
Analysis of Embedded Real-Time Systems Project Report Final

ABSTRACT

This project report describes the results of analyzing the properties of various task sets using rate-monotonic analysis and real-time synchronization protocols used to control priority inversion. The study includes an analysis of the Controller Area Network bus and the modeling of message-passing systems as task sets. Much of the work for this report was performed using TimeWiz, a commercial product developed by TimeSys Corporation.

Copyright © 2001 by Chris Lord. All Rights Reserved.

Authors:	Chris Lord
Software Version:	Not Applicable
Document Version:	0.0.100
Edit Number:	220
Last Revised:	12/13/2001 11:28 PM
Classification:	Internal Use Only
Location:	/afs/clord/Documents/18-349/Lab 5 Project Report.doc
Date Printed:	5/7/2002 10:41 PM

Table Of Contents

Part A Worst-Case Scenarios for RMS..... 1

Part B Good Scenarios for RMS 2

 Part B.a Harmonic Task Sets with 100% Utilization..... 2

 Part B.b Non-Harmonic Task Sets with 100% Utilization..... 3

 Part B.c Missing Deadlines Below 100% Utilization..... 4

Part C Comparison of Synchronization Protocols..... 5

 Part C.a Schedulable Task Set Under All Protocols..... 5

 Part C.b Schedulability Under PIP and PCP but not NPP 6

 Part C.c Schedulability Under HLP but not PIP, NPP..... 7

Part D Analysis of the Controller Area Network Bus..... 8

 Part D.a Non-Preemptive Message Transmission..... 8

 Part D.b A Design Schedulable at .5Mbs but not .25Mbps..... 8

Part E Analyzing Multiprocessor Environments..... 9

Part F Custom Dinner Time Conversation Catalog..... 10

 Part F.a The “Dinner Table” 10

 Part F.b The Participants 11

 Part F.c The Messages..... 11

 Part F.d Hardware Diagram..... 11

 Part F.e Software Diagram..... 12

Part G Problems and Challenges..... 13

Part A Worst-Case Scenarios for RMS

This section examines non-ideal cases under rate-monotonic scheduling (RMS). TimeWiz provides a useful environment for modeling RMS systems, but it doesn’t provide quite the flexibility required to create task sets as close as possible to the utilization bound given by $n(2^{1/n}-1)$ where n is the number of tasks being scheduled. The selection of five task sets for $n=\{4, 5, 6, 7, 8\}$ was facilitated with an Excel worksheet which calculated the upper bound and then used the solver capability to adjust execution times and periods until the upper bound was reached. The spreadsheet used is included in the adjacent embedded worksheet object.

Priority	Task	C	T	U	Utilization Bound Test		
					U	U(n)	Scheduable
High	1	2	44	0.045	0.04545	1.00000	Yes
	2	2	49	0.041	0.08627	0.82843	Yes
	3	2	50	0.040	0.12627	0.77976	Yes
	4	7	55	0.127	0.25354	0.75683	Yes
	5	9	57	0.158	0.41142	0.74349	Yes
	6	9	64	0.141	0.55204	0.73477	Yes
	7	5	68	0.074	0.62556	0.72863	Yes
Low	8	9	93	0.097	0.72233	0.72406	Yes

The resultant tasks are given by the following table with tasks listed from high to low priority such that tasks with shorter periods are always higher priority. It should be noted that there are many possible sets that closely approximate the utilization bound depending on which parameters (execution times and periods) the worksheet is allowed to modify in its goal-seeking.

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
4	1	5	19	0.263	0.7567	0.7568
	2	5	27	0.185		
	3	5	31	0.161		

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
	4	5	34	0.147		
5	1	3	7	0.429	0.7427	0.7435
	2	1	8	0.125		
	3	1	11	0.091		
	4	1	12	0.083		
	5	1	67	0.015		
6	1	1	4	0.250	0.7345	0.7348
	2	1	5	0.200		
	3	1	7	0.143		
	4	1	9	0.111		
	5	1	65	0.015		
7	1	8	85	0.094	0.7284	0.7286
	2	10	86	0.116		
	3	10	87	0.115		
	4	10	88	0.114		
	5	10	89	0.112		
	6	10	90	0.111		
8	1	2	44	0.045	0.7223	0.7241
	2	2	49	0.041		
	3	2	50	0.040		
	4	7	55	0.127		
	5	9	57	0.158		
	6	9	64	0.141		
	7	5	68	0.074		
	8	9	93	0.097		

Part B Good Scenarios for RMS

There are several approaches to create task sets that provide 100% utilization and are still schedulable under RMS. The simplest is to create tasks with harmonic periods (integral multiples of a fundamental frequency). This is done by selecting periods which conform to a series such as $T_i = 2^{i-1}n$ and selecting execution times which are then $1/n$ of this so that $C_i = 2^{i-1}$. This ensures that all execution times and periods are integral values and that the total utilization is 100% to an accuracy limited only by the clock granularity.

Part B.a Harmonic Task Sets with 100% Utilization

The task sets were constructed as described above such that $\tau_i = (2^{i-1}, 2^{i-1}n)$ for $n \in \{7, 13, 21\}$ and are shown in the table below (tasks are shown from high to low priority). These tasks were entered into a TimeWiz software diagram consisting of n triggers and n actions with the specified C_i and T_i values to confirm RMS schedulability.

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
7	1	1	7	0.14286	1.000	0.7286
	2	2	14	0.14286		
	3	4	28	0.14286		
	4	8	56	0.14286		
	5	16	112	0.14286		
	6	32	224	0.14286		
	7	64	448	0.14286		
13	1	1	13	0.07692	1.000	0.7120
	2	2	26	0.07692		
	3	4	52	0.07692		
	4	8	104	0.07692		
	5	16	208	0.07692		
	6	32	416	0.07692		
	7	64	832	0.07692		
	8	128	1664	0.07692		

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
	9	256	3328	0.07692	1.000	0.7047
	10	512	6656	0.07692		
	11	1024	13312	0.07692		
	12	2048	26624	0.07692		
	13	4096	53248	0.07692		
21	1	1	21	0.04762		
	2	2	42	0.04762		
	3	4	84	0.04762		
	4	8	168	0.04762		
	5	16	336	0.04762		
	6	32	672	0.04762		
	7	64	1344	0.04762		
	8	128	2688	0.04762		
	9	256	5376	0.04762		
	10	512	10752	0.04762		
	11	1024	21504	0.04762		
	12	2048	43008	0.04762		
	13	4096	86016	0.04762		
	14	8192	172032	0.04762		
	15	16384	344064	0.04762		
	16	32768	688128	0.04762		
17	65536	1376256	0.04762			
18	131072	2752512	0.04762			
19	262144	5505024	0.04762			
20	524288	11010048	0.04762			
21	1048576	22020096	0.04762			

The result of such a harmonic distribution is shown in the following timeline for $n=7$.

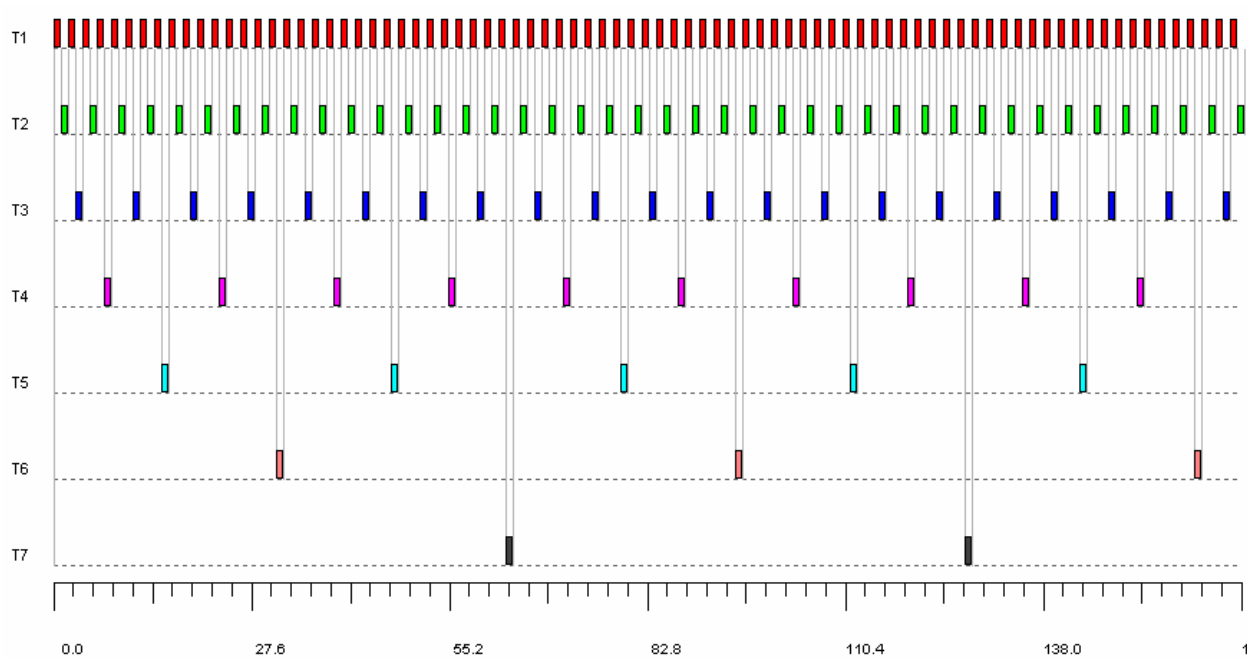


Figure B-1: 100% Utilization Using RMS and Harmonic Periods

Part B.b Non-Harmonic Task Sets with 100% Utilization

The approach above uses harmonic periods to ensure RMS schedulability of 100% utilizations. It is also possible to use harmonic frequencies to achieve the same end. This approach is distinct from the harmonic periods discussed

above and is based on calculating a period from a fundamental frequency. The task sets used a frequency of 0.01 and calculated the task parameters using the following:

$$\tau_i = \left(\frac{T_i}{n}, \frac{1}{n - (i-1)f} \right)$$

The task sets for $n \in \{3, 6\}$ and are shown in the table below. These tasks were entered into a TimeWiz software diagram consisting of n triggers and n actions with the specified C_i and T_i values to confirm RMS scheduability. They were also confirmed using the Completion Time test in the Excel worksheet.

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
3	1	11.1111	33.3333	0.33333	1.000	0.7798
	2	16.6667	50.0000	0.33333		
	3	33.3333	100.0000	0.33333		
6	1	2.7778	16.6667	0.16667	1.000	0.7348
	2	3.3333	20.0000	0.16667		
	3	4.1667	25.0000	0.16667		
	4	5.5556	33.3333	0.16667		
	5	8.3333	50.0000	0.16667		
	6	16.6667	100.0000	0.16667		

The timeline for $n=6$ is shown in the figure below. This shows clearly how this approach does not yield harmonic periods (which were prohibited in this section) yet achieves 100% utilization using RMS.

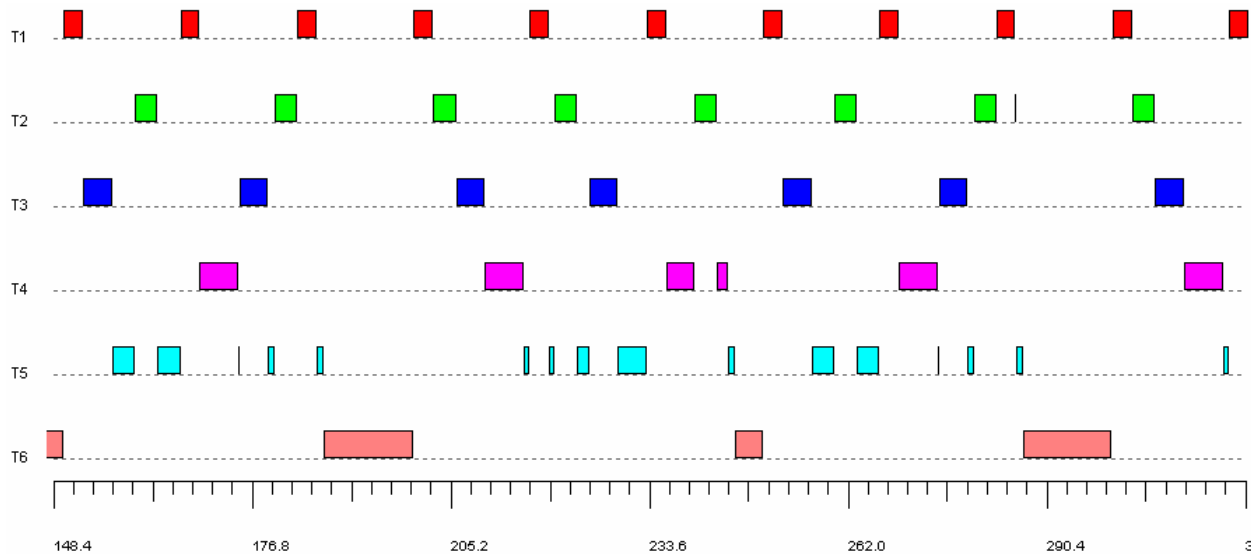


Figure B-2: 100% Utilization Using RMS and Non-Harmonic Periods

Part B.c Missing Deadlines Below 100% Utilization

In the task set given above for $n=6$, simply changing the value of T_i from 16.667 to 17 and C_i from 2.778 to 2.78 makes this task set unscheduable using RMS as confirmed using the Completion Time test. Likewise, slight rounding to obtain a utilization less than 100% on task set $n=4$ also resulted in unscheduable tasks.

n	i	C_i	T_i	U_i	ΣU_i	$U(n)$
3	1	11.0000	33.0000	0.33333	0.9967	0.7798
	2	16.6667	50.0000	0.33333		
	3	33.0000	100.0000	0.33000		
6	1	2.7800	17.0000	0.16353	0.9969	0.7348
	2	3.3333	20.0000	0.16667		

	3	4.1667	25.0000	0.16667		
	4	5.5556	33.3333	0.16667		
	5	8.3333	50.0000	0.16667		
	6	16.6667	100.0000	0.16667		

Part C Comparison of Synchronization Protocols

This section compares and contrasts three real-time synchronization protocols used to bound and minimize priority inversion: the (basic) priority inheritance protocol (PIP), the priority ceiling protocol (PCP) and the highest locker priority protocol (HLP), and the non-preemption protocol (NPP).

Part C.a Schedulable Task Set Under All Protocols

This section compares and contrasts the Priority Inheritance Protocol (PIP), the Priority Ceiling Protocol (PCP), the Highest Locker First (HLP), and the Non-Preemption Protocol (NPP). The table below compares the worst-case execution times for all four protocols using the same basic task set.

i	T_i	C_i	Resources	Priority	Worst Case Execution Times				U_i
					PIP	PCP	HLP	NPP	
1a	50	5		4	27	13	13	13	0.1
2a	100	5		6	31	21	21	21	0.05
3a	150	5		8	43	35	35	35	0.0333333
4a	200	5		10	47	47	47	47	0.025
5a	250	5		12	65	65	65	69	0.02
6a	300	5		14	87	87	87	87	0.0166667
7a	350	5		16	97	97	97	97	0.0142857
8a	400	5		18	133	133	133	133	0.0125
1b		2	M1,M2,M3,M4	4	29	15	15	15	0.04
2b		4	M2	6	35	25	25	25	0.04
3b		2	M1	8	45	37	37	37	0.0133333
4b		8	M3,M4	10	63	63	63	63	0.04
5b		8	M3,M4	12	73	73	73	77	0.032
6b		4	M5	14	91	91	91	91	0.0133333
7b		4	M5	16	123	123	123	123	0.0114286
8b		5		18	138	138	138	138	0.0125
1c		1		4	30	16	16	16	0.02
2c		5		6	40	30	30	30	0.05
3c		5		8	50	42	42	42	0.0333333
4c		5		10	68	68	68	68	0.025
5c		5		12	78	78	78	82	0.02
6c		5		14	96	96	96	96	0.0166667
7c		5		16	128	128	128	128	0.0142857
8c		5		18	143	143	143	143	0.0125

The time line for the execution of the PIP, PCP and HLP protocols is identical and is shown below.

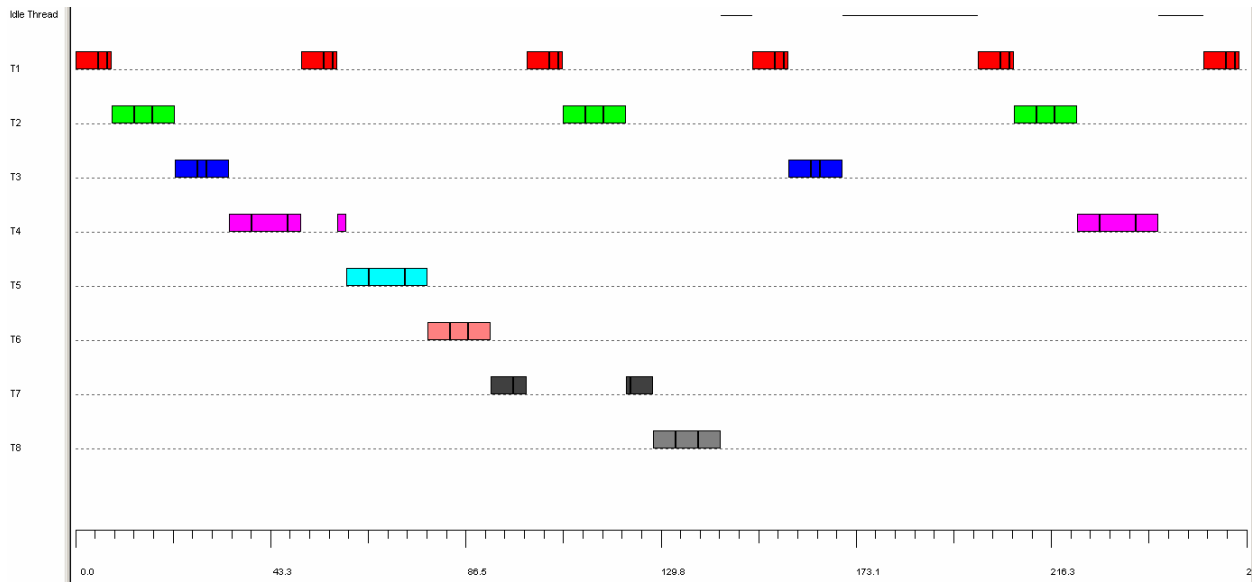


Figure C-1: PIP Timeline for Task Set Schedulable under PIP, PCP, HLP and NPP

Using NPP (which is called Interrupt Masking in the current version of TimeWiz, not Kernel Priority Protocol) provides different behavior: there is no preemption of each executable interval (each task consists of execution outside, inside and outside a critical section). The worst case for each of the tasks is only slightly different as can be seen in the following table which adds the worst-case execution times for each of the execution stages in the tasks under each of the synchronization protocols.

<i>i</i>	T_i	C_i	Priority	Worst Case Execution Time			
				PIP	PCP	HLP	NPP
1	50	8	4	86	44	44	44
2	100	14	6	106	76	76	76
3	150	12	8	138	114	114	114
4	200	18	10	178	178	178	178
5	250	18	12	216	216	216	228
6	300	14	14	274	274	274	274
7	350	14	16	348	348	348	348
8	400	15	18	414	414	414	414

All of these tasks are schedulable because the utilization, 0.666167, is less than the bound, 0.72406 necessary to schedule these under RMS.

Part C.b Schedulability Under PIP and PCP but not NPP

The first step to create a task set schedulable under PIP and PCP but not NPP was to create a boundary condition that was schedulable under all synchronization protocols.

<i>i</i>	T_i	Early Execution	Critical Section		Late Execution
		C_{ia}	C_{ib}	Resources	C_{ic}
1	50	5	9	M1,M2,M3,M4	1
2	100	3	18	M2	2
3	150	8	12	M1	2
4	200	2.5	2	M3,M4	2
5	250	5	3	M3,M4	2
6	300	2	40	M5	5
7	350	5	2	M5	2
8	400	4	2		2

The second step was to take advantage of the fact that NPP will not preempt an integral execution interval to extend a low priority interval to deliberately cause a higher priority task to miss its deadline. This can be seen in task 6 above and in the timeline below.

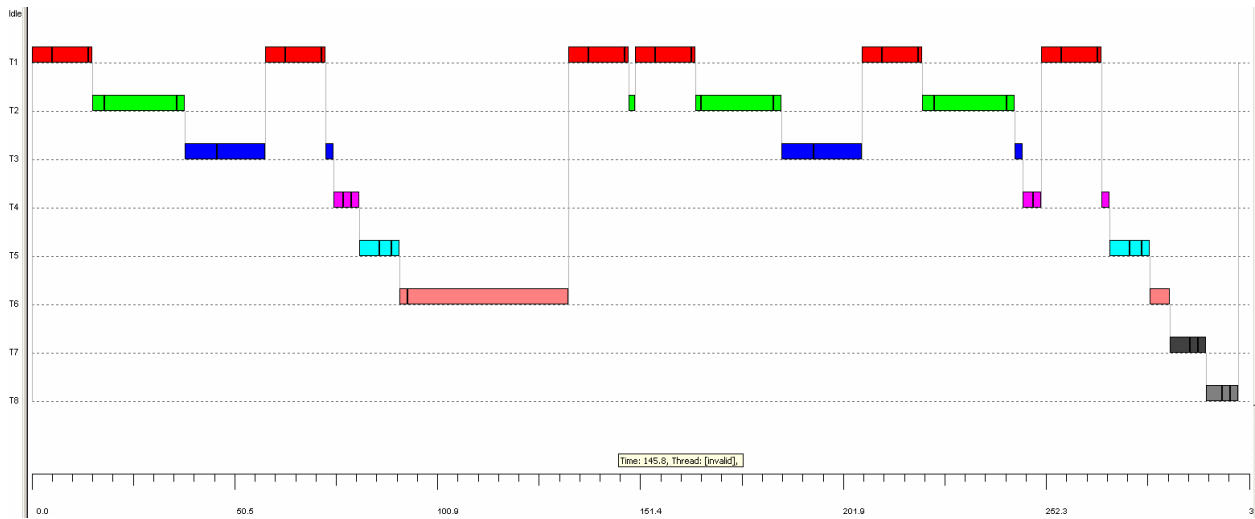


Figure C-2: Non-Preemption Protocol Timeline (Unschedulable)

Contrast this with the PIP timeline for the same task set below which is able to preempt task 6 as necessary to ensure higher priority tasks can meet their deadlines.

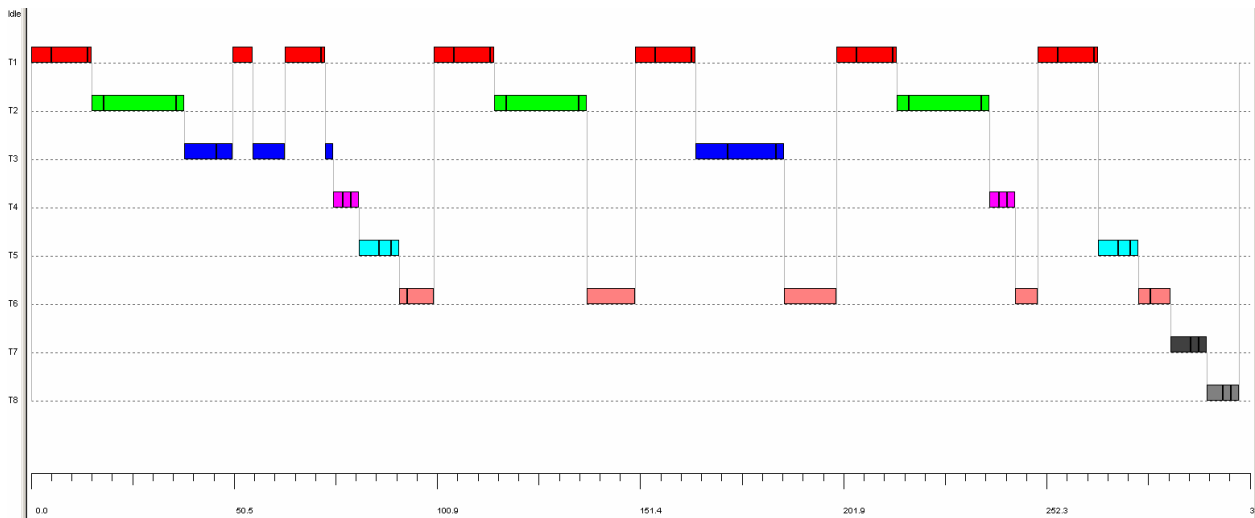


Figure C-3: Priority Inheritance Protocol Timeline (Schedulable)

Part C.c Schedulability Under HLP but not PIP, NPP

Like the prior section, the approach to create a task set that isn't schedulable under PIP but is under HLP is to adjust the execution times to create priority inversion sufficient to cause a high priority task to miss its deadline. Such a combination is shown in the table below.

i	T_i	Early Execution	Critical Section		Late Execution	Total	PIP	HLP
		C_{ie}	C_{ib}	Resources	C_{ic}	C_{il}	B_i	B_i
1	50	5	9	M1,M2,M3,M4	1	15	39	20
2	100	3	20	M2	2	25	19	12
3	150	9	12	M1	2	23	7	5
4	200	2.5	2	M3,M4	2	6.5	5	5

<i>i</i>	<i>T_i</i>	Early Execution	Critical Section		Late Execution	Total	PIP	HLP
		<i>C_{ia}</i>	<i>C_{ib}</i>	Resources	<i>C_{ic}</i>	<i>C_{il}</i>	<i>B_i</i>	<i>B_i</i>
5	250	5	5	M3,M4	2	12	0	0
6	300	2	30	M5	5	37	2	2
7	350	5	2	M5	2	9	0	0
8	400	2	2		2	6	0	0

The priority inversion can potentially cause task 1 to miss its execution deadlines because the maximum blocking time under PIP is 39 versus 20 for HLP. The inversion can be seen in the following case between task 1 and task 3.

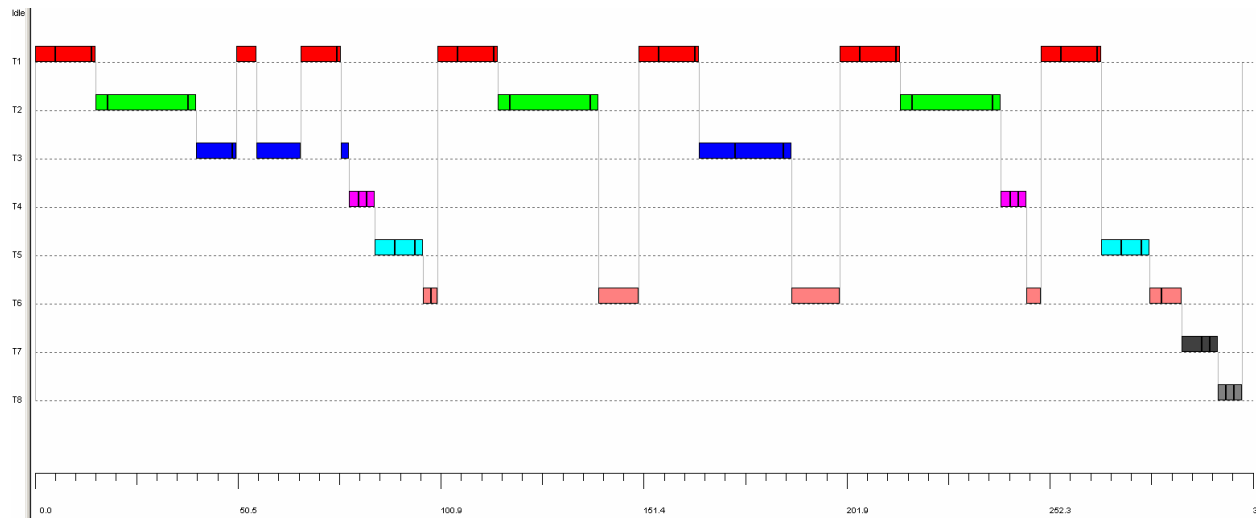


Figure C-4: Priority Inheritance Protocol Timeline (Unschedulable)

Part D Analysis of the Controller Area Network Bus

The CANbus operates in a non-preemptive fashion, that is, once a message starts transmission, it runs uninterrupted until the message completes. The effect of this behavior can easily be seen by modeling messages as fixed-priority tasks with execution proportional to the message length.

Part D.a Non-Preemptive Message Transmission

Part C.b provides an analysis and explanation as to why some task sets are schedulable with preemption but are not without. In brief, a long message of lower priority can easily cause a higher priority message to miss its deadline because, by definition, the higher priority task cannot interrupt it until the next arbitration period.

Part D.b A Design Schedulable at .5Mbps but not .25Mbps

The triggers in the following table form a message set that is not schedulable with a .25Mbps CANbus, but is schedulable at higher speeds. Clearly, the slower bus speed of the two (.5Mbps) is more desirable because it is a less expensive part. The only consideration is whether it provides enough “headroom” for anticipated modifications.

<i>i</i>	Period	Deadline	Description		1Mbps	.5Mbps	.25Mbps
1	100	100	Traction Battery Voltage	0	0.259	0.518	1.036
2	100	100	Traction Battery Current	0	0.388	0.776	1.552
3	1000	1000	Traction Battery Temperature	0	4.549	13.484	-1
4	100	100	Aux Battery Voltage	0	0.517	1.034	2.068
5	1000	1000	Traction Battery Temp	0	4.678	13.742	-1
6	100	100	Aux Battery Current	0	4.162	12.71	-1
7	5	5	Accelerator Position	0	0.646	1.292	2.584
8	5	5	Brake Pressure, Master	0	0.775	1.55	3.1

<i>i</i>	Period	Deadline	Description		1Mbps	.5Mbps	.25Mbps
9	5	5	Brake Pressure, Line	0	0.904	1.808	3.616
10	100	100	Lube Pressure	0	4.291	12.968	-1
11	5	5	Clutch Line Pressure	0	1.033	2.066	4.132
12	100	100	Vehicle Speed	0	4.42	13.226	-1
13	1000	1000	Traction Battery Ground Fault	0	4.74	13.866	-1
14	50	5	Hi&Lo Contact Open/Close	0	1.124	2.248	4.496
15	50	20	Key Switch Run	0	1.248	2.496	4.992
16	50	20	Key Switch Start	0	1.31	2.62	7.304
17	50	20	Accelerator Switch	0	1.383	2.766	7.596
18	20	20	Brake Switch	0	1.186	2.372	4.744
19	50	20	Emergency Brake	0	1.445	2.89	7.844
20	50	20	Shift Lever	0	1.526	3.052	8.168
21	1000	1000	Motor/Trans over Temperature	0	4.813	14.012	-1
22	50	20	Speed Control	0	1.994	3.988	14.168
23	50	20	12V Power Ack Vehicle Control	0	2.056	4.112	14.416
24	50	20	12V Power Ack Inverter	0	2.118	4.236	14.664
25	50	20	12V Power Ack I/M Control	0	2.438	4.876	-1
26	50	20	Brake Mode	0	2.5	6.806	-1
27	50	20	SOC Reset	0	2.691	7.188	-1
28	50	20	Interlock	0	2.753	7.312	-1
29	10	10	High Contactor Control	0	1.655	3.31	8.684
30	10	10	Low Contactor Control	0	1.913	3.826	-1
31	50	20	Reverse and 2nd Gear Clutches	0	2.826	7.458	-1
32	5	5	Clutch Pressure control	0	1.784	3.568	-1
33	1000	1000	DC/DC Converter	0	4.875	14.136	-1
34	50	20	DC/DC Converter Current Control	0	2.955	7.716	-1
35	50	20	Copy of Eme12V Power Relay	0	3.017	7.84	-1
36	1000	1000	Traction Battery Fault	0	4.948	14.282	-1
37	50	20	Brake Solenoid	0	3.079	7.964	-1
38	50	20	Backup Alarm	0	3.141	8.088	-1
39	50	20	Warning Lights	0	3.262	8.33	-1
40	50	20	Key Switch	0	3.324	8.454	-1
41	50	20	Main contactor Close	0	3.386	8.578	-1
42	5	5	Torque Command	0	2.247	4.494	-1
43	5	5	Torque Measured	0	2.376	4.752	-1
44	50	20	FW/REV	0	3.448	8.702	-1
45	50	20	FWD/REV Ack	0	3.51	8.826	-1
46	50	20	Idle	0	3.583	8.972	-1
47	50	20	Inhibit	0	3.645	9.096	-1
48	50	20	Shift in Progress	0	3.707	9.22	-1
49	10	10	Processed Motor Speed	0	2.629	7.064	-1
50	50	20	Inverter Temp Status	0	3.78	9.366	-1
51	50	20	Shutdown	0	3.842	9.49	-1
52	50	20	Status/Malfunction	0	3.971	9.748	-1
53	50	21	Main Contactor Acknowledge	0	4.033	9.872	-1

Part E Analyzing Multiprocessor Environments

To simplify task generation, all tasks were created in a worksheet which used periods which were 10 multiples of the task id. To meet the utilization goals, the period was divided by n/u which yielded approximately 15.6 for a utilization of 160%. (Of course, if the application supported cut and paste operations, it would be far more convenient rather than having to enter this by hand). The results are also shown for the second task set providing 260% utilization (using a divisor of 9.6 of the period).

<i>i</i>	T_i	U=160%		U=260%	
		C_i	U_i	C_i	U_i
1	10.0	0.6400	0.06400	1.0400	0.10400
2	20.0	1.2800	0.06400	2.0800	0.10400
3	30.0	1.9200	0.06400	3.1200	0.10400
4	40.0	2.5600	0.06400	4.1600	0.10400
5	50.0	3.2000	0.06400	5.2000	0.10400

6	60.0	3.8400	0.06400	6.2400	0.10400
7	70.0	4.4800	0.06400	7.2800	0.10400
8	80.0	5.1200	0.06400	8.3200	0.10400
9	90.0	5.7600	0.06400	9.3600	0.10400
10	100.0	6.4000	0.06400	10.4000	0.10400
11	110.0	7.0400	0.06400	11.4400	0.10400
12	120.0	7.6800	0.06400	12.4800	0.10400
13	130.0	8.3200	0.06400	13.5200	0.10400
14	140.0	8.9600	0.06400	14.5600	0.10400
15	150.0	9.6000	0.06400	15.6000	0.10400
16	160.0	10.2400	0.06400	16.6400	0.10400
17	170.0	10.8800	0.06400	17.6800	0.10400
18	180.0	11.5200	0.06400	18.7200	0.10400
19	190.0	12.1600	0.06400	19.7600	0.10400
20	200.0	12.8000	0.06400	20.8000	0.10400
21	210.0	13.4400	0.06400	21.8400	0.10400
22	220.0	14.0800	0.06400	22.8800	0.10400
23	230.0	14.7200	0.06400	23.9200	0.10400
24	240.0	15.3600	0.06400	24.9600	0.10400
25	250.0	16.0	0.06400	26.0	0.10400

The results of load-balancing and bin packing these tasks are shown in the table below. Bin-packing the first task set leaves two processors unutilized because the task set utilization is less than 200%.

CPU	160% Task Set		260% Utilization Task Set	
	Load-Balanced Utilization	Bin-Packed Utilization	Load-Balanced Utilization	Bin-Packed Utilization
1	0.448	0.768	0.728	0.728
2	0.384	0.832	0.624	0.936
3	0.384	0	0.624	0.728
4	0.384	0	0.624	0.208
Total	1.6	1.6	2.6	2.6

Bin-packing has the advantage of making it easy to add additional tasks later without having to rebalance tasks. In reference to the suitcase analogy, this is like bringing an extra duffel bag with you on a trip to load up with the stuff you accumulate by the end; there is no need to rearrange the other bags. A system that is load-balanced, on the other hand, has the advantage of reducing the worst-case execution times of the individual tasks since there is less competition on any given processor. However, if a new task needs to be added that cannot be accommodated on any of the existing processors, it will be necessary to rebalance the tasks to make room. This is the problem that occurs when you buy that last souvenir and don't have space enough for it in your suitcases without shuffling items around.

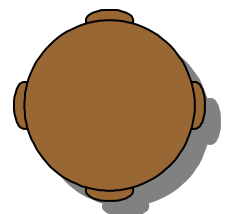
Part F Custom Dinner Time Conversation Catalog

In describing Ethernet to people, I often use the analogy of a dinner table conversation at a family gathering. In these semi-chaotic environments, many of the same underlying principles allow conversation to work in a reasonably fluid fashion. Most people listen to hear if anyone is speaking before speaking themselves. This is carrier sensing. Consider the typical behavior when two people start speaking at the same time: both usually pause for a short but random amount of time, the first to speak then claims the “channel” and the other waits until the first is done before speaking. This is collision detection with random backoff. (Token Ring can be similarly explained by passing a spoon around the table and only allowing the person with the spoon to speak.)

It occurred to me that it might be fun (and worth the extra credit) to use the basic Ethernet modeling capabilities to model dinner time conversation. I created a new catalog file, “Dinner Time.twc” which captured the behavior, strings, descriptions, bindings and icons for this new messaging environment.

Part F.a The “Dinner Table”

The “Dinner Table” communication object has properties similar to the Ethernet bus with all strings and object types renamed appropriately. It is part of the hardware hierarchy.



Part F.b The Participants

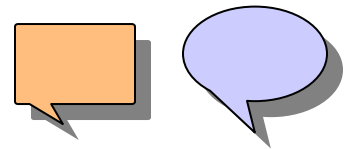
I created eight new conversation participants which were essentially variations on the CPU object (the source and destination of Ethernet messages). The icons used to identify these are shown below in the two default orientations (they can be rotated using TimeWiz).



Participants are linked to each other in a way that reflects communication preferences.

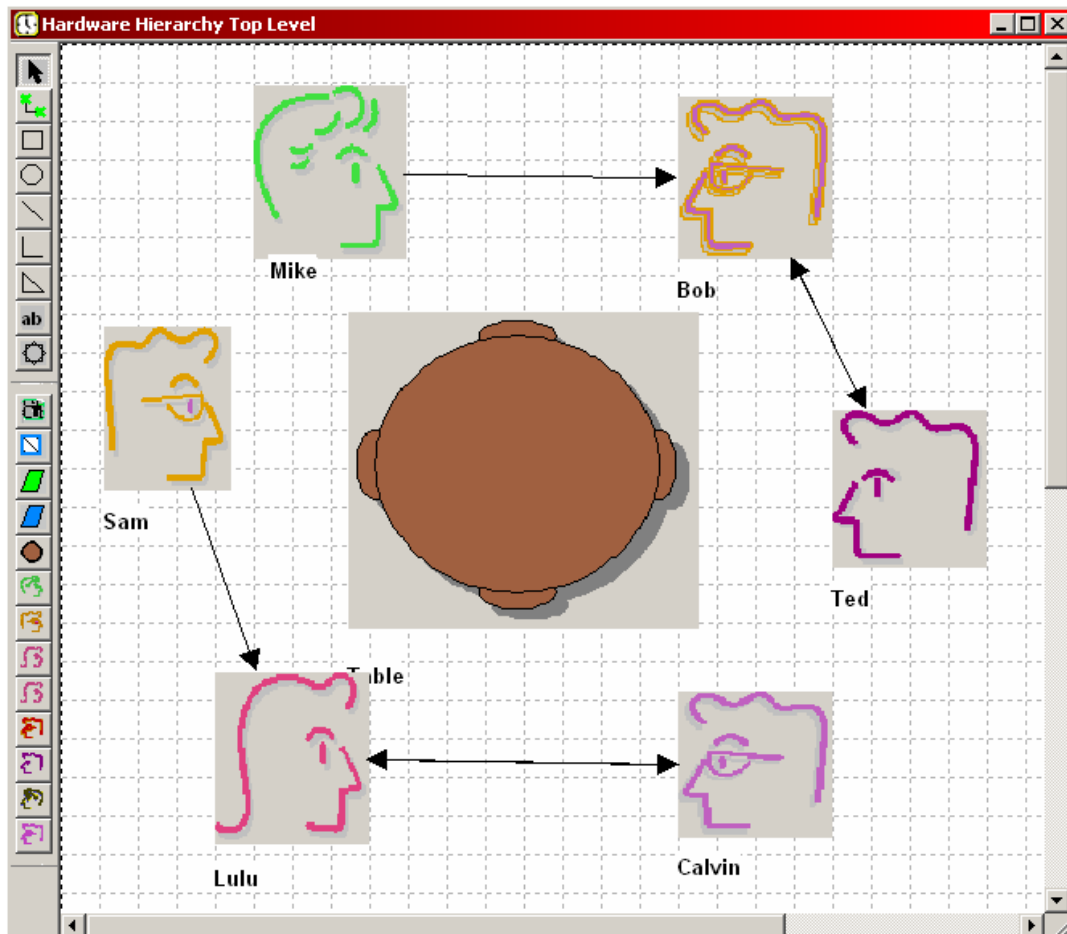
Part F.c The Messages

Messages are implemented just like normal Ethernet messages except the strings and bindings have been altered to reflect conversations. These are used in the software diagram. There is a choice of two icons for conversation topics (to reflect, perhaps, question and answer sequences).



Part F.d Hardware Diagram

The hardware hierarchy is where you create the communication network around the dinner table.



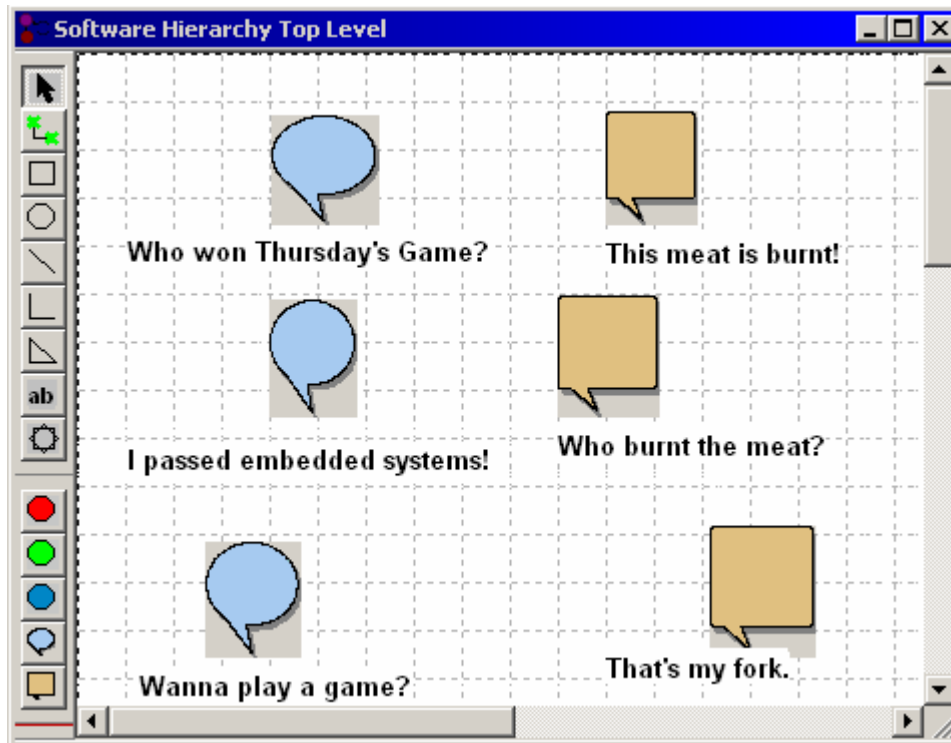
HW Table View

	Name	Schedulable	Schedulable	Normalized S	Normalized S	Scheduling P	Data Sharing	Type	OS	Context Swi
1	Mike	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
2	Sam	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
3	Lulu	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
4	Bob	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
5	Ted	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
6	Calvin	1	1	1	1	Fixed-Priority	Priority Inherit	Generic	Generic	0
7	New Object									

CPU LogicalResource CanBus Ethernet Table Participant

Part F.e Software Diagram

The software hierarchy is where you create the actual messages that will be exchanged.



Action Table View

	Name	Transmits on	From Who	To Who	Message Len	Transmission	Triggered In
1	Who won Thursday's Game?	Table	Ted	Bob	1024	0.8192	<List>
2	This meat is burnt!	Table	Mike	Lulu	1024	0.819200	<List>
3	I passed embedded systems!	Table	Sam	Mike	1024	0.8192	<List>
4	Who burnt the meat?	Table	Bob	Lulu	1024	0.819200	<List>
5	Wanna play a game?	Table	Lulu	Calvin	1024	0.8192	<List>
6	That's my fork.	Table	Mike	Bob	1024	0.819200	<List>
7	New Object						

CPU BINDABLE CanBus BINDABLE Ethernet BINDABLE Conversation BINDABLE

Part G Problems and Challenges

This lab was made twice times as long as it should have been due to TimeWiz frequently crashing (with about an 85% failure rate whenever using simulation and timelines) and general usability problems and inconsistency errors such as tables becoming read-only after some operations, temperamental timeline generation, inconsistent cut and paste operations, improper edit field handling, and the frustrating auto-switch of tools in the software diagrams. The tool has useful functionality, but the usability problems obscure this usefulness, particularly for the beginning user who has to learn all its idiosyncrasies. Another difficulty was interpreting the inconsistent terminology between the lab handout (kernel priority protocol), the documentation (kernelized monitors), and TimeWiz (interrupt masking).