6. Aperiodic events

UNIVERSIDAD DE CANTABRIA

6.1 Concepts and definitions

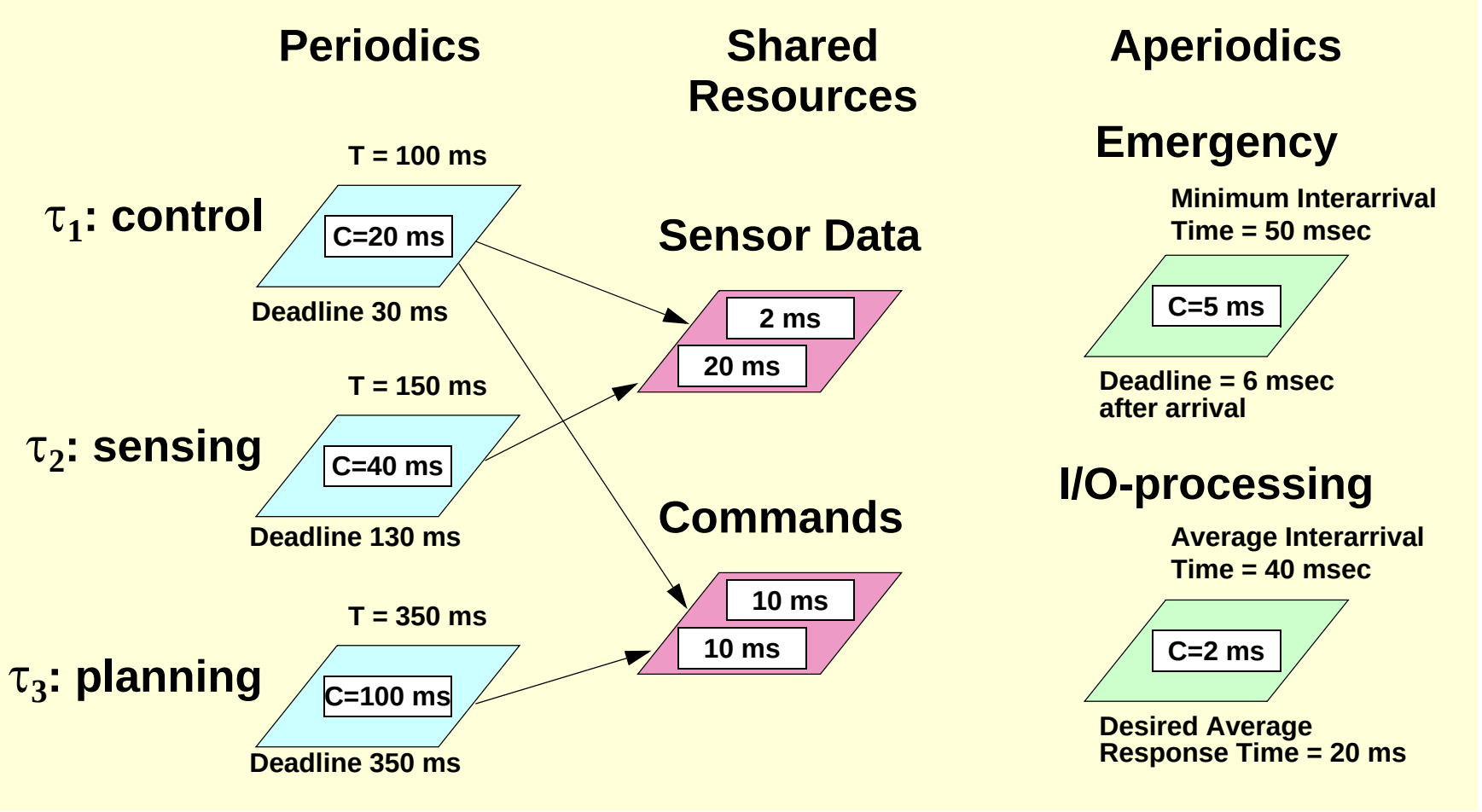
6.2 Polling servers

6.3 Sporadic servers

6.4 Analyzing aperiodic tasks

6.5 Modelling aperiodic events

Sample Problem: Aperiodics



Notes:

UNIVERSIDAD DE CANTABRIA

In this section, we will look at the following aperiodic requirements and incorporate them into the schedulability model:

- Emergency event handling task: execution time = 5 msec.; worst case interarrival time = 50 msec; deadline is 6 msec. after arrival.
- Routine event handling task: average execution time = 2 msec.; average interarrival time = 40 msec; fast average response time of 20 msec. is needed.

6.1 Concepts and Definitions

UNIVERSIDAD DE CANTABRIA

Aperiodic task: runs at irregular intervals.

Aperiodic arrival patterns:

- ***bounded***: the maximum amount of events in a time interval is bounded by physical or software constraints
  - ***sporadic***: a special case with a minimum interarrival time
  - ***bursty***: a special case with a maximum number of events per time interval
- ***unbounded***: an unbounded number of events can arrive in a small time interval.

Concepts and Definitions (cont'd)

UNIVERSIDAD DE CANTABRIA

Aperiodic deadlines:

- **bounded events may have hard or soft deadlines**
- **unbounded events can only have soft deadlines.**

The main problem is guaranteeing predictability in systems with unbounded aperiodic events.

Notes:

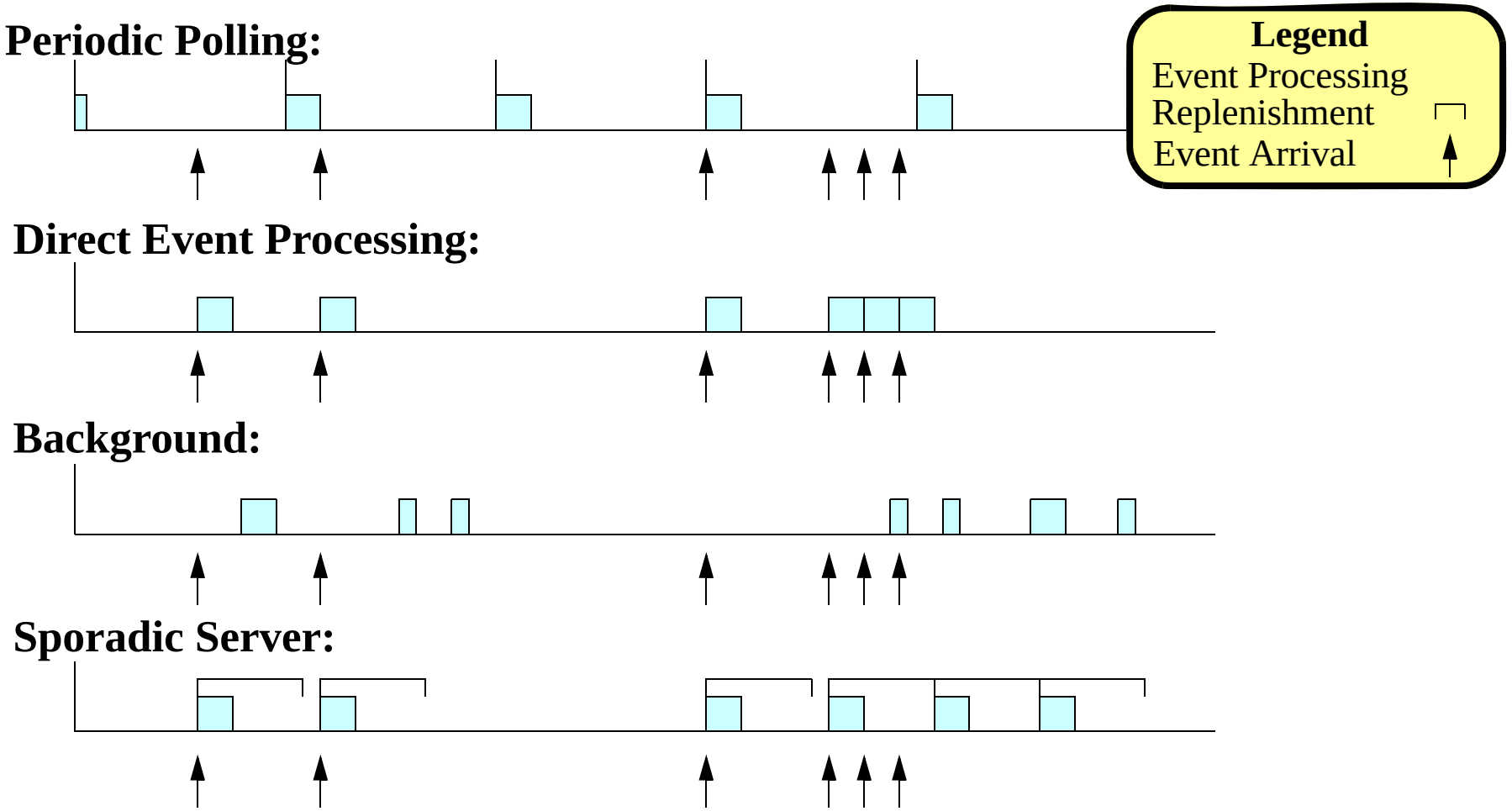
Aperiodic tasks run at irregular intervals and are used to handle the processing requirements of randomly occurring events (aperiodic events) such as operator requests or device interrupts.

Aperiodic events have two different arrival patterns:

- Bounded arrivals: The maximum amount of events arriving in a given time interval is bounded. In many cases, physical or software constraints limit the amount of events that can be generated in a given time interval. For example, a key depression in an operator console which has a debounce circuit that guarantees a minimum interarrival time between events. Response requirements on bounded aperiodic events can impose either a *hard* or a *soft* deadline on aperiodic tasks. Some special cases are sporadic or bursty arrival patterns.
- Unbounded arrivals: The event arrival is characterized by a statistical probability function, but an unbounded number of events can arrive in an arbitrarily small time interval. These events can only have soft deadlines.

The scheduling goal for unbounded aperiodic tasks is to achieve the soft deadline requirement (generally an average-case response time) while preserving the deadlines of the rest of the tasks in the system. Typically, stochastic queuing theory is used to estimate an average response time, given an average interarrival time. Besides, we must achieve a predictable response time for the rest of the tasks while the events arrive at an unpredictable rate.

Scheduling Aperiodic Tasks



Notes:

Three traditional approaches to scheduling aperiodic activities:

- Polling: One approach to handling aperiodic events is to poll periodically to check for events that may have arrived to the system. Worst and average-case response times are a direct function of the polling period. Also the overhead is a direct function of the polling period. The polling mechanism is predictable, but has either slow response times or high overheads.
- Direct event processing: Using an interrupt handler, we can give an aperiodic event immediate service. Since interrupts run at a hardware priority, all the processing of the interrupt handler will occur immediately. The average response time is very good, but this method destroys predictability for all lower priority activities if the arrival pattern is unbounded.
- Background Processing: A very low priority is assigned to the task that processes the aperiodic events. In this way, the schedulability of all higher priority activities is preserved. However, it is very difficult or impossible to meet timing requirements because the aperiodic task has to wait for the execution of all other tasks, which have higher priority.

The sporadic server is a technique for scheduling aperiodic activities that combines the predictability of the polling approach with the short response times of the direct event execution. It represents an extremely flexible solution for scheduling many kinds of aperiodic activities.

6.2 Polling servers

UNIVERSIDAD DE CANTABRIA

A periodic task polls for the arrival of the event

Pseudocode:

```
Next_Activation:=Clock;
loop
   if Event_Arrived then
      Process_Event;
   end if;
   Next_Activation:=Next_Activation+Period;
   delay until Next_Activation;
end loop;
```

Response time is reduced with a shorter period

- but the overhead increases

6.3 Sporadic Servers

UNIVERSIDAD DE CANTABRIA

Characterized by two parameters:

- initial execution capacity (C_R)
- replenishment period (T_R)

It works according to the following rules:

- If capacity>0 events are processed at normal priority, and capacity is consumed
- If capacity=0 events are processed at background priority
- Each portion of consumed capacity is replenished one replenishment period after activation with normal priority
- When a replenishment occurs, if priority = background, priority is elevated to normal

Notes:

UNIVERSIDAD DE CANTABRIA

The sporadic server reserves a certain limited amount of execution capacity for processing aperiodic events at any given priority level, while allowing to guarantee the deadlines of all the other tasks in the system. A sporadic server controls the execution of an aperiodic task and is characterized by two parameters: the *execution capacity*, and the *replenishment period*. The execution capacity represents a time period during which the aperiodic task has the right to execute at its assigned priority; it is initialized to a user-specified value that represents the maximum capacity of the server. The sporadic server works according to the following rules:

- When an aperiodic event occurs, if there is execution capacity available, the aperiodic task will execute at its assigned priority consuming the execution capacity.
- When the execution capacity is exhausted, the task continues executing at a low background priority level.
- Each portion of execution capacity consumed at normal priority is replenished (i.e., added back to the current execution capacity) after the predefined replenishment period has elapsed from the time at which that portion of execution was ready to execute at normal priority.
- When a replenishment occurs, if the sporadic server was executing at a background priority level, its priority is elevated to the normal level.

6.4 Analyzing Aperiodic Tasks: Effects on Other Tasks

UNIVERSIDAD DE CANTABRIA

Polling: **equivalent to a periodic task**

Direct event processing: **for sporadic events, modeled as a periodic task, with period equal to the minimum interarrival time**

Background: **no effect on other tasks**

Sporadic server: **modeled as a periodic task:**

- **execution time = initial execution capacity**
- **period = replenishment period**
- **priorities, capacity, and period are adjusted to meet requirements**

Notes:

UNIVERSIDAD DE CANTABRIA

The analysis of systems with aperiodic tasks has two distinct parts: checking that the soft timing requirements are met, and checking that the hard real-time requirements of periodic and bounded aperiodic events will be met. For the first part, the statistical distribution of event arrival must be used, and queuing theory or simulation results can be applied to obtain the average-case results. The second part is easily done by using the RMA techniques for periodic tasks, as we will show below:

- *Polling*: As a first approximation, since the task that processes aperiodic events is periodic, it can be directly analyzed as a periodic task.
- *Direct event processing*: When the aperiodic events are bounded, their analysis can be done by assuming the worst-case condition in which events arrive at a rate equal to the bound. If, for example, the bound corresponds to a minimum interarrival time, the aperiodic event will be equivalent to a periodic task with period equal to this interarrival time, and execution time equal to the worst-case execution time of the aperiodic task. When the aperiodic events are unbounded, system predictability is destroyed.
- *Background processing*: Since the aperiodic events are processed at the lowest priority level, they do not influence the response times of any other time-critical activities.
- *Sporadic server*: A task executing under the control of a sporadic server is equivalent to a periodic task with its period equal to the replenishment period, and its execution time equal to the initial execution capacity. Consequently, the analysis is carried out using the techniques for periodic tasks.

Analyzing Aperiodic Tasks: own analysis

UNIVERSIDAD DE CANTABRIA

Sporadic tasks

- Direct execution
  - analyze as a periodic task
- Polling server with polling period < interarrival time
  - analyze as a periodic task, and add one polling period to response time
- Sporadic server with capacity >= event execution time, and replenishment period <= interarrival time
  - analyze as a periodic task

Unbounded aperiodic tasks, and other cases

- Use probabilistic analysis: queueing theory, simulation, ...

Sample Problem: Aperiodics

UNIVERSIDAD DE CANTABRIA

The sample problem has the following requirements:

- **emergency event:**
  - **5 msec of work**
  - **arrives every 50 msec, worst-case (*sporadic* event)**
  - **hard deadline 6 msec after arrival**
- **routine event:**
  - **2 msec of work on average**
  - **arrives every 40 msec on average (*unbounded* event)**
  - **desired average response of 20 msec after arrival**

Sample Problem: Scheduling

Emergency Server (ES): we can use either direct event execution at the highest priority or a sporadic server with:

- Execution budget, $C_R = 5$
- Replenishment interval, $T_R = 50$

Routine Server (RS). To achieve fast response we will use a sporadic server with high normal priority (lower than emergency):

- We need to find the execution capacity, and replenishment period
- We will assume that, on average, the emergency server does not execute.

Notes:

The emergency server (ES) can be scheduled using an interrupt service routine (direct event execution) because it has bounded arrivals. Also, a a sporadic server may be created to service the hard-deadline emergency task. The sporadic server would be given an execution budget equal to the execution time of the emergency task, 5 msec, and a replenishment period equal to the minimum interarrival time of emergency events, 50 msec. In both cases, the analysis is the same.

For the routine server (RS) we need a relatively fast response time. Since the arrival pattern is unbounded, the best solution is to use a sporadic server. To meet the soft-deadline requirements, we will use queueing theory to determine the needed execution capacity and replenishment period. We will assume a high priority, immediately below the emergency server, due to the timing requirements. However, we will not consider the effects of the emergency server in the analysis, because we assume that an emergency situation is very unlikely and does not contribute to the average-case behavior.

Routine Server Capacity

Using M/D/1 queueing approximation, the average-case response time is:

$$W = W_q + W_s = \frac{\rho D_q}{2(1-\rho)} + W_s$$

- W_s is the service time = 2 msec. (if highest priority)
- D_q is the average interval between successive dequeuing of events; $D_q = T_R$, if C_R=2 msec.
- ρ is the traffic intensity; $\rho = Dq/I$, where I is the average interarrival time

Solving for the replenishment period:

$$T_R = (C_R - W) + \sqrt{(W - C_R)(W - C_R + 2I)} = 24ms$$

Notes:

UNIVERSIDAD DE CANTABRIA

To compute the required sporadic server parameters, we use the M/D/1 queueing theory approximation which assumes a Poisson distribution of arrivals, deterministic processing time for each event, and a single server. This approximation yields the equation for the average response time for the aperiodic events, which we want to make equal to 20 ms. The response time is composed of two components: W_q represents the average amount of time an event spends waiting in the queue before its processing is started. W_s represents the average amount of time it takes to process the event. If the sporadic server has the highest priority in the system, W_s is the time needed to process one event, which is 2 ms.

W_q is a function of D_q, the average interval between successive dequeuing of events, and ρ, the traffic intensity.If we assume that one aperiodic event is processed each replenishment period, then the initial execution capacity C_R of the sporadic server is made equal to 2 ms., and D_q is T_R. On the other hand, ρ is equal to D_q divided by the average interarrival interval between events, I. The traffic intensity, ρ, must be less than one, otherwise the length of the queue becomes unbounded.

Substituting in the equations for D_q and ρ, and solving for the replenishment period we obtain the equation shown above. The solution is:

$$T_R = (2-20) + \sqrt{(20-2)(20-2+80)} = 24ms$$

Sample Problem: Schedulability Analysis (BIP)

UNIVERSIDAD DE CANTABRIA

The task set is now:

	C	T	D	B	U
ES	5	50	6	0	.100
RS	2	24	n/a	0	.080
τ_1	20	100	100	30	.200
τ_2	40	150	130	10	.267
τ_3	100	350	350	0	.286

Notes:

The full task set now contains five tasks:

- two sporadic servers: ES and RS
- three periodic tasks: τ_1, τ_2, and τ_3

The attributes of these tasks are summarized in the table above. We assume we are still using the basic inheritance protocol, rather than priority ceiling protocol, since if the task set is schedulable using BIP, it will also be schedulable using PCP.

Sample Problem: RTA Test

UNIVERSIDAD DE CANTABRIA

Emergency server:

$$R_{ES} = C_{ES} = 5ms$$

Routine Server (this response time has no practical meaning, except that its less than the period)

$$R_{RS} = C_{ES} \cdot 1 + C_{RS} = 7ms$$

Task τ_1:

$$R_1 = C_{ES} \cdot 2 + C_{RS} \cdot 3 + C_1 + B_1 = 66ms$$

Sample Problem: RTA Test for τ_2 and τ_3

UNIVERSIDAD DE CANTABRIA

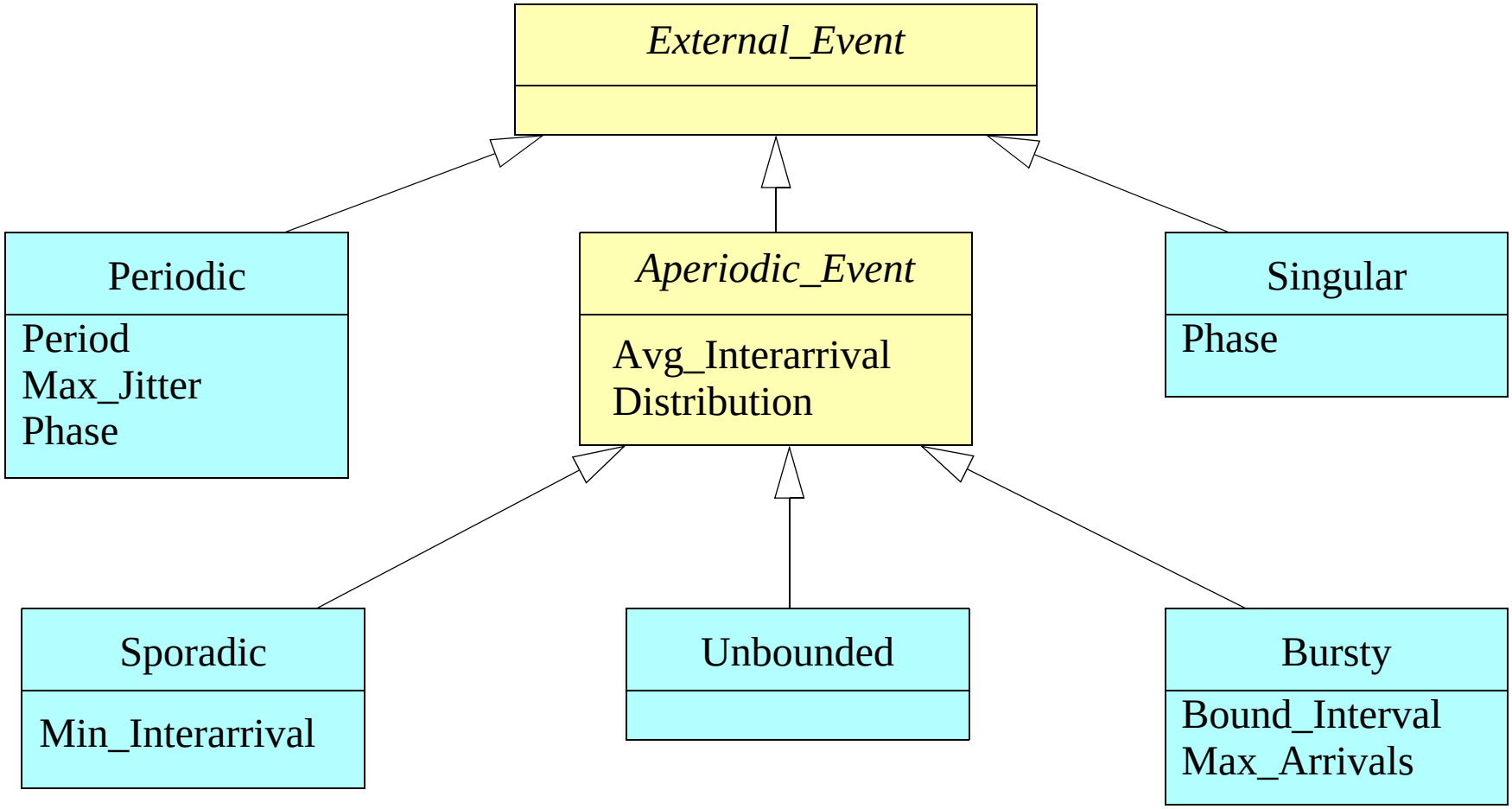
Task τ_2:

$$R_2 = C_{ES} \cdot 2 + C_{RS} \cdot 4 + C_1 \cdot 1 + C_2 + B_2 = 88ms$$

Task τ_3:

$$R_3 = C_{ES} \cdot 6 + C_{RS} \cdot 13 + C_1 \cdot 3 + C_2 \cdot 2 + C_3 + B_3 = 296ms$$

6.5 Modelling Aperiodic Events External Events



Scheduling Parameters (cont'd)