



## 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 17<sup>th</sup>, 2006, Paris, France

## Low Productivity Characterizes Real-time Programming

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



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

- n Initiated in the early 90's
- n A **programmatic approach** for facilitating the development of DRT applications
  - n Contains only **high-level intuitive and yet precise expressions of timing requirements**
  - n From the beginning the objective was to
    - n **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] }

- n **S** may be
  - n A single or compound statement, or a **function** (assume the last one)
- n A **control signal** for activating **S** in a node is derived from the progression of (real) time
  - n 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

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

5



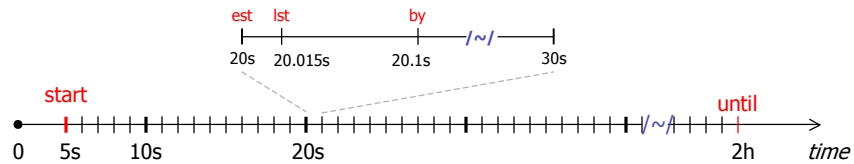
## Specification of Execution Time Windows of SpMs

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

6

## An AAC Concrete Example

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



7

## Remote Method Calls

Fundamental Mechanism in Distributed Computing

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

8

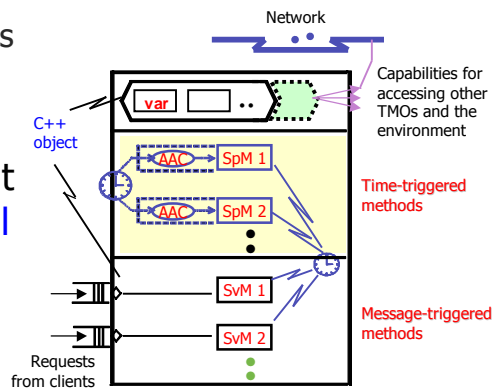
# TMO

## Time-triggered Message-triggered Object

n Object that also supports

- n Time-triggered actions
- n Remote method calls

n A **natural** and **syntactically small** but **semantically powerful extension** of the conventional object structure



9

## First Advantages of the TMO Programming Scheme

n No concerns with

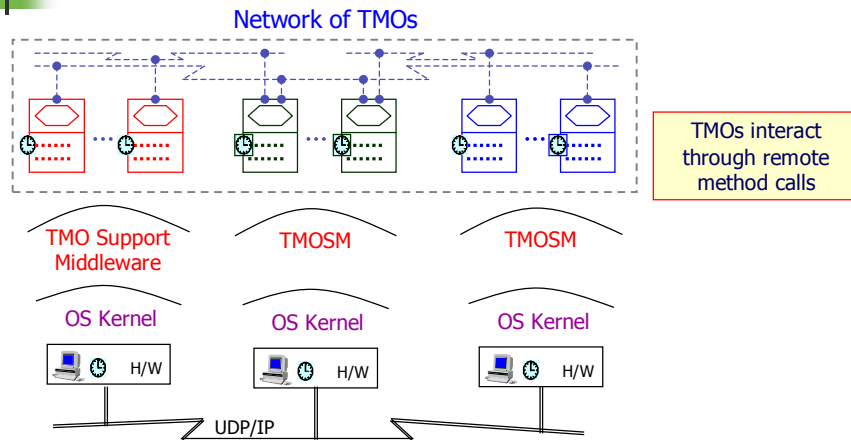
- n Processes and threads
- n Object locations
- n Communication protocols

n No specification of timing requirements in indirect terms

- n e.g., priorities

10

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



11

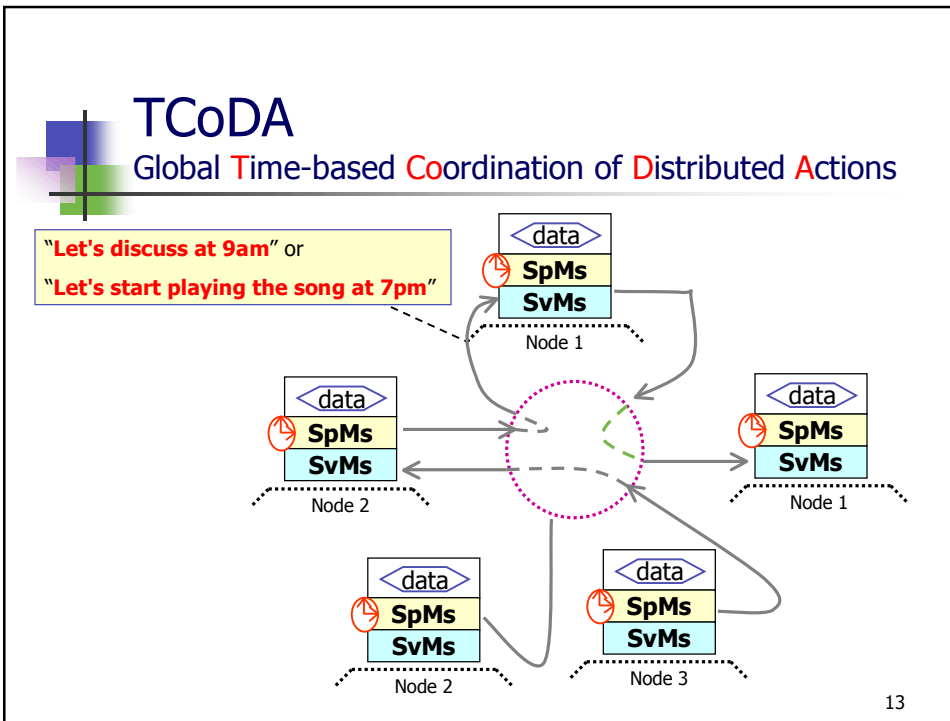
## Coordination of Distributed Time-triggered Actions

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

**At time  $T$  do  $S$ ,**  
 $\{ = \text{Start } S \text{ during } [T - \Delta, T + \Delta] \}$   
**where  $S = [ \text{TMO}_0 \text{ does } S_0, \text{TMO}_1 \text{ does } S_1, \dots ]$**

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

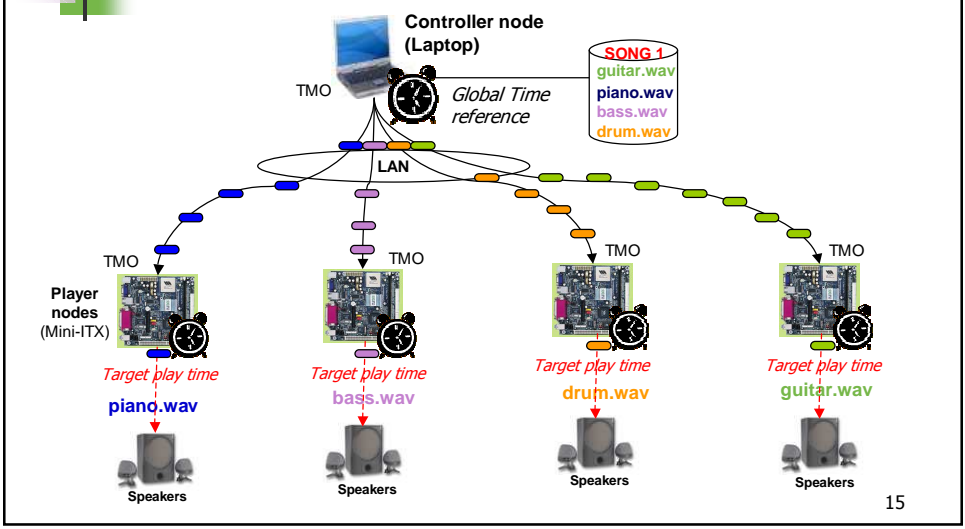
12



- ## TCoDA
- Global Time-based Coordination of Distributed Actions
- n It may **loosen the coupling among subsystems,** and **improve efficiency**
    - n 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
      - n That is, less number of messages to exchange between nodes
- 14

# Digital Music Ensemble

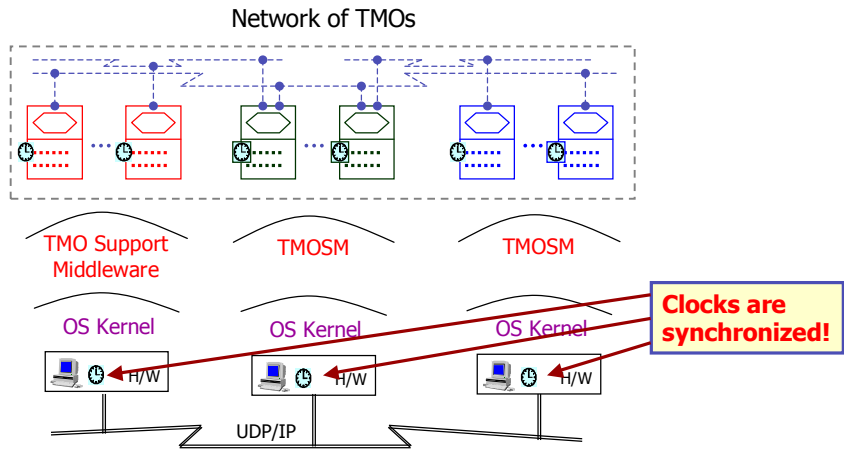
## Demonstrating the Power of TCoDA and TMO



# TCoDA Has Not Been Sufficiently Explored

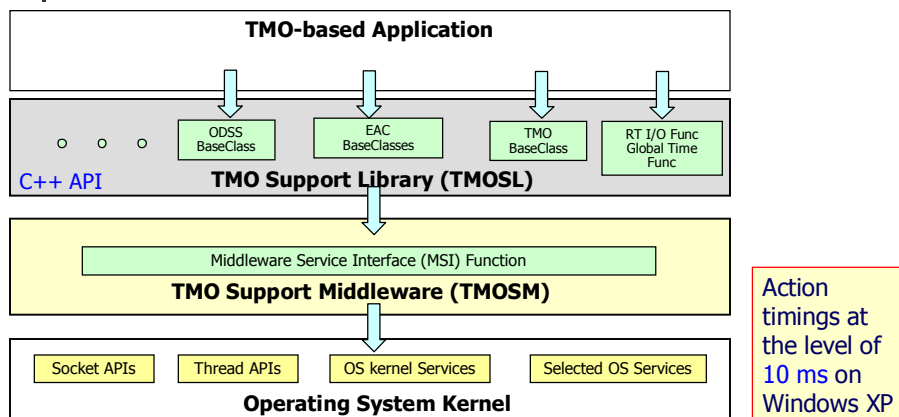
- n **Incorrect perception** about the difficulty of establishing a sufficiently precise global time base
- n **Reality is far better**
  - n 100  $\mu$ s-level-precision
    - n Conventional LAN with strict control over the network traffic
  - n 1  $\mu$ s-level-precision
    - n **Nodes equipped with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

# TMO Supports the TCoDA Principle



17

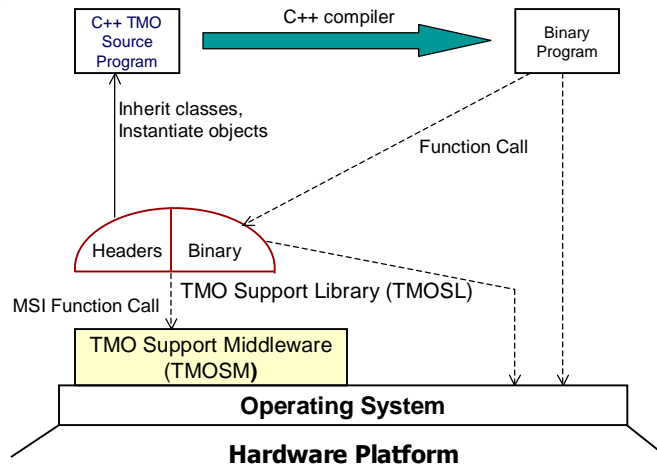
# TMOSM and TMOSL



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

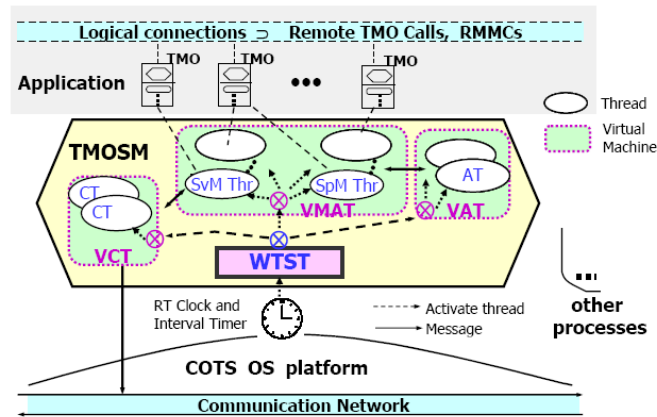
18

# 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

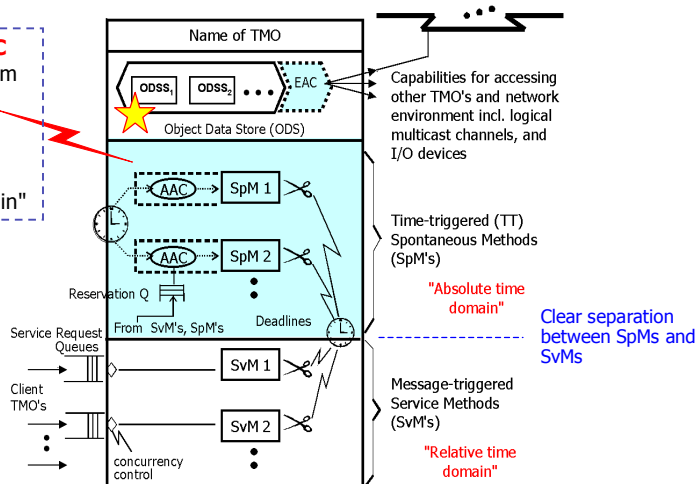
# Revisiting Spontaneous Methods

**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





## ODSS

### Object Data Store Segment

---

- n A group of data members that represents part of **the internal state** of the enclosing TMO
- n **Basic unit of storage**, which can be
  - n **Locked for exclusive access** by a certain TMO method execution
  - n **Shared** by multiple concurrent executions of TMO methods
- n **Concurrency control**
  - n 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

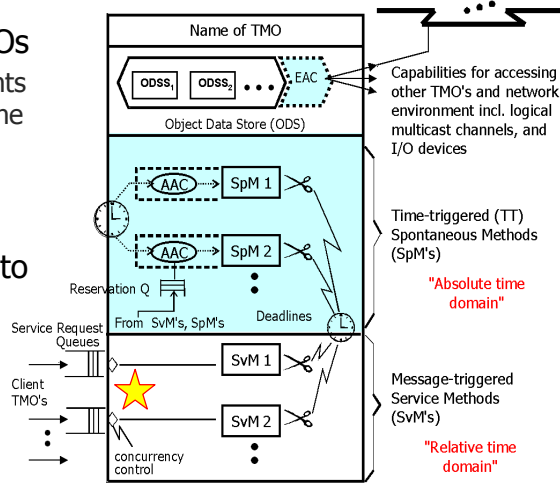
---

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

24

# SvMs Service Methods

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



25

# Timing Specification of SvMs

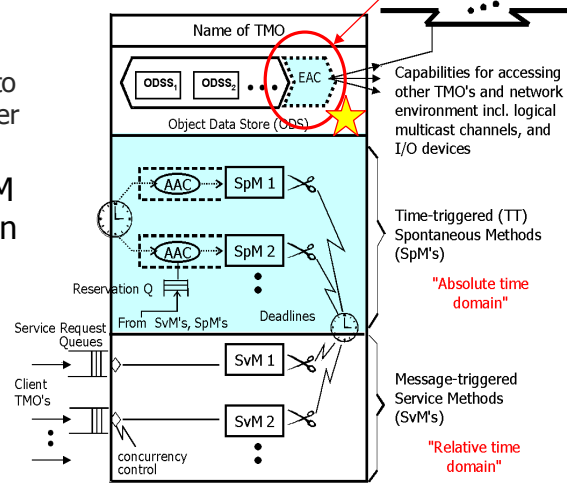
- n The registration of an SvM includes the following parameters:
  - n **Guaranteed execution time bound (GETB)**
  - Optional {
    - n Maximum number of concurrent executions (**PipelineDegree**)
    - n Maximum invocation rate (**MaxInvocations**) and minimum invocation interval (**BasicPeriod**)
      - n Statistical assurances of better service times
- n Other required parameters are:
  - n The names and access modes (RO/RW) of ODSSs
  - n The external name of the SvM
    - n A globally recognized symbolic name

26

# Invocation of SvMs

**Environment Access Capability: an ODS extension**

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

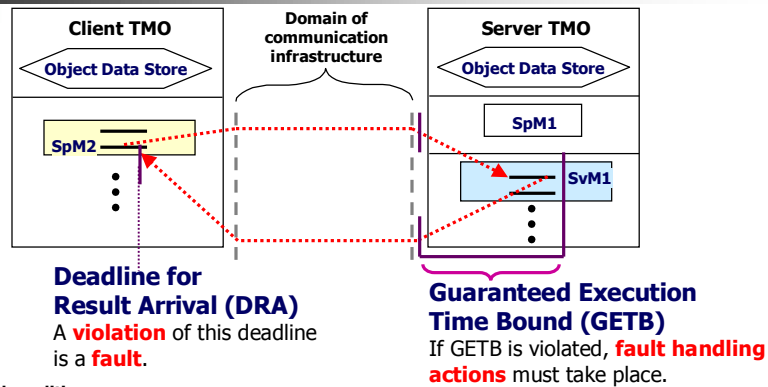


# Timing Specification in the Invocations of SvMs

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



Required condition

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

29

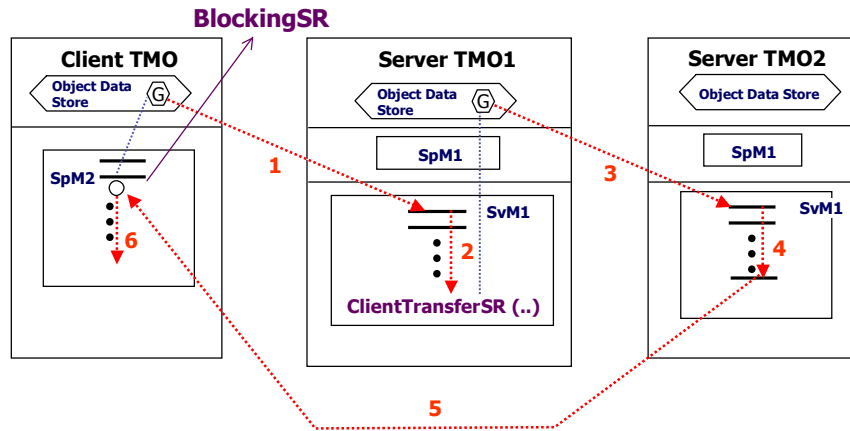
## Type of SvM Invocations

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

30



## SvM Client-Transfer Call



31



## BCC Basic Concurrent Constraint

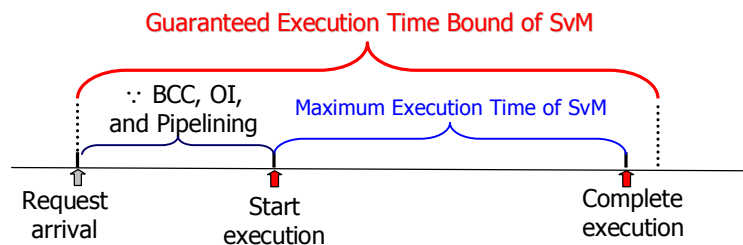
- n SpM executions are given higher priority over SvM executions
  - n An SvM is allowed to execute only if there is no SpM that requires access to the same ODSS and will execute in the time window of this SvM
- n It prevents potential conflicts between SpMs and SvMs and reduces the designer's efforts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

32

## BCC

### Basic Concurrent Constraint

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



33

## RMMC

### Real-time Multicast and Memory Replication Channel

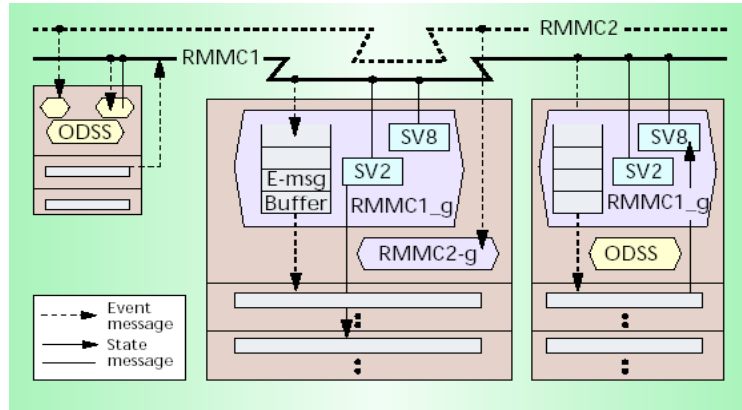
- n TMO also supports
  - n Multicast of event messages
    - n An event message must be read by every corresponding subscriber
  - n Replication of state messages
    - n Consumer are interested in the current state
- n The **producer timestamps the message at message-production time**
- n RMMC also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

34

# RMMC

## Real-time Multicast and Memory Replication Channel

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



35

# Summary

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

36



# Summary

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