

## Chapter 4. Modeling Real-Time Systems

May-05	1
--------	---



### Modeling Overview

---

- **Model**
  - Abstraction of real world process
    - With well defined purpose
    - Omit irrelevant details, capture the relevant
  - Reduce complexity, for easier analysis and better understanding
- **Real-Time System Modeling Goal**
  - Define an abstract model to design and analyze the timing behavior of a real-time system
- **Key questions:**
  - What can be abstracted, so that model remains meaningful?
  - How to capture?
  - How to analyze?

May-05	2
--------	---



## Outline

---

- Set of definitions for modeling
  - Assumptions
  - Description of action's execution time
  - Structural Elements
  - Interfaces
  - Logical Control / Temporal Control
  - Time triggered vs. Event Triggered
- WCET Analysis
  - Simple task without preemption
  - Simple task with preemption
  - Complex task

May-05	3
--------	---



## Appropriate Abstractions

---

- Assumption Coverage
  - The probability that the assumptions made in the model building process hold in reality
- Two important assumptions needed in designing a model of a **fault tolerant real-time computer system**
  - **Load hypothesis**

Statements on the response time of a computer system can only be made under the assumption that the load offered to the computer system is below a maximum load (**peak load**)
  - **Fault hypothesis**

A statement about the assumptions that relate to the **type** and **frequency of faults** that the computer system is supposed to handle.

May-05	4
--------	---



## Appropriate Abstractions (Cont.)

- Duration of Actions (or execution time)
  - **Actual duration** : given a concrete input data set  $x$  we denote by  $d_{act}(a, x)$  the number of time units of the reference clock  $z$  that occur between the start of action  $a$  and the termination of action  $a$
  - **Minimal duration** : the minimal duration  $d_{min}(a)$  is the smallest time interval it takes to complete the action  $a$ , quantified over all possible input data.
  - **Worst-case execution time (WCET)** : the worst-case execution time  $d_{wcet}(a)$  is the maximum duration it may take to complete the action  $a$  under the stated load and fault hypothesis, quantified over all possible input data.
  - **Jitter** : the jitter for an action  $a$  is the difference between the worst-case execution time  $d_{wcet}(a)$  and the minimal duration  $d_{min}(a)$ .
- Frequency of Activations
  - The maximum number of activations of an action per unit of time.

May-05	5
--------	---

## Appropriate Abstractions (Cont.)

- What is Irrelevant ? :
  - Issues of Representation
    - The representational differences can be hidden within a gateway component that transforms the representation used in one subsystem to the representation used in the other subsystem without changing the semantics.
  - Details of the Data Transformations
    - The internal program logic and the intermediate results of the program are treated as irrelevant detail at the level of a conceptual model

May-05	6
--------	---

## Structural Elements

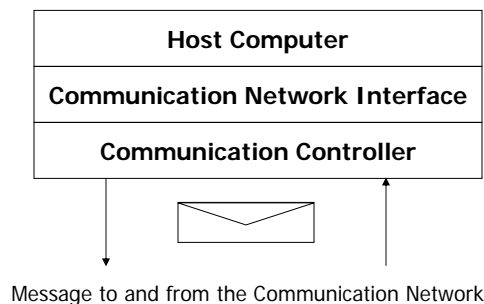
- **Task**
  - The execution of a sequential program
    - A task that does not have an internal state at its point of invocation is called a **stateless task**; otherwise, it is called a **task with state**.
- **Simple Task (S-task)**
  - A task that has **no synchronization point** within its body.
  - Whenever it is started, it can continue until its termination point is reached.
- **Complex Task (C-Task)**
  - A task contains a **blocking synchronization statement** within its body.

May-05	7
--------	---



## Structural Elements (Cont.)

- **Node**
  - A self-contained computer with its own hardware and software, which performs a set of well-defined functions within the distributed computer system.



[Fig. 1] Structure of a Node

May-05	8
--------	---



## Structural Elements *(Cont.)*

- **Fault-Tolerant Unit (FTU)**
  - An abstraction that is introduced for implementing fault tolerance by **active replication**.
  - It consists of a set of replicated nodes that are intended to produce **replica-consistent result messages**, i.e., the same results at approximately the same points in time.
  
- **Computational Cluster**
  - Comprises a set of FTUs that cooperate to perform the intended fault-tolerant service for the cluster environment.
  - The interfaces between a cluster and its environment are formed by the **gateway nodes** of the cluster.

May-05	9
--------	---



## Interfaces

An interface between two subsystems of a real-time system can be characterized by :

- **Control properties**
  - The properties of the control signals crossing the interface, e.g., which task must be activated if a particular event happens.
  
- **Temporal properties**
  - The temporal constraints that must be satisfied by the control signals and by the data that cross the interface.
  
- **Functional intent**
  - The specification of the intended functions of the interfacing partner.
  
- **Data properties**
  - The structure and semantics of the data elements crossing the interface.

May-05	10
--------	----



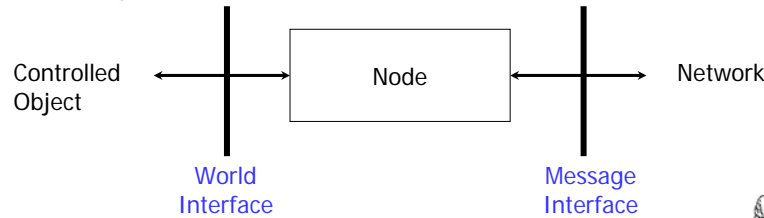
## Interfaces

- **Resource Controller**

- An intelligent interface that transforms differing representations of the same information between interfacing partners.



- **Gateway**



May-05 11



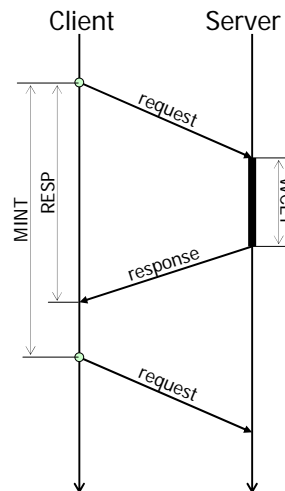
## Interfaces (Cont.)

- Temporal Obligation of Clients and Servers

Three temporal parameters characterize a client-server interaction:

- The **maximum response time, RESP**, that is expected by the client, and stated in the specification
  - The **worst-case execution time, WCET**, of the server that is determined by the implementation of the server.
  - The **minimum time, MINT**, between two successive requests by the client
- In a hard real-time environment, the implementation must guarantee that the condition

$$WCET < RESP$$



May-05 12



## Temporal Control vs. Logical Control

- **Logical control**
  - Concerned with the **control flow** within a task that is determined by the given program structure and the particular input data, in order to achieve the desired data transformation.
- **Temporal control**
  - Concerned with **determining the points in time** when a task must be activated, or when a task must be blocked, because some conditions outside the task are not satisfied at a particular moment.

May-05 13



## Temporal Control vs. Logical Control

- **Even-Triggered vs. Time-Triggered**

Two sources of a temporal control signal for the activation of a task

  - **Event trigger** : The control signal that is derived from significant state change, an **event**, in the environment or within the computer system.
  - **Time trigger** : The control signal that is derived from the progression of time.
- **Even-triggered (ET) system**
  - A system where all the control signals are derived from event triggers
- **Time-triggered (TT)**
  - A system where all the control signals are derived from time-triggers

Extreme Cases

May-05 14



## Temporal Control vs. Logical Control *(Cont.)*

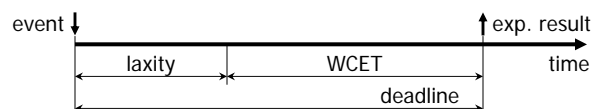
- **Interrupt**
  - External event (significant state change)
  - Triggers context switch from user task to interrupt service routine and back to a user task
  - Interrupt frequency is outside node's control
- **The worst-case administrative overhead (WCAO)**
  - administrative delays of an application task, but are not under tasks direct control
    - Context switch
    - Scheduling decisions
    - OS internal administration
- **Interrupt challenges:**
  - Handling of erroneous interrupts
  - Interrupt frequency  $> 1/WCAO$ , may leave no CPU time for user tasks

May-05	15
--------	----



## Temporal Control vs. Logical Control *(Cont.)*

- **Trigger Task**
  - A periodic time-triggered task that evaluates a trigger condition on a set of temporally accurate real-time variables.
  - Its result can be a control signal that activates another application task.
  - Only those states with a duration greater than the sampling period of the trigger task are guaranteed to be observed.
  - **Overhead of a Trigger Task**
    - The period of the trigger task must be smaller than the laxity (i.e., the difference between deadline and execution time) of an RT transaction that is activated by an event in the environment
    - If the laxity of the RT transaction is very small ( $< 1\text{msec}$ ), then the overhead associated with a trigger task can become intolerable.



May-05	16
--------	----



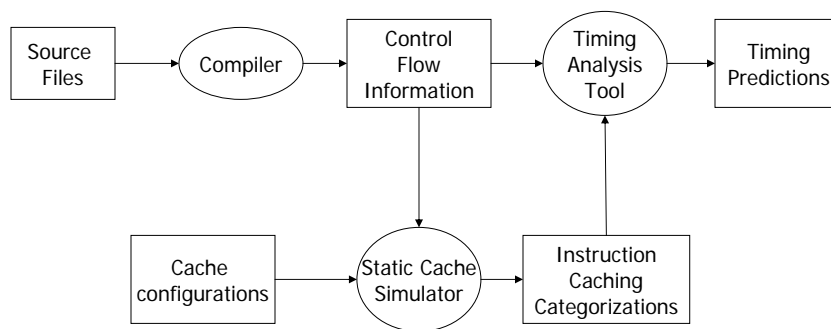
## Worst-Case Execution Time (WCET)

- The **WCET** of a task should be a tight upper bound for the time between task activation and task termination.
- **WCET of S-Tasks** (runs on dedicated HW, without preemption and requiring any OS services)
  - The WCET depends on
    - The **source code** of the task
    - The **properties of the object code generated by the compiler**
    - The characteristics of the **target HW**
  - Source Code Analysis
  - Compiler Analysis
  - Micro-architecture Timing Analysis

May-05	17
--------	----



## Worst-Case Execution Time (WCET) (Cont.)



[Fig. 2] Microarchitecture timing analysis

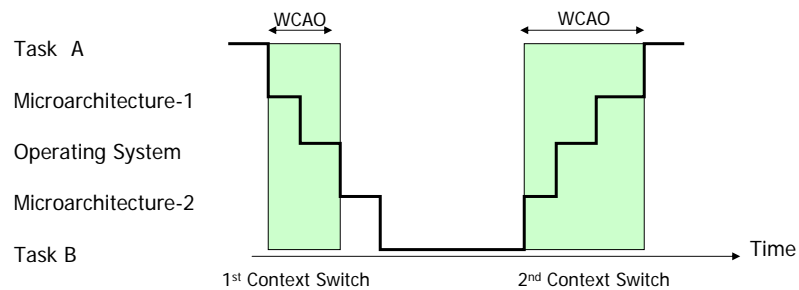
May-05	18
--------	----



## Worst-Case Execution Time (WCET) (Cont.)

- Preemptive S-Tasks

- WCET depends on the following in addition to the above:
  - WCET of the **interrupting task**
  - WCET of the **OS required for context switching**
  - The time required for **reloading the instruction cache and the data cache** of the processor whenever the context of the processor is switched.



[Fig. 3] Worst-case administrative overhead (WCAO) of a task preemption

## Worst-Case Execution Time (WCET) (Cont.)

- WCET of **Complex Tasks**

- A global problem involving all the interacting tasks within a node. It depends on
  - Performance of the task itself
  - The behavior of other tasks
  - The OS within a node.
- Additional delays that originates from the direct interactions, e.g., precedence, mutual exclusion, etc., must be considered.

## Worst-Case Execution Time (WCET) (Cont.)

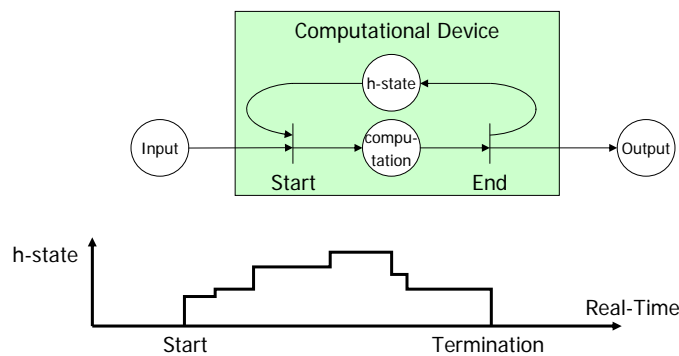
- **State of the Practice**
  - **Measurement of an implementation** to gather experimental WCET data
  - Use of **restricted architecture** that reduces the interactions among the tasks and facilitates the a priori analysis of the control structure.
  - **Analysis of subprograms** so that an effective set of test cases biased towards WCET can be generated mechanically
  - **Extensive testing of complete implementation** to validate the assumptions and to measure the safety margin between assumed WCET and measured exec times.

May-05	21
--------	----



## The History State (H-State)

- The **h-state** at any point of interruption
  - := Contents of the program counter and of all data structures that must be loaded into a “virgin” HW device to resume the operation at the point of interruption.



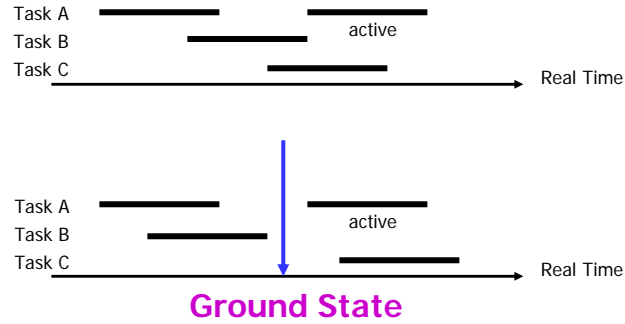
[Fig. 4] Expansion and contraction of the h-state during a computation

May-05	22
--------	----



## The History State (H-State) *(cont)*

- **Ground state**
  - The ground state of a node in a distributed system at a given level of abstraction is a state where **no task is active** and **all communication channels are flushed**, i.e., there are no messages in transit.



- **Nice for checkpointing and restart**