

CS4220 Embedded Systems CS6235 Real-Time Systems

5A: Threads and RTOS

Instructor: Calton Pu
calton.pu@cc

TA: Younggyun Koh (young@cc)

1
1

Real-Time Threads

- Thread manipulation
 - Thread creation (RTthread_forkX)
 - Thread control (RTthread_sleep/wakeup)
- High level utilities
 - Conditions in monitors (RTcondition_wait, signal, broadcast)
 - Others: yielding processor, error handling, package configuration, etc

2
2

User-Level RT Threads

- RT threads scheduled consistently
 - Both kernel and user-level must guarantee the deadlines
 - Virtual processors at the kernel level (as RT threads)
 - The current virtual processor is demultiplexed into user-level RT threads
 - When blocked in kernel, current VP becomes a kernel virtual processor

3
3

User-Level Services

- User-level scheduler
 - User-level timers maintained by a user-level scheduler, through a kernel timer
 - User-level priority management (preemption at dispatching time)
- User-level deadline handler
 - Deadline exception handler

4
4

Recurring Theme (1)

- Modularization and componentization
 - Embedded systems evolve very fast (planned obsolescence)
 - RT systems evolve slowly (expensive verification of software and hardware)
 - Software engineering reasons (need to build good quality software in very short time)
 - Cost reasons: small footprint, predictable response time
 - Windows CE example

5
5

Recurring Theme (2)

- Specialization and customization
 - Finer granularity than componentization (usually it's easier within a component)
 - Both manual and tool-based: guaranteed correct source-to-source program transformation
 - Further reduction of footprint and improvement on predictability
 - Partial evaluation example

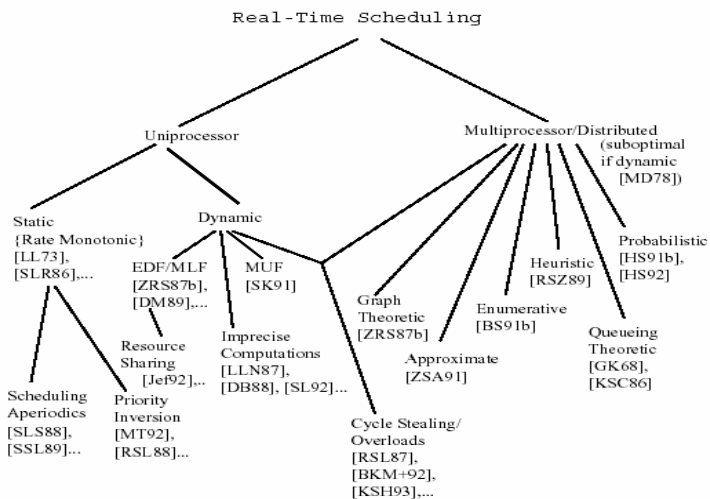
6
6

Survey of RTOS

- Main base components
 - RT thread scheduling
 - Thread management and synchronization
- Non-CPU (I/O) components
 - Virtual memory, file systems, RTDB
 - Network latency and bandwidth

7
7

RT Scheduling



Dynamic Schedules

- **Schedulability at thread creation time**
 - Periodic and sporadic tasks
 - Thread communications (conditions)
- **Priority inversion**
 - Waiting threads remain in the ready queue

9
9

Synchronization

- **Communications among threads**
 - Dependencies among threads: blocking
 - Priority inversion
- **Multiprocessors**
 - Critical sections shared by multiprocessors
 - Locks and lock-free synchronization

10
10

Real-Time Kernels

- Ada9X
 - Government standard that almost worked
- Simple run-time executives
 - VxWorks, pSOS, and many others
- Research RTOS
 - Spring, ARTS, Maruti, etc
- Add RT support to production kernels
 - RT-Mach, Posix-RT, RTX-Linux, RT-Java

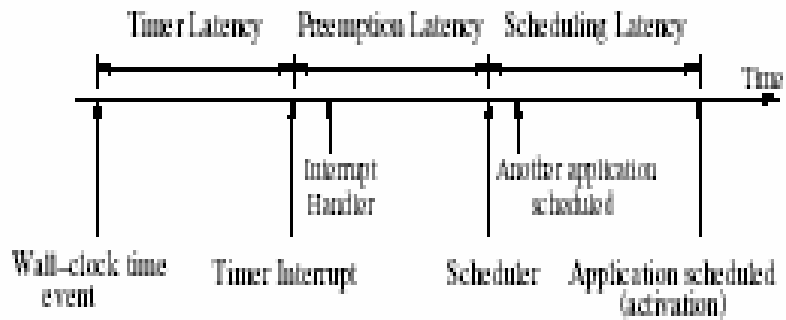
11
11

Retrofitting Approach

- Main base components of RT-Mach
 - Real-time threads
 - RT thread scheduling
 - RT thread synchronization
 - RT communications (IPC, messages)
- RT application support
 - Multimedia streaming, temporal paging system, etc

12
12

TSL Model



13
13

TSL Solutions

- Firm Timer reduces timer latency
 - Low overhead important
- Fine-grain kernel preemption reduces preemption latency
 - Responsive kernel
- Good schedulers reduce scheduling latency
 - Proportion-period scheduler

14
14

Summary of RTOS

- Fundamental trade-offs
 - Simple executives are predictable, but provide low-level services
 - High-level services increase variance
 - Moore's Law makes portability a must
- Some new directions since 1993
 - Feedback-based resource management

15
15

Motivation

- Scheduling algorithms fail at overload
- Prevent overload
 - Static schedules
 - Admission control
- Overload management
 - Congestion control
 - Adaptive (QoS-driven) resource management

16
16

Admission Control

- Asynchronous messages
 - Guaranteed dynamically
 - Conservative estimation: worst-case analysis of all messages in a node
 - Main problem: under-utilization
- How to sharpen the estimation and increase network utilization

17
17

Assumptions

- Network model
 - Diffserv architecture (with classes)
- Resource management components
 - **Configuration**: class & route determination
 - **Run-time admission control**: enforce admission policies when under overload
 - **Packet forwarding**: class-based static priority policy, FIFO within class

18
18

Utilization-Based

- Utilization-based admission control
 - Run-time decisions on flow establishment
 - Current utilization the only criterion
- Bandwidth availability along flow path
 - Safe levels of utilization
 - Determined during configuration

19
19

Conservative Estimation

- Configuration-time delay computation
 - *Traffic constraint* bounds usage per link
 - *Server delay bound*
 - Iterate the delay calculation on all servers
- Safe route selection
 - Satisfy deadlines of each class and route
 - Heuristics: min distance, min delay

20
20

Maximize Utilization

- Max utilization bounds depend on network diameter (theorem 4)
 - Given a utilization level, find safe routes
 - Converge on max utilization that has safe routes
- Their result (45%) is better than Shortest Path (33%)

21
21

Real-Time Systems

- Basic constraints of RTE systems
 - Predictable performance with limited resources
- Recurring themes
 - Modularization and component engineering
 - Specialization and customization
 - “Scheduling”: careful management of resources

22
22