10. BSY-1 Trainer Case Study

UNIVERSIDAD DE CANTABRIA

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**

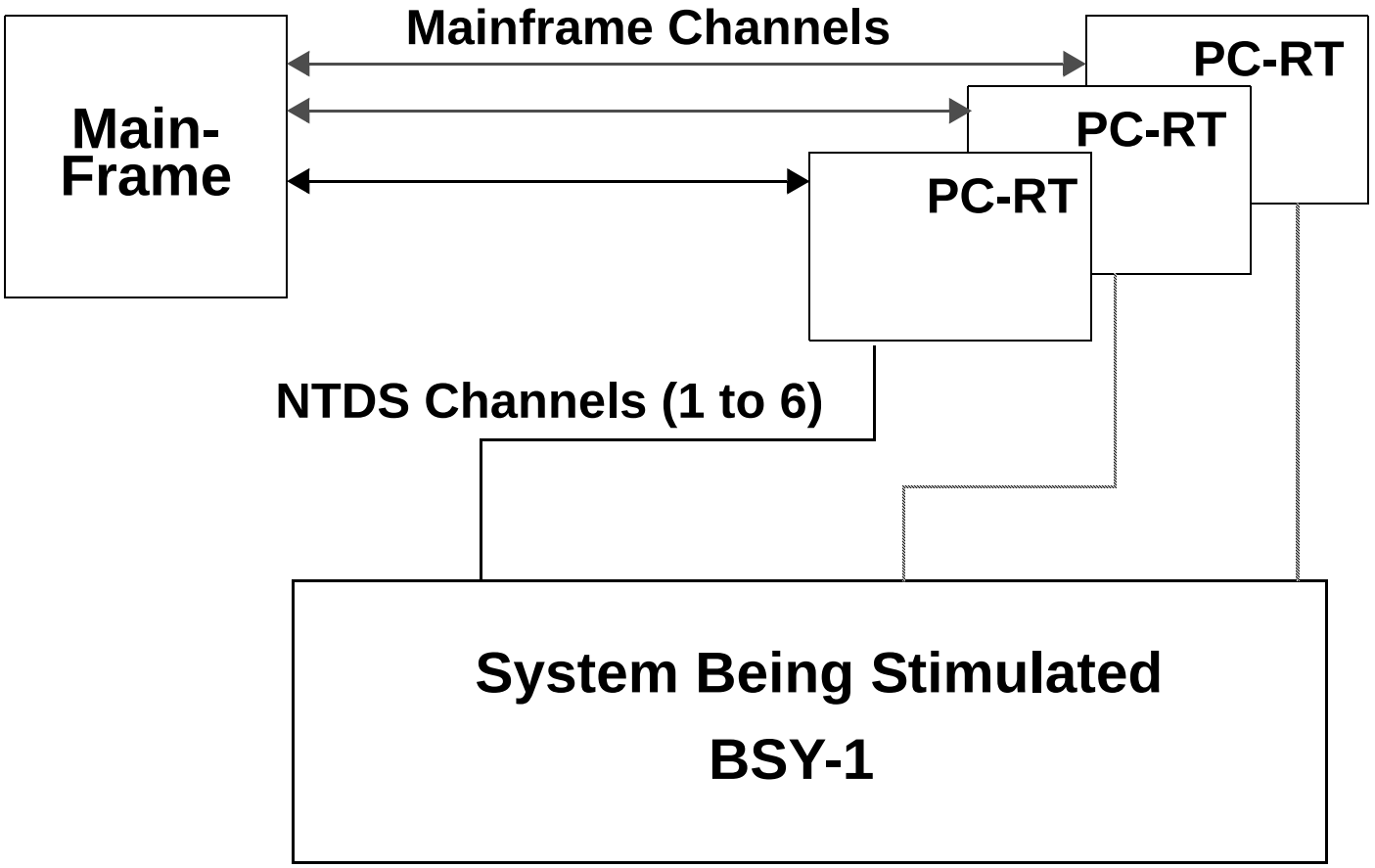
Notas:

UNIVERSIDAD DE CANTABRIA

This case study is interesting for several reasons that illustrate the power of using RMA. The PC-RT system prototype was experiencing severe timing overruns. Diagnosing the problem was difficult because the design team's analysis tools were insufficient. As a result, the measures contemplated by experienced system developers were going in the wrong direction. Even though rate monotonic *scheduling* had not been used, the application of rate monotonic analysis correctly diagnosed what was wrong and what to do to fix the problem.

This case study illustrates how to apply rate monotonic principles to a round-robin scheduler. It also illustrates some points that are more broadly applicable. It shows some representative sources of priority inversion and their devastating effects. It is also a contrast of conventional wisdom about interrupt handlers with insight gained from a rate monotonic analysis of the effects.

System Configuration

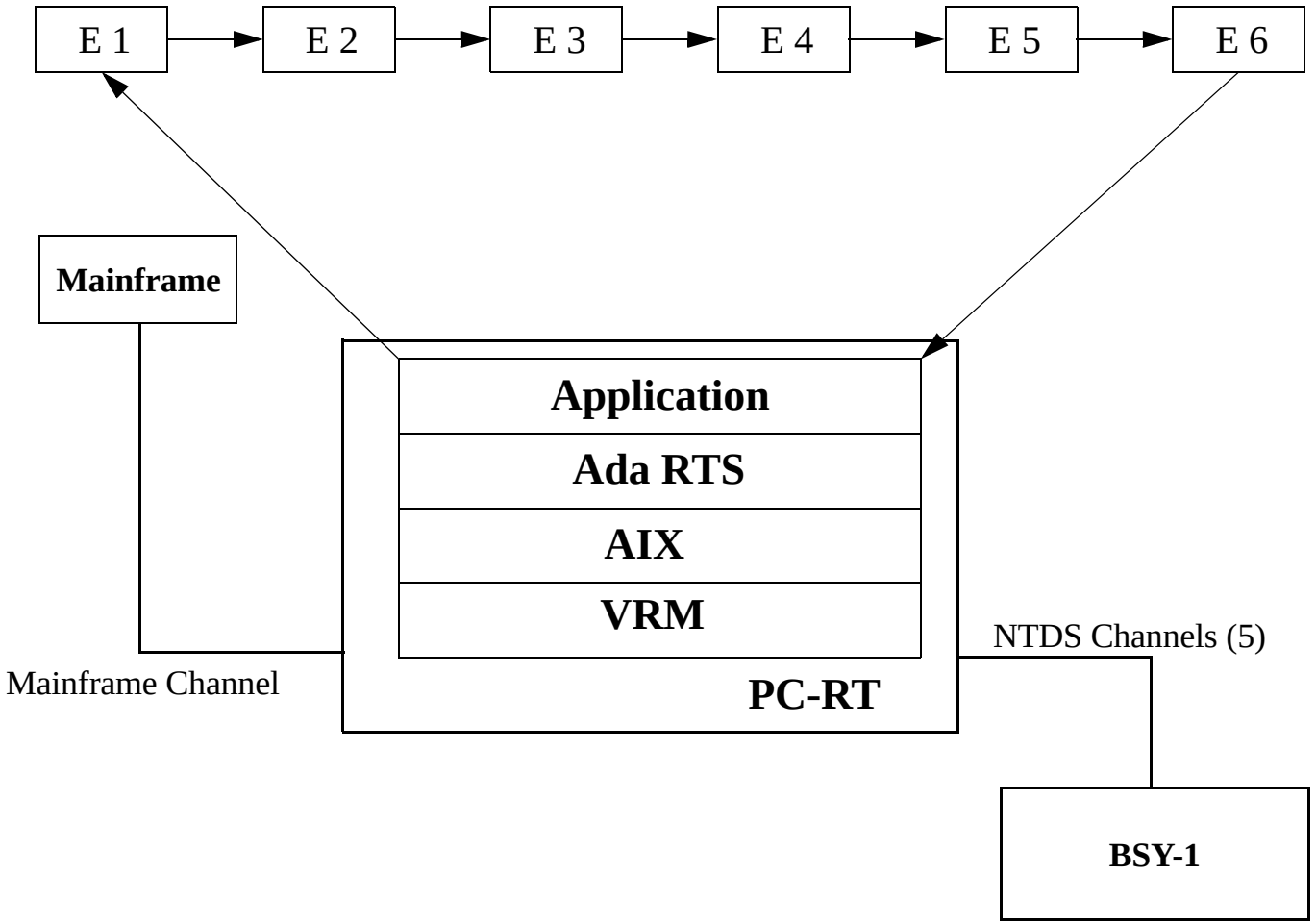


Notas:

The system is a sim/stim trainer for the BSY-1 Submarine Combat System. The BSY-1 is a sonar system on attack-class submarines. An IBM mainframe computer simulates the environment for training scenarios and generates the processed sonar data as the sonar hardware would have sent the data. The sonar data is sent to the BSY-1 system computers where the BSY-1 system software processes the data normally, allowing the crew to be trained in the use of the BSY-1 system.

The mainframe could not be directly connected to the BSY-1 hardware, so PC-RTs were used between the mainframe channels and the BSY-1 NTDS (Navy Tactical Data System) channels. The mainframe sends the data to PC-RTs which buffer the data and deliver it at the proper time to the BSY-1 computers. The PC-RTs are used because they have both the mainframe channel attachments and the Navy standard NTDS channels. The PC-RTs are programmed in Ada. The software that resides on the PC-RT is the real-time software of interest for this case study.

Software Design



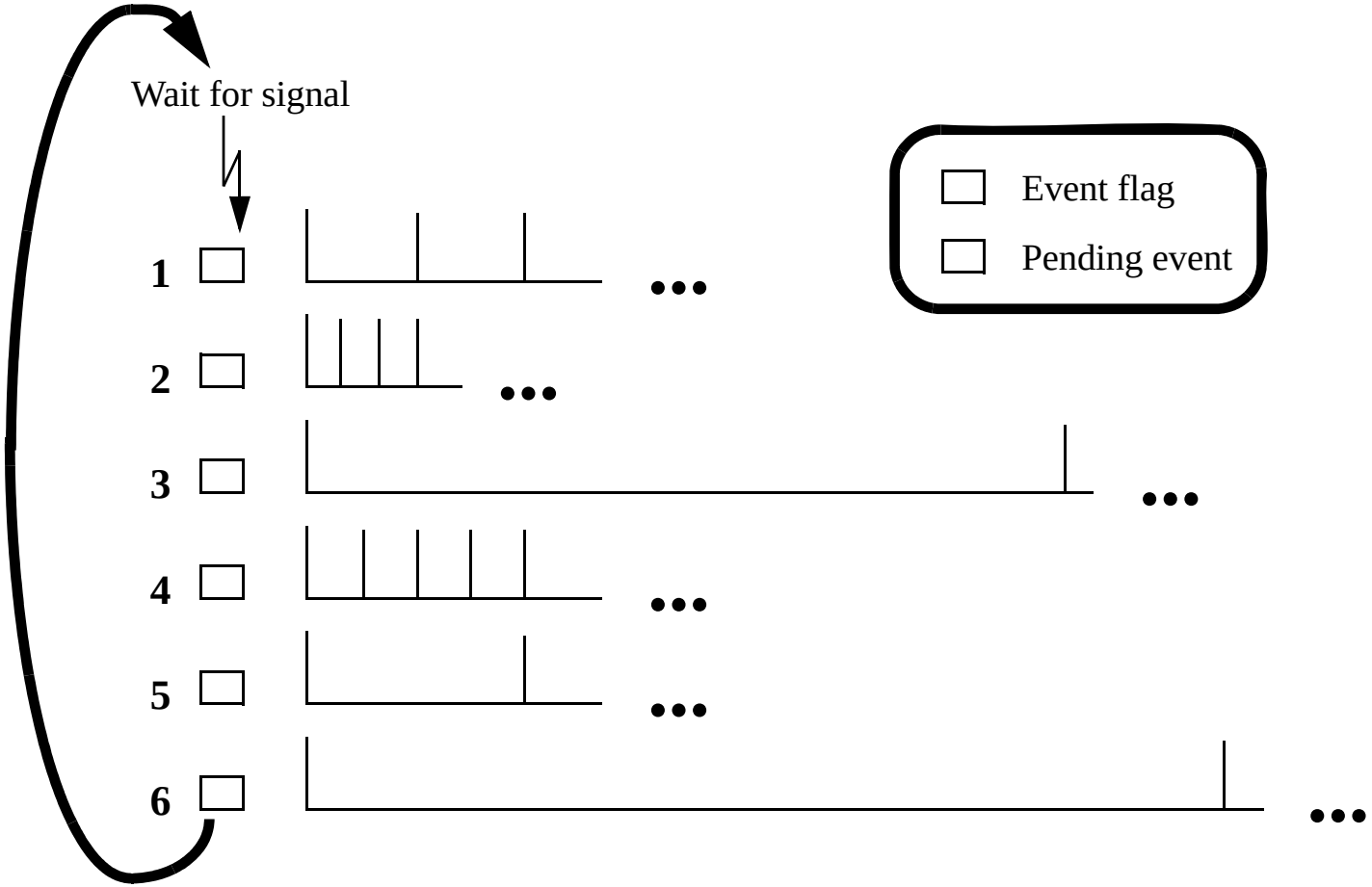
Notas:

The operating system is implemented in three levels, VRM (Virtual Resource Manager), AIX, and Ada. The VRM provides a virtual machine interface to AIX. AIX is a UNIX-like operating system. The Ada runtime system is built on AIX. The application code is written as a single Ada task being driven by a signal from the I/O drivers and an interval timer. (The application uses a round-robin scheduler, and events 1 through 6 are actually procedures executed in a fixed order. There is one event for each of 5 NTDS channels and one event for mainframe communications.) VRM-level device drivers were modified specifically for this project.

Data flows between the mainframe and the BSY-1 computers with different timing requirements. The mainframe channel operates at about a 4 Hz rate, alternating input and output actions. Each of the NTDS channels, 5 for this PC-RT, has timing requirements which reflect the type of data being transmitted.

All events are periodic. The application waits for an event signal from mainframe I/O, NTDS I/O, or timer expiration. Once a signal is received by the application code, event flags are checked to determine which events require processing. Event flags are checked in a fixed sequence. While events are being processed, new events may interrupt the application, be processed, and queue up a new signal. When the application has reacted to all events, it waits for another signal.

Scheduling Discipline



Notas:

UNIVERSIDAD DE CANTABRIA

First note that all events are periodic. The figure above shows that we have 6 periodic events of concern, each with a different period. Some events are I/O events and some are clock events. Each event results in an event flag being turned on. In the above figure, event flags 3, 5, and 6 have been turned on.

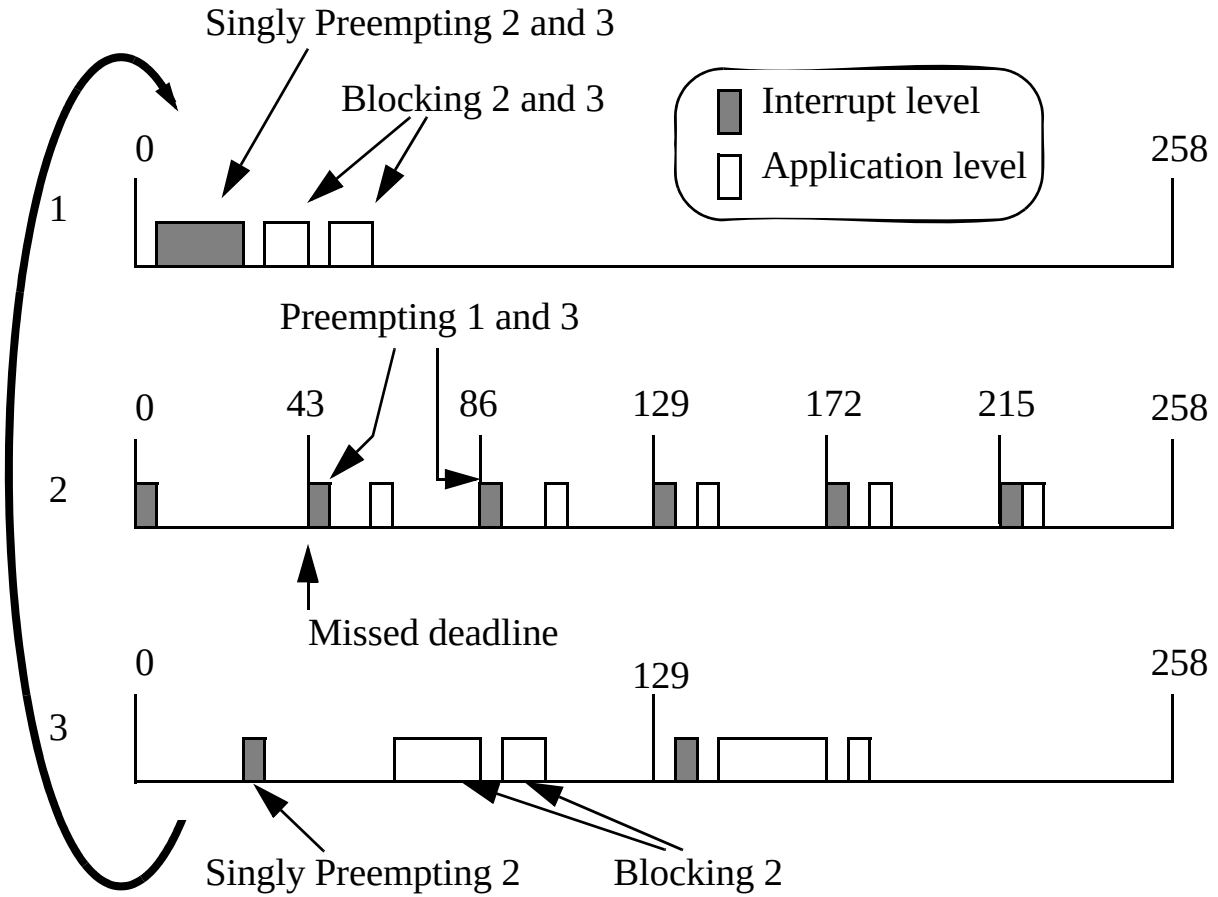
The scheduling policy can be summarized as follows:

- Application waits for signal event.
- When an event is signalled:

```
loop
    for I in 1..6 loop
        if flag(I) = True then
            -- perform processing for event I;
        end if;
    end loop;
end loop;
```

- While one event is being processed, other events may occur (and interrupts for those events will interrupt the computations in progress).
- The deadline for an event is considered to be the next arrival of that event.

Execution Sequence: Original Design



Notas:

Each event has interrupt-level and application-level processing associated with it. The interrupt-level processing is at the VRM level and is associated with one of the I/O channels. The interrupt handler sets a flag to tell the corresponding event handler it has work to do, and signals the single Ada task to run. The VRM signal is passed up through AIX and RTS to the application and triggers another cycle of computation.

This represents a hypothetical execution sequence for this design. Note that the implementation was comprised of a single Ada task. However we have separated the processing associated with each event in order to start understanding the effects that processing one event has on other events.

We can make several observations (to be confirmed later by actual data):

- Events are *not* processed in rate monotonic order.
- Application-level processing is nonpreemptive; that is, after an event is processed, it will not be checked again for processing until all other events have been checked (and processed, if event flags are set).
- Thus, high-frequency events can suffer blocking caused by nonpreemption, and excessive preemption due to non-rate-monotonic priorities.

Problem Analysis by Development Team

UNIVERSIDAD DE CANTABRIA

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.

Notas:

The lack of adequate throughput in the PC-RT was attributed to excessive overhead caused by two factors

1. Poor signal propagation from VRM through AIX to the Ada runtime.
2. Inefficient Ada runtime and poor code quality from the Ada compiler.

To address the first issue, the developers planned to move more application processing to the interrupt handler level to make it more responsive. They also planned to Improve the efficiency of AIX signals by not initiating a signal to the application when it is already cycling; rather the event flag is set.

The use of Ada was assumed to be the cause of performance problems. The planned approach was to recode 17,000 lines of Ada in 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.

Notas:

Project management initiated a rate monotonic analysis of the BSY-1 Trainer system.

- There were no apparent *C*'s and *T*'s because the application consisted of only one Ada task.
  - The application was modeled as six virtual periodic tasks: one task per communication channel event. Deadline for each event was the start of the next period.
- Each event had two components:
  - Application-level event handler (interruptible, but not preemptible)
  - VRM-level interrupt handler (not interruptible and not preemptible).

The executive provided round-robin execution for each event. Intensive measurement of the BSY-1 Trainer software revealed the data in the table. The table shows the application-level and interrupt-level processing times associated with each event and their utilizations. (The events are listed in rate monotonic order, not the original execution order.)

Note: $C = C_i + C_a$

C_i: interrupt processing time for event (VRM level).

C_a: application-level processing time for event.

C: total processing time for event.

Sched. Model: Original Design

(1) $\frac{C_1}{T_1} + \left\lceil \frac{C_2 + C_3 + C_4 + C_5 + C_6}{T_1} \right\rceil \leq U(1)$

(2) $\left\lceil \frac{C_{1,I}}{T_1} \right\rceil + \frac{C_2}{T_2} + \left\lceil \frac{C_{1,A} + C_3 + C_4 + C_5 + C_6}{T_2} \right\rceil \leq U(2)$

(3) $\left\lceil \frac{C_{1,I}}{T_1} + \frac{C_{2,I}}{T_2} \right\rceil + \frac{C_3}{T_3} + \left\lceil \frac{C_{1,A} + C_{2,A} + C_4 + C_5 + C_6}{T_3} \right\rceil \leq U(3)$

(4) $\left\lceil \frac{C_{1,I}}{T_1} + \frac{C_{2,I}}{T_2} + \frac{C_{3,I}}{T_3} \right\rceil + \frac{C_4}{T_4} + \left\lceil \frac{C_{1,A} + C_{2,A} + C_{3,A} + C_5 + C_6}{T_4} \right\rceil \leq U(4)$

(5) $\left\lceil \frac{C_{1,I}}{T_1} + \frac{C_{2,I}}{T_2} + \frac{C_{3,I}}{T_3} + \frac{C_{4,I}}{T_4} \right\rceil + \frac{C_5}{T_5} + \left\lceil \frac{C_{1,A} + C_{2,A} + C_{3,A} + C_{4,A} + C_6}{T_5} \right\rceil \leq U(5)$

(6) $\left\lceil \frac{C_{1,I}}{T_1} + \frac{C_{2,I}}{T_2} + \frac{C_{3,I}}{T_3} + \frac{C_{4,I}}{T_4} + \frac{C_{5,I}}{T_5} \right\rceil + \frac{C_6}{T_6} + \left\lceil \frac{C_{1,A} + C_{2,A} + C_{3,A} + C_{4,A} + C_{5,A}}{T_6} \right\rceil \leq U(6)$

Notas:

A schedulability model is a mathematical model based on the utilization bound test that predicts the schedulability of tasks. In order to construct a schedulability model, one must first identify all parameters that affect a task's ability to meet its deadline.

The above inequalities are a schedulability model. Note that we have arranged events in a rate monotonic order. Thus T_1 is less than T_2. Next we identify multiply preemptive, singly preemptive, and blocking effects. C_x represents the execution time for event x. This execution time is decomposed into an interrupt processing portion $(C_{x,I})$ and an application level portion $(C_{x,A})$. Where only C_x appears, $C_x = C_{x,I} + C_{x,A}$.

Consider the first event. Since a nonpreemptive, round-robin scheduling approach is used, it is possible that the highest-frequency event might have to wait for the completion of all lower-frequency events. The second term in the equation represents this as blocking.In general, for the events modeled above, the following is true:

- If the event has a shorter period than the event under consideration:
  - interrupt processing is modeled as preemption (and they really do preempt)
  - application processing is modeled as blocking (because they cannot preempt)
- If the event has a longer period than the event under consideration:
  - interrupt processing is modeled as single preemption, and application processing as blocking

Schedulability Test: Original Design

(1) $\frac{2,5}{43} + \left\lceil \frac{15,9 + 6,6 + 48,2 + 29,1 + 3,8}{43} \right\rceil$

(2) $\left\lceil \frac{2,0}{43} \right\rceil + \frac{15,9}{74} + \left\lceil \frac{(0,5) + 6,6 + 48,2 + 29,1 + 3,8}{74} \right\rceil$

(3) $\left\lceil \frac{2,0}{43} + \frac{7,4}{74} \right\rceil + \frac{6,6}{129} + \left\lceil \frac{(0,5 + 8,5) + 48,2 + 29,1 + 3,8}{129} \right\rceil$

(4) $\left\lceil \frac{2,0}{43} + \frac{7,4}{74} + \frac{6,0}{129} \right\rceil + \frac{48,2}{258} + \left\lceil \frac{(0,5 + 8,5 + 0,6) + 29,1 + 3,8}{258} \right\rceil$

(5) $\left\lceil \frac{2,0}{43} + \frac{7,4}{74} + \frac{6,0}{129} + \frac{21,5}{258} \right\rceil + \frac{29,1}{1032} + \left\lceil \frac{(0,5 + 8,5 + 0,6 + 26,7) + 3,8}{1032} \right\rceil$

(6) $\left\lceil \frac{2,0}{43} + \frac{7,4}{74} + \frac{6,0}{129} + \frac{21,5}{258} + \frac{5,7}{1032} \right\rceil + \frac{3,8}{4128} + \left\lceil \frac{(0,5 + 8,5 + 0,6 + 26,7 + 23,4)}{4128} \right\rceil$

Notas:

UNIVERSIDAD DE CANTABRIA

This is the schedulability test derived from plugging the data into the schedulability model. Utilizations are shown on the next slide.

Utilization: Original Design

	Period (msec)	Multiple Preempt	Execute	Blocking & single preempt	Total
Event 1	43	0.000	0.059	2.410	2.469
Event 2	74	0.047	0.215	1.192	1.454
Event 3	129	0.147	0.052	0.699	0.898
Event 4	258	0.194	0.187	0.165	0.546
Event 5	1032	0.278	0.029	0.039	0.346
Event 6	4128	0.284	0.001	0.015	0.300
Total			0.543		

The problem is excessive blocking and /or single preemption for events 1, 2, and 3.

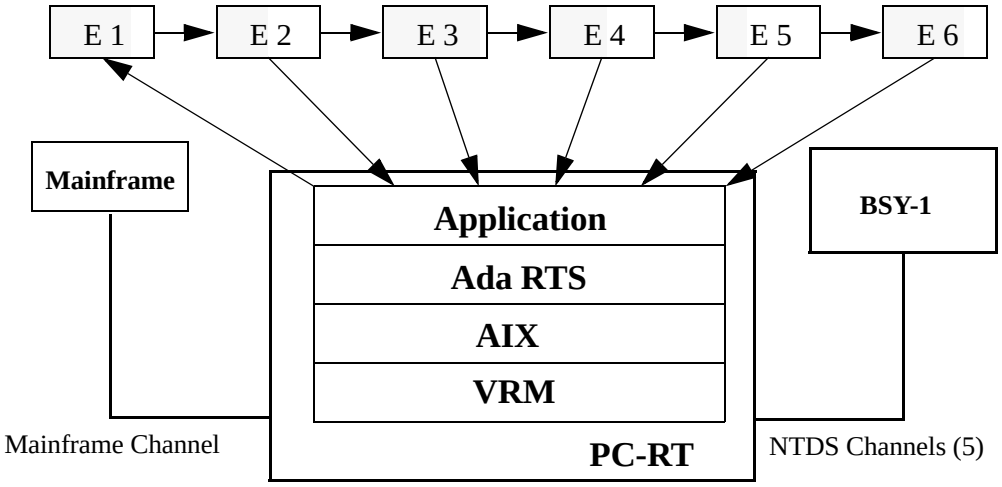
Notas:

This is a breakdown of the components of utilization for each event. It is easy to see why deadlines are being missed. Blocking and single preemption dominates the highest-rate events. Notice that single preemption usually means a depart from the optimum priority assignment, and should therefore be minimized, the same as blocking.

To achieve a schedulable system, we must find ways to reduce blocking and single 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



Notas:

Since the size of the blocking term for the first event is the reason that this event is not schedulable, the first action taken was to reduce the size of this term by eliminating the round-robin scheduler. Rather than process several events in a fixed order, the logic was changed to exit the loop and recheck the event flags after the processing of **each** event. If work needed to be done for the highest-rate event, that event was processed, and the remaining flags were checked in rate monotonic order. The processing of each event was still nonpreemptible; but RM priorities were now observed.

```
loop    -- note that events have been reordered to rate monotonic order
    for I in 1..6
        loop
            if flag(I) = True then
                -- perform processing for event I;
                exit;
            end if;
        end loop;
end loop;
```

Now a higher-rate event (i.e., an event with a shorter period) can be blocked by at most one lower-rate application-level event. Since events are processed in a rate monotonic order, the largest contribution to blocking because of application-level processing is the *maximum* application-level execution time of lower-rate events.

Sched. Model for Events in RM Order

UNIVERSIDAD DE CANTABRIA

(1) $$\frac{C_1}{T_1} + \left\lceil \frac{max(C_{2,A}, C_{3,A}, C_{4,A}, C_{5,A}, C_{6,A}) + C_{2,I} + C_{3,I} + C_{4,I} + C_{5,I} + C_{6,I}}{T_1} \right\rceil$$

(2) $$\left\lceil \frac{C_1}{T_1} \right\rceil + \frac{C_2}{T_2} + \left\lceil \frac{max(C_{3,A}, C_{4,A}, C_{5,A}, C_{6,A}) + C_{3,I} + C_{4,I} + C_{5,I} + C_{6,I}}{T_2} \right\rceil$$

(3) $$\left\lceil \frac{C_1}{T_1} + \frac{C_2}{T_2} \right\rceil + \frac{C_3}{T_3} + \left\lceil \frac{max(C_{4,A}, C_{5,A}, C_{6,A}) + C_{4,I} + C_{5,I} + C_{6,I}}{T_3} \right\rceil$$

(4) $$\left\lceil \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} \right\rceil + \frac{C_4}{T_4} + \left\lceil \frac{max(C_{5,A}, C_{6,A}) + C_{5,I} + C_{6,I}}{T_4} \right\rceil$$

(5) $$\left\lceil \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \frac{C_4}{T_4} \right\rceil + \frac{C_5}{T_5} + \left\lceil \frac{C_6}{T_5} \right\rceil$$

(6) $$\left\lceil \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \frac{C_4}{T_4} + \frac{C_5}{T_5} \right\rceil + \frac{C_6}{T_6}$$

Notas:

Consider the equation for the first event. Since interrupts for other events occur at a lower rate than event 1's rate, the interrupt processing time for each such event acts as part of event 1's single-preemption term. These times are summed to the blocking term because in the worst case, an interrupt for each lower-rate event will arrive once while processing event 1. However, because of the change in how application-level processing is performed, event 1 will only be delayed by the largest application-level processing time of all the lower-rate events. This applies similarly to the other 5 events.

Blocking terms for the other events are arrived at in the same way. Interrupt processing time for each lower-rate event acts as a part of the single-preemption term. However, an event will be prevented from executing (blocked) by at most the maximum processing time for a single lower-rate event.

Now consider the multiple-preemption terms for the other events. The schedulability model for the original design included only interrupt-level processing in this term. Now we include application-level processing because, in principle, application-level processing for a high-rate event, e.g., event 1, can be executed more than once before application-level execution of a lower-rate event, such as, event 5, is allowed to start. (This is somewhat pessimistic, however, since this form of equation assumes that the application-level processing of higher-rate events preempt the application-level processing of lower-rate events. There is actually no preemption in this system. For example, once the application-level processing of event 5 starts, it cannot be preempted by application-level processing of a higher-rate event. However, as we later introduce changes that increase the amount of preemption that can occur, our model will become more accurate.)

Sched. Test: Events in RM Order

(1) $$\frac{2,5}{43} + \left\lceil \frac{(26,7) + 7,4 + 6,0 + 21,5 + 5,7 + 2,8}{43} \right\rceil$$

(2) $$\left\lceil \frac{2,5}{43} \right\rceil + \frac{15,9}{74} + \left\lceil \frac{(26,7) + 6,0 + 21,5 + 5,7 + 2,8}{74} \right\rceil$$

(3) $$\left\lceil \frac{2,5}{43} + \frac{15,9}{74} \right\rceil + \frac{6,6}{129} + \left\lceil \frac{(26,7) + 21,5 + 5,7 + 2,8}{129} \right\rceil$$

(4) $$\left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} \right\rceil + \frac{48,2}{258} + \left\lceil \frac{(23,4) + 5,7 + 2,8}{258} \right\rceil$$

(5) $$\left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} \right\rceil + \frac{29,1}{1032} + \left\lceil \frac{3,8}{1032} \right\rceil$$

(6) $$\left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} + \frac{29,1}{1032} \right\rceil + \frac{3,8}{4128}$$

Notas:

UNIVERSIDAD DE CANTABRIA

If the sum of the *C*s for the blocking and singlepreemption (second) term of the first equation is still greater than 43, the term for the first event has not yet been reduced adequately.

The numerical values for multiple preemption, execution, and single-preemption and blocking terms for each event are summarized in the following table.

Utilization: Process Events in RM Order

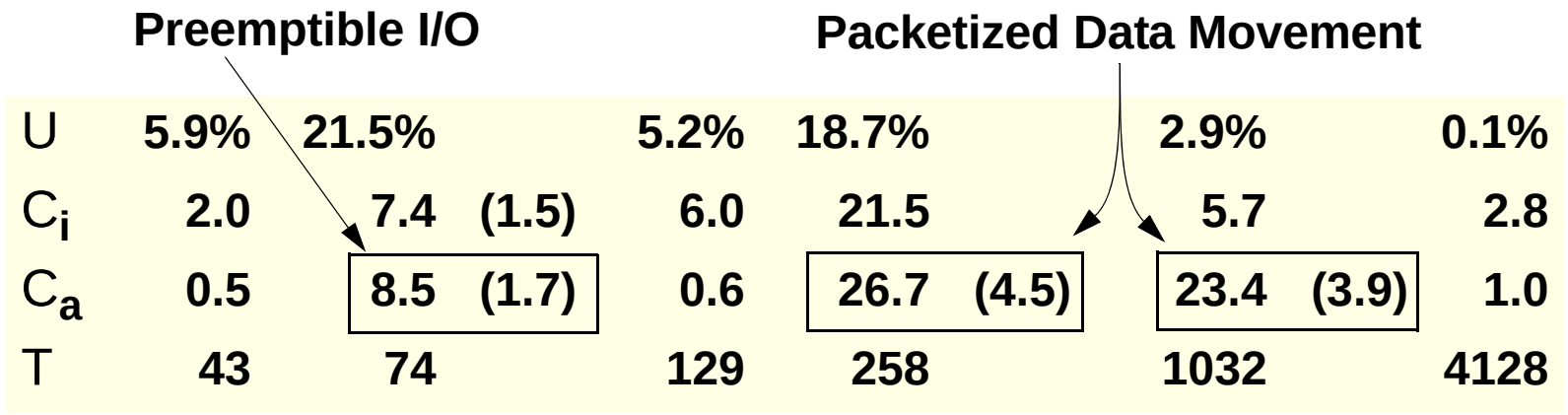
	Period (msec)	Multiple Preempt	Execute	Blocking & Single Preempt	Total	Previous Total
Event 1	43	0.000	0.059	1.631	1.690	2.469
Event 2	74	0.059	0.215	0.848	1.122	1.454
Event 3	129	0.274	0.052	0.440	0.766	0.898
Event 4	258	0.326	0.187	0.124	0.637	0.546
Event 5	1032	0.513	0.029	0.004	0.546	0.346
Event 6	4128	0.542	0.001	0.000	0.543	0.300
Total			0.543			

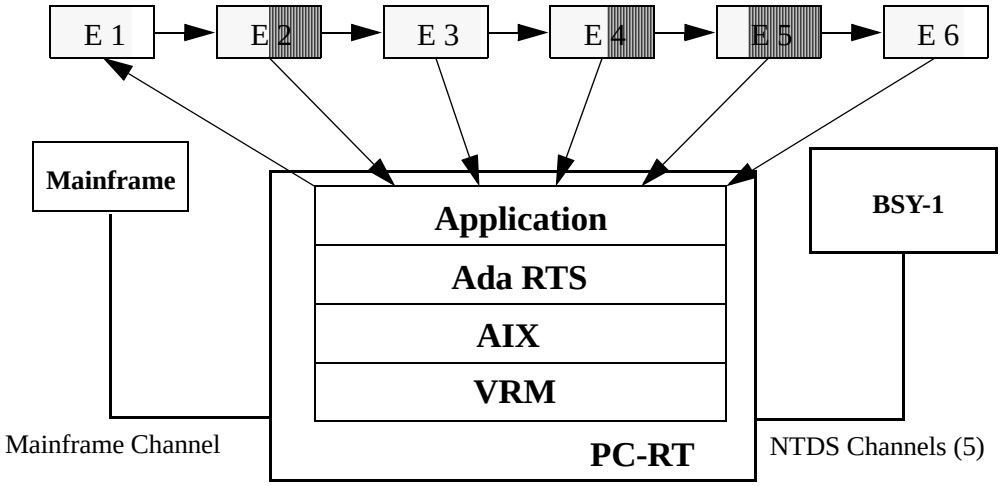
Notas:

UNIVERSIDAD DE CANTABRIA

This slide includes two improvements: rechecking flags after each event and checking those flags in rate monotonic order.

Increasing Preemptibility





Notas:

UNIVERSIDAD DE CANTABRIA

Packetized Data Movement:

The next step was to try to reduce the blocking term from application-level processing. Examining the logic for events 4 and 5 showed that most of the time was taken in moving large blocks of data. Instead of moving all this data in a single operation, it was decided to move 1K words at a time. If the processing is not complete after moving 1K words, then a flag is set indicating that more processing is required and the application-level scheduler then allows higher-rate events to be processed. The next time that this event is eligible for processing, another 1K packet will be moved and so on. By packetizing the processing, a high-rate event is blocked only for the time needed to move 1K words instead of being blocked for the entire data movement. Specifically, it takes 4.5 msec to move 1K words for the 258 msec event and 3.9 msec to move 1K words for the 1032 msec event.

Preemptible I/O (Changing asynchronous I/O to synchronous I/O):

Packetizing data movement leaves the 8.5 msec term in the 74 msec event as the highest C_a value. This event started several asynchronous I/O operations simultaneously and then waited for them all to complete. This is more efficient in terms of maximizing I/O rates, but the result is longer blocking for the higher-rate events. Changing to synchronous I/O (one operation completes before the next is started) and checking to see if higher-rate events need to be serviced after each I/O completion in effect increases the number of preemption points. The application-level processing associated with starting an individual NTDS I/O is 1.7 ms. There is an added benefit that the interrupt processing for E2 is really 5 interrupts, but only 1 may occur at a time.

Sched. Test: Packetized Data Movement &Preemptible I/O

UNIVERSIDAD DE CANTABRIA

$$(1)\ \frac{2,5}{43} + \left\lceil \frac{(4,5) + 1,5 + 6,0 + 21,5 + 5,7 + 2,8}{43} \right\rceil$$

$$(2)\ \left\lceil \frac{2,5}{43} \right\rceil + \frac{15,9}{74} + \left\lceil \frac{(4,5) + 6,0 + 21,5 + 5,7 + 2,8}{74} \right\rceil$$

$$(3)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} \right\rceil + \frac{6,6}{129} + \left\lceil \frac{(4,5) + 21,5 + 5,7 + 2,8}{129} \right\rceil$$

$$(4)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} \right\rceil + \frac{48,2}{258} + \left\lceil \frac{(3,9) + 5,7 + 2,8}{258} \right\rceil$$

$$(5)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} \right\rceil + \frac{29,1}{1032} + \left\lceil \frac{3,8}{1032} \right\rceil$$

$$(6)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} + \frac{29,1}{1032} \right\rceil + \frac{3,8}{4128}$$

Notas:

The schedulability model has not changed. We simply plug in the new worst-case blocking times. The previous maximum blocking on events 1, 2, and 3 (from application level processing) was 26.7 msec. Because of packetizing, the maximum blocking on those events (from application processing) is now 4.5 msec.

Utilization: Packetized Data Movement & Preemptible I/O

	Period (msec)	Multiple Preempt	Execute	Blocking & Single Preempt	Total	Previous Total
Event 1	43	0.000	0.059	0.977	1.035	1.690
Event 2	74	0.059	0.215	0.548	0.822	1.122
Event 3	129	0.274	0.052	0.268	0.594	0.766
Event 4	258	0.326	0.187	0.049	0.562	0.637
Event 5	1032	0.513	0.029	0.004	0.546	0.546
Event 6	4128	0.542	0.001	0.000	0.543	0.543
Total			0.543			

All events now are schedulable, except event 1.

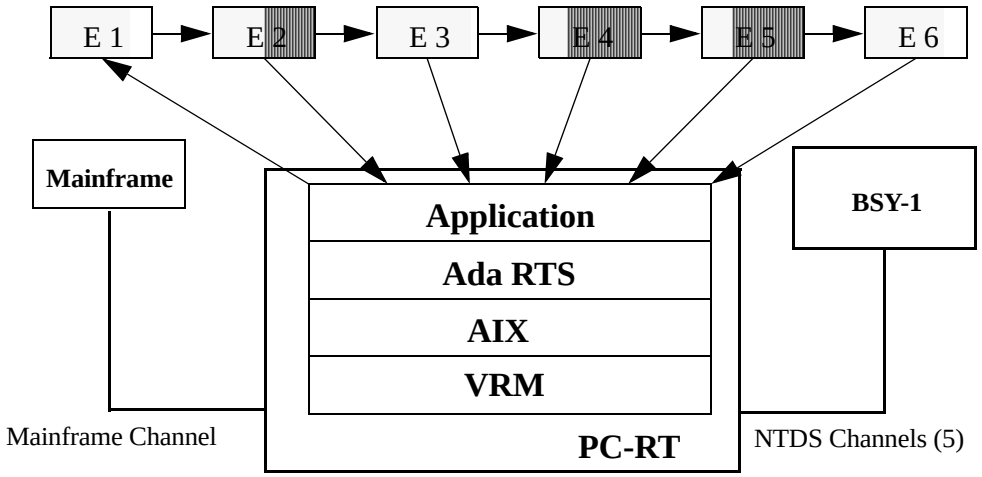
Notas:

The above table shows the numerical values for multiple preemption, execution, and blocking and single preemption for each event, assuming packetized data movement for events 4 and 5 and preemptible input/ output for event 2.

We are closer to reducing the blocking and single preemption term to an acceptable level but still not there. Interrupt processing for event 4 contributes the largest term, so the next step will be to examine that.

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



Notas:

UNIVERSIDAD DE CANTABRIA

There is a very high value in the interrupt processing time for event 4 that causes the first event to fail its schedulability test. As discussed earlier, when high levels of interrupt processing time cause high-rate events to miss their deadlines, one approach is to move some of the interrupt processing to a lower-priority application-level event.

In the BSY-1 Trainer case, the interrupt processing for event 4 consisted of moving large blocks of data. The logic was changed so that a flag was set indicating the presence of data to be moved and the data moving operation was added to the application-level processing for event 4. In addition, the data movement continued to be packetized so that no segment of processing took more than 4.5 msec. About 15.0 msec of work was moved to the application level in this way, reducing the total interrupt processing time to 6.5 msec while increasing the total application time to 26.7 + 15.0 = 41.7. This reduced the single-preemption term for higher-rate events but left the multiple-preemption terms for lower-priority events unchanged, since the total amount of processing time for event 4 was still the same.

This shows the final structure of the system, in which some of the interrupt processing for event 4 was moved to the application level, thereby reducing the single-preemption term for the highest-priority event.

Schedulability Test: Streamlined Interrupt Hdler.

UNIVERSIDAD DE CANTABRIA

$$(1)\ \frac{2,5}{43} + \left\lceil \frac{(4,5) + 1,5 + 6,0 + 6,5 + 5,7 + 2,8}{43} \right\rceil$$

$$(2)\ \left\lceil \frac{2,5}{43} \right\rceil + \frac{15,9}{74} + \left\lceil \frac{(4,5) + 6,0 + 6,5 + 5,7 + 2,8}{74} \right\rceil$$

$$(3)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} \right\rceil + \frac{6,6}{129} + \left\lceil \frac{(4,5) + 6,5 + 5,7 + 2,8}{129} \right\rceil$$

$$(4)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} \right\rceil + \frac{48,2}{258} + \left\lceil \frac{(3,9) + 5,7 + 2,8}{258} \right\rceil$$

$$(5)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} \right\rceil + \frac{29,1}{1032} + \left\lceil \frac{3,8}{1032} \right\rceil$$

$$(6)\ \left\lceil \frac{2,5}{43} + \frac{15,9}{74} + \frac{6,6}{129} + \frac{48,2}{258} + \frac{29,1}{1032} \right\rceil + \frac{3,8}{4128}$$

Notas:

This is the full set of equations for the final design.

Utilization: Streamlined Interrupt Hdler.

	Period (msec)	Multiple Preempt	Execute	Blocking & Single Preempt	Total	Previous Total
Event 1	43	0.000	0.059	0.628	0.687	1.035
Event 2	74	0.059	0.215	0.345	0.619	0.822
Event 3	129	0.274	0.052	0.152	0.478	0.594
Event 4	258	0.326	0.187	0.049	0.562	0.562
Event 5	1032	0.513	0.029	0.004	0.546	0.546
Event 6	4128	0.542	0.001	0.000	0.543	0.543
Total			0.543			

Notas:

Reducing the computation done in the interrupt handler results in a schedulable system. The BSY-1 Trainer System now passes the utilization bound test.

Summary: BSY-1 Trainer Case Study

UNIVERSIDAD DE CANTABRIA

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

Notas:

Rate monotonic analysis pointed to the changes required to reducing blocking effects.

- One change was to get the round-robin scheduler to approximate a preemptive, rate-monotonic scheduler by checking, after each event was processed, if work needed to be done by a faster event.
- Another change that reduced blocking was breaking the work done by lower-frequency tasks into smaller pieces (breaking big nonpreemptible sections into smaller nonpreemptible sections) that allowed the higher-frequency events to meet their deadlines.
- To further reduce delay effects due to excessive preemption, computations were moved from interrupt handlers to the application code.

Some observations:

- This involved changing about 300 lines of Ada (as compared to converting 17,000 lines of Ada to C).
- Conversion to C would have been attacking the wrong problem.
- Moving more computation to interrupt handlers, as originally contemplated, would have made matters worse.
- The analysis required 3 people during a 3-week period. Consider the amount of time required for changing 17,000 lines of code, compared to 300.