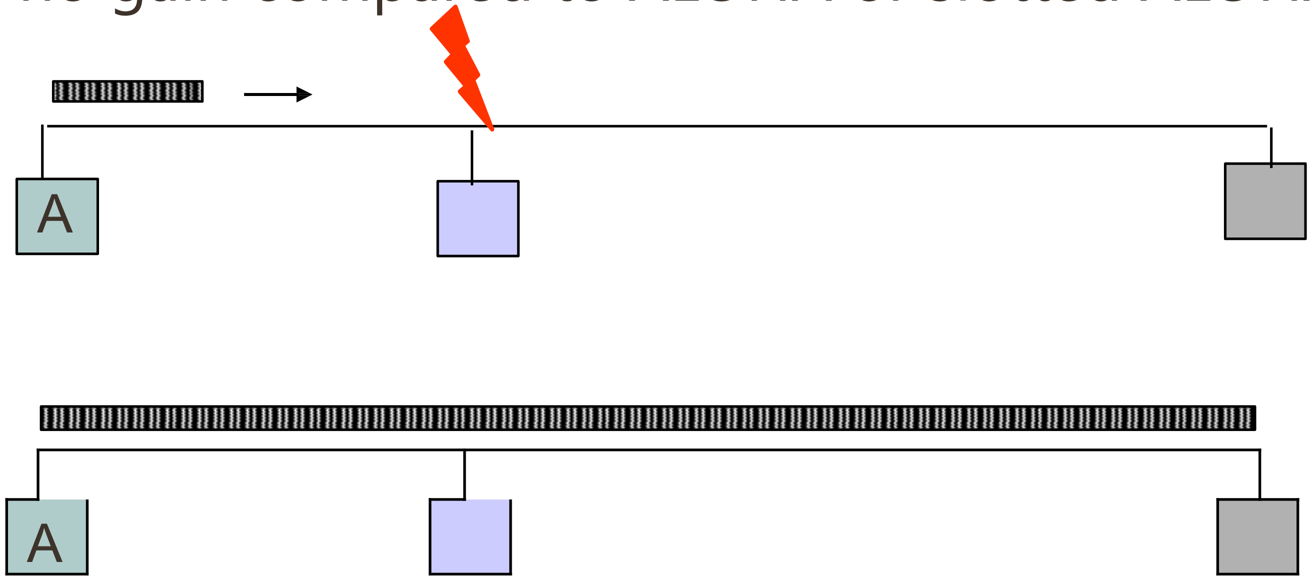
Random Access: CSMA

Curtin University

- Carrier Sense Multiple Access
- A station senses the channel before it starts transmission
- If idle, start transmission
- If busy, either wait or schedule backoff **(CSMA Options)**
- When collisions occur, they involve entire frame transmission times
- If t_{prop} >X (or if a>1), no gain compared to ALOHA or slotted ALOHA

Station A begins transmission at $t = 0$

Station A captures channel at $t = t_{prop}$

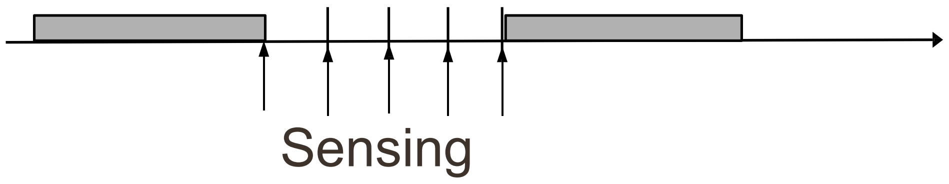


Curtin University

If a channel sensed busy,

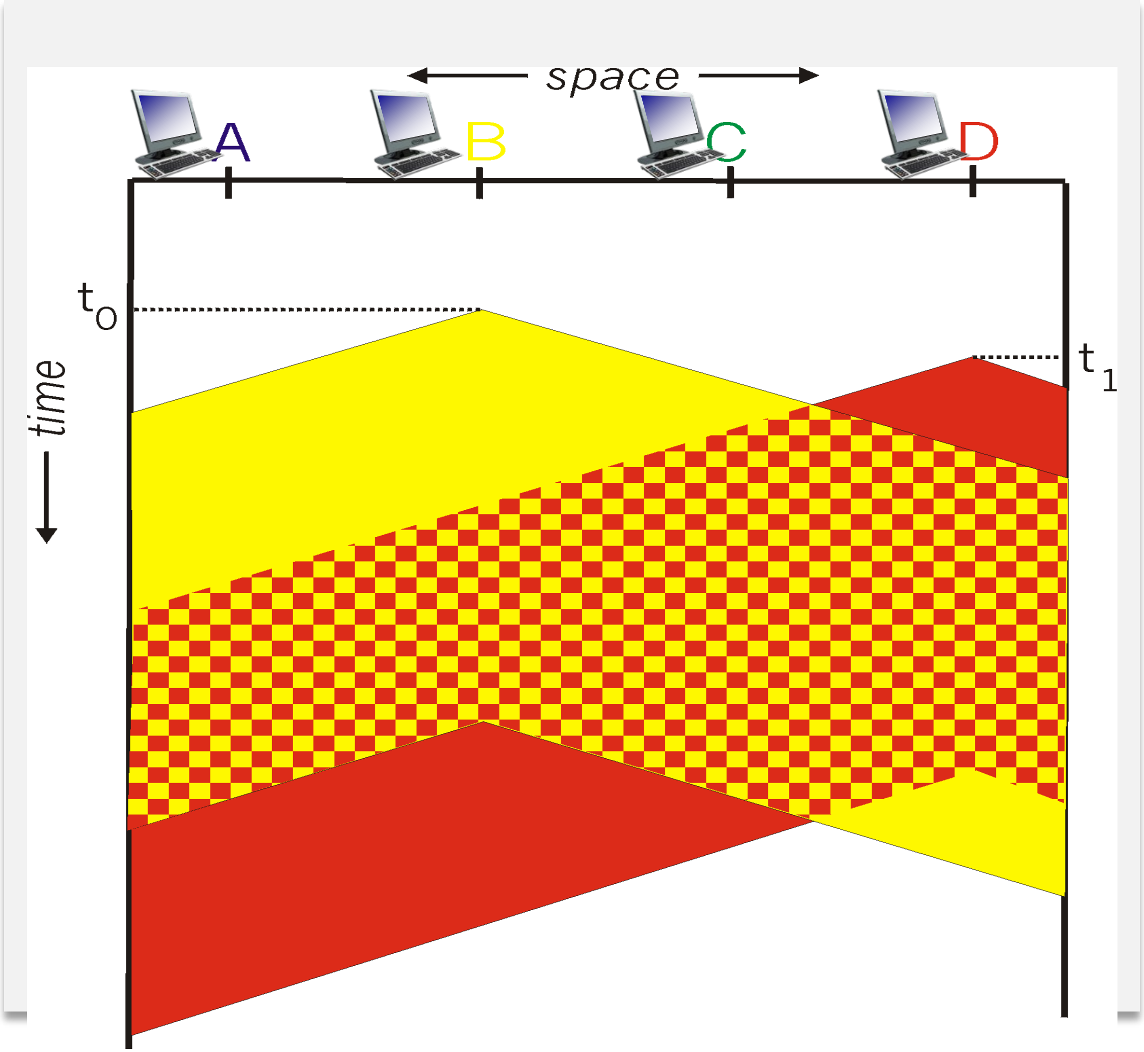
1. **1-persistent CSMA** (most greedy)
   - ✓ Start transmission as soon as the channel becomes idle
   - ✓ Low delay & low efficiency
2. **Non-persistent CSMA** (least greedy)
   - ✓ Wait a backoff period, then sense carrier again
   - ✓ High delay & high efficiency
3. **p-persistent CSMA** (adjustable greedy)
   - ✓ Wait till channel becomes idle, transmit with probability p;
   - ✓ Or wait one mini-slot time & re-sense with probability 1-p
   - ✓ Delay & efficiency can be balanced

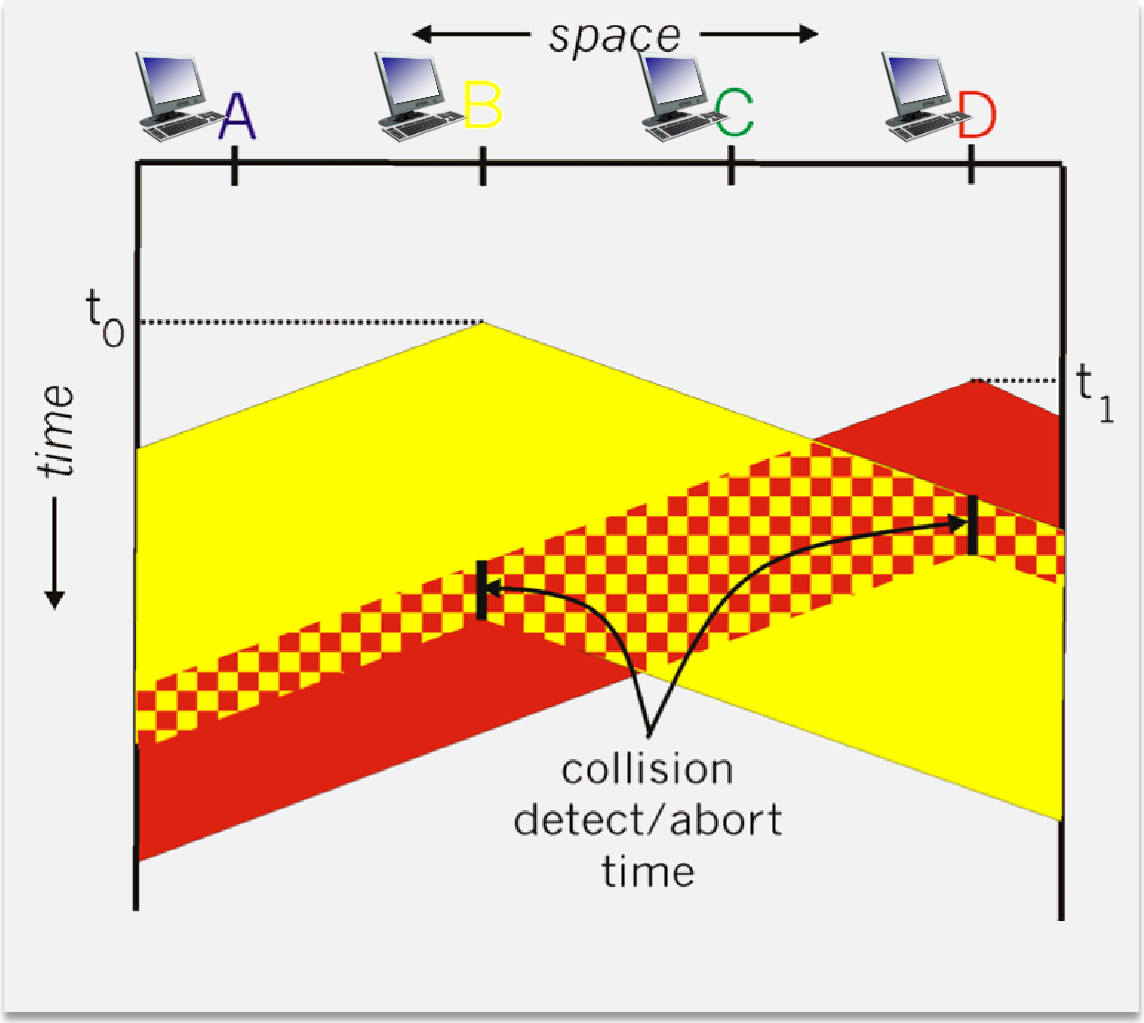
CSMA OPTIONS



Random Access: CSMA-CD

Curtin University





Random Access: CSMA-CD

Curtin University

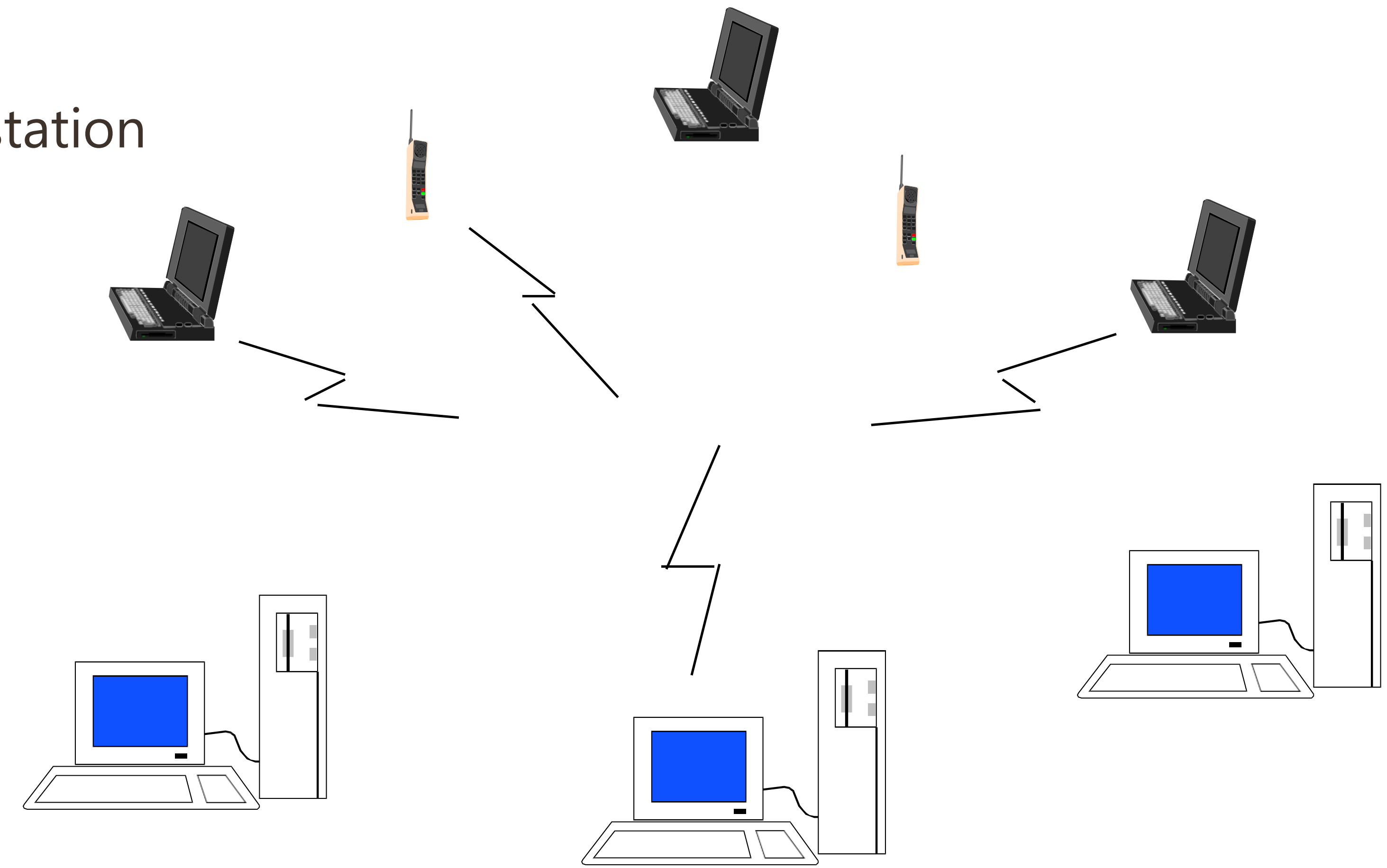
- **Collision Detection**
  - **Easy in wired LANs:** measure signal strengths, compare transmitted, received signals
  - **Difficult in wireless LANs:** received signal strength overwhelmed by local transmission strength

Choice of MAC methods

Curtin University

- **E.g. In Wireless LAN**
  - **Ad Hoc:** station-to-station
  - **Infrastructure:** stations to base station
  - **Access Method:**

Random Access & Polling



Curtin University

- **Data-link Layer**
  - Fundamentals
  - Protocols, Sub-layers
  - Sub Layers (LLC, MAC)
  - Sub Layers: Services
- **LLC: Flow Control**
  - Stop-and-Wait Protocol
  - Sliding Window Protocols
    - Go-Back-N ARQ
    - Selective-Reject ARQ
- **LLC: Error Detection**
  - Parity-check (1d, 2d)
  - Checksum
  - CRC
- **MAC: Framing**
  - Bit-stuffing / HDLC
  - Byte-stuffing / PPP
  - HDLC, PPP Applications
- **MAC: Media Access Control/Sharing**
  - Scheduling Methods
    - Polling
    - Token-passing
  - Random Access Methods
    - ALOHA
    - CSMA
    - CSMA/CD

Curtin University

THANK YOU

Make tomorrow better.