Middleware Platforms for Fault-Tolerant, QoS, Real-Time Computing

Luis Lopes

CRACS & INESC-Porto/FCUP-DCC
lblopes@dcc.fc.up.pt

July, 3rd
CRACS, FCUP-DCC & EFACEC Sistemas de Electrónica, S.A.

- Center for Research in Advanced Computing Systems (CRACS):
  - University of Porto, Faculty of Sciences, CS Department.
  - INESC-Porto Associated Laboratory.

- EFACEC SE, Transportation:
  - Flagship projects:
    - Dublin Light Metro Network.
    - Tenerife Light Metro Network.
    - Porto Light Metro Network.

- collaboration with Carnegie-Mellon University
Problems in current CORBA middleware:
- High latency, low throughput, no QoS, low level of dependability.

Why not use Fault-Tolerant CORBA and Real-Time CORBA?
- Hard integration of both standards in one solution.
- FT-CORBA’s high level approach adds overhead to the system.

Alternative solution:
- Push Fault-Tolerance and Real-Time protocols from the ORB level down to the networking infra-structure.
- Use P2P overlays with integrated real-time and fault-tolerance support built-in, rather than a client-server infra-structure.
RTP\textsubscript{M} Overview

- RTP\textsubscript{M} uses a Service Oriented Architecture (SOA) with following characteristics:
  - Resilient infrastructure:
    - P2P Networking.
    - Fault-Tolerance support.
    - Virtualization aware.
  - High performance runtime:
    - Soft real-time support.
    - QoS support.
    - Adjustable tradeoff between throughput and low latency.
  - Portability:
    - Uses ACE, a real-time framework with QoS support.
    - Initial support for Linux. Planned support for Win32, VxWorks.
RTP\textsubscript{M} Overview

- RTP\textsubscript{M} Architecture Layout:
  - User Services & Applications.
  - Kernel.
  - P2P Network.
  - ACE.

Luís Lopes
Middleware
Operating System Interface

- ACE, portable real-time framework with QoS support:
  - Supported OSes:
    - Linux, Windows, VxWorks, QNX, Mac OS X, HP-UX ...
  - Characteristics:
    - Event handling
    - Signal handling
    - IPC
    - Shared memory management
    - Tasks
    - Dynamic service configuration

P2P Networking

- **Customizable infra-structure:**
  - Each overlay can be customized to handle a particular application domain.

- **Overlay modules:**
  - Membership:
    - Handles the overlay's dynamic topology.
  - Messaging:
    - QoS aware messaging infrastructure.
  - Security:
    - Enforces the security policies.
  - Routing:
    - Maintains the routing information.
  - Discovery:
    - Controls dynamic resource searching.
Real-Time Support

- **Real-Time support:**
  - Adjustable cross-layer behavior:
    - Queuing for better throughput with added latency.
    - Direct upcalling for low latency with lower throughput.
    - Thread models allow adjustable behavior.
Fault-Tolerance & QoS Support

- **Fault-Tolerance support:**
  - Fault-Tolerance embedded support in the P2P.
  - Avoids the overhead of cross-layer interaction.

- **QoS support:**
  - Allows multiplexed and non-multiplexed connections.
  - CPU reservation for critical tasks.
  - Threading models to support real-time.
  - User customization of QoS behavior with QoS parameter.
Introduction

RTP

Final Remarks

Our Approach

Architecture

Kernel Architecture

- **Real-Time & QoS:**
  - Manages all aspects of the real-time & QoS policies.

- **Security:**
  - Contains the security policies used to control access to resource.

- **Network manager:**
  - Manages all network overlays in the system.

- **Virtualization:**
  - Manages all the aspects of the migration of services.

- **User modules:**
  - Dynamical loadable modules, allows access by the user to privileged resources.
User Applications & Services

- Access to the infra-structure done through kernel services (mostly).
- Mobility through the use of virtualization:
  - Allows cloning and service migration out-of-the-box.
  - Supports legacy services.
  - Enhanced performance using Intel VT-x and AMD-V CPU extensions.
  - Framework for virtualization abstraction (libVirt).
- Lightweight Service Mobility:
  - Only migrates the service’s state, through the use of checkpoints.
  - Service must provide the serialization implementation.
  - Allows for low-overhead service migration.
Current Status

- A real-time overlay framework:
  - Extended ACE support for low-level reservation mechanisms.
  - Network primitives with support for real-time.
- Virtualization:
  - Initial virtualization performance tests (KVM).
  - QEMU hacking for having a 1-1 mapping between virtual CPUs and physical CPUs. This allows CPU reservation in the context of virtual machines.
- First P2P overlay implementation almost complete.
## Short Term Evolution

- Complete the first prototype of $\text{RTP}_M$.
- Integration of Virtualization.
- Linux's Kernel extensions.
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Current Status</th>
<th>Future Work</th>
<th>Runtime Implementation Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Remarks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Runtime Implementation Strategies

- **Minimizes priority inversion:**
  - Priority mutexes (with OS support) (ACE).
  - Support for multiplexed and non-multiplexed connections.
  - Lock-free shared objects (libatomic_ops).

- **Optimized use of system memory**
  - Memory cache pool (ACE).
  - Zero-copy buffers.

- **Flexible execution model** (ACE & TAO)
  - Layers can execute requests:
    - Using queuing for better throughput.
    - In sequence execution for lower latency.

- **Flexible execution strategies** (ACE & TAO)
  - Leader-Followers.
  - Thread-per-Request.
  - Thread-Pool with priority lanes.
Extensions to the Linux’s kernel for added real-time and QoS support

- Design of a scheduling framework (ARTiS):
  - Earliest deadline first, rate monotonic, etc
- Asymmetric scheduling / CPU reservation (ARTiS):
  - Per-core task assignment.
  - Core reservation for critical tasks.
- Enhance networking (Linux’s community):
  - Network packet prioritization.
  - Network channels (zero copy buffers).

Integration of virtualization:

- QoS enhancement with the use of concepts brought by the XenSockets, allowing for optimized communication paths between RTP_M runtime and virtualized services.
RPC Core Service

- **C++ IDL Compiler**
  - Stub: Binds the client application with the RPC Service.
  - Skeleton: Binds the RPC Service with the User Services.
  - Uses SWIG to create bindings to other languages such as JAVA.
- RPC services such as Name Service and Event Service are fully integrated in the P2P networking.
- Supports 1-n invocations.