AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Middleware and Communication Protocols for Dependable Systems TI-MICO

“Introduction to Higher Layer CAN-BUS Protocols”

Version 26.11.2014

Background Materials

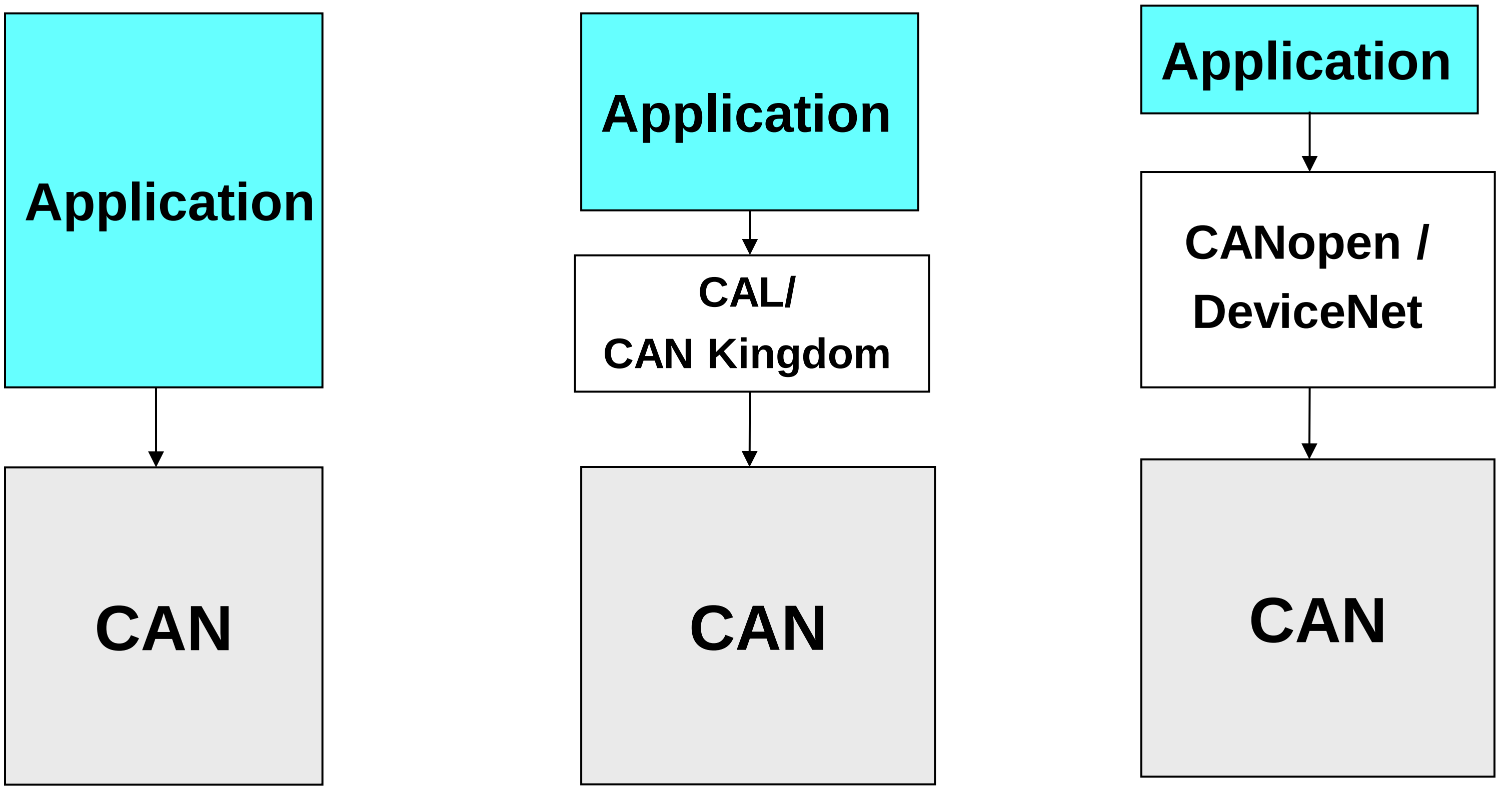
Article with commercial HLPs:

- ***"CAN-based higher layer protocols and profiles"*** by Prof. K. Etschberger, IXXAT Automation, April 2003

Research articles with proposals for HLPs:

- ***"Invocation of Real-Time Objects in a CAN Bus-System"*** by Jörg Kaiser, M. A. Livani, University of ULM, 1998.
- ***"Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN)"*** by Jörg Kaiser and M. Mock, 1999.

Three Development Scenarios



based on a layer-2 implementation

based on a standard Application layer

based on a standardized Application profile

Basic CAN versus HLP

- Basic CAN supports:
  - **Connectionless transmission**
  - **Unacknowledged** transfer of CAN-message
  - Unacknowledged remote request of a CAN-message
  - **Max. 8 data bytes**
  - No specific rules for assignment of message identifiers
- Extra functionality must be provided by **higher layer protocols** on top of basic CAN

Higher Layer Protocol functionalities

- Transmission of messages >8 bytes
- Support for a confirmed client/server like communication model
- Used for configuration purposes e.g. software download
- Used for network management
- HLP´s normally specifies schemas for **allocation of message identifiers**
- Some HLP´s supports an **object model**

CAN Higher Layer Protocols (1)

- CAN is used as the basis for several major "7-layer" protocol developments such as:
  - **CAL: CAN Application Layer** (CiA: CAN in Automation)
  - **CAN Kingdom** (Kvasar)
  - **CANopen** (CiA: Can in Automation)
  - **DeviceNet** (Rockwell Automation, ODVA)
  - **SDS (Smart Distribution Systems)** (Honeywell)
  - **Volcano** (Developed by Volvo)
  - **SAE J1939** (Society of Automotive Engineers)
  - **TTCAN: Time Triggered CAN** (Bosch)

CAN Higher Layer Protocols (2)

- Each of these higher layer protocol architectures is essentially **complete industry-specific network solutions** packaged to include defined requirements for the:
  - physical layer,
  - address structure & message structure
  - conversation structure
  - data structure
  - application/network interface
  - Standardized deviced

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

CAN Milestones

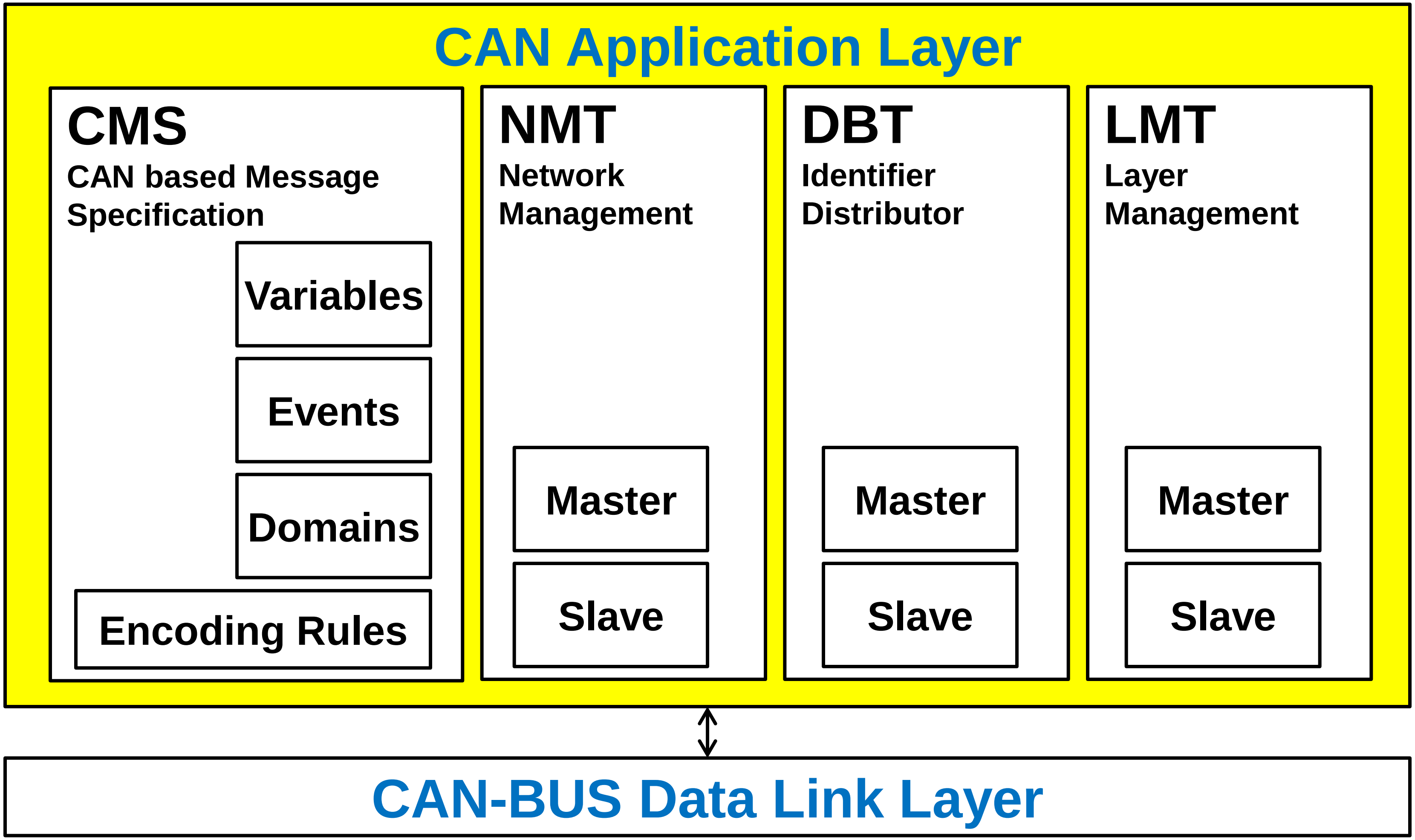
1983:	Start of the Bosch internal project to develop an in-vehicle network
1986:	Official introduction of CAN protocol
1987:	First CAN controller chips from Intel and Philips Semiconductors
1991:	Bosch's CAN specification 2.0 published
1991:	CAN Kingdom CAN-based higher-layer protocol introduced by Kvaser
1992:	CAN in Automation (CiA) international users and manufacturers group established
1992:	CAN Application Layer (CAL) protocol published by CiA
1992:	First cars from Mercedes-Benz used CAN network
1993:	ISO 11898 standard published
1994:	1st international CAN Conference (iCC) organized by CiA
1994:	DeviceNet protocol introduction by Allen-Bradley
1995:	ISO 11898 amendment (extended frame format) published
1995:	CANopen protocol published by CiA
2000:	Development of the time-triggered communication protocol for CAN (TTCAN)

Ref: www.can-cia.de

CAL: **C**AN **A**pplication **L**ayer

- Standardized by CiA in 1996
- CAN Application Layer for **Industrial Applications** (CiA DS 201-207, ver. 1.1).
- Has support for a client/server model with a confirmed CAL service.
- Specifies standard communication objects
  - Variable, event and domain objects

CAL Architecture



CANopen

- **CANopen**: a standardized application for distributed **industrial automation systems**
- Based on CAN standard and **CAL** (CAN Application Layer protocol)
- In **Europe** the definitive standard for the implementation of industrial CAN-based system solutions
- Standardized by CiA (CAN-in-Automation)
- Devices profiles
  - e.g. digital/analog I/O modules, drives, encoders, MMI-units, controllers
- Two types of communication mechanisms:
  - **unconfirmed transmission** of data frames to transfer process data
  - **confirmed transmission** of data (for configuration purpose)

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

DeviceNet

- **DeviceNet** developed by Rockwell Automation in 1995
- Main CAN automation technology in **USA** and **Asia**
- **ODVA**: Open DeviceNet Vendor Assocation (>300 members)
- DeviceNet is a **connection-based** communication model (ConnectionId = CAN identifier)
- Two message types: explicit and I/O messages
- Max 64 nodes in a DeviceNet network

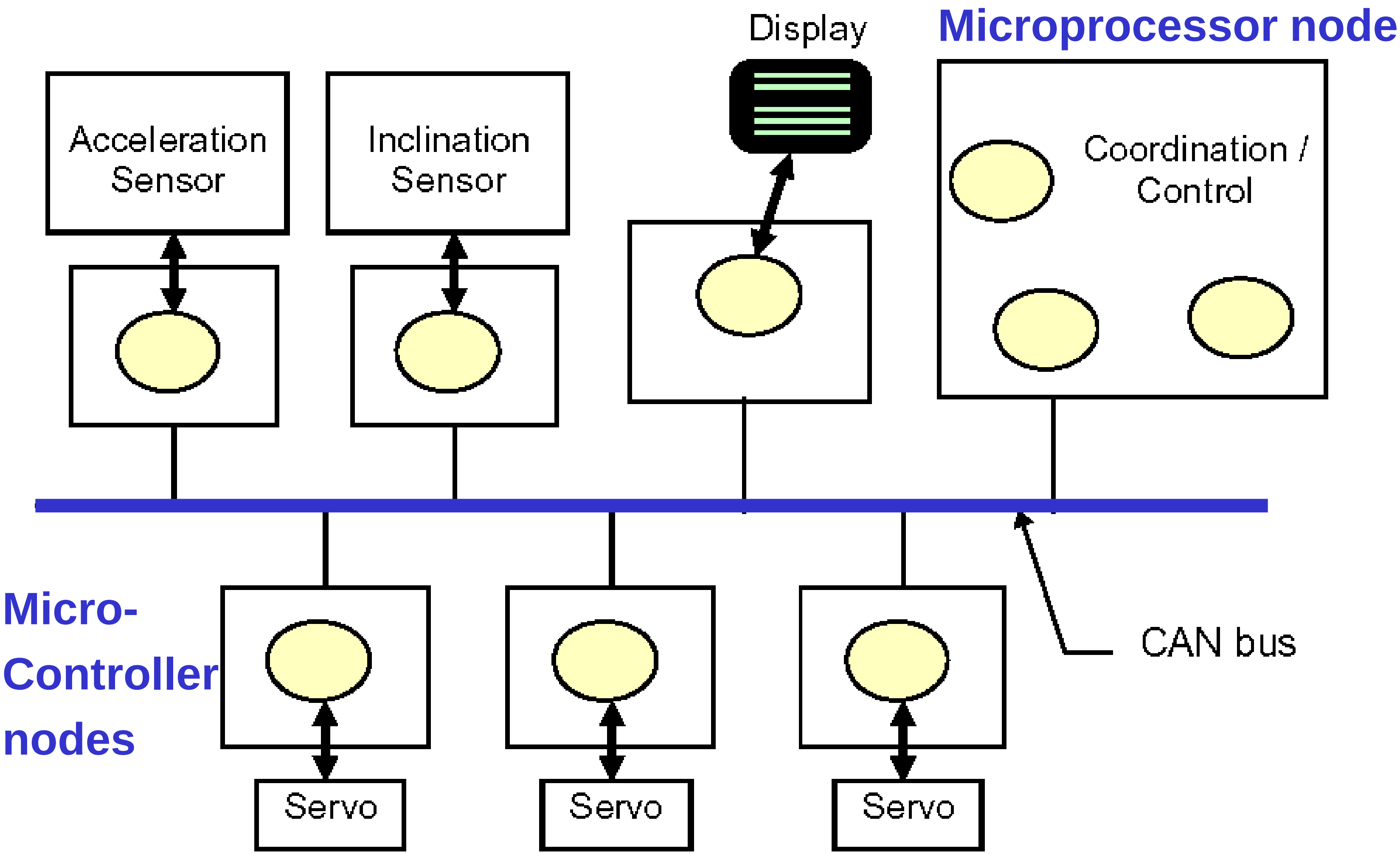
Research approaches for own HLPs

- ***Inspiration articles for developing own HLP:***
  - ***"Invocation of Real-Time Objects in a CAN Bus-System"***
  - ***"Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN)"***

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

The System Model

Example: a simple active suspension system



CAN-bus and OO

- The CAN message format is used to uniformly **invoke methods** on the object
- **Method name and operation encoded** in data field
- A **group** of objects can be addressed with a single CAN message

The System Model (1)

- An **object has a unique name** and a set of associated **operations**
- The unique name of the object is translated to a short form system name during run-time and maintained by a configuration service
- Active, autonomous objects:
  - Export information without a previous request

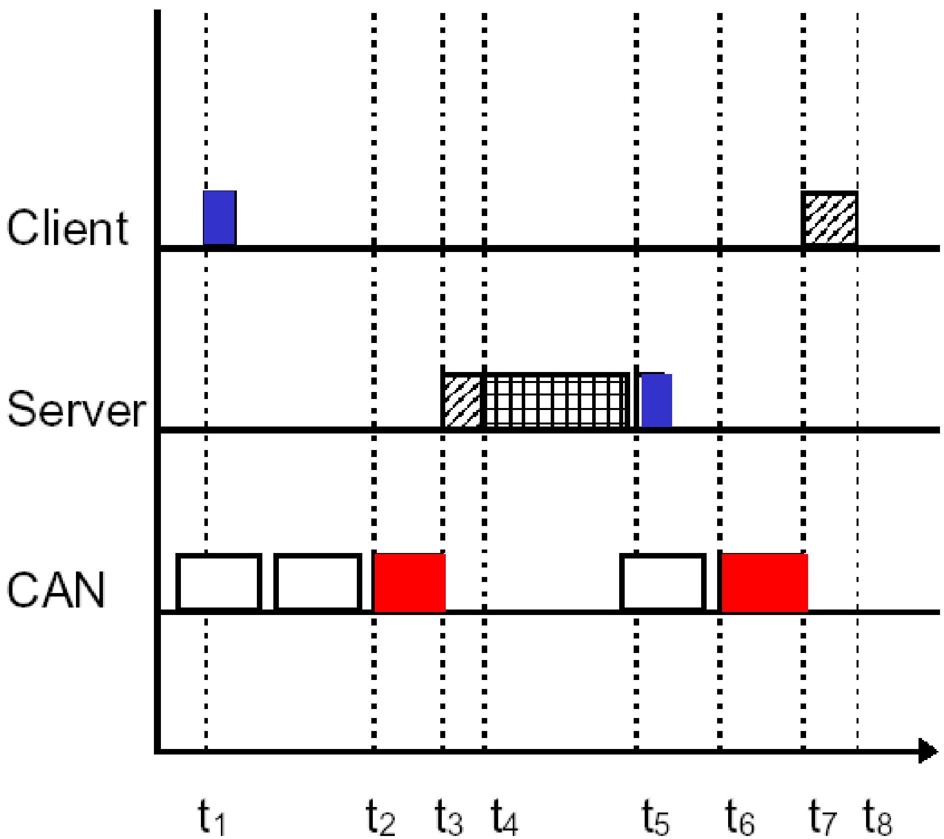
The System Model (2)

- Normal OO distributed communication
  - Is based on **synchronous** method invocation to a single object
- In real-time control systems
  - It is beneficial to provide **groups of objects** and to use **asynchronous** multicast.
  - Examples:
    - Simple and fast distribution of messages i.e. alarms and sensor data,
    - **Replicated objects** forming a group to achieve fault-tolerance

CAN-bus Properties

- The CAN bus supports consistent multicasts
  - It allows to address **a group of N objects** with one message
- CAN provides atomicity of message transfer
  - Either all operational nodes correctly receives a message or none of them
  - If a node locally detects a transmission failure it invalidates the current message
    - Automatically the sender will retransmit the message

Activities of a Method Invocation



t_1 ... Client requests Service

t_2 ... Request transmission starts

t_3 ... Request transmission ends

t_4 ... Server receives request

t_5 ... Server returns the result

t_6 ... Result transmission starts

t_7 ... Result transmission ends

t_8 ... Client receives the result

... Queueing time at sender

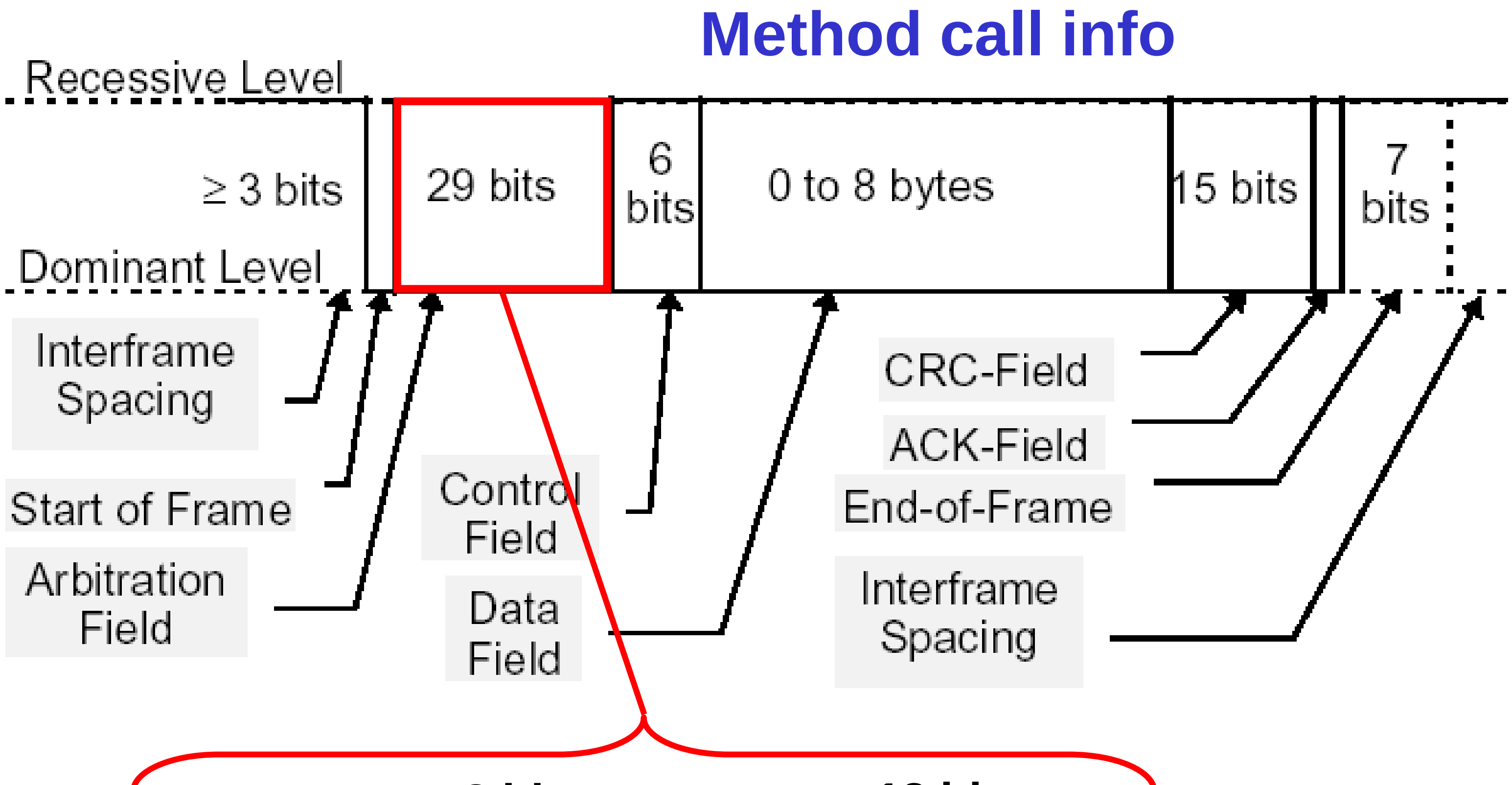
... Delay due to busy bus

... Transmission time

... Computation on server

... Delivery time at receiver

CAN Message Format (Data Frame)



8 bit	8 bit	13 bit
Dynamic Priority	**TxNode**	**Group + objects**

Arbitration Field Structure

8 bit	8 bit	13 bit
Dynamic Priority	TxNode	Group + objects

Dynamic Priority:

- Its value is changed over time by the transmitting node
- By relating the priority to the time until transmission Deadline, a deadline-driven scheduling can be achieved

TxNode:

- Identifies the sending node
  - the unique ID of the sending node guarantees that different senders may never generate equal arbitration fields

Different Group Configurations

13 bit

Max. 256 groups, group size < 32

8 bits	8 bits	8 bits	5 bits
Priority	TxNode	RxGroup	RxObj

Max. 512 groups, group size < 16

8 bits	8 bits	9 bits	4 bits
Priority	TxNode	RxGroup	RxObj

Max. 1024 groups, group size < 64

8 bits	8 bits	10 bits	3 bits*
Priority	TxNode	RxGroup	RxObj

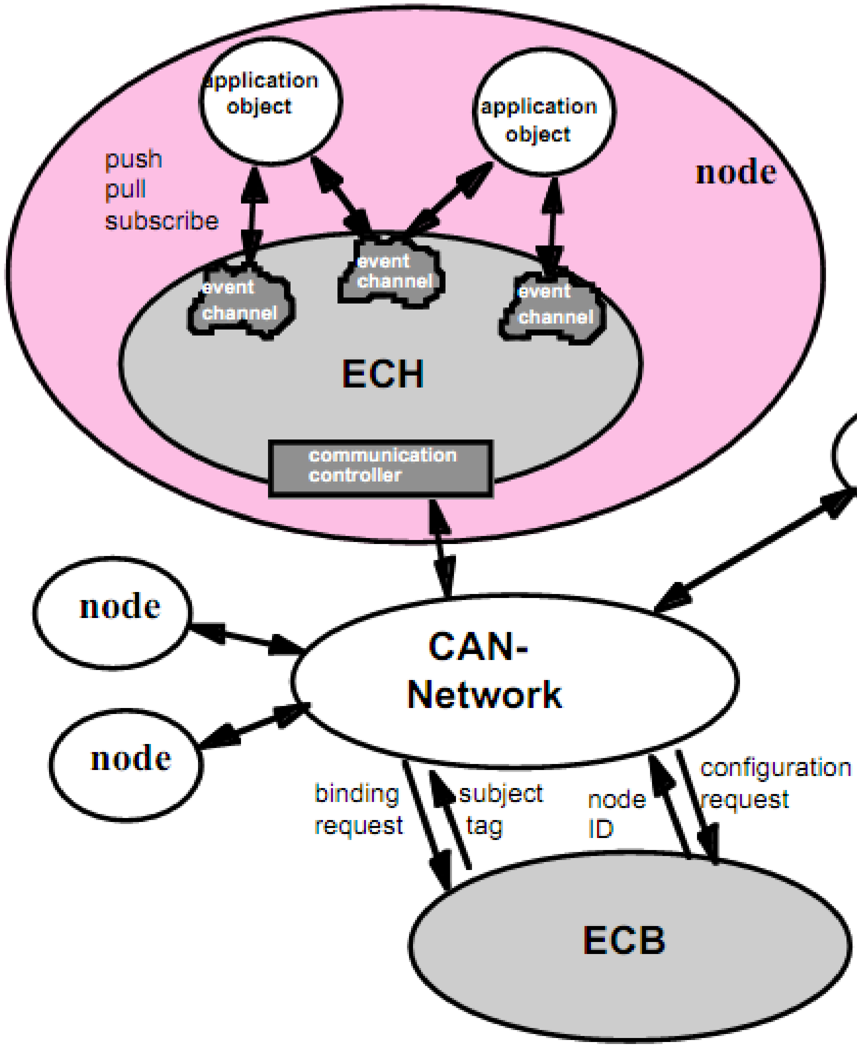
*) in this large system, for individual addressing of group members, at least 3 bits of addressing information must be placed into the data field.

The method name and method parameters are contained in the data field (8 bytes)

Multicast Addressing and Message Filtering

- CAN is a broadcast medium
- Group communication is realized by programming the receive buffers to receive messages selectively
- The **group name (RxGroup)** is used as the **key field** of the associative filter
  - all other bits are masked out
- All messages which pass the filter belongs to a group
- The MCA now uses the **RxObj** field to decide whether the message is to the entire group (RxObj=0) or to **a local group member**

RT-Publish/subscribe Architecture



ECH: Event Channel Handler
ECB: Event Channel Broker

ECB assigns tags as Identifiers and Assigns Node IDs

Summary

- HLPs are necessary to provide standardized industry solutions at the application level
- HLPs have different models for the identifier allocation
- OO RMI communication can be build on top of a CAN bus
- Group communication can be supported in an effective way
- Hard- and soft real-time communication can be obtained

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

References

- [Etschberger]: "**Controller Area Network** – basics, protocols, chips and applications", by Konrad Etschberger, IXXAT Press, 2001
- ODVA: Open DeviceNet Vendors Association www.odva.org
- CiA: CAN in Automation: http://www.can-cia.org/