The TMO Scheme for Wide-Area Distributed Real-Time Computing and Distributed Time-Triggered Simulation

Kane Kim and Stephen F. Jenks

University of California, Irvine

{khkim, sjenks}@uci.edu

For presentation at
NGS 2007
March 2007

UCI
DREAM Lab

Outline

- Introduction
- Time-triggered, Message-triggered Object (TMO) Scheme
- Distance-aware TMO (DA-TMO) for use in WAN Environments
- TMO-based Distributed Real-time Computing (DRC) Applications
- TMO-structured Distributed Time-triggered Simulation (DTS)
- Conclusion

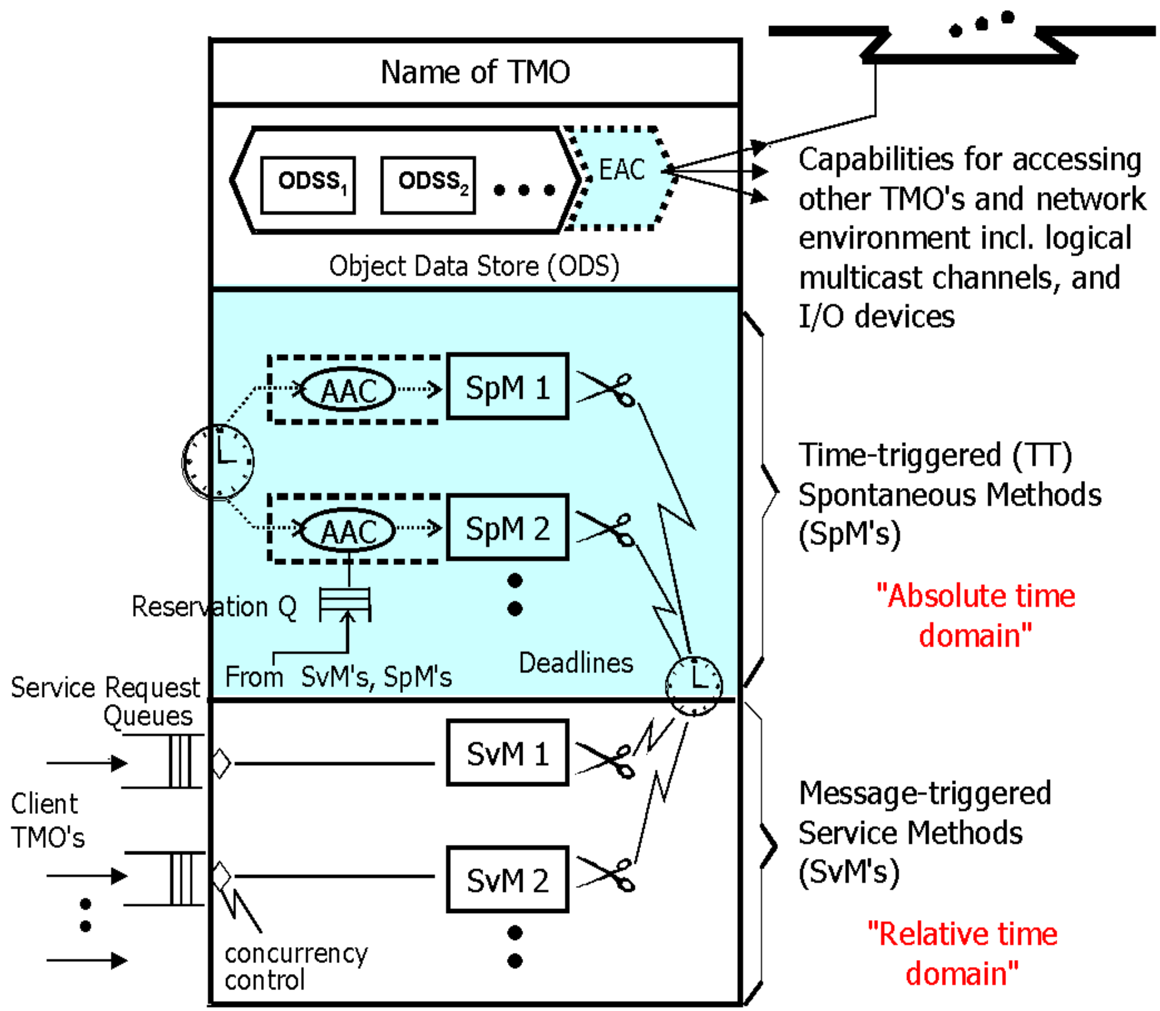
Local-area DRC => Wide-area DRC

- While local area distributed real-time computing (DRC) is a steadily advancing technology field with many immature aspects in its core at this time, wide-area DRC is in its infancy.
- Our efforts to extend the DRC technology established for use in local area network (LAN) environments to fit into the WAN environments:
  - The basic building-block of our technology framework is the *Time-triggered Message-triggered Object* (TMO) specification and programming scheme.
  - The TMO scheme includes establishment and use of a *global time base* which provides consistent real-time information available in all distributed computing nodes.
  - The TMO scheme for local-area DRC has been established in a sound form and its practicality and attractiveness have been extensively demonstrated. However, its extension to fit into wide-area-network based DRC is in an early stage.
- In this paper, we present a brief review of the progresses made recently in extending the TMO scheme for use in WAN environments.

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 distributable HRT objects and distributable 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



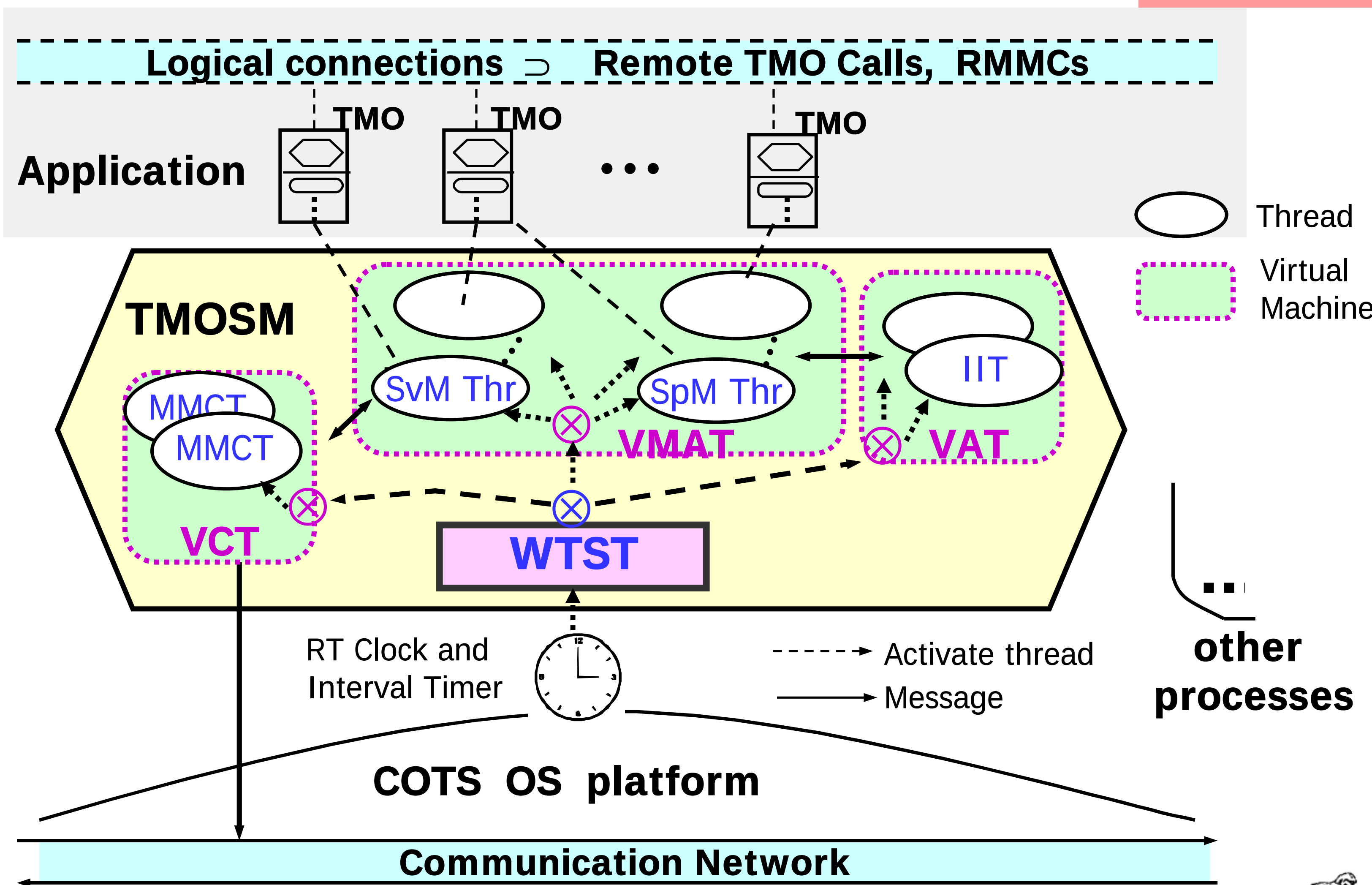
UCI DREAM Lab

Middleware and APIs

- TMO Support Middleware (TMOSM)
  - A middleware architecture providing execution support mechanisms and being easily adapted to a variety of commercial kernel+hardware platforms
  - Uses well-established services of commercial OSs, e.g., process and thread support services, short-term scheduling services, and low-level communication protocols, in a manner transparent to the application programmer
  - *Non-Blocking Buffer (NBB)* to avoid blocking of threads due to semaphores or locks
  - *Kernel Abstraction Layer (KAL)* to improve portability
- TMO Support Library (TMOSL)
  - User-friendly programming interfaces wrapping the execution support services of TMOSM
  - Consists of a number of C++ classes and approximates a programming language directly supporting TMO as a basic building-block
- Visual Studio for TMO (ViSTMO)
  - A GUI (graphic user interface) approach to designing an initial skeleton of each TMO and letting a tool generate a code-framework for each TMO

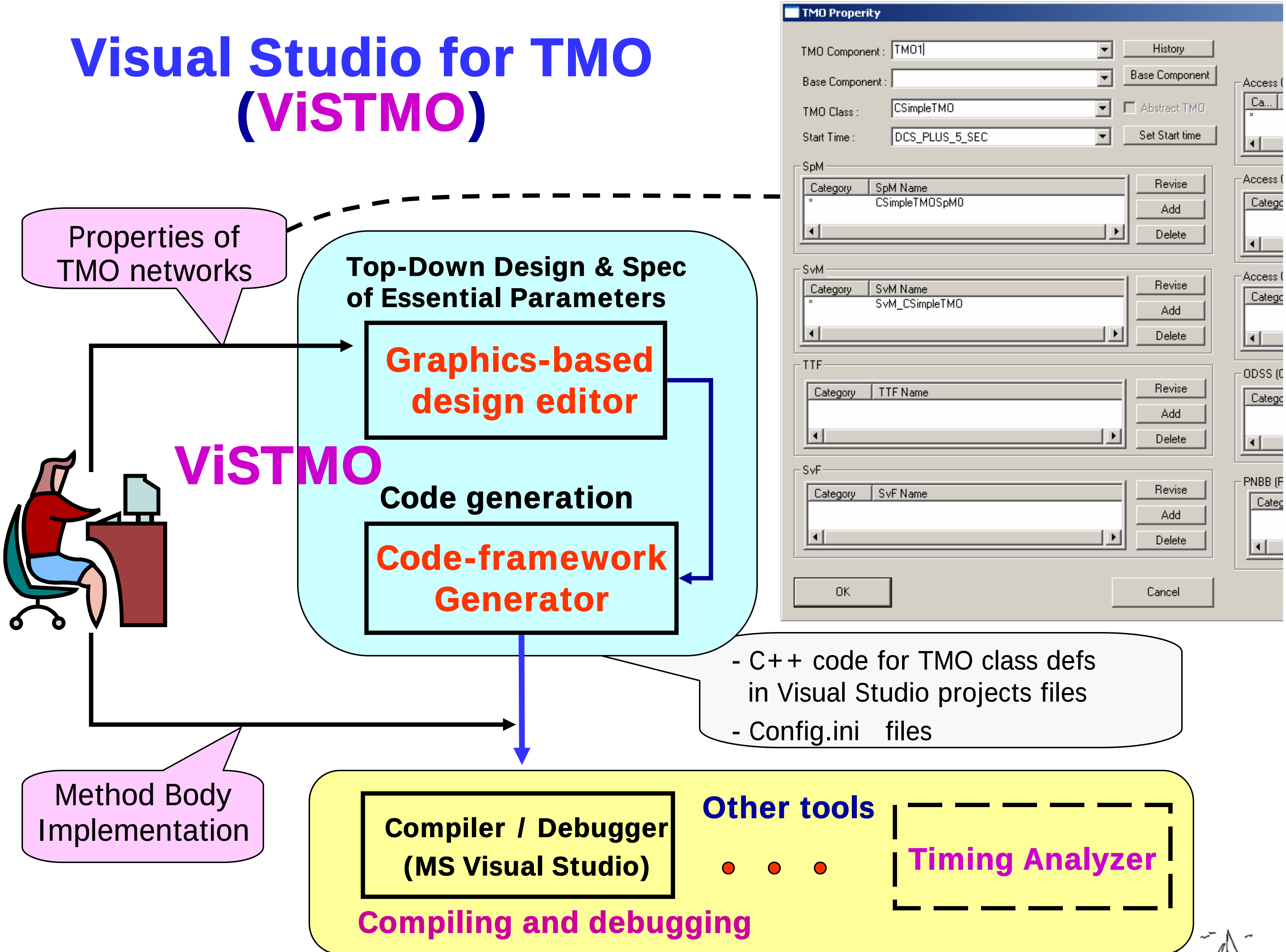
TMO Support Middleware (TMOSM) on Windows XP & CE -- TMOSM / XP, CE, or Linux / Socket

Currently running



UCI
DREAM Lab

Visual Studio for TMO (ViSTMO)



Distance-aware TMO (DA-TMO)

- The large communication latency inherent in a DRC prevents the current TMOSM instantiations from cooperating and interacting frequently among themselves.
- A newly extended TMO model called the *distance-aware TMO* (*DA-TMO*) is introduced in order to establish an effective building-block for wide-area DRC systems.
  - DA-TMO programmers should expect that TMOSM instantiations supporting nearby TMOs will interact with a relatively high frequency whereas TMOSM instantiations supporting TMOs separated by long distances will interact less frequently.
  - They should also expect that a call by a client TMO for a service offered by a remote TMO can involve searches for information not readily available in the local TMOSM instantiation.
- Efforts to extend *TMO Network Configuration Manager (TNCM)* and other parts of TMOSM to support DA-TMO are underway.

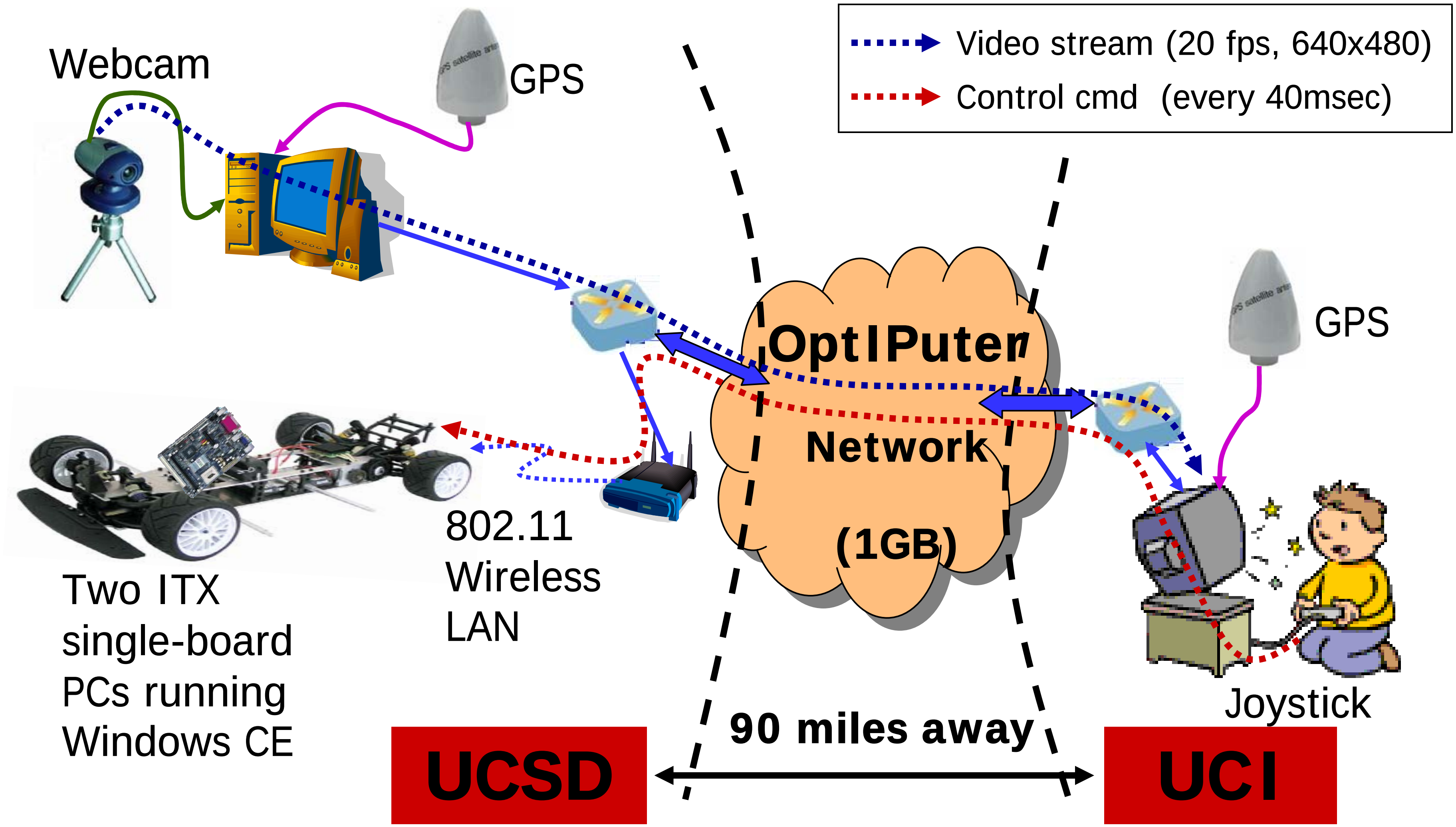
Distance-aware TMO (DA-TMO) (cont)

- The clock synchronization module of TMOSM has been enhanced to take advantage GPS facilities which serve as a source of global time of micro-second precision.
- Middleware support components for dynamic creation and destruction of TMOs have been incorporated into TMOSM.
- Member sites of a WAN are often machines of PC cluster types. We have thus been developing a version of TMOSM for such a cluster.

High-quality Multimedia Streaming Service

- An approach for realizing high-quality tele-audio services over networks by applying the global time based coordination of distributed actions (TCoDA) principle was realized.
- A TMO-based audio streaming application over heterogeneous platforms, e.g., Windows XP, Windows CE.NET, and Linux 2.6, was constructed. In LAN-based experiments, the maximum intra-stream jitter was merely 17ms.
- Further experiments involving both LANs and WANs are under way.
- A video streaming service of a similar kind was studied, too, with highly promising results and demonstrations.

Wide-Area DRC Testbed: TMO Turtle



- Application-to-application msg transmission delay <= 60 msec!!!

UCI
DREAM Lab

Basic Requirements in RT Simulation

- Real-Time Simulation := Accurate mode of simulation in which the simulator components show the timing behavior that are the same as or similar to the timing behavior of the simulation targets.
- Every computer-based simulation execution engine 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:

    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.

Distributed RT Simulation

- As the complexities of RT simulators grow, the use of distributed and parallel RT simulation approaches become imperative.
- In distributed real-time simulation, *simulator objects (or processes)* are distributed among multiple nodes.
- Synchronization of the simulation-steps of distributed simulator objects is then a key challenge.
  - A simulation-step executed by the distributed nodes as a group must include the activities necessary to keep the executions of the simulation-step by the nodes synchronized.
  - The simulator clock for one simulator object must commence the n-th tick neither before the (n-1) - th tick by the clock driving another simulator object nor after the (n+1) - th tick by the latter clock.

Distributed Time-triggered Simulation (DTS)

- Essence of the DTS approach
  1) Every node is equipped with an RT clock and executes each simulation-step upon reaching of the RT clock at the predetermined value.
  2) 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 (not counting the message exchanges which may be needed at a certain low frequency for re-synchronizing the real-time clocks of the nodes) .
  – DTS approach enables easy design of simulator objects which use different ticking rates.

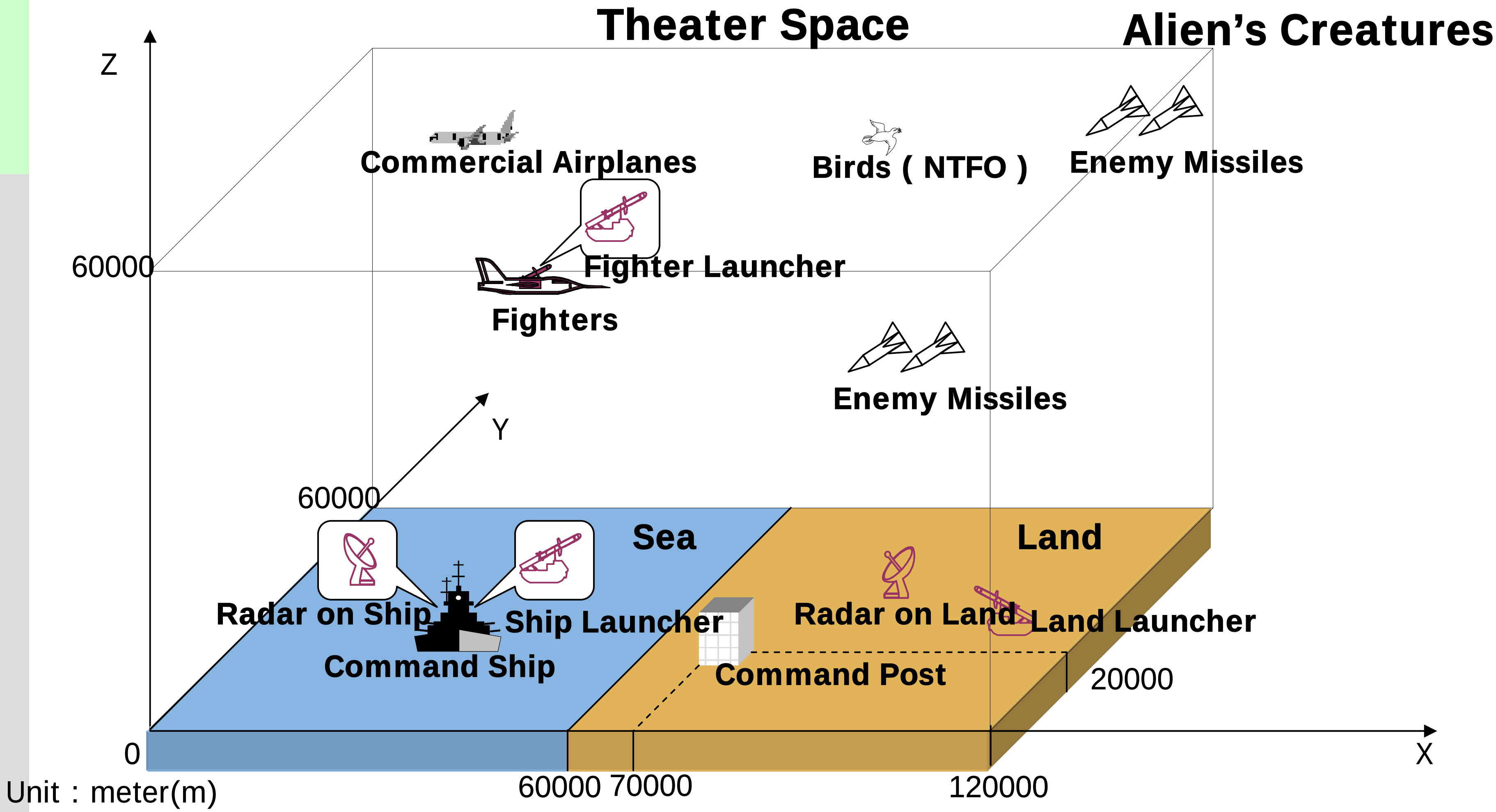
TMO-structured DTS

- 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).

Update Dependency

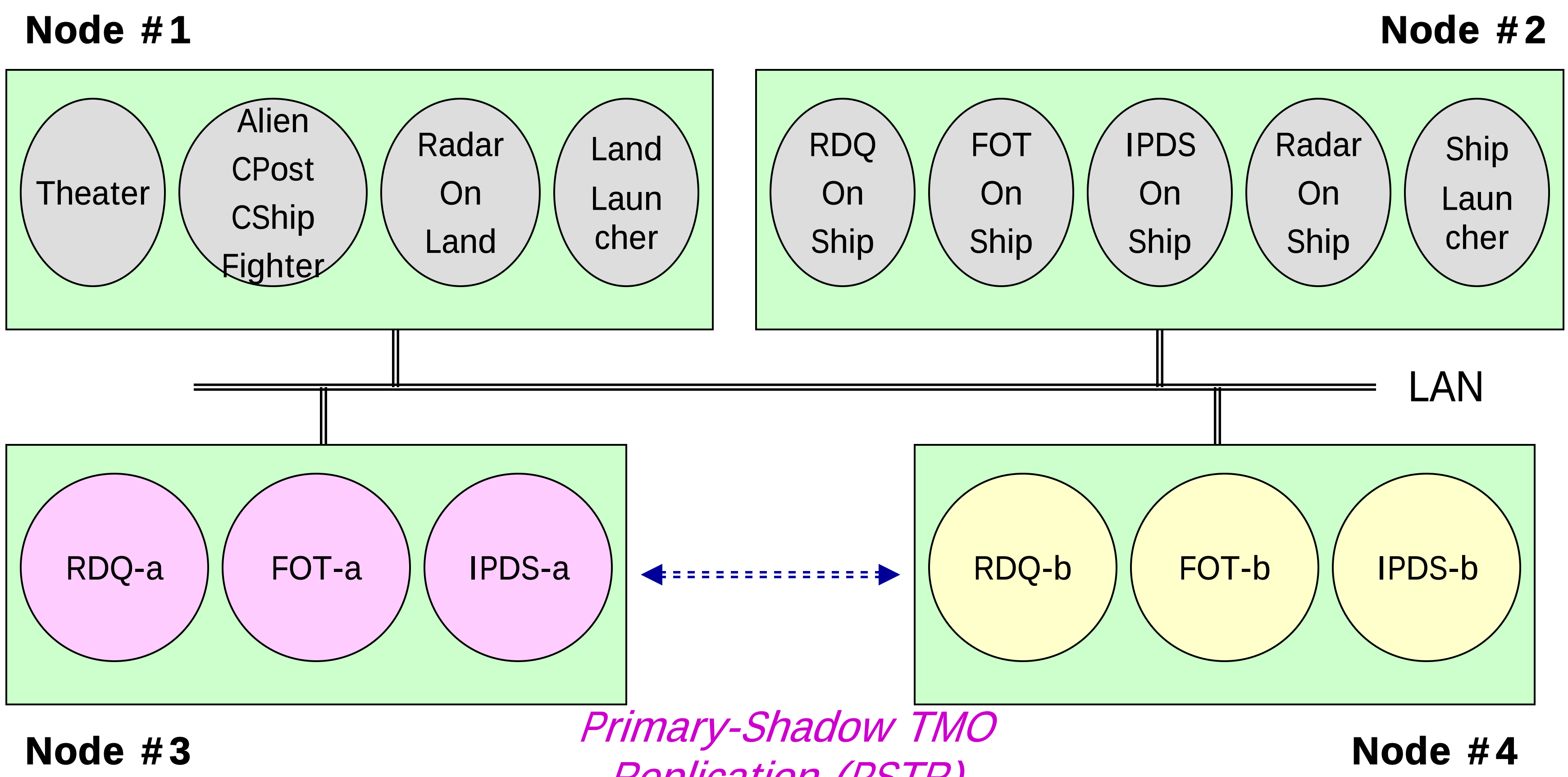
- A fundamental obstacle in parallel / distributed execution of real-time simulation actions is the *update-dependency*.
- When a simulation target item covered by one simulator node is update-dependent on another simulation target item covered by another simulator node, update activities of the two nodes must be serialized.
- The 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.
- Several basic approaches dealing with the techniques for minimizing the impacts of the update-dependency among distributed simulator objects were formulated and experimental research is under way.

TMO-structured DTS Testbed:
Coordinated Anti-Missile Interceptor Network (CAMIN)



Goal: Defend the target (Command Ship) from enemy missiles!!!

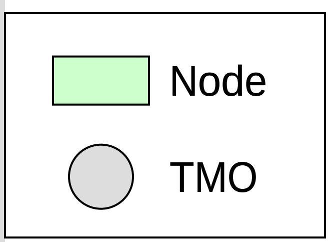
CAMIN with Fault-tolerance Support



Primary-Shadow TMO Replication (PSTR)

with

Supervisor-based Network Surveillance (SNS)



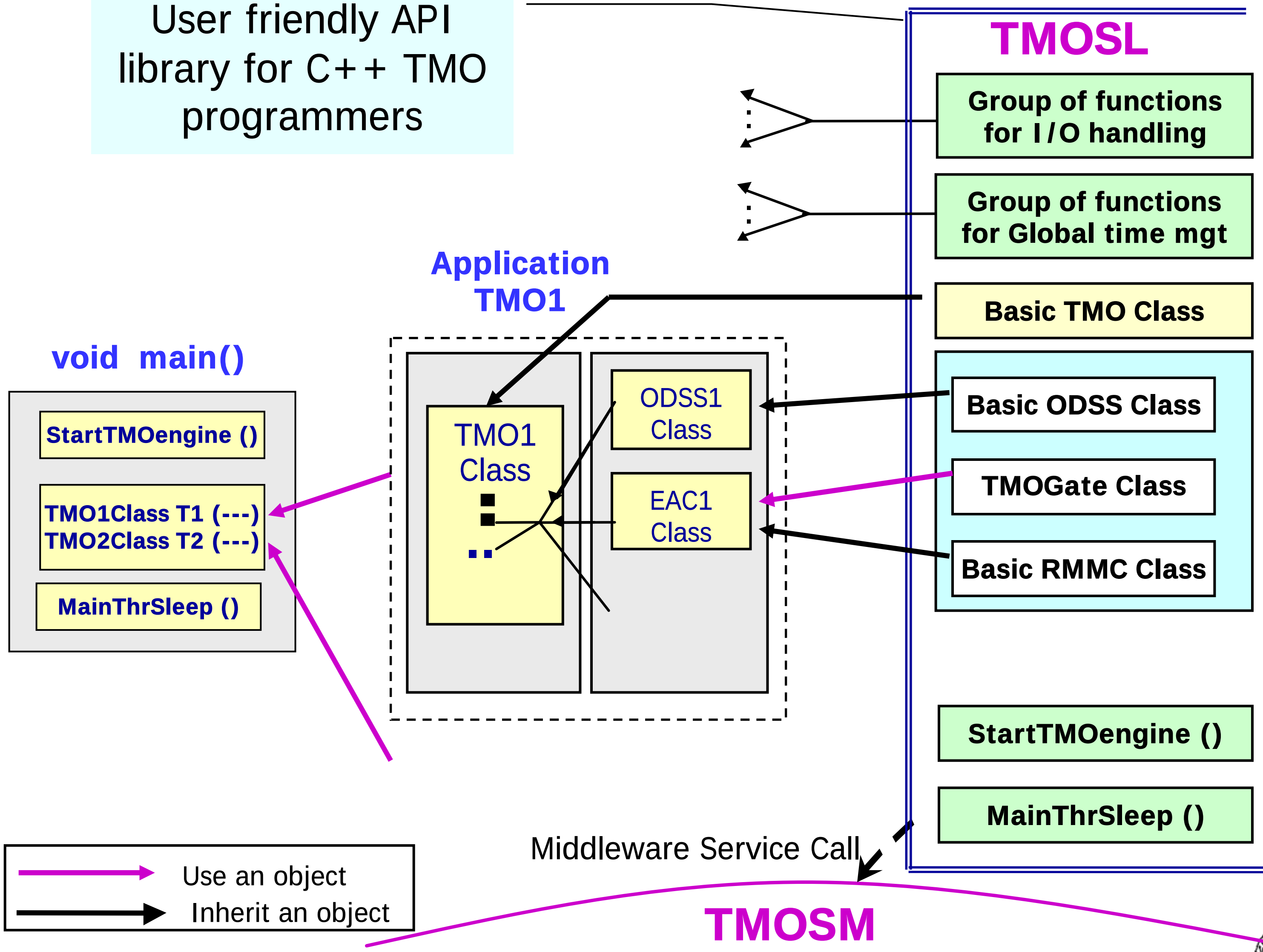
Conclusion

- The TMO scheme for wide area DRC is promising, especially with the advent of a new-generation network infrastructure such as OptIPuter. Nevertheless, this field is in an early stage.
- The TMO-structured DTS has been demonstrated in reasonably convincing forms but its optimal use requires much further research.

backup

TMOSM Support Library (TMOSL)

User friendly API library for C++ TMO programmers



UCI DREAM Lab

TMO Structure and Design Paradigms

(TM1) All time references in a TMO are references to global time.

(TM2) TMO is a distributed computing (DC) component.

(TM3) TMO has been devised to contain only high-level intuitive and yet precise expressions of timing requirements.

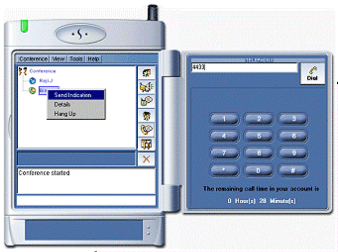
(TM4) TMO is also an autonomous active DC component.

(TM5) A logical multicast channel facility, called *Real-time Multicast and Memory-replication Channel* (RMMC), is used for message communication among TMOs in addition to the regular RPC style service request calls.

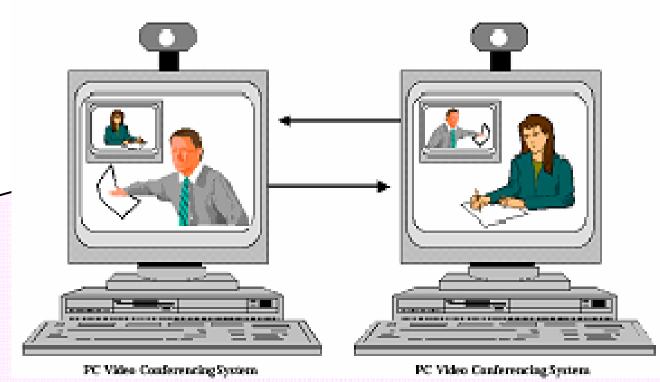
(TM6) The *basic concurrency constraint* (BCC) incorporated along with the time-triggered Spontaneous Methods (SpMs) eases design-time guaranteeing of timely services of TMOs by having SpM executions not disturbed by SvM (Service Method) executions.

(TM7) An RT computer system will always take the form of a network of TMOs, which may be produced in a top-down multi-step fashion, called the *TMO Network Development Methodology* (TMONDeM).

High-quality Multimedia Streaming Service



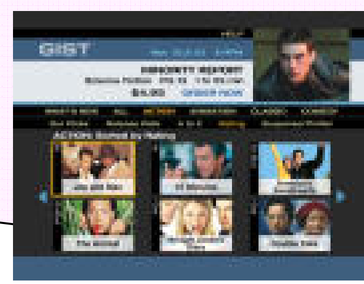
Telephony (Voice-over-IP)



Videoconferencing

Distributed orchestra

Remote Surgery



Interactive TV

Distance Learning

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 single TMO into a TMO network
- Easy programming and debugging of timing characteristics and concurrency control
- Efficient distributed and parallel processing in heavy-load simulations thanks to lack of massive message exchange for synchronization purposes
- Unified development environment of both simulation targets and simulator itself