Scheduling Aperiodic Tasks

- Hybrid task set: periodic tasks + aperiodic tasks
- Problem: Arrival time is unknown
- Sporadic task with a hard deadline
  - Inter-arrival time must be lower bounded
  - Schedulability analysis: treated as a periodic task with period = minimum inter-arrival time
- Aperiodic task with a soft deadline
  - Possibly unbounded inter-arrival time
  - Goals:
    - maintain hard guarantees on periodic tasks
    - reduce response time of aperiodic tasks

Background Scheduling

- Treat aperiodic tasks as lowest-priority tasks
- Advantages
  - Simple
  - Aperiodic tasks has no impact on the schedulability of periodic tasks
- Disadvantage
  - Aperiodic tasks have very long response times when the utilization of periodic tasks is high
  - Acceptable only if System is not busy
  - Aperiodic tasks can tolerate long delays

Polling Server

- Polling server (PS): a periodic task used to serve aperiodic requests
  - Period: ps
  - Capacity: cs
- Rules
  - Released periodically with period ps
  - Serves any pending aperiodic requests
  - Suspends itself if
    - it has used up its capacity, or
    - no aperiodic request is pending
  - Server capacity is replenished to cs in the next period

Schedulability

- The aperiodic requests have the same impact on periodic tasks as a periodic task.
- n tasks with m PS':
  \[ U_p + U_s + Us \leq U_b(n+m) \]
- Can have multiple PS' (with different periods) for different aperiodic requests
- Disadvantage: If an aperiodic request "misses" the execution of PS, it has to wait till the next period \( \rightarrow \) long response time.

Deferrable Server (DS)

- Unlike PS, DS preserves unused capacity until the end of the current period
- Better response to aperiodic requests
- However, DS’ impact on periodic tasks is different from an periodic task

Utilization Bound with DS

- Under RMS
  \[ U_b = U_p + n \left( \frac{U_p + 2}{2U_p + 1} \right) \]
- As \( n \rightarrow \infty \):
  \[ U_b = U_p + \ln \left( \frac{U_p + 2}{2U_p + 1} \right) \]
- When \( U_p = 0.186 \), min \( U_b = 0.652 \)
- System is schedulable if
  \[ U_p \leq \ln \left( \frac{U_p + 2}{2U_p + 1} \right) \]
Pointers
- Class hand-out
- Rate Monotonic
- EDF
- General
  - Real-Time Systems, Jane Liu.

Real-Time Operating Systems
- Proprietary kernels
- Real-time extensions to general-purpose OS

Proprietary Kernels
- Commonly used for small embedded systems
- Homegrown kernels
  - Highly specialized for specific applications
    - e.g., nuclear power plant
  - Less common
- Commercial RTOS

Features for Efficiency
- Small
- Minimal set of functionality
- Fast context switch
- Fast and time bounded response to interrupts
- Fixed or variable partitions of memory
  - May not support paging or virtual memory
  - Often support locking code and data in memory
- Sequential file that can accumulate data at fast rate
  - May be memory-based

Features for Real-Time
- Priority-based preemptive scheduling
  - At least 32 priority levels, commonly 128-256 priority levels
  - Priority inheritance/ceiling protocol
  - Usually does not directly support EDF
- System calls
  - Bounded execution times
  - Short non-preemptable code
- High-resolution system clock
  - Resolution down to nanoseconds
  - But it takes about a microsecond to process a timer interrupt

Code Size
<table>
<thead>
<tr>
<th>Name</th>
<th>Code Size</th>
<th>Target CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>pOSEK</td>
<td>2K</td>
<td>Microcontrollers</td>
</tr>
<tr>
<td>pOGS</td>
<td></td>
<td>P8-&gt;ARM Thumb</td>
</tr>
<tr>
<td>VxWorks</td>
<td>286K</td>
<td>Pentium -&gt; Strong ARM</td>
</tr>
<tr>
<td>QNX Nutino</td>
<td>&gt;100K</td>
<td>Pentium II -&gt; NEC</td>
</tr>
<tr>
<td>QNX RealTime</td>
<td>100K</td>
<td>Pentium II -&gt; SH4</td>
</tr>
<tr>
<td>OS-9</td>
<td></td>
<td>Pentium -&gt; SH4</td>
</tr>
<tr>
<td>Chorus OS</td>
<td>10K</td>
<td>Pentium -&gt; Strong ARM</td>
</tr>
<tr>
<td>ARIEL</td>
<td>19K</td>
<td>SH2, ARM Thumb</td>
</tr>
<tr>
<td>Creem</td>
<td>560 bytes</td>
<td>ATMEL 8051</td>
</tr>
</tbody>
</table>

- QNX context switch = 2400 cycles on x86
- pPOSEK context switch > 40 µs
- Creem -> no preemption
Other important features

- Conformance to Standards
- Real-Time POSIX API
- Modularity and configurability
  - Small kernel
  - Pluggable modules
- Networking support
  - TCP/IP

Development Environment

- Self-hosted system: applications are developed on the target platform
- OS must support compilers, debuggers, performance profilers
- Large memory demand
  - E.g., LynxOS
- Cross-platform development
  - E.g., Tornado environment for VxWorks OS

Example: VRTX

- Two versions
  - VRTXsa
    - RT-POSIX compliant
    - Full real-time support
  - VRTXmc
    - Optimized for power and footprint
- First RTOS certified by FAA
  - 100% code coverage in testing
  - e.g., Used by Boeing MD-11

Example: VxWorks

- Not a UNIX system, but provides most POSIX functions
- System calls with timeout
  - E.g., Semaphore operations.
- Used by NASA

Real-Time Extensions to General-Purpose OS

- Generally slower and less predictable than proprietary RTOS
- Much greater functionality and development support
- Standard interfaces
- Useful for soft real-time and distributed complex applications

Categories of RT-Linux

- Compliant kernels
  - modified native RTOS
  - Linux binaries can run without modifications
  - E.g., LynxOS
- Dual kernels
  - Hard real-time kernel sits below Linux
  - Real-time kernel traps all interrupts and schedules all processes
  - Linux runs as a low-priority process
  - No memory protection between dual kernels
  - E.g., RT-Linux (FSLabs)
- Core kernel modifications
  - E.g., TimeSys Linux, Manta Vista Linux