Final Demo

- In class on 4/23
- 10 min per team
- Test your demo in advance
- Send your slides and video to Ruixuan by 11am
- All expected to attend the entire session
- It’ll be fun! 😊
Final Report

- Send report and materials to Ruixuan by 11:59pm, 4/30

- Report
  - Style follows conference papers in the reading list
  - 6 pages, double column, 10-pts font
  - Use templates on the class web page

- Supporting materials
  - Source code
  - Documents: README, INSTALL, HOW-to-RUN
  - Slides of your final presentation
Suggested Report Outline

Abstract

1. Introduction
2. Goals
3. Design
4. Implementation
5. Experiments
6. Lessons Learned
7. Related Works
8. Conclusion and Future Works
Peer Review

- For fairness in projects.

- Email me on 4/30
  - Percentage of contributions of each team member.
  - Brief justification.
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

Displaying temperature
Processing data
Collecting data

Alert!

$T' - T > 20$

Sensing temperature
Approach to Aperiodic Tasks

- Arrival times of aperiodic tasks are unknown
  - Possibly unbounded inter-arrival time
  - Impossible to provide offline guarantees

- Aperiodic task with a soft deadline
  - Maintain hard guarantees on periodic tasks
  - Reduce response time of aperiodic tasks

- Aperiodic task with a hard deadline
  - Online guarantees through admission control
Theory vs. Middleware

- Theoretical techniques for online 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
  \[
  \sum_{1 \leq j \leq 4} \frac{U_j(1-U_j/2)}{1-U_j} \leq 1
  \]
Event Channel (EC) dispatches events locally.

Gateway forwards events to remote hosts.
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 waits for the reply

- **Noncritical Tasks**
  - AC may eject noncritical periodic tasks when new critical ones arrive
    *(criticality-aware 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 ≤ 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 (2nd 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
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