

Real-Time in Embedded Systems

(2) Ch. 2.6 – 2.9

Contents

□ Analysis of RTSs

- Timing Properties
- Methods for Timing Analysis
- Example of AnalysisComponent-Based Design of RTS

□ Component-Based Design of RTS

- Timing Properties and CBD
- Real-Time Operating Systems
- Real-Time Scheduling

□ Testing and Debugging of RTSs

Timing Properties

□ Execution Time

- execution time of single task
- result is time (i.e., the number of clock cycles)
- not account for types of interfering background
- role is to find out how much computing resources is needed to execute the task

Execution Time

□ Types of Execution times

- Worst-case execution time (WCET)
 - maximum amount of computer resources required to execute the task
- Best-case execution time (BCET)
- Average execution time (AET)
 - less importance for RTS analysis

Timing Properties – Response Time

- time takes from the *invocation* to the *completion* of the task
- assume that a task does not voluntarily suspend itself during execution
 - not call primitives as sleep() or delay()
 - involuntarily suspension is allowed
- include delays caused by contention of resources

Other Timing Properties

☐ End-to-End Delay

- timing of externally visible events
- upper bound on the delay between input and output

☐ Jitter

- metric for variability in time
- Execution Time Jitter
 - ☐ the differences between the task's BCET and WCET
- Response Time Jitter

Example of Jitter

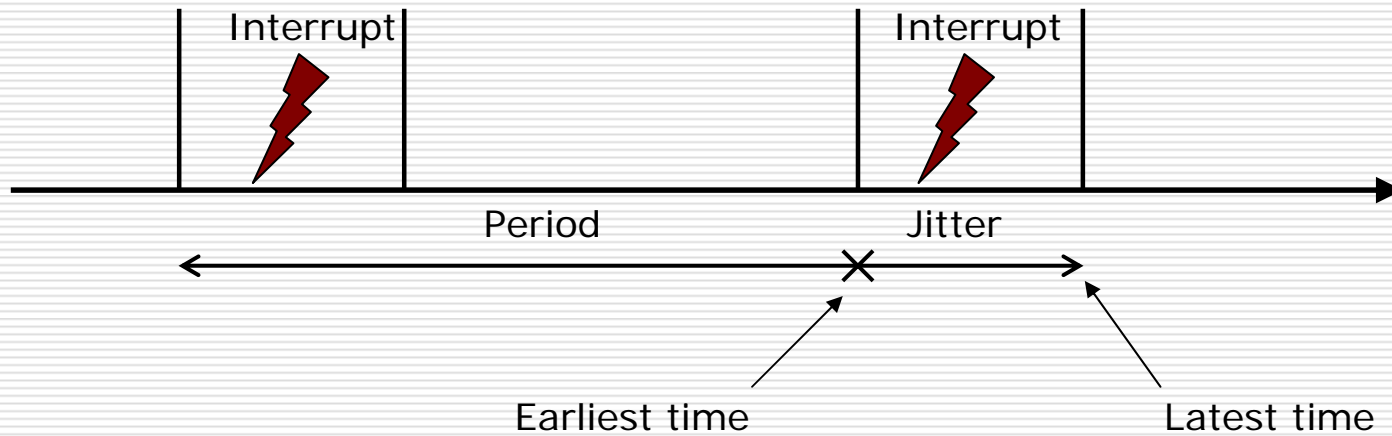


Figure 2.3 Jitter used as a bound on variability in periodicity

Methods for Timing Analysis

- **safe** and **tight** is important factor for estimates of timing analysis
 - estimate is safe if it is underestimation of the WCET
 - estimate is tight if it is close to the WCET

Methods for Timing Analysis (1)

□ Execution-Time Estimation

- WCET is the most important but the most difficult execution time to obtain
- Methods for get WCET
 - static analysis
 - counting the number of clock cycle of codes
 - very difficult get good results
 - cache, branch predictors
 - dynamic analysis
 - equivalent to testing
 - does not guarantee safe

Methods for Timing Analysis (2)

□ Schedulability Analysis

- to see whether or not a system is *schedulable*
- *task is schedulable if all task deadlines will always be met*
- 2 classes of Schedulability analysis technique
 - response-time analysis
 - utilization analysis
 - both are based on similar types of task models but not provided by commercial RTOSs.

Other Methods for Timing Analysis

□ End-to-End Delay Estimation

- calculate the response time for each task/message in the end-to-end chain and to summarize these response times

□ Jitter Estimation

- simply assume BCET is zero

Example of Analysis

□ Analysis of Tasks

Task	<i>T</i>	<i>C</i>	<i>D</i>	Prio
X	30	10	30	High
Y	40	10	40	Medium
Z	52	10	52	Low

TABLE 2.2 Example Task Set for Analysis

- T: task's period
- C: WCET
- D: deadline
- $T = D$ in example

Analysis of Tasks

□ Utilization Based Analysis

- Task is schedulable if

- $U_i = C_i / R_i$

$$U_{\text{tot}} \leq n(2^{1/n} - 1)$$

- As a result, current tasks are not schedulable

Message	<i>T</i>	<i>C</i>	<i>D</i>	Prio	<i>U</i>
X	30	10	30	High	0.33
Y	40	10	40	Medium	0.25
Z	52	10	52	Low	0.23
Total					0.81
Bound					0.81

TABLE 2.3 Result of RM Test

Analysis of Tasks

□ response-time analysis

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

- this formula is not an close form, thus

$$R_i^{m+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^m}{T_j} \right\rceil C_j$$

Analysis of Tasks

□ the result of response-time analysis

TABLE 2.4 Result of Response-Time Analysis for Tasks

Task	T	C	D	Prio	R	$R \leq D$
X	30	10	30	High	10	Yes
Y	40	10	40	Medium	20	Yes
Z	52	10	52	Low	52	Yes

Analysis of Tasks

□ Analysis of Messages

- use CAN-bus
- S : data size (bit)
- time is bit-time

Message	T	S	D	Id
X	350	8	300	00010
Y	500	6	400	00100
Z	1000	5	800	00110

TABLE 2.5 Example CAN Message Set

Analysis of Tasks

- the transmission time, C_i is given:

$$C_i = 85i + 47 + \left\lfloor \frac{34 + 85i - 1}{4} \right\rfloor$$

- The response-time equation is given:

$$R_i = w_i + C_i$$
$$w_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_j + 1}{T_j} \right\rceil C_j$$

Analysis of Tasks

□ the result Response-Time Analysis for CAN

Message	<i>T</i>	<i>S</i>	<i>D</i>	Id	Prio	<i>C</i>	<i>W</i>	<i>R</i>	$R \leq D$
X	350	8	300	00010	High	130	130	260	Yes
Y	500	6	400	00100	Medium	111	260	371	Yes
Z	1000	5	800	00110	Low	102	612	714	Yes

TABLE 2.6 Result Response-Time Analysis for CAN

Components-Based Design of RTS

- **main challenge of designing CBD of RTSs**
 - Constraints on extra-functional properties
 - timing, QoS, dependability
 - The need to statically predict these extra-functional properties
 - Scarce Resources

Components-Based Design of RTS -Timing Properties and CBD

□ Execution Time

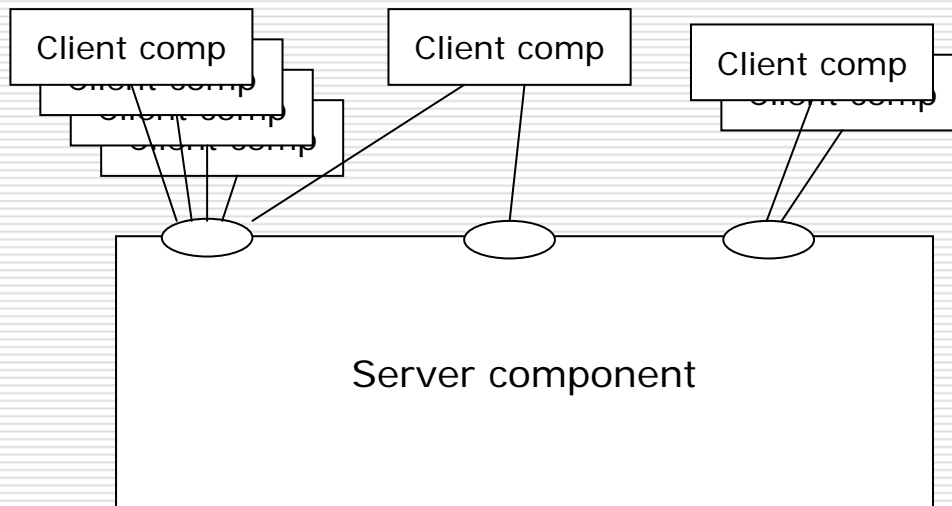
- for a component used in a real-time context, an execution time measure will have to be derived

Components-Based Design of RTS - Timing Properties and CBD

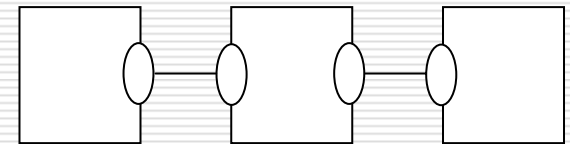
□ Issues of Response Time

- relation between task and Component
- how to handle delays in distributed system

(a)



(b)



Components-Based Design of RTS - Timing Properties and CBD

□ Response Time

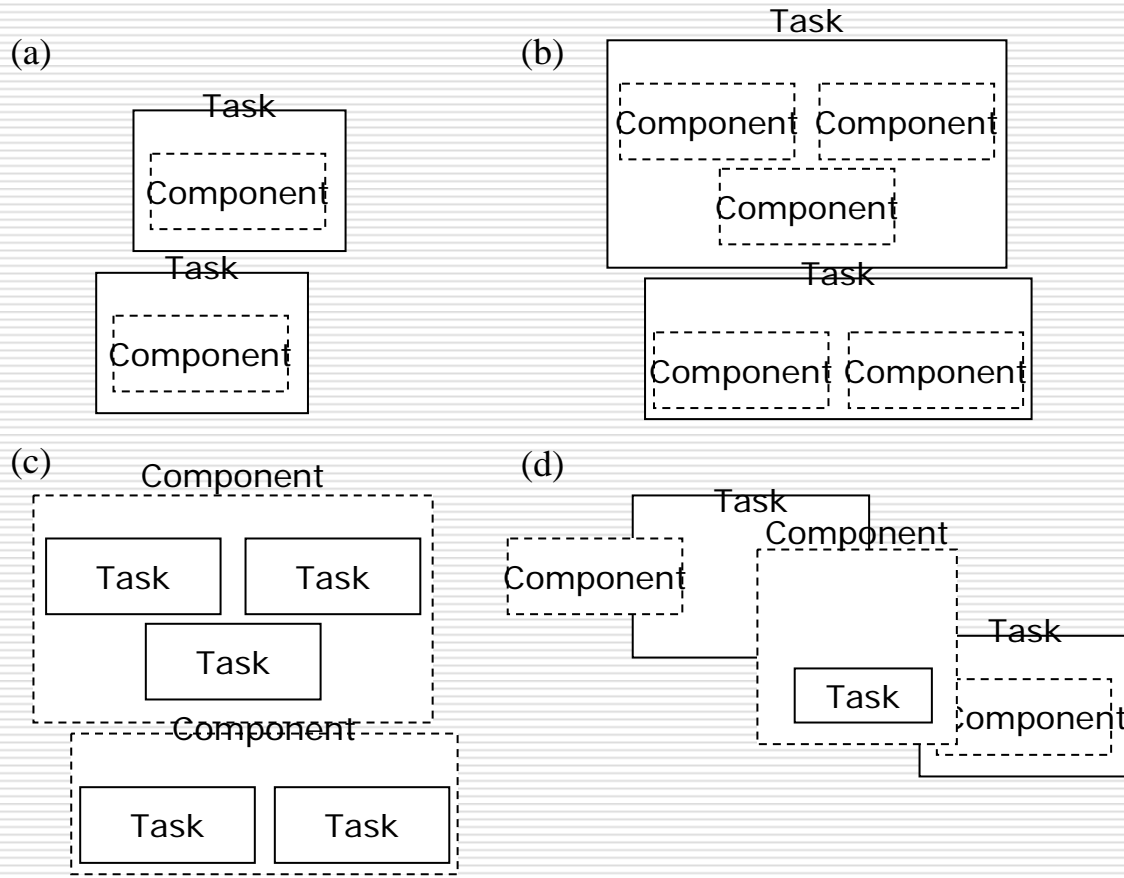
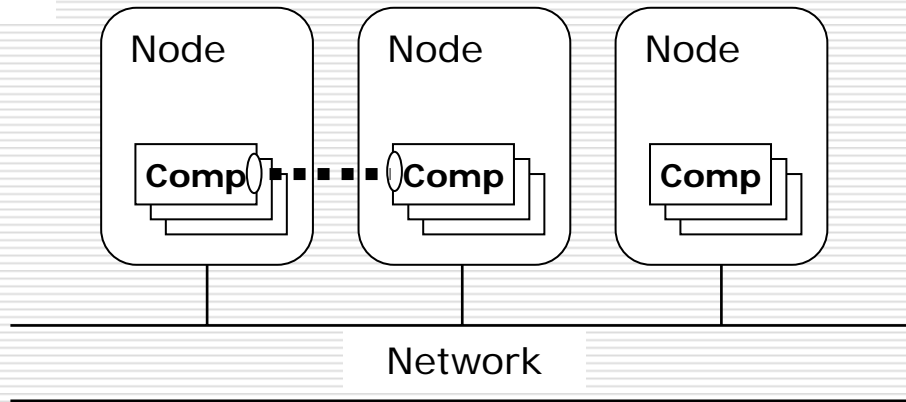


Figure 2.5 Tasks and components: (a) one-to-one correspondence, (b) one-to-many correspondence, (c) many-to-one correspondence, (b + c) many-to-many correspondence, and (d) irregular correspondence.

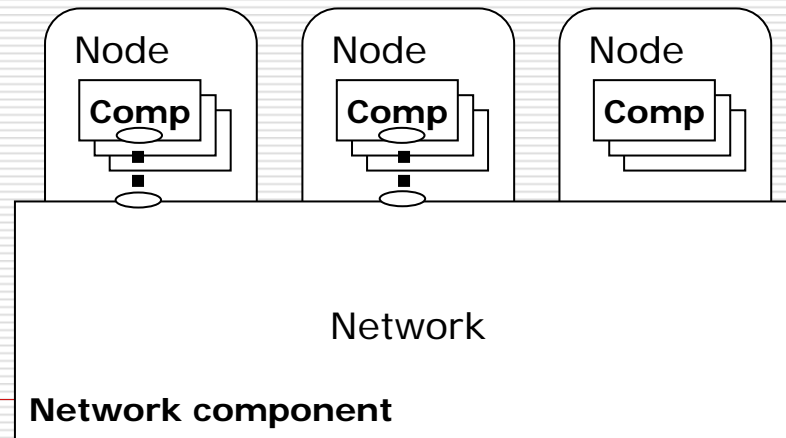
Components-Based Design of RTS - Timing Properties and CBD

□ Response Time

(a)



(b)



Components-Based Design of RTS - Real-Time Operating Systems

☐ **Component-Based RTOSs**

- Most RTOS allows for offline configuration, but it's not same as RTOS being CBD
- For an RTOS to be component based
 - ☐ components conform to a component model
 - ☐ typically not the case in most configurable RTOSs

☐ **RTOS that supports CBD**

- Koala, Chimera RTOS, Rubus, TTOS, ...

Components-Based Design of RTS - Real-Time Scheduling

- ❑ **Ideally, the response time of a component should be independent of environment**
- ❑ **In most cases, it's unrealistic**
 - the execution time of the task will be different in different target environments
 - response time is additionally dependent on the other tasks competing for the same resources

Testing and Debugging of RTSs

- ❑ **Difficult to test and debug RTS**
- ❑ **Even some cases, it's impossible to test or debug**
- ❑ **Solution to Test and Debugging**
 - To build a simulator
 - To record the RTS's behavior during test and replay it