

Communication Networks for the Next-Generation Vehicles

Syed Masud Mahmud, Ph.D.
Electrical and Computer Engg. Dept.
Wayne State University
Detroit MI 48202
(313) 577-3855,
smahmud@eng.wayne.edu

Speaker's Background

- Received Ph.D. in Electrical Engineering from the University of Washington, Seattle (1984).
- Worked for Oakland University, Rochester, Michigan, during 1984-1988.
- Has been working for Wayne State University since 1988.
- Published about 80 technical papers in referred journals and conference proceedings.

Speaker's Background

- Current areas of interest are
 - Intelligent Vehicles,
 - Intelligent Transportation Systems,
 - In-Vehicle Networking,
 - Time Triggered Protocols for Real-Time Applications,
 - Collision Warning and Collision Avoidance systems,
 - Security in Wireless Communications, and
 - Embedded Systems.

Outline of the Talk

- Need for New Types of In-Vehicle Networks
- CAN Protocol
- Time Triggered CAN (TTCAN)
- Hierarchical Networks
- Fault Tolerant Hierarchical Networks
- Simulation Models for Event- and Time-Triggered Systems

Need for New Types of In-Vehicle Networks

- Drive-by-wire,
- Telematics,
- Entertainment,
- Multimedia,
- Pre-Crash Warning,
- Hit Avoidance,
- Remote Diagnostic
- Remote Software Update, etc.

Need for New Types of In-Vehicle Networks (contd..)

- Drive-by-Wire and Active Collision Avoidance Systems need *fault tolerant* networks with *time-triggered* protocols, to guarantee *deterministic latencies*.

Need for New Types of In-Vehicle Networks (contd..)

- Multimedia Systems need networks with *high bandwidth* to transfer video files
- Body control electronics need *low-bandwidth* networks to keep the cost down.

Need for New Types of In-Vehicle Networks (contd..)

- Different sets of vehicle electronic modules will require different types of networks.

Need for Partitioned In-Vehicle Networks

- Since the complexity of the network is increasing and the demand for bandwidth is growing, future vehicles will require many partitioned networks.

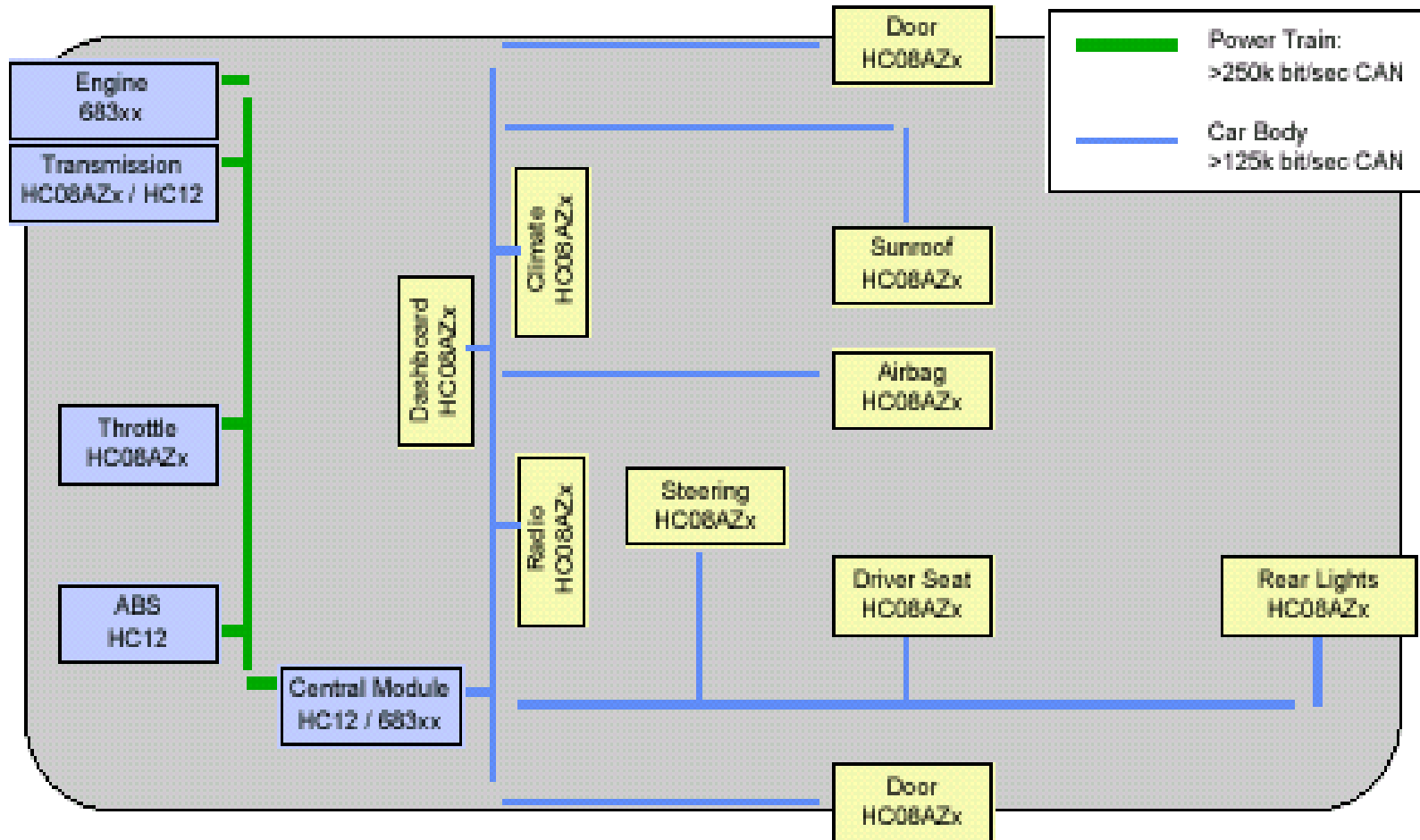
CAN Protocol

- CAN is a serial protocol.
- It's a multimaster, multicast protocol.
- It supports distributed real-time control with high level of data integrity.
- It was defined by BOSCH for automotive applications.
- Currently CAN is not restricted to auto industry.

CAN Protocol

- It satisfies the communication needs of a wide range of applications, from high-speed networks to low-cost multiplexing.
- Multiple CANs with different speeds can be used in a particular system.

A Typical CAN System



Courtesy of Motorola

Layers of CAN

- CAN has been divided into three layers.
 - The Object Layer
 - The Transfer Layer
 - The Physical Layer
- The Object and Transfer Layers comprise all services and functions of the Data Link layer defined by the ISO/OSI model.

CAN Object Layer

- The scope of the Object Layer includes:
 - Determining which messages are to be transmitted
 - Deciding which messages received by the transfer layer are to be used.
 - Providing an interface to the application layer related hardware

CAN Transfer Layer

- Transfer Layer mainly deals with the following issues:
 - Controlling the framing
 - Performing arbitration
 - Checking errors
 - Signaling errors
 - Fault confinement
 - Determines whether or not the bus is free for a new transmission.
 - Determines whether reception of a message has just started.

CAN Physical Layer

- It performs the actual transfer of bits between different nodes.
- It varies according to the requirements of individual applications.

CAN: Basic Concepts

Transmitter

- A node which originates a message is called the transmitter.
- It remains as the transmitter until the bus becomes idle or it loses arbitration.

Receiver

- A node is called the receiver if it is not sending a message, and the bus is not idle.

CAN: Basic Concepts

Message Routing

- The content of a message is described by an identifier.
- The identifier describes the meaning of the data, so that all nodes are able to decide whether or not the data is to be acted upon by them.

Priorities

- The identifier defines a static message priority during bus access.

CAN: Basic Concepts

Multi-master

- When the bus is free any node may start to send a message. The highest priority node will be successful in sending the message.

Multicast

- Multiple nodes may receive and act upon the message simultaneously.

CAN: Basic Concepts

Bus Values

- The bus can have one of two complementary values: ***dominant*** and ***recessive***.
- During simultaneous transmission of dominant and recessive bits, the resulting bus value will be dominant.
- For example, in the wired-AND implementation of the bus, the dominant value will be '0' and the recessive value will be '1'.

CAN: Basic Concepts

Acknowledgement

- All receivers check the consistency of the message being received.
- The receivers acknowledge a consistent message and flag an inconsistent message.

CAN Message Frames

- There are four types of CAN message frames.
 - Data frame
 - Remote frame
 - Error frame
 - Overload frame

CAN Message Frames

Data Frame

- It carries data from a transmitter to the receivers.

Remote Frame

- It is transmitted by a node to request the transmission of a Data frame with the same identifier.

CAN Message Frames

Error Frame

- It is transmitted by any node after detecting a bus error.

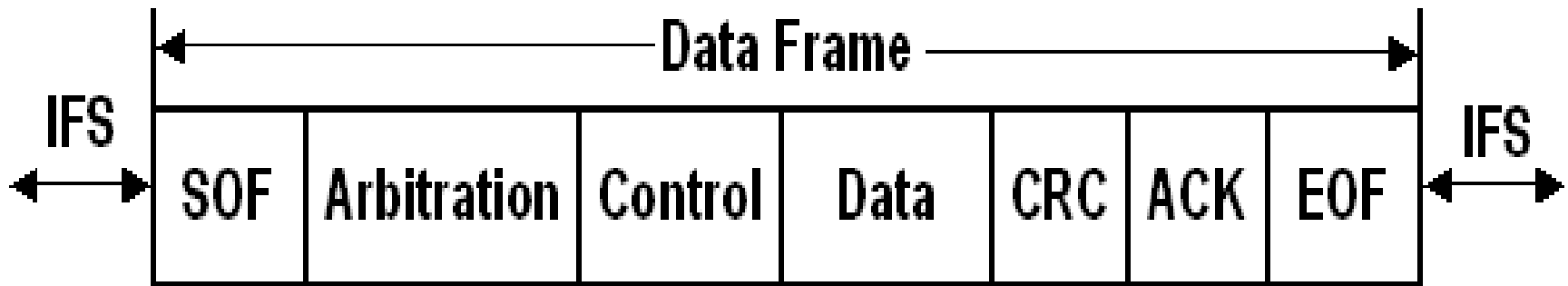
Overload Frame

- It is used to provide additional delay between two data or remote frames.

Data Frame

There are seven fields in a Data frame.

- Start of Frame (**SOF**): It marks the beginning of a data or remote frame. It consists of only one dominant bit. All nodes have to synchronize with the leading edge of the start of frame bit.



Arbitration Field of Data Frame

- **Arbitration Field:** This field consists of the identifier and the RTR bit.
- **Identifier:** There are 11 or 29 bits in the identifier. These bits are transmitted in the order from ID10 to ID0. ID10 is the most significant bit of the 11-bit identifier



Arbitration Field of Data Frame

- The Identifier performs the following operations:
 - Labels the content (type) of a message.
 - Performs acceptance test of messages.
 - Arbitrates & determines the priority of the message.



Arbitration

- Carrier Sense, Multiple Access with Collision Detect (CSMA/CD)
- Method used to arbitrate and determine the priority of messages.
- Uses enhanced capability of non-destructive bit-wise arbitration to provide collision resolution.



Arbitration Technique

- Any potential bus conflicts are resolved by bit-wise arbitration.
- A dominant state (**logic 0**) has precedence over a recessive state (**logic 1**).
- Example: Assume that Node-1, Node-2 and Node-3 are going to compete for the bus with the following identifiers:
 - Node-1 0010100 . .
 - Node-2 0001100 . .
 - Node-3 0000111 . .

Arbitration Technique

- Transmission of identifier bits by Nodes 1, 2 and 3 are as follows:

Node-1	0	0	1	-	-	-	-
Node-2	0	0	0	1	-	-	-
Node-3	0	0	0	0	1	1	1

Node-1 lost
arbitration at
this point

Node-2 lost
arbitration at
this point

RTR Bit

- RTR bit (Remote Transmission Request Bit): In Data frames the RTR bit must be dominant. In Remote frames it must be recessive.

RTR bit

d

r

Type of Frame

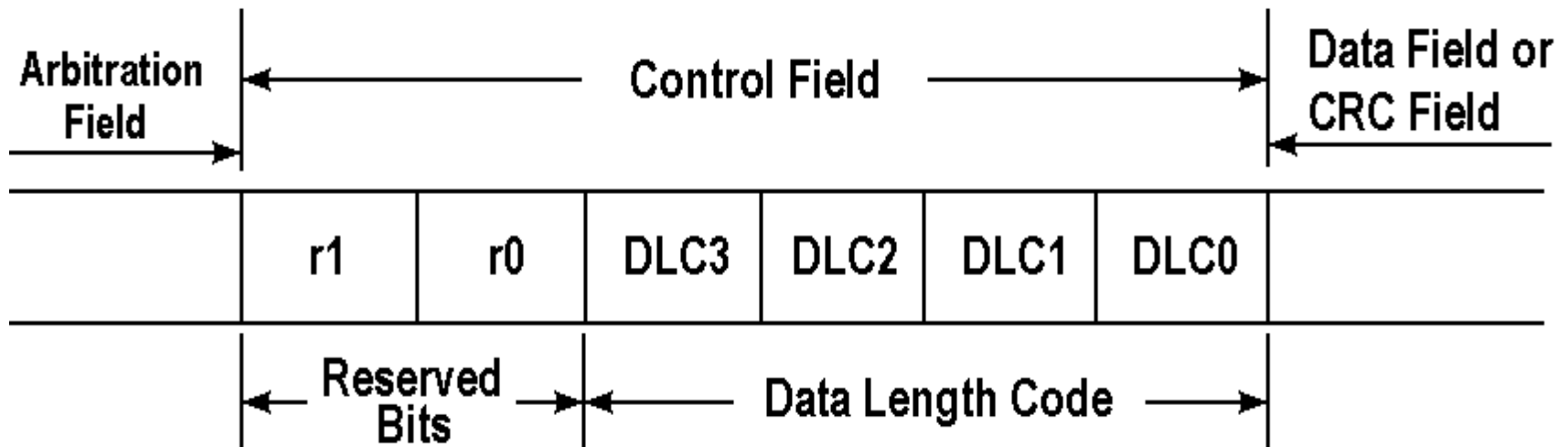
Data Frame

Remote Frame



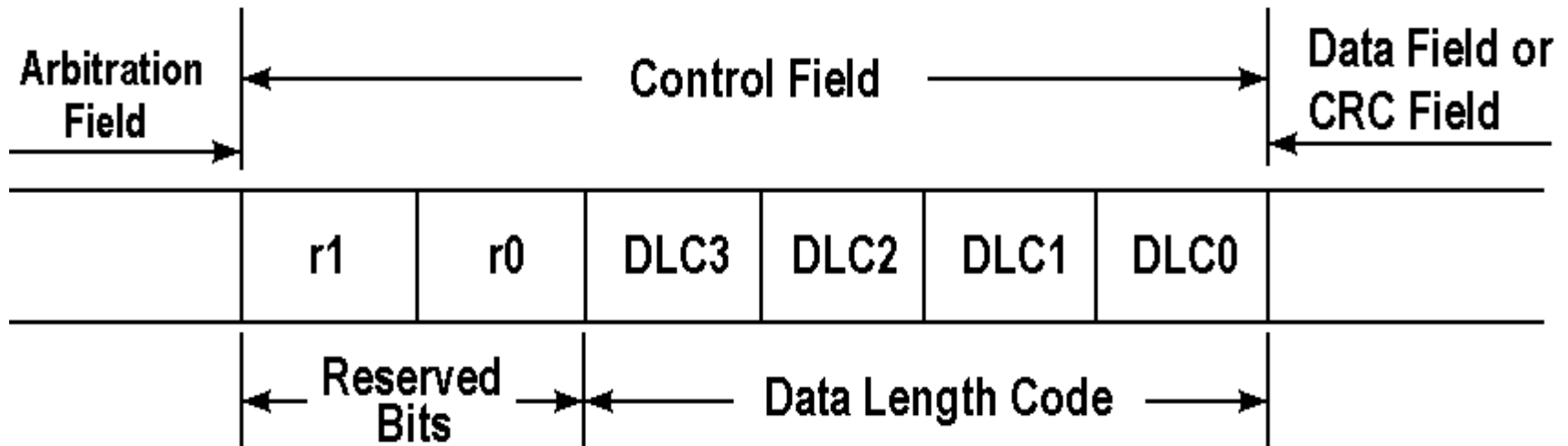
Control Field

- There are 6 bits in the control field.
- Out of the 6 bits, 4 bits are used to indicate the length of the data and the other two bits are reserved for future expansion.



Control Field

- The reserved bits must be sent as dominant.



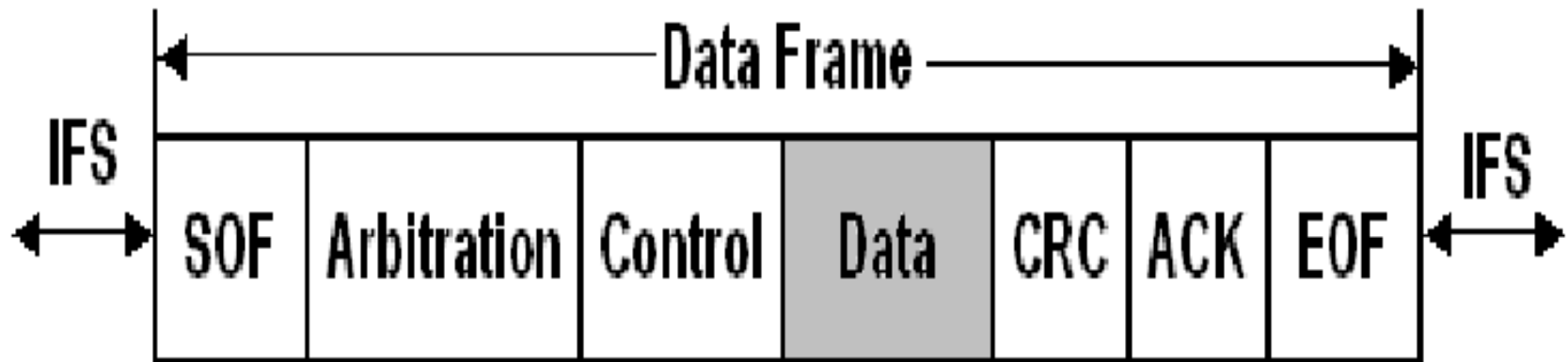
Data Length Code

- The DLC bits can code data lengths from 0 to 8 bytes (**d: dominant, r: recessive**).

DLC3	DLC2	DLC1	DLC0	Length
d	d	d	d	0
d	d	d	r	1
d	d	r	d	2
d	d	r	r	3
d	r	d	d	4
d	r	d	r	5
d	r	r	d	6
d	r	r	r	7
r	d	d	d	8

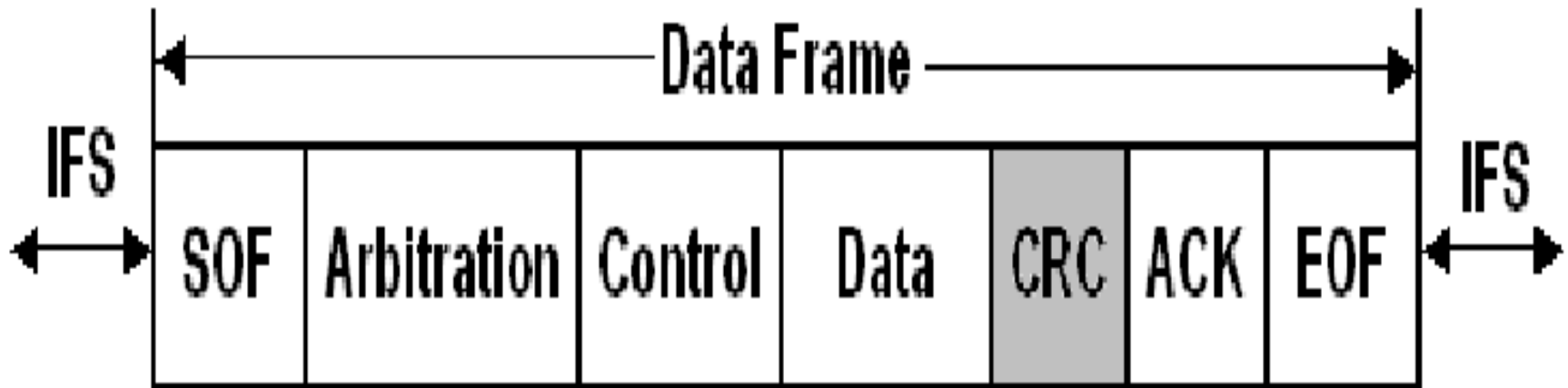
Data Field

- The data field contains the data to be transmitted.
- The data field can contain 0 to 8 bytes.
- The most significant bit of a byte is sent first.



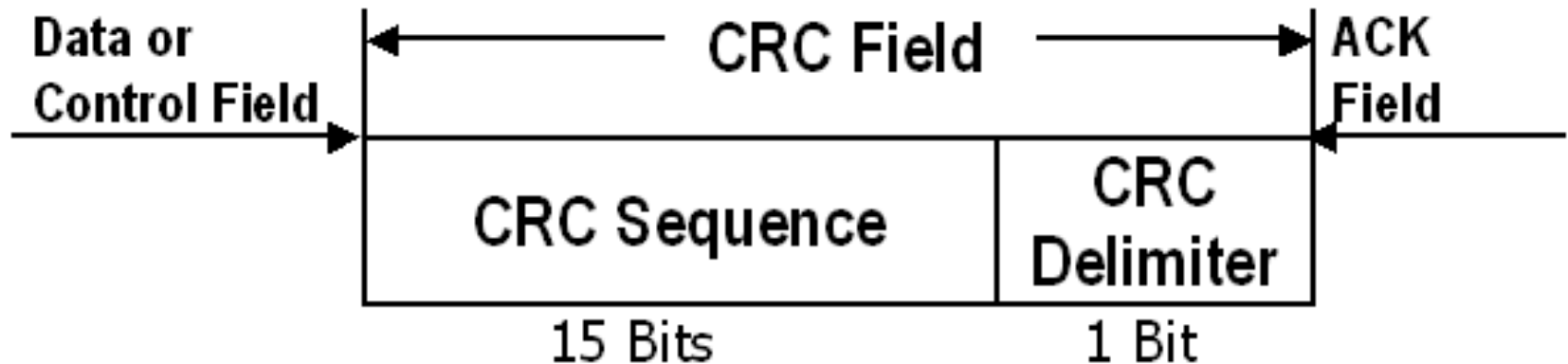
CRC Field

- It's a 16-bit field used to improve transfer reliability.
- It contains a 15-bit CRC Sequence followed by a 1-bit CRC Delimiter.



CRC Field

- The CRC Sequence is calculated by using destuffed bits from the SOF to the last bit of the Data Field.
- The CRC delimiter is a single **recessive** bit.



CRC Field

- During message transfer, all receivers compute a CRC value.
- Every receiver then compares its calculated CRC value with the CRC value sent by the transmitter.
- If both match, then the receiver acknowledges the message.
- If they do not match, then the receiver does not acknowledge the message and sends an **Error Frame**.

Bit-Stream Coding

- The frame segments: **Start of frame**, **Arbitration field**, **Control field**, **Data field** and **CRC Sequence** are coded by the method of bit stuffing.
- Whenever a transmitter detects five consecutive bits of identical value in the bit-stream to be transmitted, it automatically inserts a complementary bit in the actual transmitted bit-stream.

Bit-Stream Coding

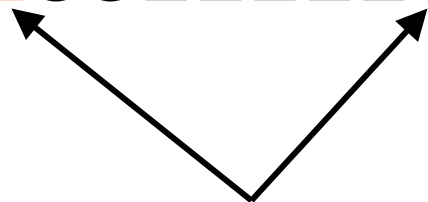
Example

- Original Bit Sequence:

000000011111111

- Transmitted Bit Seq.:

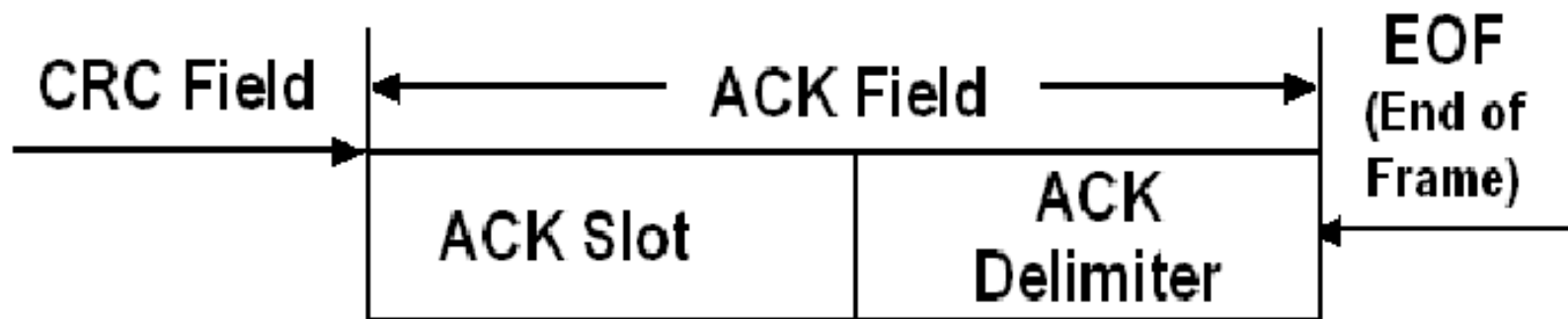
00000100111110111



Stuff Bits

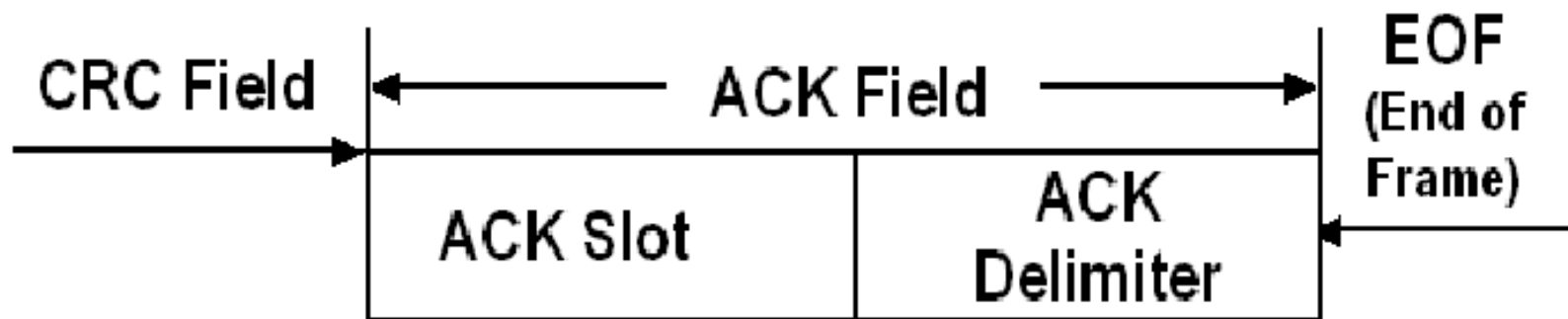
Acknowledge Field

- The ACK field consists of two bits.
- One bit for ACK slot and another bit for ACK delimiter.
- The TRANSMITTING node sends two recessive bits through the ACK field.



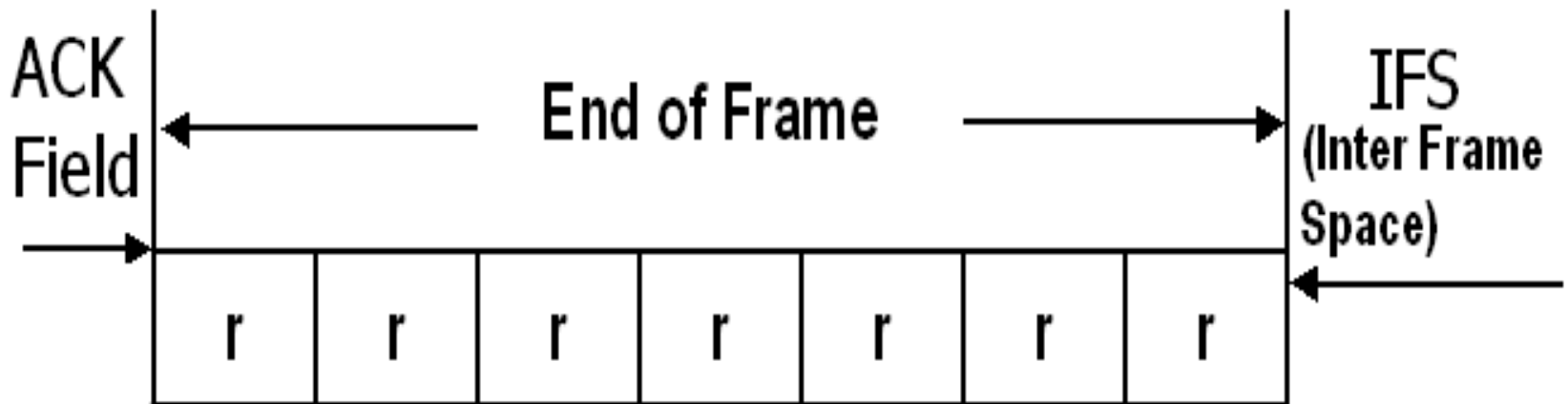
Acknowledge Field

- The RECEIVING node sends a dominant bit through the ACK slot after receiving a valid message.
- Since the CRC and ACK delimiters are both recessive bits, the ACK slot is always **surrounded by two recessive bits**.



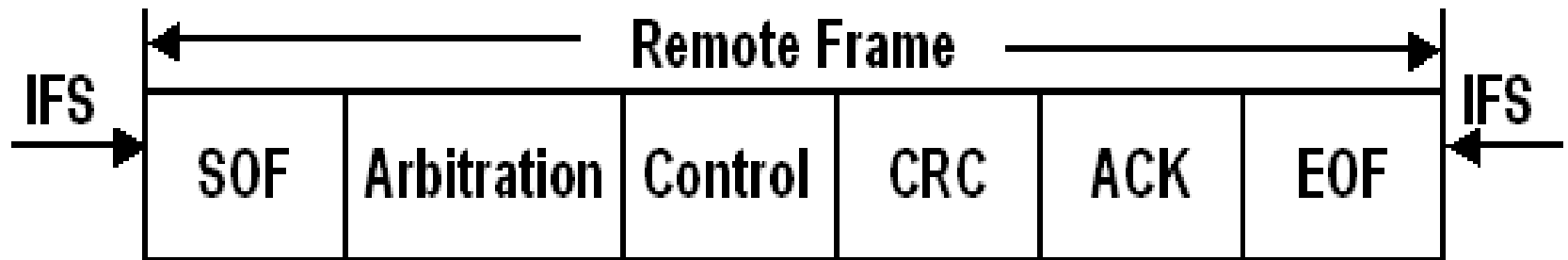
End of Frame

- Each data frame and remote frame is delimited by a flag sequence consisting of seven recessive bits.
- This is a contradiction to the Bit Stuffing technique.



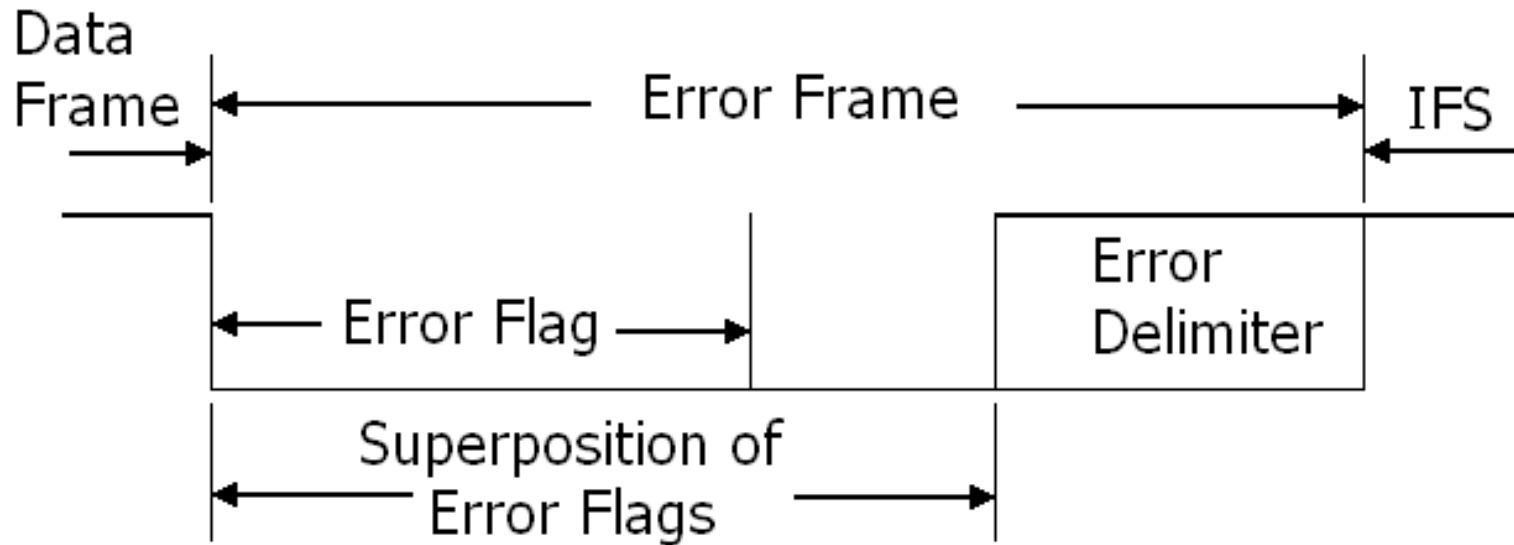
Remote Frame

- A node which needs data from another node (remote node) can request the remote node to transmit data by sending a remote frame.
- It has six fields: Start of frame, Arbitration, Control, CRC, ACK and End of frame (**It has no Data Field**)



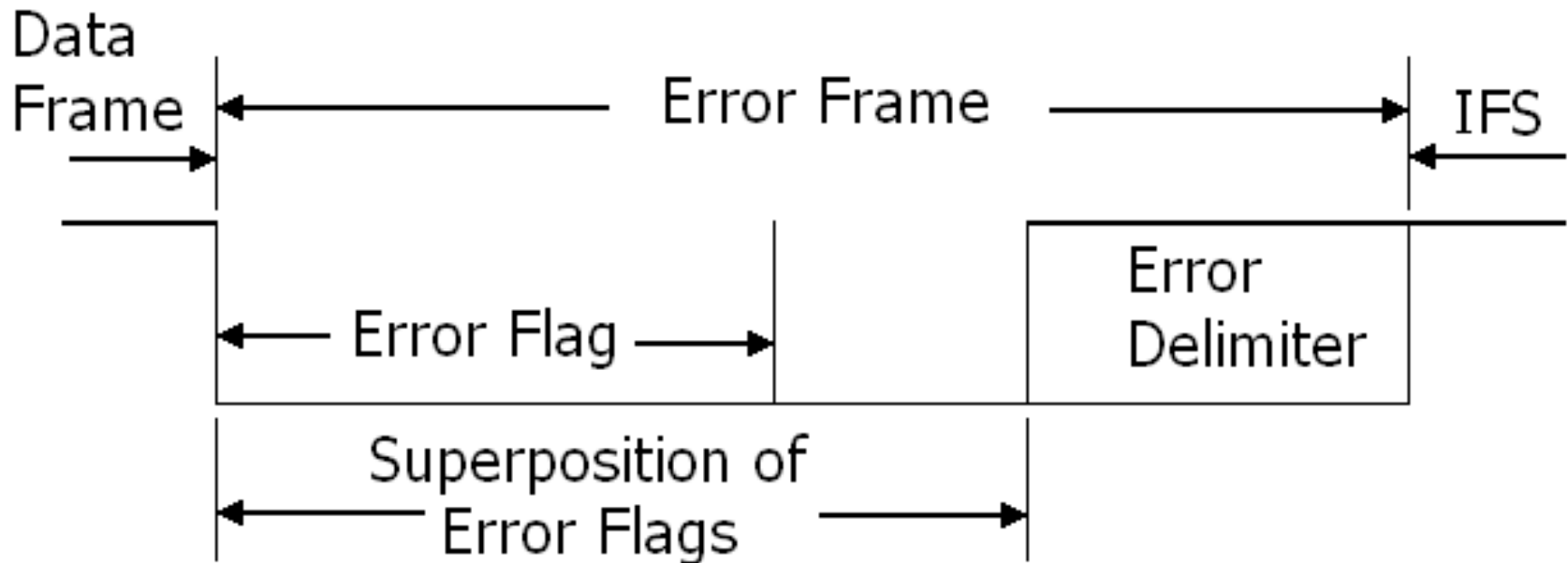
Error Frame

- The Error frame contains two distinct fields.
- The first field is given by the superposition of Error flags contributed from different nodes.



Error Frame

- The second field is the Error delimiter

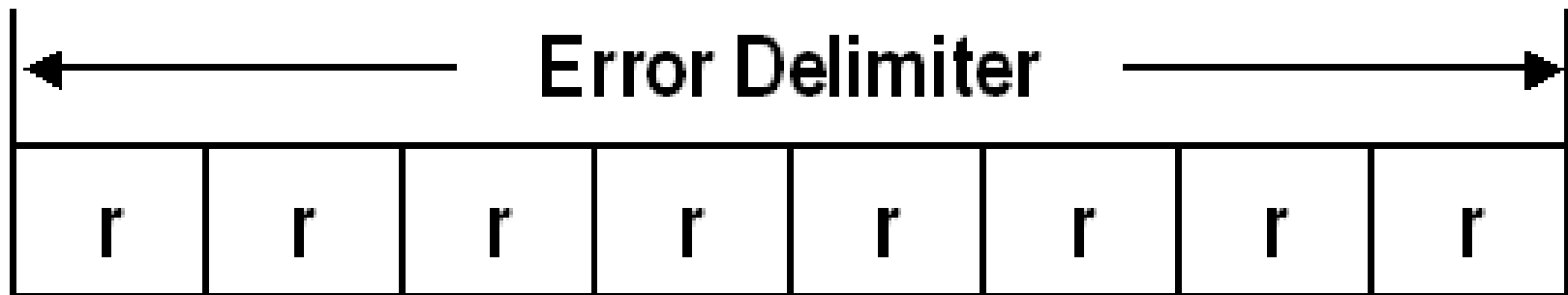


Error Flag

- There are two types of error flags: an Active error flag and a Passive error flag.
- Active error flag consists of six consecutive dominant bits (**d d d d d d**).
- Passive error flag consists of six consecutive recessive bits (**r r r r r r**) unless it is overwritten by dominant bits from other nodes.

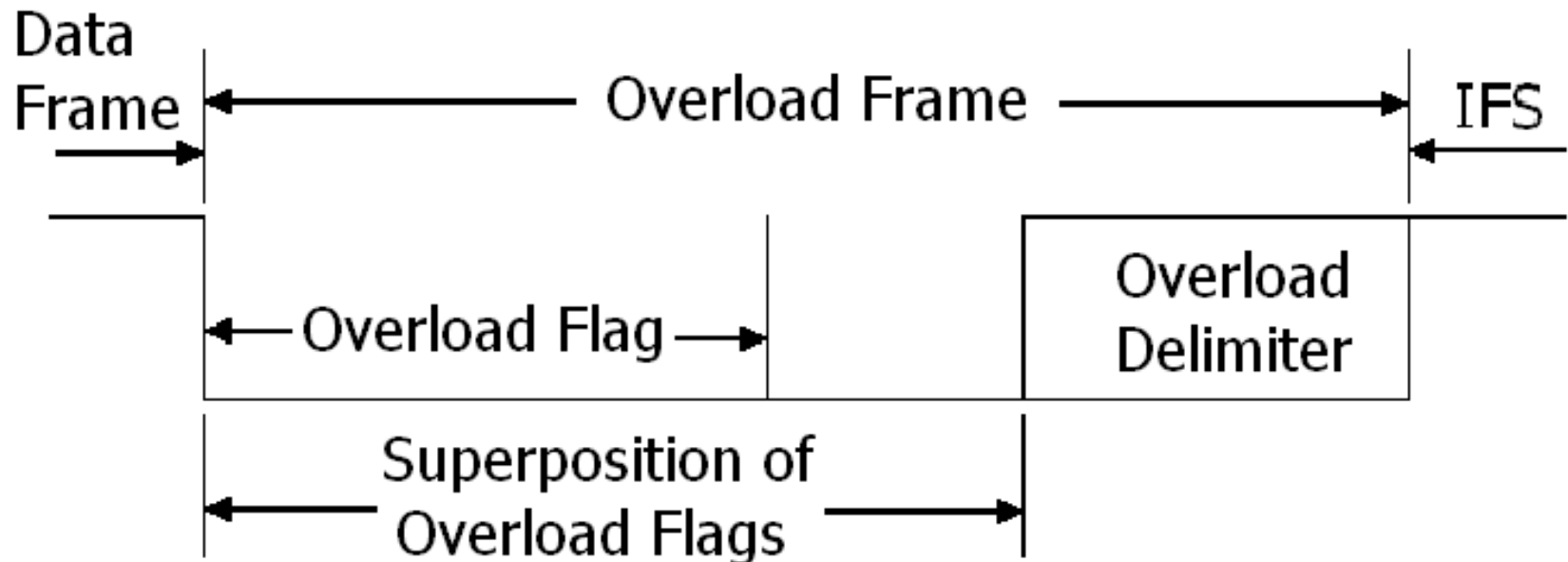
Error Delimiter

- The Error delimiter consists of **eight** recessive bits (**r r r r r r r r**).
- After transmission of an Error Flag each node sends recessive bits and monitors the bus until it detects a recessive bit. Afterwards it starts transmitting seven more recessive bits.



Overload Frame

- The Overload frame contains two fields: Overload flag and Overload delimiter.



Overload Frame

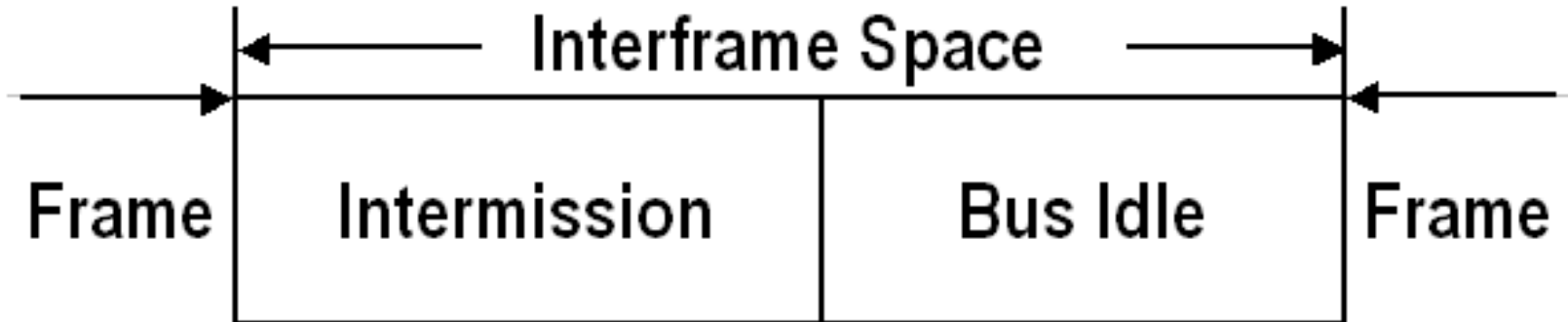
- There are two types of Overload conditions, both of which lead to the transmission of an Overload flag:
 1. Where the internal conditions of a receiver are such that the receiver requires a delay for the next Data frame or Remote frame.
 2. On detection of a dominant bit during INTERMISSION.

Overload Flag and Overload Delimiter

- Overload Flag consists of six dominant bits (**d d d d d d**). It is similar to that of an ACTIVE error flag.
- Overload Delimiter consists of **eight** recessive bits (**r r r r r r r r**).

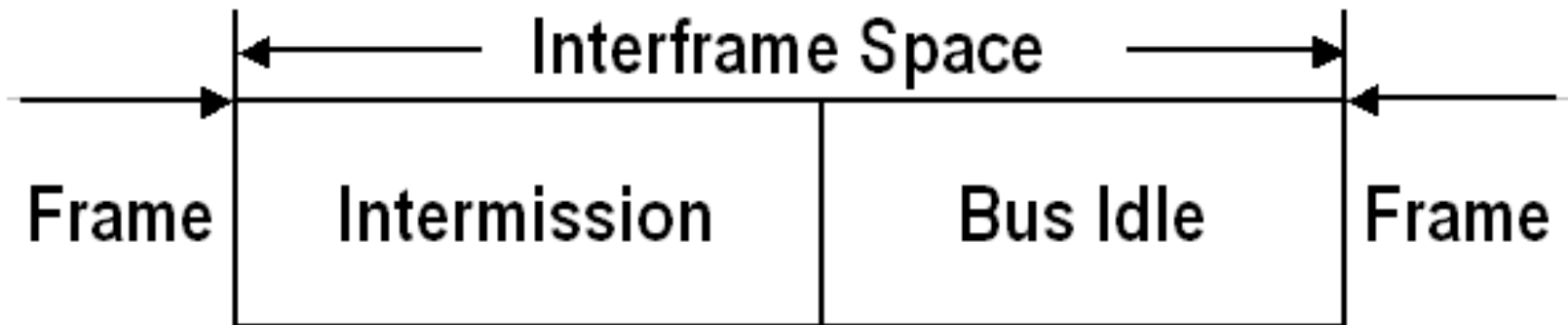
Inter-frame Space

- Data frames and Remote frames are separated from preceding frames by a field called Inter-frame space



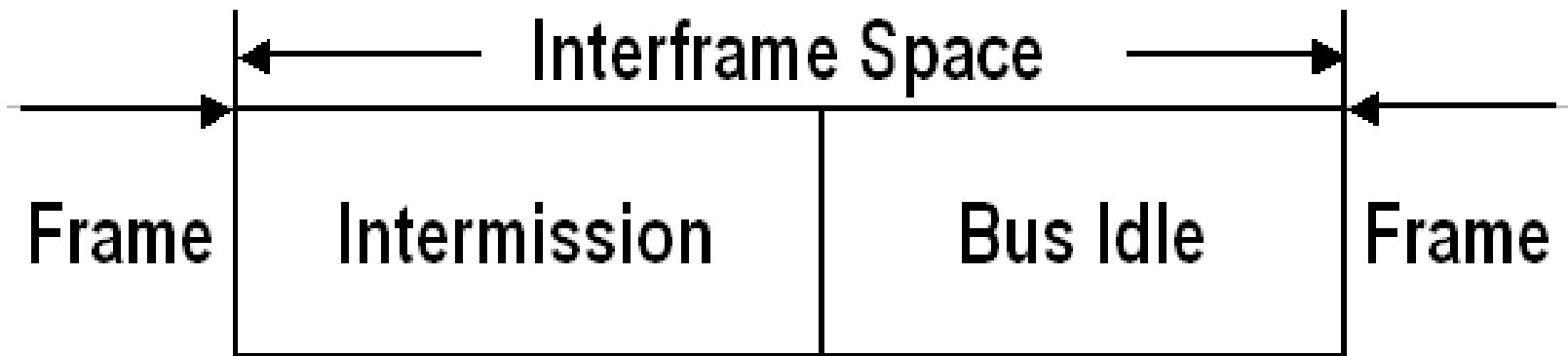
Intermission

- Intermission consists of three recessive bits (**r r r**).
- During Intermission no node is allowed to start transmission of a Data frame or Remote frame. The only action permitted is signaling of an Overload condition.



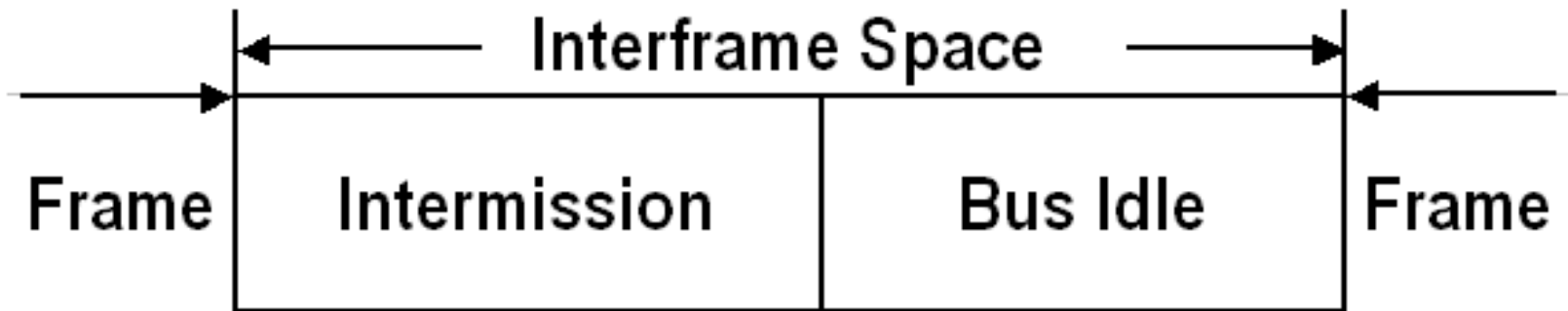
Bus Idle

- The bus idle period may be of arbitrary length.
- The bus is recognized to be free, and any node having something to transmit can start transmission.



Bus Idle

- A message, pending during the transmission of another message, is started in the first bit following the Intermission.
- The detection of a dominant bit on the bus is considered as the Start of a Frame.



Error Detection

- CAN implements five error detection mechanisms.
 - Three at the message level
 1. Cyclic Redundancy Checks (CRC)
 2. Frame Checks
 3. Acknowledgment Error Checks
 - Two at the bit level
 1. Bit Error
 2. Stuff Error

CRC Error

- The CRC sequence consists of the result of the CRC calculation by the transmitter.
- The receivers calculate the CRC in the same way as the transmitter.
- A CRC error is detected if the calculated result is not the same as that received in the CRC sequence.

Frame Error

- If a receiver detects an invalid bit in any one of the following positions, a Form Error (or Format Error) is flagged:
 - CRC Delimiter
 - ACK Delimiter
 - End of Frame Field
 - Intermission Field of Interframe Space (This field must have 3 recessive bits).

Acknowledge Error

- Each receiving node writes a dominant bit into the ACK slot.
- If a transmitter determines that a message has not been acknowledged then an ACK Error is flagged.
- ACK errors may occur because of transmission errors (bits have been corrupted) or there is no operational receivers.

Bit Error

- A node which is sending a bit on the bus also monitors the bus.
- The node must detect, and interpret as a Bit error, the situation where the bit value monitored is different from the bit value being sent.
- An exception to this is the sending of recessive bit during the Arbitration field or during the ACK slot; in this case no Bit error occurs when a dominant bit is monitored.

Stuff Error

- Bit stuffing is used to guarantee enough edges in the NRZ bit stream to maintain synchronization.
- After five identical and consecutive bit levels have been transmitted, the transmitter will automatically inject (stuff) a bit of the opposite polarity into the bit stream.

Stuff Error

- Receivers of the message will automatically delete (destuff) such bits.
- If any node detects six consecutive bits of the same level, a stuff error is flagged.

Error Signaling

- A node detecting an error condition signals this by transmitting an Error flag. An error-active node will transmit an ACTIVE error flag; an error-passive node will transmit a PASSIVE error flag.
- Whenever a **Bit error** is detected by any node, that node will start transmission of an Error flag at the **next bit time**.

Error Signaling

- Whenever a CRC error is detected, transmission of an Error flag will start at the bit **following the ACK delimiter**, unless an Error flag for another error condition has already been started.

Fault Confinement

- A method for discriminating between **temporary** errors and **permanent** failures.
- Temporary errors may be caused by external conditions, voltage spikes, etc.
- Permanent failures may be caused by bad connections, faulty cables, defective transmitters or receivers, or long lasting external disturbances.

Error Counts

- To facilitate fault confinement two counts are implemented in every bus node:
 - Transmit Error Count
 - Receive Error Count
- These counts are modified according to certain rules.

CAN Node Status

- With respect to error confinement, a node may be in one of three states: **Error-Active**, **Error-Passive**, or **Bus-Off**.

Node Status	Value of either Transmit or Receive Error Count
Error-Active	1 to 127
Error-Passive	128 to 255
Bus-Off	> 255

Bus-Off

- A bus-off node is not allowed to have any influence on the bus.

Error-Passive → Error-Active → Normal

- Transmit Error Count is decremented by 1 after a successful transmission.
- Receive Error Count is decremented by 1 after a successful reception.



Requirements of a CAN Controller

- Simple user interface to the CPU
- Message filtering and buffering
- Protocol handling
- Physical layer interface

Problem-1

- The following sequence of bits are the first 40 bits of a data frame that a node received from the CAN bus.

**000001001100000100100000101
1000001100110**

- a) Show the value of the 11-bit identifier in decimal.
- b) Show the 6-bit control field in binary.
- c) Determine the length of the data field in bytes.
- d) Show the first two data bytes in hex.

Solution of Problem-1

- Identify the Stuff Bits

**000001001100000100100000101
1000001100110**

- In the above sequence, the underlined bits are the stuff bits.
- The destuffed bit sequence is as follows:

**000000011000000010000001100
000100110**

Solution of Problem-1

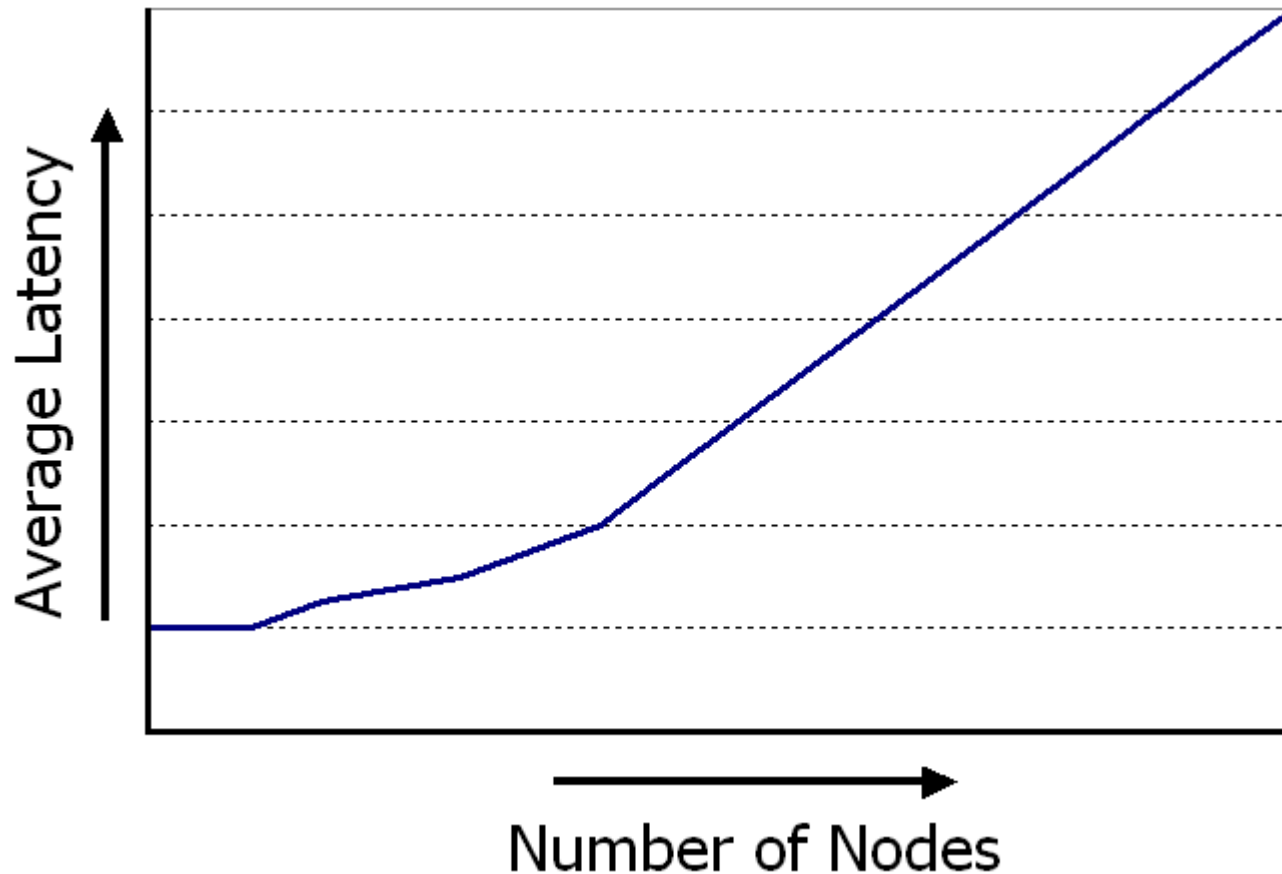
0 00000011000 0 000100
SOF Identifier RTR Control

00001100
1st Data Byte

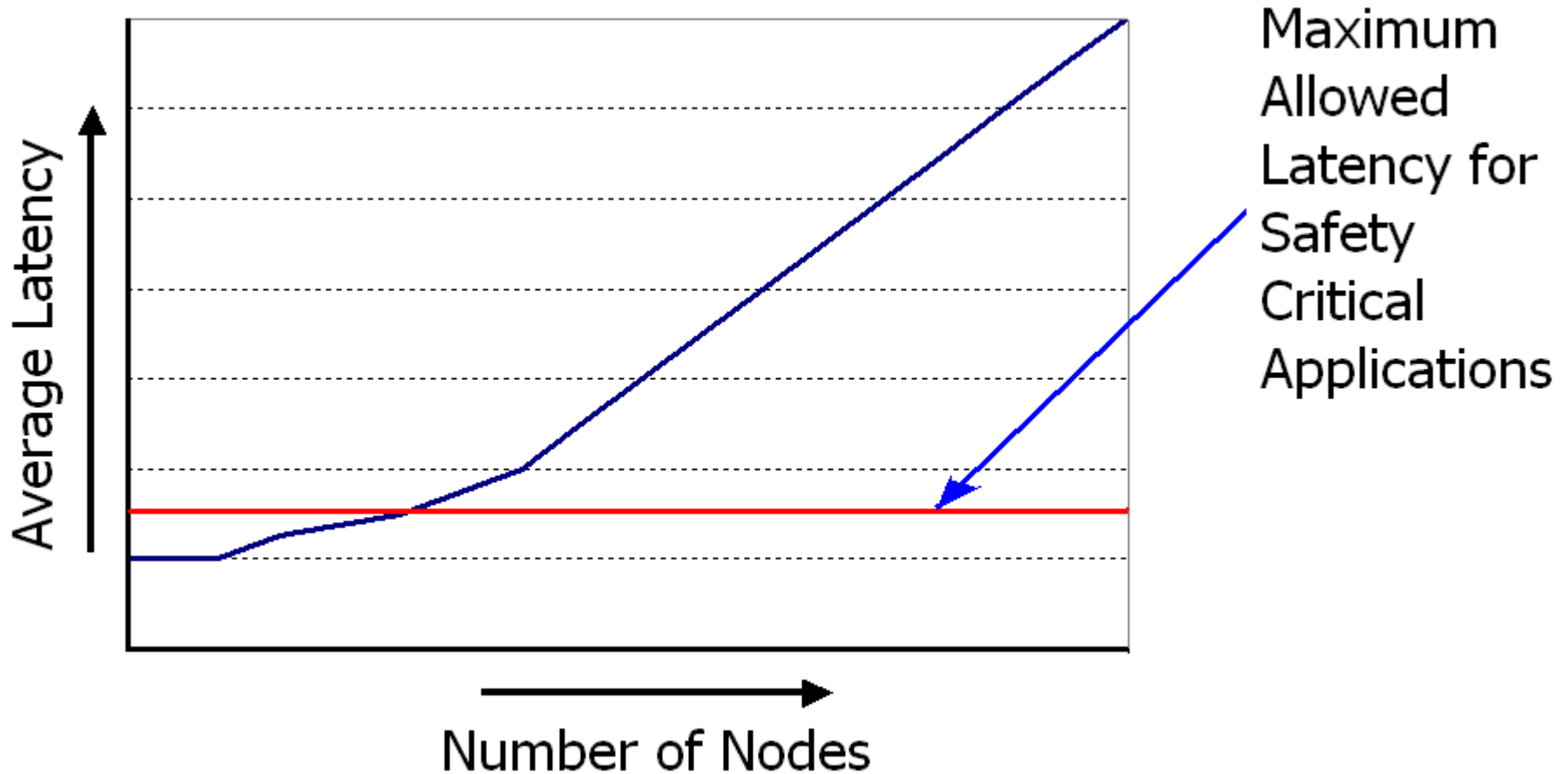
00010011
2nd Data Byte

- a) Identifier = 24, b) Control = 000100
- c) Length of Data = 4 bytes
- d) First two data bytes are: \$0C and \$13

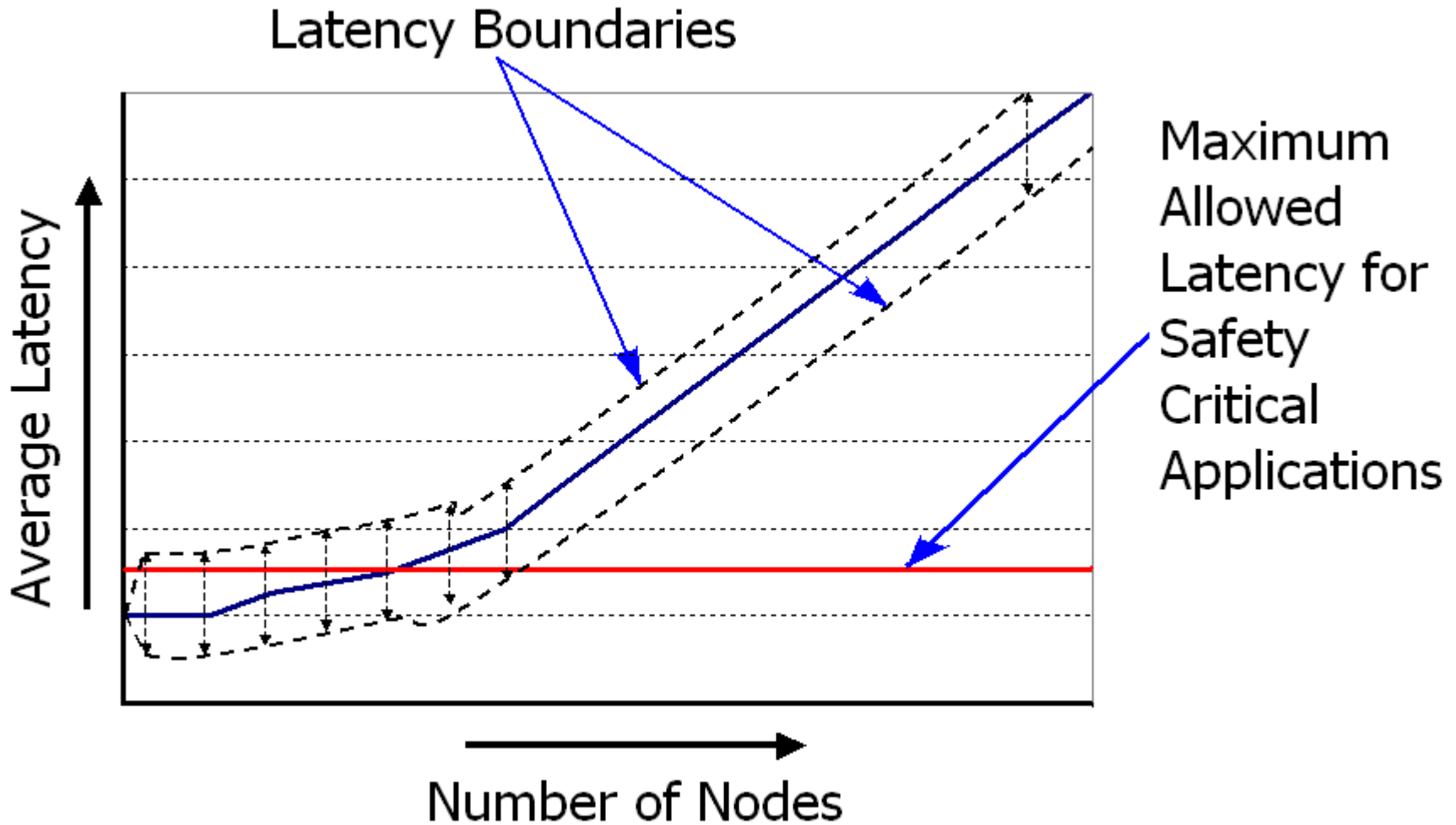
Average Message Latency Versus Number of Nodes



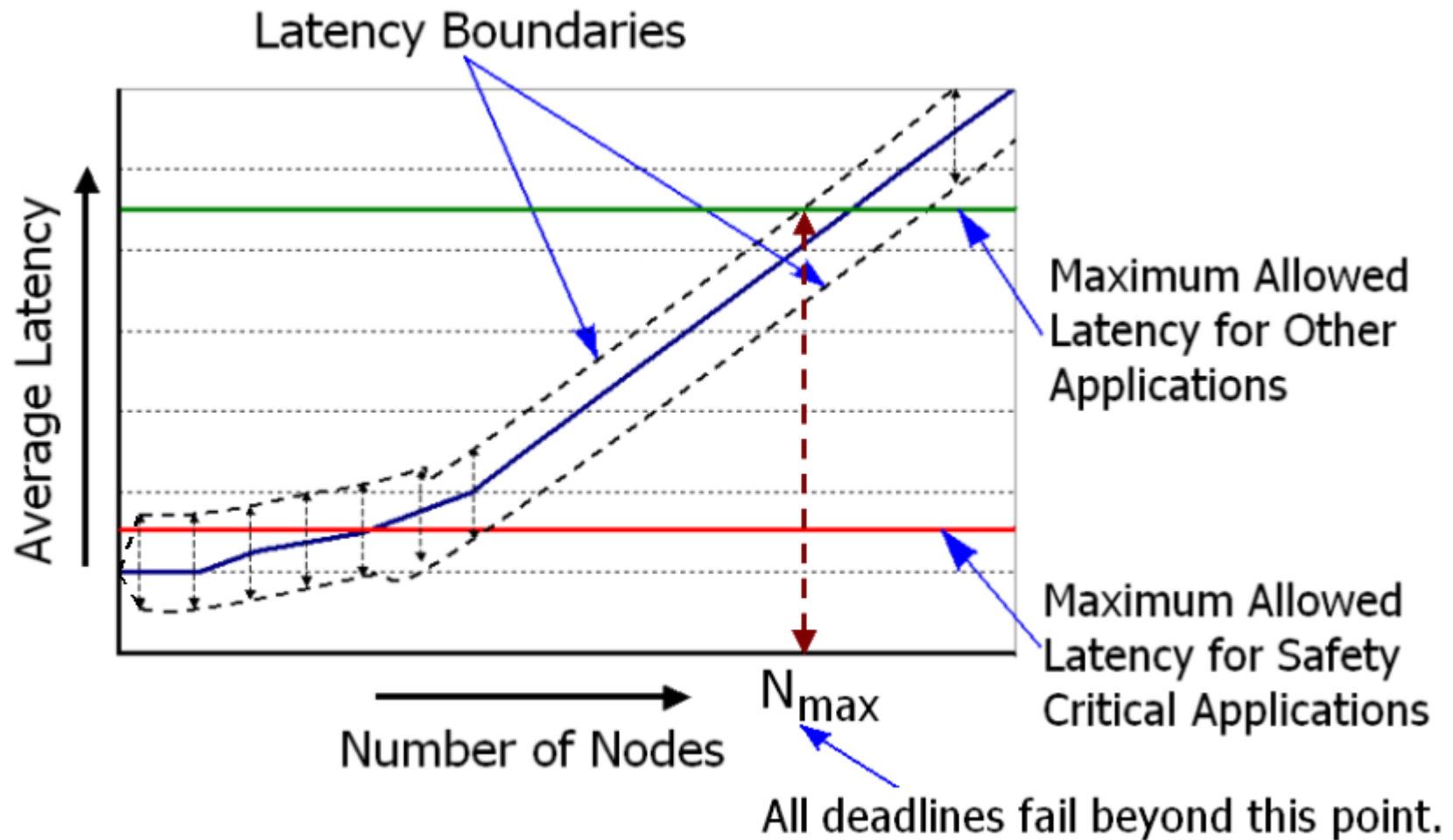
Average Latency and Maximum Allowed Latency



Latency Boundaries



Failure of all Deadlines



Need for Time Triggered Protocols

- Safety critical messages must meet their deadlines even at maximum bus load.

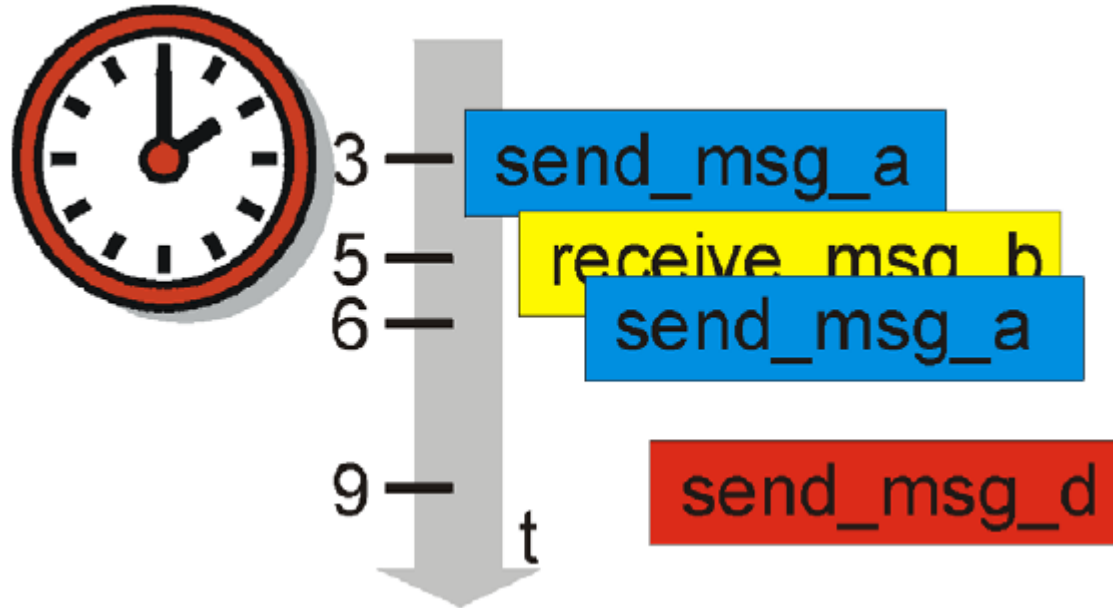
Time Triggered CAN (TTCAN)

- TTCAN is a higher layer protocol above the standard CAN protocol.
- A message that needs a guaranteed latency can be transmitted at a specific time slot, without competing with other messages for the bus.

Time Triggered CAN (TTCAN)

- The activities of a TTCAN system are determined by the progression of a **globally synchronized** time.

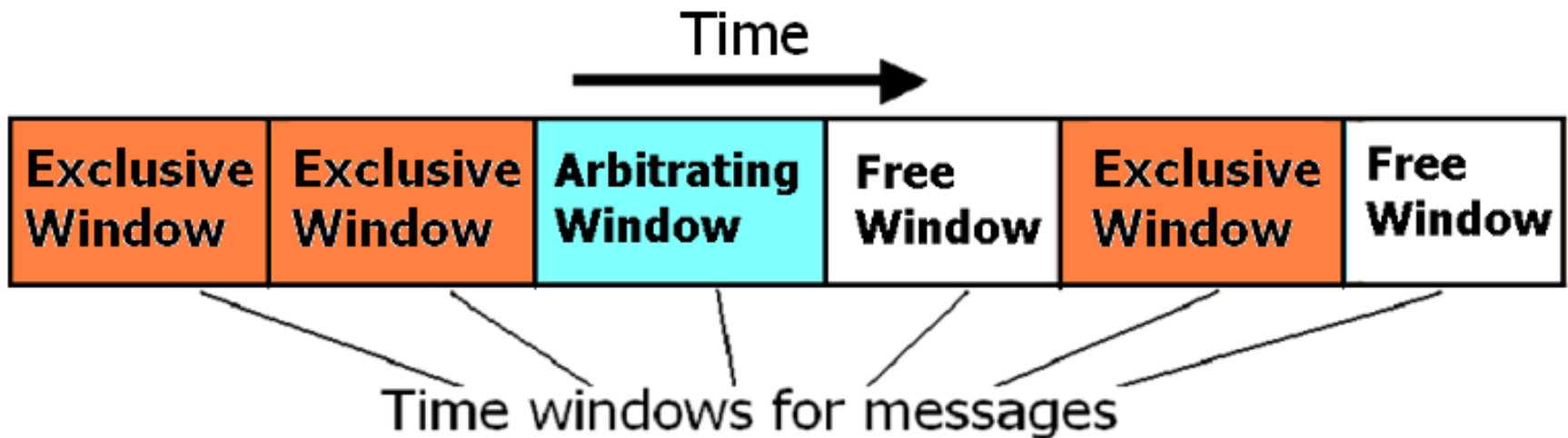
TTCAN Principle



- Message "a" is sent if the system clock reaches 3 and 6 while message "b" is sent at 5, and message "d" is sent at 9.

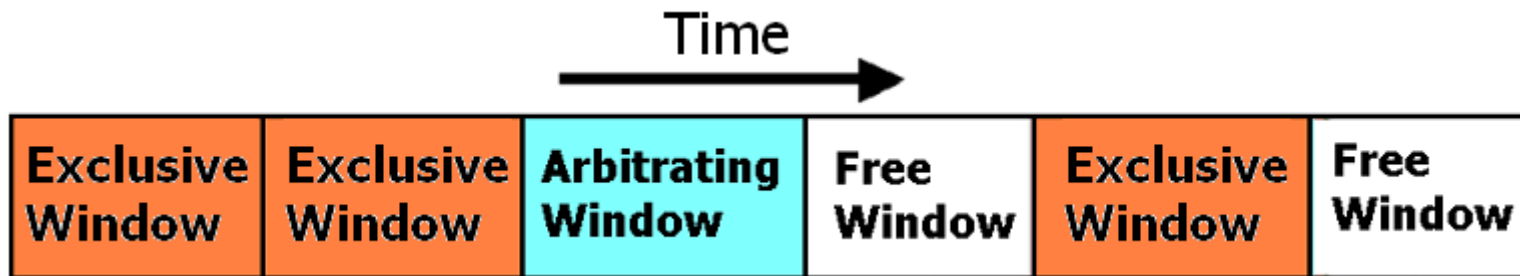
TTCAN Time Windows

- There are three time windows in a TTCAN system:
 - Exclusive Window.
 - Arbitrating Window.
 - Free Window



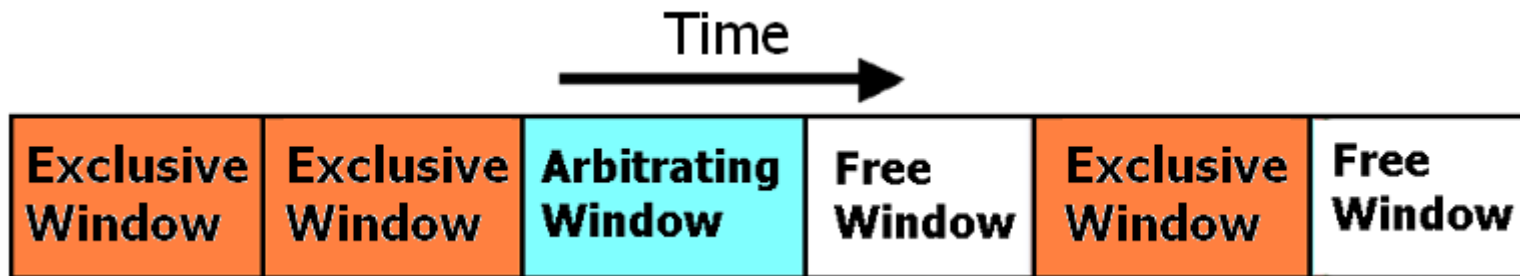
Exclusive Window

- It is used by a message that needs guaranteed latencies.
- A particular Exclusive Window is reserved for a particular message.
- Thus, the message does not have to compete for the bus.



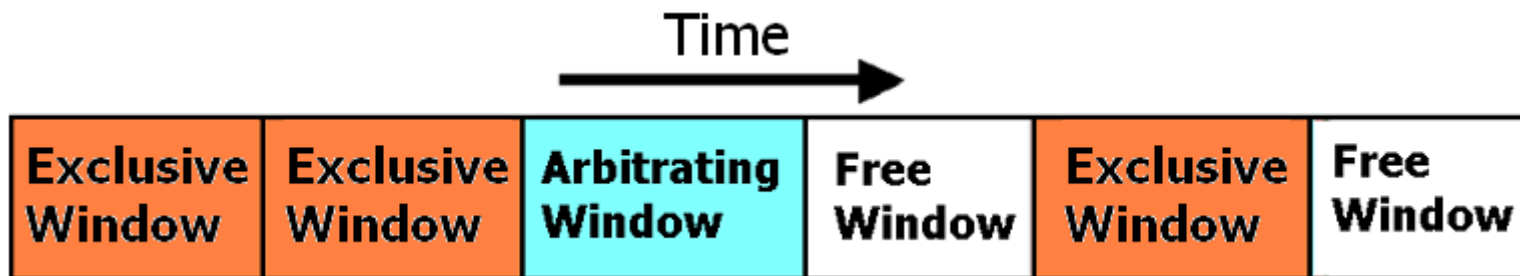
Arbitrating Window

- It is used by messages that can tolerate latency jitters.
- Like the standard CAN system, various messages compete for the bus during this time window.
- The highest priority message (among the competing messages) get the bus.



Free Window

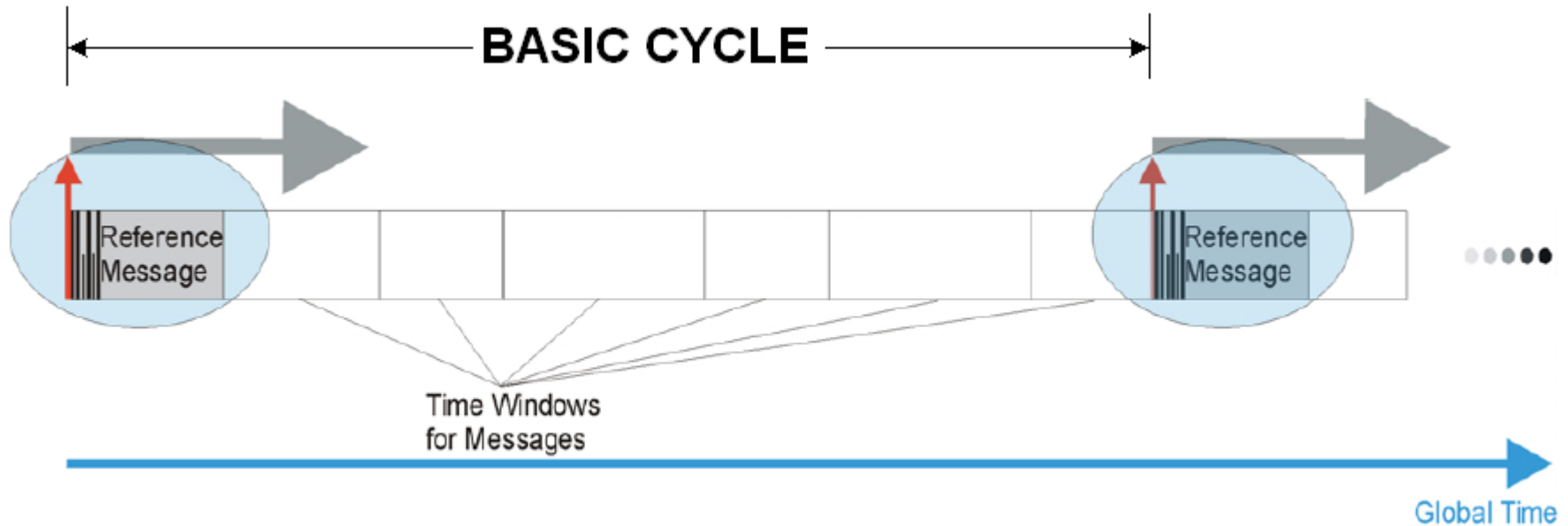
- It is reserved for the future expansion of the system.
- Depending upon the need (during system expansion) some Free Windows can be converted to new Exclusive Windows and others can be converted to new Arbitrating Windows.



Master Node

- A node called the Master Node is responsible for maintaining global time.
- The Master Node periodically broadcasts its local time (global time for the entire system) through a message called the **Reference Message**.

Reference Message



- All nodes read the global time from the reference message and then synchronize their own local time with the global time.

Local Offset

- A node captures its own local time at the beginning of a Reference Message.
- The node then extracts the global time from the Reference Message.
- It then computes the local offset as:
Local Offset = Global Time – Local Time.
- After that, the node keeps track of the current Global Time by adding the Local Offset with its current Local Time.

Keeping Track of the Global Time by a Node

$$\text{Current Global Time} = \text{Current Local Time} + \text{Local Offset}$$

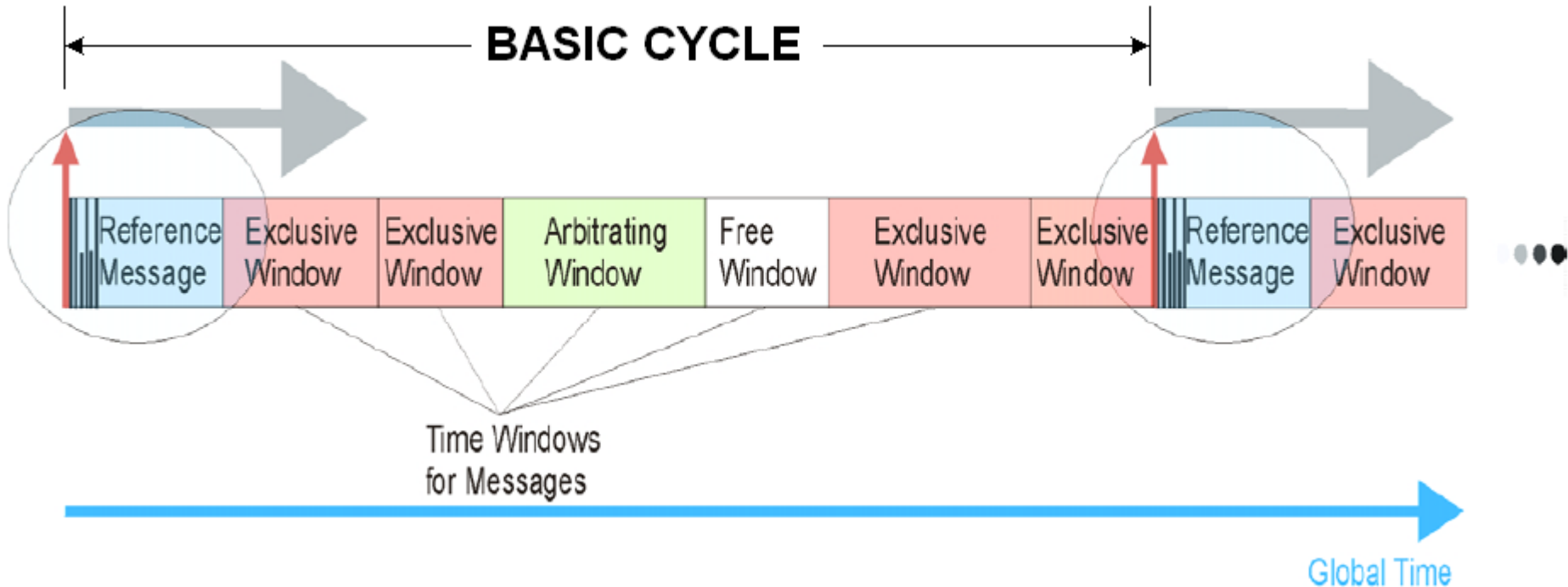
Potential Master Nodes

- There are several potential master nodes in a TTCAN system.
- If the potential master nodes detect the absence of the Reference Message within a timeout period, then they understand that the Master Node is dead.

Potential Master Nodes

- The potential masters then compete for the bus to become the new master.
- The highest priority active potential master will become the new master of the system.
- The new master will then start broadcasting reference messages.

Basic Cycle

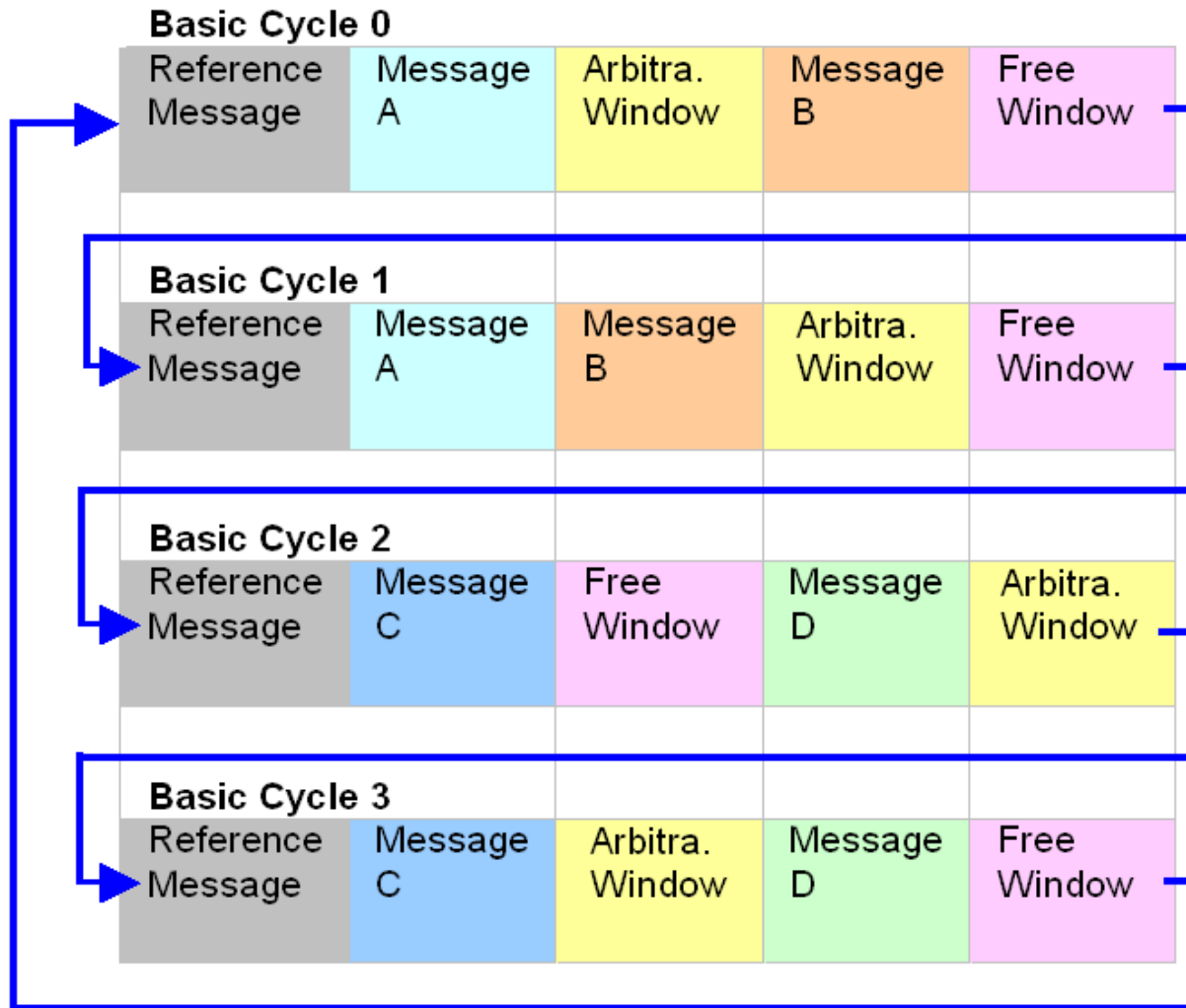


- The period between two consecutive reference messages is called the **basic cycle**.

The System Matrix of a TTCAN System

- Several basic cycles are connected to build a matrix cycle called the **System Matrix**.
- This allows to combine multiple sending patterns, e.g. sending every basic cycle, sending every second basic cycle, or sending only once within the whole system matrix.

System Matrix



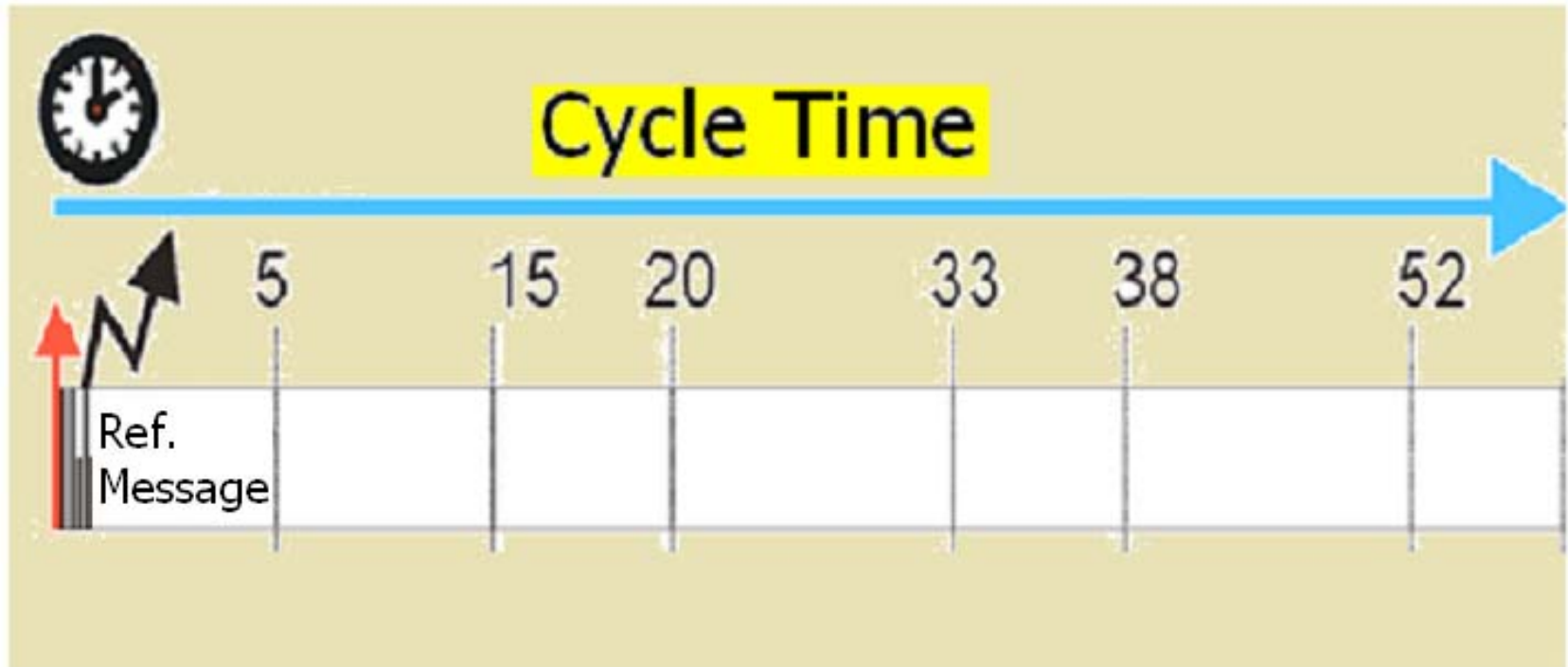
Cycle Time

- The cycle time at a node is defined as:

Cycle Time = Current Local time – The Local Time when the last Reference Message was received

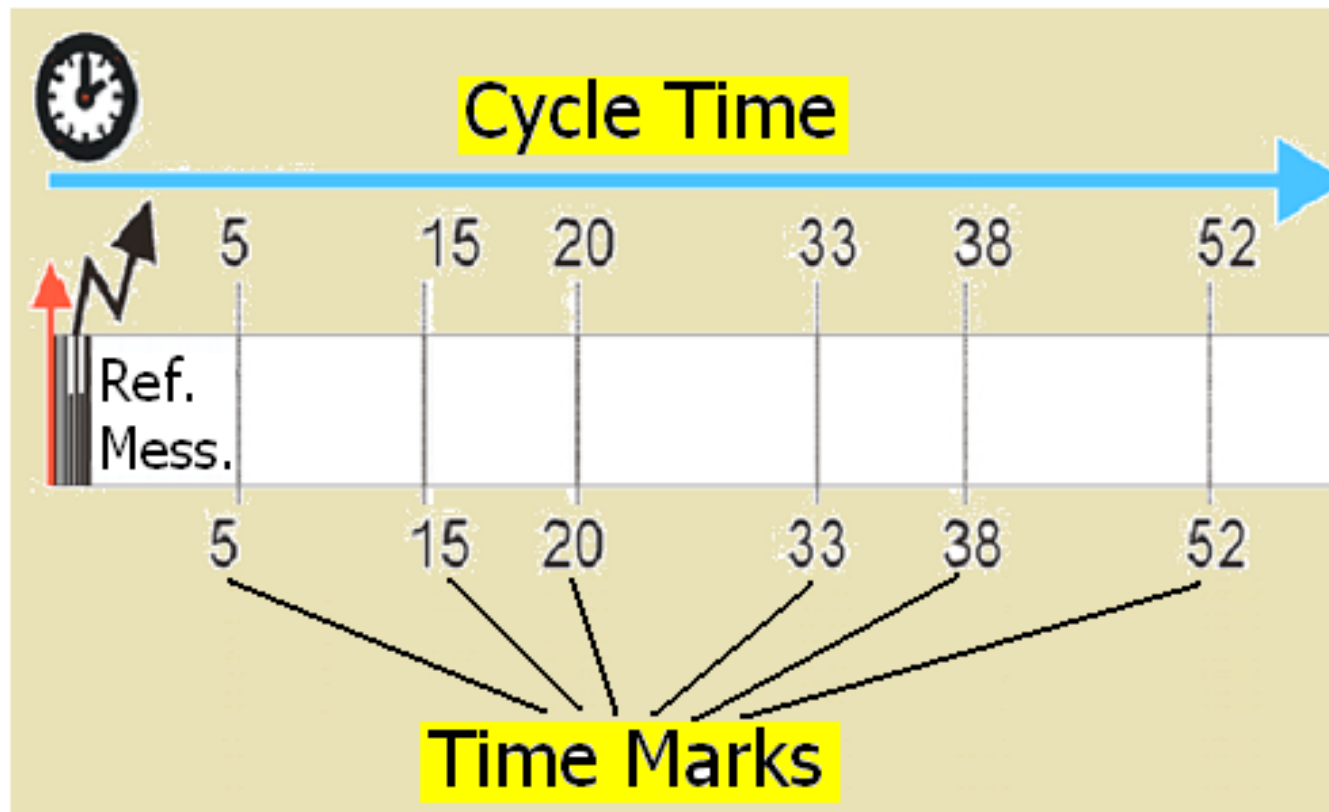
- In other words, it is the offset of the local time with respect to the beginning of the last Reference Message

Cycle Time



Time Marks

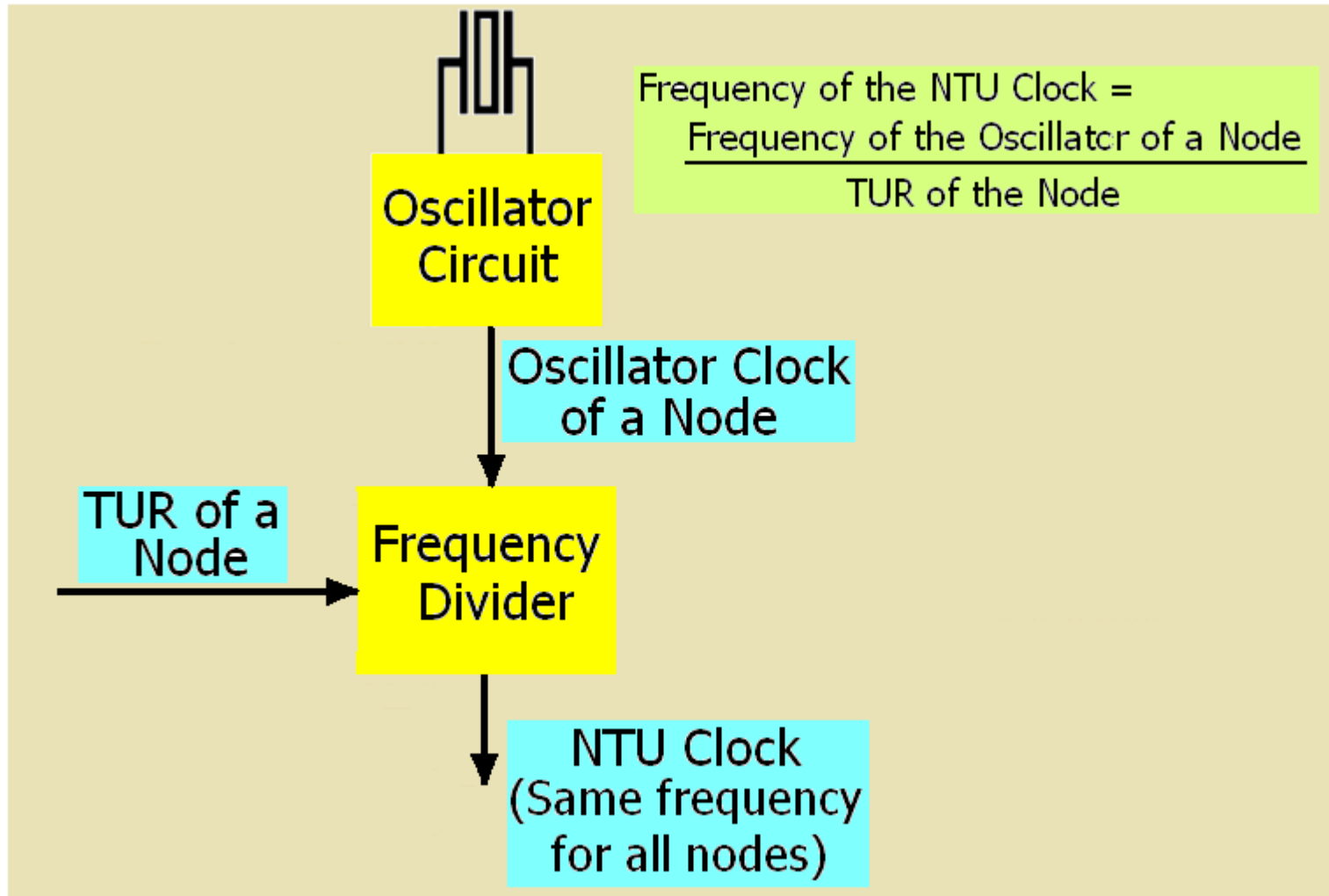
The Time Marks specify the beginning of the TTCAN Windows.



NETWORK TIME UNIT (NTU)

- The granularity of any timing information within a TTCAN system is the Network Time Unit (NTU).
- All time parameters in a TTCAN system are measured in terms of the NTU.
- For example, the Current Global Time is 620 NTUs or the Current Cycle Time at a node is 20 NTUs

NETWORK TIME UNIT (NTU)



NETWORK TIME UNIT (NTU)

- To establish a system wide NTU, each local node has to divide its oscillator frequency by a ratio called the *Time Unit Ratio (TUR)*.
- For a given value of NTU, the value of TUR will vary from node to node depending upon the oscillator frequency of the node.

TURs of Different Nodes

- Assume that $NTU = 0.25$ micro sec. Nodes 1, 2, 3 and 4 have oscillators of frequencies 8MHz, 6MHz, 10MHz and 12MHz, respectively. What are the values of TURs of different nodes?
- Since $NTU = 0.25$ micro sec, the frequency of NTU clock is 4MHz.

TURs of Different Nodes

- TUR of Node-1 = $8\text{MHz}/4\text{MHz} = 2$
- TUR of Node-2 = $6\text{MHz}/4\text{MHz} = 1.5$
- TUR of Node-3 = $10\text{MHz}/4\text{MHz} = 2.5$
- TUR of Node-4 = $12\text{Hz}/4\text{MHz} = 3$
- However, if off-the-shelf micros are used to build a TTCAN system, then it might be difficult to have TUR values like 1.5, 2.5 and 3. Normally the Timers of a micro support pre-scale values 1,2,4,8,16, . . and so on.

Retransmission of a Faulty Message is not allowed in a TTCAN System

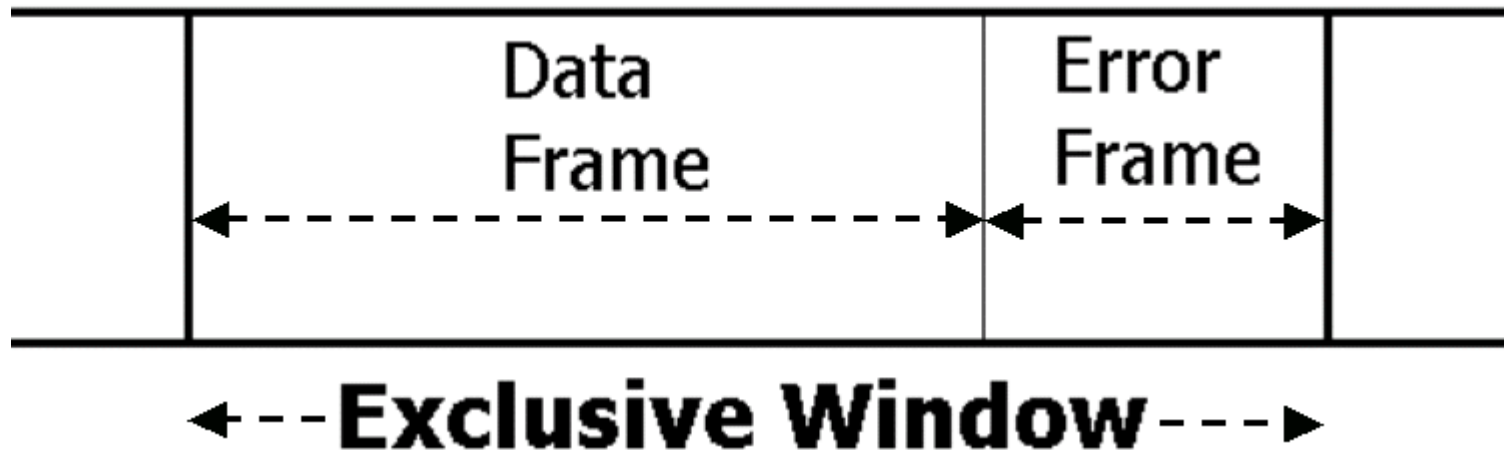
- In case of a fault, retransmission of a message is not allowed through any windows (Exclusive or Arbitrating).
- This is due to the fact that the retransmitted message may cross over the boundary of the current window.

Retransmission of a Faulty Message

- Thus, in a TTCAN system, if a safety critical message can not go through its Exclusive Window, then it will be delayed significantly.
- The message that could not go will have to wait until its next Exclusive Window.

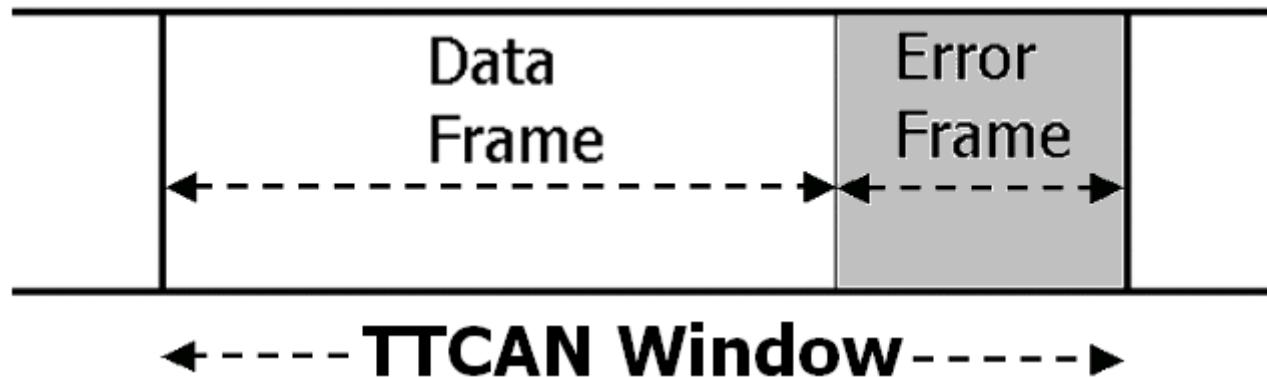
Length of a TTCAN Window

- A TTCAN window must be long enough to carry its message plus possible error flags



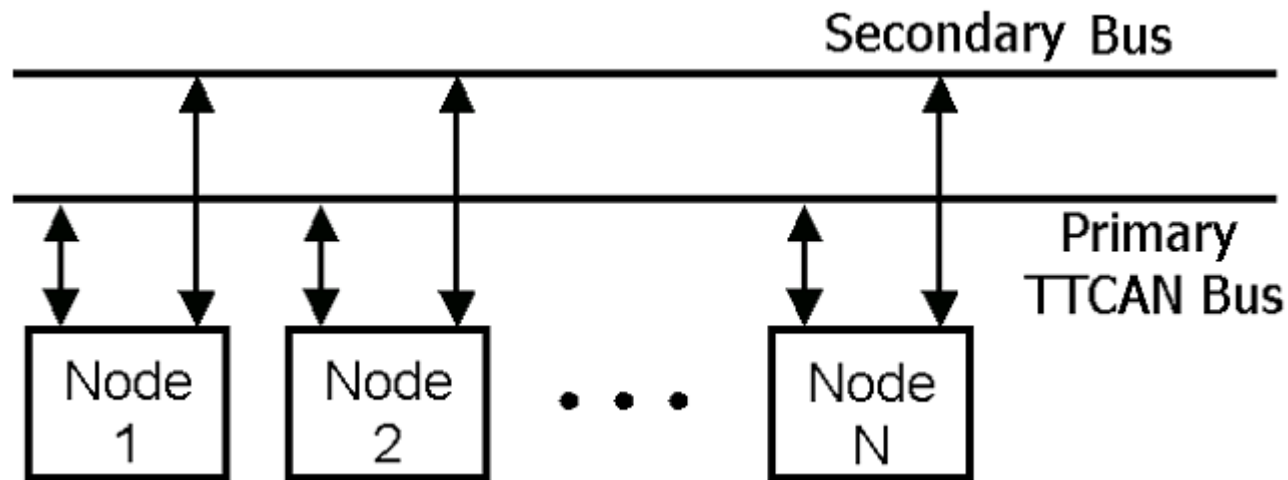
Wastage of Some Bus Bandwidth

- Since every TTCAN window has to keep some space for transmitting an Error Frame, some bus bandwidth is wasted.



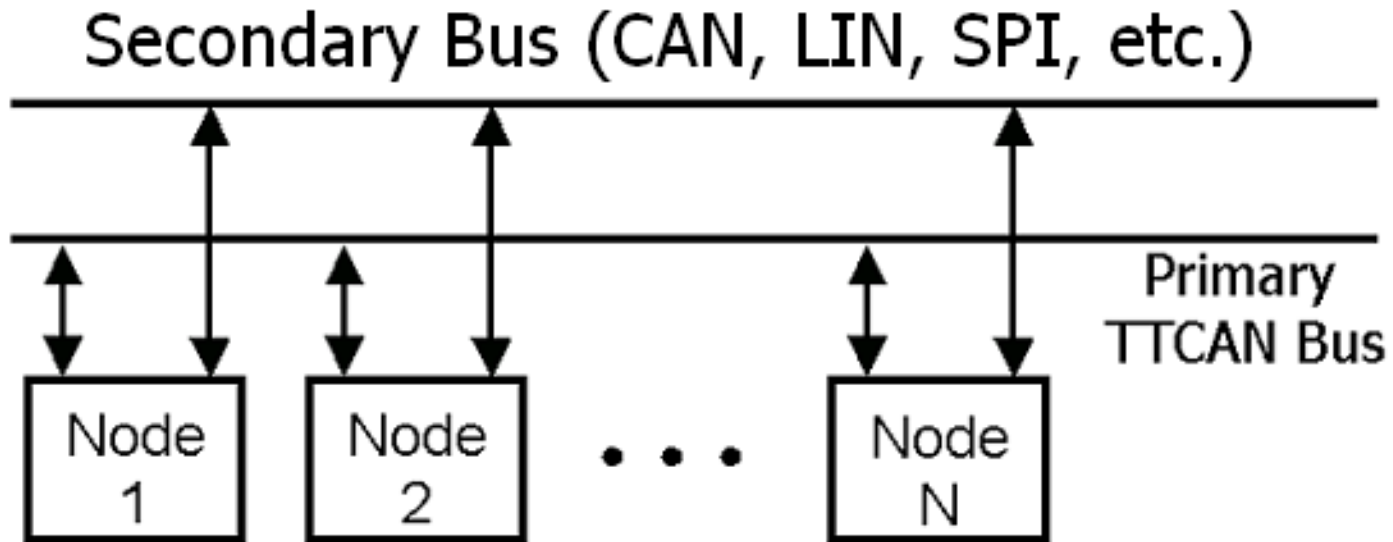
Fault Tolerant TTCAN

- Safety critical messages could be significantly delayed if any faults occur in a TTCAN system.
- Thus, a fault tolerant design is desirable.

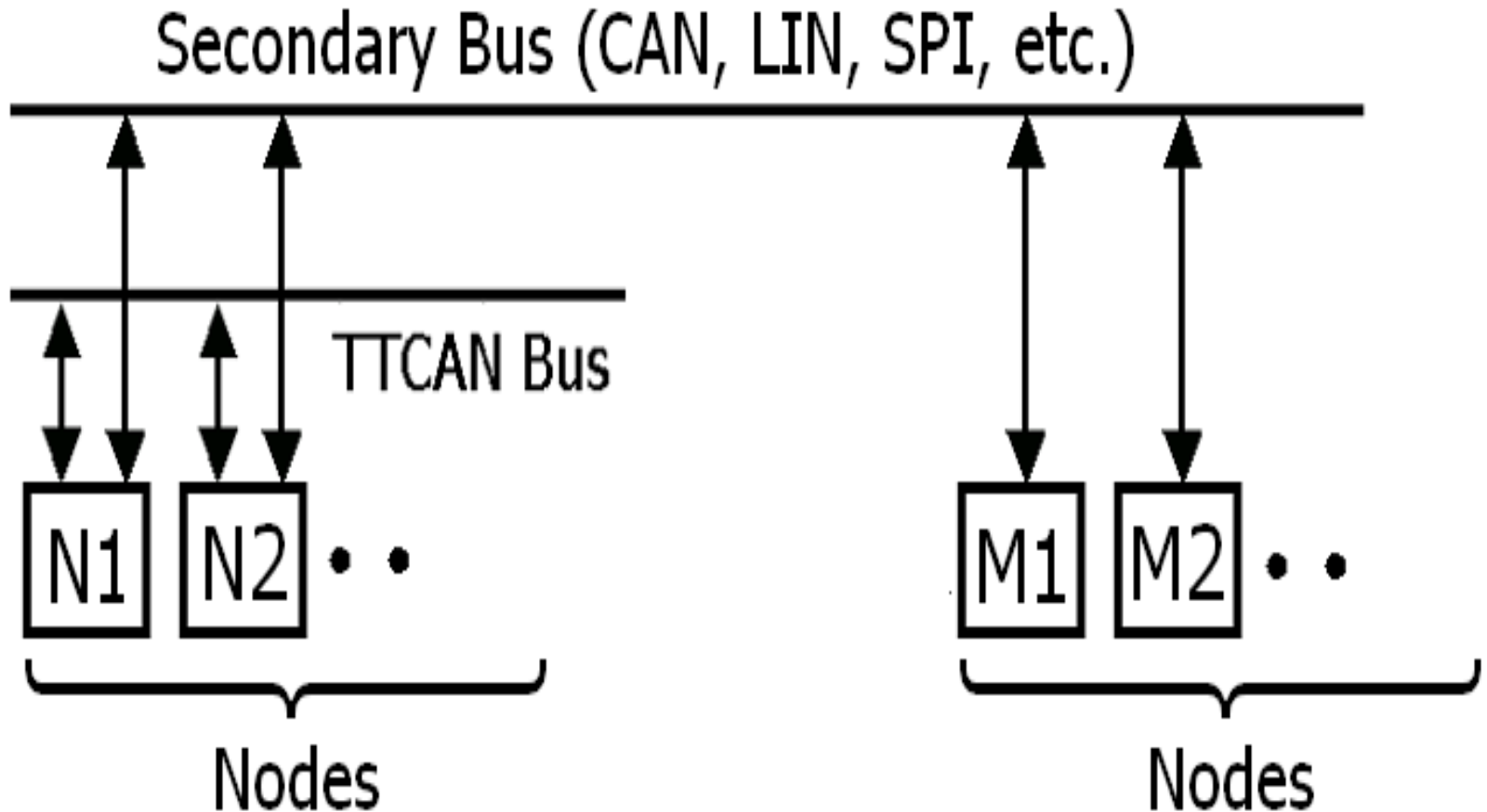


Secondary Bus

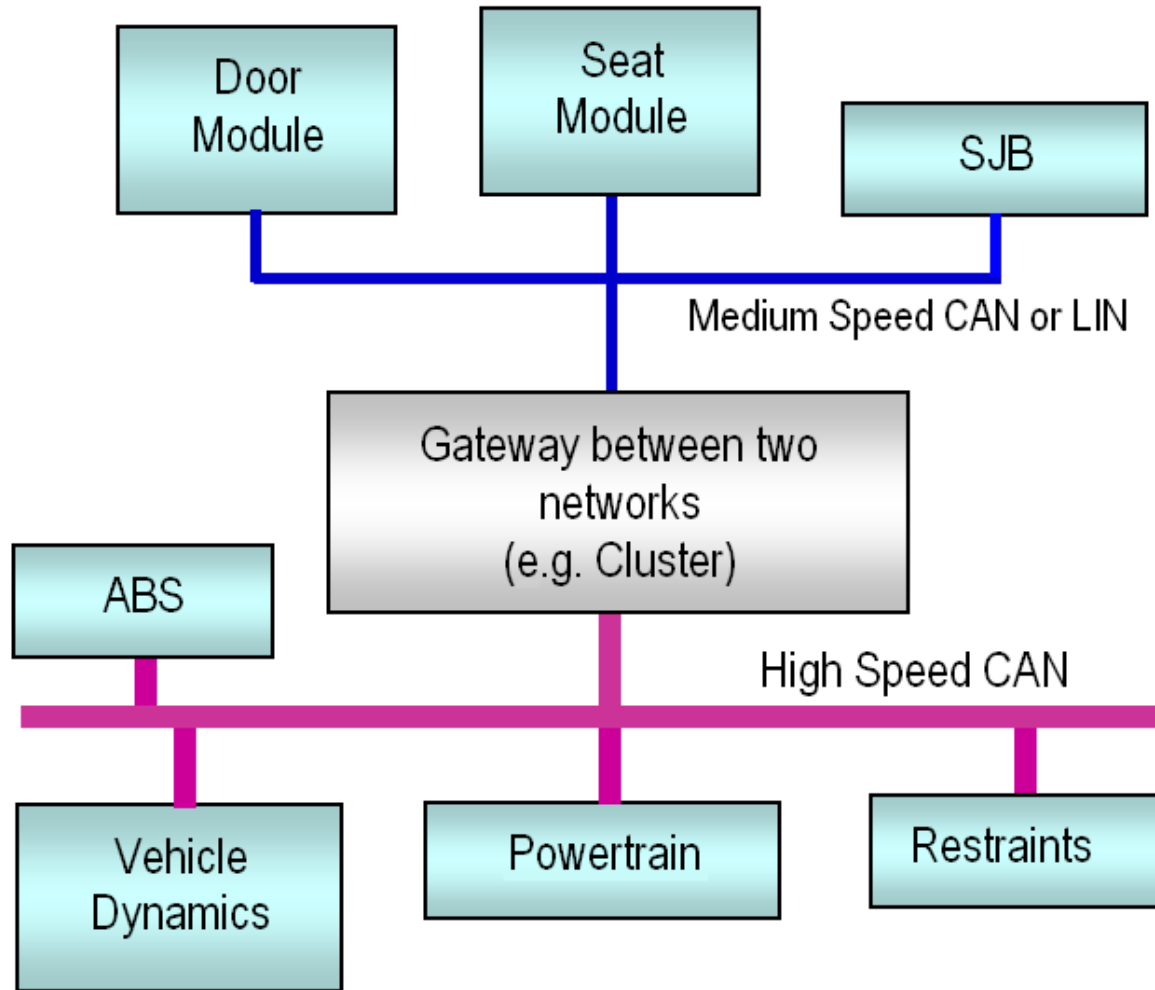
- The secondary bus could be a standard CAN or any other type of bus.
- It could be even shared by other systems.



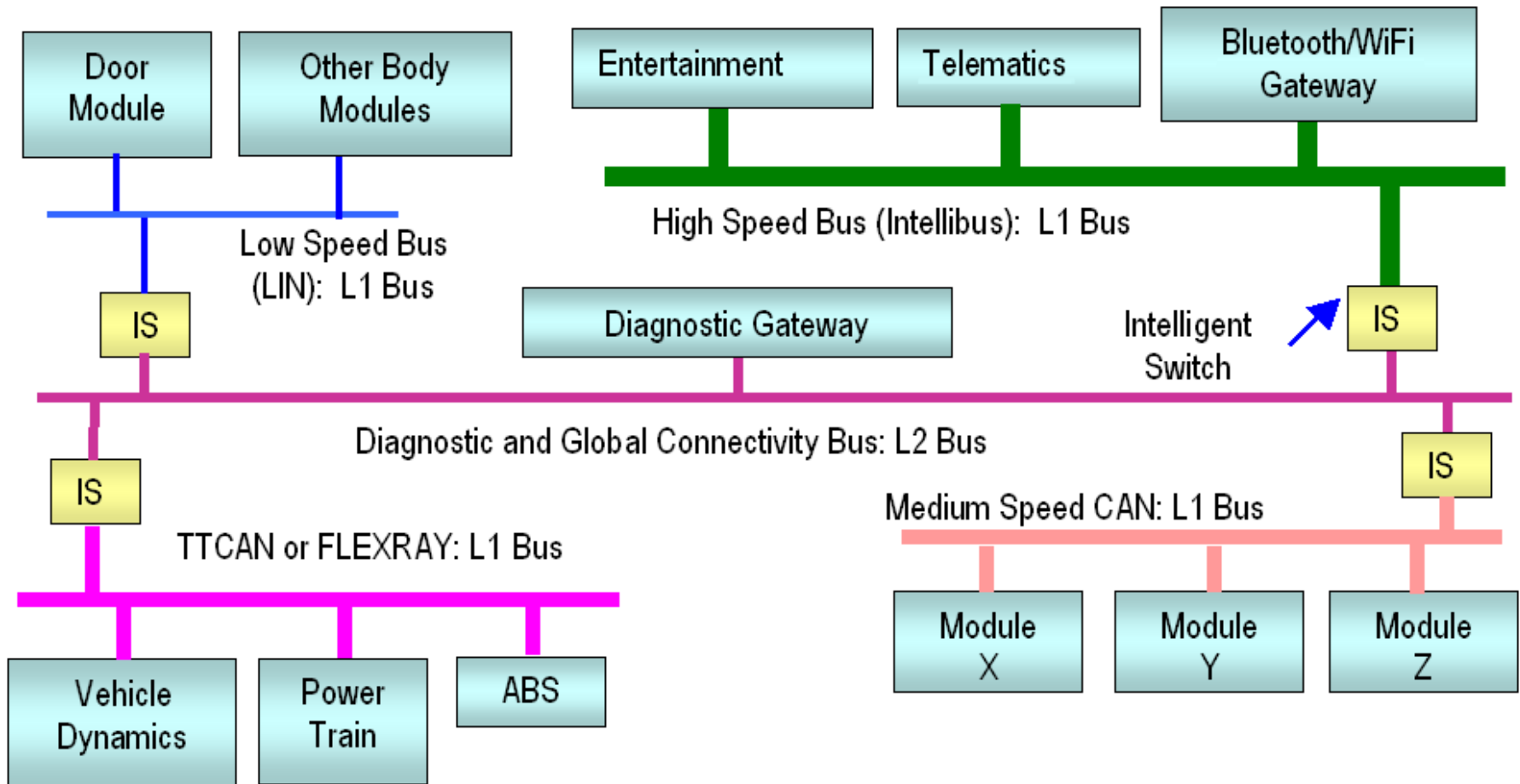
Shared Secondary Bus



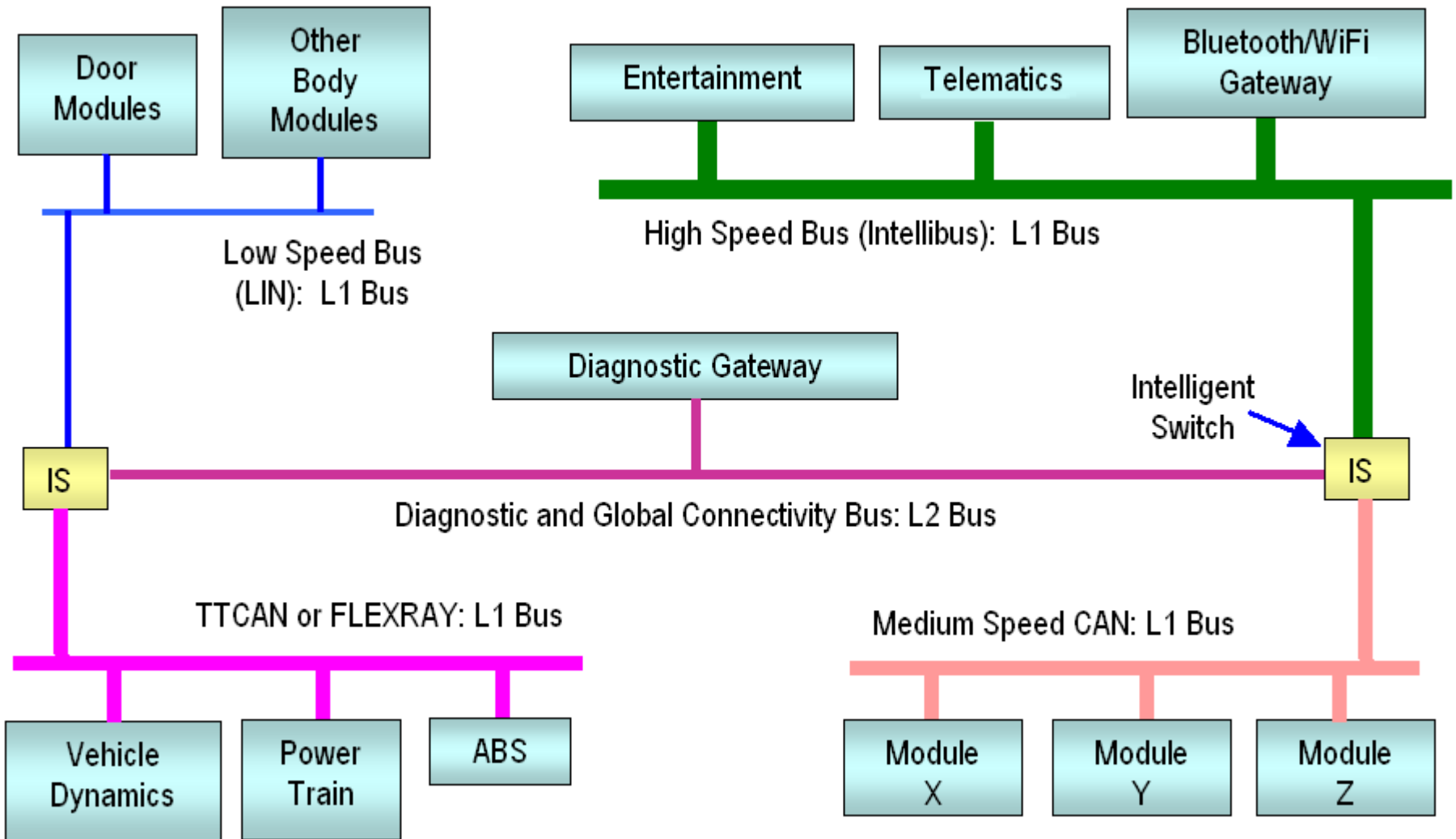
Current In-Vehicle Network



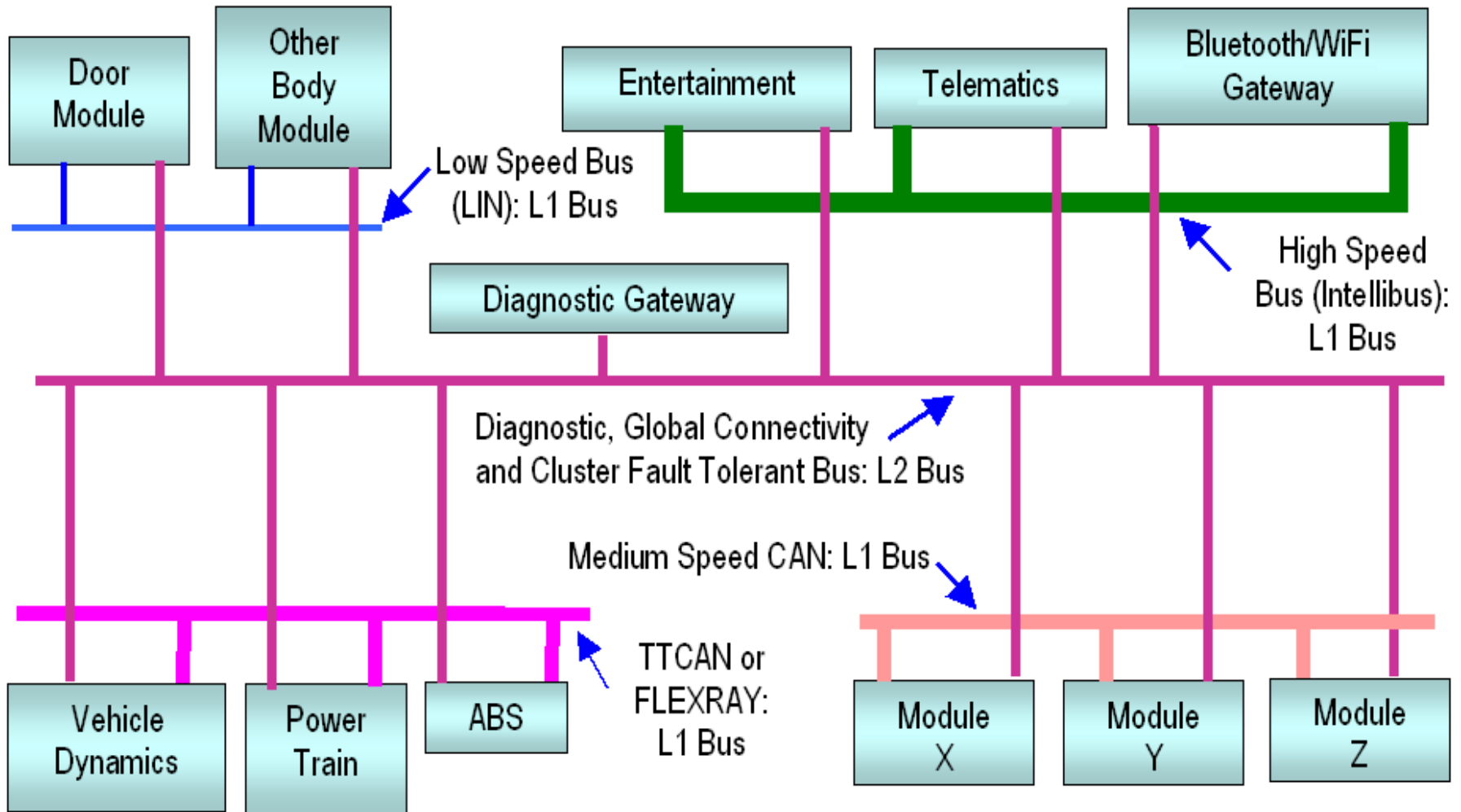
Hierarchical Bus System



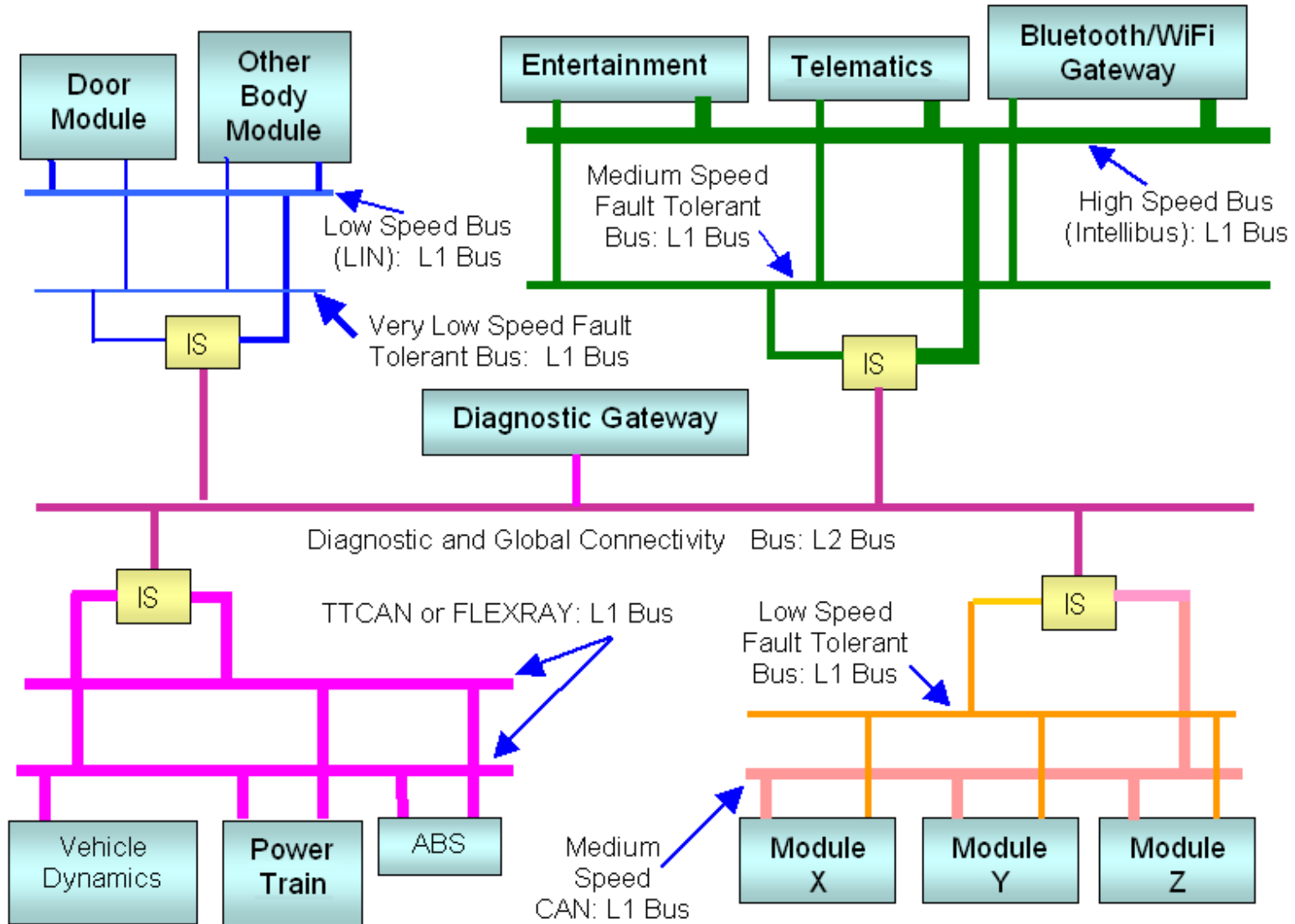
Hierarchical Bus System



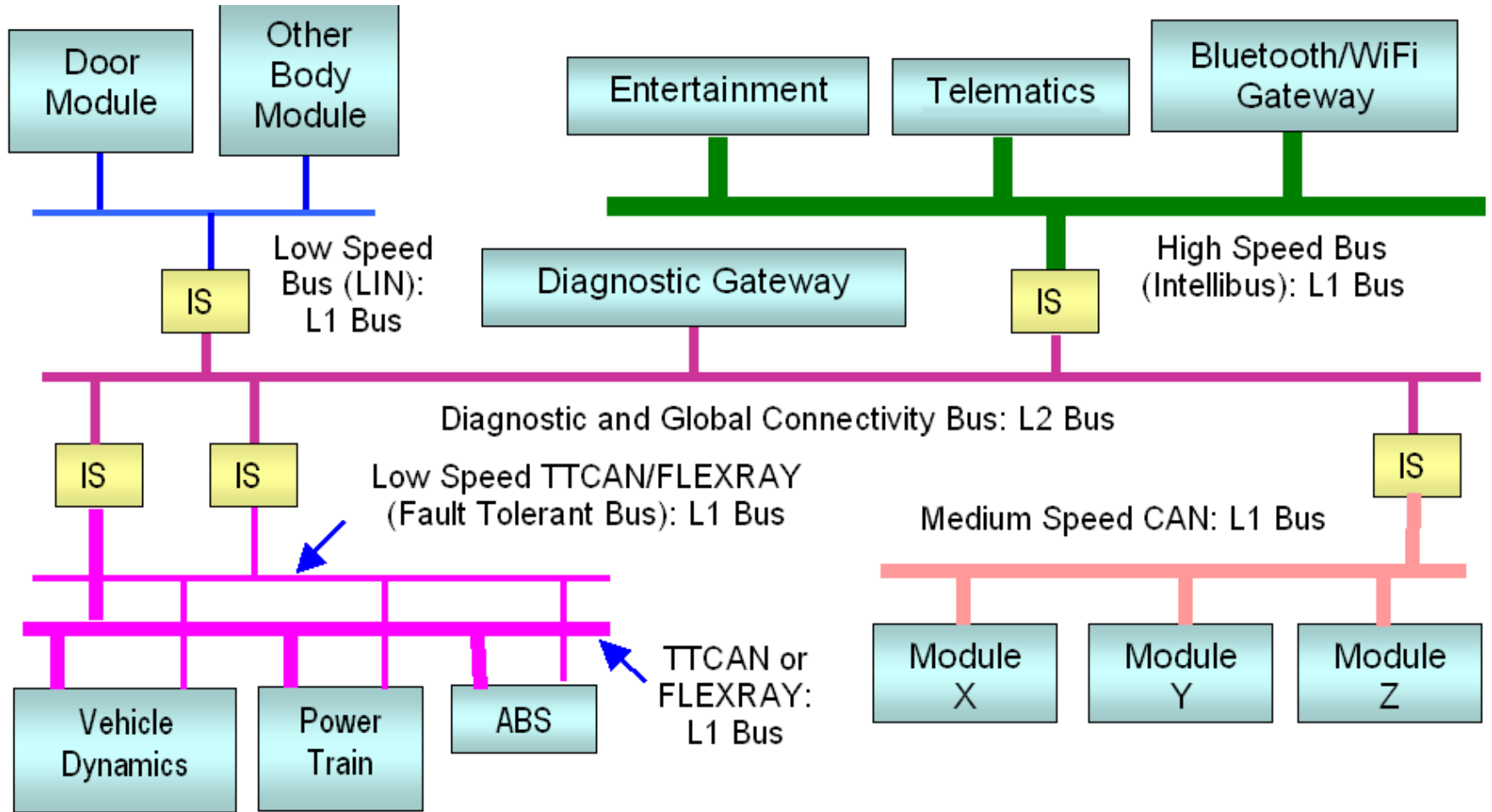
Fault Tolerant Hierarchical Bus System



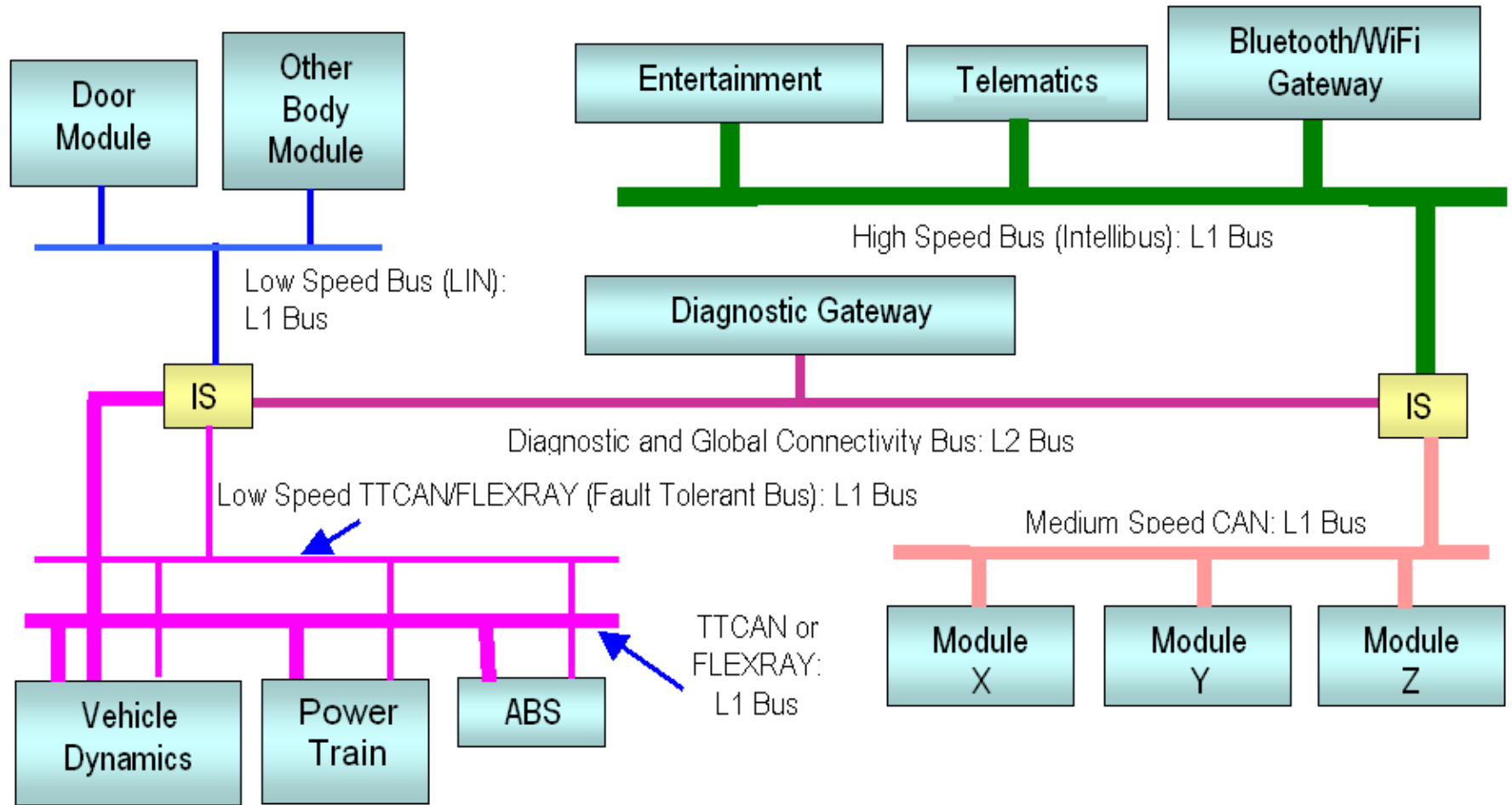
Fault Tolerant Hierarchical Bus System



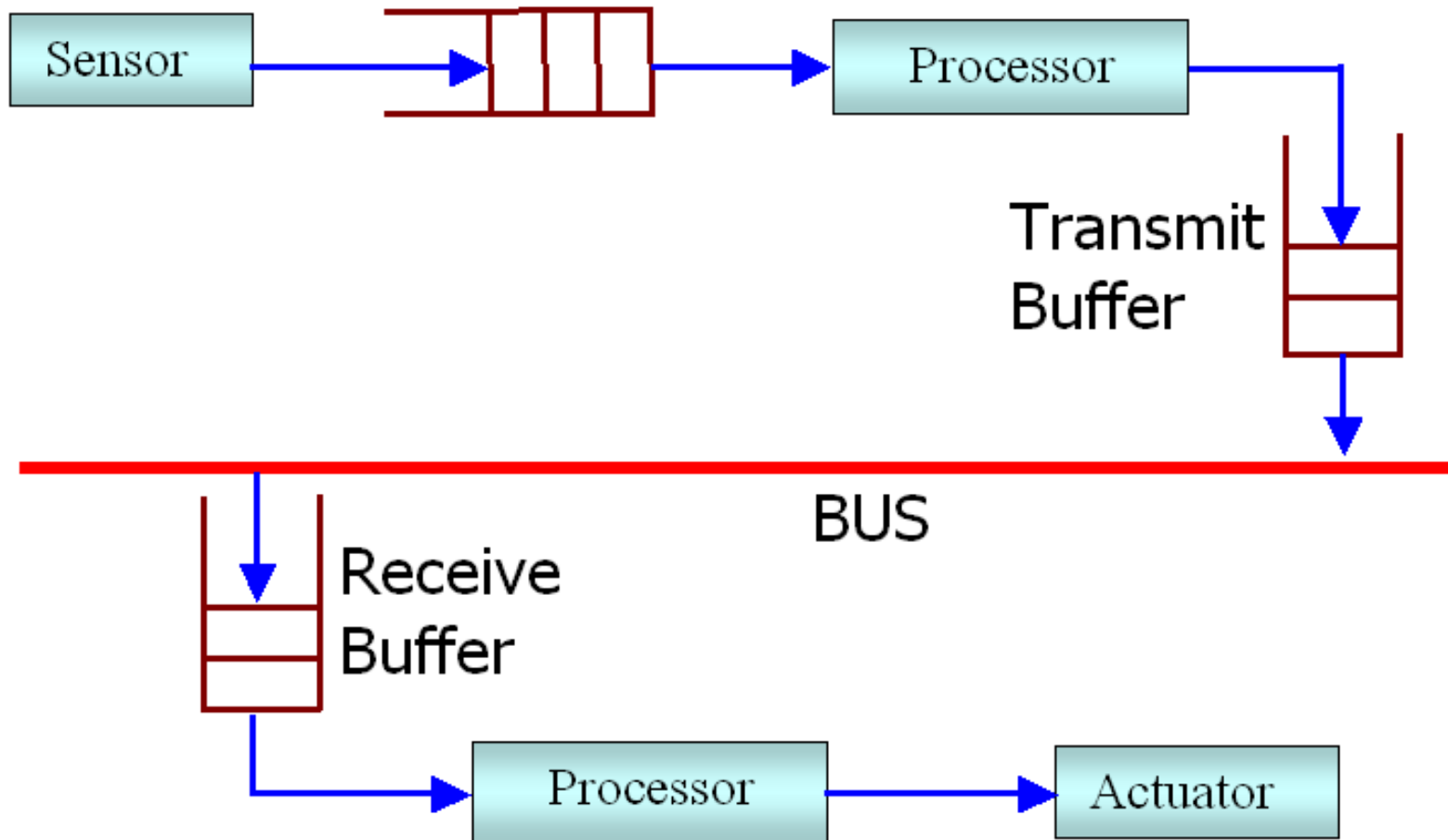
Partial Fault Tolerant Hierarchical Bus System



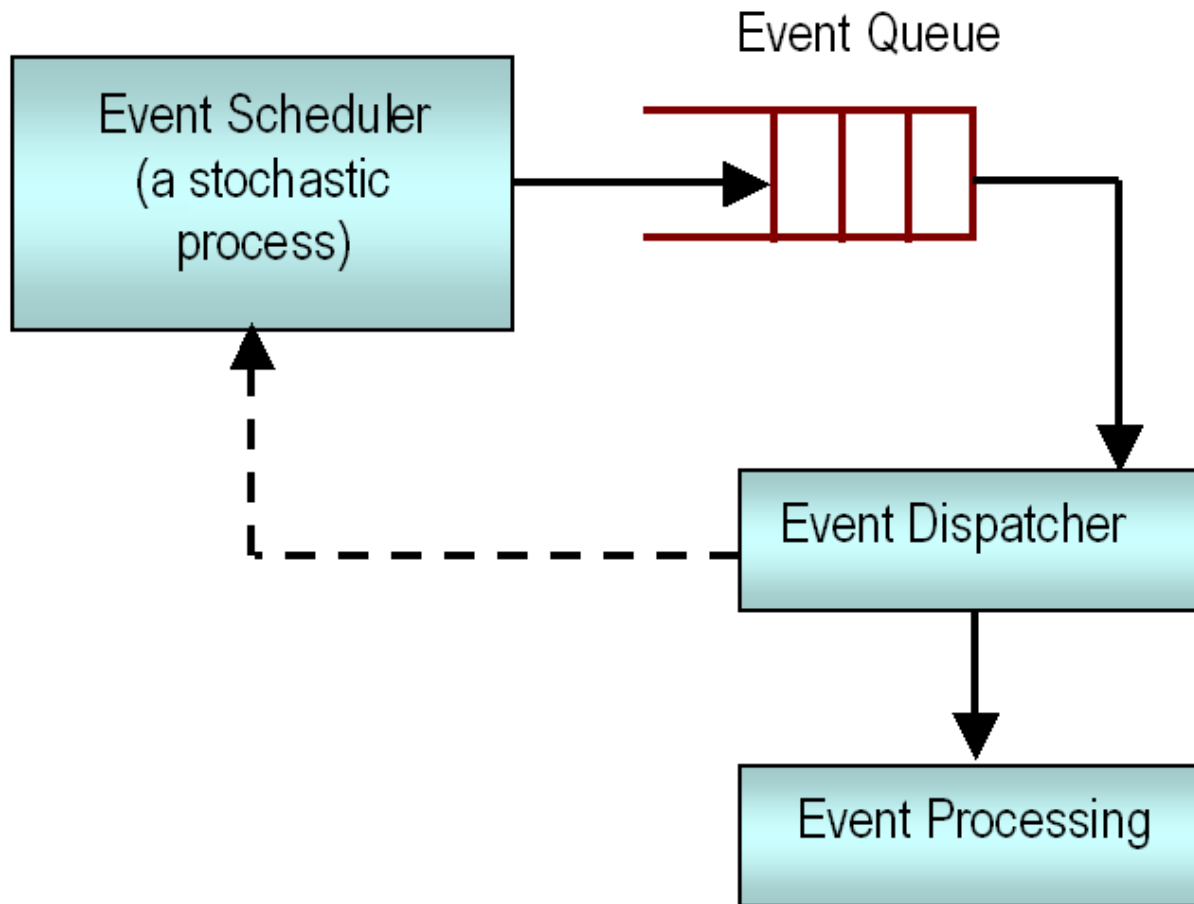
Partial Fault Tolerant Hierarchical Bus System



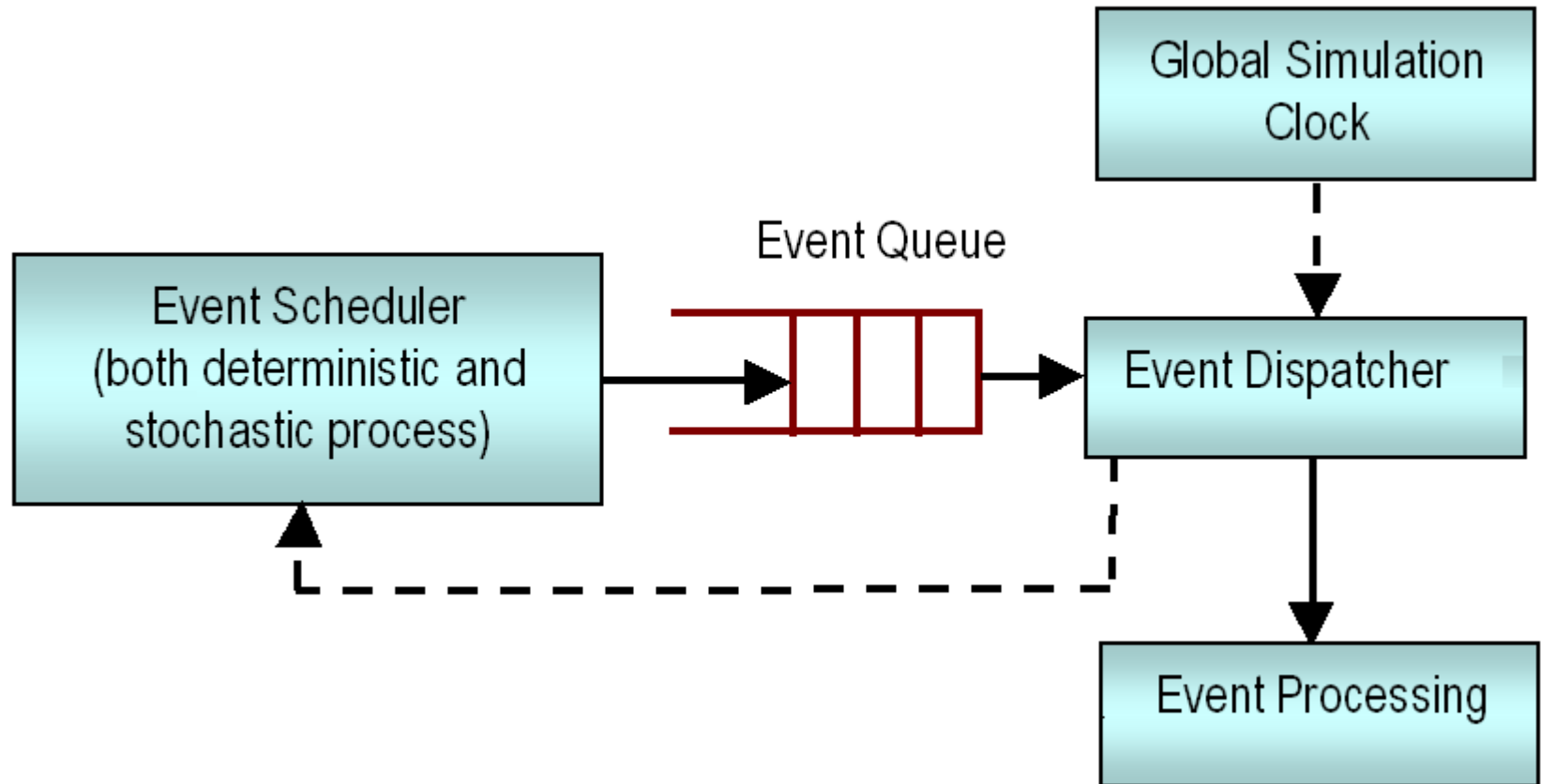
Information Flow from Sensor to Actuator



Simulation Model for an Event-Based System.



Simulation Model for Event- and Time-Based Systems.



THE END

References

1. CAN Protocol Standard, Motorola, Document Number: BCANPSV2.0.
2. Overview of CAN, Motorola.
3. Thomas Fuehrer, Bernd Mueller, Florian Hartwich and Robert Hugel, "Time-Triggered CAN (TTCAN)," Proc. of the SAE 2001 World Congress, Detroit, Michigan, March 5-8, 2001, paper number: 2001-01-0073.
4. Holger Zeltwanger, "Time-Triggered Communication on CAN," Proc. of the SAE 2002 World Congress, Detroit, Michigan, March 4-7, 2002, paper number: 2002-01-0437.
5. Syed Masud Mahmud and Sheran Alles "In-Vehicle Network Architecture for the Next-Generation Vehicles," Proc. of the SAE 2005 World Congress, April 11-14, 2005, Detroit, Michigan, USA, Paper Number: 2005-01-1531