Middleware Support for Aperiodic Tasks in Distributed Real-Time Systems

Yuanfang Zhang, Chenyang Lu and Chris Gill
Department of Computer Science and Engineering
Washington University in St. Louis
Patrick Lardieri and Gautam Thaker
Advanced Technology Laboratories
Lockheed Martin

This work was supported in part by the DARPA Adaptive and Reactive Middleware Systems (ARMS) program (contract N66001-03-1-8015) and NSF CAREER award CNS-0448554. Approved for public release. Distribution unlimited.

Outline

- Problem
- Integrated middleware architecture
  - Admission control service
  - Deferrable server mechanism
- Empirical evaluation
- Conclusions

Motivating Applications

- Aperiodic tasks in many mission-critical distributed real-time applications
  - have hard end-to-end deadlines
  - are critical to the system
- Examples
  - Fire detection
  - Target detection

Gap between Theory and Middleware

- Theoretical techniques exist for aperiodic scheduling
  - Aperiodic Utilization Bound (AUB) [Abdelzaher, 2004]
  - Aperiodic Servers:
    - Polling Server [Sha 1986]
    - Deferrable Server [Strosnider 1995]
    - Priority Exchange [Lehoczky 1987]
    - Sporadic Server [Sprunt 1989]
    - Slack Stealing [Lehoczky 1992]
- Existing middleware lacks support for real-time aperiodic tasks
  - Lack scheduling mechanism
  - Lack online admission control

Contributions:

- Integrated middleware architecture within TAO’s federated event channel
  - End-to-end scheduling service for aperiodic/periodic tasks
  - Online admission control service
- Support two alternative approaches:
  - Aperiodic Utilization Bound (AUB)
  - Deferrable Server (DS)

Aperiodic Utilization Bound: Theory

- End-to-end Task $T_i$
  - $D_j$: End-to-end deadline
  - $C_j$: Subtask execution time on node $j$
  - Aperiodic or periodic

- Current job set $S(t)$
  - Jobs that have arrived but whose deadlines have not expired
  - Note: A job may still be "current" after completion

- Synthetic utilization of processor $j$:
  $$U_j(t) = \sum_{i \in S_j(t)} \frac{C_j}{D_j}$$

Releasing rule: when a processor becomes idle, the contribution of all completed aperiodic jobs to the processor’s synthetic utilization is removed.
Utilization Bounds for End-to-End Aperiodic Tasks

- Under end-to-end deadline monotonic scheduling, all end-to-end deadlines are met if
  \[
  \sum_{j=1}^{N} u_j (1 - \frac{U}{2}) \leq 1
  \]
- For arbitrary task graphs
  \[
  \frac{u_j (1 - \frac{U}{2}) + \max_{j \neq 1} \left( \frac{u_j (1 - \frac{U}{2})}{1 - u_j} \right)}{1 - u_j} \leq 1
  \]

Example

\[
\begin{align*}
U_1 &= 0.2 + 0.1 + 0.1 = 0.4, \quad U_2 = 0.13 + 0.1 + 0.02 = 0.25, \\
U_3 &= \max(0.06, 0.1, 0.1) = 0.1 \\
\sum_{j=1}^{N} \frac{u_j (1 - \frac{U}{2})}{1 - u_j} &= 0.93 \leq 1
\end{align*}
\]

TAO’s Federated Event Channel

- Event Channel (EC) dispatches events locally according to their priorities.
- Gateway forwards events to remote processors.

Admission Control Service

- Round trip delay < 1.4 ms
- Considered in schedulability analysis

Design Decisions

- Critical Tasks
  - Send an event to notify the central admission controller
  - Hold the task in a waiting queue and wait for the reply
- Noncritical Tasks
  - AC may eject noncritical periodic tasks when new critical ones arrive (critically-awareness admission policy)
- Aperiodic Tasks
  - Update the synthetic utilization when an aperiodic job is released or reaches its deadline
  - When the CPU is idle, “idle detector” reports the completed aperiodic jobs to AC (resetting rule)
- Periodic Tasks
  - Do not update synthetic utilization at every release or deadline of a periodic job.

Deferrable Server: Theory

- A periodic server executes all aperiodic tasks
- Budget: maximum time the server can run in a period
  - Bound aperiodic task’s impact on periodic tasks
- Algorithm
  - Server is suspended when its budget runs out or there are no pending aperiodic events
  - Server’s budget is replenished at the end of each period
  - Requires new run-time scheduling mechanism
- Schedulability Analysis
  - worst case end-to-end response time <= end-to-end deadline
Deferrable Server: Middleware Mechanism
- First middleware implementation of bandwidth preserving server
  - Server thread processes aperiodic events (2nd highest priority)
  - Budget manager thread manages the budget and controls the execution of server threads (highest priority)

Experimental Platforms
- Task Set
  - One task set contains 4 aperiodic tasks and 5 periodic tasks
  - Randomly generate 60 task sets with synthetic utilization of 0.3, 0.4, 0.5 and 0.6, respectively
- Testbed

Comparison of AUB and DS
- Resetting mechanism is effective
- DS and AUB are comparable
- DS requires more complex run-time mechanism

Impact of Criticality
- AUB or DS: do not eject noncritical tasks to accept new critical tasks
- AUB or DS with Criticality: eject noncritical tasks to accept new critical tasks

Conclusions
- Integrated middleware architecture for real-time end-to-end aperiodic/period tasks
  - Admission control service
    - Aperiodic Utilization Bound (AUB)
    - Deferrable Server (DS)
  - Scheduling service: DS on top of priority-based OS
  - Integration with TAO’s federated event channel
  - Empirical results
    - DS is highly efficient at the middleware level
    - Admission control incurs acceptable delays
    - AUB is comparable to DS while requiring simpler run-time mechanisms

Reading
Real-Time Object Request Broker Middleware

- **Common Object Request Broker Architecture (CORBA)**
  - CORBA specifications
  - OMG is the standards body
  - Over 800 companies
  - CORBA defines interfaces, not implementations
  - Object Request Brokers (ORB) allow clients to invoke operations on distributed objects transparently from
    - Object location
    - Programming language
    - Operating System
    - Communication protocols and interconnect
    - Hardware

- **CORBA Reference Model**
  - Client invokes operations on objects.
  - An Object includes
    - An interface specified by an Interface Definition Language (IDL)
    - Servant(s) that implements the IDL interface

- **Stubs and Skeletons**
  - Translate between platform-dependent data formats and a common data representation.
  - Generated by an IDL compiler based on the IDL interface.
  - Ensure platform/language transparency.

- **ORB Core**
  - Deliver requests to objects and responses to clients
  - Communicate using a General Inter-ORB Protocol (GIOP)
    - e.g., Internet Inter-ORB Protocol (IIOP) on TCP
  - Typically a run-time library linked to applications

- **Object Adapter**
  - Demultiplexes each incoming request to the right servant/operation
  - Make the upcall to the operation
Limitations of CORBA

- Lacks real-time specification interfaces to applications
- Applications cannot specify task rate, deadline, and importance
- Lacks real-time enforcement
- Does not map task specification to priorities of threads
- Contains significant priority inversion
- Lacks performance optimization
- Poor worst-case and average latency

Latencies and Priority Inversions

- Open-source Real-Time CORBA
- \( \geq 1,000,000 \) SLOC
- 100+ person years of effort
- Pioneered R&D on DRE middleware design & optimizations

The ACE ORB (TAO)

nORB: A Small-footprint ORB