

# CORBA Services Supporting High-Level Real-Time Objects

K.H. (Kane) Kim  
UCI DREAM Lab  
(Distributed Real-time Ever Available Microcomputing Lab)  
khkim@uci.edu, <http://dream.eng.uci.edu/>

For presentation at OMG's WS-1 on  
Real-time and Embedded Distributed Object Computing, July 2000

## Outline

- Main messages
- High-level RT object: **TMO (Time-triggered Message-triggered Object)**
- **TMOSM (TMO Support Middleware)**
  - Middleware architecture supporting TMO execution
- **TMOES (TMO Execution Support)**
  - Adaptation of TMOSM in the form of a CORBA Service
  - **TNCM (TMO Network Configuration Manager)**
  - Service request paths and types

Jul-00

UCI  
DREAM Lab



## Main Messages

- Time to push for **high-level real-time (RT) object** programming
- A new-generation RT programming style called the **TMO (time-triggered message-triggered object)** programming style [1,2]
  - Passed the feasibility-proof stage.
- Middleware support for **CORBA-compliant TMOs**: Possible as a CORBA service, **TMOES (TMO execution service)**.
  - In the absence of an **RT ORB**, TMOES may use a CORBA service called the **Cooperating Network Configuration Management (CNCM)**: It maintains a network environment in which at least the message traffic is regulated to effect a reasonable communication delay bound.
- An interesting near-term issue:  
Cost-effective to **migrate** some parts of the two CORBA services **down** to the ORB and IDL areas ?

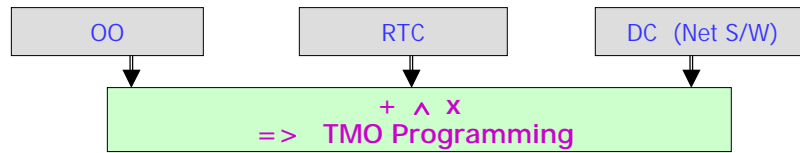
- [1] Kim, K.H.(Kane), "[Object Structures for Real-Time Systems and Simulators](#)", *IEEE Computer*, August 1997, pp.62-70.
- [2] Kim, K.H., "[APIs for Real-Time Distributed Object Programming](#)", *IEEE Computer*, June 2000 (special issue on OO RT distributed Computing), pp.72-80.

Jul-00

UCI  
DREAM Lab



## High-Level High-Precision RT OO Programming



Main goal:

Quantum jump (e.g., > 200% improvement) in *software development efficiency X product reliability* of RT distributed computing software

Via:

Bringing up the conventional C/assembly-level RT programming to an *abstract-language-level RT programming*

Jul-00

UCI  
DREAM Lab



## High-Level High-Precision RT OO Programming (cont)

- The new-generation RT distributed software engineering method must
  - Be based on a "general high-level programming style" which can be accommodated with minimal efforts by current-generation business application programmers (using C++ and Java) \*
  - Require specification of both
    - the interactions among distributed program components and
    - the timing requirements of various actionsin natural intuitively appealing forms only.

E.g., Not force designers to deal directly with non-intuitive notions such as priorities\*\* for the sake of meeting application timing requirements

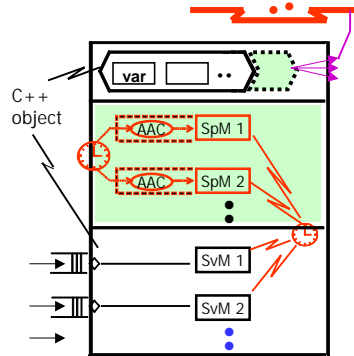
Jul-00

UCI  
DREAM Lab



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

- Established in early 1990's with a concrete syntactic structure and execution semantics for economical reliable design and implementation of RT systems.
- 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.
  - A natural & syntactically small but semantically powerful extension of the conventional object structure



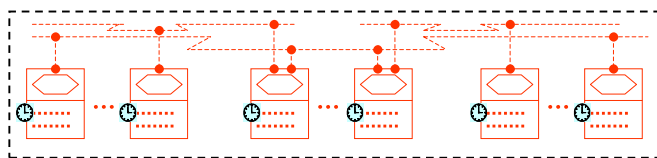
UCI  
DREAM Lab



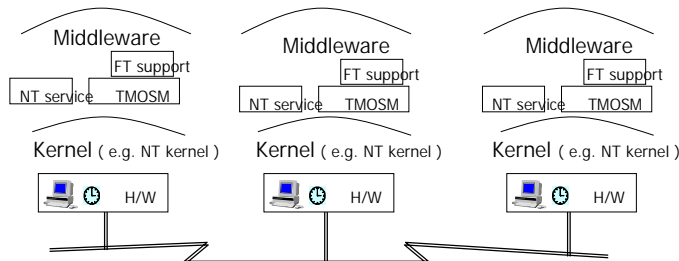
Jul-00

# 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 requirements in indirect terms (e.g., priorities)
- Only start-windows and completion deadlines for object methods and
  - time-windows for output actions

UCI  
DREAM Lab



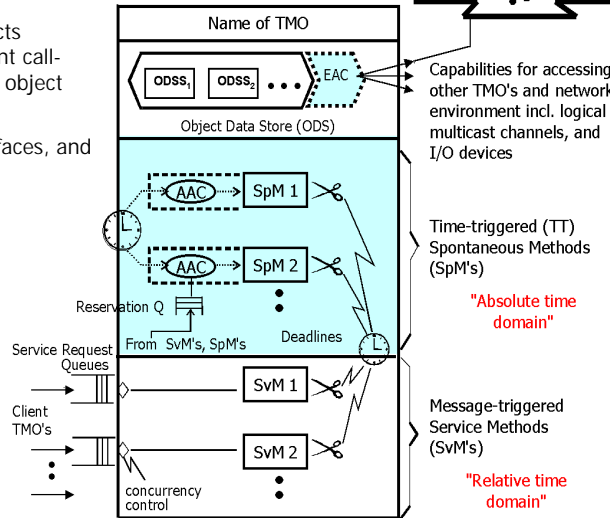
Jul-00

# TMO Programming Scheme

- Distributed computing component:

EAC (Environment Access Capability) section provides

- list of *gate* objects providing efficient call-paths to remote object methods,
- I/O device interfaces, and etc;

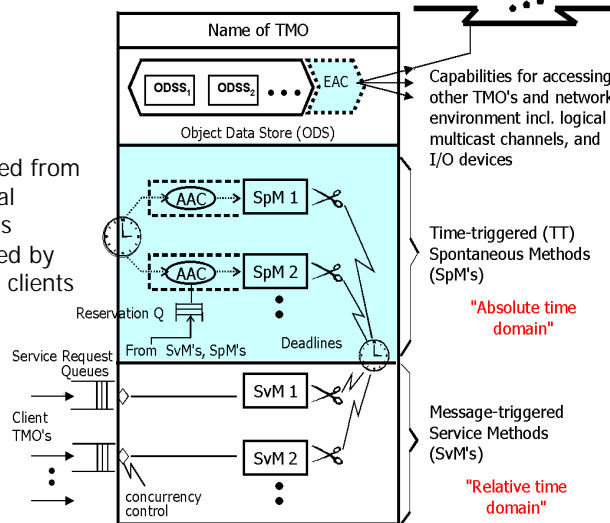


Jul-00

# TMO Programming Scheme (cont)

- Time-triggered (TT-) or spontaneous methods (SpM's):

- Clearly separated from the conventional service methods (SvM's) triggered by messages from clients



Jul-00

UCI  
DREAM Lab

## TMO Structuring Scheme (cont)

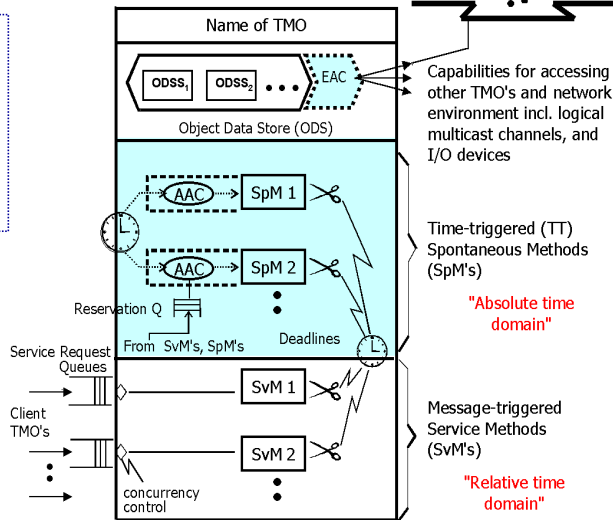
- Time-triggered (TT-) or spontaneous methods (SpM's):  
Clearly separated from the service methods (SvM's) triggered by messages from clients

L0

### Example of AAC:

```
{ "start-during (10am, 10:05am)
  finish-by 10:10am",
  "start-during (10:30am, 10:35am)
  finish-by 10:40am" }
```

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



Jul-00

## Spontaneous methods (SpMs)

- Triggering times for SpM's
  - must be fully specified as **constants at design-time**
  - appear in the first clause of the SpM specification called the **autonomous activation condition (AAC)**

```
ab "AAC-begin"
  { [AAC name:]
    "for t = from 10am to 10:50am every 30min
      start-during (t, t+5 min) finish-by t+10min"
  }*
ae "AAC-end "
```

==::

```
{ "start-during (10am, 10:05am) finish-by 10:10am",
  "start-during (10:30am, 10:35am) finish-by 10:40am" }
```

Jul-00

UCI  
DREAM Lab

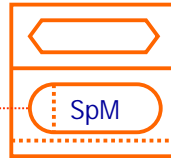


## TMO Programming Scheme (cont)

- Time-triggered (TT-) or spontaneous methods (SpM's):  
Clearly separated from the service methods (SvM's) triggered by messages from clients

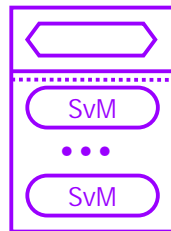
- Effect of a **thread**

No EAC, No SvM, and 1 SpM  
( **Activate once at a specified time;**  
**No guaranteed completion time** )



- Conventional passive object

No EAC, No SpM, and  
**No guaranteed completion time**



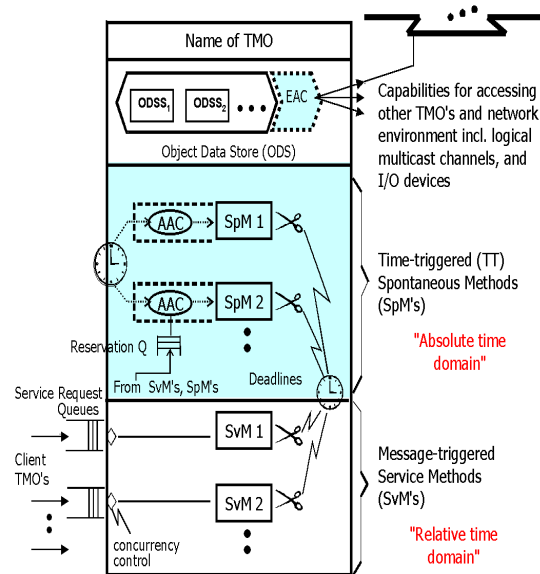
Jul-00

UCI  
DREAM Lab



## TMO Programming Scheme (cont)

- Time-triggered (TT-) or spontaneous methods (SpM's):
- Time-window** imposed on each **output action** and **method completion**
  - Guaranteed service time
  - Client's deadline demand
  - Statistical assurances of better service times

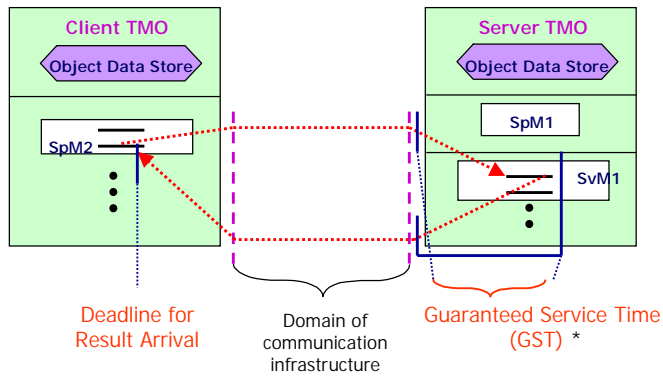


Jul-00

UCI  
DREAM Lab



## Return Deadline vs. Guaranteed Service Time



\* A violation of this deadline is a fault.

\* If GST is violated, *Fault handling actions must take place.*

- \* **Deadline for result arrival** - Call initiation time
  - > Max trans times imposed on comm infrastruc + GST
  - > Time consumed by communication infrastructure + GST

Jul-00

UCI  
DREAM Lab



## Statistical Assurances of Better Service Times

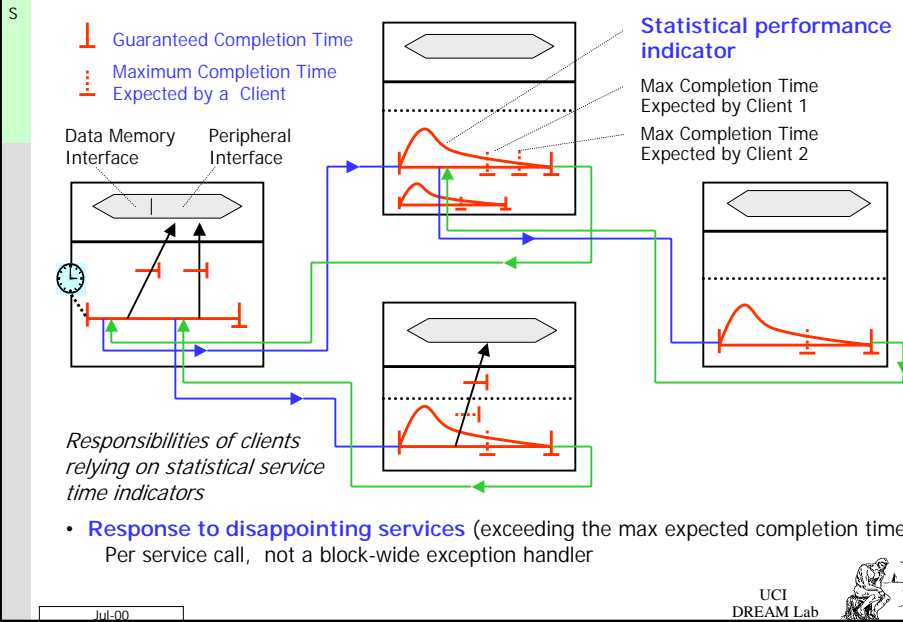
- **Reality:** Often, GSTs may be of rather poor qualities but their typical service times are far better than the client's requirements.
- A safe extension: To **augment** each server component with a *statistical performance indicator*.
- The designer of a client component may use candidate server components whose statistical performance indicators are good even if GSTs of the candidate components are not acceptable.
- Client component must contain an **alternate logic** to be invoked to accomplish the application objective in time, in case a DRA (deadline for result arrival) violation results from a call to the server component.

Jul-00

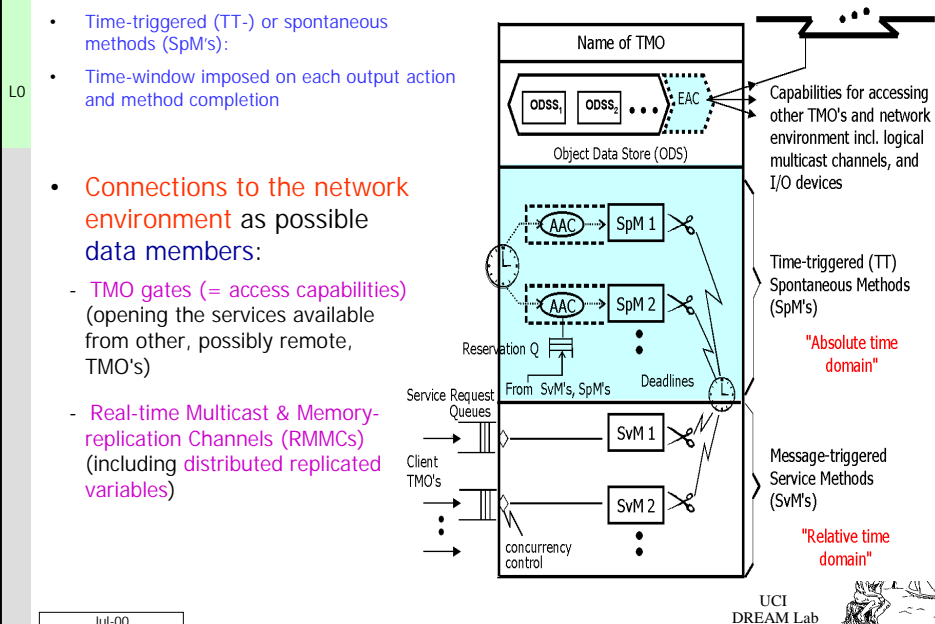
UCI  
DREAM Lab



## Statistical Assurances of Better Service Times (cont)



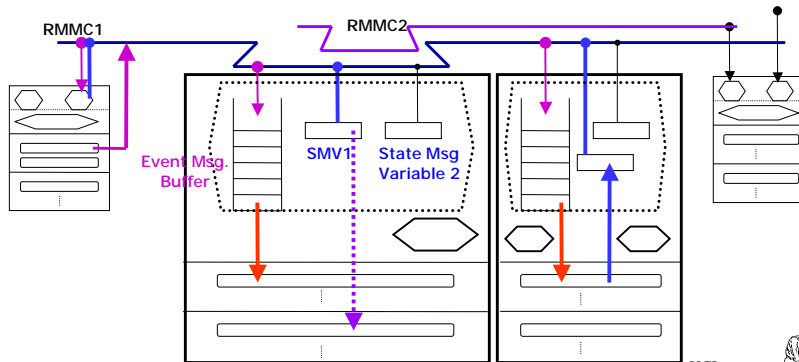
## TMO Programming Scheme (cont)



## Real-time Multicast & Memory-replication Channel (RMMC)

S

- Logical multicast channel facilitating (many-to-many) message communication among the distributed TMO's  
= Logical bus + distributed replicated variables
- No object naming or addressing: Only channel ID is used.
- Event messages
  - ➔ Announce (RMMC1)
  - ➔ Receive (RMMC1)
- State messages
  - ➔ G-update (RMMC1, SMV1)
  - ⋯➔ Read (RMMC1, SMV1)



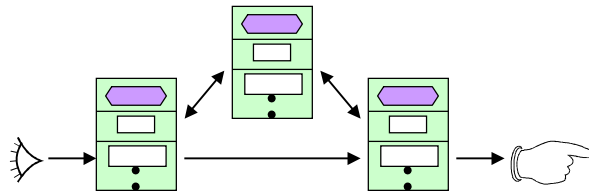
Jul-00

UCI  
DREAM Lab



## Interaction among TMO's

L1



- Mode 1: via Remote SvM calls
  - Blocking calls with return deadlines imposed
  - Non-blocking calls and subsequent result checks with deadlines imposed
  - Client-transfer calls
- Mode 2: via General message multicasts over RMMCs  
(Real-time Multicast and Memory-replication Channels)
  - Event messages
  - State messages

Jul-00

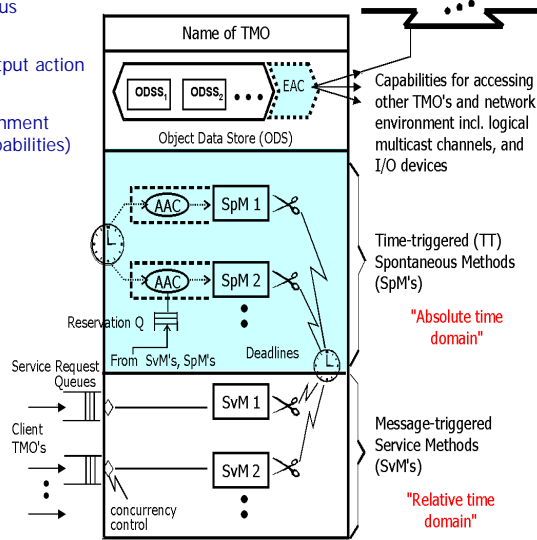
UCI  
DREAM Lab



## TMO Programming Scheme (cont)

- Time-triggered (TT-) or spontaneous methods (SpM's)
- Time-window imposed on each output action and method completion
- Connections to the network environment (e.g., RMMC's and TMO access capabilities) as possible data members

- **Basic concurrency constraint (BCC):**
  - SpM executions not disturbed by SvM executions.
  - Eases design-time guaranteeing of timely services of TMO's



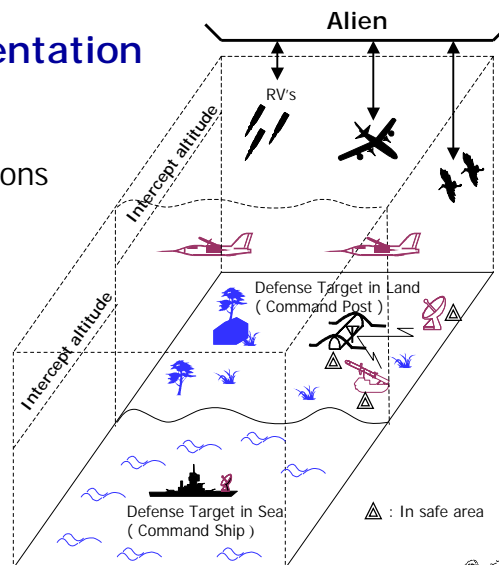
Jul-00

UCI  
DREAM Lab

## TMO-Network Structured Top-Down Design & Implementation

- Applicable to all conceivable applications

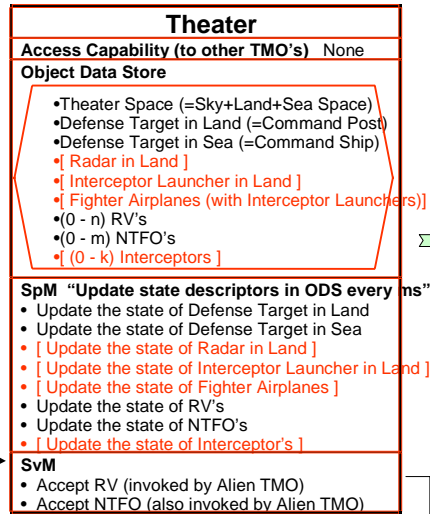
**CAMIN**  
(Coordinated Anti-Missile Interceptor Network) Theater



Jul-00

UCI  
DREAM Lab

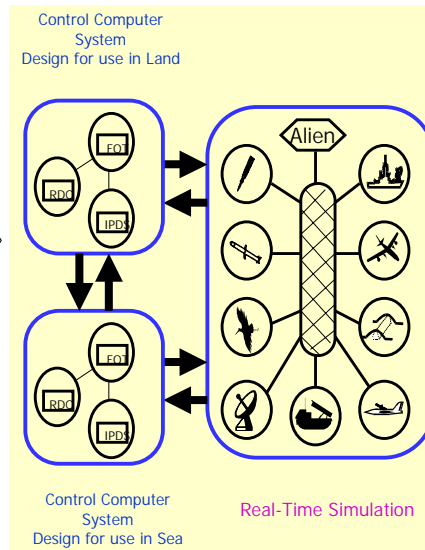
Starting with a single TMO specifying the application env't, sensors, actuators, & control strategy adopted



Alien

Jul-00

Multi-step refinement produces an executable TMO network



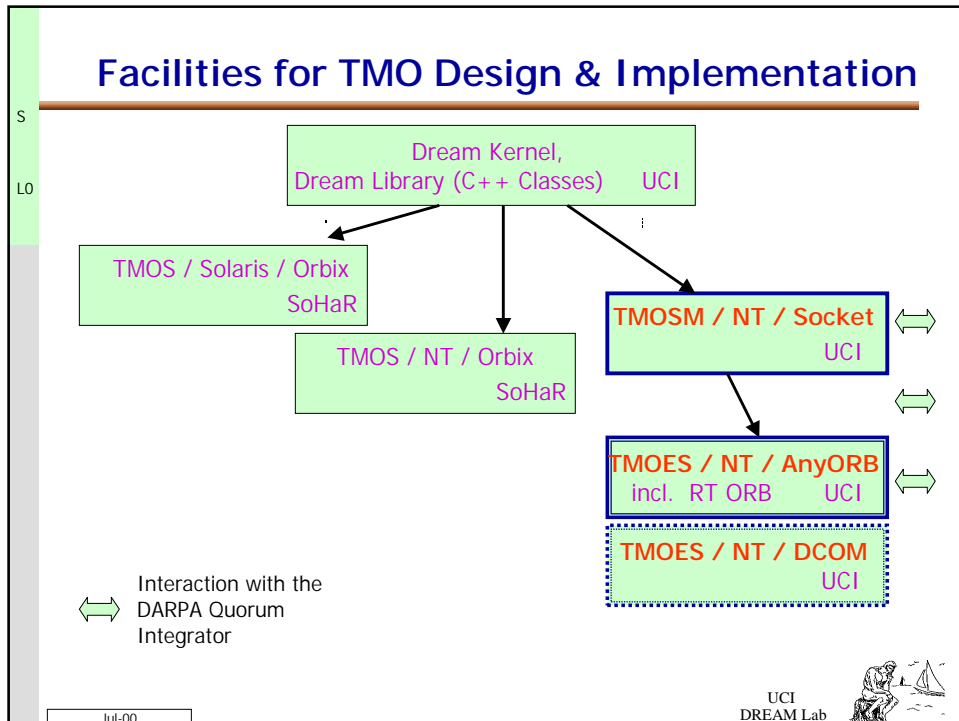
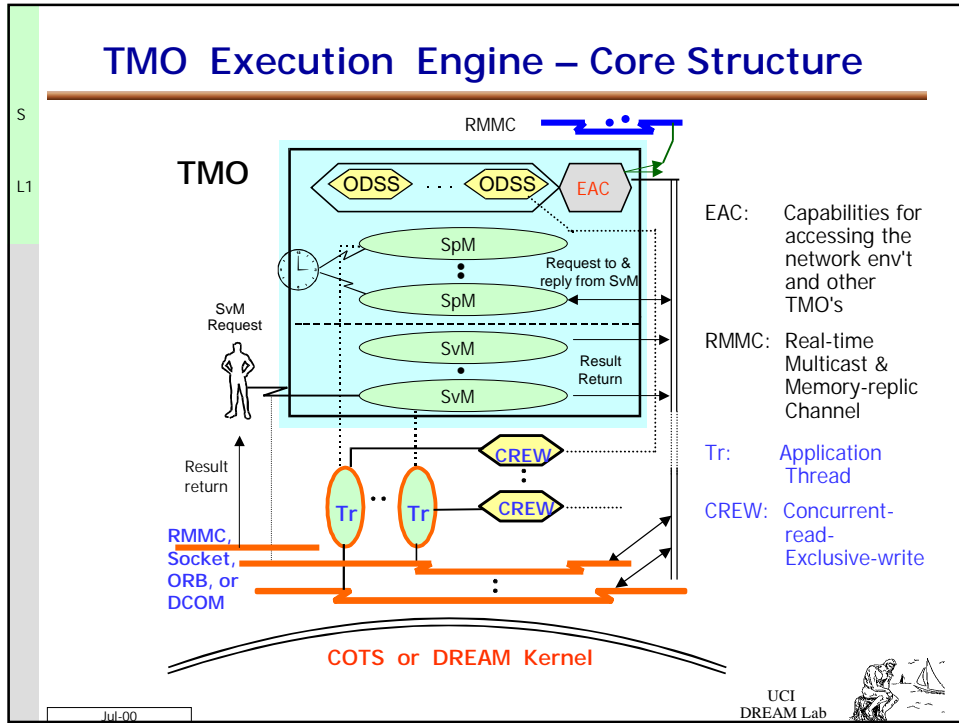
UCI  
DREAM Lab

## Benefits of the TMO Network Structuring Scheme

- **Uniform structuring** of
  - RT computation and non-RT computation,
  - a control computer system and a RT simulator of its application environment, and
  - all the way from the requirement specification, through the multi-step design, to the final implementation
- => Major improvements in RT Distributed SE
- => Major improvements in the reliability of RT Dist. Software
- Easy **timeliness-guaranteed design** and easy **incorporation of time-bounded fault tolerance mechanisms**
- Testing based on an **RT simulator of the environment** should yield **high testing coverage**
  - => Can considerably shorten the period for testing control computer systems in actual application environments.

Jul-00

UCI  
DREAM Lab



## TMOSM (TMO Support Middleware)

- A middleware architecture supporting TMO execution
- Supports **distributed, real-time programming** on COTS platforms
  - Allows programmers to express action timings *flexibly* and *well-structured forms* (at the level of 10 milliseconds with an implementation based on Windows NT)
- User-friendly C++ API, **TMOSL (TMO support library)**
- High **portability** and expandability
  - Can be ported to most modern OS with small effort
- Communication based on UDP.
  - **Can be based on IP multicast, CORBA, or DCOM.**
- Two application demos, **CAMIN** and **DOFS**, are available at <http://dream.eng.uci.edu>

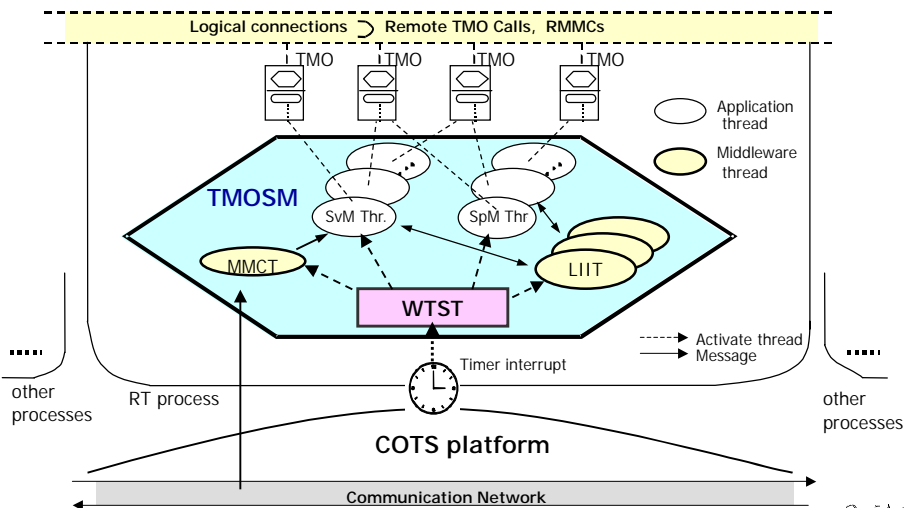
Jul-00

UCI  
DREAM Lab



## TMOSM / NT / Socket

\*: An Execution Engine Supporting TMO's



Jul-00

UCI  
DREAM Lab

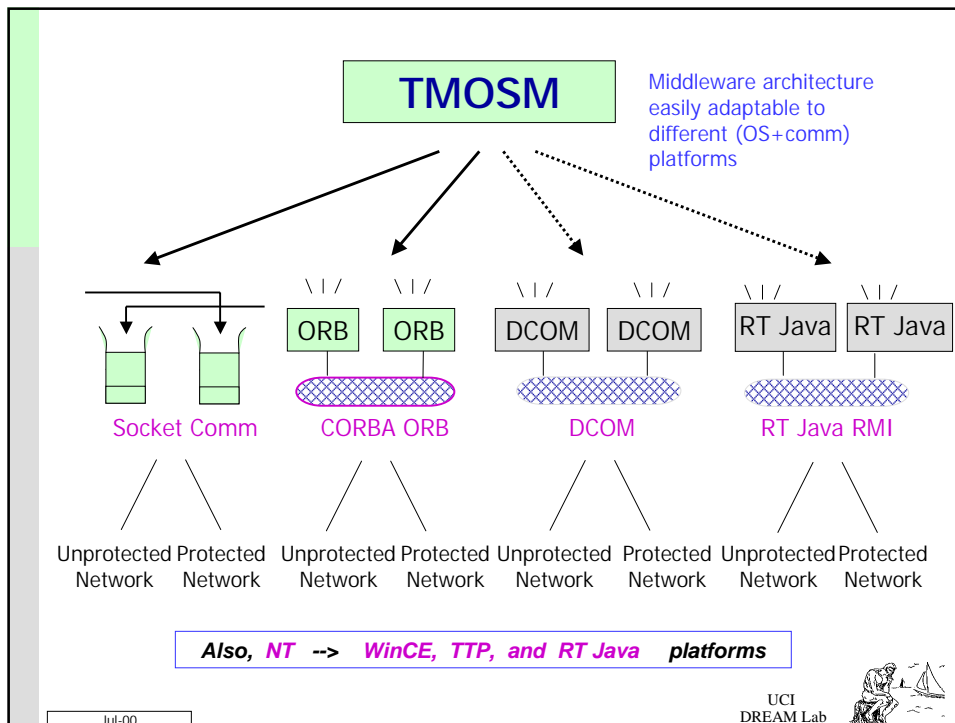


## TMOSM -- Thread Structure

- **WTST (Watchdog Timer & Scheduler Thread): *Master Micro-Thread***
  - Manages the scheduling / activation of all other threads in TMOSM and checks if there are deadline violations
- **MMCT (Middleware Message Communication Thread)**
  - Distributes messages coming through the communication network to their destination threads and sends messages destined for other middleware instantiations
- **LIIT (Local I/O Interface Thread)**
  - Manages local I/O activities such as serial character I/O and disk I/O
- **VMST (Virtual Main System Thread)**
  - A **virtual thread** representing all application and utility threads including:
    - SpM threads
    - SvM threads
    - *Utility threads*

Jul-00

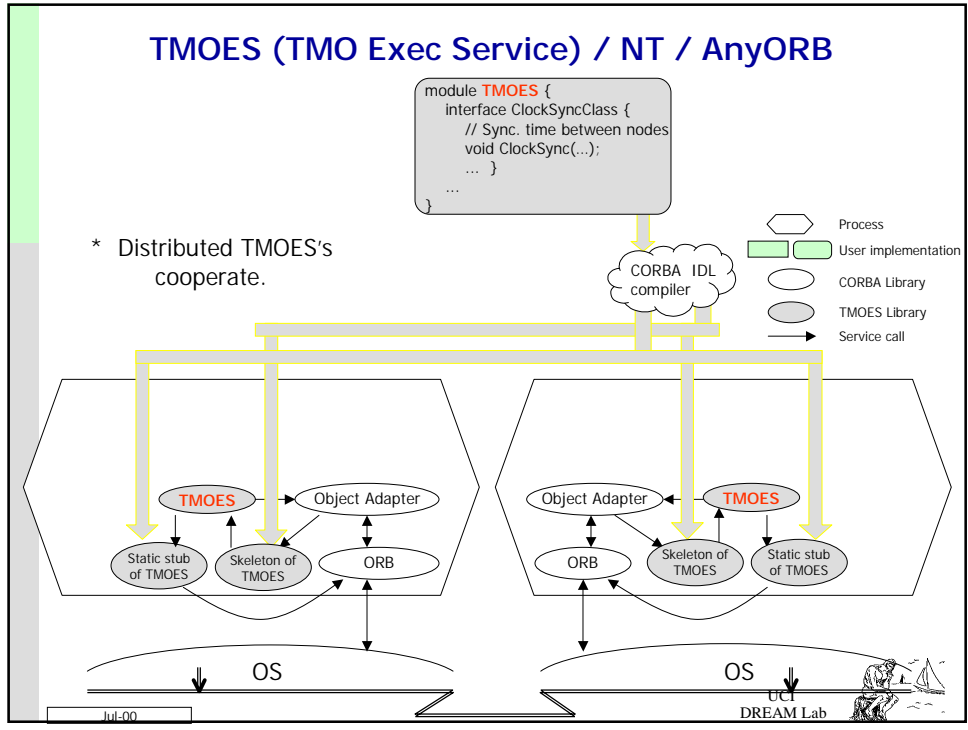
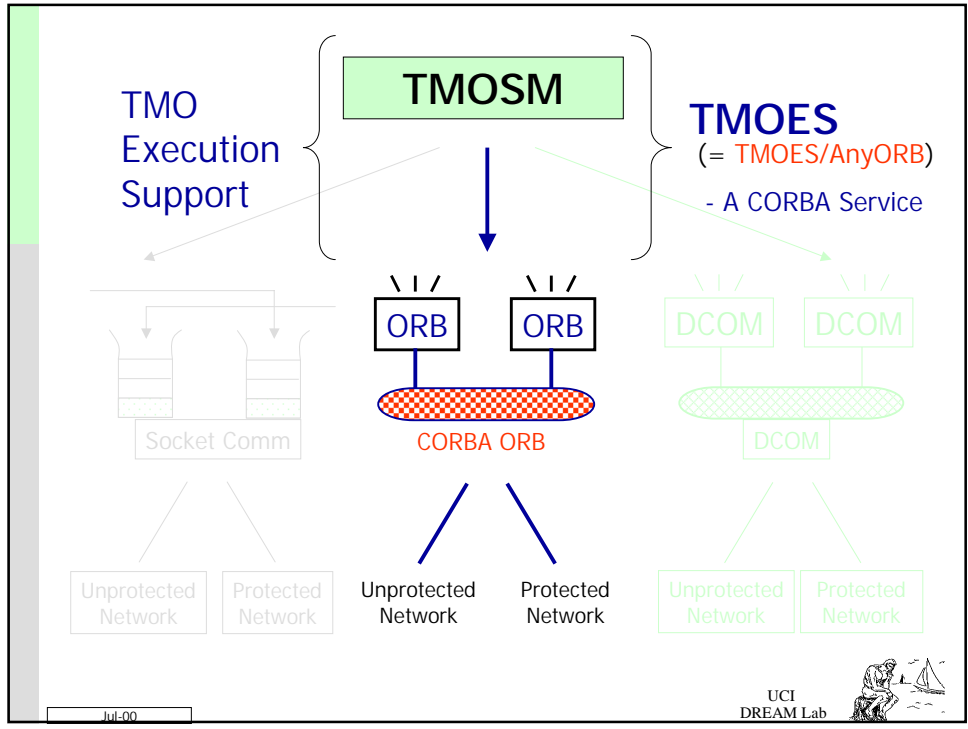
UCI  
DREAM Lab

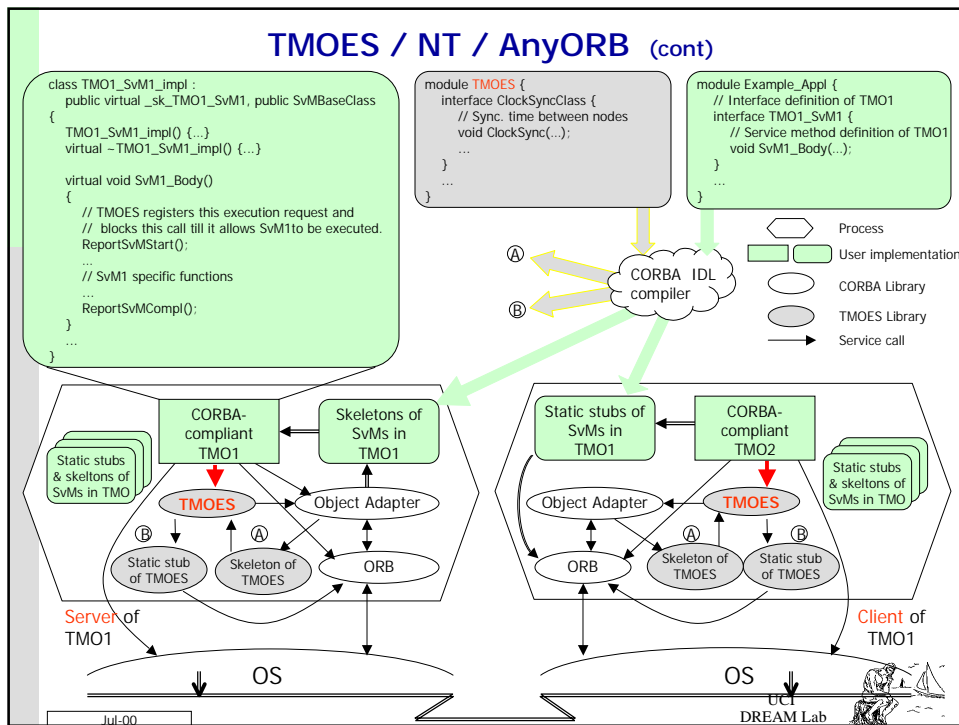
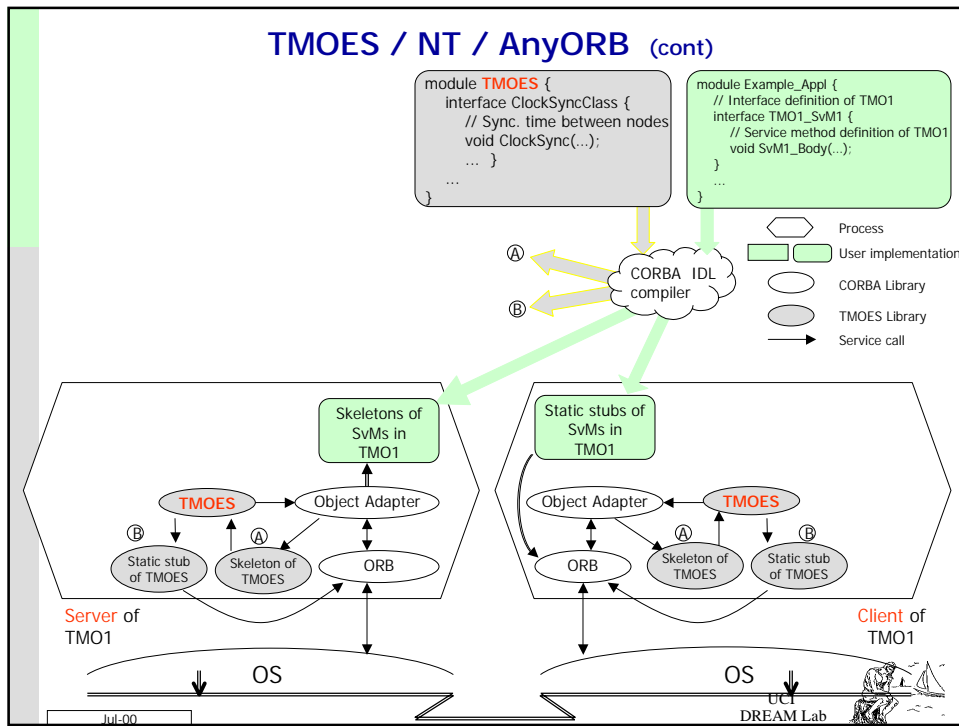


Jul-00

UCI  
DREAM Lab

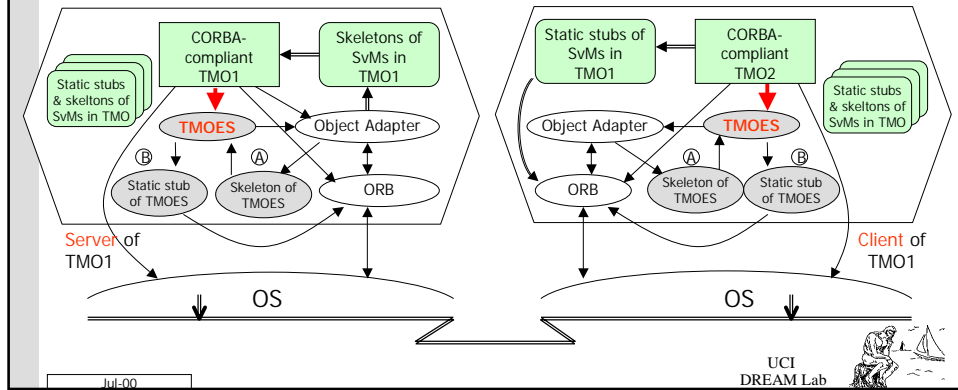






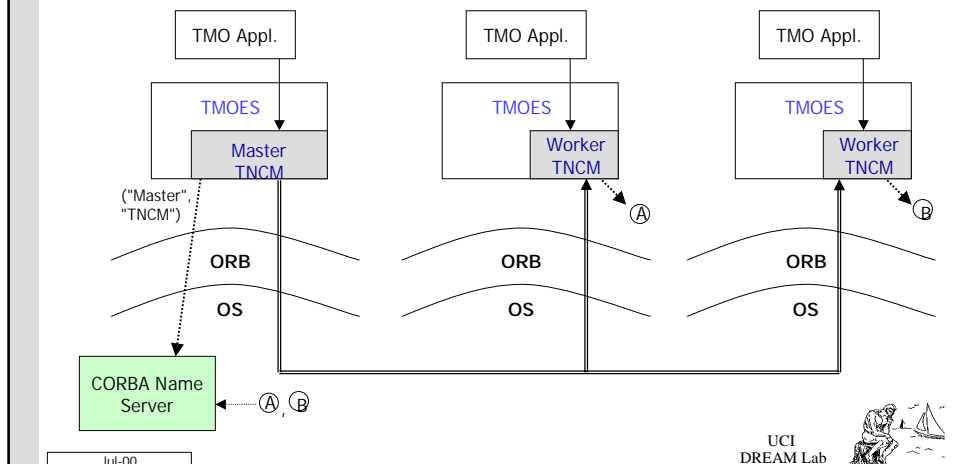
## TMOES / NT / AnyORB (cont)

- A part of the local TMOES receives requests for cooperation from remote TMOES's *via the stub-skeleton paths*.
- An application TMO is allowed to *directly call* a part of the local TMOES for services.
  - This is for performance reasons only, not for any logical / functionality reason.
  - Candidate for migrating into ORB ?



## TMO Net Configuration Manager (TNCM)

- A part of TMOES (or a CORBA service, Coop Net. Config. Mgr) which maintains configuration information such as node name, location, etc.
- One TNCM is the *master TNCM*, while other TNCMs are *worker TNCMs*.
- Each TNCM registers itself with the CORBA Name Server.



## TMO Net Configuration Manager (TNCM) (cont.)

### • Master TNCM

- Accepts a registration of each worker TNCM and synchronizes the worker's clock with the master's clock.
- Sends the TMOES list and the start time to each worker after all workers are registered.

### • Worker TNCM

- Registers itself with the master TNCM and follows an order to synchronize clocks.
- Waits for the TMOES list and the start time from the master.

#### Master Procedure

M1:  
TMO Appl. Calls StartTMOES() at the beginning.

M2:  
TNCM Master registers itself with CORBA Name Server as ("MASTER", "TNCM").

M3:  
Master waits for registration requests from TNCM Workers

M4:  
Upon registering a new TNCM Worker, Master performs Clock Sync. with the Worker.

M5:  
After all Workers are registered, Master sends the list of registered TMOESs and "Start TMO application at xxxx ms" order to each Worker.

#### Worker Procedure

W1:  
TMO Appl. Calls StartTMOES() at the beginning.

W2:  
Worker contacts CORBA Name Server to register itself and obtain a ref. to TNCM Master.

W3:  
Using a ref. to Master, Worker registers itself with TNCM Master.

W4:  
Waits for Clock Sync. request and perform it.

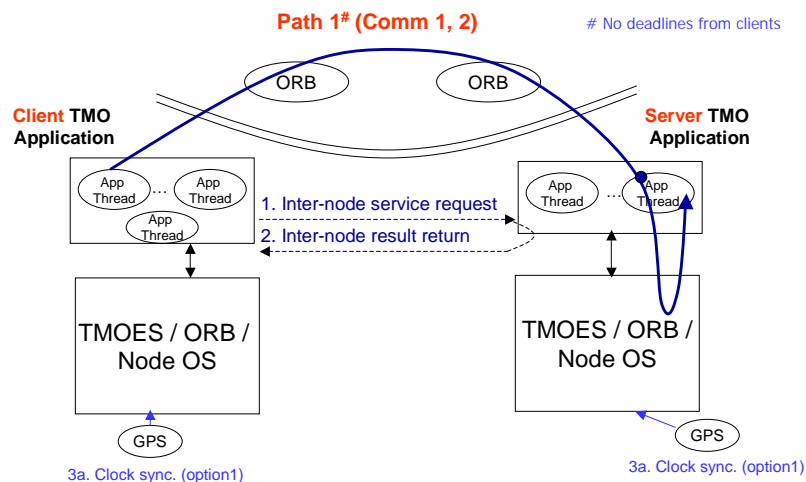
W5:  
Worker waits for "Start TMO application at xxxx" order. Worker sleeps until xxxx and then starts TMO Appl.

Jul-00

UCI  
DREAM Lab



## TMOES Communication Mechanisms

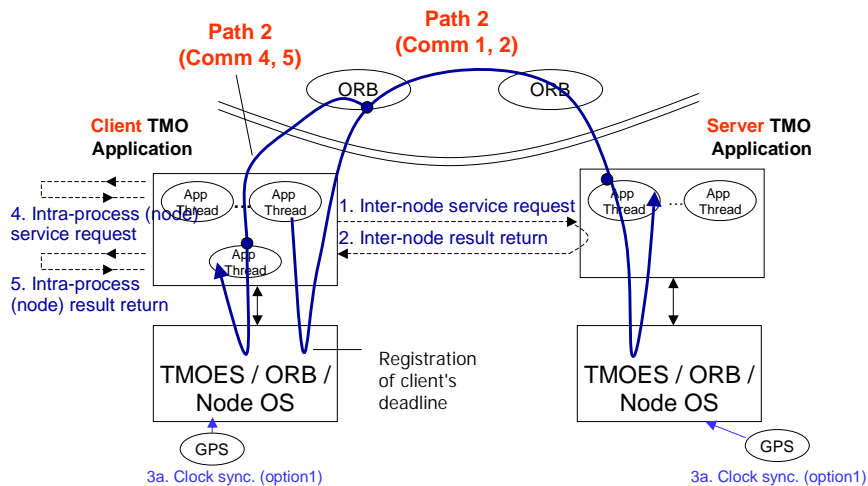


Jul-00

UCI  
DREAM Lab



## TMOES Communication Mechanisms (cont.)

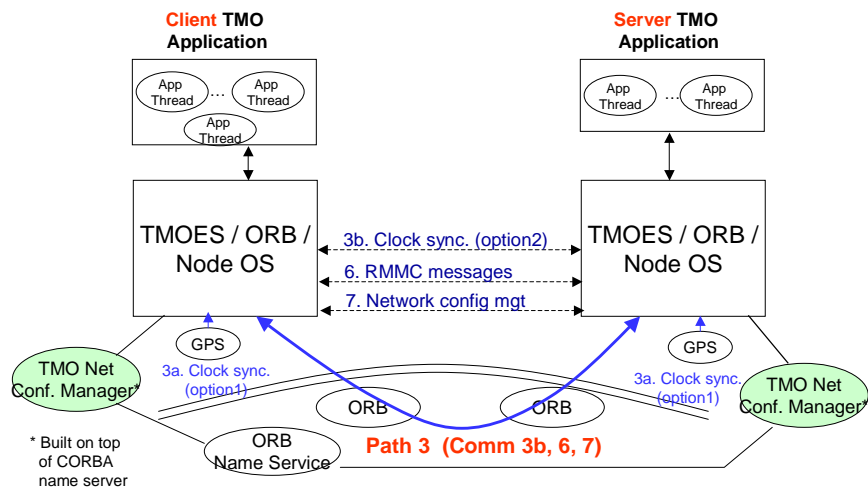


Jul-00

UCI  
DREAM Lab



## TMOES Communication Mechanisms (cont.)

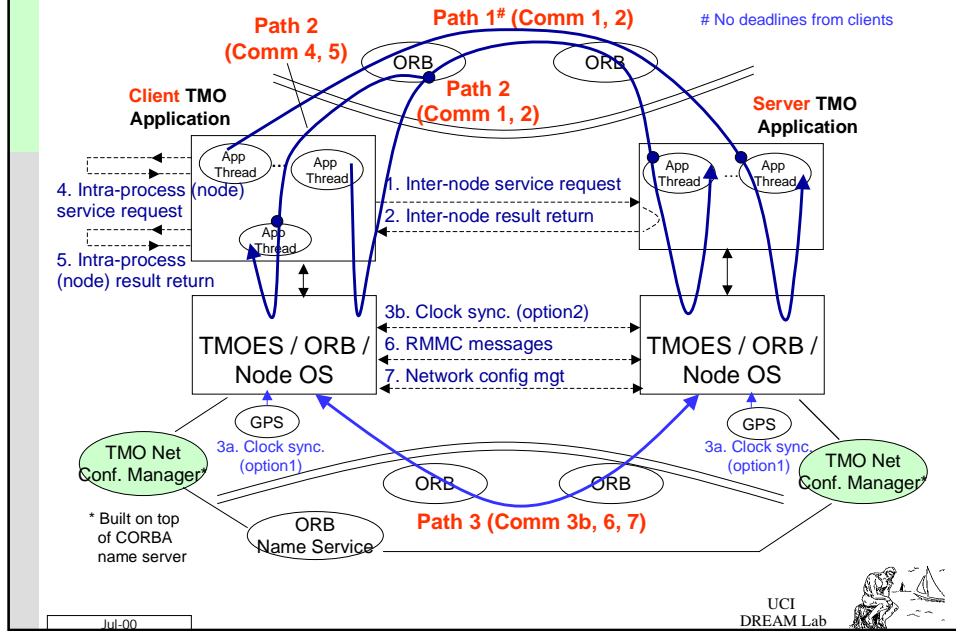


Jul-00

UCI  
DREAM Lab



## TMOES Communication Mechanisms (cont.)



## Object-Method Call Types

### Types of object-method calls in basic CORBA

- Blocking call** (in both static interface & DII)  
No deadline
- One-way call** (in both static interface & DII)  
No result return and no deadline
- Deferred synchronous call** (DII)  
No deadline

To determine when the request is done, the caller must use the `poll_response()`. `get_response()` is a blocking call.

(An application-level flag created in the ODS of the client will be used as an output parameter to be set by the service method to indicate the completion of the service.)

### Types of object-method calls in CORBA + TMO

- Blocking call without deadline**  
= basic CORBA type 1  
Approach 1. Use CORBA IDL
- Blocking call with deadline**  
Approach 1. `BlockingCallForAgent`(void \* AgentFuncName, long Deadline);
- NonBlocking call without result return**  
= basic CORBA type 2 (= one-way call)  
Approach 1. Use CORBA IDL
- NonBlocking call with result return**  
= basic CORBA type 3 (= Deferred sync call)  
Approach 1. Use CORBA IDL  
Approach 2. `NonBlockingCallForAgent`(void \* AgentFuncName);  
\* If a deadline were to be imposed on the arrival of the return result, then approach 2 must be used.
- 4a. NonBlocking Check Result**  
Approach 1. Application-level flag must be checked.  
Approach 2. `NonBlockingCheckAgentResult()`;  
A status flag for each agent service call maintained inside TMOES must be checked.



## Object-Method Call Types (cont)

### Types of object-method calls in CORBA + TMO

#### 4b. Blocking Check Result Without Deadline

Approach:

1. Application-level flag must be checked.
2. **BlockingCheckAgentResult()**;

#### 4c. Blocking Check Result With Deadline

This can be done only if approach 2 was used for making a non-blocking call for an SvM. A violation of the deadline will be handled as in case of **BlockingCallForAgent()**.

Approach: 1. **BlockingCheckAgentResultWithDeadline()**;

#### 5. Client Transfer Call

Realized by a restricted programming style rather than a middleware service.

#### 6. RMMC Message

API: **ConnectToRMMC()**, **DisconnectRMMC()**, **AnnounceEventMsg()**, **RecvEventMsg()**, **G-UpdateStateMsgVar()**, **ReadStateMsgVar()**.

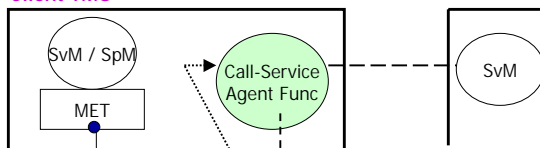
Jul-00

UCI  
DREAM Lab



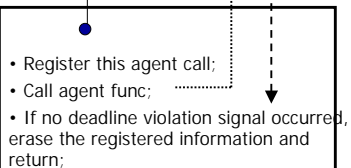
## BlockingCallForAgent() in TMOES

### Client TMO



BlockingCallForAgent( • , deadline)

### TMOES



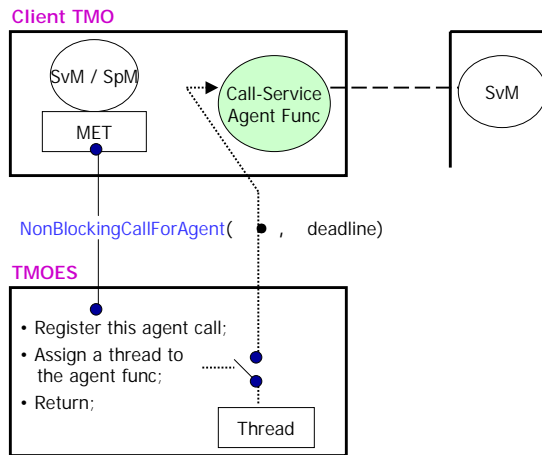
❖ BlockingCallForAgent() and a few other SvM-call related operations are float functions.

Jul-00

UCI  
DREAM Lab



## NonBlockingCallForAgent() in TMOES



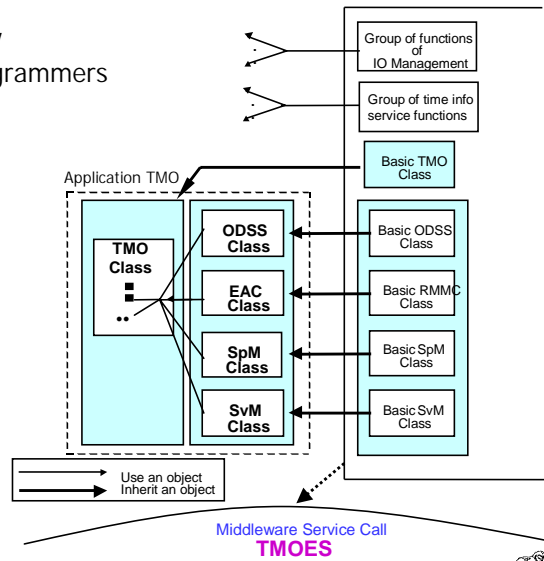
Jul-00

UCI  
DREAM Lab



## TMOESL (TMOES Library)

User-friendly API library  
for CORBA TMO programmers



Jul-00

UCI  
DREAM Lab



## Conclusion

---

- **TMOES** is an adaptation of TMOSM to the CORBA environment
  - Requires no change in the ORB core
- TMOES enables efficient development of CORBA-compliant TMO-structured RT distributed applications supported by **COTS ORBs**
  - **TMOES / NT / AnyORB** tested on OmniORB and TAO
- Future research
  - **Migration** of functionality of TMOES down to ORB and IDL areas
  - TNCM (TMO Net Config Mgr) needs to be expanded and formalized more fully

Jul-00

UCI  
DREAM Lab

