

# "The Distributed Time-Triggered Simulation Scheme : Core Principles and Supporting Execution Engine"

**K. H. (KANE) Kim**

DREAM Lab.  
Department of EECS  
UC Irvine

Mar-07	1
--------	---

UCI  
DREAM Lab



## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- An Overview of the TMO Scheme
- TMO-Structured DTS
- Major Issues in Efficient Implementation of DTS
- An Execution Engine for TMO-Structured DTS
- Conclusion

Mar-07	2
--------	---

UCI  
DREAM Lab



## Outline

---

- Introduction
  - Main Challenges in Distributed RT Simulation and the DTS Framework
  - An Overview of the TMO Scheme
  - TMO-Structured DTS
  - Major Issues in Efficient Implementation of DTS
  - An Execution Engine for TMO-Structured DTS
  - Conclusion

Mar-07

3

UCI  
DREAM Lab



## Motivation

---

- **Object-oriented (OO) real-time (RT) distributed computing** has become a rapidly growing branch of computer science and engineering.
- The **demands for RT simulator developments** are steadily increasing in the area of next-generation virtual reality applications and RT computer control applications such as those found in manufacturing plants.
- **Real-Time Simulation** := Accurate mode of simulation in which the simulator components (or *simulator objects*) show the timing behavior that are the same as or similar to the timing behavior of the simulation targets (i.e., simulated entities).
- As the complexities of RT simulators grow, the use of **distributed / parallel RT simulation approaches** becomes imperative. However, **effective decomposition of a simulation model for parallel RT execution** is not a well-understood subject.

Mar-07

4

UCI  
DREAM Lab



## Motivation (cont.)

---

- In addition, **practical distributed RT simulation techniques** have not been established in sufficiently reliable forms for industrial use.
- The most fundamental challenge is **maximizing the concurrency** in RT distributed simulation **while maintaining the consistency** of distributed RT simulator nodes.

Mar-07

5

UCI  
DREAM Lab



## Distributed Time-triggered Simulation

---

- A new approach of RT simulation which is conceptually simple and easy to use but widely applicable has developed  
=> **Distributed Time-triggered Simulation (DTS)**
- How things work in DTS
  - Simulation targets are modeled as TMOs and the simulation engine based on a parallel or distributed computing platform faithfully executes TMO models to effect RT simulation.
  - TMO is capable of representing uniformly and with variable degrees of precision both RT embedded computer systems and their application environments.
- Key ideas of DTS
  - Let distributed simulator nodes or simulator objects advance to the next simulation step simultaneously when the globally available RT clock reaches a certain time-point.
  - Remove costly message exchanges for sync among simulating nodes.

Mar-07

6

UCI  
DREAM Lab



## Distributed Time-triggered Simulation (cont.)

---

- Experimental research examples
  - Coordinated Anti-Missile Interceptor Network (CAMIN) based on a defense command-control application scenario
  - Distributed Object-oriented Freeway Simulator (DOFS) based on an advanced freeway traffic control scenario

Mar-07

7

UCI  
DREAM Lab



## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
  - An Overview of the TMO Scheme
  - TMO-Structured DTS
  - Major Issues in Efficient Implementation of DTS
  - An Execution Engine for TMO-Structured DTS
  - Conclusion

Mar-07

8

UCI  
DREAM Lab



## Basic Requirements in RT Simulation

- Every computer-based simulator has a **simulator clock** for driving new simulation activities (a new **simulation step**).
- Simulator clock must be based on an RT clock to tick at a steady rate.
- All computational activities taking place during a ticking interval of the simulator clock may be viewed as one **simulation step**.
- The **ticking rate** of the simulator clock in an RT simulator must be chosen with the following understanding:

*For all (real-time) events which should be simulated during any ticking interval of the simulator clock, the exact microscopic timings of those events may be transparent to the user of the RT simulator.*

- Only the resulting state of the simulation at the end of the ticking interval may be seen by the user.
- The ticking interval must be long enough to accommodate the message communication for the essential data flow among distributed simulator objects.

Mar-07

9

UCI  
DREAM Lab



## Distributed Time-Triggered Simulation

- Simulation targets are modeled as **RT objects** such as TMOs which **support time-triggered methods** in addition to the normal service methods of the conventional objects.
- Simulation engine based on a parallel or distributed computing platform faithfully **executes RT object models** to effect RT simulation.
- Distributed simulator nodes or objects advance to the next simulation step simultaneously **when the globally available RT clock reaches a certain time-point**.

Mar-07

10

UCI  
DREAM Lab



## Distributed Time-Triggered Simulation (cont)

- Essence of the DTS approach
  - Every node is equipped with an RT clock and **executes each simulation-step upon reaching of the RT clock at the predetermined value.**
  - Every simulation-step is designed to be **completed within one ticking interval.**
- Major advantages of the DTS approach
  - **Synchronization** of simulation-steps executed by distributed simulator objects under the DTS scheme **does not require message exchanges** among the host nodes.
  - DTS approach enables easy design of simulator objects which use **different ticking rates.**

Mar-07

11

UCI  
DREAM Lab



## Outline

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- **An Overview of the TMO Scheme**
- TMO-Structured DTS
- Major Issues in Efficient Implementation of DTS
- An Execution Engine for TMO-Structured DTS
- Conclusion

Mar-07

12

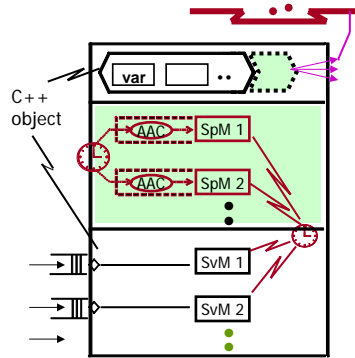
UCI  
DREAM Lab



# High-Level RT Object: TMO

## The Time-triggered Message-triggered Object (TMO) programming and specification scheme

- Meant to be a **natural easy-to-use extension** of the **C++/Java** technology into an **RT distributed software component programming** technology
- Supports design of **distributed HRT objects** and **distributed non-RT objects** within **one general structure**
- Contains **only high-level intuitive** and yet precise **expressions** of timing requirements
- Formulated from the beginning with the objective of **enabling design-time guaranteeing** of timely actions



UCI  
DREAM Lab

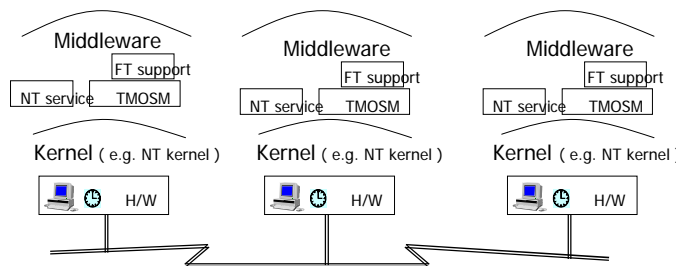
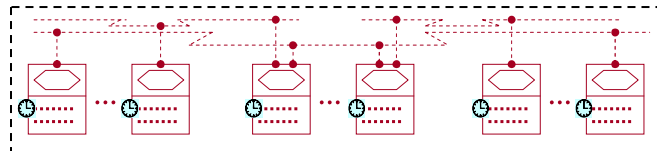


Mar-07 13

# Making Applic Programmers' Life Easier

Structure as **TMO networks** relying on intelligent execution facilities

**Real-Time Distributed Computing Applications**



No concerns with

- **Processes & Threads**
- **Object locations** (except in avoiding overloaded nodes)
- **Low-level comm protocols**

No specification of timing reqts in indirect terms (e.g., priorities)

- **Only start-windows and completion deadlines for object methods**
- **time-windows for output actions**

UCI  
DREAM Lab



Mar-07 14

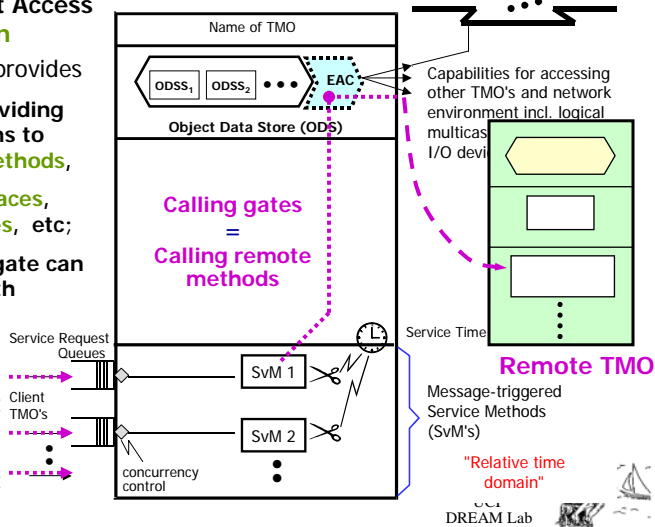
# High-level Distributed Computing Component

## Abstract-style Remote Method Calls

### EAC (Environment Access Capability) section

(an ODS extension) provides

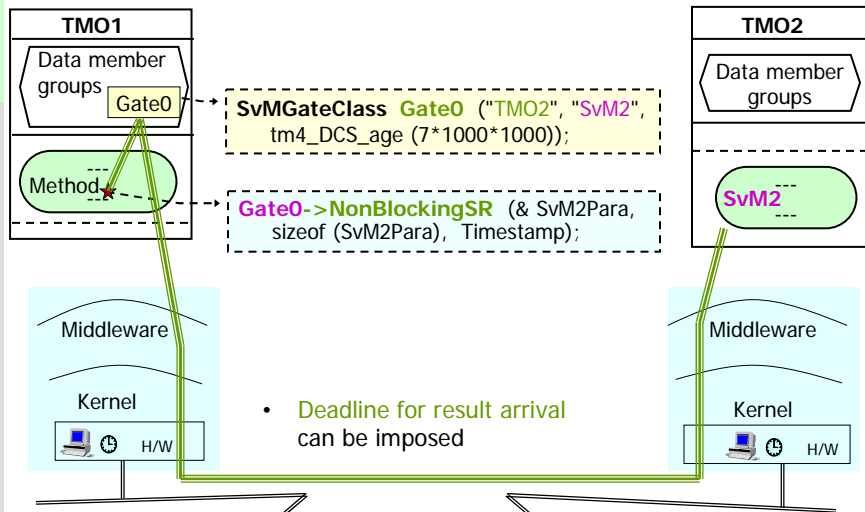
- Gate objects providing efficient call-paths to remote object methods,
- I/O device interfaces, channel interfaces, etc;
- Client's call to a gate can be associated with deadline for result return, nothing more;
- Each gate to a remote method is associated with guaranteed service or a statistical service guarantee;



Mar-07 15

UCI DREAM Lab

# Only 2 Statements for Making a Remote Method Call



Mar-07 16

UCI DREAM Lab

## Time-Triggered Spontaneous Methods (SpMs)

- Clearly separated from the **Service Methods (SvMs)** triggered by messages from clients

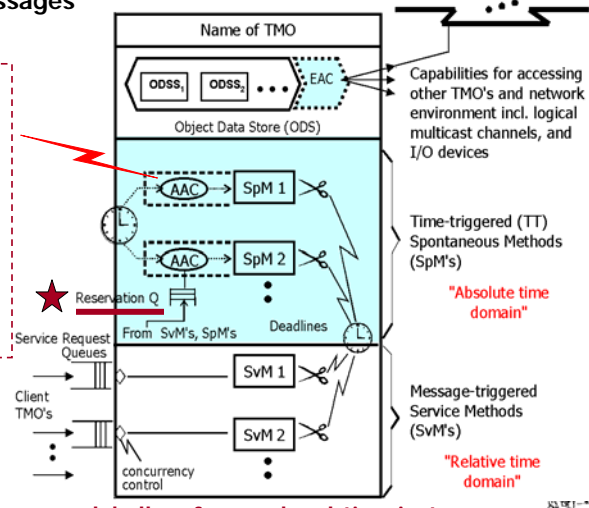
**Example of AAC:**

```

{"start-during
 (10am, 10:05am)
 finish-by 10:10am",
 "for t = from 10am
 to 10:50am
 every 30min
 start-during
 (t, t+5 min)
 finish-by t+10min" }.
    
```

Actions to be taken at **real times** determined at the **design time** appear only in SpM's

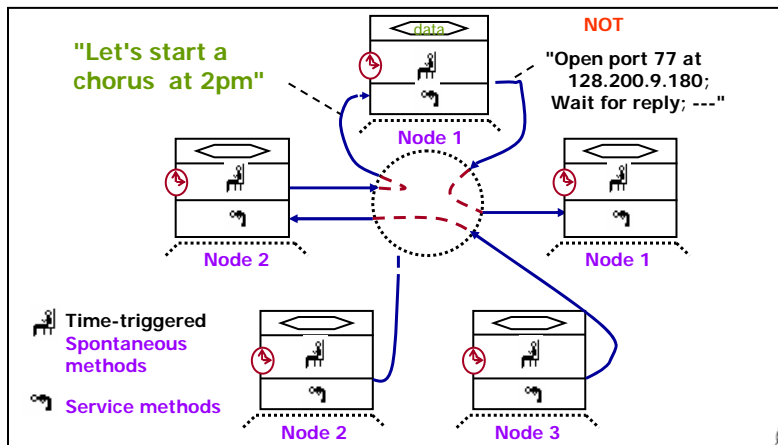
- All time references are **globally referenced real-time instances**.



## Time-Based Coordination of Distributed Actions

**TT methods play key roles**

Imagine the Advantages of  
 A group of cooperating **people with wrist-watches over**  
 A group of **people not using the globally referenced time**



## Multi-step Multi-level Design

---

- The attractive basic design style facilitated by the TMO structuring is to produce a network of TMOs meeting the application requirements in a top-down multi-step fashion.
- The engineering of an application system can start with a single TMO representation of the entire application environment (including the computer system to be designed).
- Then, proceed through step-by-step expansion of the initial single TMO model toward a final implementation in the form of a network of TMOs executing on engines.
- We will see examples of this practice later in this presentation.

Mar-07

19

UCI  
DREAM Lab



## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- An Overview of the TMO Scheme
- **TMO-Structured DTS**
- Major Issues in Efficient Implementation of DTS
- An Execution Engine for TMO-Structured DTS
- Conclusion

Mar-07

20

UCI  
DREAM Lab



## TMO-structured DTS

- DTS approach **facilitated by the TMO programming scheme**
  - Each simulation application can be modeled and constructed by one TMO or a network of TMOs (distributed TMOs).
  - Object data store (ODS) contains **state representations** of the simulated targets.
  - TT methods or SpM's **execute simulation steps and update states**.
- TT methods are mechanisms for approximately simulating **continuous state changes of target items** in the application environment.
- **Natural parallelism** can be precisely represented by use of multiple TT methods which may be activated simultaneously.
- Precision of TMO-structured simulation is a function of the **activation frequencies of TT methods** (the ticking rate of the target simulator clock).

Mar-07

21

UCI  
DREAM Lab



## Attractive Features of TMO-structured DTS

- Uniform structuring of DTS from requirement specification to the detailed implementation
- Highly predictable timing performance due to the explicitly specified timing characteristics during design time
- Systematic expansion of a TMO into a TMO network
- Easy programming and debugging of timing characteristics and concurrency control
- Efficient distributed and parallel processing in heavy-load simulations due to lack of massive message exchange for synchronization purposes

Mar-07

22

UCI  
DREAM Lab



## Examples of TMO-structured DTS

- Coordinated Anti-Missile Interceptor Network (CAMIN)
- Distributed Object-oriented Freeway Simulator (DOFS)

Mar-07

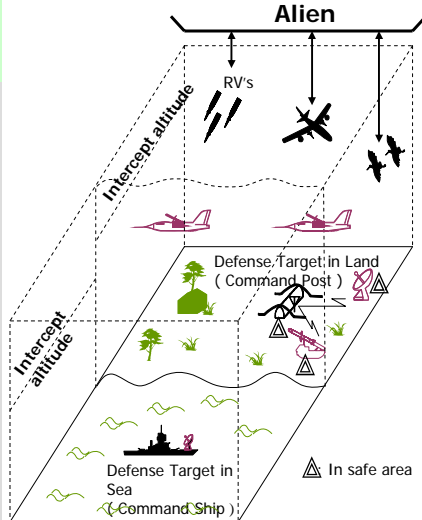
23

UCI  
DREAM Lab



## Example 1: CAMIN

**CAMIN** (Coordinated Anti-Missile Interceptor Network) Theater



Mar-07

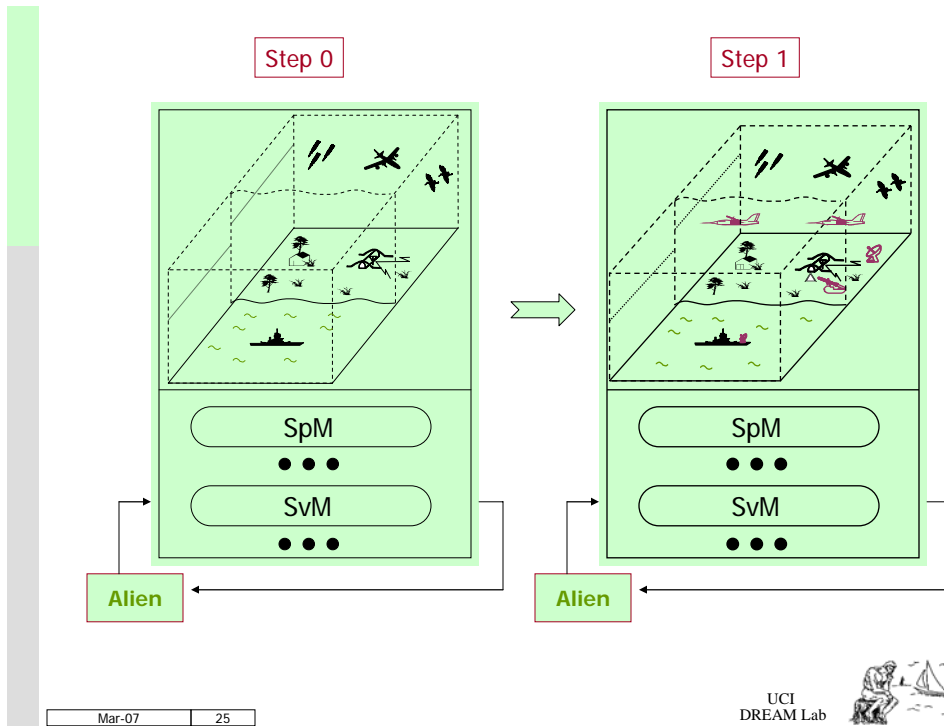
24

Theater	
Access Capability (to other TMO's)	None
Object Data Store	<ul style="list-style-type: none"> <li>• Theater Space (=Sky+Land+Sea Space)</li> <li>• Defense Target in Land (=Command Post)</li> <li>• Defense Target in Sea (=Command Ship)</li> <li>• [ Radar in Land ]</li> <li>• [ Interceptor Launcher in Land ]</li> <li>• [ Fighter Airplanes (with Interceptor Launchers) ]</li> <li>• (0 - n) RV's</li> <li>• (0 - m) NTFO's</li> <li>• [ (0 - k) Interceptors ]</li> </ul>
SpM "Update state descriptors in ODS every ms"	<ul style="list-style-type: none"> <li>• Update the state of Defense Target in Land</li> <li>• Update the state of Defense Target in Sea</li> <li>• [ Update the state of Radar in Land ]</li> <li>• [ Update the state of Interceptor Launcher in Land ]</li> <li>• [ Update the state of Fighter Airplanes ]</li> <li>• Update the state of RV's</li> <li>• Update the state of NTFO's</li> <li>• [ Update the state of Interceptors ]</li> </ul>
SvM	<ul style="list-style-type: none"> <li>• Accept RV (invoked by Alien TMO)</li> <li>• Accept NTFO (also invoked by Alien TMO)</li> </ul>

Alien

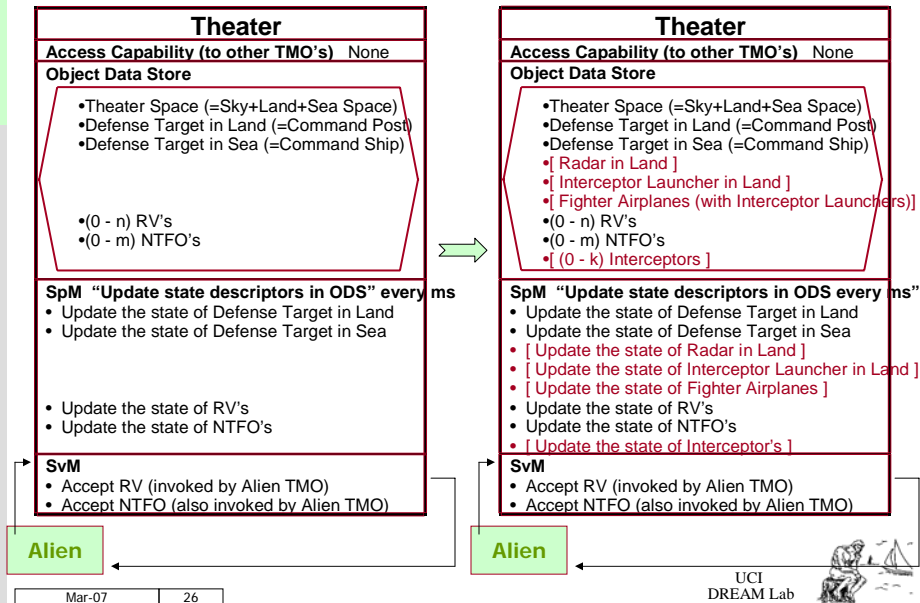
UCI  
DREAM Lab

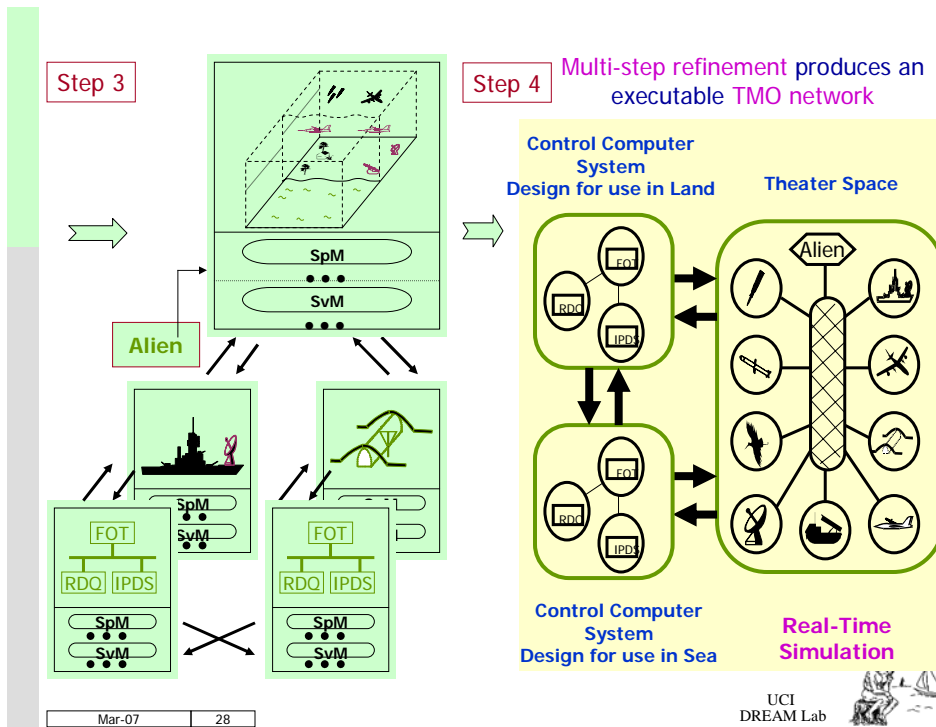
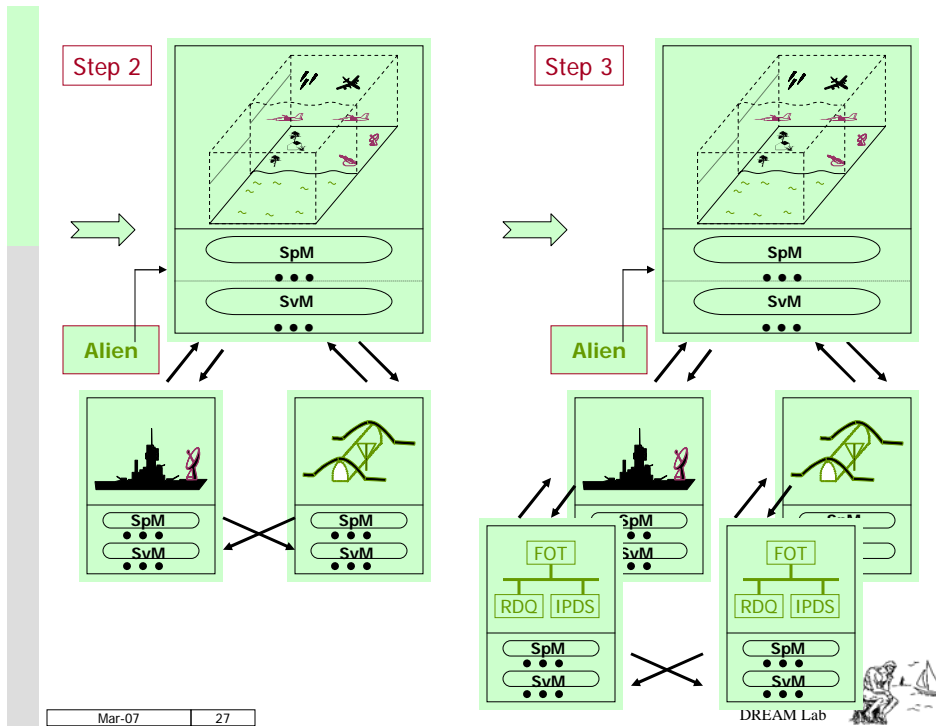




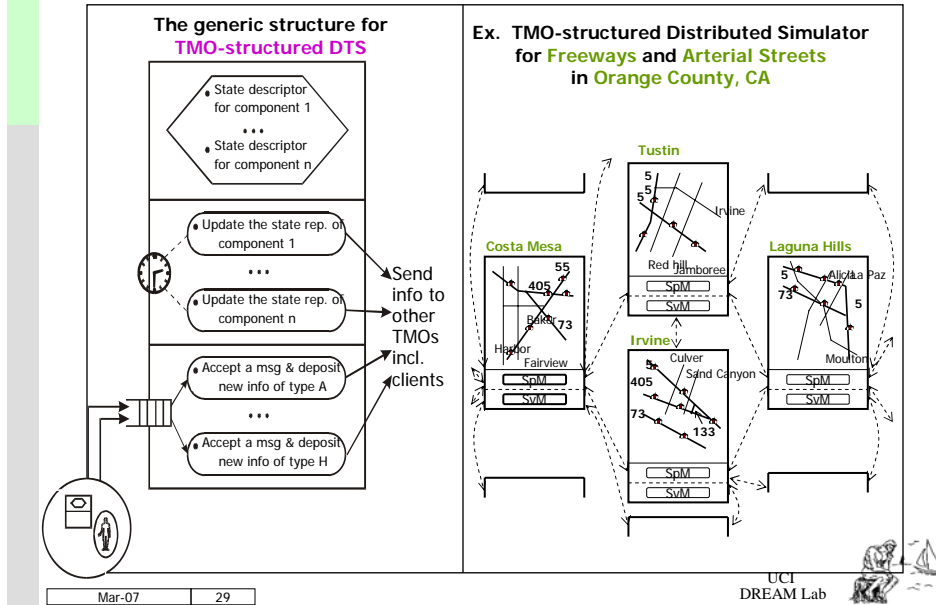
Step 0: High-level specification of the initial application environment

Step 1: High-level design of application environment: Incorporation of sensors, actuators, & control strategy

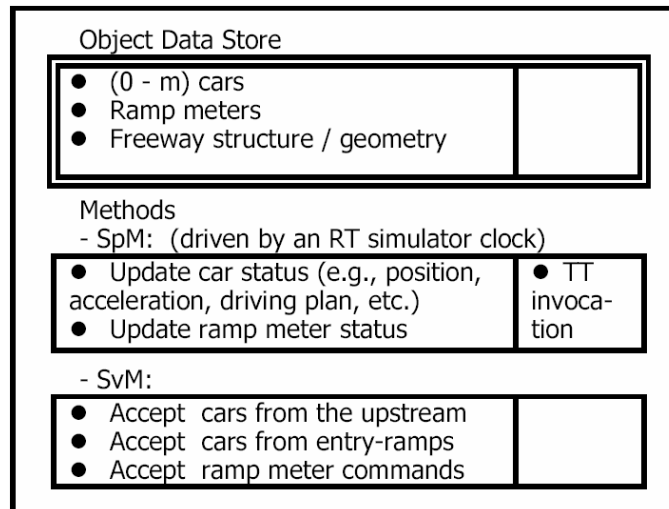




## Example 2: DOFS



## TMO-structured Simulation Model for a Freeway Segment



## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- An Overview of the TMO Scheme
- TMO-Structured DTS
- **Major Issues in Efficient Implementation of DTS**
- An Execution Engine for TMO-Structured DTS
- Conclusion

Mar-07

31

UCI  
DREAM Lab



## Global Server and Monitor (GSM) Object

---

- **Global server and monitor (GSM)** TMO supports other TMOs simulating the application scenario. For example, Theater TMO of the CAMIN simulation can be a GSM TMO.
- This GSM TMO maintains information on how the simulated target space is occupied by facilitating updating positions of simulated items, detecting collisions between moving items, notifying other TMOs of those collisions so that the notified TMOs can simulate the post-collision behavior of their simulation targets.
- Each ticking interval of the simulator clock must cover the time spent in interaction between a GSM TMO(s) and monitored TMOs.
- If the workload of a GSM object becomes too large, then multiple GSM objects, each supporting a different group of TMOs simulating the physical environment items, can be utilized.

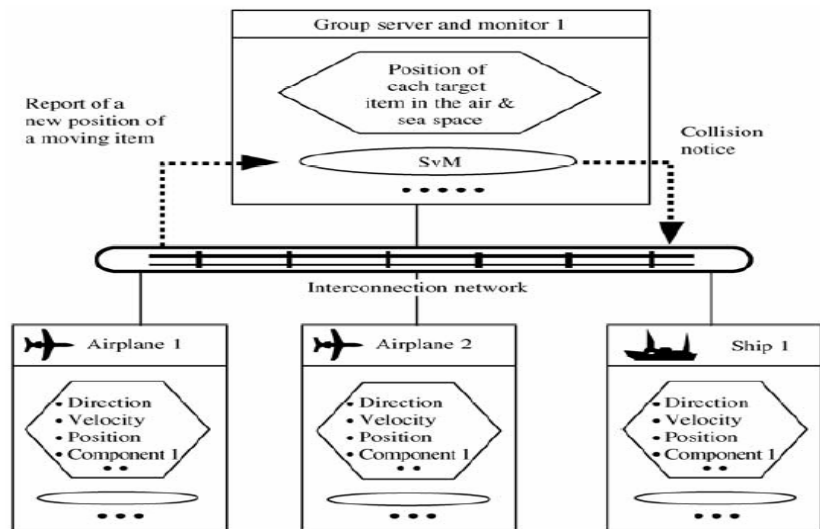
Mar-07

32

UCI  
DREAM Lab



## An Example of GSM Object in CAMIN



An expanded network of TMOs in CAMIN

Mar-07

33

UCI  
DREAM Lab



Decomposition and Update dependency are not covered in EECS123

## TMO Decomposition

- In most cases it is not worth decomposing a TMO containing tightly coupled data members into a group of TMOs supported by a GSM TMO.
- Such decomposition can be justified only if each update of the state descriptor for simulated targets is highly time-consuming and thus parallel updating of multiple TMOs is really necessary.
- The speed gain from such parallel updating should be measured against the overhead incurred in interactions between the GSM TMO and its supported TMOs.

Mar-07

34

UCI  
DREAM Lab



## Inconsistent States

- New state of simulator may be an **inconsistent** state if the resulting state cannot exist in real situations.
- The new state value must be checked whether it leads the entire simulator to a consistent state.
- Observations
  - Every simulator object has some degrees of dependency to other simulator objects in the same simulation system.
  - Some of these dependencies can affect the execution and/or performance of the simulator.
  - The characteristics of these dependencies are highly application dependent.

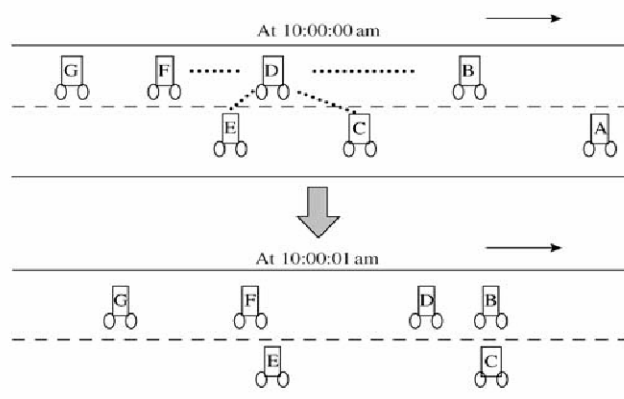
Mar-07

35

UCI  
DREAM Lab



## Single Node Simulation of Cars



- Simulated states of cars at two consecutive simulation steps
- State of cars = F (speed change prob., lane change prob.)
- Update of car states = F (old states of itself and adjacent cars, other parts of the freeway)

Mar-07

36

UCI  
DREAM Lab



## Inconsistent States

- New state of simulator may be an **inconsistent** state such as the case when D "flew" over B.
- One solution:
  1. Sort cars from the front to rear.
  2. If the order is to update front cars first and rear cars later, then whenever a value for the new state of a car is calculated, a check is made whether it is in conflict with the already calculated new states of the cars in the front.
  3. If a conflict is detected, the value is discarded and an attempt is made to produce a new value until a conflict-free value is produced.

Mar-07

37

UCI  
DREAM Lab



## Update Dependency

- When two simulator objects cannot be updated independently, they are said to be **update-dependent** upon each other.
- **Parameters** of update dependency in freeway simulation
  - Ticking rate (or the length of the ticking interval)
  - Speeds of the cars
  - Distances between cars
- Can be **broken** by changing any of those parameters
  - For example, by increasing the ticking rate
- Update dependency is a **transitive relation**. Therefore, a chain of update dependency prevents DTS approach from exploiting the full potential of parallelism in the distributed, parallel execution of the simulation system.

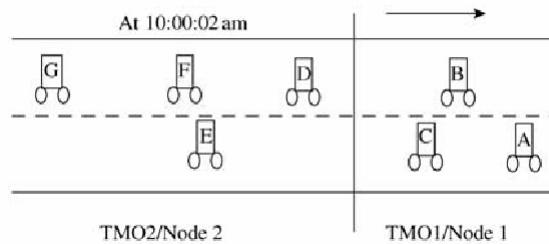
Mar-07

38

UCI  
DREAM Lab



## Distributed Simulation of Cars



- Partitioning of freeway segments into two TMO nodes running on separate nodes
- At the boundary between two freeway segments, take-over of car objects occurs from TMO2 to TMO1.
- Update dependency among cars does **not change** by this partitioning.

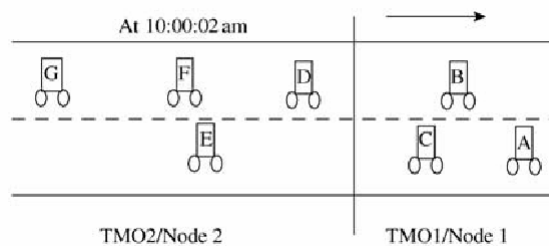
Mar-07

39

UCI  
DREAM Lab



## Update Dependency among Distributed Simulator Objects



- Both TMOs run in parallel and each TMO updates front cars first.
- Cars are to be updated to the states effective at 10:00:03.
- If car D is update-dependent on car B and car C, then TMO2 needs new state values, B(10:00:03) and C(10:00:03), calculated by TMO1 in order to establish D(10:00:03) => **no parallelism!**

Mar-07

40

UCI  
DREAM Lab



## Minimizing Update Dependency

---

- One approach is to let TMO2 forward at the end of each simulation step all the cars which are in its territory and update-dependent on the cars in the territory of TMO1 to TMO1.
- One major drawback of this approach is in the increased ticking interval which must cover the aforementioned exchanges between adjacent TMOs.
- Finding a more efficient approach is a challenging subject for future research.
- Therefore, the key factor that determines the efficiency of distributed / parallel RT simulation is the update-dependency.

Mar-07

41

UCI  
DREAM Lab



## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- An Overview of the TMO Scheme
- TMO-Structured DTS
- Major Issues in Efficient Implementation of DTS
- **An Execution Engine for TMO-Structured DTS**
- Conclusion

Mar-07

42

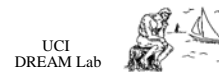
UCI  
DREAM Lab



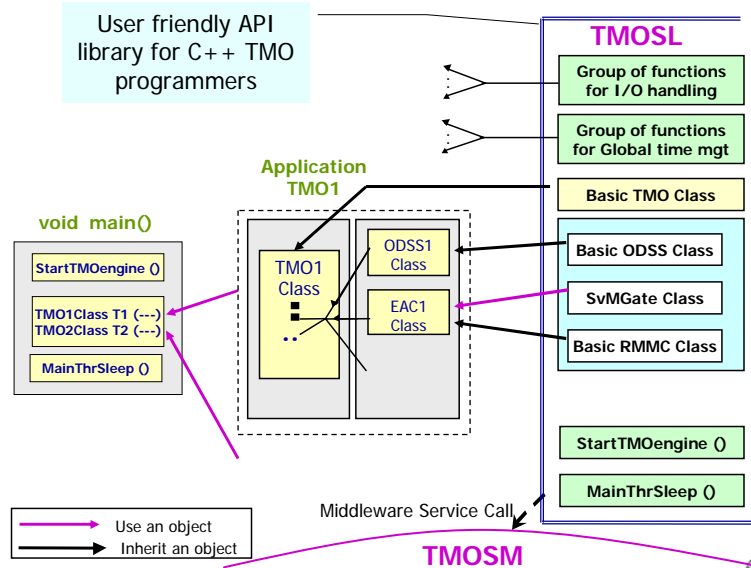
## TMO Support Middleware and Library

- TMO Support Middleware (TMOSM)
  - A cost-effective way to support execution of TMOs
  - A middleware subsystem running on commercial-off-the-shelf (COTS) platforms
  - The most obvious and important requirement that a TMO execution engine must meet is to accurately honor the timing specifications associated with various application program-segments.
- TMOSM Support Library (TMOSL)
  - User friendly API library for C++ TMO programmers

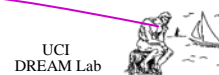
Mar-07 43



## TMOSM Support Library (TMOSL)



Mar-07 44



## TMOSM Architecture

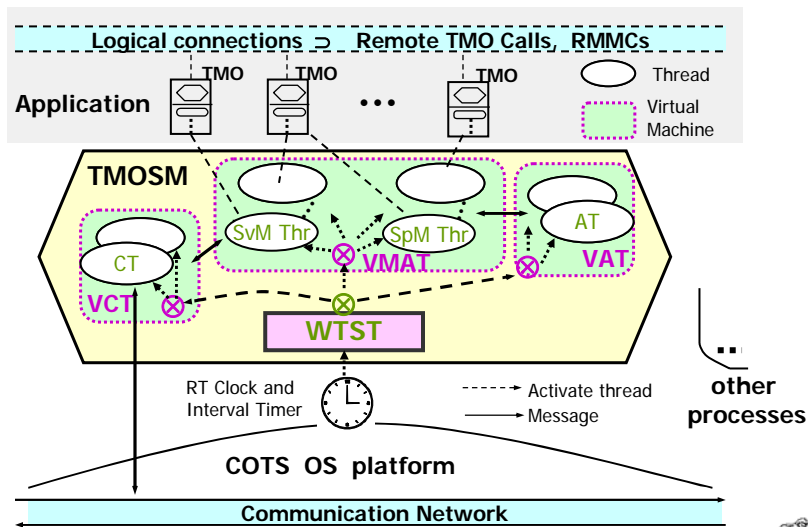
- The TMO Support Middleware (TMOSM) consists of a number of virtual machines (VMs) and threads that interact with and support the execution of application TMOs.
- Each VM is responsible for a major part of the functionality of TMOSM. Each VM maintains a number of application threads.
- The innermost core of TMOSM is a *super-micro* thread called the WTST (Watchdog Timer & Scheduler Thread). It is a "super-thread" in that it runs at the highest possible priority level. It is also a "micro-thread" in that it manages the scheduling / activation of all other threads in TMOSM.
- WTST leases processor and memory resources to three VMs in a time-sliced and periodic manner. Each VM can be viewed conceptually as being periodically activated to run for a time-slice.
- Whenever WTST assigns a time-slice to a VM, the VM in turn passes the time-slice onto one of the application threads that belong to it.

Mar-07 45

UCI  
DREAM Lab



## Internal Thread Structure of TMOSM



Mar-07 46

UCI  
DREAM Lab



## Virtual Machines

- VCT (VM for Communication Threads)
  - The application threads maintained by this VM are those dedicated to handling the sending and receiving of middleware messages.
  - Middleware messages are exchanged through the communication network among the middleware instantiations running on different DC nodes to support interaction among RC DC objects, i.e., TMOs.
- VMAT (VM for Main Application Threads)
  - The application threads maintained by this VM are those dedicated to executing methods of TMOs with maximal exploitation of concurrency.
  - Normally to each execution of a method of an application TMO is dedicated a main application thread.
  - In principle, TMO method executions may proceed concurrently whenever there are no data conflicts among the method executions.

Mar-07

47

UCI  
DREAM Lab



## Virtual Machines (cont.)

- VAT (VM for Auxiliary Threads)
  - This VM maintains a pool of threads which are called auxiliary threads.
  - Some auxiliary threads are designed to be devoted to controlling certain peripherals under orders from TMO methods (executed by main application threads).
  - Others wait for orders for executing certain application program-segments and such orders come from main application threads in execution of TMO methods.
  - Use of this VAT has been motivated partly by the consideration that it should be easier to analyze the temporal predictability of the application computations handled by each VM, i.e., those handled by VMAT and those by VAT, than to analyze the temporal predictability of the application computations when there is no VAT and thus VMAT alone handles the combined set of application computations.

Mar-07

48

UCI  
DREAM Lab



## Ease of Execution Time Analysis

- Several features of the TMOSM architecture contributes to simplifying the analysis of the execution time behavior of application TMOs running on TMOSM.
- The strictly periodic nature of virtual machines and the dedication of each virtual machine to a specific functionality enable largely independent analysis of the part of the execution time behavior of application TMOs that depends on a particular virtual machine.
- The execution time of an I/O can be in general specified, analyzed, and measured in a larger time gain than that which can be used in specifying and analyzing the computation relying on the CPU only.
- Dedicating TTFs and SvFs to handling such I/O activities enables the high-precision analysis of the execution time behavior of the CPU-intensive computation.

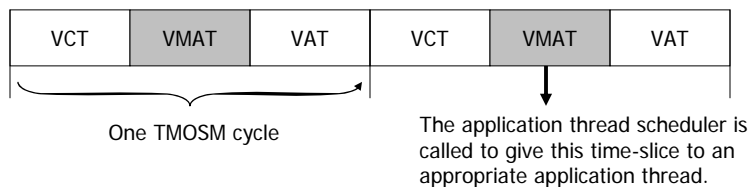
Mar-07

49

UCI  
DREAM Lab



## Two-level Scheduling



- TMOSM adopts a two-level scheduling approach: virtual machine scheduler and application thread scheduling.
- WTST first executes the virtual machine scheduler to select the next virtual machine, one among VCT, VMAT, and VAT, in RR fashion.
- Each virtual machine has its own thread scheduler by which all threads in the virtual machine can be scheduled according to the scheduling policy in each virtual machine (by default, EDF).
- This is one of the architectural aspects of TMOSM which are useful in realizing the highly predictable behavior of the middleware.

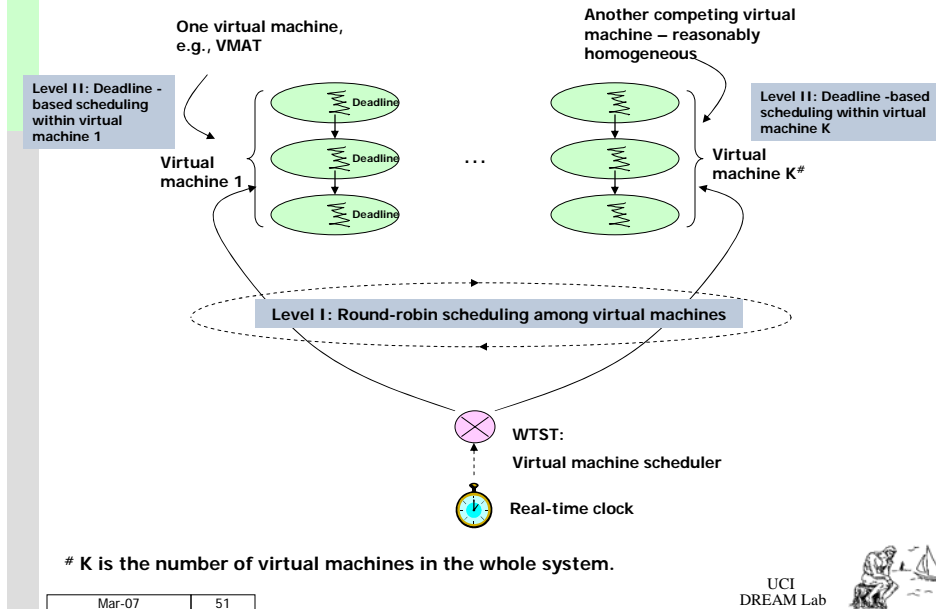
Mar-07

50

UCI  
DREAM Lab



## Two-level Group Scheduling



## Ticking Rate & Simulator Execution Engine

- The precision of the RT simulator is proportional to the ticking rate.
- For efficient distributed simulation, minimizing the update-dependency is necessary.
- This update-dependency can also be broken by increasing the ticking rate.
- Yet the ticking interval must be long enough to accommodate necessary message communications.
- Therefore, it is desirable to use **computing platforms yielding small message delays**.
  - One example can be use of TDMA scheme for media access.
  - TMOSM currently enforces TDMA access by the nodes to an Ethernet bus.
  - Use of highly parallel computing platforms which supports multiple channels that can simultaneously carry messages is an attractive approach.

Mar-07 52

## Outline

---

- Introduction
- Main Challenges in Distributed RT Simulation and the DTS Framework
- An Overview of the TMO Scheme
- TMO-Structured DTS
- Major Issues in Efficient Implementation of DTS
- An Execution Engine for TMO-Structured DTS
- Conclusion

Mar-07

53

UCI  
DREAM Lab



## Conclusion

---

- DTS scheme is a **fundamentally new** type of an approach for distributed RT simulation.
- DTS scheme **obviates the need for massive message communication** among simulator nodes for synchronization purposes.
- DTS scheme requires a **global time base** and advanced computing platforms to exploit the full potential of this scheme.
- **TMO-structured DTS scheme** provides **several attractive features** including uniform structuring of DTS, highly predictable timing performance, efficient distributed and parallel processing and so on.
- Based on the experiences of CAMIN and DOFS development, TMO-structured DTS scheme can be a **promising solution for distributed / parallel RT simulation**.
- The key factor that determines the efficiency of distributed / parallel RT simulation is the **update-dependency**.

Mar-07

54

UCI  
DREAM Lab

