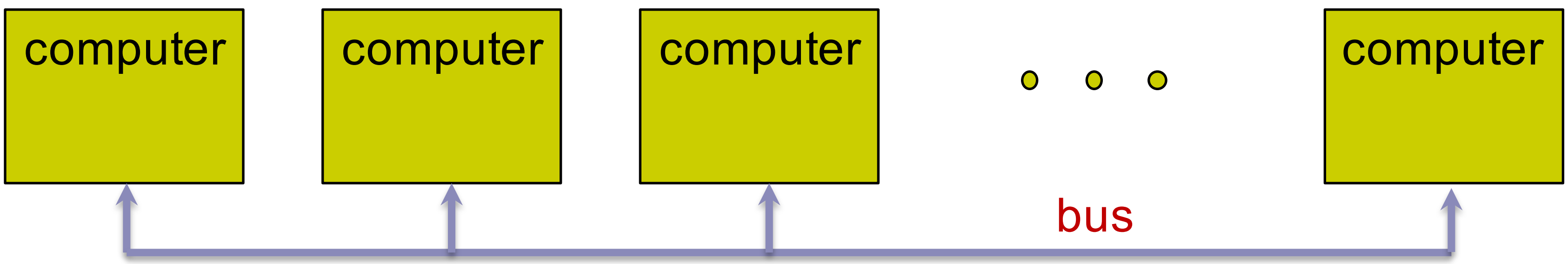
CAN bus
(Controller Area Network)

Part 2

ECE 2534

A brief look at networking issues

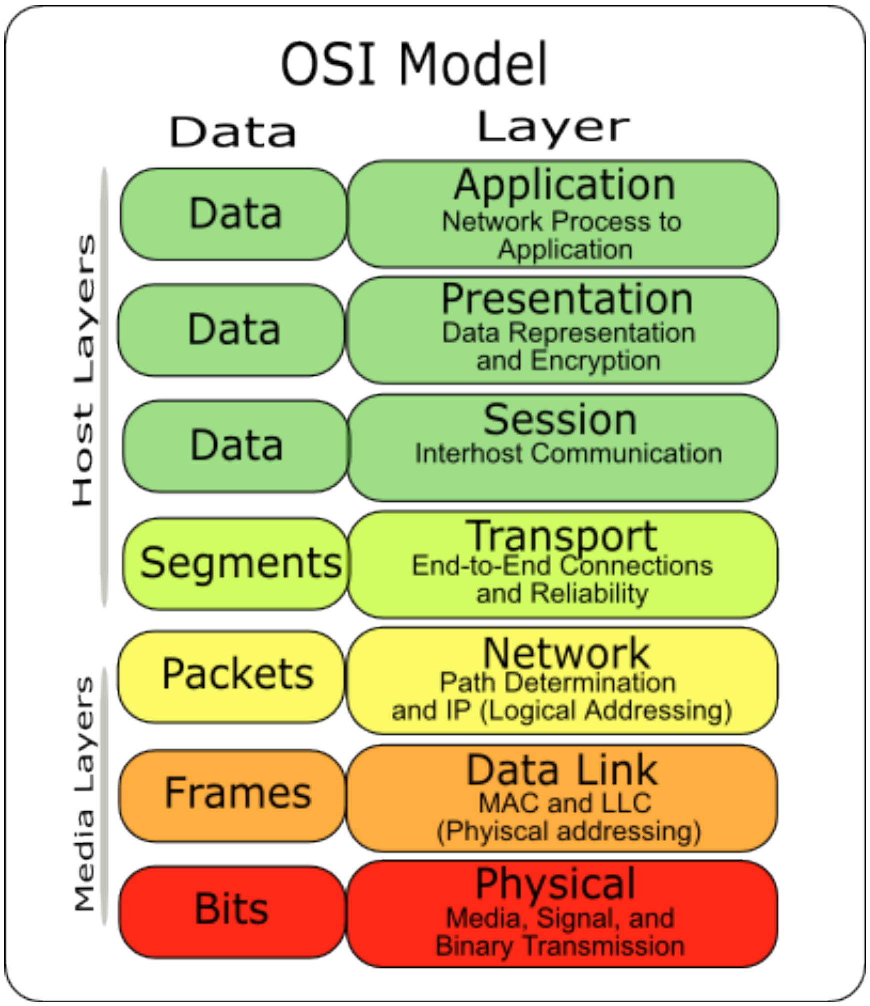
- Many topologies are possible for computer networks
- CAN uses a "bus" topology



- A network protocol is a formal set of rules that governs how computers and other network devices exchange information over a network

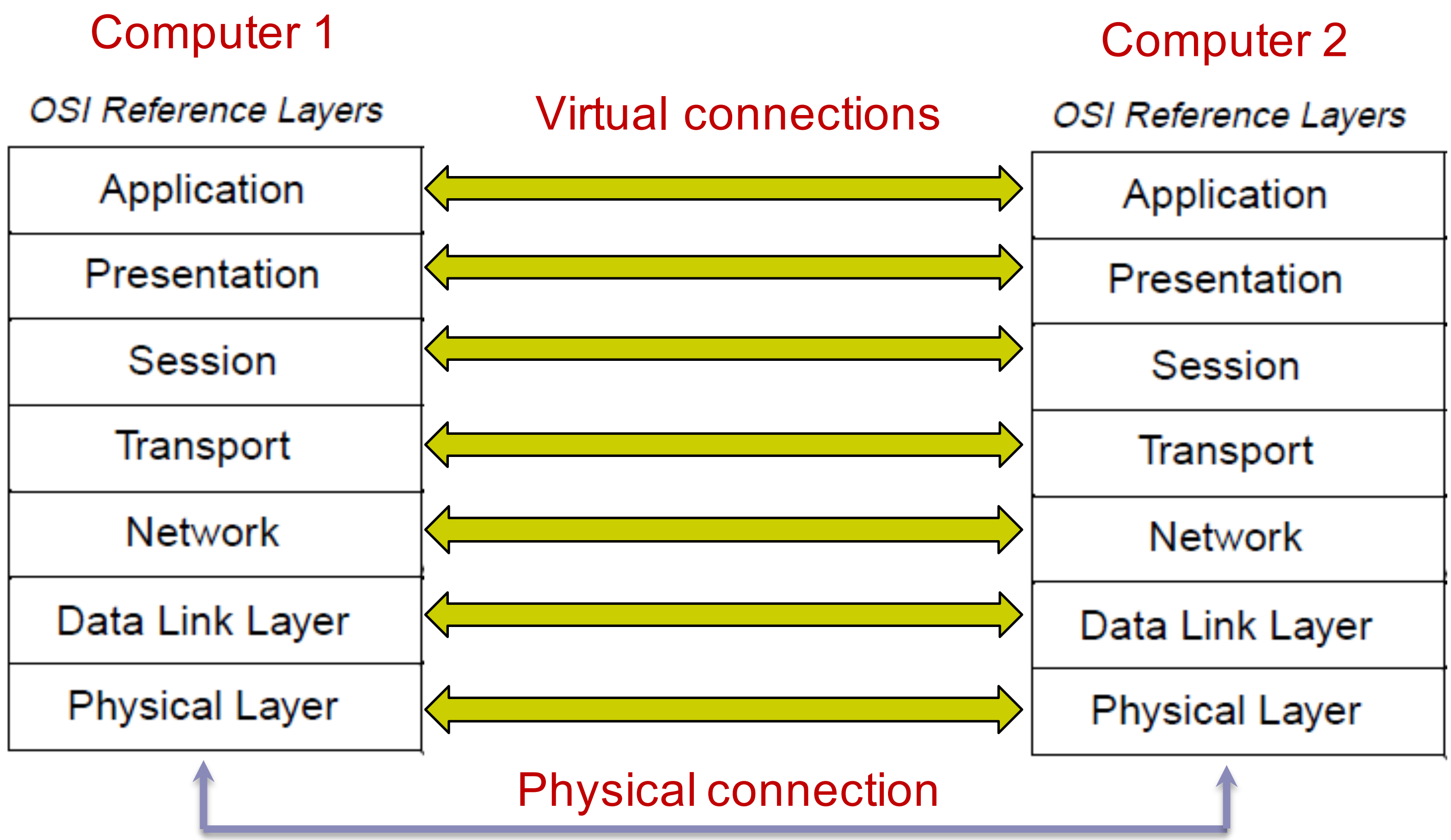
The ISO/OSI reference model

International Standards Organization / Open Systems Interconnection

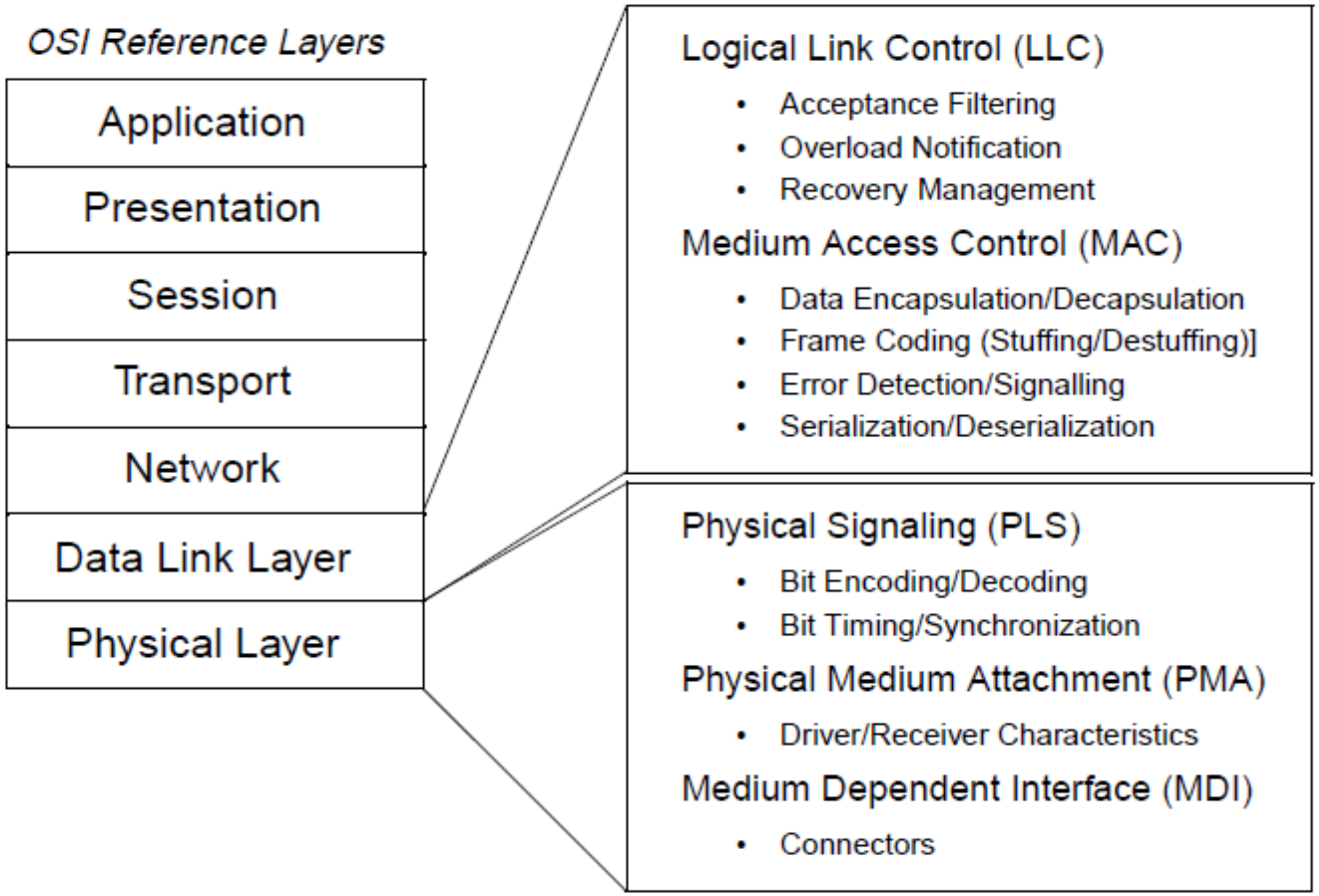


- Developed in the 1980s to overcome communication problems between different vendors' computers
- Now it is used primarily as a point of departure to discuss actual network protocols

- In the OSI model, each layer is responsible for communication at its level only
- Assume that all lower levels are doing their jobs



CAN is only concerned with the lower levels



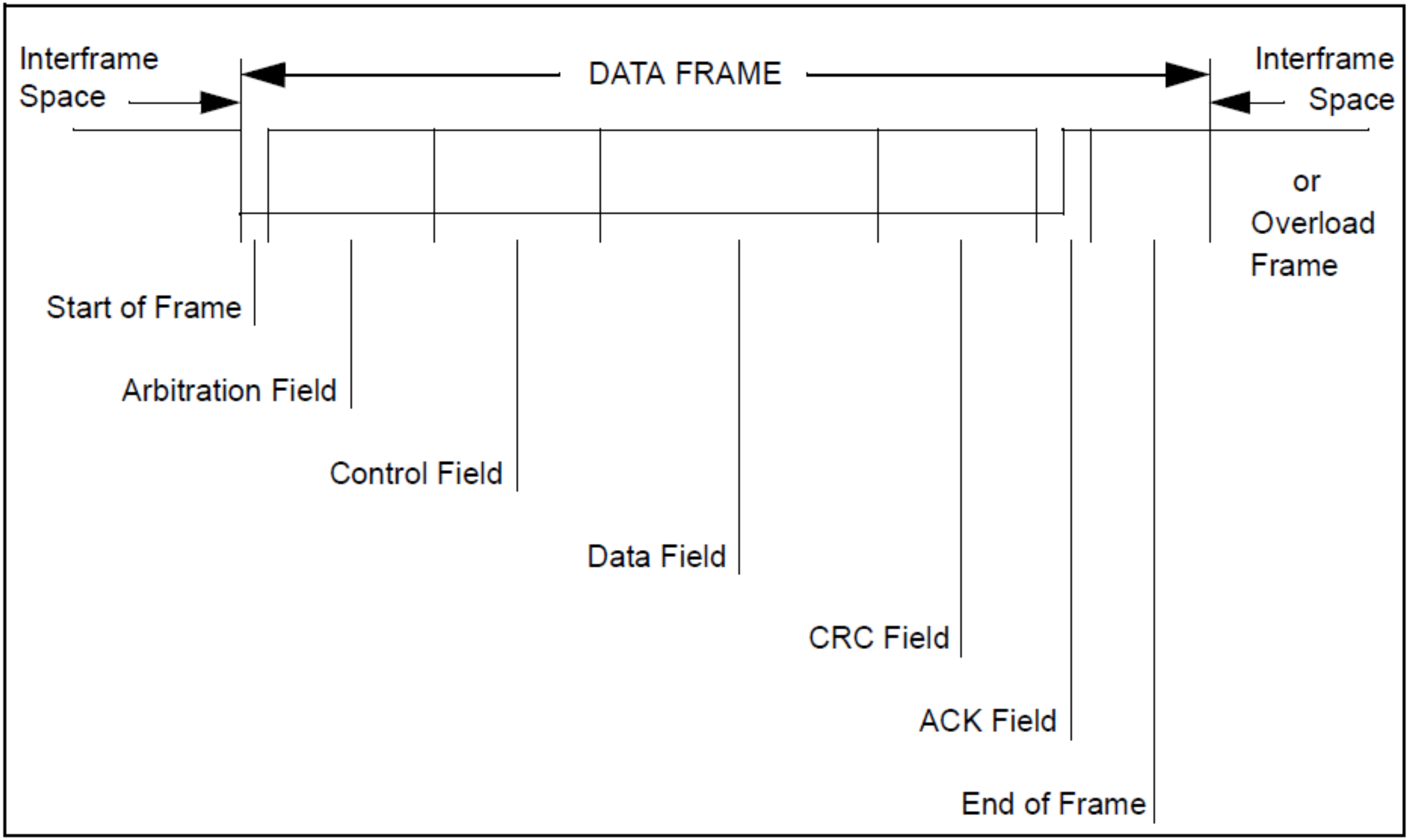
CAN doesn’t specify the actual physical/electrical medium, except to require “dominant” and “recessive” logic states

CAN message types

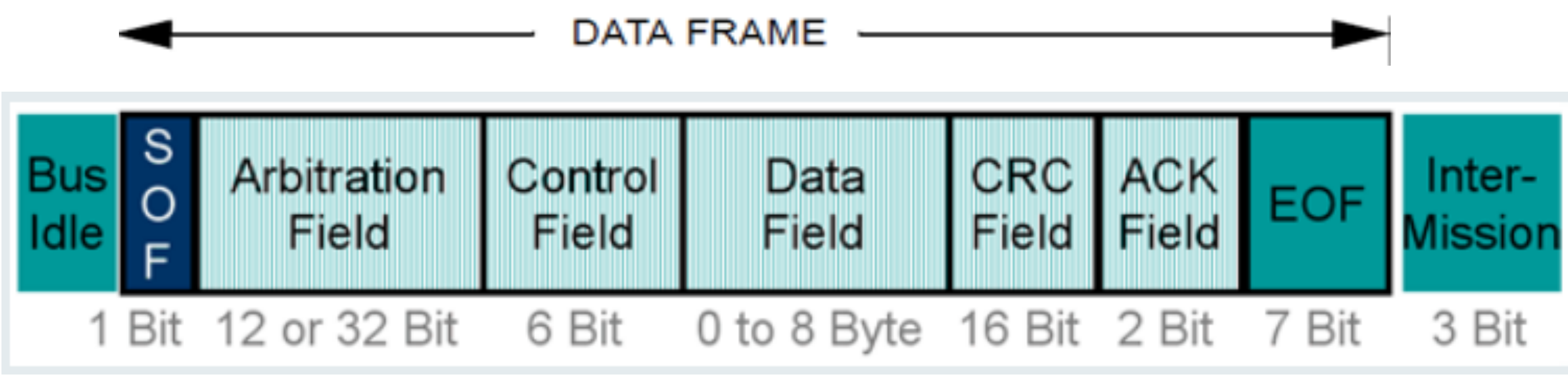
1. DATA FRAME
   - Carries regular data
2. REMOTE FRAME
   - Used to request the transmission of a DATA FRAME with the same ID
3. ERROR FRAME
   - Transmitted by any unit detecting bus error
4. OVERLOAD FRAME
   - Used to force a time interval between frame transmissions

- Reminders:
  - Multi-master bus
  - Any node is allowed to transmit
  - CSMA/CD – a node that wants to transmit waits for the bus to become idle, and then starts to transmit… but watches the bus while transmitting signals to detect collisions
- After each transmitted frame, normally there should be at least 3 ‘r’ bits of “interframe space”

A DATA FRAME is composed of 7 different fields:



A DATA FRAME is composed of 7 different fields:



Size of standard DATA FRAME: 44 + 8N bits, where N is the number of bytes in the data field

Source:
http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/data-frame.php?navanchor=3010395

Format of standard DATA FRAME

Field name	Length (bits)	Description
Start-of-frame	1	Announces intention to send a frame
Arbitration	12	11-bit message priority + 1-bit RTR (d)
Control	6	1-bit IDE (d) + 1-bit reserved (d) + 4-bit data length code
Data	0 to 64	0 to 8 bytes of data, with length specified in Control field
CRC	16	15-bit cyclic redundancy check + 1-bit delimiter (r)
ACK	2	First bit: transmitter sends (r), and any receiver can assert (d) Second bit: 1-bit delimiter (r)
End-of-frame	7	Value must be (r)

(d) = dominant = 0 (r) = recessive = 1

DATA FRAME

- Start of frame (SOF):
  - 1 dominant bit
  - A frame can only start when the bus is IDLE; all stations synchronize to the leading edge of the this bit
- Arbitration field:
  - Encodes the priority level of this message
  - Bit order is most-significant to least-significant
  - 11-bit identifier + 1-bit RTR
    (CAN 2.0 also allows 29-bit identifier)
  - RTR = Remote Transmission Request:
    dominant for <u>data</u> frames
    (recessive for <u>request</u> frames)

CAN bus arbitration example

Suppose 3 computers happen to begin transmitting simultaneously:

Node 1 wants to transmit identifier `11001011101`

Node 2 wants to transmit identifier `11001101010`

Node 3 wants to transmit identifier `11001011001`

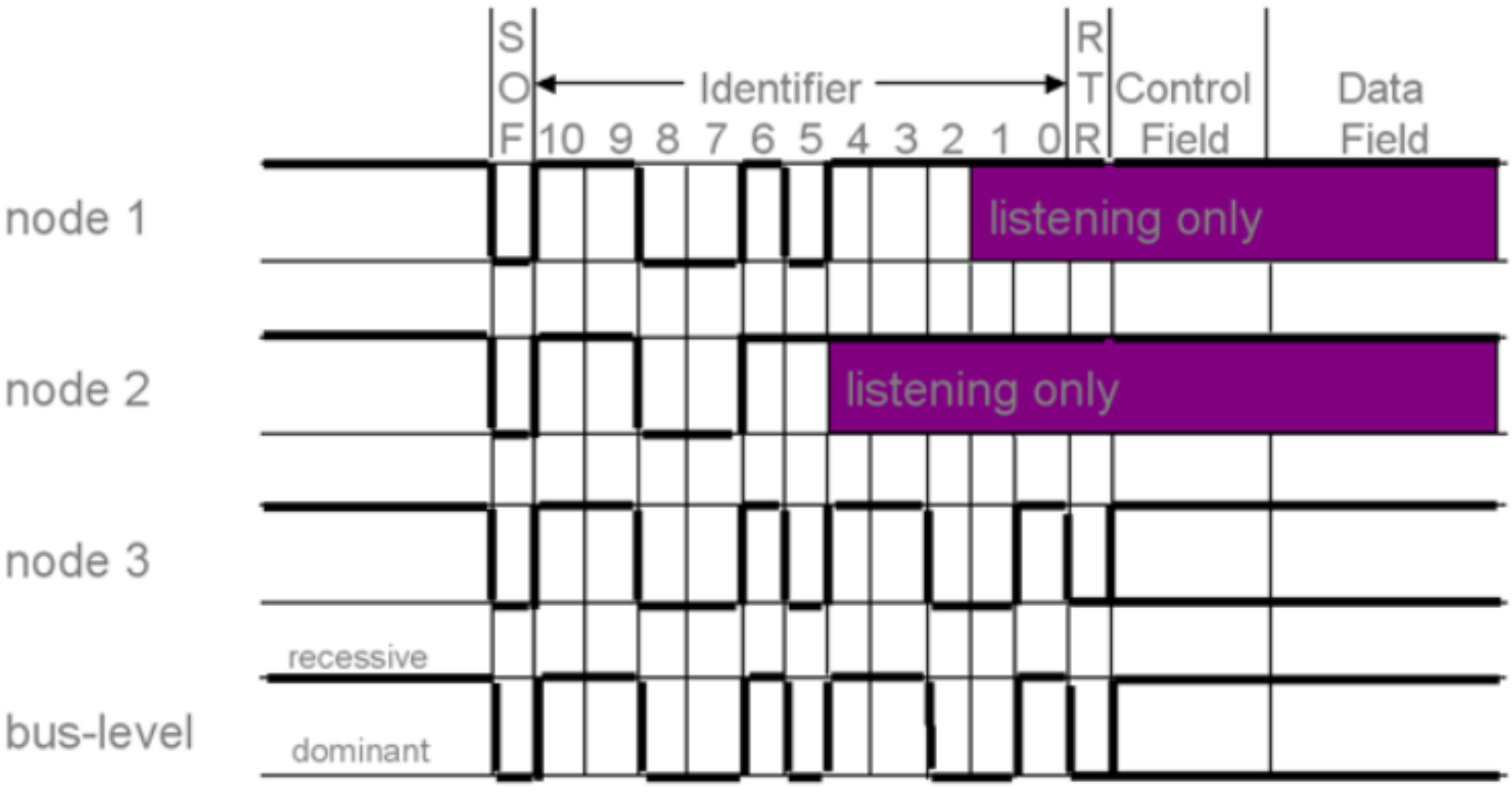
CAN bus arbitration example

Suppose 3 computers happen to begin transmitting simultaneously:

Node 1 wants to transmit identifier `11001011101`

Node 2 wants to transmit identifier `11001101010`

Node 3 wants to transmit identifier `11001011001`



Node 3 wins arbitration and transmits his data.

DATA FRAME

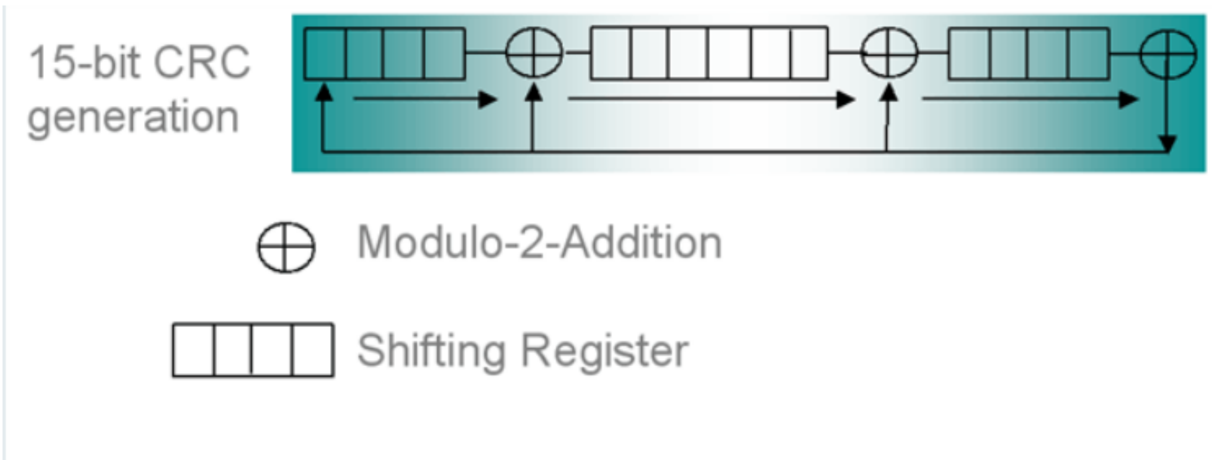
- Control field:
  - 1-bit IDE = “ID Extended”
    dominant for standard data frames
    (recessive for extended data frames)
  - 1-bit reserved, must be transmitted as dominant
  - 4-bit Data Length Code

No. of Data Bytes	Data Length Code (DLC)			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d/r	d/r	d/r

- Data field:
  - 0 to 8 bytes, as specified in the control field

DATA FRAME

- CRC field:
  - 15-bit Cyclic Redundancy Check code, followed by
  - 1-bit delimiter = dominant
  - Don't worry about how it works -- it's an easy-to-implement method of detecting bit errors
  - The CRC method used here can detect up to 5 bit errors scattered throughout all previous fields



$$x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$$

DATA FRAME

- ACK field:
  - 1st bit: transmitter sends recessive, and any receiver can assert dominant (to signal successful reception)
  - 2nd bit: recessive
- End-of-frame field:
  - Sequence of 7 recessive bits
  - Avoids confusion with other frame types

CAN message types

1. DATA FRAME
   - Carries regular data
2. REMOTE FRAME
   - Used to request the transmission of a DATA FRAME with the same ID
3. ERROR FRAME
   - Transmitted by any unit detecting bus error
4. OVERLOAD FRAME
   - Used to force a time interval between frame transmissions

REMOTE FRAME

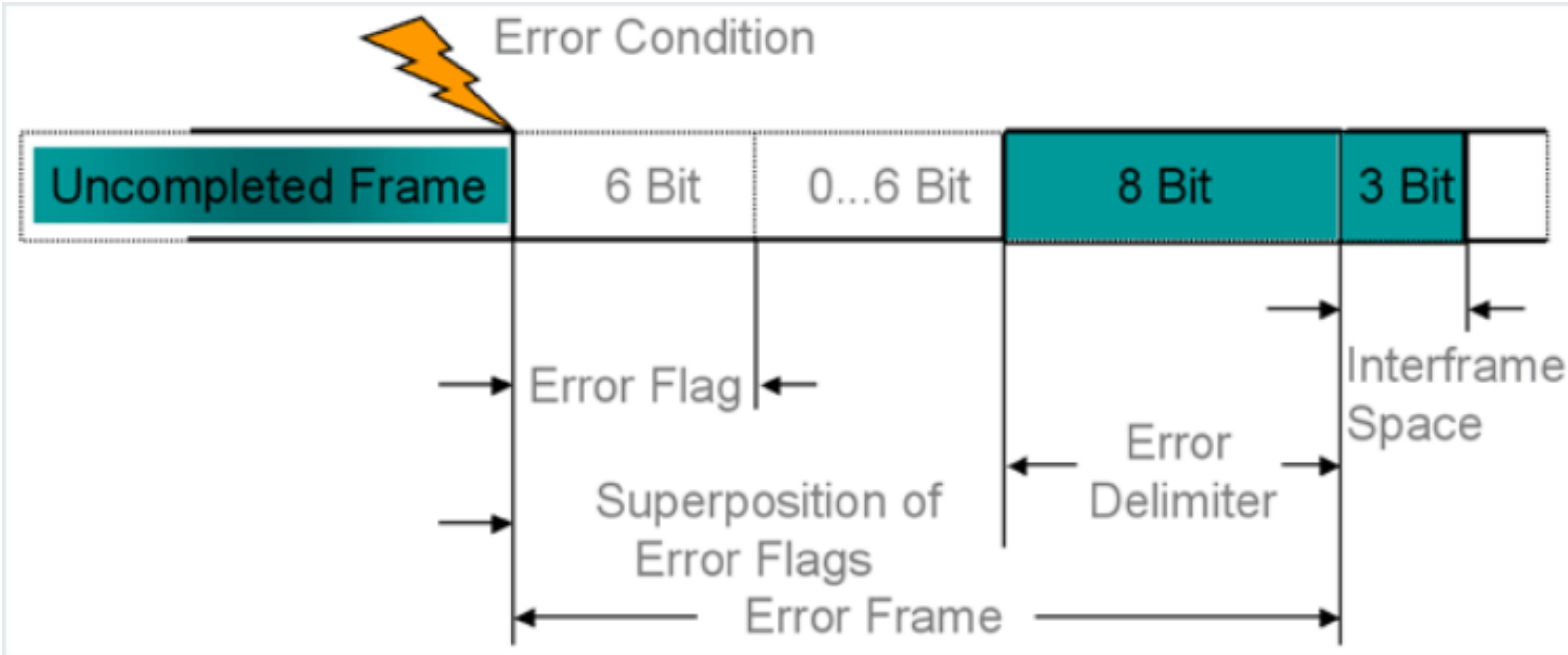
- Used by node that wants to request data from another node
  - “How’s the oil pressure right now?”
- Same as DATA FRAME except…
  - RTR bit is set to recessive
  - No data field
  - Data Length Code value is ignored

- What happens if DATA FRAME and a REMOTE FRAME with the same identifier begin transmitting at the same time?

Reminder: CAN message types

1. DATA FRAME
   - Carries regular data
2. REMOTE FRAME
   - Used to request the transmission of a DATA FRAME with the same ID
3. ERROR FRAME
   - Transmitted by any unit detecting bus error
4. OVERLOAD FRAME
   - Used to force a time interval between frame transmissions

ERROR FRAME



- Consists of 2 fields
  - Error flag: 6 dominant bits
    (plus up to 6 additional bits, as other nodes see the error and report it also)
    Notice this will overwrite any recessive bits being sent
  - Error delimiter: 8 recessive bits

ERROR FRAME

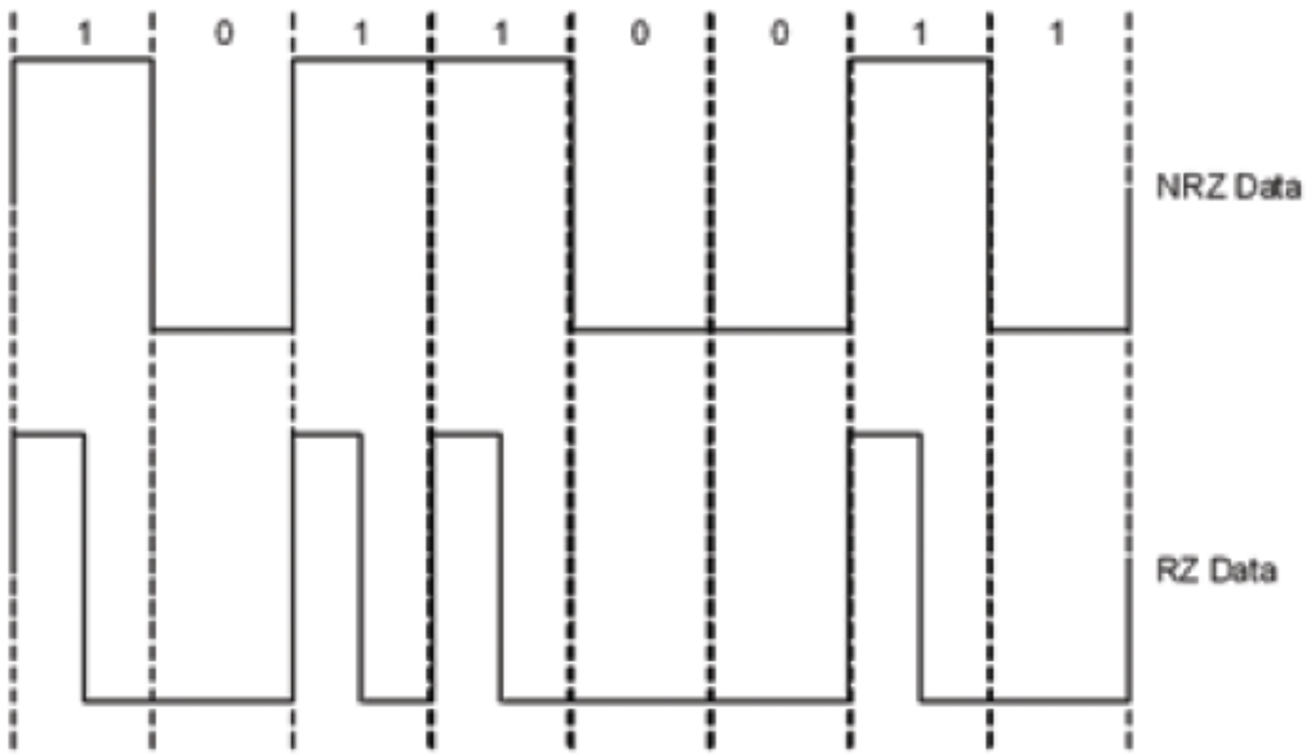
- When something goes wrong, any node can transmit an ERROR FRAME
- There are 5 error types
  - Bit error
    (transmitter sees a different bit than it sends)
  - Stuffing error (see below)
  - CRC error
  - Form error (transmitter detects a ‘d’ bit in one of the places where ‘r’ is required)
  - ACK error
    (transmitter does not see a ‘d’ in ACK slot)

Stuffing? (Thanksgiving was last week)

- Potential problems can arise when long sequences of 1 or 0 are sent
- Remember, no clock signal is being transmitted
  - Receivers are expected to synchronize based on recessive-to-dominant <u>transitions</u> in the data stream
  - If there are no transitions in the data stream, receivers can drift out of sync with the transmitter
- To address the problem, CAN uses bit-stuffing to force transitions to occur

More terminology

- CAN uses NRZ (Non-Return-to-Zero) transmission, which is what we'd consider to be normal
- An alternative is RZ (Return-to-Zero), which means that a 1 signal level appears for only a fraction of the bit period



Bit stuffing on CAN bus

- Whenever 5 bits of the same polarity are to be transmitted in a row, the transmitter automatically inserts ("stuffs") an opposite-polarity bit into the data stream after those 5 bits
- Receiving nodes will use the transition for synchronization, but will ignore the stuffed bit for data purposes
- If 6 consecutive bits with the same polarity are detected between the Start of Frame and the CRC delimiter, then the bit stuffing rule has been violated
  - Receivers should send a Error Frame
  - The transmitter should repeat the message

Example of <u>bit stuffing</u> on CAN bus

- Partial sequence to transmit:
  `110000000011111100111`
  `121234567812345121 23`

- Result after bit-stuffing:
  `1100000`**`1`**`000111111`**`0`**`00111`
  `1212345`**`S`**`123123456`**`S`**`12123`

- The receiver will count bits and "de-stuff" the sequence:
  `110000000011111100111`
  `121234567812345121 23`

ERROR FRAME

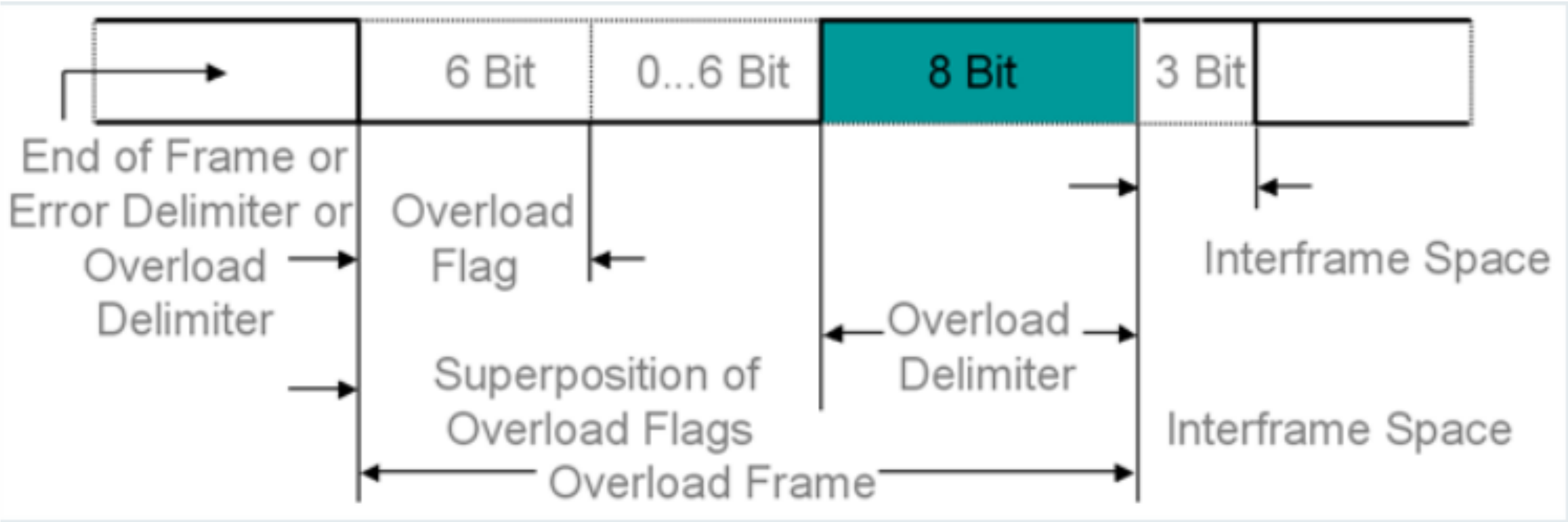
- When a node detects any of these errors,
  - It discards the current message
  - It transmits an ERROR FRAME
- The transmitter is expected to re-transmit the message that was interrupted by the ERROR FRAME

Reminder: CAN message types

1. DATA FRAME
   - Carries regular data
2. REMOTE FRAME
   - Used to request the transmission of a DATA FRAME with the same ID
3. ERROR FRAME
   - Transmitted by any unit detecting bus error
4. OVERLOAD FRAME
   - Used to force a time interval between frame transmissions

OVERLOAD FRAME

- When a node needs more time, it can transmit an OVERLOAD FRAME to delay other frames
- Can only start during interframe spacing
- Consists of
  - 6 ‘d’ bits (plus up to 6 additional bits, as other nodes see the OVERLOAD FRAME and repeat it), followed by
  - 8 ‘r’ bits



Fault confinement on CAN bus

- Goal: Don’t allow a faulty node to monopolize the network
- A node can be in one of 3 possible fault modes
  - Error-Active – node sends all frames including error (This is the normal operational mode)
  - Error-Passive – node sends all frames excluding error
  - Bus-Off – node sends no more frames
- A node’s state is based on counts of transmit errors and receive errors
  - Each error frame increases the count by 8
  - Each successful frame decreases the count by 1

Fault confinement on CAN bus

- Maintain these counts in 2 counters:
  - Transmit Error Counter (TEC)
  - Receive Error Counter (REC)
- Changes in fault modes:
  - Error-Active state
    whenever TEC < 128 and REC < 128
  - Error-Active state → Error-Passive state
    iff TEC > 128 or REC > 128
  - Error-Passive state → Bus-Off state
    iff TEC > 256

Is that all?

- No, there are many more details
- CAN is a real-world, “mission-critical” protocol
- CAN is possibly the most complex peripheral on the PIC32
  (the Ethernet peripheral could be an exception)