UCIrvine
University of California, Irvine

Time-triggered Message-triggered Object Programming Scheme

K.H. (Kane) Kim
Director of DREAM Laboratory
University of California, Irvine

Juan A. Colmenares
DREAM Lab
University of California, Irvine

*Presented by **Juan A. Colmenares** at the 2006 Monterey Workshop*

October 17th, 2006, Paris, France

Low Productivity Characterizes Real-time Programming

- Providing time guarantees is essential for distributed real-time (DRT) applications
  - Very complicated if many factors impact response times
- Conventional practice leads to low productivity
  - Avoid most of the software layers
  - Implement application software in C or assembly
- Increasing demands for challenging new applications that require precisely timing actions
  - Distributed multimedia processing, drive-by-wired, and time-sensitive health care
- This practice cannot continue

Time-triggered Message-triggered Object (TMO) Programming Scheme

- Initiated in the early 90's
- A programmatic approach for facilitating the development of DRT applications
  - Contains only high-level intuitive and yet precise expressions of timing requirements
  - From the beginning the objective was to
    - Enable design-time guarantees of timely actions

3

Time-triggered Action
Essence of Real-time Programming

At time T do S

{ = Start S during [T - Δ, T + Δ] }

- S may be
  - A single or compound statement, or a function (assume the last one)
- A control signal for activating S in a node is derived from the progression of (real) time
  - A control signal is generated whenever the real-time clock within a node reaches the preset value T specified in a scheduling table

4

Time-triggered Objects
Object-oriented Support for Time-triggered Actions

- Include a new group of member functions
  - Spontaneous methods (SpMs)
    - Also called time-triggered methods
- SpMs are executed within specified time windows

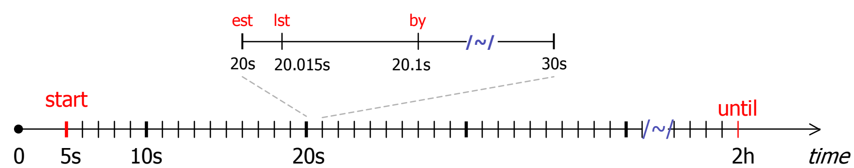
Specification of Execution Time Windows of SpMs

- Example 1
  - Start-during (9am, 9:15am)
  - Finish-by 9:40am
- Example 2
  - For t = from 10am to 10:50am, every 30min
  - Start-during (t, t+5min)
  - Finish-by t+10min
- It is intuitive and explicit

(Examples 1 and 2: Autonomous Activation Condition (AAC))

An AAC Concrete Example

- MicroSec from = 5 * 1000 * 1000; // 5 s
- MicroSec until = 2 * 60 * 60 * 1000 * 1000; // 2 h
- MicroSec every = 1 * 1000 * 1000; // 1 s
- MicroSec est = 0; (early start time)
- MicroSec lst = 15 * 1000; // 15 ms (late start time)
- MicroSec by = 100 * 1000; // 100 ms



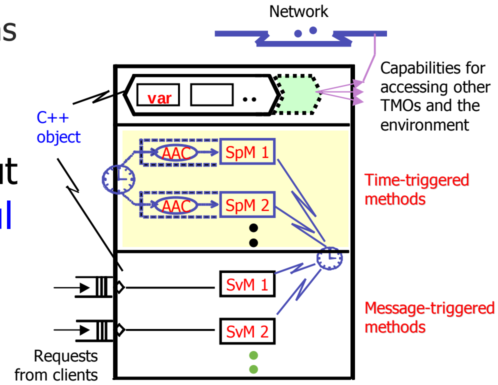
Remote Method Calls

Fundamental Mechanism in Distributed Computing

- Provide transparency in terms of
  - Location
  - Low-level communication protocols (e.g., TCP/IP)
- Message-triggered object
  - Implement remote methods
    - Also called service methods (SvMs)
    - Activated by messages from clients
- Real-time application designers must provide guaranteed completion times of SvMs

TMO
Time-triggered Message-triggered Object

- Object that also supports
  - Time-triggered actions
  - Remote method calls
- A natural and syntactically small but semantically powerful extension of the conventional object structure



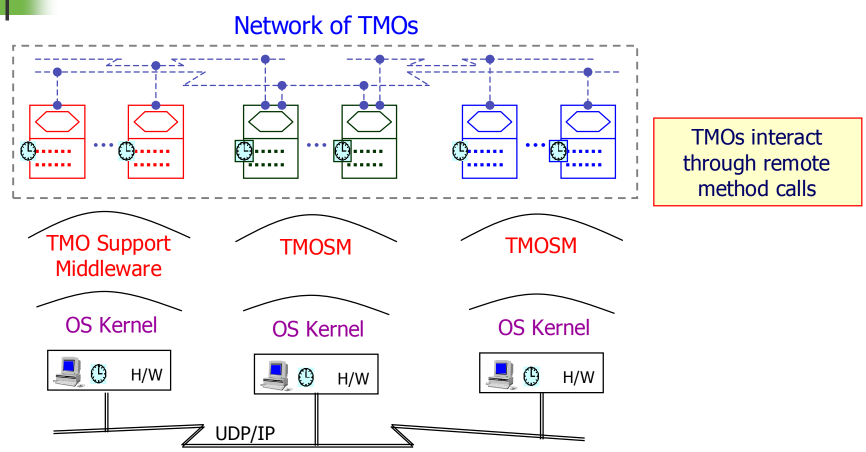
9

First Advantages of the TMO Programming Scheme

- No concerns with
  - Processes and threads
  - Object locations
  - Communication protocols
- No specification of timing requirements in indirect terms
  - e.g., priorities

10

Structure of TMO-based Distributed Real-time (DRT) Application



Coordination of Distributed Time-triggered Actions

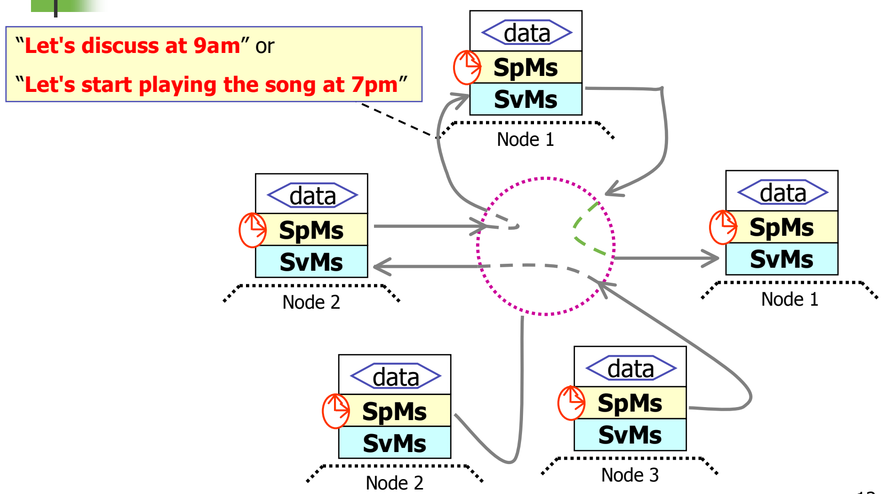
- A DRT system should be able to perform coordinated time-triggered actions that take place in different nodes

  At time T do S,
  { = Start S during [T - Δ, T + Δ] }
  where S = [TMO_0 does S_0, TMO_1 does S_1, …]

- Key requirement
  - A global time base accessible to all nodes
    - i.e., clock synchronization

TCoDA

Global Time-based Coordination of Distributed Actions



13

TCoDA

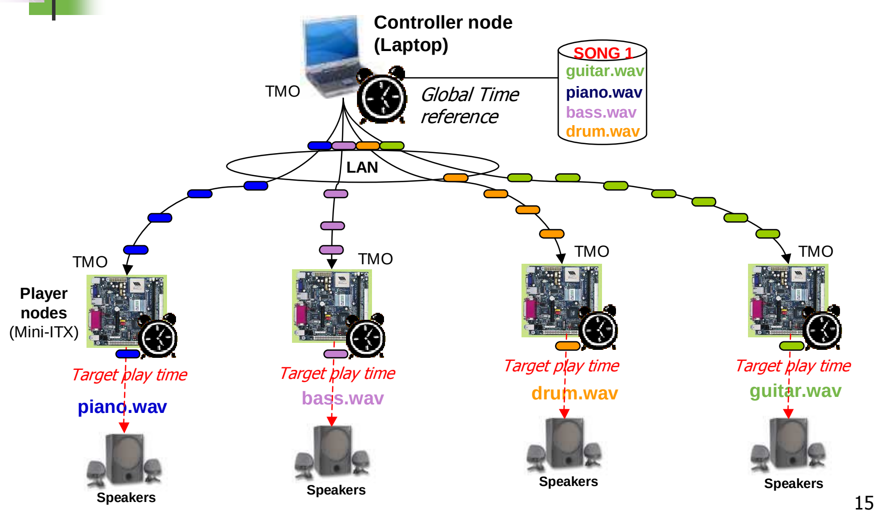
Global Time-based Coordination of Distributed Actions

- It may loosen the coupling among subsystems, and improve efficiency
  - Several nodes can be designed so that they simultaneously start to perform certain actions at 10:00am without exchanging any message if they observe certain conditions by 9:59am
    - That is, less number of messages to exchange between nodes

14

Digital Music Ensemble

Demonstrating the Power of TCoDA and TMO



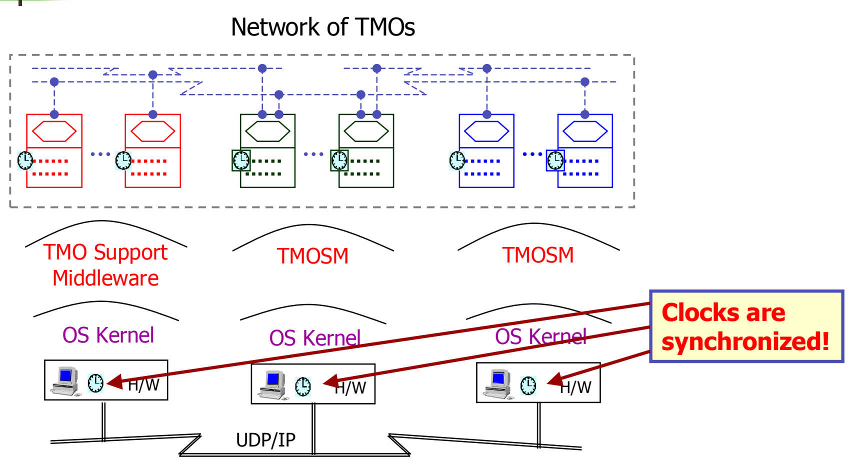
15

TCoDA Has Not Been Sufficiently Explored

- Incorrect perception about the difficulty of establishing a sufficiently precise global time base
- Reality is far better
  - 100 μs-level-precision
    - Conventional LAN with strict control over the network traffic
  - 1 μs-level-precision
    - Nodes equipped with GPS receivers capturing time announcements with microsecond-level accuracy, even if nodes are dispersed over an area larger than a campus

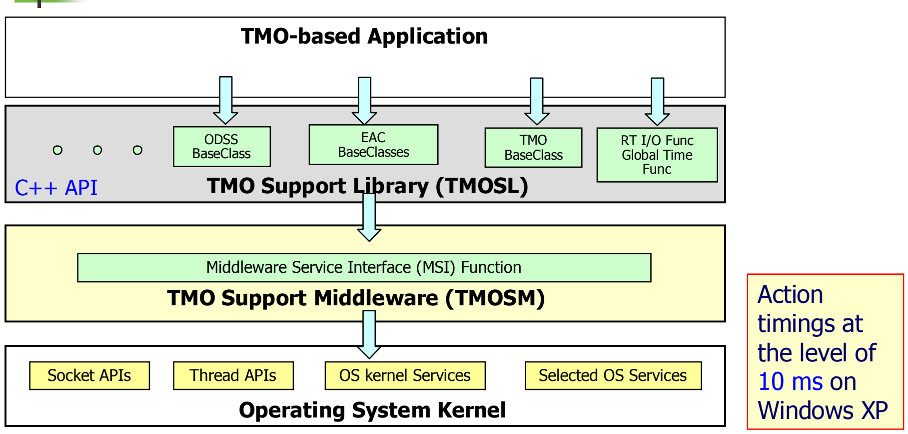
16

TMO Supports the TCoDA Principle



17

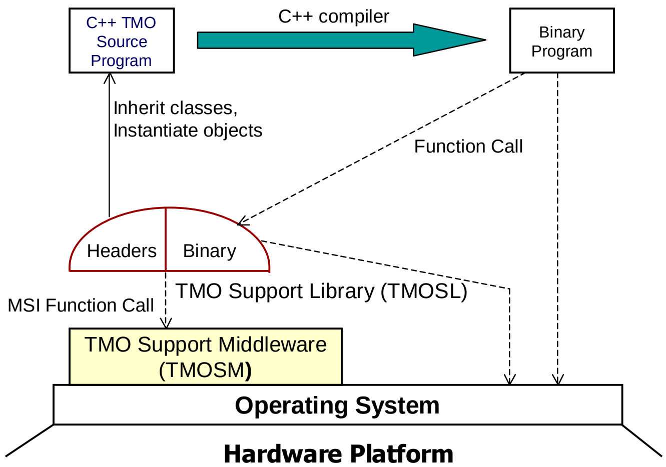
TMOSM and TMOSL



Supported OSs: Windows XP, Windows CE, and Linux 2.6

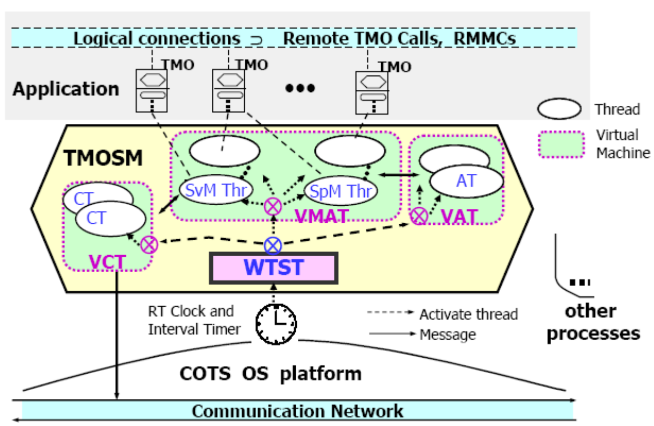
18

TMO Programming Environment



19

TMOSM Architecture



WTST: Watch-dog Timer and Scheduler Thread **VCT**: VM for Communication Threads
VMAT: VM for Main Application Threads **VAT**: VM for Auxiliary Threads

20

A More Detailed View of TMO Programming Scheme

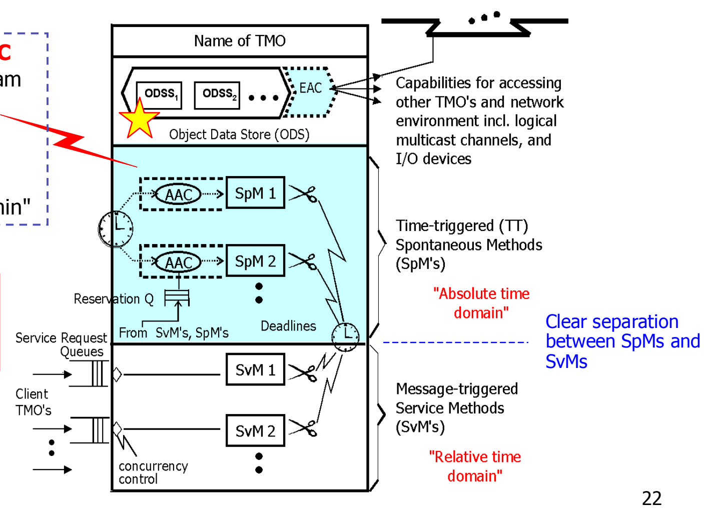
21

Revisiting Spontaneous Methods

Example of AAC
"for t = from 10am
to 10:50am
every 30min
start-during
(t, t+5 min)
finish-by t+10min"

SpMs must be registered to the execution engine

Generally done in the constructor



22

ODSS
Object Data Store Segment

- A group of data members that represents part of the internal state of the enclosing TMO
- Basic unit of storage, which can be
  - Locked for exclusive access by a certain TMO method execution
  - Shared by multiple concurrent executions of TMO methods
- Concurrency control
  - When a TMO method (SpM or SvM) is invoked, all the ODSSs to be accessed during that method execution are locked before the execution begins

23

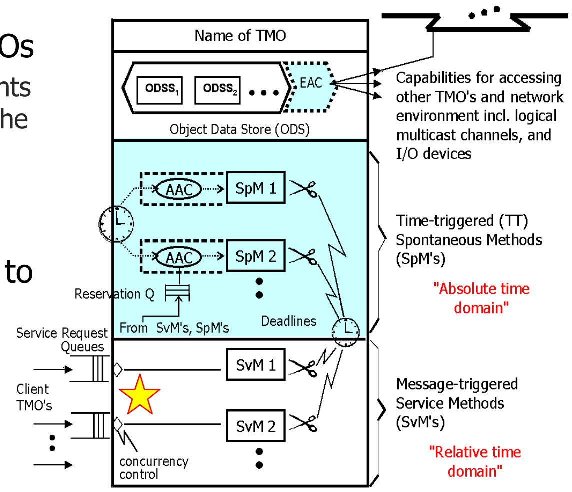
ODSS
Object Data Store Segment

- my_odss.ReleaseODSS ();
  - Used when the associated ODSS will not be accessed during the remainder of the method execution
- Such an early release enables earlier initiation of other TMO method executions
- Once a ODSS is released via ReleaseODSS (), it cannot be locked again in the method execution

24

SvMs
Service Methods

- Invoked by client TMOs
  - Local and remote clients call a SvM exactly in the same way
- SvMs are ordinary methods of a TMO (sub)class registered to TMOSM



25

Timing Specification of SvMs

- The registration of an SvM includes the following parameters:
  - Guaranteed execution time bound (GETB)
  - Maximum number of concurrent executions (PipelineDegree)
  - Maximum invocation rate (MaxInvocations) and minimum invocation interval (BasicPeriod)
    - Statistical assurances of better service times

Optional

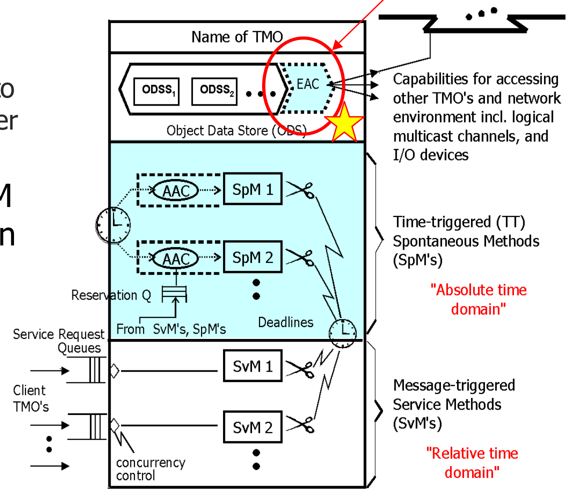
- Other required parameters are:
  - The names and access modes (RO/RW) of ODSSs
  - The external name of the SvM
    - A globally recognized symbolic name

26

Invocation of SvMs

Environment Access Capability: an ODS extension

- Gate objects
  - EACs that allow us to invoke SvMs on other TMOs
- Client's call to a SvM (through a gate) can be associated with deadline for result return

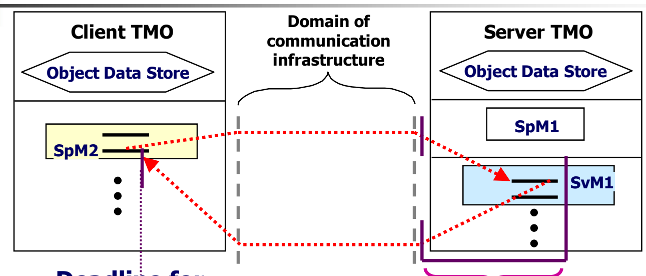


Timing Specification in the Invocations of SvMs

- SvMGateClass gate1 (_T("TMO2"), _T("SvM2"), …);
- gate1.BlockingSR (&param, sizeof(param), dra, ort);
  - dra = Deadline Result Arrival
  - ort = Official Release Time
    - Time at which the invoked SvM should be executed
    - If 0, SvM is executed ASAP

Toshiba has a patent on the idea of using ORT !
Invented in 1996 and US Patent was granted in April 2001, but we learned the invention only in February 2005 although we have been using it since mid-1999.

Return Deadline vs. Guaranteed Service Time



Deadline for Result Arrival (DRA)
A **violation** of this deadline is a **fault**.

Guaranteed Execution Time Bound (GETB)
If GETB is violated, **fault handling actions** must take place.

Required condition

DRA - Call initiation time **>**
Max. trans. time imposed on comm. infrastructure + **GETB** **>**
Time consumed by communication infrastructure + **GETB**

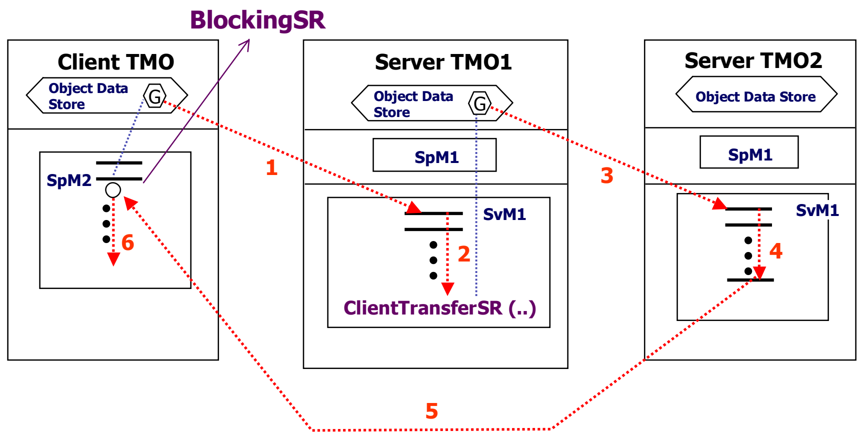
29

Type of SvM Invocations

- Blocking calls
  - with return deadlines imposed and official release times
- Non-blocking calls and subsequent result checks
  - with deadlines imposed and official release times
- One-way calls
  - With official release times
- Client-transfer calls
  - An SvM passes the client's request to another SvM, and the latter returns the result to the client

30

SvM Client-Transfer Call

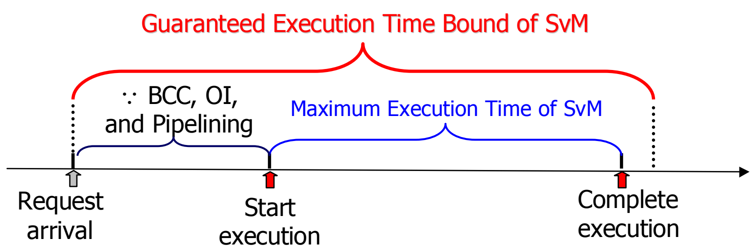


BCC
Basic Concurrent Constraint

- SpM executions are given higher priority over SvM executions
  - An SvM is allowed to execute only if there is no SpM that requires access to the same ODSS and will execute in the time window of this SvM
- It prevents potential conflicts between SpMs and SvMs and reduces the designer's efforts in guaranteeing timely service capabilities of TMO
  - Note that this BCC does not impose any restriction on concurrent execution of SpMs or concurrent execution of SvMs
- Causes potential SvM starvation

BCC
Basic Concurrent Constraint

- In general, the maximum execution time of an SvM depends on how many SvMs and non-conflicting SpMs compete for machine resources



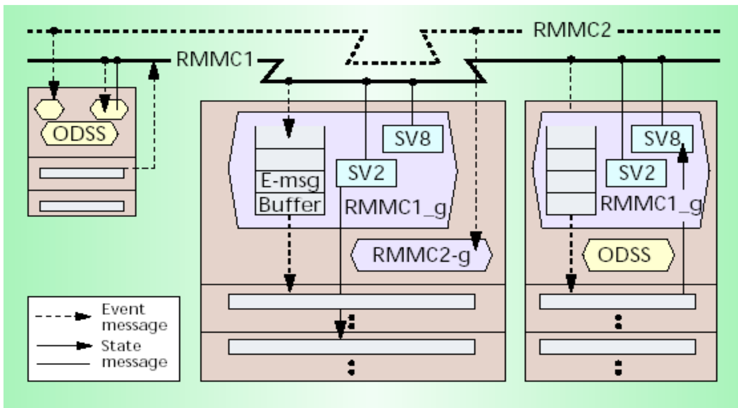
RMMC
Real-time Multicast and Memory Replication Channel

- TMO also supports
  - Multicast of event messages
    - An event message must be read by every corresponding subscriber
  - Replication of state messages
    - Consumer are interested in the current state
- The producer timestamps the message at message-production time
- RMMC also supports Official Release Time

RMMC

Real-time Multicast and Memory Replication Channel

- RMMCs accessed by TMO methods
  - Access gates for 2 RMMCs (RMMC1 and RMMC2) declared in the TMOs



35

Summary

- TMO programming scheme
  - Simplifies the development of DRT applications
  - Allows us to specify the timing requirements of our DRT applications in explicit and intuitive manners
  - Supports fundamental principles of real-time programming
    - Time-triggered actions
    - Global-time-based coordination of distributed actions (TCoDA)

36

Summary

TMO interaction mechanisms	Supported features	Timing specification
Sporadic Methods (**SpMs**)	Time-triggered actions	AAC { **For t = from 10am to 10:50am, every 30min** **Start-during (t, t+5min), Finish-by t+10min** }
Service Methods (**SvMs**)	Remote methods calls	Server TMO { **Guaranteed execution time bound**, …}
		Client TMO { **Deadline for Result Arrival**, **Official Release Time** }
Real-time Multicast and Replication Memory Channel (**RMMC**)	Multicast of event messages and replication of state messages	Producer { **Official Release Time** }

37