Middleware Support for Aperiodic Tasks in Distributed Real-Time Systems

Chenyang Lu
Motivating Applications

• Aperiodic tasks in many distributed real-time applications
  – have **hard** end-to-end deadlines
  – are **critical** to the system

• Examples
  – Fire detection
  – Target detection

Alert! $T'-T > 20$

Sensing temperature

Displaying temperature

Processing data

Collecting data
Theory vs. 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]

- Middleware lacks support for real-time aperiodic tasks
  - Lack scheduling mechanism
  - Lack online admission control
Contributions

- Middleware architecture within federated event channel
  - End-to-end scheduling for aperiodic & periodic tasks
  - Online admission control service

- Support and compare two alternative approaches
  - Aperiodic Utilization Bound (AUB)
  - Deferrable Server (DS)
Aperiodic Utilization Bound

- End-to-end Task $T_i$
  - $D_i$: End-to-end deadline
  - $C_{ij}$: 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 $j$:
  \[
  U_j(t) = \sum_{T_i \in S(t)} \frac{C_{ij}}{D_i}
  \]

  **Resetting rule:** when a processor becomes idle, remove the contribution of all completed aperiodic jobs from the processor’s synthetic utilization

- End-to-end deadline of a task is guaranteed if

\[
\sum_{1 \leq j \leq 4} \frac{U_j(1-U_j/2)}{1-U_j} \leq 1
\]
TAO Federated Event Channel

- Event Channel (EC) dispatches events locally.
- Gateway forwards events to remote hosts.
Admission Control Service

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

**Application Processor 1**
- Task arrives
- Send request
- Accept?
- Update util
- Set timer
- Timer fires
- Idle report
- Reject task
- Release task
- Not accepted

**Application Processor 2**
- Accept task
- Sched. analysis
- Accept

**Application Processor 3**
- Accept
- Timer fires
- Idle report
- Reject task
- Release task
- Not accepted
Design Decisions

- **Critical Tasks**
  - Send an event to notify the central admission controller
  - Hold the task in a waiting queue and waits for the reply

- **Noncritical Tasks**
  - AC may eject noncritical periodic tasks when new critical ones arrive
    *(criticality-awareness admission policy)*

- **Aperiodic Tasks**
  - Update utilization when an aperiodic job is released or reaches its deadline
  - When CPU idles, idle detector reports completed aperiodic jobs to AC
    *(resetting rule)*

- **Periodic Tasks**
  - Do not update synthetic utilization at every release or deadline.
Deferrable Server: Theory

- A periodic server executes all aperiodic tasks
- Budget: maximum time the server can run in a period
  - Bound aperiodic tasks’ impact on periodic tasks
- Algorithm
  - Server is suspended when its budget runs out
  - Server’s budget is replenished at the end of each period
- Schedulability Analysis
  - Worst case end-to-end response time \( \leq \) end-to-end deadline

- Admission test: end-to-end response time analysis
- Requires run-time scheduling mechanism
DS: Middleware Implementation

- First DS implementation on top of priority-based OS (e.g., Linux, POSIX)
- Server thread processes aperiodic events (2\textsuperscript{nd} highest priority)
- Budget manager thread (highest priority) manages the budget and controls the execution of server thread
Experimental Platforms

- **Task Set**
  - One task set contains 4 aperiodic tasks and 5 periodic tasks
  - Randomly generate 60 task sets with synthetic utilization of 0.3-0.6

![Diagram of experimental platforms]

- **ron.cse**
  - Pentium4 2.80GHz
  - 512MB RAM
  - 512KB cache
  - KURT-Linux 2.4.22

- **hermoine.cse**
  - Pentium4 2.80GHz
  - 512MB RAM
  - 512KB cache
  - KURT-Linux 2.4.22

- **norbert.cse**
  - Pentium4 2.53GHz
  - 512MB RAM
  - 512KB cache
  - KURT-Linux 2.4.22

- **harry.cse**
  - Pentium4 2.53GHz
  - 512MB RAM
  - 512KB cache
  - KURT-Linux 2.4.22

- **100Mbps Ethernet switch**
Comparison of AUB and DS

- Resetting mechanism is effective
- DS and AUB are comparable
  - But 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

**Synthetic utilization per processor**
6 Critical Tasks (4 aperiodic tasks and 2 periodic tasks)

**AUB with criticality is more effective than DS for critical tasks**
Conclusions

- Middleware 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 federated event channel

- Empirical results
  - DS is efficient at the middleware level
  - Admission control incurs acceptable delays
  - AUB is comparable to DS with simpler run-time mechanisms
Paper

- Y. Zhang, C. Lu, C. Gill, P. Lardieri and G. Thaker,
  Middleware Support for Aperiodic Tasks in Distributed Real-Time Systems,