Analysis of Hierarchical Fixed-Priority Pre-emptive Scheduling

Robert Davis and Alan Burns

Real-Time Systems Research Group

University of York

Roadmap

- Motivation
- Hierarchical Scheduling Problem
- Response Time Analysis
- Empirical Investigation
- Conclusions

Motivation

- Automotive and Avionics applications
- Emerging trend: multiple applications on a single processor
  - Made possible by the advent of advanced high performance microprocessors
  - Driven by the desire for cost reductions and functionality enhancement
  - Strong requirement for temporal isolation, systems must behave as if they were composed of multiple microprocessors

System Model

- Multiple applications on a single processor
  - Each application comprises multiple tasks
  - A Server is used to schedule each application
  - Server parameters:
    - Priority, period (T_S), capacity (C_S)
  - Each Server schedules a set of tasks
  - Task parameters:
    - Priority, period (T_i), deadline (D_i), execution time (C_i).
    - Worst-Case Response Time (R_i)
  - Fixed Priority Pre-emptive Scheduling
    - high level: server scheduling
    - low level: task scheduling

Servers

- Periodic Server
  - Invoked with a fixed period
  - Tasks executed until the server's capacity is exhausted, then suspended until capacity replenished at next period
  - If no tasks ready, server's capacity idled away
- Deferrable Server
  - Similar to Periodic Server
  - Server capacity deferred if no tasks ready, can be used later in the period
  - Any remaining capacity discarded at end of server period

Servers (continued)

- Sporadic Server
  - Differs from Periodic and Deferrable Servers: Capacity only replenished once it has been used
  - Capacity used at time t replenished at $t+T_S$
  - Worst-case interference due to Sporadic Server is the same as a periodic task
  - Complexity and overheads typically greater than Periodic and Deferrable servers: keeping track of replenishment times and amounts

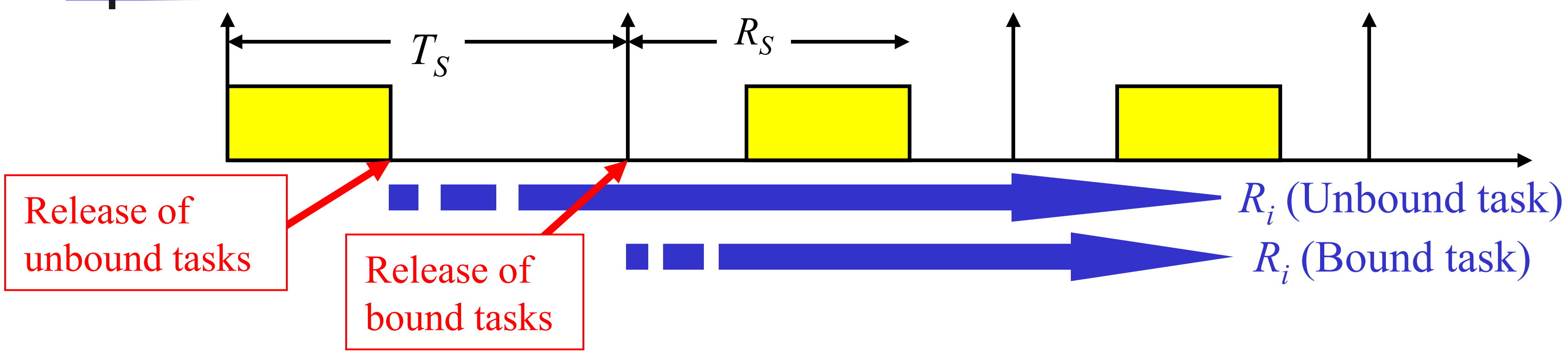
Bound and Unbound Tasks

- "Bound" tasks
  - Periodic task with a period an exact multiple of the server's period
  - Always arrive coincident with release of the server (replenishment of server capacity)
  - Release jitter effectively zero
  - No tasks can be bound to a Sporadic Server
- "Unbound" tasks
  - All tasks that are not "bound"
  - Tasks may be periodic, sporadic etc.
  - Release jitter effectively T_S-C_S

Schedulability Analysis

- Using Response Time Analysis:
  1. Determine scenario (critical instant) leading to worst-case response time for a task
  2. Calculate worst-case response time given critical instant arrival pattern
  3. Compare worst-case response time with task deadline

Critical Instant

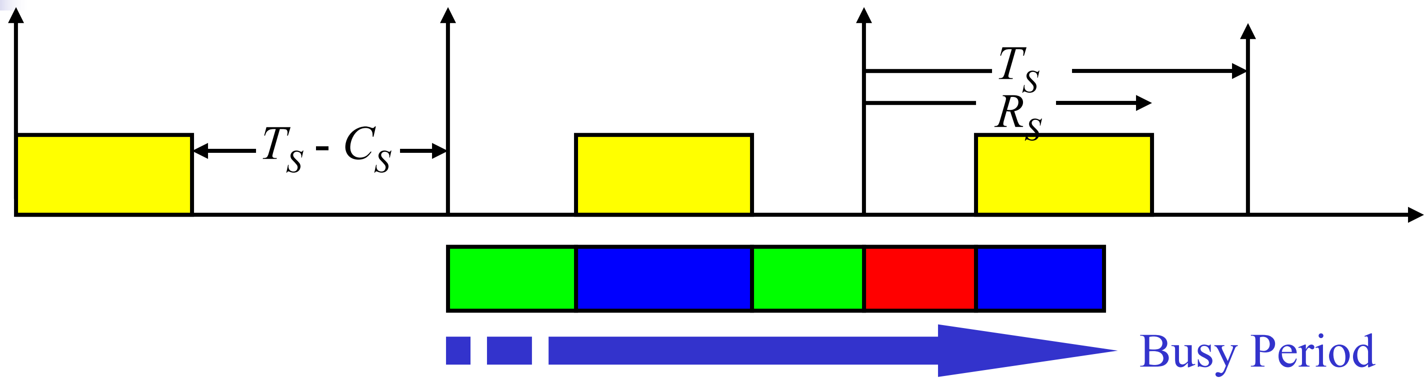


- Server capacity exhausted as early as possible then…
- Task of interest (if unbound) and all higher priority unbound tasks released.
- Task of interest (if bound) and all higher priority bound tasks released at the start of the server's next period along with the server.
- Subsequent server capacity available as late as possible due to interference from higher priority servers

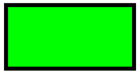
Exact Analysis

- To determine response time:
  1. Derive formula for the load $L_i(w)$ at priority i and higher released in a busy period of length w.
  2. Derive a formula for the length $w_i(L)$ of the priority i busy period that finishes when the server completes execution of the load L.
  3. Combine the above formulae to form a recurrence relation that can be solved to find the worst-case response time of the task at priority i.

Busy Period



- Three components:


'Gaps' in complete periods


Load due to tasks executed by the server


Interference in the last server period

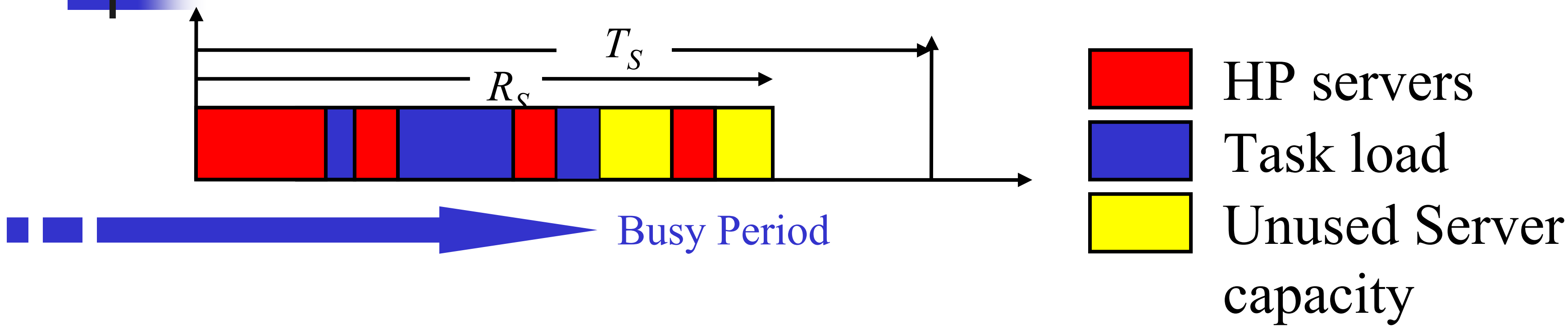
Busy Period

- Task load $L_i(w) = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w + J_i}{T_j} \right\rceil C_j$
- Gaps in complete server periods $\left(\left\lceil \frac{L_i(w)}{C_S} \right\rceil - 1 \right) (T_S - C_S)$
- Interference in final period $I_S(w)$

Interference

- Three models
  1. $(T_S - C_S)$ [Saewong 2002]
     - Safe but pessimistic
  2. $(R_S - C_S)$
     - Removes much pessimism, but some remains…
  3. $I_S(w)$ Exact computation…

Exact worst-case Interference



worst-case interference from higher priority servers in final period given by:

$$I_S(w) = \sum_{\forall X \in hps(s)} \left\lceil \frac{w - \left(\left\lceil \frac{L(w)}{C_S} \right\rceil - 1 \right) T_S + J_X}{T_x} \right\rceil C_X$$

Response Time Computation

$$w^{n+1} = L(w^n) + \left(\left\lceil \frac{L(w^n)}{C_S} \right\rceil - 1\right)(T_S - C_S) + \sum_{\forall X \in hps(s)} \left\lceil \frac{w^n - \left(\left\lceil \frac{L(w^n)}{C_S} \right\rceil - 1\right) T_S + J_X}{T_x} \right\rceil C_X$$

- Recurrence starts with $w_i^0 = C_i + \left(\left\lceil \frac{C_i}{C_S} \right\rceil - 1\right)(T_S - C_S)$
- ends when $w_i^{n+1} = w_i^n$ in which case $w_i^{n+1} + J_i$ is the task's worst case response time alternatively, recurrence ends when $w_i^{n+1} > D_i - J_i$ in which case the task is unschedulable

Example Analysis

- Simple system: Two Deferrable Servers
  - Considers two highest priority tasks executed by the lower priority server (full details in the paper)
  - Unbound tasks:

Task	C_i	T_i	D_i	Response Times R_i		
				(1)	(2)	(3) Exact
1	10	50	50	46	42	38
2	8	100	100	88	84	82

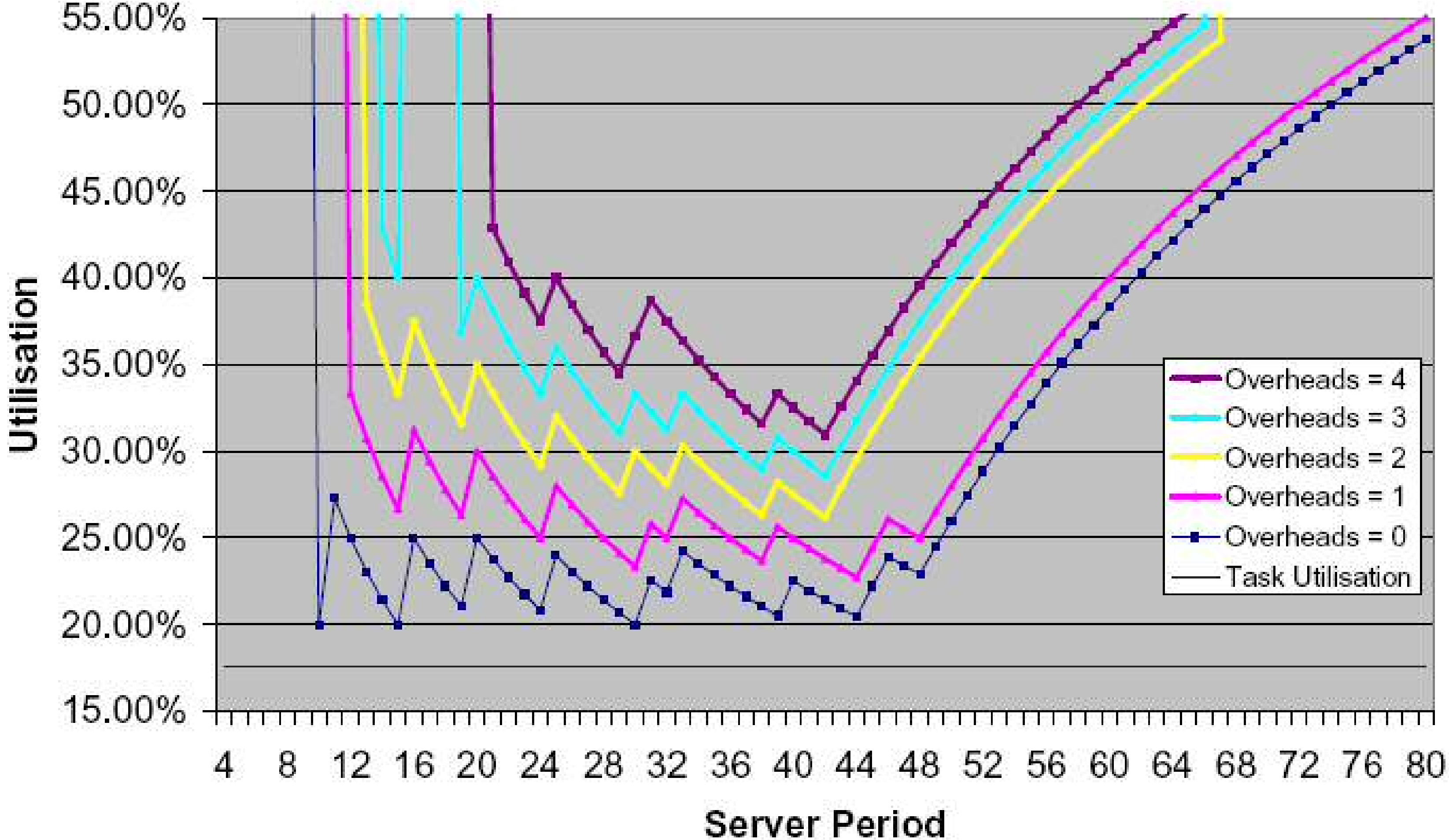
  - Bound tasks:

    Exact response times 26 and 70 - reduced by $T_S - C_S$ w.r.t unbound tasks

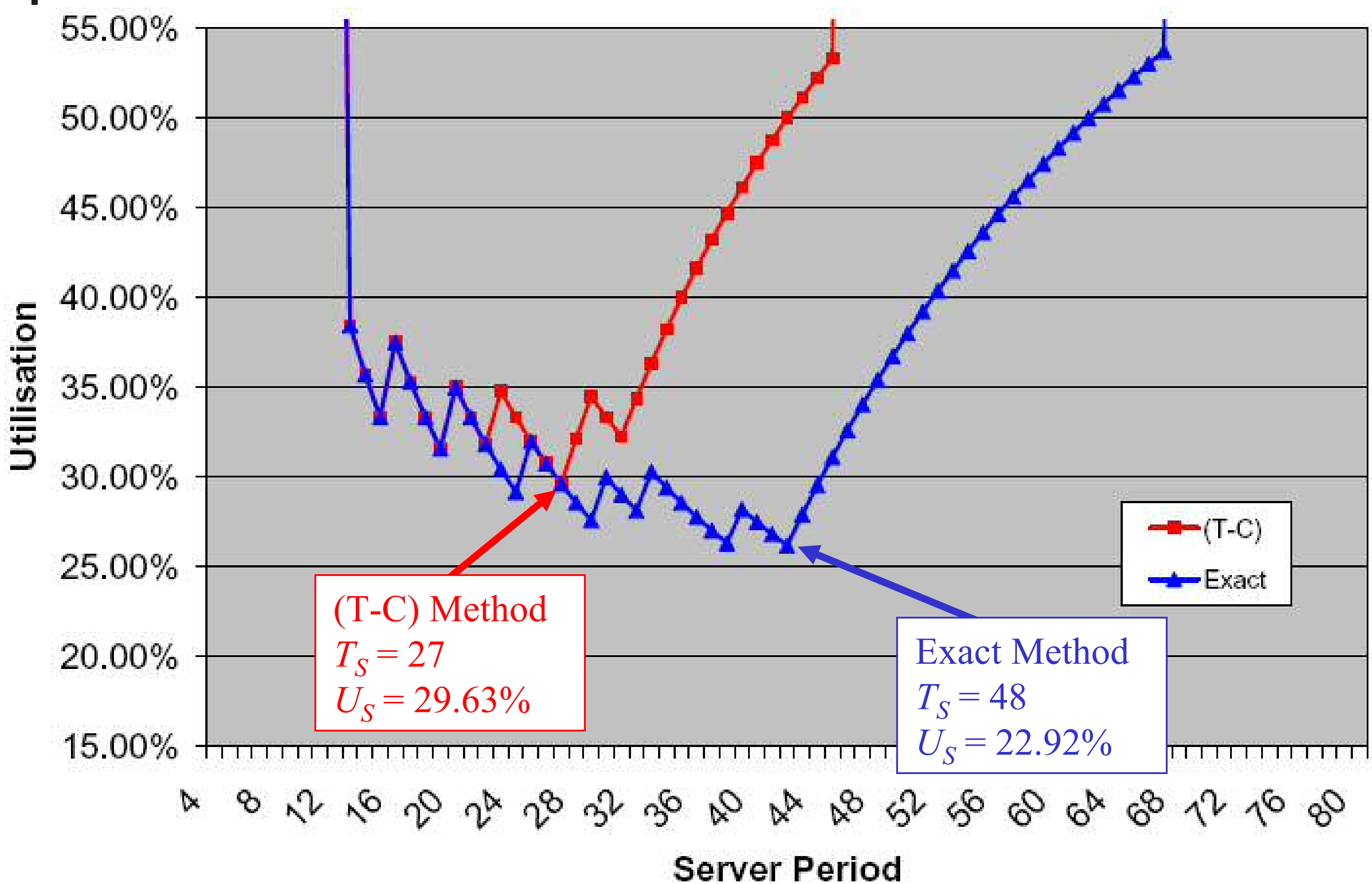
Empirical Investigation

- Plots minimum server utilisation required for schedulable system against server period
- Compares effects of:
  1. Server overheads
     - Essential otherwise infinitesimal server period is optimal
  2. Analysis methods
     - Exact v. previously published approaches
  3. Server Algorithms
     - Periodic v. Deferrable Server
  4. Bound v. Unbound tasks
     - Advantages of synchronising server and task release

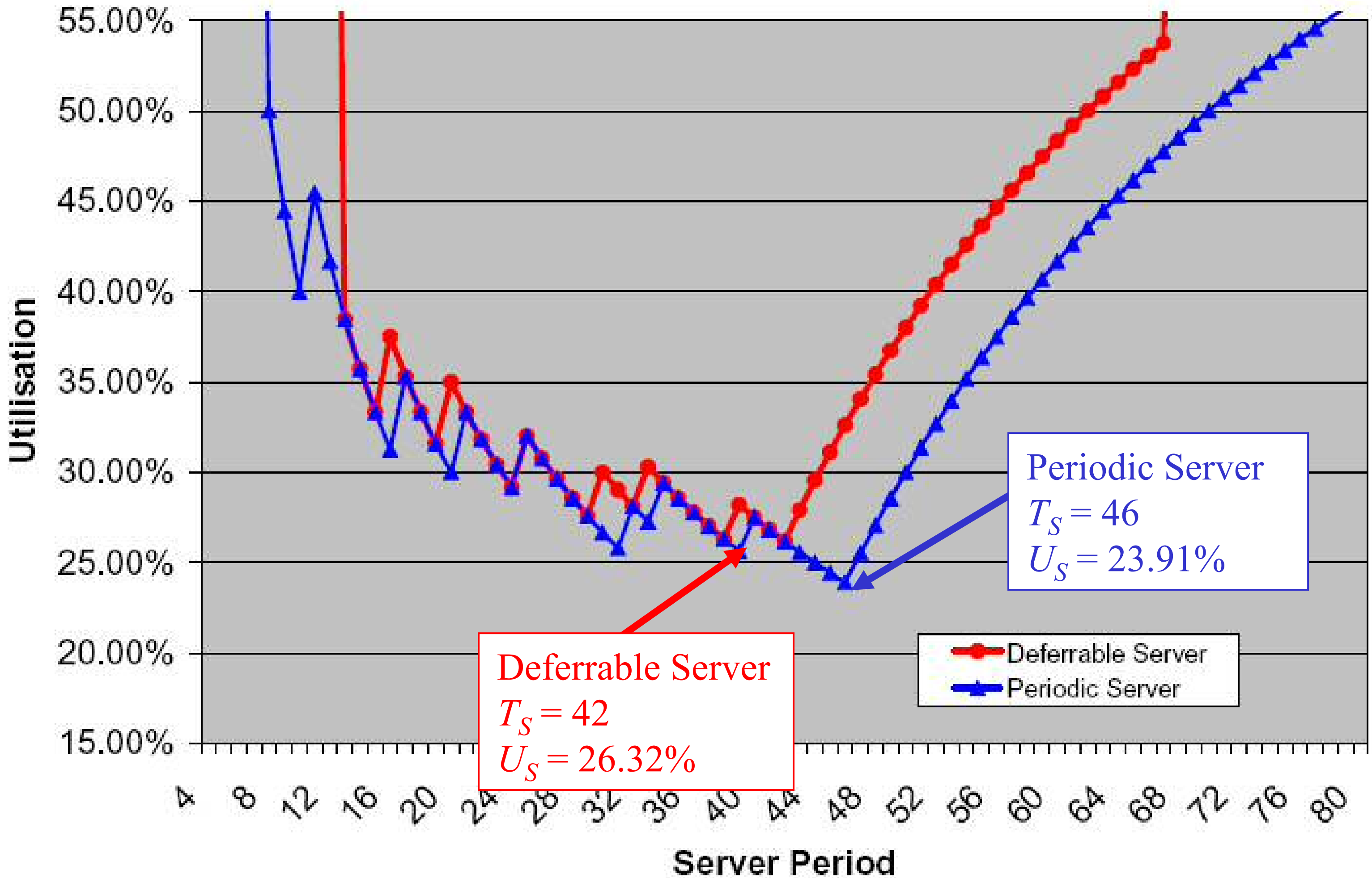
Server Overheads: Exact Analysis



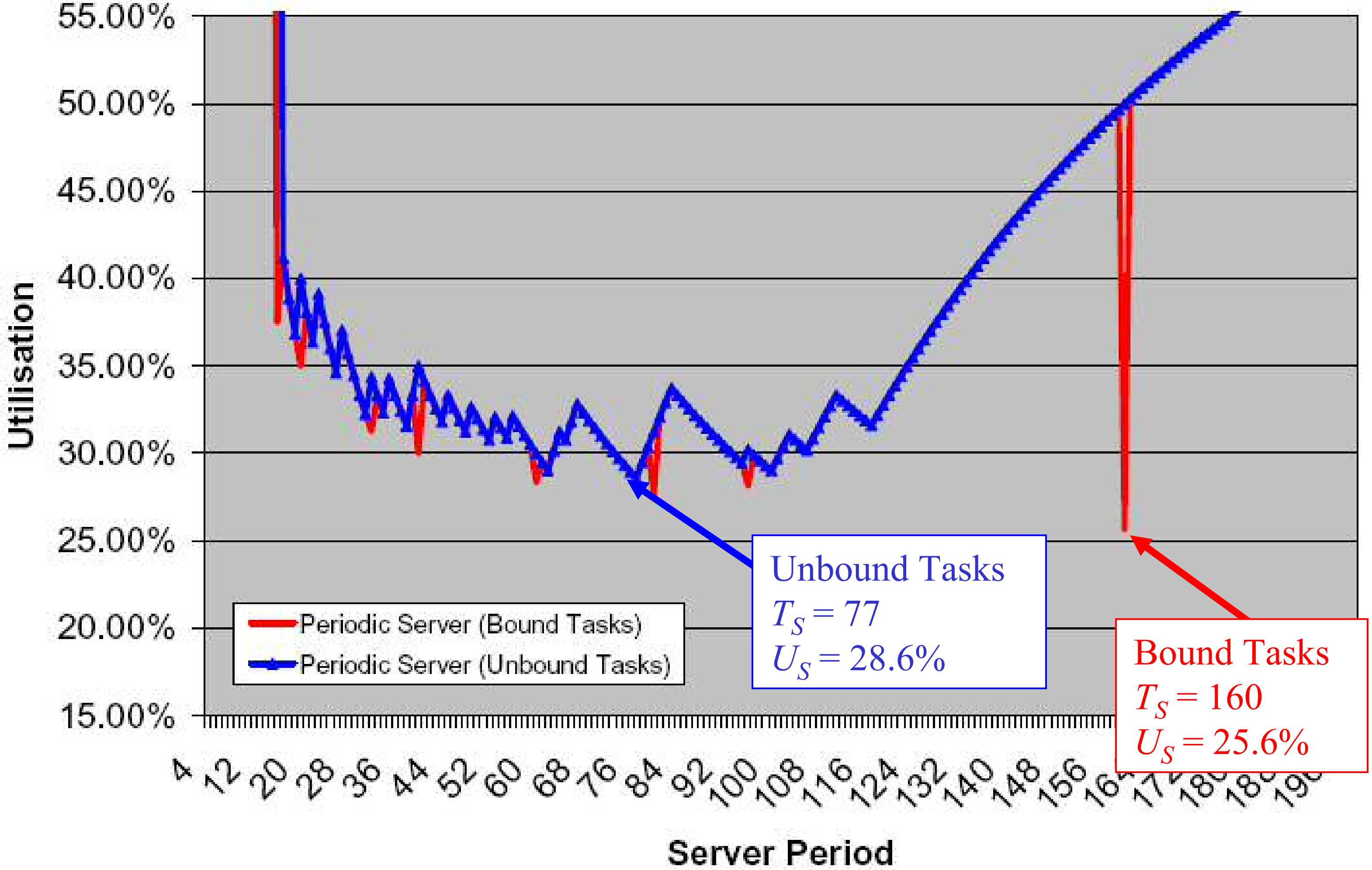
Comparison of Analysis Methods



Comparison of Server Algorithms



Bound v Unbound Tasks



Contribution

- Exact Response Time Analysis
  - For hierarchical fixed priority pre-emptive scheduling
  - Hard deadline tasks scheduled under Periodic, Deferrable and Sporadic Servers
    - Reduces computed worst-case response times w.r.t. previous work.
    - Improves minimum server utilisation required for systems to be deemed schedulable

Contribution (continued)

- Analysis extended to "bound" and "unbound" tasks
  - Binding tasks
    - reduces worst-case response times
    - Reduces minimum server utilisation required
    - influences optimal server period
- Comparison of Server Algorithms
  - Metric is ability to guarantee deadlines of hard real-time tasks (not aperiodic responsiveness!)
  - Simple Periodic Server <u>completely</u> dominates Deferrable and Sporadic Server algorithms on this metric

Technical Report

- Robert Davis, Alan Burns, *"Hierarchical Fixed Priority Pre-emptive Scheduling"* Department of Computer Science Technical Report YCS385, University of York, April 2005

- Report also includes
  - Extending exact schedulability analysis to include blocking due to global and local resource access.
  - Research into server parameter selection algorithms (choosing server priority, period and capacity)