

# Resource Sharing in Hierarchical Fixed-Priority Pre-emptive Systems

A decorative graphic on the left side of the slide, consisting of a vertical black line intersecting a horizontal black line. To the left of the vertical line are three overlapping squares: a blue one at the top, a red one in the middle, and a yellow one at the bottom.

Robert Davis and Alan Burns  
Real-Time Systems Research Group  
University of York

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Roadmap

---

- Motivation
- Hierarchical Scheduling Problem
  - System model
  - Schedulability analysis for independent applications
- Resource Access Policies
  - SRP and HSRP
  - Schedulability analysis with shared resources
  - Example
- Conclusions

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# 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
  - Requirements:
    - Temporal isolation: applications must behave as if they were running on individual microprocessors
    - Access to shared resources under mutual exclusion  
Examples: memory mapped peripherals, FLASH memory, data structures etc.

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# System Model

---

- Multiple applications on a single processor
  - Each application comprises multiple tasks
    - Task parameters: Priority, period ( $T_i$ ), deadline ( $D_i$ ), execution time ( $C_i$ ), Release jitter ( $J_i$ )
    - Worst-Case Response Time ( $R_i$ )
    - Assume  $D_i \leq T_i$
  - A Periodic Server is used to schedule each application
    - Server parameters: Priority, period ( $T_s$ ), capacity ( $C_s$ )
    - Tasks executed until the server's capacity is exhausted, then suspended until capacity replenished at next period
    - If no tasks ready then capacity assumed to be idled away (e.g. by an idle task carrying out BIT, memory checks etc.)

A decorative graphic on the left side of the slide, consisting of overlapping yellow, red, and blue squares with a black crosshair.

# System Model

---

- Fixed Priority Pre-emptive Scheduling
  - Global scheduling of servers
  - Local scheduling of tasks within a server

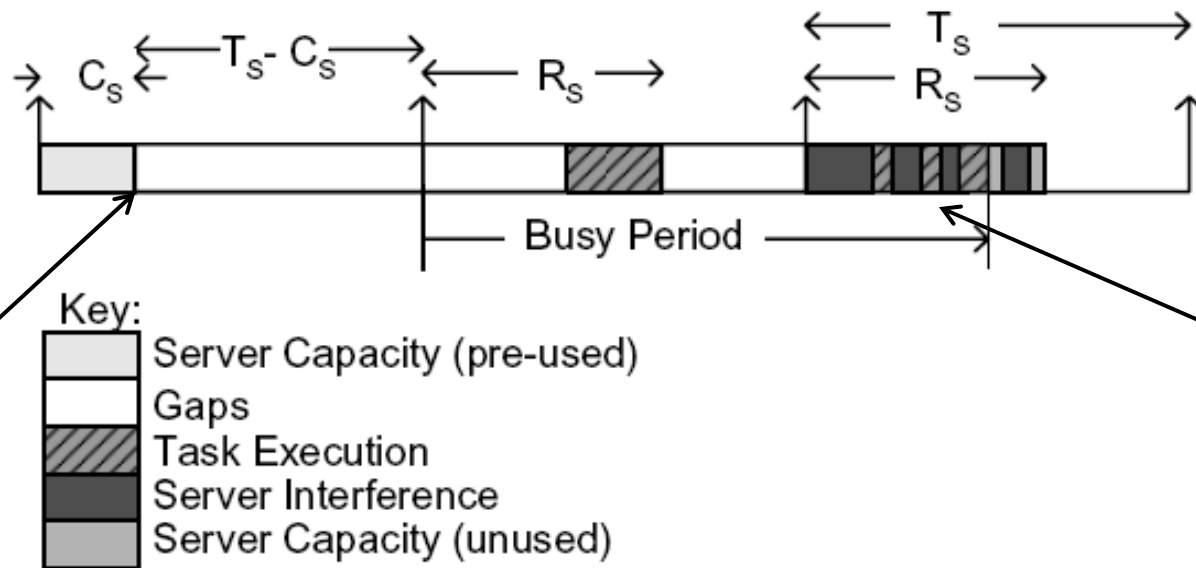
A decorative graphic on the left side of the slide consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Schedulability Analysis

---

- Using Response Time Analysis:
  - Determine worst-case scenario (critical instant) leading to worst-case response time for a task
  - Calculate busy period and hence worst-case response time given critical instant arrival pattern
  - Compare worst-case response time with task deadline

# Critical Instant



1. Server capacity exhausted  
2. Tasks arrive

3. Server capacity available as late as possible

1. Server capacity exhausted as early as possible then...
2. Task of interest and all higher priority tasks arrive just after server capacity exhausted
3. Server capacity available as late as possible due to interference from higher priority servers

# Busy period ( $w_i$ )

- Three components:

1. Task load released during the busy period

$$L_i(w_i) = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_j$$

Note,  $J_j$  is the task jitter which is increased by  $(T_S - C_S)$  due to the operation of the server

2. Gaps in complete server periods

$$\left( \left\lceil \frac{L_i(w_i)}{C_S} \right\rceil - 1 \right) (T_S - C_S)$$



A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Busy period ( $w$ )

---

3. Interference from higher priority servers in the final server period that completes task execution

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

# Response Time Computation

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

- 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

A decorative graphic on the left side of the slide consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Resource Access Policies

---

- Local Resources
  - Shared by tasks in a single application
  - Stack Resource Policy [T.P. Baker 1991]
- Global Resources
  - Shared by tasks in multiple applications
  - Hierarchical Stack Resource Policy – introduced here
    - Based on and compatible with SRP

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Local resources

---

- Stack Resource Policy
  1. Each local resource has a local ceiling priority equal to the highest priority of any task that accesses the resource
  2. Whilst a task accesses a local resource, its priority is increased to the local ceiling priority of the resource
  3. If the server's capacity is exhausted whilst a task is accessing a local resource, then execution of the task is simply suspended until the server's capacity is replenished

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Global resources

---

- Hierarchical Stack Resource Policy
  1. Each global resource has a global ceiling priority equal to the highest priority of any server that executes a task that accesses the resource
  2. Whilst a task accesses a global resource, the priority of its server is increased to the global ceiling priority of the resource
  3. Whilst a task accesses a global resource, the priority of the task is increased to the highest local priority within its application
  4. If the server's capacity is exhausted whilst a task is accessing a global resource, then the server continues to execute the task until the resource access is completed
  5. (Optionally) if a server overruns, then the capacity allocated at the start of its next period is reduced by the amount of the overrun

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Blocking Factors

---

- Definitions:

$B_{SO}$  longest time for which a task in server  $S$  can access a global resource. (Overrun time for server  $S$ )

$B_S$  longest time for which a task in a server of lower priority than  $S$  can access a global resource with a ceiling priority equal to or higher than  $S$ . (Blocking time for server  $S$ ).

$B_i$  longest time for which a task in the same application and of lower priority than task  $\tau_i$  can access either a global resource or a local resource with a ceiling priority equal to or higher than  $\tau_i$ . (Blocking time for task  $\tau_i$ ).

# Server Schedulability

- Worst-case scenario for server  $S$ 
  - Blocked by a lower priority server for  $B_S$
  - Additional interference due to overruns of higher priority servers

- With overrun & payback:

$$w_S^{n+1} = C_S + B_S + \sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} B_{XO} + \sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} \left\lceil \frac{w_S^n}{T_X} \right\rceil C_X$$

- Don't need to account for overrun of  $S$  in analysis of  $S$
- Overrun in one period leads to reduction in capacity replenished in next period
- Server 'execution time' in next period due to overrun + replenished capacity cannot exceed server capacity

A decorative graphic on the left side of the slide consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Server Schedulability

---

- With overrun & no payback:
  - Must account for overrun of server  $S$

$$w_S^{n+1} = C_S + B_{SO} + B_S + \sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} \left\lceil \frac{w_S^n}{T_X} \right\rceil (C_X + B_{XO})$$

- Server schedulable if its capacity can be fully consumed within its period



# Task Schedulability

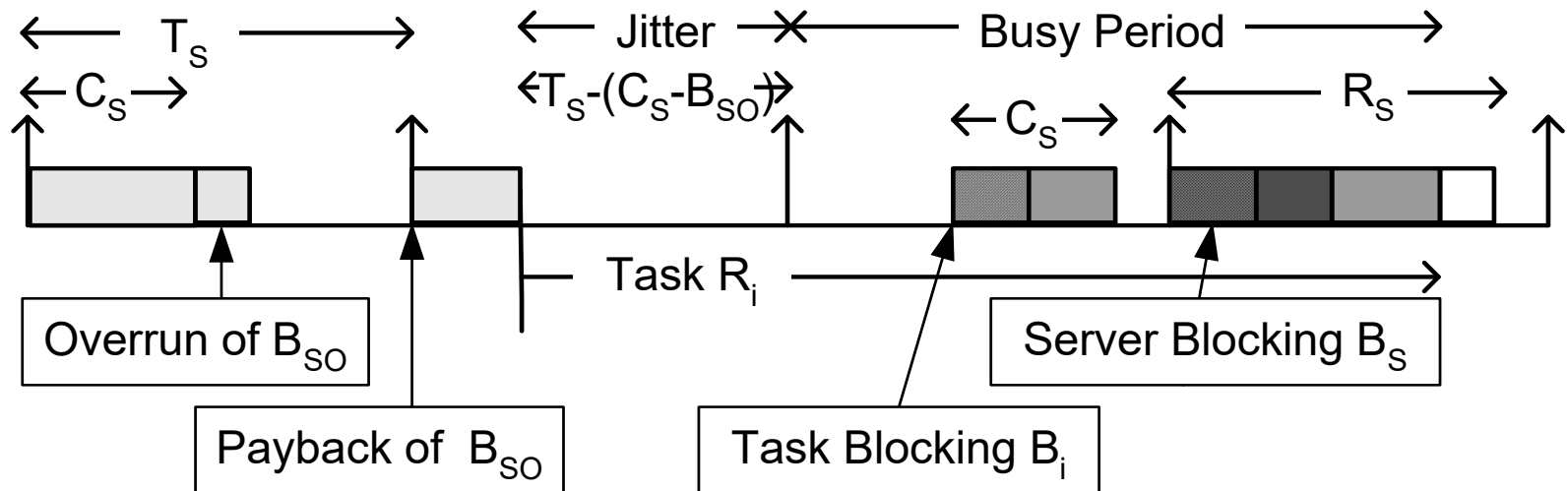
- Depends on two factors
  - Worst-case load to be executed during the busy period
  - Worst-case time the server takes to execute this load
- Task Load:
  - SRP and HSRP serialise access to resources
  - Maximum blocking of task  $\tau_i$  by lower priority tasks is  $B_i$

$$L_i(w_i) = B_i + C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_j$$

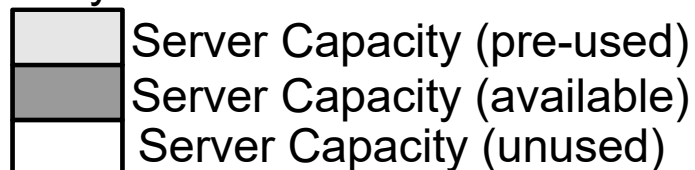
- Task jitter increased by:
  - $T_S - (C_S - B_{SO})$  with payback mechanism
  - $T_S - C_S$  without payback mechanism

# Task Schedulability

- Worst-case scenario for server to execute task load



Key:



# Task Schedulability

- Response Time Computation (overrun & payback)

$$w_i^{n+1} = L_i(w_i^n) + \left( \left\lceil \frac{L_i(w_i^n)}{C_S} \right\rceil - 1 \right) (T_S - C_S) + B_S +$$

$$\sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} B_{XO} + \sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} \left[ \frac{\max \left( 0, w_i^n - \left( \left\lceil \frac{L_i(w_i^n)}{C_S} \right\rceil - 1 \right) T_S \right)}{T_X} \right] C_X$$

- Re-compute task load  $L_i(w)$  each iteration
- Task jitter increased by  $T_S - (C_S - B_{SO})$  due to operation of the server
- Response time is  $w_i^n + J_i$

# Task Schedulability

- Response Time Computation (overrun & no payback)

$$w_i^{n+1} = L_i(w_i^n) + \left( \left\lceil \frac{L_i(w_i^n)}{C_S} \right\rceil - 1 \right) (T_S - C_S) + B_S +$$

$$\sum_{\substack{\forall X \in hp(S) \\ \text{servers}}} \left[ \frac{\max \left( 0, w_i^n - \left( \left\lceil \frac{L_i(w_i^n)}{C_S} \right\rceil - 1 \right) T_S \right)}{T_X} \right] (C_X + B_{XO})$$

- Re-compute task load  $L_i(w)$  each iteration
- Task jitter increased by  $T_S - C_S$  due to operation of the server
- Response time is  $w_i^n + J_i$

# Example

- Server parameters:

Server	Period	Capacity	$T - C$	$U$
$S_A$	2000	500	1500	25%
$S_B$	10000	2500	7500	25%
$S_C$	20000	5000	15000	25%

- Server response times:

Server	No Resources	HSRP No payback*	HSRP payback
$S_A$	500	1200	850
$S_B$	3500	5750	4700
$S_C$	10000	19550	14700

Global resource shared between all 3 applications, access time 350

\*includes overrun of the server

# Example (continued)

- Task parameters:

Task	$T$	$D$	$C$	$U$
$\tau_1$	25000	25000	2300	9.6%
$\tau_2$	50000	50000	4800	9.2%
$\tau_3$	100000	100000	2400	2.4%

Tasks for application  $B$

All tasks in application  $B$  access a local resource for 500 and a global resource for 350

- Task response times:

Task	No Resources	HSRP No payback	HSRP payback
$\tau_1$	10800	19000	19350
$\tau_2$	40400	42800	42450
$\tau_3$	89200	90750	90750

- Payback mechanism can result in task response times being larger or smaller (Note, with payback, could make server capacity larger)

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Alternative methods #1

---

- “Non-pre-emptive” resource access
  - Special case of Hierarchical Stack Resource Policy (HSRP)
  - Global ceiling priority of all resources set to the highest priority of *any* server
  - Can be analysed using analysis for HSRP (overrun & no payback)
  - HSRP dominates non-pre-emptive approach for both:
    - Server schedulability
    - Task schedulability
  - Non-pre-emptive approach useful if:
    - All global resource accesses are very short
    - Tasks in all applications share the same global resources

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Alternative methods #2

---

- “Prevent and pass-on”
  - Uses ceiling priorities as per HSRP
  - When resource access required:
    - First check if sufficient server capacity remains
    - If not, then suspend server until next replenishment
    - Any capacity remaining when server suspended is available in the next server period
  - Schedulability
    - Tasks: similar to ‘overrun & payback’ model
    - Servers: worse than ‘overrun & payback’
      - Due to need to accommodate additional preserved capacity in the server period



A decorative graphic on the left side of the slide consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Alternative methods #3

---

- “Suspend & use next server’s capacity”
  - Uses ceiling priorities as per HSRP
  - When resource locked and server capacity exhausted
    - Suspend server
    - If a task in another server needs the resource, then complete resource access using that server’s capacity
  - Schedulability
    - Each pre-empting server may result in a reduction in available capacity due to the need to subsequently unlock a resource
    - Double reduction in schedulability:
      - resource unlocking for other applications
      - Extra interference due to increased capacity of higher priority servers needed for resource unlocking
  - Implementation issues
    - Next server could also run out of capacity whilst unlocking a resource on behalf of another server and so on

A decorative graphic on the left side of the slide consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Recommendations

---

- In hierarchical fixed priority pre-emptive systems, global resources accesses have a large cumulative effect on schedulability
  - Important to make resource access times as short as possible
- Hierarchical Stack Resource Policy (HSRP)
  - An effective and analysable method of handling global resource access
- Payback mechanism?
  - Improves server schedulability which may permit larger server capacities
  - May or may not improve task response times
    - depends on system parameters
    - But larger server capacities also improve task schedulability

A decorative graphic on the left side of the slide, consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Contribution

---

- Motivation
  - Trend towards multiple applications on a single processor in both Automotive Electronics and Avionics
  - Real-world applications share resources both globally and locally: memory mapped peripherals, data buffers, shared comms devices etc.
- Contribution
  - Definition of **HSRP**, an appropriate resource locking protocol for hierarchical fixed priority pre-emptive systems based on priority ceilings and the **SRP**.
  - Schedulability analysis for HSRP.

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair.

# Conclusions

---

- Techniques and analysis now available to design and develop hierarchical, multiple application, real-world systems using fixed priority pre-emptive scheduling
  
- Areas of Future Work
  - Choice of Server parameters ( $T$  and  $C$ )
  - Policies for resource access that avoid server overruns
  
- Acknowledgements
  - Research partially funded by:
    - EPSRC DIRC project
    - EU *frescor* project