Fill in the empty fields of the following TDL message-scheduling example, and provide additional diagrams as shown in Figure 2.

The example comprises four modules with several modes of operation, distributed on three nodes. In every mode a set of tasks is executed with different task invocation periods. Modules M1 and M2 execute on node N1, module M3 on node N3, and module M4 on node N3. The distribution of the four modules together with the export/import relationships implies their communication requirements.

Module M1 (located on node N1)

Mode	Mode period	Tasks with their LETs in brackets
f11	40 ms	inc (40 ms), dec (40 ms)
f12	40 ms	inc (40 ms), dec (20 ms)

Module M2 (located on node N1)

Mode	Mode period	Tasks with their LETs in brackets
f12	40 ms	inc (40 ms), dec (40 ms)
f84	10 ms	inc (5 ms), dec (10 ms)
f45	40 ms	inc (10 ms), dec (8 ms)
f24	20 ms	inc (20 ms), dec (10 ms)

Module M3 (located on node N2)

Mode	Mode period	Tasks with their LETs in brackets
f11	40 ms	inc (40 ms), dec (40 ms)
f88	5 ms	inc (5 ms), dec (5 ms)
f48	10 ms	inc (10 ms), dec (5 ms)

Module M4 (located on node N3)

Mode	Mode period	Tasks with their LETs in brackets
res40	40 ms	sum (40ms)
res20	20 ms	sum (20ms)
res10	10 ms	sum (10ms)
res5	5 ms	sum (5ms)
res2	2 ms	sum (2ms)
res1	1 ms	sum (1ms)

Task Name	WCET	Description
inc	1 ms	increases a counter value, starts at 20, restarts at 20 when value equals 100; the current counter value is provided as output port of the task
dec	1 ms	decreases a counter value, start at 200, restarts at 200 when value equals 10; the current counter value is provided as output port of the task
sum	1 ms	computes the sum of counters, that is of all output ports of tasks executing in modules M1, M2 and M3

Bus Period Calculation

Above we defined mspGCDM as the GCD (greatest common divisor) of mode periods and mode switch periods in all modes in module M. In our case study the mode switch period equals the mode period in all modes, as otherwise the mode switches would not be harmonic, meaning that a mode switch must not occur during the LET of every task invocation of the currently active mode.

mspGCDM1 = GCD (40, 40) = 40 ms
mspGCDM2 = GCD (40, 10, 40, 20) = 10 ms
mspGCDM3 = GCD (40, 5, 10) = 5 ms
mspGCDM4 = GCD (40, 20, 10, 5, 2, 1) = 1 ms

The bus period of the generated bus schedule is computed as the GCD of the mspGCDM of each module M that communicates on the bus. As module M4 does only receive, but not send on the bus, we only have to consider modules M1 to M3:

bus period = GCD (mspGCDM1, mspGCDM2, mspGCDM3) = GCD (40, 10, 5) = 5 ms

List of Messages

The following table contains a list of all messages that have to be sent by the tasks in all modules and modes. The release and deadline times are relative to the mode period of the task that sends the message.

Message	Module	Mode	Task	Invocation	Size	Release	Deadline
1	M1	f11	inc	1	4	1 ms	40 ms
2	M1	f11	dec	1	4	1 ms	40 ms
3	M1	f12	inc	1	4	1 ms	40 ms
4	M1	f12	dec	1	4	1 ms	20 ms
5	M1	f12	dec	2	4	21 ms	40 ms
6	M2	f11	inc	1	4	1 ms	40 ms
7	M2	f11	dec	1	4	1 ms	40 ms
8	M2	f24	inc	1	4	1 ms	20 ms
9	M2						
10	M2						
11	M2						
12	M2						
13	M2						
14	M2						
15	M2						
16	M2						
17	M2						
18	M2						
19	M2						
20	M2						
21	M2	f84	inc	2	4	6 ms	10 ms
22	M2	f84	dec	1	4	1 ms	10 ms
23	M3	f11	inc	1	4	1 ms	40 ms
24	M3						
25	M3						
26	M3						
27	M3						
28	M3						
29	M3	f88	dec	1	4	1 ms	5 ms

Module M1: Messages and Frames

Every mode period is divided into phases that are equal to the size of a bus period. The phase number in the table below is the phase in which the task invocation ends. Release and deadline are relative to the phase and not to the mode period as in the list above. The list is ordered by mode and then by the phase number.

Message	Module	Mode	Task	Invocation	Size	Phase	Release	Deadline
1	M1	f11	inc	1	4	8	0 ms	5 ms
2	M1	f11	dec	1	4	8	0 ms	5 ms
4	M1	f12	dec	1	4	4	0 ms	5 ms
3	M1	f12	inc	1	4	8	0 ms	5 ms
5	M1	f12	dec	2	4	8	0 ms	5 ms

As described above, the release and deadline of a frame is obtained by calculating the maximum of all release times and the minimum of all deadlines of all bound messages. For this module every message has release 0 ms and deadline 5 ms, so the release and deadline for the frame is equal to that. The size of the frame is determined by the maximum size of the bound messages in the same phase in any mode. For this module this is phase 8

when in both modes 8 bytes are needed as two tasks end in this phase and send messages sized 4 bytes each. Another byte per message is added for the tag that identifies node, module, mode, task invocation, and the phase of the mode in which the message has been produced. The resulting frame size is therefore 10 bytes.

Module M2: Messages and Frames

Message	Module	Mode	Task	Invocation	Size	Phase	Release	Deadline
6	M2	f11	inc	1	4	8	0 ms	5 ms
7	M2	f11	dec	1	4	8	0 ms	5 ms
9	M2	f24	dec	1	4	2	0 ms	5 ms
8	M2	f24	inc	1	4	4	0 ms	5 ms
10	M2	f24	dec	2	4	4	0 ms	5 ms
11	M2	f45	inc	1	4	2	0 ms	5 ms
15	M2	f45	dec	1	4	2	0 ms	3 ms
12	M2	f45	inc					
16	M2	f45	dec					
17	M2	f45	dec					
13	M2	f45	inc					
18	M2	f45	dec					
14	M2	f45	inc					
19	M2	f45	dec	5	4	8	0 ms	5 ms
20	M2	f84	inc					
21	M2	f84	inc					
22	M2	f84	dec					

Following the algorithm for the creation of frames, we show the assignment of messages to frames for this module step-by-step assuming a threshold value of 0.7.

According to the algorithm, we start with mode f11 and phase 8. As we start with an empty set of frames at this point, the first messages is bound to a new frame which inherits their release and deadline constraints. The size corresponds to the message size plus the tag sized 1 byte.

Frame	Size	Release	Deadline	Bound Messages
1	5	0 ms	5 ms	6

The next message in this phase is message 7 which has also a release of 0 ms and a deadline of 5 ms. The evaluation of the metric determines if the message is bound to the frame. We use the following metric calculation as introduced in the formulas below.

$$overlapping = Min(frame.d, msg.d) - Max(frame.r, msg.r)$$

$$metric_{overlapping} = \frac{\frac{overlapping}{frame.d - frame.r} + \frac{overlapping}{msg.d - msg.r}}{2}$$

$$enlargement = Max(0, msg.size - frames.available)$$

$$metric_{enlargement} = 1 - \frac{Min(enlargement, frame.size)}{frame.size}$$

$$metric = \frac{metric_{overlapping} + metric_{enlargement}}{2}$$

The overlapping metric always yields 1 if the values for deadline (d) and release (r) are the same for the message and the frame as it is the case when considering message 7 an frame 1. This can be shown by setting frame.d = msg.d = d and frame.r = msg.r =r in the formula above:

$$overlapping = Min(d, d) - Max(r, r) = d - r$$

$$metric_{overlapping} = \frac{\frac{d-r}{d-r} + \frac{d-r}{d-r}}{2} = \frac{1+1}{2} = 1$$

For the enlargement metric we have msg.size = 5 bytes (including the tag), frame.size = 5 bytes and frame.available = 0 as we already bound message 6 to the frame. This yields to the following enlargement metric:

$$enlargement = Max(0,5-0) = 5$$

$$metric_{enlargement} = 1 - \frac{Min(5,5)}{5} = 1 - \frac{5}{5} = 0$$

This leads us to the following value of the overall metric:

$$metric = \frac{1+0}{2} = \frac{1}{2} = 0.5$$

As we use a threshold value of 0.7 and the metric is below this value, the message is not bound to frame 1 but a new frame is created:

Frame	Size	Release	Deadline	Bound Messages
1	5	0 ms	5 ms	6
2	5	0 ms	5 ms	7

Next we consider mode f24, starting with the first phase, which is phase 2 in this case. Note that in every new phase the available bytes of all frames are reset to the frame size. We have only one task ending in this phase which produces message 9 and as both frames have equal size and timing constraints we assume it gets bound to the first one. The algorithm does not determine to which frame the message is actually bound, so this behavior depends on how the algorithm is actually implemented in software.

Frame	Size	Release	Deadline	Bound Messages
1	5	0 ms	5 ms	6, 9
2	5	0 ms	5 ms	7

In phase 4 of mode f24 two messages are produced. Again the messages can be assigned in any order, as all timing constraint and size requirements are the same and therefore also the calculated metrics are equal. We assume the following assignment of messages to frames 1 and 2:

Frame	Size	Release	Deadline	Bound Messages
1	5	0 ms	5 ms	6, 9, 10
2	5	0 ms	5 ms	7, 8

In mode f45 two tasks end in phase 2. They produce message 11 with release 0 ms and deadline 5 ms and message 15 with release 0 ms and deadline 3 ms. Let's assume message 11 is assigned to frame 2, so there would be frame 1 left for message 15. Here the timing requirements of the message and the frame do not match and so the calculation of the metric, especially the overlapping metric, determines if the message is bound to the frame or a new one is created. As the message fits completely into the available bytes of the frame, the enlargement metric is 1. The overlapping metric equals to the following if we apply it to frame 1 (frame.r = 0, frame.d = 5) and message 15 (msg.r = 0, msg.d = 3):

$$overlapping = Min(5,3) - Max(0,0) = 3 - 0 = 3$$

$$metric_{overlapping} = \frac{\frac{3}{5-0} + \frac{3}{3-0}}{2} = \frac{\frac{3}{5}+1}{2} = \frac{4}{5} = 0.8$$

Consequently, the overall metric is obtained by:

$$metric = \frac{0.8+1}{2} = \frac{1.8}{2} = 0.9$$

As 0.9 is above our threshold of 0.7, message 15 gets bound to frame 1 and the deadline of frame 1 is reduced to 3 ms:

Frame	Size	Release	Deadline	Bound Messages
1	5	0 ms	3 ms	6, 9, 10, 15
2	5	0 ms	5 ms	7, 8, 11

In phase 4 of mode45 two messages are produced. Message 12 …… The metric calculation yields 0.9 when applied to frame 1 and 1.0 when the message is measured against frame 2, to which it therefore gets bound. Message 16 has its release at 0 ms and deadline at 1 ms. Adding this message to frame 1 would considerable restrict it but our metric calculation returns 0.83, so it gets bound to the frame which now has a new deadline of 1 ms.

Frame	Size	Release	Deadline	Bound Messages
1				
2				

In phases 5, 6 and 7 only one task ends in every phase. …..

Frame	Size	Release	Deadline	Bound Messages
1				
2				

In phase 8 ….

Frame	Size	Release	Deadline	Bound Messages
1				
2				

The last mode to consider in this module is f84. In phase 1 we have message 20 with release 1 ms and deadline 5 ms. ……

Frame	Size	Release	Deadline	Bound Messages
1				
2				

The last phase to consider in this mode is phase 2 with two messages. ……

Frame	Size	Release	Deadline	Bound Messages
1				
2				

Module M3: Messages and Frames

Message	Module	Mode	Task	Invocation	Size	Phase	Release	Deadline
23	M3							
24	M3							
26	M3							
25	M3							
27	M3							
28	M3							
29	M3							

In module M3 …...

List of Frames

The following table lists the frames created for modules M1, M2 and M3 with their sizes and timing properties and which messages they can contain in different modes.

Frame	Module	Node	Size	Release	Deadline	Contained Messages
1	M1	N1	10	0 ms	5 ms	1, 2, 3, 4, 5
2	M2	N1				
3	M2	N1				
4	M3	N2				

Bus Schedule Generation

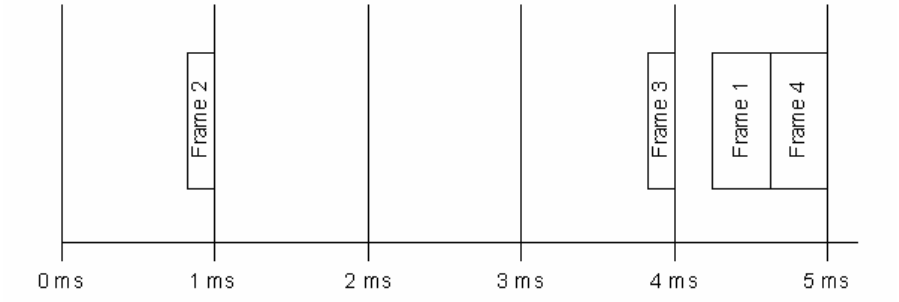

Fig 1. Location of frames inside the bus period

For the identified frames a suitable bus schedule based on their release time and deadline must be generated. As described above this is done using the LRT (Latest Release Time) scheduling algorithm. This results in a bus schedule where frame 2 is scheduled so that its transmission ends at …., frame 3 ends at ……. and frame 1 and 4 end transmission at the end of the bus period at …….. As frame 4 is released later than frame 1 it is scheduled after frame 1.

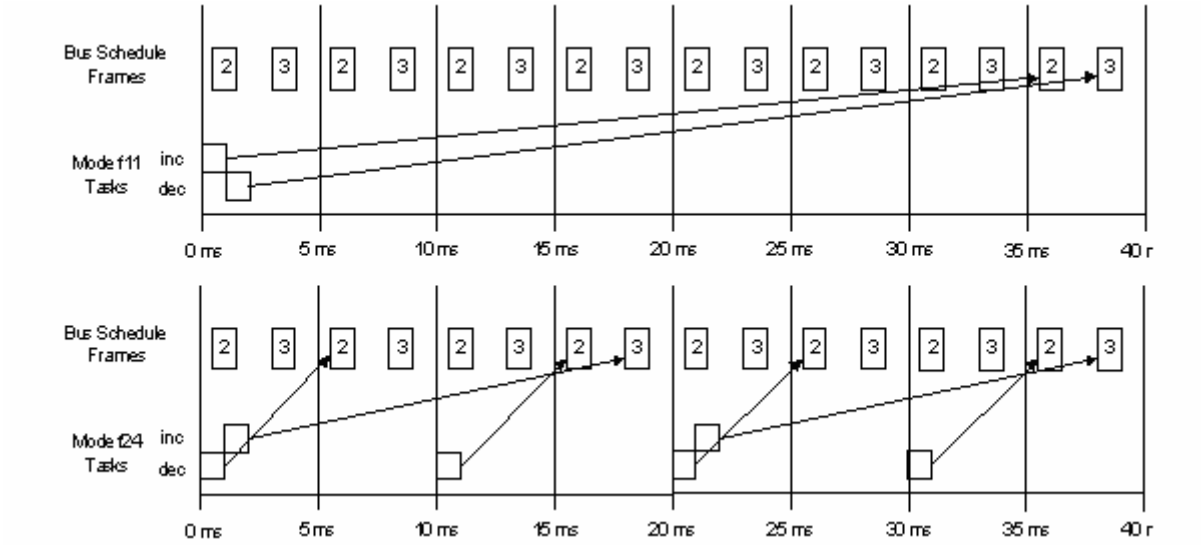

Fig 2. Assignment of task invocations to frames in module M2

This is a schematic representation which demonstrates which frames are used to transfer the messages of the tasks running in module M2 in its different modes. The boxes labeled with 2 and 3 represent frame 2 and frame 3 of the bus schedule (see Fig. 10). The empty boxes represent the WCET of the tasks inc and dec. The arrows point to the frame that is used to transfer the message produced by the task from which an arrow originates.

References

[1] Thomas A. Henzinger, Benjamin Horowitz, and Christoph M. Kirsch. Giotto: A time-triggered language for embedded programming. Proceedings of the First International Workshop on Embedded Software (EMSOFT), Lecture Notes in Computer Science 2211, Springer-Verlag, 2001, pp. 166-184.

[2] J. Templ, 2004, TDL Specification and Report. Technical Report, Department of Computer Science, University of Salzburg, http://www.cs.uni-salzburg.at/pubs/reports/T004.pdf

[3] H. Kopetz, 1997, Real-time Systems: Design Principles for Distributed Embedded Applications. Kluwer, 1997

[4] Jane W. S. Liu. Real-Time Systems. Prentice-Hall, 2000