Sporadic Server Scheduling in Linux
Theory vs. Practice

Mark Stanovich
Theodore Baker
Andy Wang
Real-Time Scheduling Theory

- Analysis techniques to design a system to meet timing constraints
- Schedulability analysis
  - Workload models
  - Processor models
  - Scheduling algorithms
Real-Time Scheduling Theory

- Analysis techniques to design a system to meet timing constraints
- Schedulability analysis
  - Workload models
  - Processor models
  - Scheduling algorithms
Periodic Task

Task = \{T, C, D\}

jobs (j_1, j_2, j_3, \ldots)

Period = T

Deadline = D

Computation time

\( \text{WCET} = C \)

Release time

time
Periodic Task

sched_setscheduler(SCHED_FIFO)

clock_nanosleep()
Periodic Task

- Assumptions
  - WCET is reliable
  - Arrivals are periodic
- Not realistic for most tasks
Polling Server

Replenishment period

Job arrivals

Initial budget
Polling Server

- Type of aperiodic server
- CPU time no worse than an equivalent periodic task
  - Can be modeled as a periodic task
    - WCET = Initial Budget
    - Period = Replenishment Period
- Budget consumed as CPU time is used
  - CPU time forfeited if not used
- Replenish budget every period
Polling Server

- Good
  - Bounds CPU time
  - Analyzable workload
  - Simplicity
- Can be better
  - Faster response time if budget is not forfeited
Sporadic Server

Replenishment period

Job arrivals

Initial budget

replenishments
Sporadic Server

- Originally proposed by Sprunt et. al.
- Parameters
  - Initial budget
  - Replenishment period
- Bounds CPU interference for other tasks
- Fits into the periodic task workload model
- Better avg. response time than polling server
Sporadic Server

- Scheduling algorithm for fixed-task-priority systems
- Can be used in UNIX priority model
- `SCHED_SPORADIC` is a version of SS defined in POSIX definition
Implementation

- Linux 2.6.38
  - Softirq threading patch ported from earlier RT patch
  - Sporadic server implementation
- Uniprocessor
Sporadic Server Performance

- Metrics
  - Interference for lower priority tasks
  - Average response time
An experiment

A

Sends UDP packet with current timestamp

B

Receives UDP packets

Calculate response time based on arrival at UDP layer

Measure CPU time for 10 second burst
Measuring CPU Time

- Reger's “hourglass” technique
- Constantly read time stamp counter
  - Detect preemptions by larger gaps
  - Sum execution chunks
- Hourglass thread lower than SS thread
  - Measures interference from SS thread
Measuring CPU Time

- Network receive thread
  - Sporadic and polling server
    - Budget = 1 msec
    - Period = 10 msec
  - SCHED_FIFO
- Hourglass thread
  - SCHED_FIFO
  - Lower priority than network receive thread
CPU Utilization

- 10% utilization
- SCHED_FIFO
- sporadic server
- polling server

CPU utilization vs. sent packets (1000 pkts/sec)
Response Time

The chart shows the response time in milliseconds for different server types: SCHED_FIFO, sporadic server, and polling server. The x-axis represents the number of sent packets (1000 pkts/sec), while the y-axis shows the response time. The graph indicates the performance of these server types under varying load conditions.
Interference

- SS budget limited to CPU demand
- Additional overheads lower priority tasks
  - Context switch time
  - Cache eviction and reloading
- Not in theoretical workload model
- Guarantees of theory require interference to be included in the analysis
Polling Server

- $SS_{budget} + |2 \times CS_{time}|$

= aperiodic job arrival

= aperiodic job CPU time

Diagram showing the timing of events in a polling server.
Sporadic Server

- \( \text{time} = \text{aperiodic job CPU time} \)
- \( \text{arrival} = \text{aperiodic job arrival} \)
- \( \text{budget} = \text{replenishment period} \)

\[
SS_{\text{budget}} + \left| 2 \times CS_{\text{time}} \times \text{max\_repl} \right|
\]
Over Provisioning

- All context switch time may not be used
  - e.g., one replenishment per period
- Account for CS time on-line
  - Charge SS for each preemption
CPU Utilization

![Graph showing CPU utilization vs sent packets (1000 pkts/sec)]

- 10% utilization
- SCHED_FIFO
- sporadic server
- polling server
Response Time

The graph shows the response time (milliseconds) plotted against the number of sent packets (1000 pkts/sec). The graph compares different scheduling methods:

- **SCHED_FIFO** (red line with x marks)
- **sporadic server** (green square line)
- **polling server** (blue dotted line)

The x-axis represents the number of sent packets, while the y-axis represents the response time in milliseconds, ranging from 0.01 to 1000 milliseconds.
Analysis

- Light load
  - Sporadic Server
    - Low response time
  - Polling Server
    - High response time

- Heavy load
  - Sporadic Server
    - High response time
    - Dropped packets
  - Polling Server
    - Low response time
    - No dropped packets
Can we get the best of both?

Sporadic Server  
Light loads

Polling Server  
Heavy loads
Hybrid Server

- How to switch
  - Ensure bounded interference
  - SS with 1 replenishment is same as polling server
  - Coalesce replenishments
    - Push replenishments further into the future

- Switching point
  - Server has work but no budget
Sporadic Server
Sporadic Server
Response Time

![Graph showing response time for different server behaviors](image)
CPU Utilization

- 10% utilization
- SCHED_FIFO
- sporadic server coalesce (immediate)
- polling server

CPU utilization vs. sent packets (1000 pkts/sec)
Switching

- Immediate coalescing may be too extreme
  - CPU time could be used for better response time
- Gradual approach
  - Coalesce a few replenishments
Sporadic Server

time
Sporadic Server
Sporadic Server

time
Response Time

- SCHED_FIFO
- sporadic server coalesce (immediate)
- polling server
- sporadic server coalesce (gradual)
CPU Utilization

- 10% utilization
- SCHED_FIFO
- sporadic server coalesce (immediate)
- polling server
- sporadic server coalesce (gradual)

sent packets (1000 pkts/sec) vs CPU utilization
Conclusion

- Theoretical analysis provides solid guarantees
- Implementation must match abstract models
  - Additional interference terms need to be considered
- SS can fit into the theoretical analysis
Deferrable Server
Deferrable Server

- Bandwidth Preserving
  - Allow server to retain budget
- Periodically replenish budget
- WCET $\neq$ Budget
Response Time

- SCHED_FIFO
- Sporadic server coalesce (immediate)
- Polling server
- Sporadic server coalesce (gradual)
- Sporadic server (max_repl = 1)

**Y-axis:** Response time (milliseconds)

**X-axis:** Sent packets (1000 pkts/sec)
Replenishment Policy

- Initial budget
- Replenishment period
- Replenishment
- Arrival time (work available for server)
Bandwidth Preservation

- Initial budget
- Replenishment period
- Arrival time (work available for server)
- Replenishment

Diagram shows the flow of bandwidth over time, including initial budget, replenishment period, and arrival time.
Sporadic Server