Guaranteeing Real-Time Performance Using Rate Monotonic Analysis

Embedded Systems Conference
September 20-23, 1994

Presented by

Ray Obenza

Software Engineering Institute
Carnegie Mellon University
Pittsburgh PA 15213

Sponsored by the U.S. Department of Defense

Rate Monotonic Analysis

Introduction

Periodic tasks

Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer

Purpose of Tutorial

Introduce rate monotonic analysis

Explain how to perform the analysis

Give some examples of usage

Convince you it is useful

Tutorial Format

Lecture

Group exercises

Case study

Questions welcome anytime

RMARTS Project

Originally called Real-Time Scheduling in Ada Project (RTSIA).

- **focused on rate monotonic scheduling theory**
- **recognized strength of theory was in analysis**

Rate Monotonic Analysis for Real-Time Systems (RMARTS)

- **focused on analysis supported by (RMS) theory**
- **analysis of designs regardless of language or scheduling approach used**

Project focused initially on uniprocessor systems.

Work continues in distributed processing systems.

Real-Time Systems

Timing requirements

- meeting deadlines

Periodic and aperiodic tasks

Shared resources

Interrupts

What's Important in Real-Time

Criteria for real-time systems differ from that for time-sharing systems.

	Time-Sharing Systems	Real-Time Systems
Capacity	High throughput	Schedulability
Responsiveness	Fast average response	Ensured worst-case latency
Overload	Fairness	Stability

- ***schedulability*** **is the ability of tasks to meet all hard deadlines**
- ***latency*** **is the worst-case system response time to events**
- ***stability*** **in overload means the system meets critical deadlines even if all deadlines cannot be met**

Scheduling Policies

CPU scheduling policy: a rule to select task to run next

- **cyclic executive**
- **rate monotonic/deadline monotonic**
- **earliest deadline first**
- **least laxity first**

Assume preemptive, priority scheduling of tasks

- **analyze effects of non-preemption later**

Rate Monotonic Scheduling (RMS)

Priorities of periodic tasks are based on their rates: highest rate gets highest priority.

Theoretical basis

- **optimal fixed scheduling policy (when deadlines are at end of period)**
- **analytic formulas to check schedulability**

Must distinguish between scheduling and analysis

- **rate monotonic scheduling forms the basis for rate monotonic analysis**
- **however, we consider later how to analyze systems in which rate monotonic scheduling is not used**
- **any scheduling approach may be used, but all real-time systems should be analyzed for timing**

Rate Monotonic Analysis (RMA)

Rate monotonic analysis is a method for analyzing sets of real-time tasks.

Basic theory applies only to independent, periodic tasks, but has been extended to address

- **priority inversion**
- **task interactions**
- **aperiodic tasks**

Focus is on RMA, not RMS.

Why Are Deadlines Missed?

For a given task, consider

- ***preemption*: time waiting for higher priority tasks**
- ***execution*: time to do its own work**
- ***blocking*: time delayed by lower priority tasks**

The task is *schedulable* if the sum of its preemption, execution, and blocking is less than its deadline.

Focus: identify the biggest hits among the three and reduce, as needed, to achieve schedulability

Rate Monotonic Theory - Experience

IBM Systems Integration Division delivered a “schedulable” real-time network.

Theory used successfully to improve performance of IBM BSY-1 Trainer.

Incorporated into IEEE FutureBus+ standard

Adopted by NASA Space Station Program

European Space Agency requires as baseline theory.

Supported in part by Ada vendors

Rate Monotonic Analysis - Products

Journal articles (e.g., *IEEE Computer*, Hot Topics)

Videotape from SEI

Courses from Telos and Tri-Pacific

***A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems* from Kluwer**

CASE tools from Introspect and Tri-Pacific

Operating systems and runtimes from Alsys, DDC-I, Lynx, Sun, Verdix and Wind River

Standards: Futurebus+, POSIX, Ada 9X

Summary

Real-time goals are: fast response, guaranteed deadlines, and stability in overload.

Any scheduling approach may be used, but all real-time systems should be analyzed for timing.

Rate monotonic analysis

- **based on rate monotonic scheduling theory**
- **analytic formulas to determine schedulability**
- **framework for reasoning about system timing behavior**
- **separation of timing and functional concerns**

Provides an engineering basis for designing real-time systems

Plan for Tutorial

Present basic theory for periodic task sets

Extend basic theory to include

- context switch overhead
- preperiod deadlines
- interrupts

Consider task interactions:

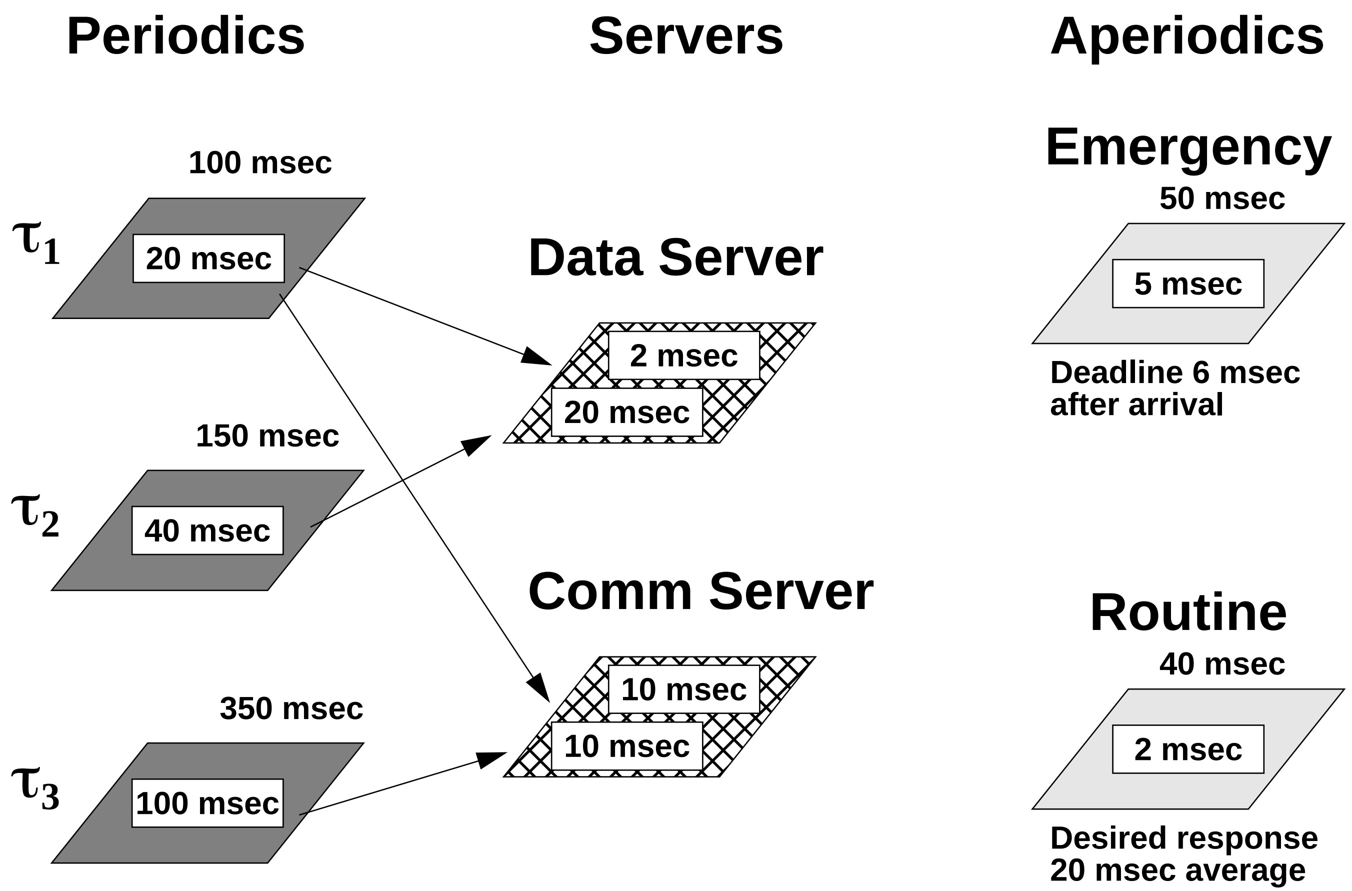
- priority inversion
- synchronization protocols (time allowing)

Extend theory to aperiodic tasks:

- sporadic servers (time allowing)

Present BSY-1 Trainer case study

A Sample Problem



τ_2's deadline is 20 msec before the end of each period

Rate Monotonic Analysis

Introduction

Periodic tasks

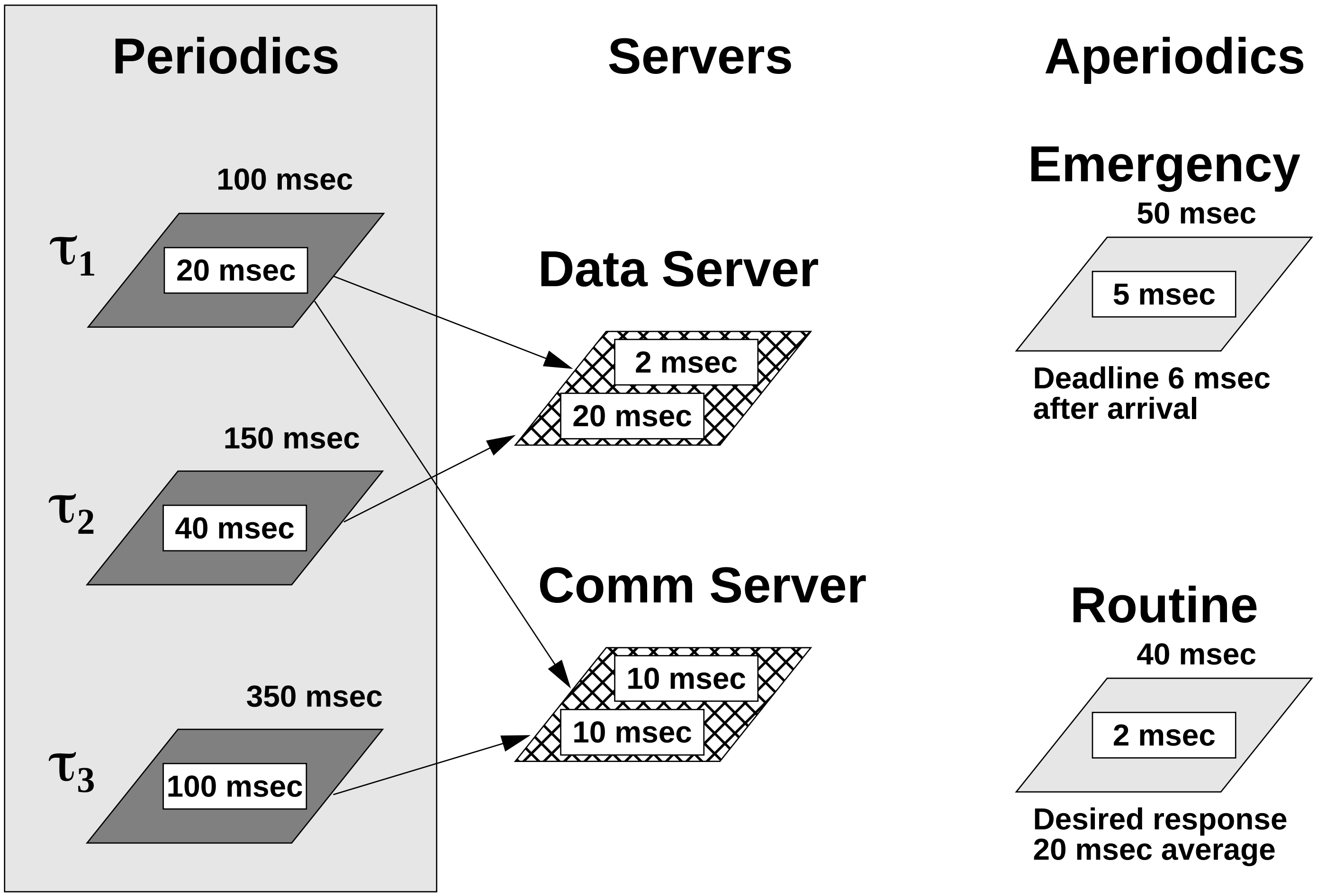
Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case Study: BSY-1 Trainer

A Sample Problem - Periodics



τ_2's deadline is 20 msec before the end of each period.

Concepts and Definitions - Periodics

Periodic task

- initiated at fixed intervals
- must finish before start of next cycle

Task's CPU utilization: $U_i = \frac{C_i}{T_i}$

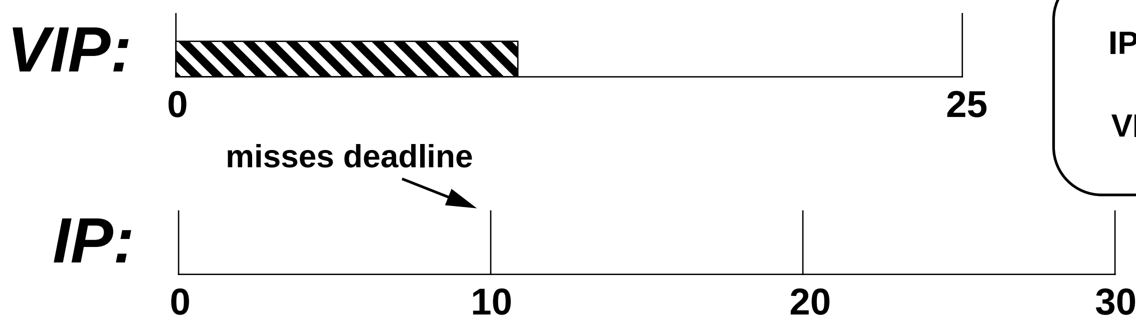
- C_i = compute time (execution time) for task τ_i
- T_i = period of task τ_i

CPU utilization for a set of tasks:

$$U = U_1 + U_2 + \ldots + U_n$$

Example of Priority Assignment

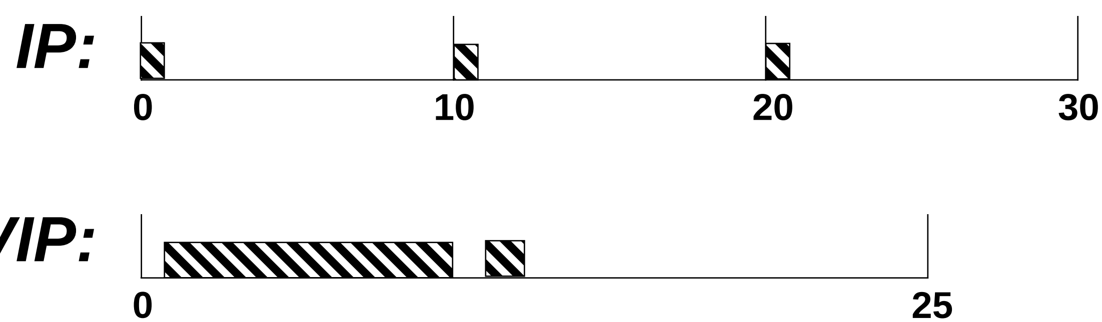
Semantic-Based Priority Assignment



IP: $U_{IP} = \frac{1}{10} = 0.10$

VIP: $U_{VIP} = \frac{11}{25} = 0.44$

Policy-Based Priority Assignment



Schedulability: UB Test

***Utilization bound(UB) test*: a set of *n* independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if**

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \le U(n) = n\left(2^{1/n} - 1\right)$$

U(1) = 1.0	***U(4) = 0.756***	***U(7) = 0.728***
U(2) = 0.828	***U(5) = 0.743***	***U(8) = 0.724***
U(3) = 0.779	***U(6) = 0.734***	***U(9) = 0.720***

For harmonic task sets, the utilization bound is *U(n)=1.00* for all *n*.

Note: UB test = Techniques 1 and 2 in handbook.

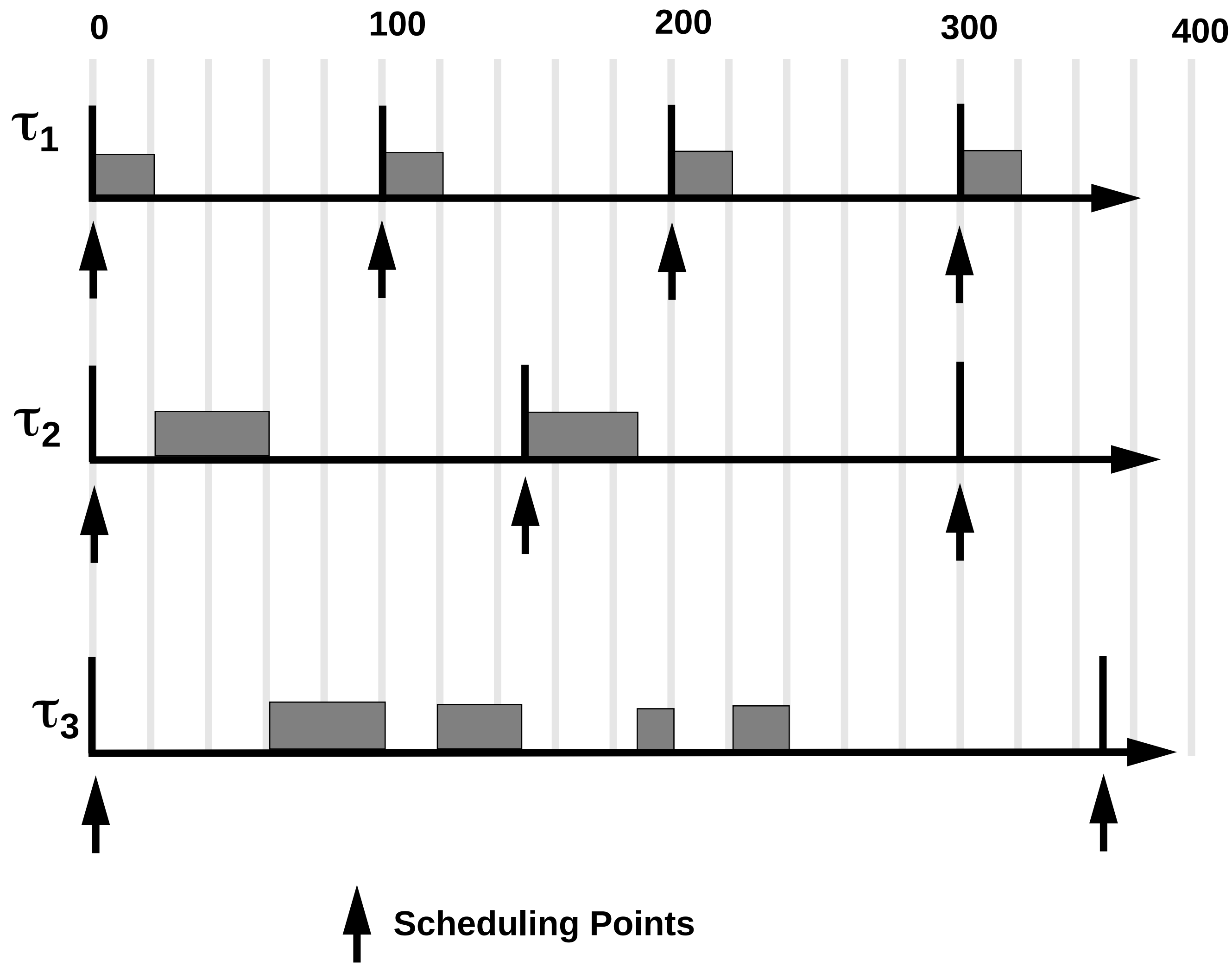
Sample Problem: Applying UB Test

	C	*T*	*U*
Task τ_1:	20	100	0.200
Task τ_2:	40	150	0.267
Task τ_3:	100	350	0.286

Total utilization is .200 + .267 + .286 = .753 < U(3) = .779

The periodic tasks in the sample problem are schedulable according to the UB test.

Timeline for Sample Problem



Exercise: Applying the UB Test

Given:

Task	*C*	*T*	*U*
τ_1	1	4	
τ_2	2	6	
τ_3	1	10	

a. What is total utilization?

b. Is the task set schedulable?

c. Draw the timeline.

d. What is the total utilization if $C_3 = 2$?

Toward a More Precise Test

UB test has three possible outcomes:

$$0 \le U \le U(n) \Rightarrow Success$$

$$U(n) < U \le 1.00 \Rightarrow Inconclusive$$

$$1.00 < U \Rightarrow Overload$$

UB test is conservative.

A more precise test can be applied.

Schedulability: RT Test

Theorem: for a set of independent, periodic tasks, if each task meets its first deadline, with worst-case task phasing, the deadline will always be met.

***Response time (RT) test*: let a_n = response time of task i. a_n may be computed by the following iterative formula:**

$$a_{n+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_n}{T_j} \right\rceil C_j \qquad \text{where } a_0 = \sum_{j=1}^{i} C_j$$

Test terminates when $a_{n+1} = a_n$.

Task i is schedulable if its response time is before its deadline: $a_n \leq T_i$

Example: Applying RT Test -1

Taking the sample problem, we increase the compute time of τ_1 from 20 to 40; is the task set still schedulable?

Utilization of first two tasks: 0.667 < U(2) = 0.828

- **first two tasks are schedulable by UB test**

Utilization of all three tasks: 0.953 > U(3) = 0.779

- **UB test is inconclusive**
- **need to apply RT test**

Example: Applying RT Test -2

Use RT test to determine if τ_3 meets its first deadline: i = 3

$$a_0 = \sum_{j=1}^{3} C_j = C_1 + C_2 + C_3 = 40 + 40 + 100 = 180$$

$${}_1 = \boldsymbol{C}_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_0}{T_j} \right\rceil C_j = \boldsymbol{C}_3 + \sum_{j=1}^{2} \left\lceil \frac{a_0}{T_j} \right\rceil C_j$$

$$= 100 + \left\lceil \frac{180}{100} \right\rceil (40) + \left\lceil \frac{180}{150} \right\rceil (40) = 100 + 80 + 80 = 260$$

Example: Applying the RT Test -3

$$= \boldsymbol{C}_3 + \sum_{j=1}^{2} \left\lceil \frac{a_1}{T_j} \right\rceil C_j = 100 + \left\lceil \frac{260}{100} \right\rceil (40) + \left\lceil \frac{260}{150} \right\rceil (40) = 30$$

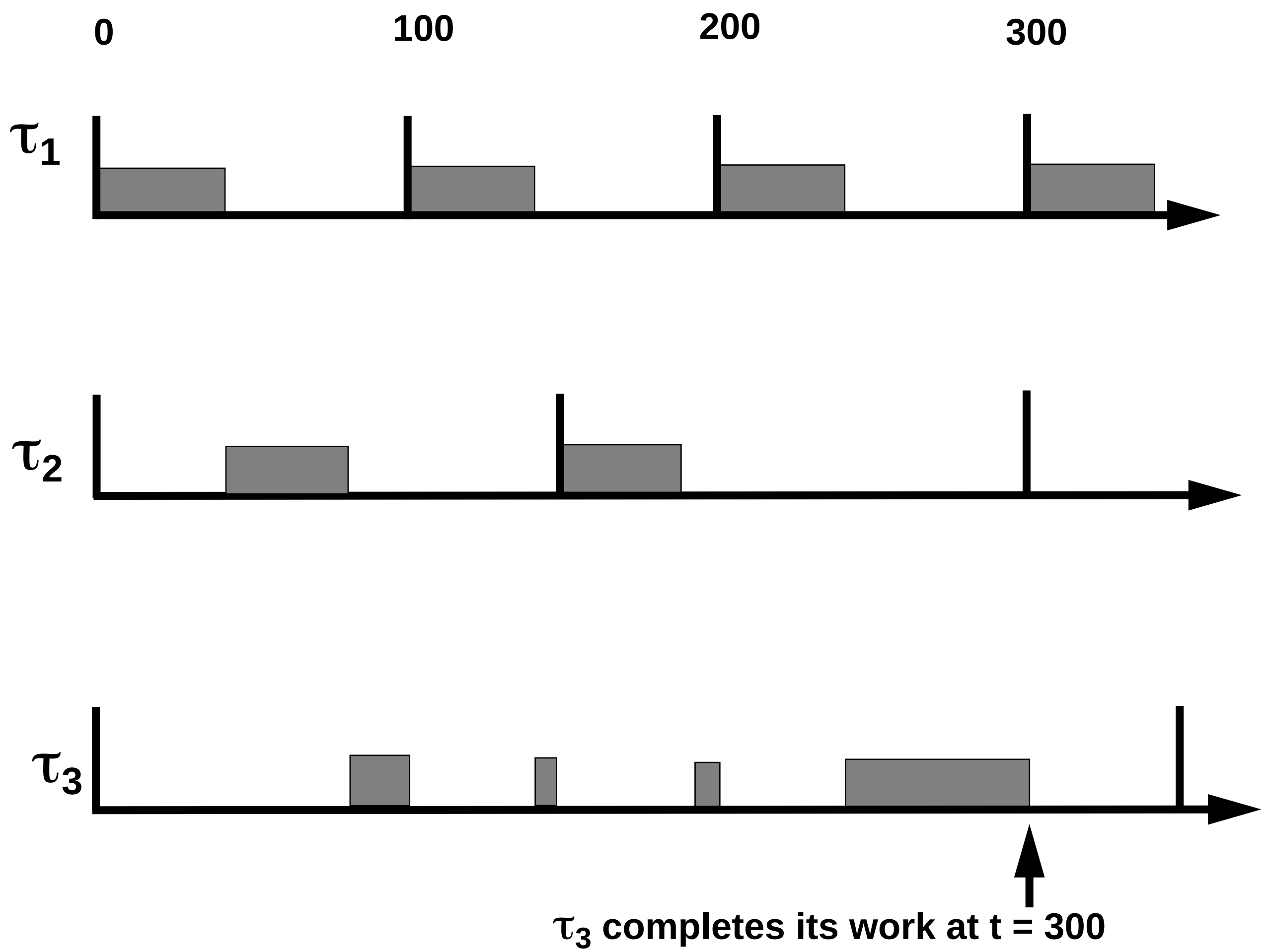
$$= \boldsymbol{C}_3 + \sum_{j=1}^{2} \left\lceil \frac{a_2}{T_j} \right\rceil C_j = 100 + \left\lceil \frac{300}{100} \right\rceil (40) + \left\lceil \frac{300}{150} \right\rceil (40) = 30$$

$$a_3 = a_2 = 300 \quad \text{Done!}$$

Task τ_3 is schedulable using RT test.

$$a_3 = 300 < T = 350$$

Timeline for Example



Exercise: Applying RT Test

Task τ_1: $C_1 = 1$ $T_1 = 4$

Task τ_2: $C_2 = 2$ $T_2 = 6$

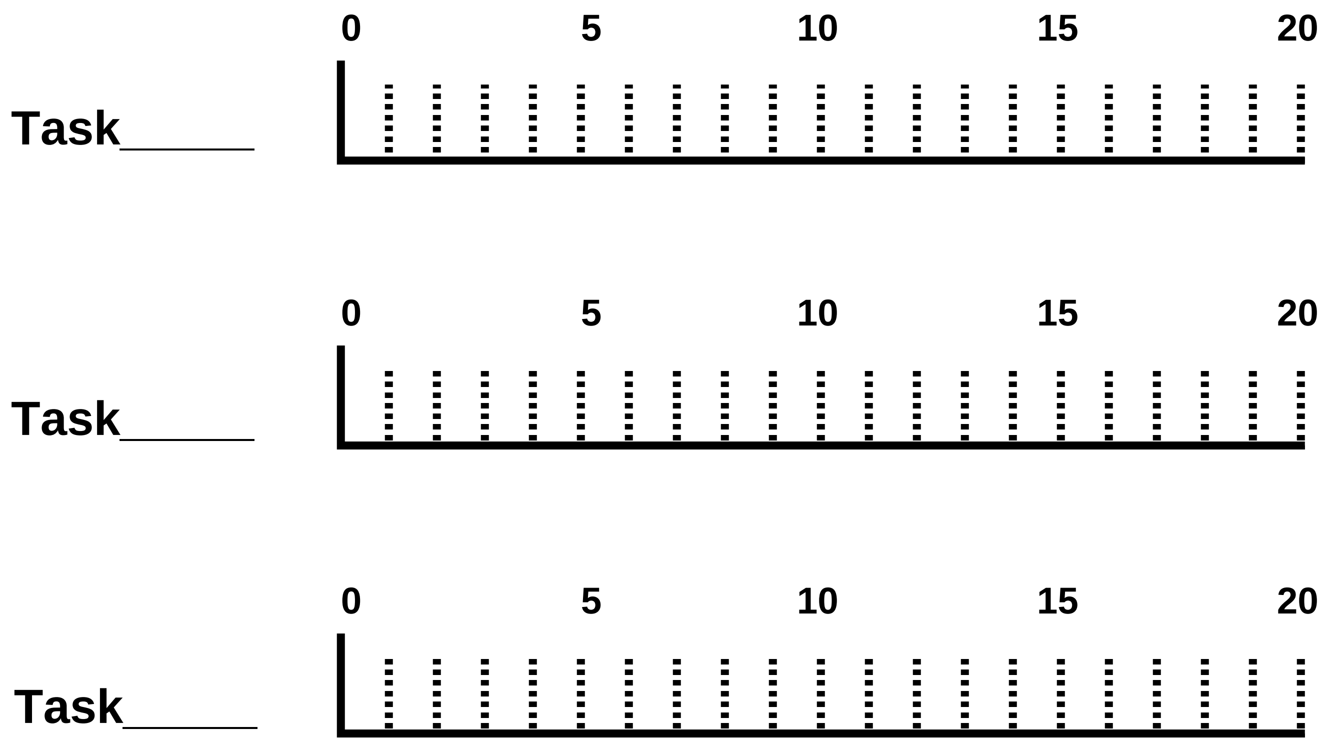
Task τ_3: $C_3 = 2$ $T_3 = 10$

a) Apply UB test

b) Draw timeline

c) Apply RT Test

Exercise: Worksheet



Summary

UB test is simple but conservative.

RT test is more exact but also more complicated.

To this point, UB and RT tests share the same limitations:

- **all tasks run on a single processor**
- **all tasks are periodic and noninteracting**
- **deadlines are always at the end of the period**
- **there are no interrupts**
- **rate monotonic priorities are assigned**
- **there is zero context switch overhead**
- **tasks do not suspend themselves**

Rate Monotonic Analysis

Introduction

Periodic tasks

Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer

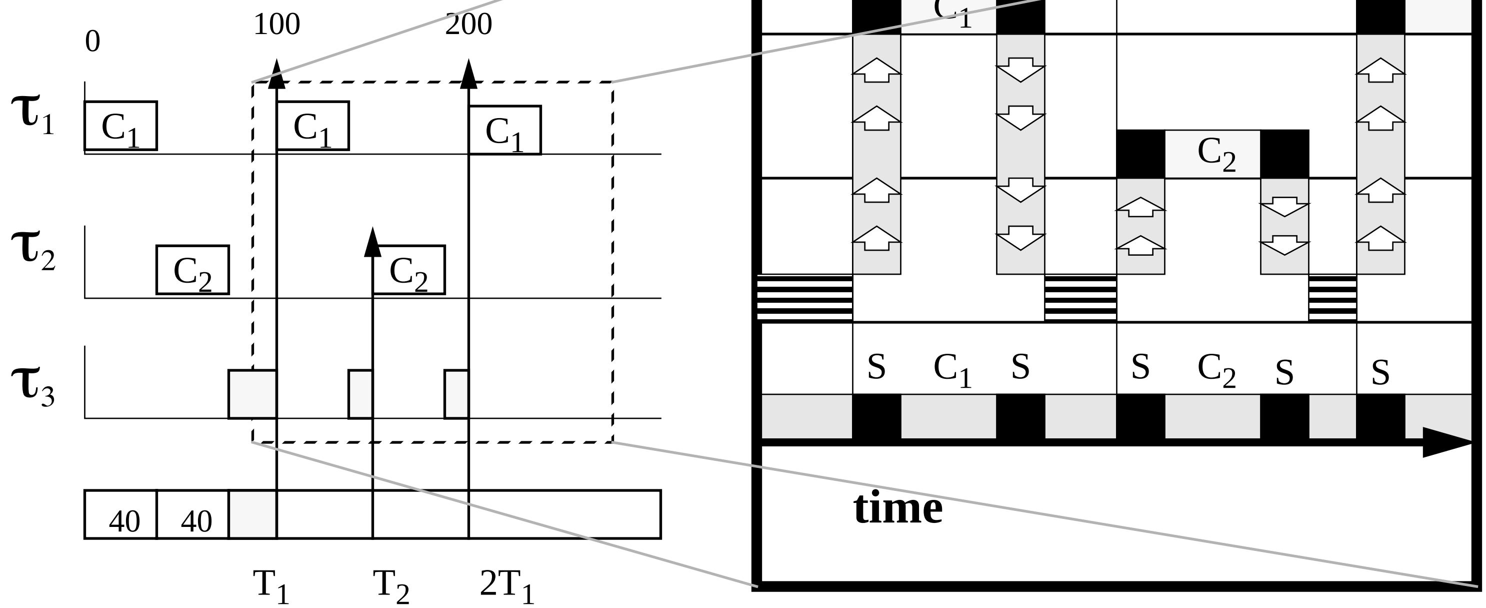
Extensions to Basic Theory

This section extends the schedulability tests to address

- **nonzero task switching times**
- **preperiod deadlines**
- **interrupts and non-rate-monotonic priorities**

Modeling Task Switching as Execution Time

$$U_i = \frac{C_i + 2S}{T_i}$$



Two scheduling actions per task
(start of period and end of period)

Modeling Preperiod Deadlines

Suppose task τ, with compute time *C* and period *T*, has a preperiod deadline *D* (i.e. *D* < *T*).

Compare total utilization to modified bound:

$$U_{total} = \frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \leq U(n, \Delta_i)$$

where Δ_i is the ratio of D_i to T_i .

$$U(n, \Delta_i) = \begin{pmatrix} n\left(\left(2\Delta_i\right)^{1/n} - 1\right) + 1 - \Delta_i, & \frac{1}{2} < \Delta_i \leq 1.0 \\ \Delta_i, \quad \Delta_i \leq \frac{1}{2} & \end{pmatrix}$$

Schedulability with Interrupts

Interrupt processing can be inconsistent with rate monotonic priority assignment.

- **interrupt handler executes with high priority despite its period**
- **interrupt processing may delay execution of tasks with shorter periods**

Effects of interrupt processing must be taken into account in schedulability model.

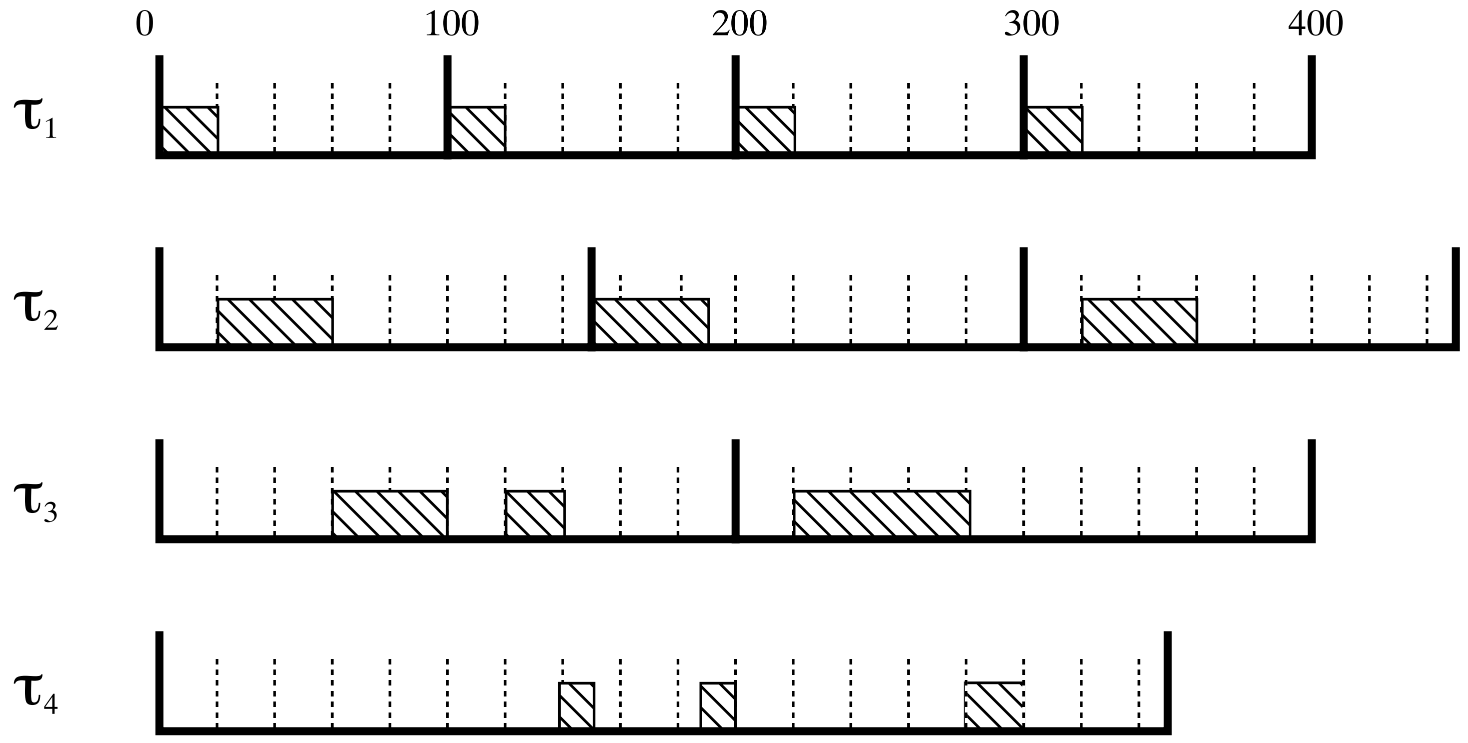
Question is: how to do that?

Example: Determining Schedulability with Interrupts

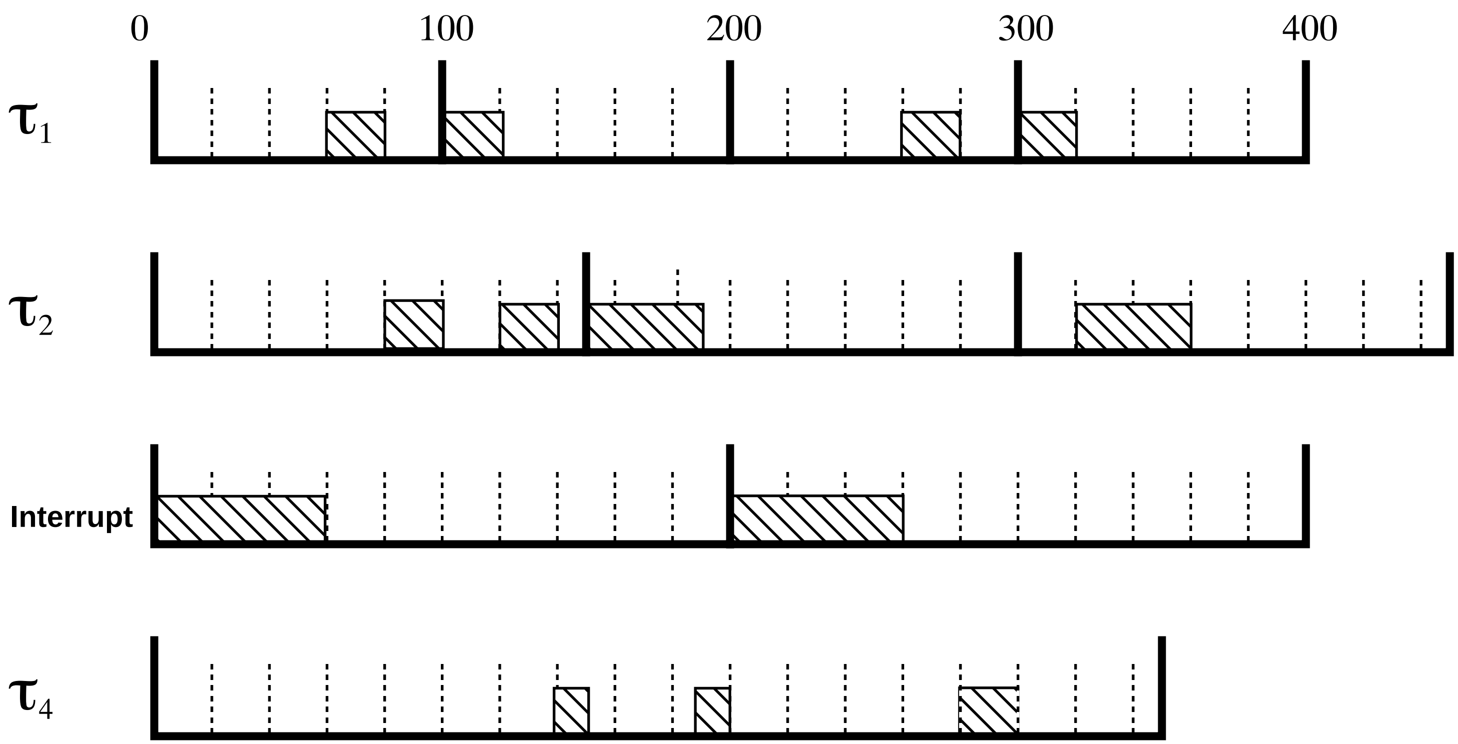
	C	*T*	*U*
Task τ_1:	**20**	**100**	**0.200**
Task τ_2:	**40**	**150**	**0.267**
Task τ_3:	**60**	**200**	**0.300**
Task τ_4:	**40**	**350**	**0.115**

τ_3 is an interrupt handler

Example: Execution with Rate Monotonic Priorities



Example: Execution with an Interrupt Priority



Resulting Table for Example

Task (i)	Period (T)	Execution Time (C)	Priority (P)	Deadline (D)
τ_3	200	60	HW	200
τ_1	100	20	High	100
τ_2	150	40	Medium	150
τ_4	350	40	Low	350

UB Test with Interrupt Priority

Test is applied to each task.

Determine effective utilization (f_i) of each task *i* using

$$f_i = \sum_{j \in Hn} \frac{C_j}{T_j} + \frac{C_i}{T_i} + \frac{1}{T_i} \sum_{k \in H1} C_k$$

Preemption from tasks that can hit more than once (with period less than D_i)

Execution of task under test

Preemption from tasks that can hit only once (with period greater than D_i)

Compare effective utilization against bound, U(n).

- **n = num(*Hn*) + 1**
- **num(*Hn*) = the number of tasks in the set *Hn***

UB Test with Interrupt Priority: τ_3

For τ_3, no tasks have a higher priority: *H* = *Hn* = *H1* = { }.

$$f_3 \quad = \quad 0 + \frac{C_3}{T_3} + 0 \quad \leq U(1)$$

Note that utilization bound is U(1): num(*Hn*) = 0.

Plugging in numbers:

$$f_3 = \frac{C_3}{T_3} = \frac{60}{200} = 0.3 < 1.0$$

UB Test with Interrupt Priority: τ_1

To τ_1, τ_3 has higher priority: *H* = {τ_3}; *Hn* = { }; *H1* = {τ_3}.

$$f_1 = 0 + \frac{C_1}{T_1} + \frac{1}{T_1}\sum_{k=3} C_k \le U(1)$$

Note that utilization bound is U(1): num(*Hn*) = 0.

Plugging in the numbers:

$$f_1 = \frac{C_1}{T_1} + \frac{C_3}{T_1} = \frac{20}{100} + \frac{60}{100} = 0.800 < 1.0$$

UB Test with Interrupt Priority: τ_2

To τ_2: *H* = {τ_1,τ_3}; *Hn* = {τ_1}; *H1* = {τ_3}.

$$f_2 = \sum_{j=1} \frac{C_j}{T_j} + \frac{C_2}{T_2} + \frac{1}{T_2}\sum_{k=3} C_k \quad \leq U(2)$$

Note that utilization bound is U(2): num(*Hn*) = 1.

Plugging in the numbers:

$$f_2 = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_2} = \frac{20}{100} + \frac{40}{150} + \frac{60}{150} = 0.867 \quad > 0.828$$

UB Test with Interrupt Priority: τ_4

To τ_2: *H* = {τ_1,τ_2,τ_3}; *Hn* = {τ_1,τ_2,τ_3}; *H1* = { }.

$$f_4 = \sum_{j = 1, 2, 3} \frac{C_j}{T_j} + \frac{C_4}{T_4} + 0 \quad \leq U(4)$$

Note that utilization bound is U(4): num(*Hn*) = 3.

Plugging in the numbers:

$$f_4 = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \frac{C_4}{T_4}$$

$$= \frac{20}{100} + \frac{40}{150} + \frac{60}{200} + \frac{40}{350} = 0.882 \quad > 0.756$$

Exercise: Schedulability with Interrupts

Given the following tasks:

Task (i)	Period (T)	Execution Time (C)	Priority (P)	Deadline (D)
τ_{int}	6	2	HW	6
τ_1	4	1	High	3
τ_2	10	1	Low	10

Use the UB test to determine which tasks are schedulable.

Solution: Schedulability with Interrupts

$$\frac{C_{int}}{T_{int}} \le U(1) \qquad 0.334 < 1.0$$

{H1}

$$\frac{C_1}{T_1} + \frac{C_{int}}{T_1} \le U(1, .75) \qquad 0.250 + 0.500 = 0.750 = U(1, .75)$$

{Hn}

$$\frac{C_{int}}{T_{int}} + \frac{C_1}{T_1} + \frac{C_2}{T_2} \le U(3)$$

$$0.334 + 0.250 + 0.100 = 0.684 < 0.779$$

Basic Theory: Where Are We?

We have shown how to handle

- **task context switching time: include 2*S* within *C***
- **preperiod deadlines: change bound to U(n, Δ_i)**
- **non-rate-montonic priority assignments**

We still must address

- **task interactions**
- **aperiodic tasks**

We still assume

- **single processor**
- **priority-based scheduling**
- **tasks do not suspend themselves**

Other Important Issues

Mode change

Multiprocessor systems

Priority granularity

Overload

Spare capacity assessment

Distributed systems

Post-period deadlines

Rate Monotonic Analysis

Introduction

Periodic tasks

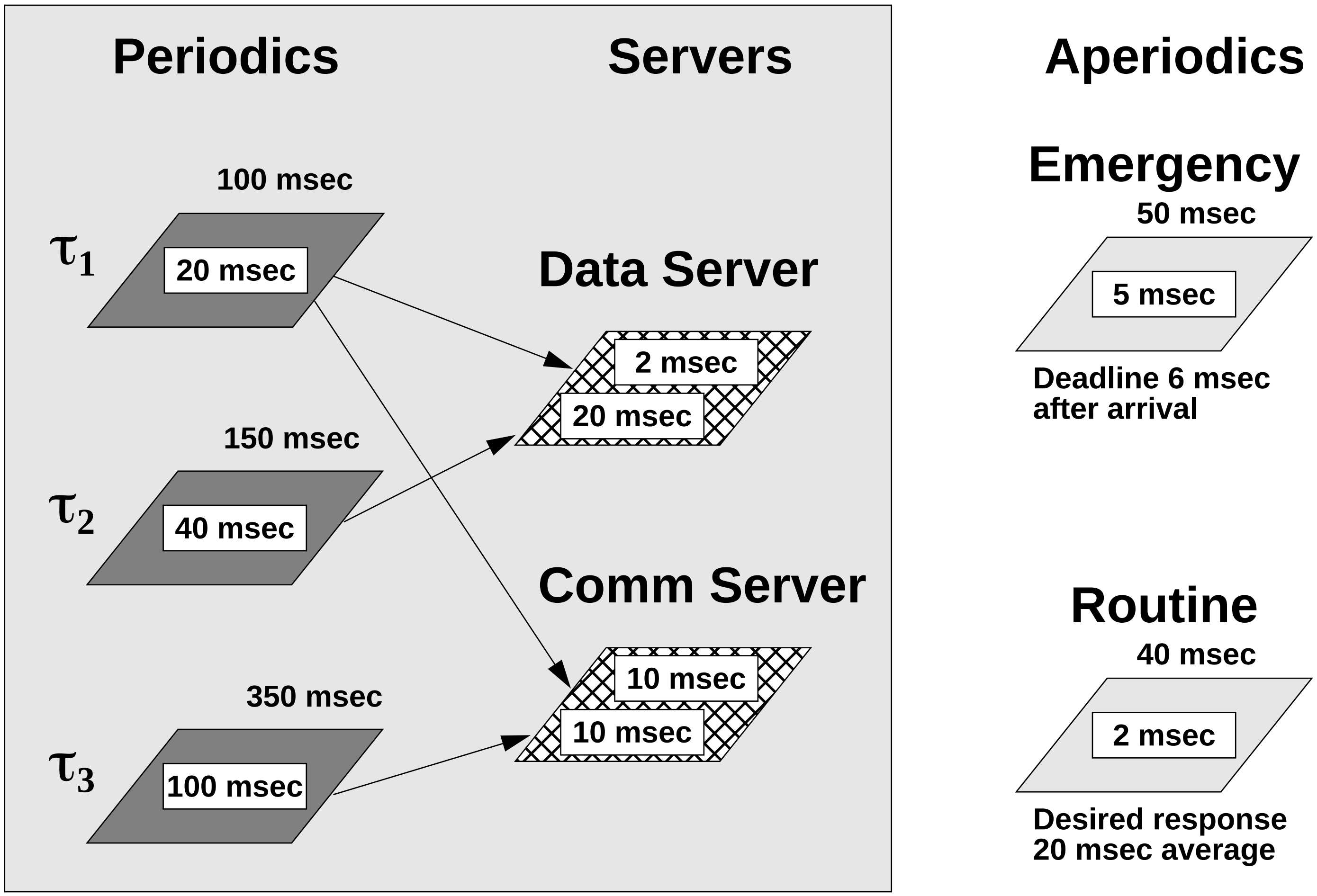
Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer

Sample Problem: Synchronization

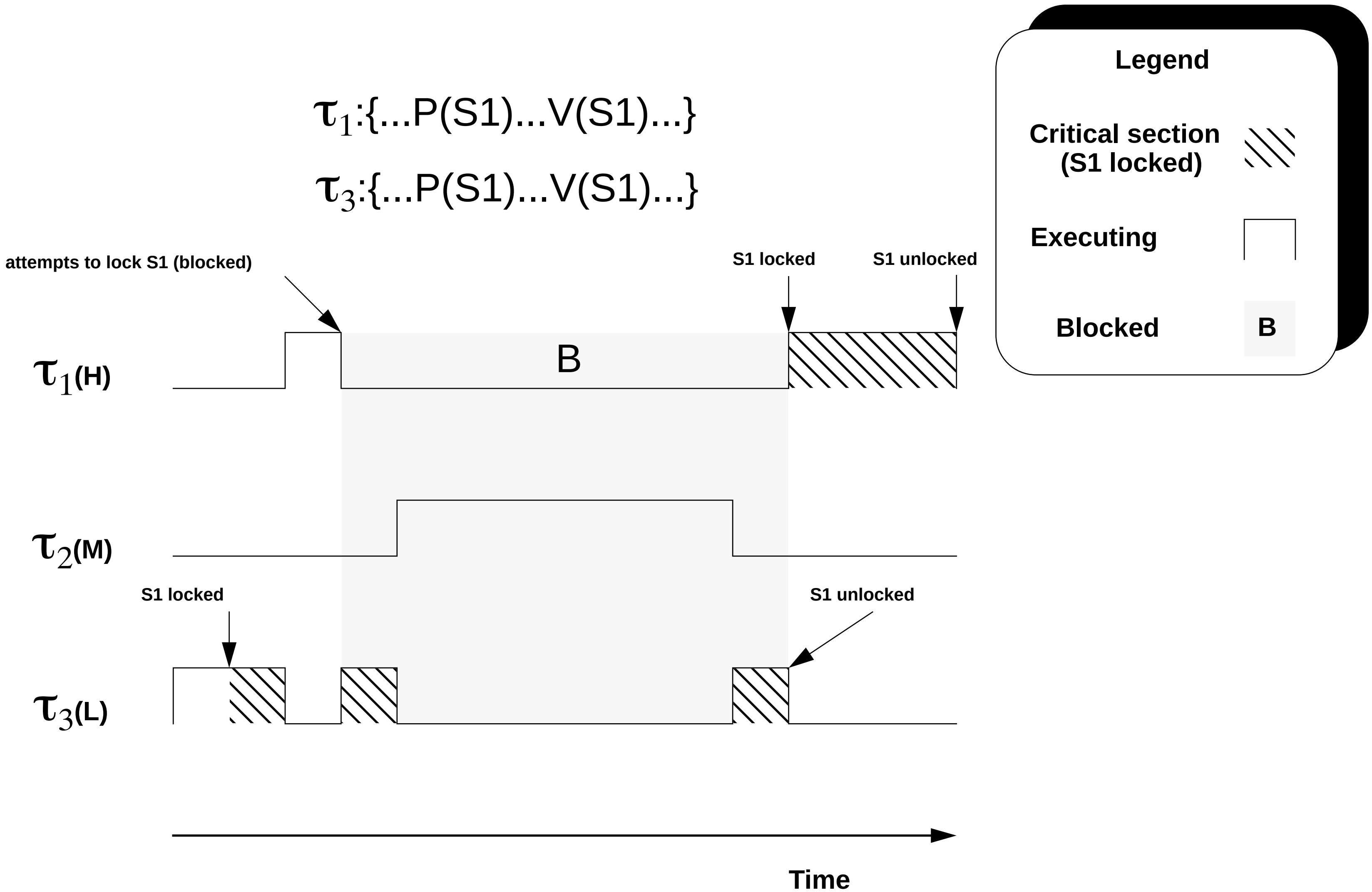


τ_2's deadline is 20 msec before the end of each period.

Priority Inversion in Synchronization

τ_1:{...P(S1)...V(S1)...}

τ_3:{...P(S1)...V(S1)...}



Priority Inversion

Delay to a task's execution caused by interference from lower priority tasks is known as *priority inversion*.

Priority inversion is modeled by blocking time.

Identifying and evaluating the effect of sources of priority inversion is important in schedulability analysis.

Sources of Priority Inversion

Synchronization and mutual exclusion

Non-preemptable regions of code

FIFO (first-in-first-out) queues

Accounting for Priority Inversion

Recall that task schedulability is affected by

- **preemption: two types of preemption**
  - **can occur several times per period**
  - **can only occur once per period**
- **execution: once per period**
- **blocking: at most once per period for each source**

The schedulability formulas are modified to add a “blocking” or “priority inversion” term to account for inversion effects.

UB Test with Blocking

Include blocking while calculating effective utilization for each tasks:

$$f_i = \sum_{j \in Hn} \frac{C_j}{T_j} + \frac{C_i}{T_i} + \frac{B_i}{T_i} + \frac{1}{T_i} \sum_{k \in H1} C_k$$

Hn **Preemption (can hit** ***n*** **times)**

Execution

Blocking

H1 **Preemption (can hit once)**

RT Test with Blocking

Blocking is also included in the RT test:

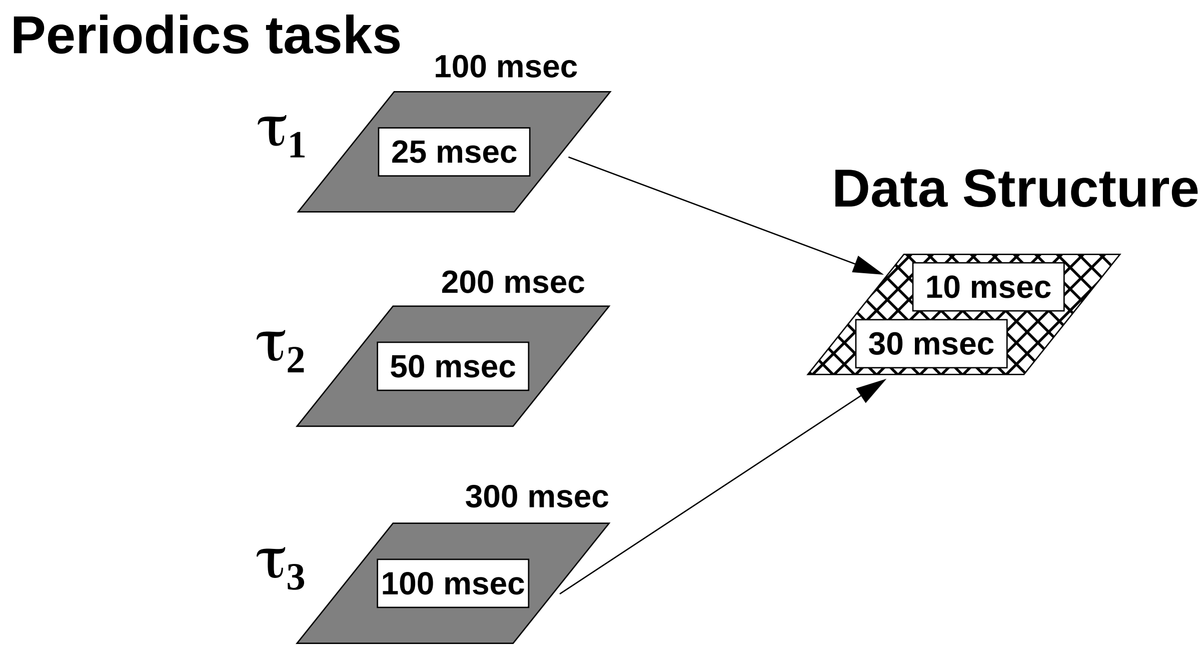
$$a_{n+1} = B_{\boldsymbol{i}} + \boldsymbol{C}_{\boldsymbol{i}} + \sum_{j=1}^{i-1} \left\lceil \frac{a_n}{T_j} \right\rceil C_j$$

$$\text{where } a_0 = B_{\boldsymbol{i}} + \sum_{j=1}^{i} C_j$$

Perform test as before, including blocking effect.

Example: Considering Blocking

Consider the following example:



What is the worst-case blocking effect (priority inversion) experienced by each task?

Example: Adding Blocking

Task τ_2 does not use the data structure. Task τ_2 experiences no priority inversion.

Task τ_1 shares the data structure with τ_3. Task τ_1 could have to wait for τ_3 to complete its critical section. But worse, if τ_2 preempts while τ_1 is waiting for the data structure, τ_1 could have to wait for τ_2's entire computation.

This is the resulting table:

Task	Period	Execution Time	Priority	Blocking Delays	Deadline
τ_1	100	25	High	30+50	100
τ_2	200	50	Medium	0	200
τ_3	300	100	Low	0	300

UB Test for Example

Recall UB test with blocking:

$$f_i = \sum_{j \in Hn} \frac{C_j}{T_j} + \frac{C_i}{T_i} + \frac{B_i}{T_i} + \frac{1}{T_i} \sum_{k \in H1} C_k$$

$$f_1 = \frac{C_1}{T_1} + \frac{B_1}{T_1} = \frac{25}{100} + \frac{80}{100} = 1.05 > 1.00$$ Not schedulable

$$f_2 = \frac{C_1}{T_1} + \frac{C_2}{T_2} = \frac{25}{100} + \frac{50}{200} = 0.50 < U(2)$$

$$f_3 = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} = \frac{25}{100} + \frac{50}{200} + \frac{100}{300} = 0.84 > U(3)$$

RT test shows
τ_3 is schedulable

Synchronization Protocols

No preemption

Basic priority inheritance

Highest locker's priority

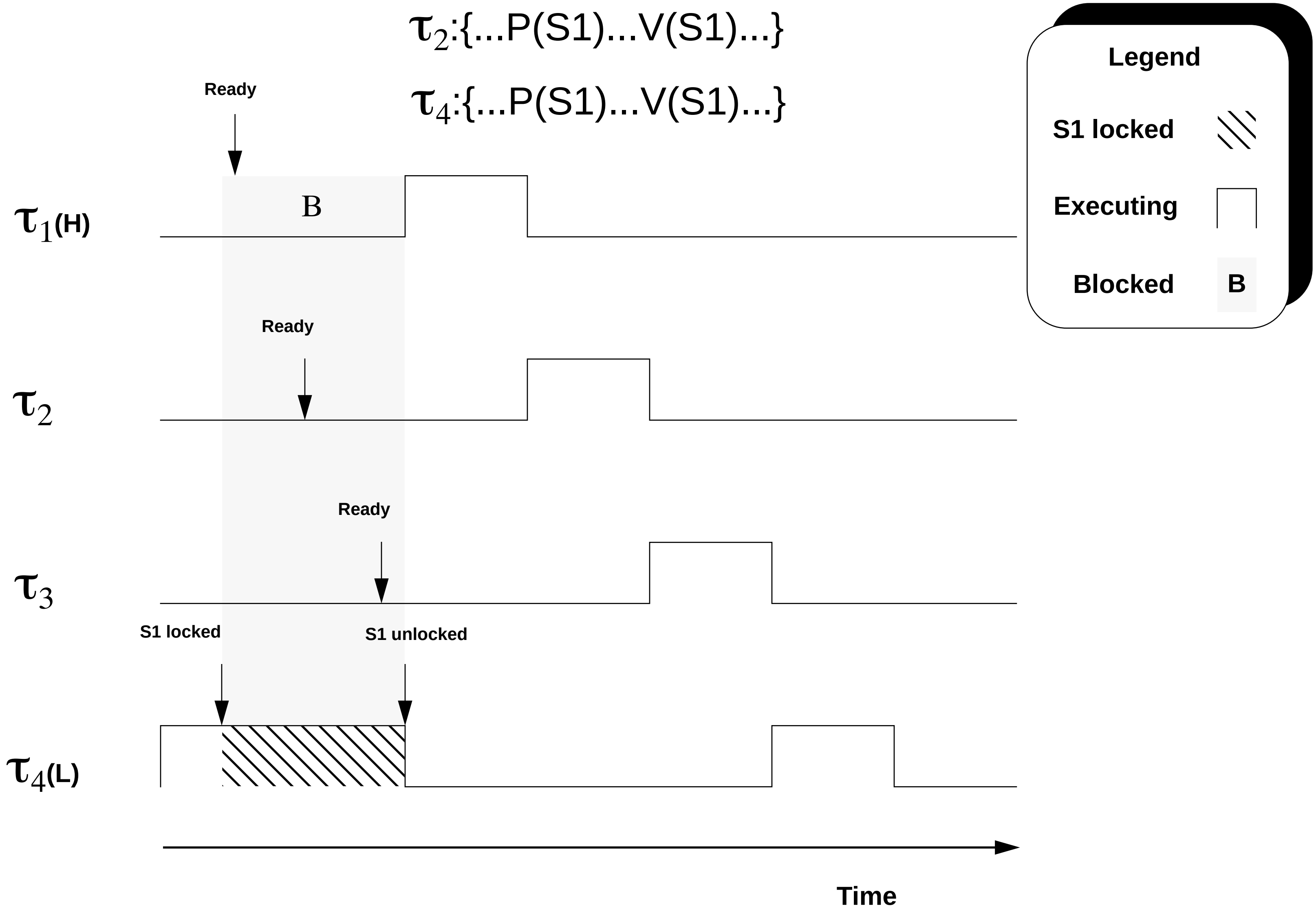
Priority ceiling

Each protocol prevents unbounded priority inversion.

Nonpreemption Protocol

τ_2:{...P(S1)...V(S1)...}

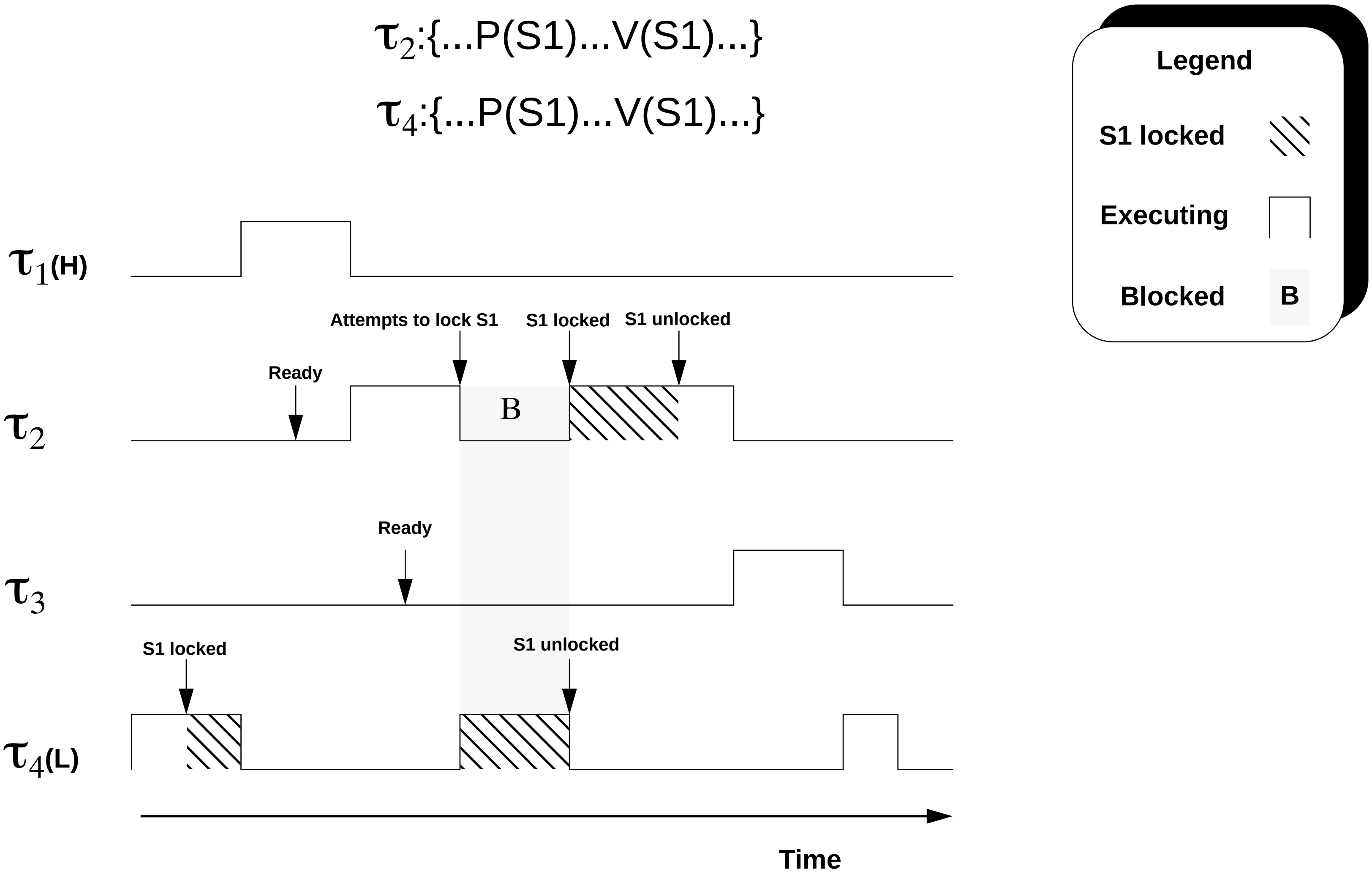
τ_4:{...P(S1)...V(S1)...}



Basic Inheritance Protocol (BIP)

τ_2:{...P(S1)...V(S1)...}

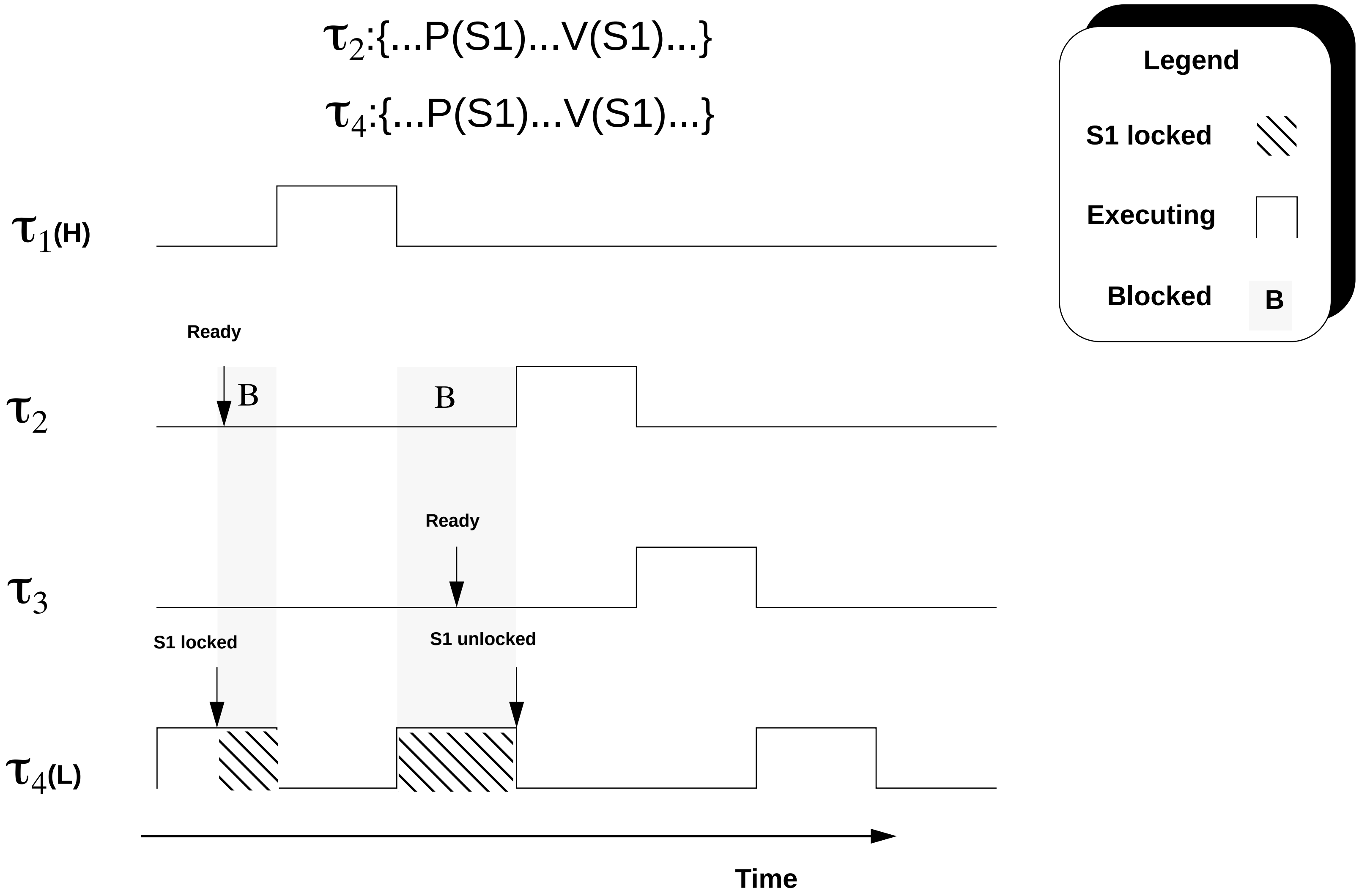
τ_4:{...P(S1)...V(S1)...}



Highest Locker's Priority Protocol

τ_2:{...P(S1)...V(S1)...}

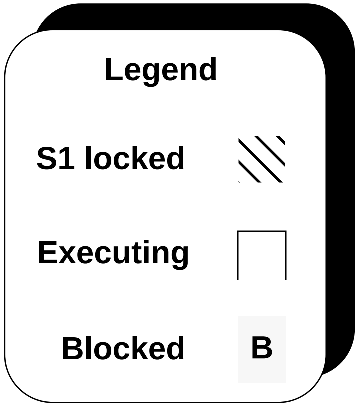
τ_4:{...P(S1)...V(S1)...}

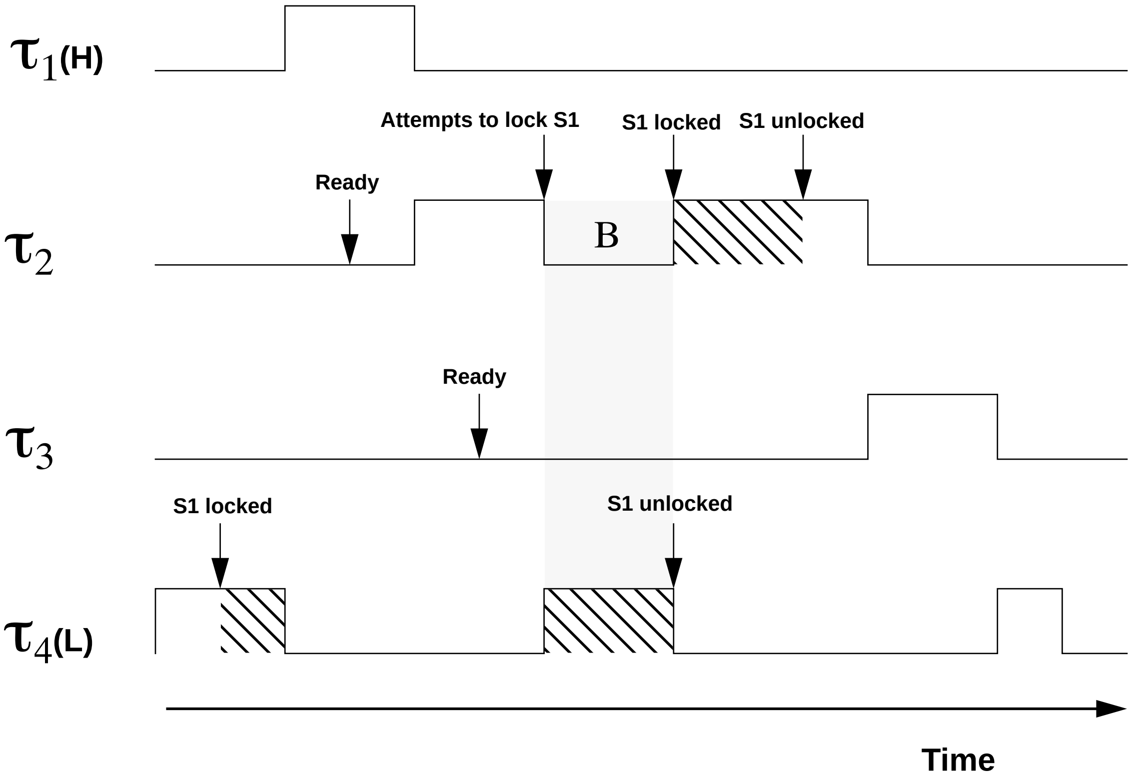


Priority Ceiling Protocol (PCP)

τ_2:{...P(S1)...V(S1)...}

τ_4:{...P(S1)...V(S1)...}



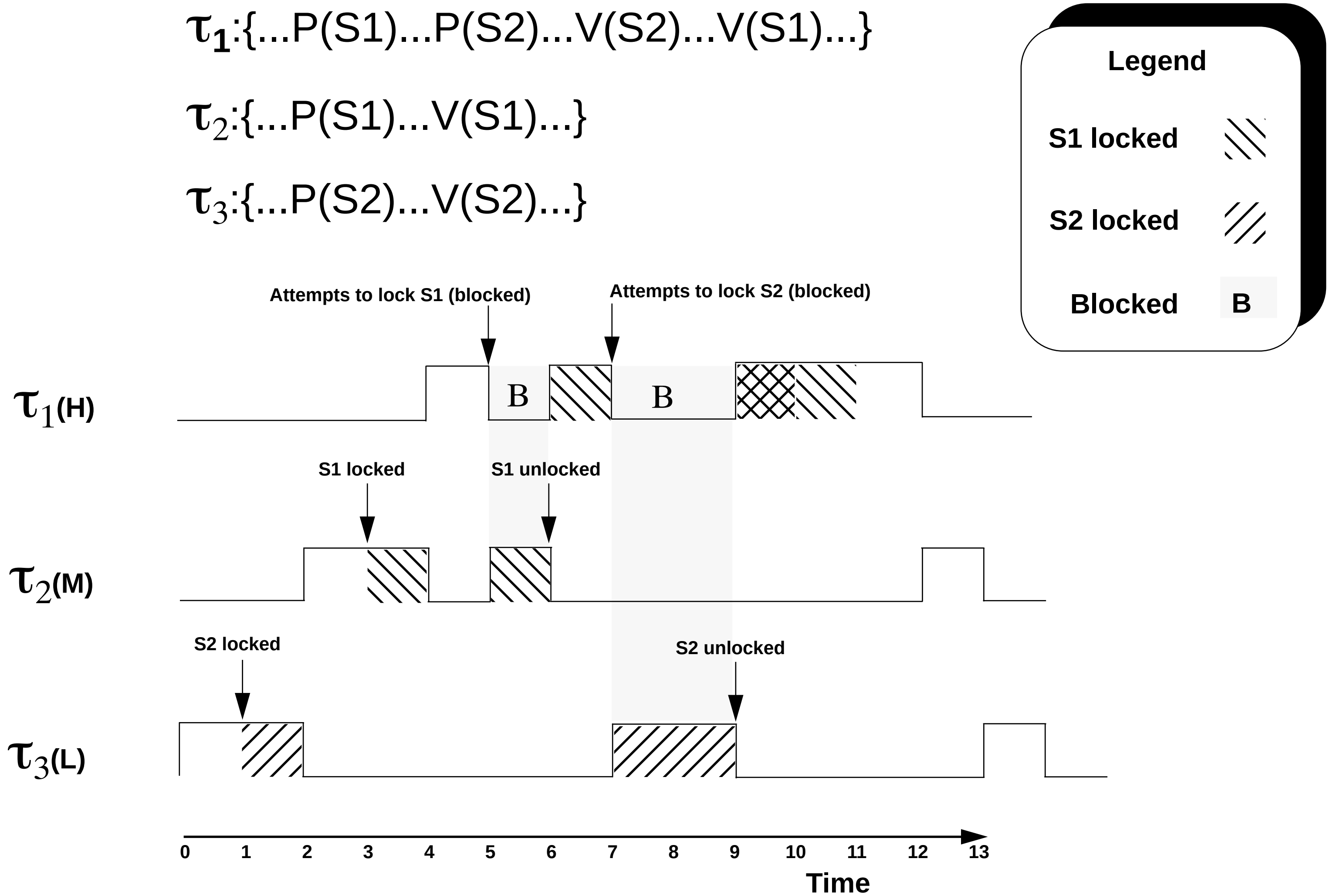


Example Of Chained Blocking (BIP)

τ_1:{...P(S1)...P(S2)...V(S2)...V(S1)...}

τ_2:{...P(S1)...V(S1)...}

τ_3:{...P(S2)...V(S2)...}

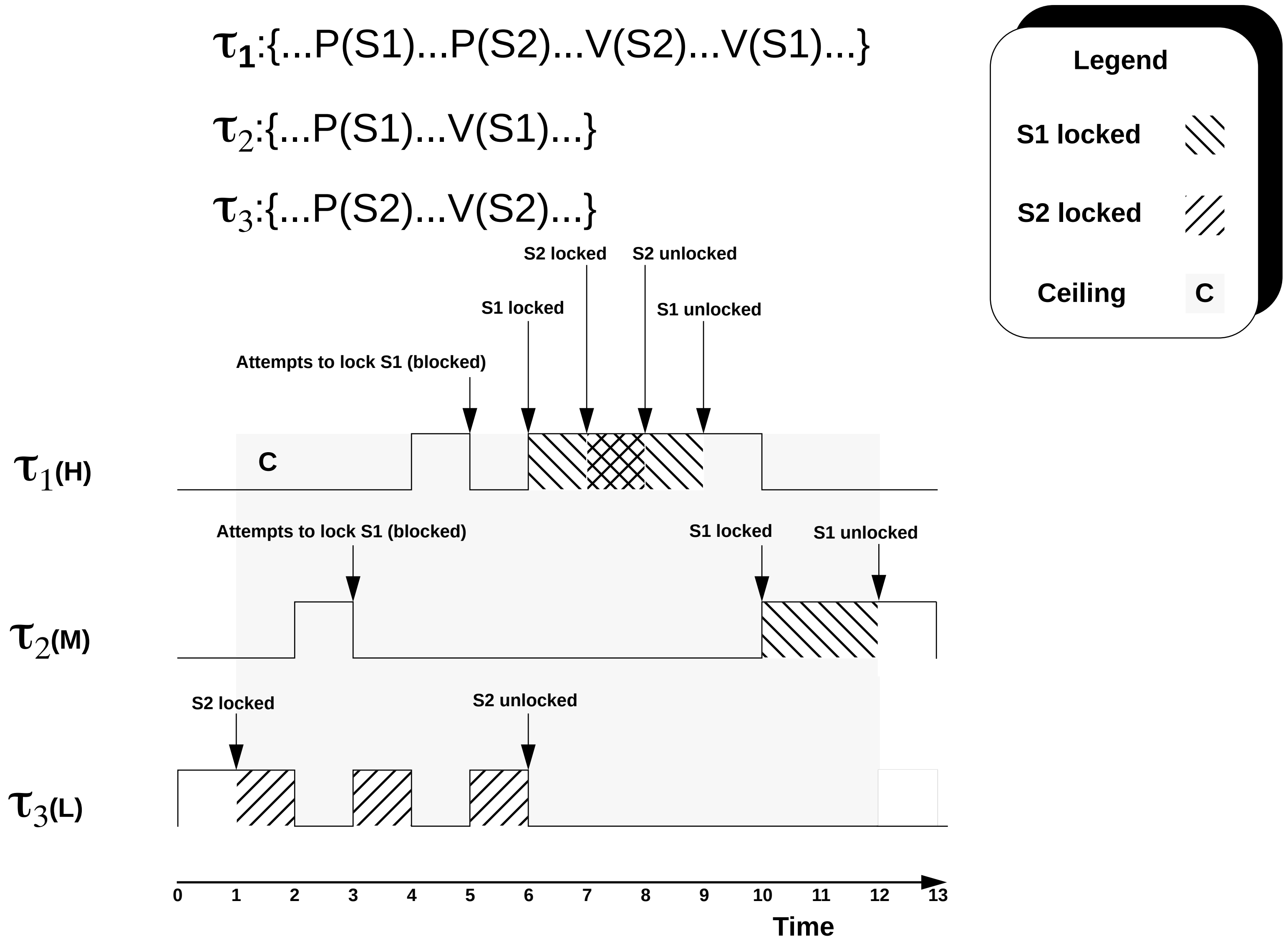


Blocked At Most Once (PCP)

τ_1:{...P(S1)...P(S2)...V(S2)...V(S1)...}

τ_2:{...P(S1)...V(S1)...}

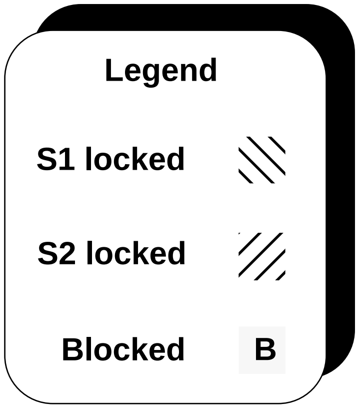
τ_3:{...P(S2)...V(S2)...}

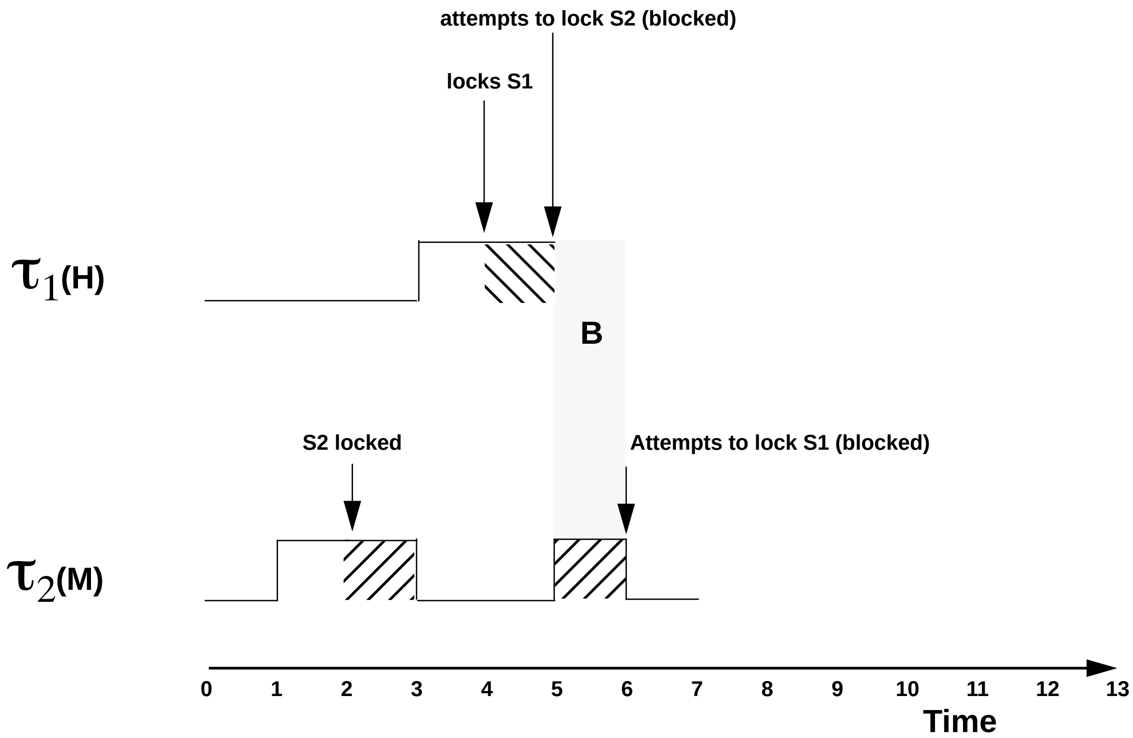


Deadlock: Using BIP

τ_1:{...P(S1)...P(S2)...V(S2)...V(S1)...}

τ_2:{...P(S2)...P(S1)...V(S1)...V(S2)...}

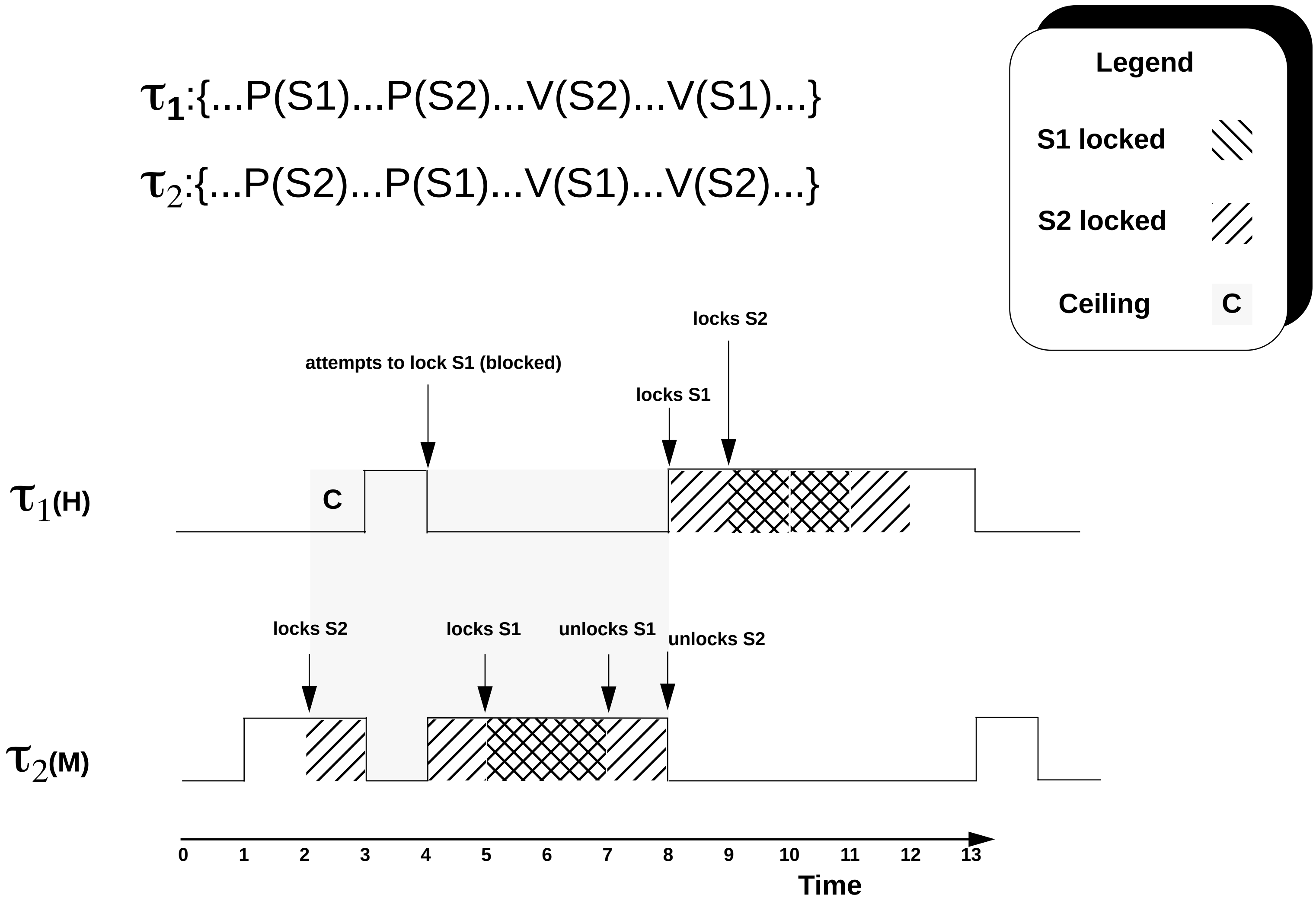




Deadlock Avoidance: Using PCP

τ_1:{...P(S1)...P(S2)...V(S2)...V(S1)...}

τ_2:{...P(S2)...P(S1)...V(S1)...V(S2)...}



Summary of Synchronization Protocols

Protocol	Bounded Priority Inversion	Blocked at Most Once	Deadlock Avoidance
Nonpreemptible critical sections	Yes	Yes[1]	Yes[1]
Highest locker's priority	Yes	Yes[1]	Yes[1]
Basic inheritance	Yes	No	No
Priority ceiling	Yes	Yes[2]	Yes

[1] Only if tasks do not suspend within critical sections

[2] PCP is not affected if tasks suspend within critical sections

Sample Problem with Synchronization

When basic priority inheritance protocol is used:

Task	Period	Execution Time	Priority	Blocking Delays	Deadline
τ_1	100	20	High	20+10	100
τ_2	150	40	Medium	10	130
τ_3	350	100	Low	0	350

UB Test for Sample Problem

This format is sometimes called a schedulability model for the task set:

$$f_1 = \frac{C_1}{T_1} + \frac{B_1}{T_1} = \frac{20}{100} + \frac{30}{100} = 0.500 < U(1)$$

$$f_2 = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{B_2}{T_2} = \frac{20}{100} + \frac{40}{150} + \frac{10}{150} = 0.534 < 0.729$$

$$U(2, .80) = 0.729$$

$$f_3 = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} = \frac{20}{100} + \frac{40}{150} + \frac{100}{350} = 0.753 < U(3)$$

Rate Monotonic Analysis

Introduction

Periodic tasks

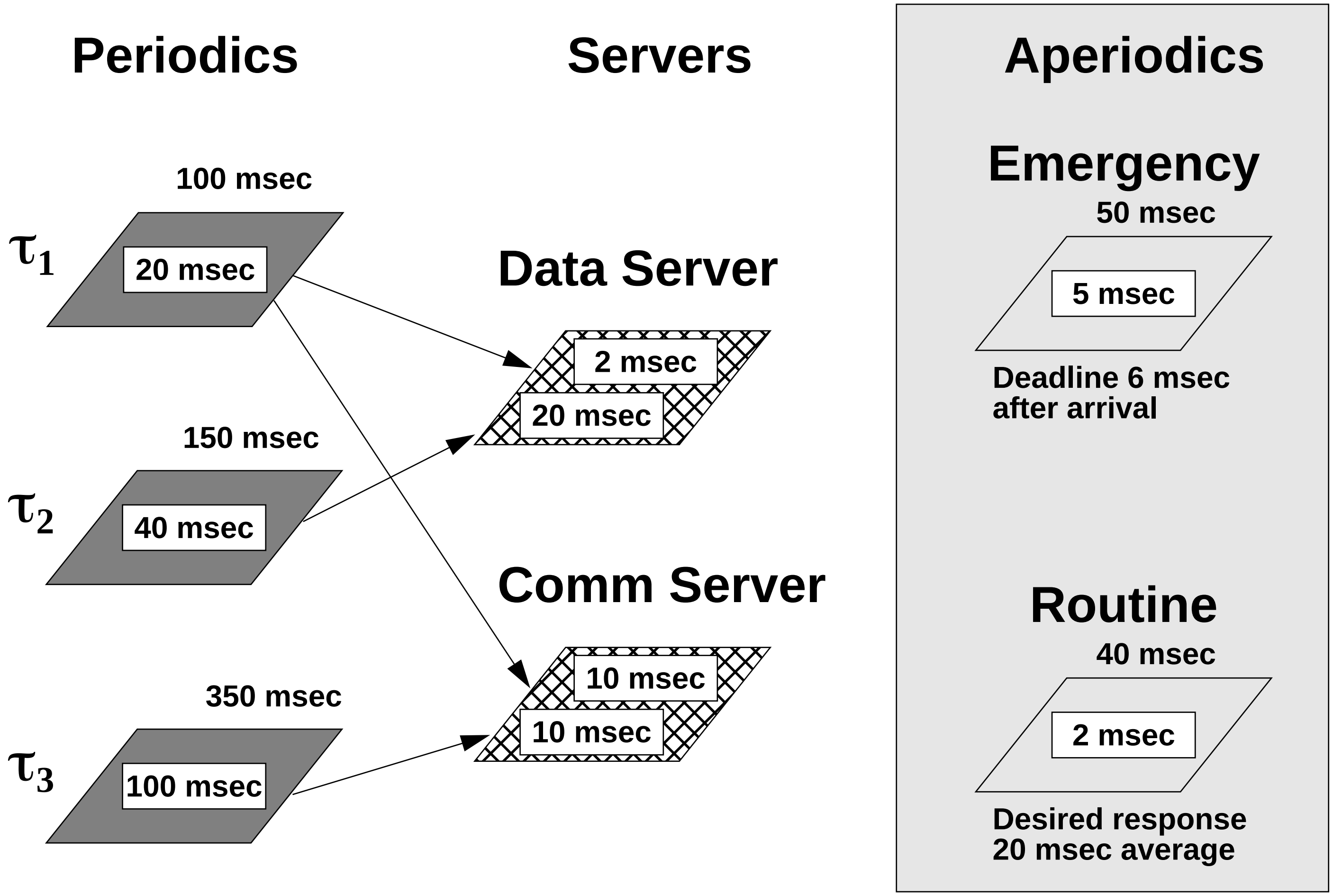
Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer

Sample Problem: Aperiodics



τ_2's deadline is 20 msec before the end of each period.

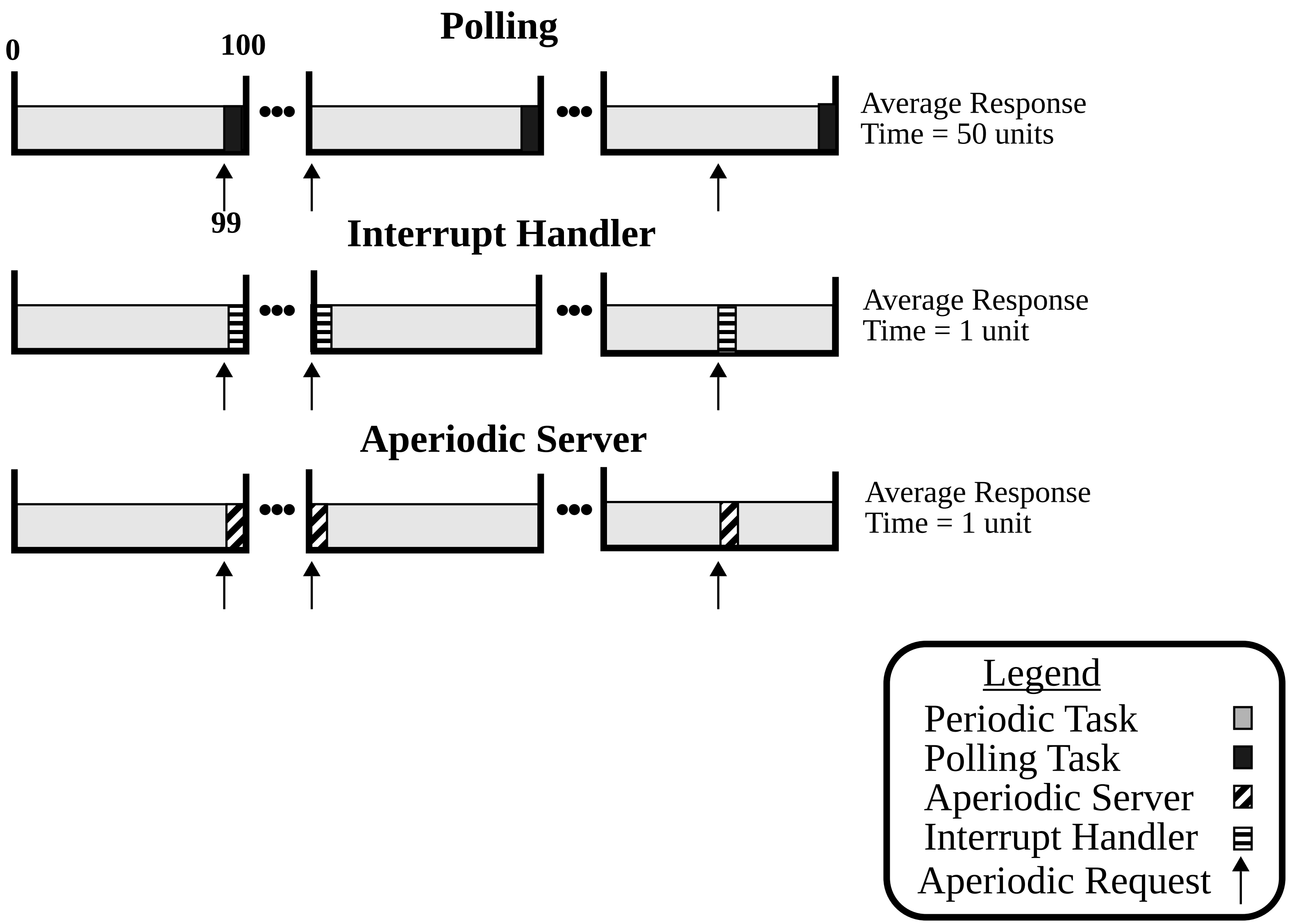
Concepts and Definitions

Aperiodic task: runs at unpredictable intervals

Aperiodic deadline:

- **hard, minimum interarrival time**
- **soft, best average response time**

Scheduling Aperiodic Tasks



Aperiodic Servers

Can be compared to periodic tasks:

- **fixed execution budget**
- **replenishment interval (period)**

Priority adjusted to meet requirements

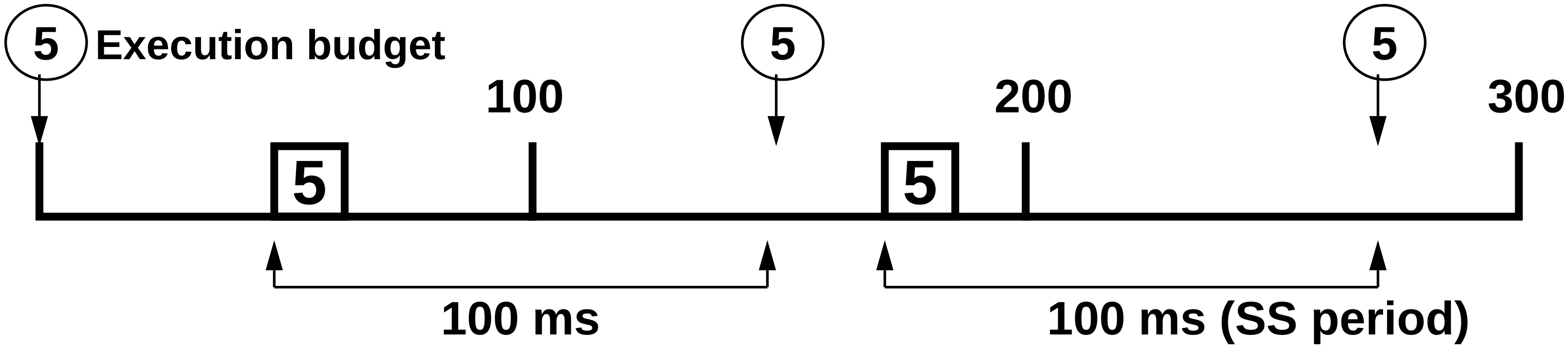
Sporadic Server (SS)

Modeled as periodic tasks:

- **fixed execution budget *(C)***
- **replenishment interval *(T)***

Priority adjusted to meet requirements

Replenishment occurs one “period” after start of use.



Sample Problem: Aperiodics

The sample problem has the following requirements:

- **emergency event:**
  - **5 msec of work**
  - **arrives every 50 msec, worst-case**
  - **hard deadline 6 msec after arrival**

- **routine event:**
  - **2 msec of work on average**
  - **arrives every 40 msec on average**
  - **desired average response of 20 msec after arrival**

Sample Problem: Sporadic Servers

Emergency server (ES); for minimum response:

- **set execution budget to processing time: $C = 5$**
- **set replenishment interval to minimum interarrival time: $T = 50$**

Routine server (RS); for average response:

- **set execution budget to processing time: $C = 2$**
- **use queueing theory to determine required replenishment interval, T**

Then assign priorities based on periods, T_i, of tasks.

Routine Server Period

Using M/D/1 queueing approximation:

$$W = \frac{\frac{(T_R)^2}{I}}{2\left(1 - \frac{T_R}{I}\right)} + C_R$$

I = average interarrival time between events

W = average response time

C_R = capacity of sporadic server = processing time

T_R = required sporadic server replenishment period

Routine Server Budget

Computing server replenishment interval:

$$T_R = (C_R - W) + \sqrt{(W - C_R)(W - C_R + 2I)}$$

$$T_R = (2 - 20) + \sqrt{(20 - 2)(20 - 2 + 80)}$$

$$T_R = 24$$

Note: For more details, see RMA handbook.

Sample Problem: Schedulability Analysis (BIP)

The task set is now:

Task	Period	Execution Time	Priority	Blocking Delays	Deadline
τ_E	50	5	Very High	0	6
τ_R	24	2	High	0	24
τ_1	100	20	Medium	20	100
τ_2	150	40	Low	10	150
τ_3	350	100	Very Low	0	350

Sample Problem: Schedulability Analysis

Using the RT test, worst-case response times are

- τ_E: **5 ms**
- τ_R: **7 ms**
- τ_1: **56 ms**
- τ_2: **88 ms**
- τ_3: **296 ms**

All requirements for sample problem are satisfied.

Rate Monotonic Analysis

Introduction

Periodic tasks

Extending basic theory

Synchronization and priority inversion

Aperiodic servers

Case study: BSY-1 Trainer

BSY-1 Trainer Case Study

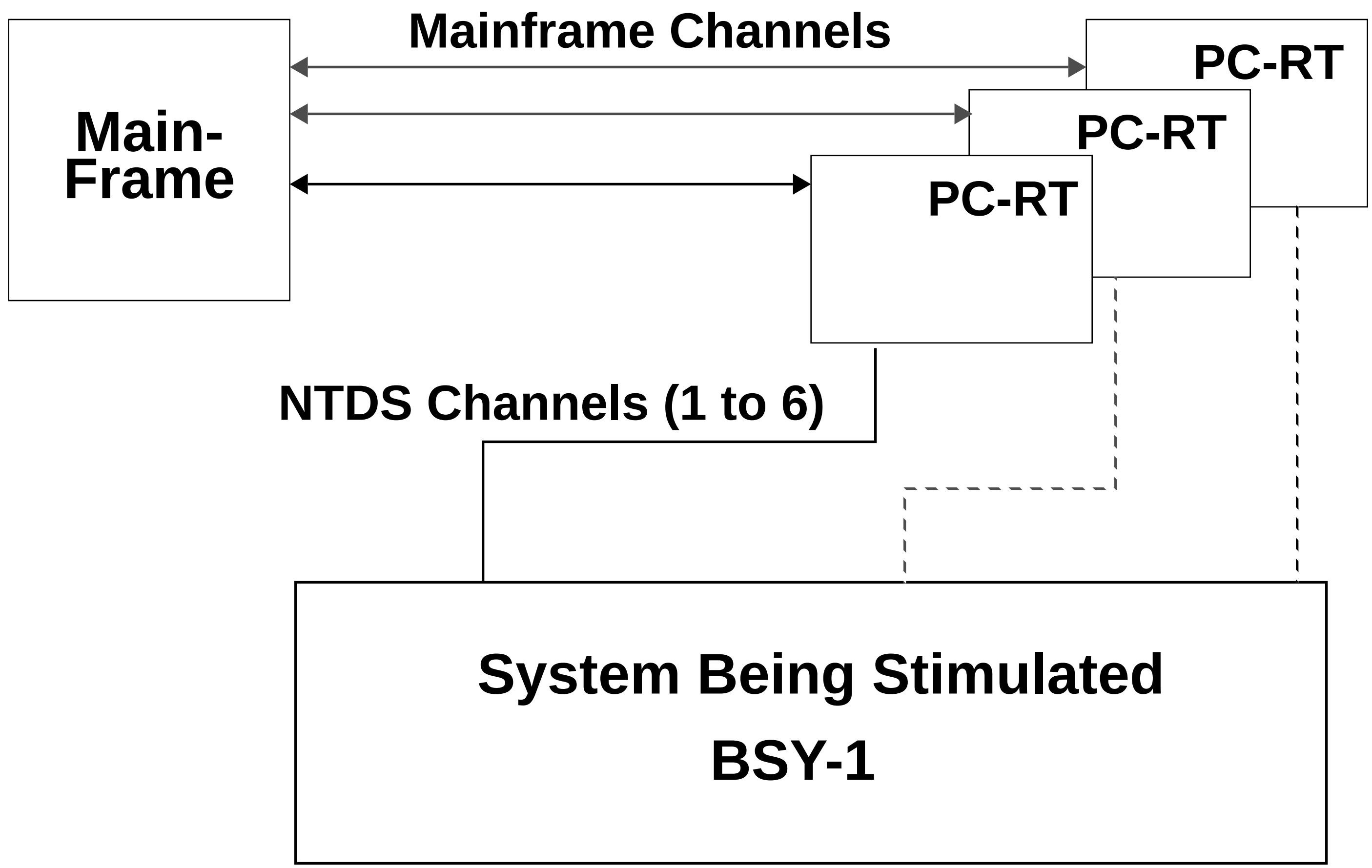
This case study is interesting for several reasons:

- **RMS is not used, yet the system is analyzable using RMA**
- **“obvious” solutions would not have helped**
- **RMA correctly diagnosed the problem**

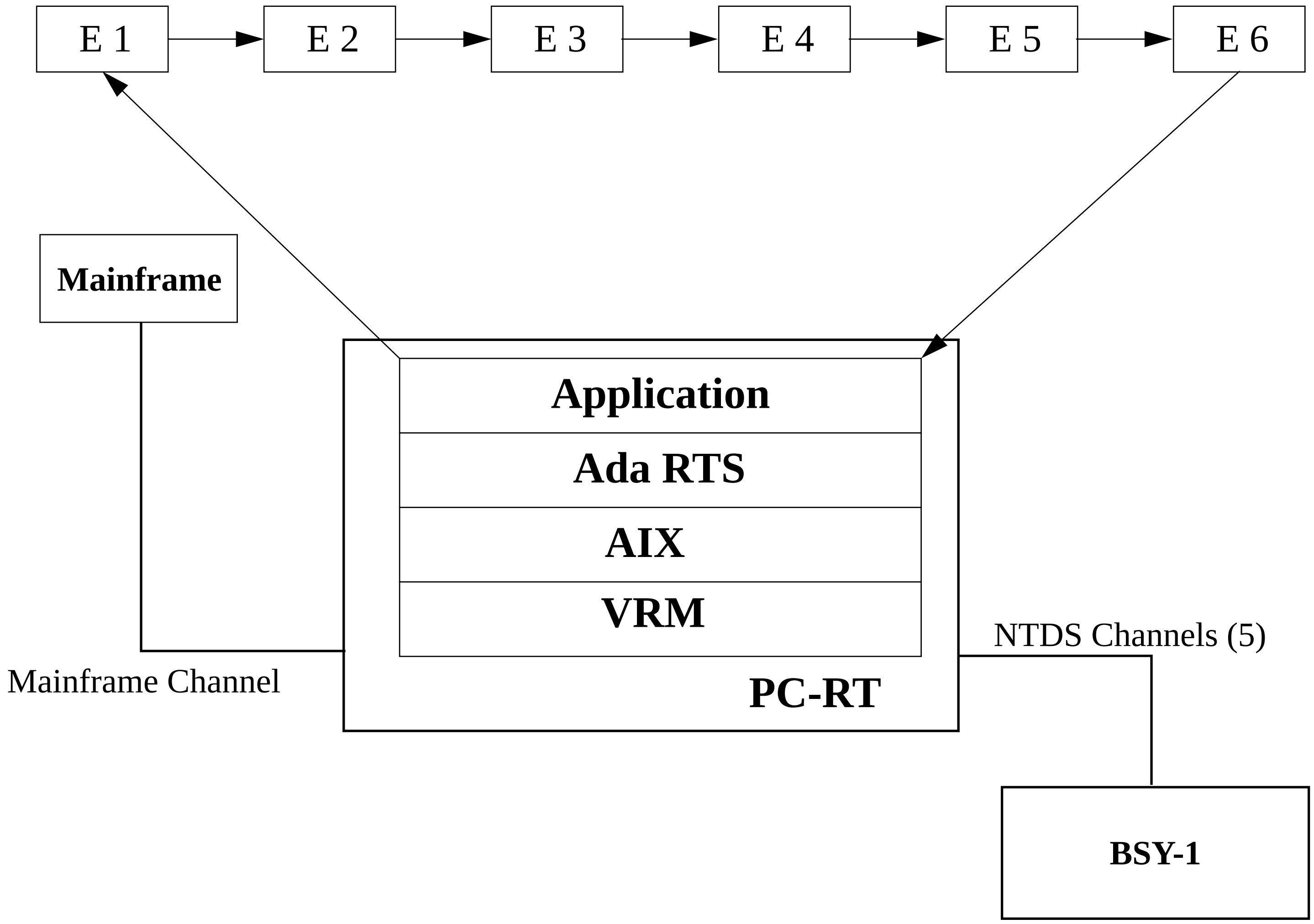
Insights to be gained:

- **devastating effects of nonpreemption**
- **how to apply RMA to a round-robin scheduler**
- **contrast conventional wisdom about interrupt handlers with the results of an RMA**

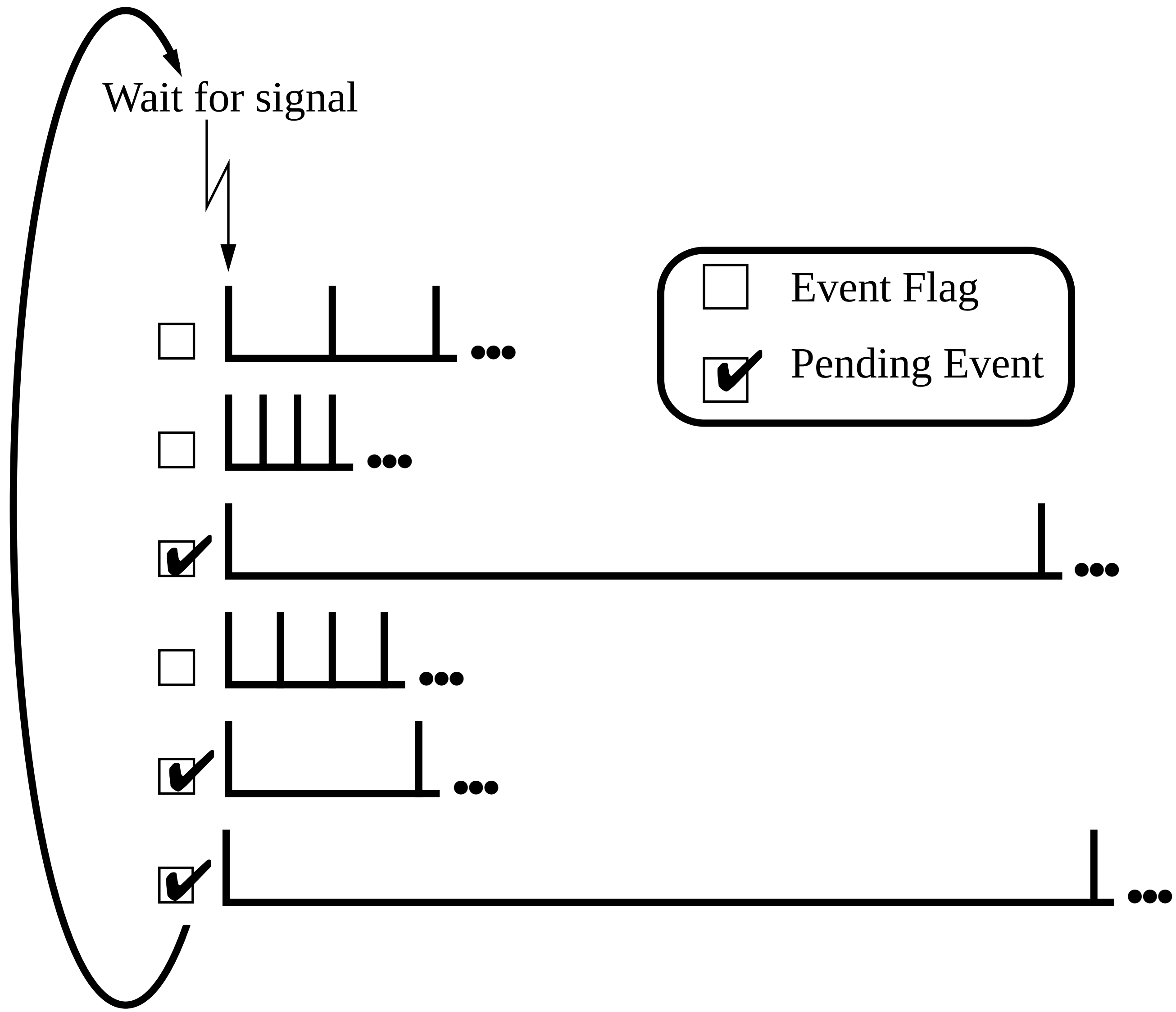
System Configuration



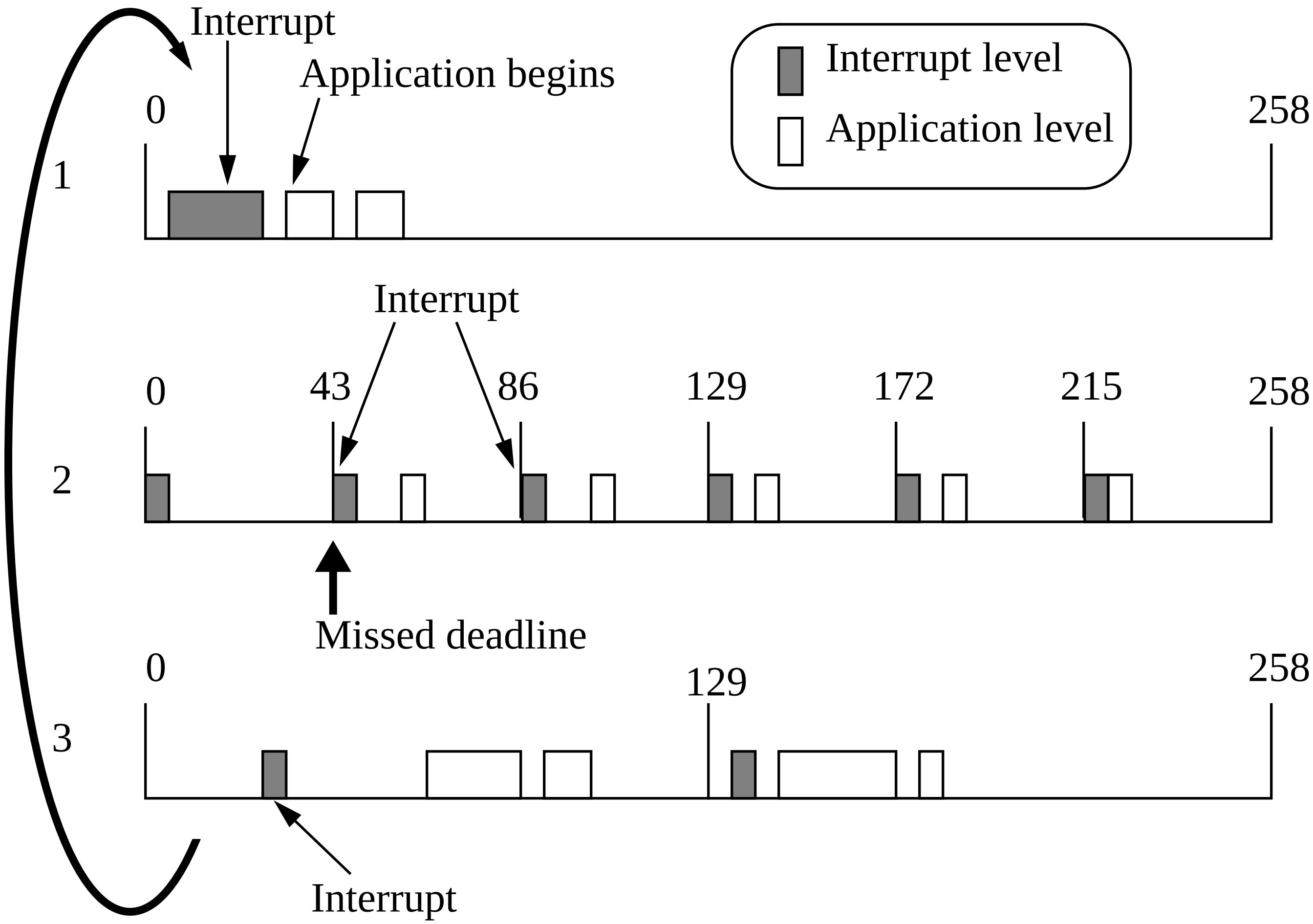
Software Design



Scheduling Discipline



Execution Sequence: Original Design



Problem Analysis by Development Team

During integration testing, the PC-RT could not keep up with the mainframe computer.

The problem was perceived to be inadequate throughput in the PC-RT.

Actions planned to solve the problem:

- **move processing out of the application and into VRM interrupt handlers**
- **improve the efficiency of AIX signals**
- **eliminate the use of Ada in favor of C**

Data from Rate Monotonic Investigation

	C_i (msec)	C_a (msec)	C (msec)	T (msec)	U
Event 1	2.0	0.5	2.5	43	0.059
Event 2	7.4	8.5	15.9	74	0.215
Event 3	6.0	0.6	6.6	129	0.052
Event 4	21.5	26.7	48.2	258	0.187
Event 5	5.7	23.4	29.1	1032	0.029
Event 6	2.8	1.0	3.8	4128	0.001
Total					0.543

Observe that total utilization is only 54%; the problem cannot be insufficient throughput.

Analyzing Original Design

Event ID	Arrival Period	Execution Time	Priority	Blocking Delays	Deadline
e1i	43	2.0	HW	0	n/a
e2i	74	7.4	HW	0	n/a
e3i	129	6.0	HW	0	n/a
e4i	258	21.5	HW	0	n/a
e5i	1032	5.7	HW	0	n/a
e6i	4128	2.8	HW	0	n/a
e1a	43	0.5	SW	0	43
e2a	74	8.5	SW	0	74
e3a	129	0.6	SW	0	129
e4a	258	26.7	SW	0	258
e5a	1032	23.4	SW	0	1032
e6a	4128	1.0	SW	0	4128

Schedulability Model: Original Design

e1a $$\frac{C_{1a}}{T_{1a}} + \left[\frac{C_{1i} + C_{2i} + C_{2a} + C_{3i} + C_{3a} + C_{4i} + C_{4a} + C_{5i} + C_{5a} + C_{6i} + C_{6a}}{T_1}\right] \le U(1)$$

e2a $$\left[\frac{C_{1i} + C_{1a}}{T_1}\right] + \frac{C_{2a}}{T_2} + \left[\frac{C_{2i} + C_{3i} + C_{3a} + C_{4i} + C_{4a} + C_{5i} + C_{5a} + C_{6i} + C_{6a}}{T_2}\right] \le U(2)$$

e3a $$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2}\right] + \frac{C_{3a}}{T_3} + \left[\frac{C_{3i} + C_{4i} + C_{4a} + C_{5i} + C_{5a} + C_{6i} + C_{6a}}{T_3}\right] \le U(3)$$

e4a $$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3}\right] + \frac{C_{4a}}{T_4} + \left[\frac{C_{4i} + C_{5i} + C_{5a} + C_{6i} + C_{6a}}{T_4}\right] \le U(4)$$

e5a $$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3} + \frac{C_{4i} + C_{4a}}{T_4}\right] + \frac{C_{5a}}{T_5} + \left[\frac{C_{5i} + C_{6i} + C_{6a}}{T_5}\right] \le U(5)$$

e6a $$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3} + \frac{C_{4i} + C_{4a}}{T_4} + \frac{C_{5i} + C_{5a}}{T_5}\right] + \frac{C_6}{T_6} + \left[\frac{C_{6i}}{T_6}\right] \le U(6)$$

Schedulability Test: Original Design

e1a $$\frac{0.5}{43} + \left[\frac{2.0 + 7.4 + 8.5 + 6.0 + 0.6 + 21.5 + 26.7 + 5.7 + 23.4 + 2.8 + 1.0}{43}\right] \leq \boldsymbol{U}(\mathbf{1})$$

e2a $$\left[\frac{2.0 + 0.5}{43}\right] + \frac{8.5}{74} + \left[\frac{7.4 + 6.0 + 0.6 + 21.5 + 26.7 + 5.7 + 23.4 + 2.8 + 1.0}{74}\right] \leq \boldsymbol{U}(\mathbf{2})$$

e3a $$\left[\frac{2.0 + 0.5}{43} + \frac{7.4 + 8.5}{74}\right] + \frac{0.6}{129} + \left[\frac{6.0 + 21.5 + 26.7 + 5.7 + 23.4 + 2.8 + 1.0}{129}\right] \leq \boldsymbol{U}(\mathbf{3})$$

e4a $$\left[\frac{2.0 + 0.5}{43} + \frac{7.4 + 8.5}{74} + \frac{6.0 + 0.6}{129}\right] + \frac{26.7}{258} + \left[\frac{21.5 + 5.7 + 23.4 + 2.8 + 1.0}{258}\right] \leq \boldsymbol{U}(\mathbf{4})$$

e5a $$\left[\frac{2.0 + 0.5}{43} + \frac{7.4 + 8.5}{74} + \frac{6.0 + 0.6}{129} + \frac{21.5 + 26.7}{258}\right] + \frac{23.4}{1032} + \left[\frac{5.7 + 2.8 + 1.0}{1032}\right] \leq \boldsymbol{U}(\mathbf{5})$$

e6a $$\left[\frac{2.0 + 0.5}{43} + \frac{7.4 + 8.5}{74} + \frac{6.0 + 0.6}{129} + \frac{21.5 + 26.7}{258} + \frac{5.7 + 23.4}{1032}\right] + \frac{1.0}{4128} + \left[\frac{2.8}{4128}\right] \leq \boldsymbol{U}(\mathbf{6})$$

Utilization: Original Design

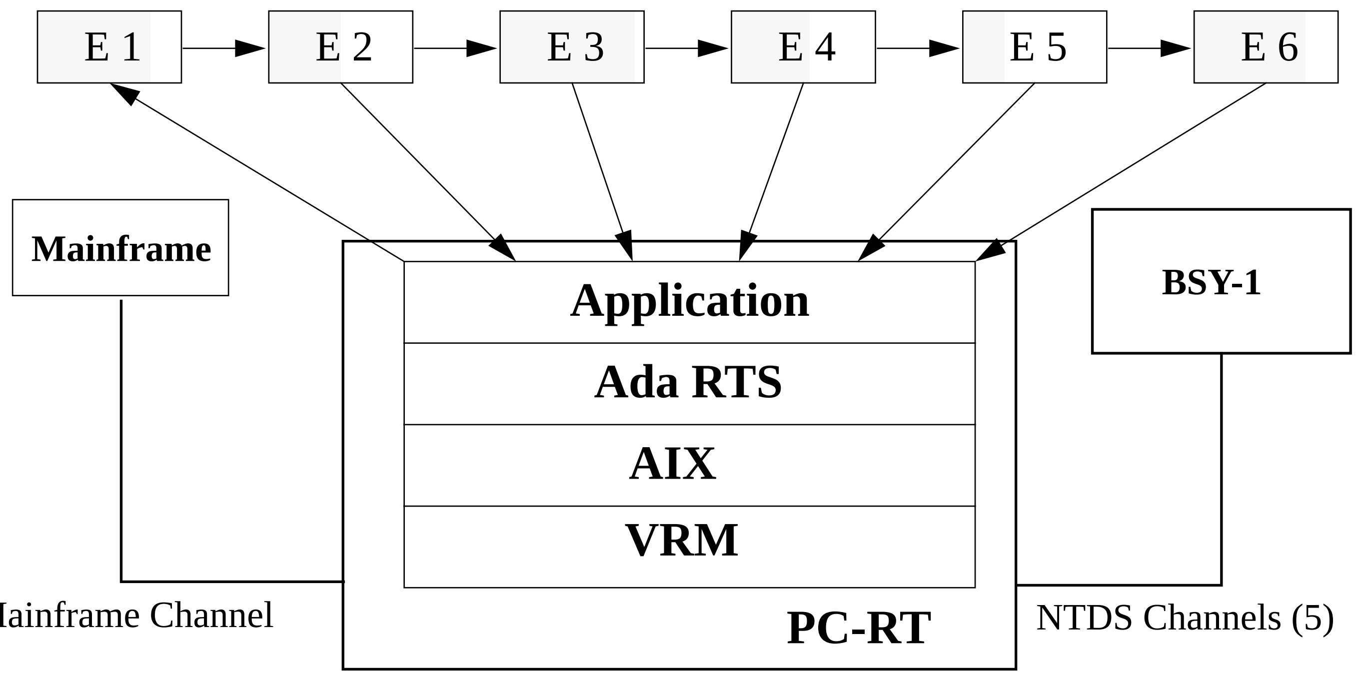
Event	Period (msec)	Preempt {*Hn*}	Execute	Preempt {*H1*}	Total (f_i)
1a	43	0.000	0.012	2.456	2.468
2a	74	0.059	0.115	1.286	1.460
3a	129	0.274	0.005	0.676	0.955
4a	258	0.326	0.104	0.211	0.641
5a	1032	0.513	0.023	0.010	0.546
6a	4128	0.542	0.001	0.001	0.544

Effective utilizations (f_i) for events 4, 5, and 6 are all under 69%. These events are schedulable.

The problem for events 1, 2, and 3 is excessive *H1* preemption.

Process Events in RM Order

U	5.9%	21.5%	5.2%	18.7%	2.9%	0.1%
C_i	2.0	7.4	6.0	21.5	5.7	2.8
C_a	0.5	8.5	0.6	26.7	23.4	1.0
T	43	74	129	258	1032	4128



Schedulability Model: Process Events in RM Order

e1a

$$\frac{C_{1a}}{T_1} + \left[\frac{max\,(C_{2a}, C_{3a}, C_{4a}, C_{5a}, C_{6a}) + C_{1i} + C_{2i} + C_{3i} + C_{4i} + C_{5i} + C_{6i}}{T_1}\right]$$

e2a

$$\left[\frac{C_{1i} + C_{1a}}{T_1}\right] + \frac{C_{2a}}{T_2} + \left[\frac{max\,(C_{3a}, C_{4a}, C_{5a}, C_{6a}) + C_{2i} + C_{3i} + C_{4i} + C_{5i} + C_{6i}}{T_2}\right]$$

e3a

$$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2}\right] + \frac{C_{3a}}{T_3} + \left[\frac{max\,(C_{4a}, C_{5a}, C_{6a}) + C_{3i} + C_{4i} + C_{5i} + C_{6i}}{T_3}\right]$$

e4a

$$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3}\right] + \frac{C_{4a}}{T_4} + \left[\frac{max\,(C_{5a}, C_{6a}) + C_{4i} + C_{5i} + C_{6i}}{T_4}\right]$$

e5a

$$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3} + \frac{C_{4i} + C_{4a}}{T_4}\right] + \frac{C_{5a}}{T_5} + \left[\frac{C_{6a} + C_{5i} + C_{6i}}{T_5}\right]$$

e6a

$$\left[\frac{C_{1i} + C_{1a}}{T_1} + \frac{C_{2i} + C_{2a}}{T_2} + \frac{C_{3i} + C_{3a}}{T_3} + \frac{C_{4i} + C_{4a}}{T_4} + \frac{C_{5i} + C_{5a}}{T_5}\right] + \frac{C_{6a}}{T_6} + \frac{C_{6i}}{T_6}$$

Schedulability Test: Process Events in RM Order

e1a $$\frac{0.5}{\mathbf{43}} + \left[\frac{(\mathbf{26.7}) + 2.0 + \mathbf{7.4} + \mathbf{6.0} + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{43}}\right]$$

e2a $$\left[\frac{\mathbf{2.5}}{\mathbf{43}}\right] + \frac{8.5}{\mathbf{74}} + \left[\frac{(\mathbf{26.7}) + 7.4 + \mathbf{6.0} + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{74}}\right]$$

e3a $$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}}\right] + \frac{0.6}{\mathbf{129}} + \left[\frac{(\mathbf{26.7}) + 6.0 + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{129}}\right]$$

e4a $$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}}\right] + \frac{26.7}{\mathbf{258}} + \left[\frac{(\mathbf{23.4}) + 21.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{258}}\right]$$

e5a $$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}}\right] + \frac{23.4}{\mathbf{1032}} + \left[\frac{1.0 + 5.7 + 2.8}{\mathbf{1032}}\right]$$

e6a $$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}} + \frac{\mathbf{29.1}}{\mathbf{1032}}\right] + \frac{1.0}{\mathbf{4128}} + \left[\frac{2.8}{\mathbf{4128}}\right]$$

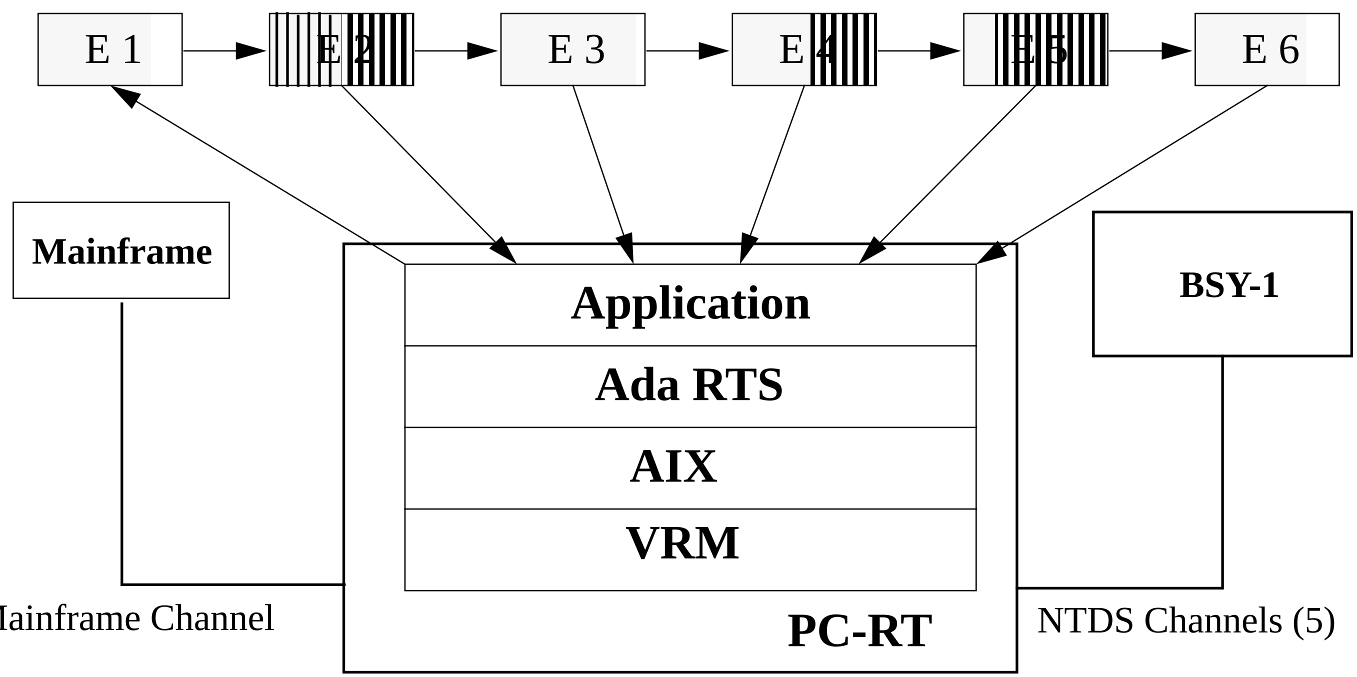
Utilization: Process Events in RM Order

Event	Period (msec)	Preempt {*Hn*}	Execute	Preempt {*H1*}	Total (f_i)	Previous Total
1a	43	0.000	0.012	1.677	1.689	2.468
2a	74	0.059	0.115	0.948	1.122	1.460
3a	129	0.274	0.005	0.487	0.766	0.955
4a	258	0.326	0.104	0.207	0.637	0.641
5a	1032	0.513	0.023	0.010	0.546	0.546
6a	4128	0.542	0.001	0.001	0.544	0.544

Increasing Preemptibility

Preemptible I/O (applied to E 2) | **Packetized Data Movement** (applied to E 4 and E 5)

	E 1	E 2	E 3	E 4	E 5	E 6
U	5.9%	21.5%	5.2%	18.7%	2.9%	0.1%
C_i	2.0	7.4 (1.5)	6.0	21.5	5.7	2.8
C_a	0.5	8.5 (1.7)	0.6	26.7 (4.5)	23.4 (3.9)	1.0
T	43	74	129	258	1032	4128



Schedulability Test: Preemptible I/O and Packetized Data Movement

e1a

$$\frac{0.5}{\mathbf{43}} + \left[\frac{(4.5) + 2.0 + 1.5 + \mathbf{6.0} + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{43}}\right]$$

e2a

$$\left[\frac{\mathbf{2.5}}{\mathbf{43}}\right] + \frac{8.5}{\mathbf{74}} + \left[\frac{(4.5) + 7.4 + \mathbf{6.0} + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{74}}\right]$$

e3a

$$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}}\right] + \frac{0.6}{\mathbf{129}} + \left[\frac{(4.5) + 6.0 + \mathbf{21.5} + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{129}}\right]$$

e4a

$$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}}\right] + \frac{26.7}{\mathbf{258}} + \left[\frac{(3.9) + 21.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{258}}\right]$$

e5a

$$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}}\right] + \frac{23.4}{\mathbf{1032}} + \left[\frac{1.0 + 5.7 + 2.8}{\mathbf{1032}}\right]$$

e6a

$$\left[\frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}} + \frac{\mathbf{29.1}}{\mathbf{1032}}\right] + \frac{1.0}{\mathbf{4128}} + \left[\frac{2.8}{\mathbf{4128}}\right]$$

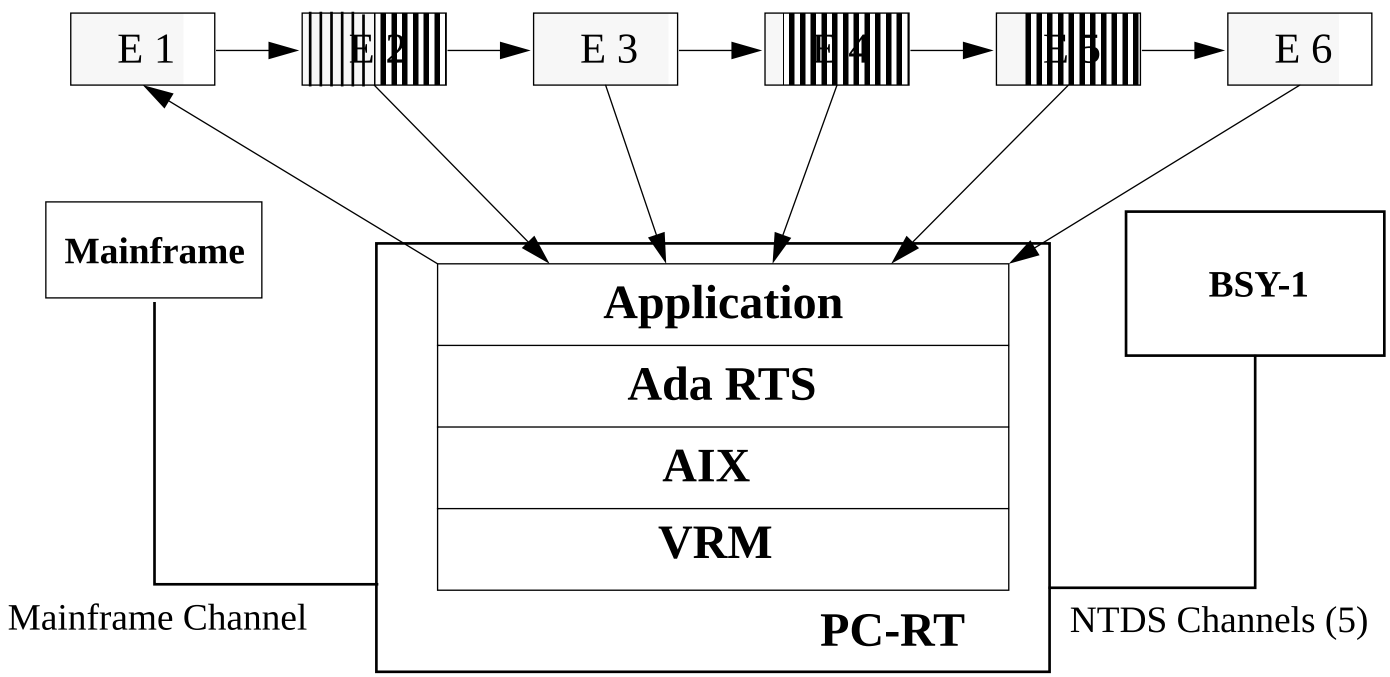
Utilization: Preemptible I/O and Packetized Data Movement

Event	Period (msec)	Preempt {*Hn*}	Execute	Preempt {*H1*}	Total (f_i)	Previous Total
1a	43	0.000	0.012	1.024	1.036	1.689
2a	74	0.059	0.115	0.648	0.822	1.122
3a	129	0.274	0.005	0.314	0.593	0.766
4a	258	0.326	0.104	0.132	0.562	0.637
5a	1032	0.513	0.023	0.010	0.546	0.546
6a	4128	0.542	0.001	0.001	0.544	0.544

According to the utilization bound test, all events now are schedulable, except event 1.

Streamlined Interrupt Handler

U	5.9%	21.5%		5.2%	18.7%		2.9%		0.1%
C_i	2.0	7.4	(1.5)	6.0	6.5		5.7		2.8
C_a	0.5	8.5	(1.7)	0.6	41.7	(4.5)	23.4	(3.9)	1.0
T	43	74		129	258		1032		4128



Schedulability Test: Streamlined Interrupt Handler

e1a

$$\frac{0.5}{\mathbf{43}} + \left\lceil \frac{(4.5) + 2.0 + 1.5 + \mathbf{6.0} + 6.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{43}} \right\rceil$$

e2a

$$\left\lceil \frac{\mathbf{2.5}}{\mathbf{43}} \right\rceil + \frac{8.5}{\mathbf{74}} + \left\lceil \frac{(4.5) + 7.4 + \mathbf{6.0} + 6.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{74}} \right\rceil$$

e3a

$$\left\lceil \frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} \right\rceil + \frac{0.6}{\mathbf{129}} + \left\lceil \frac{(4.5) + 6.0 + 6.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{129}} \right\rceil$$

e4a

$$\left\lceil \frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} \right\rceil + \frac{41.7}{\mathbf{258}} + \left\lceil \frac{(3.9) + 6.5 + \mathbf{5.7} + \mathbf{2.8}}{\mathbf{258}} \right\rceil$$

e5a

$$\left\lceil \frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}} \right\rceil + \frac{23.4}{\mathbf{1032}} + \left\lceil \frac{1.0 + 5.7 + 2.8}{\mathbf{1032}} \right\rceil$$

e6a

$$\left\lceil \frac{\mathbf{2.5}}{\mathbf{43}} + \frac{\mathbf{15.9}}{\mathbf{74}} + \frac{\mathbf{6.6}}{\mathbf{129}} + \frac{\mathbf{48.2}}{\mathbf{258}} + \frac{\mathbf{29.1}}{\mathbf{1032}} \right\rceil + \frac{1.0}{\mathbf{4128}} + \left\lceil \frac{2.8}{\mathbf{4128}} \right\rceil$$

Utilization: Streamlined Interrupt Handler

Event	Period (msec)	Preempt {*Hn*}	Execute	Preempt {*H1*}	Total (f_i)	Previous Total
1a	43	0.000	0.012	0.675	0.687	1.036
2a	74	0.059	0.115	0.445	0.619	0.822
3a	129	0.274	0.005	0.198	0.477	0.593
4a	258	0.326	0.162	0.074	0.562	0.562
5a	1032	0.513	0.023	0.010	0.546	0.546
6a	4128	0.542	0.001	0.001	0.544	0.544

Summary: BSY-1 Trainer Case Study

Recall original action plan:

- improve efficiency of AIX signals
- move processing from application to interrupts
- recode 17,000 lines of Ada to C

Final actions:

- increase preemption and improve AIX
- move processing from interrupts to application
- modify 300 lines of Ada code
- RMA took 3 people 3 weeks