HARD-RoCS: Goals and Approaches

Kane Kim
UCI DREAM Lab
khkim@uci.edu, http://dream.eng.uci.edu/

For presentation at
the WG 10.4 Meeting, Sal
January 2003

Why am I working on Middleware?

• Even the RT OS kernel field was invaded by industry giants some years ago
  – I had to run away

• There is hope that good ideas / techniques / mechanisms demonstrated in middleware can be picked up by the OS industry and migrate them into future Dist. OS products.
My Research Paths of Last 25 Years

- Main attempts

Concurrent Programming  \(\rightarrow\)  Fault Tolerance  \(\rightarrow\)  RT Applications

Better Computing System Construction Technology: DREAM
(Dist. RT Ever Available Microcomputing)?

Major Dependability Techniques Pursued


PTC  D-CONV  DRB  PRHB  SNS  PSTR  RFRM  PPTR  RIF-driven reconf mgt & resource mgt

Process replicas

TMO replicas

Network surveillance

RT FT multicasts

No demo? demo?
Gadgets Used

- Cromemco Z2 (Z80 based) nodes connected via RS 232 serial comm cables
  - 1979
  - Extended Concurrent Pascal Machine
  - CP/M -> SCP DOS
- Rack-mounted OEM Z8000 single-board computer nodes + Central Data cabinet Z8000 nodes connected via dual-port shared memories
  - 1983
  - ECPM
  - Z8000 Monitor
- Rack-mounted OEM Z8000 single-board computer nodes connected via Crossbar switches
  - 1985
  - ECPM
  - Z8000 Monitor
  - 1991

Gadgets Used (cont)

- PC nodes connected via RS 232 serial comm cables
  - 1985
  - ECPM
  - MS DOS
- M68000 box nodes connected via RS232 serial comm cables
  - 1993
- PC nodes connected via Ethernet
  - 1988
- iPaq network
- Network of MEMS prototype nodes
- PC nodes connected via TTP monitors which are in turn connected via TTP Bus
  - 2003
- OptIPuter: Optically connected wide area distributed comp systems
Research Objectives

- A middleware architecture enabling easy composition and analyzable QoS
  - Tentatively named HARD-RoCS (Highly Autonomous RT Distributed Robust Computing Support)
- Pluggable, analyzable, and parameterized (PAP) components of middleware supporting real-time objects, fault tolerance, and security enforcement
  - Including their prototype implementations
- Demonstration of middleware (HARD-RoCS) and its components with real application scenarios

Target Figures of Merit

- Ease of developing networked embedded computing systems yielding tight service time bounds and recovery time bounds. (=> Development time and End-to-end timing analyzability)
- Values of service time bounds and recovery time bounds achieved for a specified range of possible type-frequencies of faults
- Ability to replace and adapt component versions (=> PAP (Pluggable, Analyzable, Parameterized) Components)
Target Applications

- Distributed multi-media processing
  - Next-generation RT VR (Virtual Reality)
  - Multi-party multi-media conferencing and collaboration
  - Distributed orchestra (?)
- Transportation automation
- Military command-control
- Non-stop web servers
- Time-sensitive health care
  - Monitoring of out-patients and patients under intensive care
  - Remotely controlled surgery

Powerful Building-Block Needed

- Without **concrete QoS requirement specifications**, it is difficult to avoid blind/fuzzy search for dependable computing approaches.
  - Spec must be concrete in both **output accuracy requirements** and **risks caused by QoS losses**.
- **Top-level services** and associated QoS requirement specifications must be specified and each decomposition of such specifications must be facilitated
  - Such decomposition is essentially a design based on **stepwise refinement**
- A powerful building-block is needed
  => **TMO** in my view
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

TMO (Time-triggered Message-triggered Object) Scheme

- A natural easy-to-use extension of the C++ / Java technology into an RT distributed software component programming technology

High-Level Specification of Timing Requirements

Basically, Clients' DRAs and Servers' GCTs (+ Deadlines for Output Actions)

- No priority, No thread
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

TMO Programming Scheme (cont)

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;

Capabilities for accessing other TMO’s and network environment incl. logical multicast channels, and I/O devices

Service Time Guarantees

Message-triggered Service Methods (SvMs)

“Relative time domain”
**TMO Programming Scheme (cont)**

### Ease of Creation of a Remote Service Call

- **Just two statements** are needed to create a remote service call to an SvM in another TMO!!
  - Gate declaration
  - Service call thru the Gate

```c
TMOGateClass Gate0 ("TMO2", "SvM2", tm4_DCS_age, (7*1000*1000));
Gate0->NonBlockingSR (&SvM2Para, sizeof(SvM2Para), Timestamp);
```

### 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.

**TMO Programming Scheme (cont)**

### Time-Triggered Spontaneous Methods (SpM’s)

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

```c
Name of TMO
```

```
AAC
SpM 1
SpM 2
```

- **Capabilities for accessing other TMO’s and network environment i.e. logical multicast channels, and I/O devices**
- **Time-triggered (TT) Spontaneous Methods (SpM’s)**
- **Message-triggered Service Methods (SvM’s)**
- **“Absolute time domain”**
- **“Relative time domain”**
**Time-Based Coordination of Distributed Actions**

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

"Let's start a chorus at 2pm"

NOT
"Open port 77 at 128.200.9.180; Wait for reply; ---"

**TMO Programming Scheme (cont)**

- **Connections to the network environment** as possible data members:
  - **TMO gates** (= access capabilities) (opening the services available from other, possibly remote, TMO's)
  - **Real-time Multicast & Memory-replication Channels (RMMCs)** (incl. distributed replicated variables)
    - **Location-transparent grouping** of cooperating TMOs
    - **An alternate mechanism for inter-connecting TMO groups**
    - Yields a **tight time bound** for a fault-tolerant multicast
  - **Based on the RFRM (Release-time based Real-Time Fault-Tolerant Multicast) scheme**
• Time-triggered (TT-) or spontaneous methods (SpM’s)
• Time-window imposed on each output action & method completion
• Connections to the network env’t (e.g., RMMC’s and TMO access capabilities) as data members

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

Pipelining of SvM executions are supported.
**TMO Support Middleware on Windows NT/XP:**

**TMOSM / XP / Socket**

- Logical connections ≤ Remote TMO Calls, RMMCs
- Application

**TMO Support Middleware on Windows NT/XP:**

**Time-Grains to be used in TMO Programming with TMOSM/XP**

- Time-grain: 10 millisecond
- Time-grain: 3 millisecond
- Time-grain: 1 millisecond
TMOSM Support Library (TMOSL)

User friendly API library for C++
TMO programmers

void main()
StartTMOengine ()
TMOSM Class
TMOSM Support Library (TMOSL)

Also, formal education started at UCI!!
Visual Studio for TMO (ViSTMO)

- Properties of TMO networks
- Top-Down Design & Spec of Essential Parameters
  - Graphics-based design editor
- Code generation
  - Code-framework Generator
  - C++ code for TMO class defs in Visual Studio projects files
  - Config.ini files
- Method Body Implementation
- Other tools
  - Compiler / Debugger (MS Visual Studio)
- Timing Analyzer
  - Compiling and debugging

An example of a TMO network

- Node 1: Invest Adv TMO
  - Node 2: Nasdaq TMO
    - Node 3: Dow TMO
      - Node 4: Prophet Smith
  - Node 5: Nasdaq TMO
    - Node 6: DowJones RT DB
  - Node 7: SpM1
  - Node 8: SvM1

UCI, GWU, Purdue
Technical Approach
Exploration of New Design Paradigms

- Time-based coordination of distributed actions (TCoDA)
- High-level programming of time-sensitive distributed actions
- **End-to-end timing analyzability**

\[
\text{f} \{
\begin{align*}
\text{Timing analyzability at every layer and in every component}
\end{align*}
\}
\]

A Middleware Architecture Skeleton - **HARD-RoCS**
• Will support design and execution of highly autonomous and yet timeliness-guaranteed distributed computing objects such as TMOs.

• With RT DOS available, time-based coordination of distributed actions becomes an easy job for application developers.

• Will enable QoS (quality-of-service)-driven scheduling & management of distributed resources, thereby making it easier for application developers to produce implementations meeting application timing requirements.

• An API model approximating an idealistic real-time distributed object language which does not require a new compiler, will be developed.
A small subset of functionality will be kept. E.g., TT-function, deadline violation alarm, etc

- Functionality is mostly kept

- RT DOS: VM (COTS)

- QOS-Driven Scheduling

- TMOSM

- RT DOS (Distributed Object Support) (cont)

- The API which wraps the services of RT DOS and is an approximation of an idealistic RT distributed object language, TMOSL 3.0, has been established.
  - An earlier version, TMOSL 2.2, along with the corresponding RT DOS, TMOSM 2.2, was used effectively in an undergraduate senior-year course on “Introduction to RT Distributed Programming” at UCI a year ago.
  - The lecture notes and that earlier version (2.2), now called the α version, have been tested at several R&D organizations.
  - The β version (3.0) is currently used in the course at UCI. Plan to distribute this version to many more organizations during the winter quarter (Jan – March) of 2003.

- Plan to port this onto Linux.
Display data

- Sensor values are collected and sensors and relays are coordinated in a timely manner in spite of various resource failures.

Risk Incursion Function (RIF)

- Risks: Damaging impacts of deadline violations and/or other QoS losses on the application mission.

- RIF (Risk Incursion Function, a.k.a., Benefit Loss Function)
  - := Relation (Loss in timed value accuracy of each output action, Potential application damage)
  - := Relation (QoS loss, Risk)
Risk Incursion Potential Function (RIPF)

System-level RIF and Derived RIF (= RIPF)

**RIPF** (Risk Incursion Potential Function) = Derived RIF
= Relation (Accuracy loss in intermediate output, Potential risk)

- Derivation of RIPFs := Allocation of Time Budgets & Budget-Overrun Penalties
RIF (RIPF) Examples

Type I: Hard Deadline
- Risk vs. earliest possible output time
- Convex function (Polynomial function, i.e., ax^3 + bx^2 + cx + d)
- Output action time
- Earliest possible output time
- Deadline
- Type II: Soft Deadline
- Risk vs. earliest possible output time
- Concave function (i.e., ax + b or sqrt(x) + c)
- Output action time
- Earliest possible output time
- Deadline

Type III: Soft deadline followed by a hard deadline
- Convex function (Polynomial function, i.e., ax^3 + bx^2 + cx + d)
- Output action time
- Earliest possible output time
- Deadline
- Type II: Soft Deadline
- Risk vs. earliest possible output time
- Concave function (i.e., ax + b or sqrt(x) + c)
- Output action time
- Earliest possible output time
- Deadline

Top-down Multi-step Allocation of Time Budgets & Budget-Overrun Penalties

Entire Application viewed as one TMO

The TMO is divided into multiple TMOs, The RIPFs are also derived from the RIFs

Elaborated as a TMO network (basic scheduling unit is SvM supported by a thread)
Technical Approach & Deliverable - NS (Network Surveillance)

- Basically a (partially or fully) decentralized mode of detecting faulty and repaired status of distributed computing & comm components.
- An important metric: Detection latency bound
- Basic techniques that exploit time-based synchronization principles will be used as initial cornerstones in building up this technology base:

1. The supervisor-based network surveillance (SNS) scheme (by Kim) for network surveillance in point-to-point network based systems;
2. The time-triggered protocol (TTP) scheme (by Kopetz) for network surveillance in bus-LAN based systems.

Progress - NS (Network Surveillance)

Supervisor-based NS (SNS) scheme

- Duties of the elected supervisor
  - Performs basic duties of workers
  - Judges the health conditions of worker nodes and links among them
  - Informs relevant nodes of detected faults
- Basic duties of workers
  - Exchanges heartbeats with neighbors at frequency f,
  - Sends a suspicion report (on the health of a neighbor) to the supervisor
- Special duties of the supervisor’s neighbors
  - Makes a group decision on the health of the supervisor
  - Participate in a new supervisor election
- One prototype implementation on XP has been produced.
Progress -
**NS (Network Surveillance)**

- A hierarchical extension of SNS has been developed.
  - A system may use many desktops, pocket PCs, motes, etc.

- A version is being incorporated into the Singing Spy Tracker demo involving notebooks, iPaqs, and motes.

---

Technical Approach & Deliverable -
**RORS (RT Object Replica Support)**

- Will support fault-tolerant execution of RT objects with tightly bounded time costs.

- Important metric: **Recovery time bound**

- Techniques exploiting **synchronized active replica** construction principles
  - Will be used as initial cornerstones in building up this technology base:
    1. The **Primary-Shadow TMO Replication (PSTR)** scheme (by Kim) which is an extension of the DRB (distributed recovery block) scheme for fault tolerance in RT object-based systems.
    2. The **Time-Triggered Architecture (TTA)** scheme (by Kopetz) for replication with voting in bus-LAN based systems.

- Will also support passive replicas: **PPTR** (Primary-Passive TMO Replication).

- Location-aware replicas will be supported.
**Progress - RORS (RT Object Replica Support)**

- A (quick-and-dirty) prototype implementation of a subset of PSTR has been implemented.
- A C3 distributed computing testbed, which is called CAMIN (Coordinated Anti-Missile Interceptor Network) and based on a LAN of 5 PCs, has been used to experiment with the prototype implementation.
- Full PAP versions of the PSTR support component and the passive replica support component are under development.

---

**Technical Approach & Deliverable - FTRM (Fault-tolerant Time-bounded Reconfig Mgt)**

- Will be layered on layered on RT RL / RT DOS / RT platform.
- Will support fault-tolerant time-bounded reconfiguration of mobile servers.
- Temporary and permanent failures of server objects (incl. both application objects and middleware objects, e.g, RT RL), server stations, and comm infrastructure components will be handled.
- A glue among NS, RORS, and RT RL.
- **Adaptive configuration mgt:**
  - Aimed for maintaining the minimum required quality of services (QoS’s) of critical components for longest possible periods.
**Progress - FTRM**  
*(Fault-tolerant Time-bounded Reconfig Mgt)*

- A high-level framework of an RIF-driven FTRM has been established.

---

**Technical Approach & Deliverable - AFTRM**  
*(Attack-resistant Fault-tolerant time-bounded reconfig mgt)*

- An extension of the FTRM middleware
- Will possess the additional capability & mechanisms for enabling applications to resist:
  - not only against random failures of computing and communication components
  - but also against security attacks
  - via authentication and time-bounded reconfiguration of mobile servers.
- The security attacks to be considered include denial-of-service attacks.
- Will be capable of fault-tolerant authentication.
Progress - AFTRM

- A high-level framework for Hierarchical Authentication has been established.
  - Contribution of Bharat Bhargava’s research group at Purdue
- Incorporation of this scheme into the RT RL (Registry-Lookup) support component is under way.
- Resource allocation schemes which disable denial-of-service attacks are under study.

But

- I should perhaps forget this.
  Instead, focus on completing ROAFTS (= HARD-RoCS – Security Mechanisms) & writing a monograph.

Summary

- I have had a chronic disease of underestimating how long it takes to demonstrate a small idea!