The Time-Triggered Paradigm

Günther Bauer
Institut für Technische Informatik
Technische Universität Wien
Overview

- Introduction
- Related Concepts
- Design Principles
- Time-Triggered Communication
- Conclusion
Target Systems

- Distributed embedded fault-tolerant safety-critical real-time computer systems
What do we trigger?

- Transmission of a message
- Execution of a task
- Output to controlled (real-world) object
- (Timer)
Event Triggers

- Reception of message
- Termination of task execution
- Significant change of state of some external (real-world) object
- (Timeout)
Time Trigger

• The only event that may trigger a significant action is the tick of the clock
• Schedule determines the action(s) to be triggered at a specific clock tick
• Schedule is generated off-line before runtime and is (temporarily) static
• Based on ticks of synchronized global clock
Properties of Time-Triggered Systems

- Predictable behaviour
- Implicit support of cooperation of components through global schedule
- Timely (small jitter)
- Simple
- Inflexible
Overview

- Introduction
- Related Concepts
- Design Principles
- Time-Triggered Communication
- Conclusion
Related Concepts

- Sparse time base
- Time and state
- Real-time entities / images
- State / event information
- State / event messages
Sparse Time Base

- Clock synchronization in distributed systems provides nodes with time base of limited precision
- Time stamps referring to the same event may differ at distinct nodes
- Replica determinism requires that this difference is either irrelevant or resolved by executing an agreement protocol
Sparse Time Base (ctd.)

- Events that carry timestamps differing by one cannot be temporarily ordered
Sparse Time Base (ctd.)

- Nodes B and C need to execute an agreement protocol to maintain replica determinism
Sparse Time Base (ctd.)

- If events can be constrained to intervals of activity, consistent temporal order can be locally established without the need for agreement protocol
Time and State

“The state enables the determination of a future output solely on the basis of the future input and the state the system is in. In other words, the state enables a “decoupling” of the past from the present and future. The state embodies all past history of a system. Knowing the state “supplants” knowledge of the past. Apparently, for this role to be meaningful, the notion of past and future must be relevant for the system considered.” (Mesarovic, “Abstract System Theory”, p.45)
Time and State (ctd.)

- Notion of time and state are inseparable
- Sparse time base can act as the borderline between past and future
- Consistent view of state and time considerably simplifies replication
- Logical time is imprecise whenever physical time is essential
Real-Time Entities

• A set of state variables modelling the dynamics of a real-time application
• RT entity has static (name, value domain, ...) and dynamic (actual value and derivatives) attributes
• Dynamic attributes are captured through observations:

\[ \text{Observation} := < \text{Name}, \text{Value}, t_{\text{obs}} > \]
Real-Time Images

• A temporally accurate picture of an RT entity within the computer system
• Interval throughout which an observation is temporally accurate is parameter of application
• Observation is true forever, but is a valid RT image only for some interval $d_{acc}$ starting at time $t_{obs}$
State / Event Information

- At an interface information is exchanged either through state information or event information

\[
\text{State Information} := \langle \text{Name}, \text{Value}, t_{\text{obs}} \rangle
\]

\[
\text{Event Information} := \langle \text{Name}, \text{Value difference}, t_{\text{event}} \rangle
\]
State Information

- Idempotent
- At sender:
  - At-least-once transmission
  - Non-consuming sending
- At Receiver:
  - Update in place
  - Non-consuming read
Event Information

• Not idempotent
• At sender:
  – Exactly-once transmission
  – Consuming sending
• At receiver:
  – Queuing
  – Consumed upon read
State / Event Messages

• State messages: periodic exchange of state information
• Event messages: sporadic exchange of event information
• State messages and event messages are two extremes of a spectrum of information type and transmission trigger combinations
State Information Message Variations

The Time-Triggered Paradigm
Overview

• Introduction
• Related Concepts
• Design Principles
• Time-Triggered Communication
• Conclusion
Design Principles

• Temporal firewall
• Composability
• Scalability
• Dependability
Temporal Firewall

• An interface that prevents control error propagation by design is called a *temporal firewall*
Communication Network Interface (CNI)

- CNI is most important interface in TT distributed computer system
- CNI contains unidirectional data interfaces and unidirectional control interfaces ➔ elementary interface
- CNI decouples local processing from global interactions
CNI as Temporal Firewall
Composability

“An architecture is composable with respect to a specified property if the system integration will not invalidate this property once the property has been established at the subsystem level” (Kopetz, “Real-Time Systems: Design Principles for Distributed Embedded Applications”, p.34)
Composability (ctd.)

- Independent development of nodes
- Stability of prior services
- Constructive integration of the nodes to generate the emerging services
- Replica determinism
- Diagnosis
Composability: Independent Development of Nodes

- Architecture design vs. node design
- Composability requires precise specification of node services at the level of architecture design
- CNI of a TT distributed computer system provides specification of node services both in value domain and time domain
Composability: Stability of Prior Services

• The validated service of a node – both in the value domain and the time domain – is not refuted by the integration into a system

• Information-pull interfaces at receiving part of the CNI ensures that the host computer system can operate independently of the communication system
Composability: Constructive Integration

- Concerned with design of communication system
- Integration of nodes needs to be linear (not circular)
- If $n$ nodes are integrated and operable, the integration of the $n+1^{st}$ node must not disturb the operation of the $n$ previously integrated nodes
- Offline-established static communication schedule supports constructive integration
Composability: Replica Determinism

• Replica determinism requires replicated nodes to maintain the same (externally visible) state and produce the same output at about the same time
• Replica determinism significantly simplifies realization of fault tolerance through node replication
• Replica determinism decreases testing and debugging efforts
Composability: Diagnosis

• Diagnostic deficiency if a monitoring device cannot tell the origin of a (possibly faulty) message
• TT communication simplifies diagnosis through temporal firewalls and use of static off-line established communication schedule
Scalability

- Extensibility: Neither communication bandwidth nor processing power should be a bottleneck in a distributed system
- Complexity: The effort required to understand a particular function should remain constant and be independent of system size
Scalability: Extensibility

• A node may be expanded to a cluster of nodes without changing the interface to the original cluster
• Within the bandwidth capacity of the initial systems nodes may request additional bandwidth or new nodes may be added (cf. composability)
Dependability

- Support of fault tolerance strategies through active replication
- Nodes of the distributed system form fault-containment regions
- Functions of distinct fault-containment regions need to fail statistically independently
Overview

• Introduction
• Related Concepts
• Design Principles
• Time-Triggered Communication
• Conclusion
Time-Triggered Communication

- Synchronous system
- Common communication schedule
- Clock synchronization service
- Using A Priori Knowledge
- Non-Communication Services
Synchronous System

- Known upper bound on the transmission time of a message
- Every node maintains a local clock with known bounded rate of drift with respect to real time
- There are known upper and lower bounds on the time required by a process to execute a step
Time-Triggered Communication

- Synchronous system
- Common communication schedule
- Clock synchronization service
- Using *A Priori* Knowledge
- Non-Communication Services
## Common Communication Schedule

<table>
<thead>
<tr>
<th>When</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

The Time-Triggered Paradigm
Common Communication Schedule (ctd.)

<table>
<thead>
<tr>
<th>When</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>When</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>When</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>When</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Time-Triggered Paradigm
Time-Triggered Communication

- Synchronous system
- Common communication schedule
- Clock synchronization service
- Using *A Priori* Knowledge
- Non-Communication Services
Clock Synchronization

- Initial synchronization
- Re-start
- Integration
- Continuous synchronization
Clock Sync: Initial Synchronization

- Common schedule but no common notion of time available (generally unknown phase shifts between nodes)
- Number of nodes willing to perform initial synchronization generally unknown (need to deal with non-tolerable faults)
- Event-triggered
- Conditional timeliness guarantees required
Clock Sync: Re-Start

- Phase shift between (correct) nodes bounded
- Number of (correct) nodes may be assumed to be bounded
- (Conditional) timeliness guarantees required
- Self stabilization
Clock Sync: Integration

- Algorithm to allow for joining a synchronized set of nodes
- Distinctive (node needs to be able to tell integration from other sync phases)
- Timeliness guarantees required
- Need for self-revealing schedule (either implicit or explicit)
Clock Sync: Continuous Synchronization

• Accounts for inevitable drifts between local clocks of distinct nodes
• Implements “global clock” with slot granularity in TT communication system
• Generally needs to deal with faulty nodes
Clock Sync: Continuous Synchronization (ctd.)

- Determine points at which synchronization is to be performed
- Reading clocks of remote nodes
- Executing convergence / agreement
  - Convergence: Using (FT) averaging function to compute local correction from readings
  - Agreement: Using reading from particular node to be used as reference
Time-Triggered Communication

• Synchronous system
• Common communication schedule
• Clock synchronization service
• Using A Priori Knowledge
• Non-Communication Services
Using A *Priori* Knowledge: Message Identification

- Target address contained in a message results in space partitioning problem
- Arrival time of particular message implicitly contains its ID
- Arrival time of particular message may imply sender ID
Using A Priori Knowledge: Clock Synchronization

- Trigger for clock synchronization may be derived from common schedule
- Difference between actual arrival time and expected arrival time is measure for deviation between sender’s clock and receiver’s clock
- Depending on particular TT communication protocol clock synchronization may do without any explicit information (neither in schedule nor in messages)
Using *A Priori* Knowledge: Clock Synchronization (ctd.)

- Synchronization (convergence function) is executed at defined point within schedule (e.g., at the begin of a round)
- Every node is granted exactly one sending slot per round
Using A *Priori* Knowledge: Flow Control

• Message load in peak-load scenario is known before taking the system to operation
• Push/pull interfaces
• Jitter and latency of message transport are (assumed to be) known
Using *A Priori* Knowledge: Enforcing Fault Containment
Using A *Priori* Knowledge: Enforcing Fault Containment (ctd.)
Using *A Priori* Knowledge: Enforcing Fault Containment (ctd.)
Using *A Priori* Knowledge: Enforcing Fault Containment (ctd.)

- Guardian is distinct component and, thus, fault containment region
- Guardian basically needs same functionality as communication controller (schedule, synchronized time)
- Guardian prohibits timing error propagation of nodes
Time-Triggered Communication

- Synchronous system
- Common communication schedule
- Clock synchronization service
- Using *A Priori* Knowledge
- Non-Communication Services
Non-Communication Services

• Synchronized global time base
  – Smaller granularity (time stamping, …)
  – Larger granularity (agreed round counters)
• Acknowledgement
• Group membership
• Remote procedure calls
• …
Overview

• Introduction
• Related Concepts
• Design Principles
• Time-Triggered Communication
• Conclusion
Conclusion

• Design of TT systems is guided by small set of concepts
• TT systems provide simple, deterministic computing infrastructure
• TT systems are already used in safety-critical applications in aerospace applications
• TT systems are likely to find their way to automotive applications
http://www.vmars.tuwien.ac.at/
Thank you & enjoy the coffee break!