Lecture 16: Real-Time Networks and Networked Control Systems

[These slides]

- Background
- Real-Time Networks
- Protocol Stack
- Network Examples
  - CAN
  - TTP
- Networked Control Systems

These slides are partly based on material from Luis Almeida, Universidade de Aveiro, Portugal
Background

Distributed architectures are pervasive in many application fields:
- Industrial automation
- Transportation systems (airplanes, cars, trucks, trains, …)
- Multimedia systems (remote surveillance, industrial monitoring, video on demand, …)

In many cases with critical timeliness and safety requirements

Increasingly common that control loops are closed over networks (= networked control)
Background

Motivations for distributed architectures:
- Processing closer to data source /sink
  - "Intelligent" sensors and actuators
- Dependability
  - Error-containment within nodes
- Composability
  - System composition by integrating subsystems
- Scalability
  - Easy addition of new nodes with new or replicated functionality
- Maintainability
  - Modularity and easy node replacement
  - Simplification of the cabling
Background

Today there are many different networks with real-time capabilities aiming at different application domains, e.g.

- ATINC629, SwiftNet, SAFEbus – avionics
- WorldFIP, TCN – trains
- CAN, TT-CAN, FlexRay – cars
- ProfiBus, WorldFIP, P-Net, DeviceNet, Ethernet – automation
- Firewire, USB - multimedia
VW Phaeton

• 11,136 electrical parts
• 61 ECUs (Electronic Controller Units == CPUs)
• Optical bus for high bandwidth infotainment data
• 35 ECUs connected by 3 CAN-busses sharing
  – 2500 signals
  – In 250 CAN messages

The VW Phaeton
Adapted from (Loehold, WFC2004)
Volvo XC 90 network topology
Contents

• Background
• **Real-Time Networks**
• Protocol Stack
• Network Examples
  – CAN
  – TTP
• Networked Control Systems
Service requirements

Typical service requirements in real-time networks:

- Efficient transmission of **short data** (few bytes)
- **Periodic** transmission (control, monitoring) with **short periods** (ms), **low latency**, and **small jitter**
- Fast transmission (ms) of **aperiodic requests** (alarms, commands, …)
- Transmission of **non-real-time data** (configuration information, log data, …)
- Multicasting as well as unicasting (peer to peer)
The Network in a Distributed System

- The network is a fundamental component in a distributed system supporting all the interactions among the nodes.
- Hence, it is also a critical resource since loss of communication results in the loss of all global system services.
Network Interfaces

- The network extends up to the **Communication Network Interface (CNI)** that is the interface between the communication systems and the node host processor.
Messages & Transactions

- Interactions are supported by message passing
- A message is a **unit of information** that should be transferred at a given time from a sender to one or more receivers
- Contains both the **data** and the **control information** that is relevant for the proper transmission of the data (e.g., sender, destination, checksum, …)
- A network **transaction** is the sequence of actions within the communication systems required to transfer the message
- Might include messages containing only control information, i.e., **control messages**
- Some networks automatically break large messages into smaller packets (**fragmentation/reassembly**)
- A **packet** is the smallest unit of information that is transmitted
- The **data efficiency** of the network is the ratio between the the time to transmit **effective data** bits and the total duration of the transaction
Timing Figures

• Typical figures concerning the temporal behaviour of the network:
  
  – **Network induced delay** – extra delay caused by the transmission of data over the network. Some applications, e.g., control, are very sensitive to this
  
  – **Delay jitter** – variations in the network induced delay. Some applications, e.g., multimedia streaming, are very sensitive to this, but not so sensitive to the delay.
  
  – **Buffer requirements** – when the instantaneous transmission from a node is larger than the capacity of the network to dispatch, the traffic must be stored in buffers. Too small buffers lead to packet losses
  
  – **Packet loss probability** – packet losses can in addition to the above also be caused by unreliable network media. One example of this is wireless networks.
Timing Figures, cont

- **Throughput (bandwidth)** – amount of data, or packets, that the network dispatches per unit of time (bit/s and packet/s)

- **Arrival/Departure rate** – rate at which data arrives at/from the network

- **Burstiness** – measure of the traffic submitted to the network in a short interval of time. Bursts may have a negative impact on the real-time performance of the network and impose high buffering requirements. **Traffic shaping** can be used to control the characteristics of the traffic generated by a node.
Real-Time Messages

- Real-time messages can have **event** or **state semantics**
- **Events** are perceived changes in the system state. All events are significant for the state consistency across sender and receiver.
- **Event messages** must be queued at the receiver and removed upon reading. **Correct order** in delivery must be enforced.
- **State messages** (containing state data) can be read many times and overwrite the values of the previous message concerning the same real-time entity.
Event or Time Triggering

• According to the type of message (event or state) conveyed by the network, it can be
  – Event-triggered (event messages)
    or
  – Time-triggered (state messages)
Event-Triggered Network

What if m1 is lost?
There is a notion of **network time**
- All clocks are globally synchronized

Transactions carrying **state data** are triggered at **predefined time instants**

- Receivers have a **periodic refresh** of the system state
- The submitted communication load is well determined
Time-Triggered Network

Loosing a message gives inconsistency only until next message arrives
Event vs Time Triggering

Time-triggered networks:
- Are more deterministic
  - Transmission instants are predefined
  - Fault-tolerance mechanisms are easier to design
- Are less flexible in reacting to errors
  - Retransmissions are often not possible because the traffic schedule is fixed
  - A lost message is not recovered until the next period of the message stream
- Are less flexible with respect to changes
  - Everything must be known a priori and very little can be changed dynamically (cp. static cyclic CPU scheduling)
- The communication protocols are often quite complex
Event vs Time Triggering

Event-Triggered Networks:

- Low level of determinism
  - Events can occur at any time
- More complex fault-tolerance schemes
- Very flexible with respect to errors
  - Retransmissions can be carried out immediately
- The communication protocols are normally quite simple
Contents

• Background
• Real-Time Networks
• Protocol Stack
• Network Examples
  – CAN
  – TTP
• Networked Control Systems
The OSI Protocol Stack

The OSI Reference Model

- Common services for a class of apps. (FTP, HTTP, ..)
- Data semantics (conversion, compression, encryption, ..)
- Remote actions (name services, directory, billing, ..)
- End-to-end comm. control, e.g., (TCP, UDP)
- Routing, logical addressing
- Physical addressing, medium access control
- Topology, medium, bit-encoding
Protocol Stack

- Layer 1
  - Layer 1 protocol
  - Layer 1/2 interface
  - Layer 1
- Layer 2
  - Layer 2 protocol
  - Layer 2/3 interface
  - Layer 2
- Layer 3
  - Layer 3 protocol
  - Layer 3/4 interface
  - Layer 3
- Layer 4
  - Layer 4 protocol
  - Layer 4/5 interface
  - Layer 4
- Layer 5
  - Layer 5 protocol
  - Host 1
  - Host 2

Physical medium
Layer Interfaces

Very large computing and communication overhead for short real-time data traffic!!
Real-Time Protocol Stack

• The end-to-end communication delay must be bounded
  – All services at all layers must be time-bounded
  – Requires appropriate time-bounded protocols

• The 7 OSI layers impose a considerable overhead…

• Many real-time networks
  – are dedicated to a well-defined application (no need for presentation)
  – use single network domain (no need for routing)
  – use short messages (no need to fragment/reassemble)
Collapsed OSI Model

- Application accesses the Data Link directly
- Other layers may be present but are not fully stacked
- In industrial automation these networks are called fieldbuses

Diagram:
- Node A
  - Application
  - Data Link
  - Physical
- Node B
  - Application
  - Data Link
  - Physical
- CNI
Physical Layer

- Interconnection topology
- Physical medium
- Coding of digital information
- Transmission rate
- Maximum interconnection length
- Max. number of nodes
- Immunity to EMI (Electro-magnetic interference)
- ....
Physical Layer: Topology

- Tree
- Mesh (wired)
- Mesh (wireless)
- Star
- Bus
- Ring
Physical Layer: Medium

- Copper wiring
  - Cheaper cables and interfaces (+), suffers EMI (-)

- Optical fibres
  - Immune to EMI, wide bandwidth, low attenuation (+), expensive cables and interfaces (-)

- Wireless – Radio Frequency (RF)
  - Mobility, flexibility (+), very susceptible to EMI (-), multi-path fading (-), attenuation (-), open medium (+/-)

- Wireless – Infra-red light (IR)
  - Mobility, flexibility (+), line-of-sight (-), open medium (+/-)
Data Link Layer

Issues related to:

- Addressing
- Logical Link Control (LLC)
  - Transmission error control
- Medium Access Control (MAC)
  - for shared media
Data Link Layer: Addressing

• Direct addressing
  – The sender and receiver(s) are explicitly identified in every transaction, using physical address (MAC addresses in Ethernet)

• Indirect (source) addressing
  – The message contents are explicitly identified (e.g. temperature of sensor X). Receivers that need the message retrieve it from the network (as in CAN)

• Indirect (time-based) addressing
  – The message is identified by the time instant at which it is transmitted (as in TTP – Time Triggered Protocol)
Data Link Layer: LLC

• Logical Link Control (LLC)
  – Deals with the information transfer at this level
  – Upper sub-layer of the data link layer
  – Typical services are:
    • **Send with immediate acknowledge**
      Sender waits for acknowledge from receiver
    • **Send without acknowledge**
      No synchronization between data and receiver
    • **Connection-oriented services**
      A connection must be established between parts before any communication may take place
Data Link Layer: Transmission Error Control

Part of the LLC

Specifies error detection and action upon this. Typical actions are

- **Forward error correction** (FEC)
  - Error correcting codes (more related to the physical layer)
- **Automatic Repeat reQuest** (ARQ)
  - The receiver triggers a repeat request upon error
- **Positive Acknowledgement and Retry** (PAR)
  - The sender resends if ACK is not received

From a real-time perspective, ARC and PAR may induce longer delivery delays as well as extra communication load
Data Link Layer: MAC

Medium Access Control (MAC)

- Lower sub-layer of the data link layer
- Determines the order of the network access by contending nodes and, thus, the network access delay
- Is of paramount importance for the real-time behaviour of networks that use a shared medium
MAC: Master-Slave

- Access is granted by the Master node
- Nodes synchronized with the master
- Requires one control message per data message

- Ex. WorldFIP, Ethernet Powerlink, Bluetooth (within piconets)
MAC: Token Passing

- Access is granted by the possession of a token
- Order of access enforced by token circulation
- Real-time operation requires bounded token holding time

- Ex. FDDI, PROFIBUS
MAC: TDMA

Time-Division Multiple Access

- Access granted in a dedicated time-slot
- Time slots are pre-defined in a cyclic framework
- Requires global clock synchronization
- High data efficiency
- Typically uses static table-based scheduling

- Ex. TTP/C, TT-CAN, PROFINET
MAC: CSMA

Carrier-Sense Multiple Access:

– Set of protocols based on sensing bus inactivity before transmitting (asynchronous bus access)
– There may be collisions
– Upon collision, nodes back off and retry later, according to some specific rule (this rule determines to a large extent the real-time features of the protocols)
Carrier-Sense Multiple Access with Collision Detection

- Used in shared Ethernet (hub instead of switch)
- Collisions are destructive and are detected within collision window
- Upon collision, the retry interval is random and the randomization window is doubled for each retry until 1024 slots
- Non-deterministic (e.g., chained collisions)
- Not suitable for a real-time network. However,
  - The physical Ethernet layer is often used in real-time networks.
  - It is possible to get real-time performance on an Ethernet network, if the access to the medium is scheduled in some way, i.e., collisions are avoided
MAC: CSMA/BA

Carrier-Sense Multiple Access with Bit-Wise Arbitration

- Bit-wise arbitration with non-destructive collisions
- Upon collision, highest priority node is unaffected. Nodes with lower priorities retry right after
- Deterministic
- E.g. CAN
- Sometimes also called CSMA/CR (Collision Resolution) or CSMA/CA (Collision Arbitration), although the latter really is something else
Contents

• Background
• Real-Time Networks
• Protocol Stack
• Network Examples
  – CAN
  – TTP
• Networked Control Systems
CAN

- Controller Area Network
- Created by Bosch for use in the automotive industry
- Used in most European cars today
- Adopted by GM as an in-house standard
- Expanded to industrial automation
- Defines physical and data link layer
- Multi-master, broadcast, serial bus
- Transmission rate from 5 Kbit/s to 1 Mbit/s
- Small sized messages – 0-8 bytes
- Relatively high overhead (47 bits + stuff bits)
- On transmission, nodes synchronize on bit level
CAN

• Can is like Ethernet …
  – Everybody with a message to send waits until the bus is quiet, then starts transmitting, and if a node detects a collision it backs off and retries later

• … but more deterministic
  – A CAN bus has a special electrical property that allows it to handle collisions better
The CAN Protocol

- Messages are called frames
- A frame is tagged by an identifier
  - Indicates the contents of the frame (used for addressing)
  - Used in the arbitration for prioritizing frames (the frame with the lowest identifier is selected to send in case of collision)
- The CAN physical layer behaves as a wired AND, i.e. if any node sends a logical 0, then all nodes receive 0 bit.
Bit Stuffing

- CAN frames are bit stuffed.
  - If 5 bits in a row are the same sign then the protocol inserts a bit of the opposite sign
  - Used to ensure enough edges to maintain synchronization
  - Used to distinguish data frames from special error handling frames
- All nodes receive all frames.
- The handling of the CAN bus communication within a node is done by a special CAN controller (card/chip).
- The CAN controller throws away frames not needed by the node using ID filtering hardware.
- Messages that are waiting to be sent are queued in a priority sorted list in the CAN controller.
The Basic Protocol

- Frames start by sending the identifier field most significant bit first
- When sending the identifier the frame is in arbitration
  - other frames may be sent too
  - need to find the highest priority frame
- If a node sends a 1 (recessive bit) but reads back a 0 (dominant bit) then it gives up and backs off
  - there must be a higher priority frame being sent
- Restarts sending the same frame when the bus is idle again
## Arbitration

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>1344</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame 2</td>
<td>1306</td>
<td>1</td>
</tr>
<tr>
<td>Frame 3</td>
<td>1498</td>
<td>1</td>
</tr>
<tr>
<td><strong>Bus</strong></td>
<td><strong>1</strong></td>
<td></td>
</tr>
<tr>
<td>Frame</td>
<td>Value 1</td>
<td>Value 2</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Frame 1</td>
<td>1344</td>
<td>10</td>
</tr>
<tr>
<td>Frame 2</td>
<td>1306</td>
<td>10</td>
</tr>
<tr>
<td>Frame 3</td>
<td>1498</td>
<td>10</td>
</tr>
<tr>
<td>Bus</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Frame</td>
<td>Value</td>
<td>Type</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>Frame 1</td>
<td>1344</td>
<td>101</td>
</tr>
<tr>
<td>Frame 2</td>
<td>1306</td>
<td>101</td>
</tr>
<tr>
<td>Frame 3</td>
<td>1498</td>
<td>101</td>
</tr>
<tr>
<td>Bus</td>
<td></td>
<td>101</td>
</tr>
</tbody>
</table>
## Arbitration

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>1344</th>
<th>1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame 2</td>
<td>1306</td>
<td>1010</td>
</tr>
<tr>
<td>Frame 3</td>
<td>1498</td>
<td>1011</td>
</tr>
<tr>
<td>Bus</td>
<td>1010</td>
<td></td>
</tr>
<tr>
<td>Frame 1</td>
<td>1344</td>
<td>10101</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Frame 2</td>
<td>1306</td>
<td>10100011010</td>
</tr>
<tr>
<td>Frame 3</td>
<td>1498</td>
<td>1011</td>
</tr>
<tr>
<td>Bus</td>
<td>13000011010</td>
<td></td>
</tr>
</tbody>
</table>
CAN and hard real-time

- Need to know the worst-case frame latencies (end-to-end delay)
- Through testing

Through analysis
- Fixed priority scheduling theory can be applied
- Bus = shared resource, cp CPU
- Frame = job (invocation of a task)
**TTP - Time Triggered Protocol**

- Not just a network, more of a architecture
- Shared broadcast bus, 2-25 Mbit/s
- Popular in car industry for safety-critical applications, e.g., X-by-wire
- Design goals
  - Fault-tolerance
  - Messages latencies that are easy to calculate and have no latency
- Mostly periodic messages
- Replicated broadcast communication channels
- Replicated nodes are grouped into FTUs - Fault Tolerant Units
- Access to the network through TDMA (statically allocated)

\[ \text{slot#1 | slot#2 | slot#3 | slot#4 | slot#5 | slot#6 | slot#7} \]

- More deterministic than CAN (cp. static scheduling vs dynamic scheduling)
TTP Design

- Message transport with predictable low latency: simple protocol with known WCET & minimal overhead
- Fault tolerance: allow node and network failures without loss of functionality
- High precision clock synchronization
- Off-line network traffic scheduling
- TTP info
  - TTTech (www.tttech.com)
  - TTP Forum (www.ttpforum.org)
FlexRay

TTP competitor

Developed by European car manufacturers

Combines time-driven and event-driven communication

Event-driven communication allowed in special slots in the TDMA structure

Similar extensions are also possible in TTP
Contents

• Background
• Real-Time Networks
• Protocol Stack
• Network Examples
  – CAN
  – TTP
• Networked Control Systems
Networked Control Systems

- Networked Control System
  - wired or wireless
Networked Control Structures

A networked control system is a spatially distributed control system with sensor, actuator and controller communication supported by a network.
Networked Control Loop Timing

- Networked embedded control implies temporal non-determinism
  - network communication
  - real-time scheduling
- Degraded control performance due to
  - sampling interval jitter
  - non-negligible input-output latencies with jitter
  - lost samples
- However
  - Most control loops are fairly robust towards temporal non-determinism

- network interface processing delay
- queuing delay
- transmission delay
- propagation delay
- link layer resend delay
- transport layer ACK delay
- ....
Control under network delay

- Delays in communication due to buffering, propagation delays, collisions/resends, ....
- Delays can be fixed or varying, known (measurable) or unknown

[Diagram showing network components and Delays]

Delays and losses INRIA→UMd

[Bolot, 1993]
Clocks and Time Stamps

- The controller must know when the data was sampled/sent in order to compensate for varying network latencies.

- Approaches:
  - Global clock
    - clock synchronization
    - complexity overhead
  - Local clocks
    - associate time-stamps with data packets
    - absolute or relative
    - OK if the data always comes from the same sensor node
    - A problem if the source node of the data changes
Compensate for Input-Output Delays

Sampled model with varying delay $\tau_k$:

$$x(k + 1) = \Phi x(k) + \Gamma_0(\tau_k)u(k) + \Gamma_1(\tau_k)u(k - 1)$$

- Design the feedback based on the average (expected) input-output delay

$$u(k) = -L \begin{pmatrix} \hat{x}(k) \\ u(k - 1) \end{pmatrix}$$

- Modify the observer to take into account current delay $\tau_k$:

$$\hat{x}(k + 1) = \Phi \hat{x}(k) + \Gamma_0(\tau_k)u(k) + \Gamma_1(\tau_k)u(k - 1) + K(y(k) - C\hat{x}(k))$$
Compensate for Network Delays

Only part of the current loop delay ($\tau_{sc}$) can now be measured!

- Time-varying state feedback $L_k$ based on $\tau_{sc}^k + E\{\tau_{ca}\}$
- Let the actuator node record the total delay
- The total delay is communicated back to the controller
- Make the observer time-varying as before
Cross-Layer Design

- Control applications can often adapt to varying network conditions.
- Network information needed at application layer → cross-layer designs.
- Specially important if full OSI or IP protocol stacks are used.

Sensor to Controller:
- when collision happens
- resample rather than resend old data
General Conclusions

• Real-time communication increasingly important in several fields
• The traditional OSI/IP stacks are not well suited for real-time networks
• Collapsed OSI stack (physical, data link, application) better for real-time networks
• Networked control systems increasingly common
• Problems with delays, jitter and lost packets if non-real time networks are used
• Can partly be compensated for by control methods
• In wireless systems the problems become worse