Programming Execution-Time Servers in Ada 2005

by Alan Burns and Andy Wellings

presented at the RTSS 2006

Seminar by António Barros

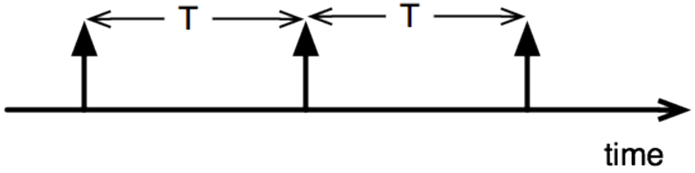
Presentation outline

- The basics
- Execution-time servers
  - Deferrable server
  - Sporadic server
- Ada 2005 execution-time mechanisms
- Building servers in Ada 2005
- Conclusions

The basics

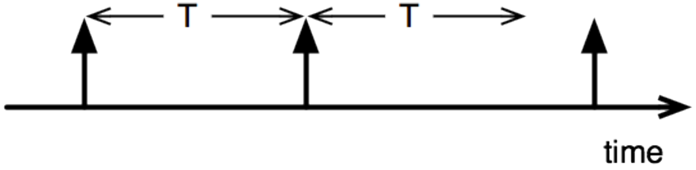
Periodic tasks

- Characterised by:
  - period (T)
  - worst-case execution time (C)
  - relative deadline (D)



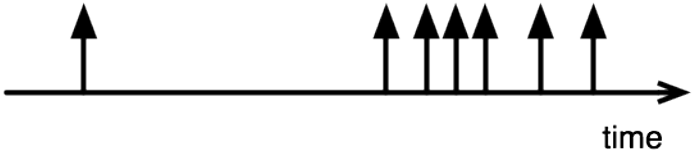
Sporadic tasks

- Characterised by:
  - minimum inter-arrival time (T)
  - worst-case execution time (C)
  - relative deadline (D)



Aperiodic tasks

- Characterised by:
  - single-shot
  - unknown arrival time
  - worst-case execution time (C)
  - relative deadline (D)



The problem

- Periodic and sporadic tasks are tame...
- … but aperiodic tasks are far too unpredictable!

Dealing with it

- Aperiodic tasks executing with background priority... :-(
  - Executing on spare time.
- Use of execution-time servers... :-)
  - Minimum bandwidth assured.

Execution-time servers

Execution-Time Servers

- Characterised by:
  - *budget* – execution time guaranteed for clients
  - *replenishment period* – time to replenish the budget
- Servers require clients to register first!

Deferrable server

- Budget is replenished at the beginning of each period.
  - Foreground priority while budget is not depleted.
- Budget become exhausted.
  - Background priority until next replenishment.

Sporadic server (POSIX)

- A client is released at instant *t*...
  - executes *c* inside budget: at *t*+*T* budget is increased *c*
  - budget is exhausted: wait until replenishment
  - executes *x* and depletes budget: wait for next replenishment, and at *t*+*T* budget is increased *x*

Ada 2005 execution-time mechanisms

Timing events

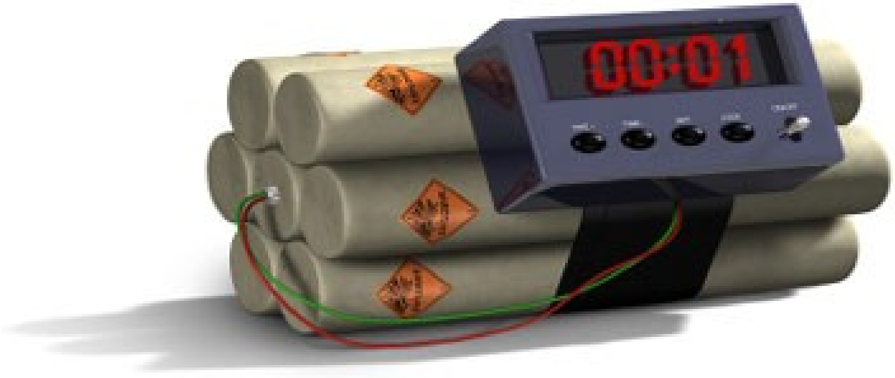
- Events triggered by the progression of the system clock.
- A handler is executed when the associated time is reached.

Execution Time Clocks

- Mechanism that measures the time spent by the system executing a task and services on its behalf.
- Must measure up to 50 years with a maximum granularity of 1 ms.

Execution Time Timers

- A “one-shot” event triggered that executes code when the execution time of a task reaches a specified value.



Group Budgets

- Allows to create execution-time servers.
  - Permits to group tasks...
  - Allocate the group an amount of CPU time...
  - Set a handler to execute when group budget is exhausted...
  - Replenish the budget...
  - Check out the group members and available budget...

Building servers in Ada 2005

Deferrable server

- Group Budget keeps track of group CPU time consumed.
- Single Timing Event signals replenishment periods.

Deferrable server
Adding a task

```
procedure Register(T : Task_Id := Current_Task) is
begin
 if First then
  First := False;
  G_Budget.Add(Params.Budget);
  T_Event.Set_Handler(Params.Period,Timer_Handler’Access);
  G_Budget.Set_Handler(Group_Handler’Access);
 end if;

 G_Budget.Add_Task(T);

 if G_Budget.Budget_Has_Expired then
  Set_Priority(Params.Background_Pri, T);
 else
  Set_Priority(Params.Foreground_Pri, T);
 end if;
end Register;
```

Deferrable server
Replenishing budget

```
procedure Timer_Handler(E : in out Timing_Event) is
 T_Array : Task_Array := G_Budget.Members;
begin
 G_Budget.Replenish(Params.Budget);

 for I in T_Array’range loop
  Set_Priority(Params.Foreground_Pri,T_Array(I));
 end loop;

 E.Set_Handler(Params.Period,Timer_Handler’Access);
end Timer_Handler;
```

Deferrable server
Managing budget exhaustion

```
procedure Group_Handler(G : in out Group_Budget) is
 T_Array : Task_Array := G_Budget.Members;
begin

 for I in T_Array’range loop
  Set_Priority(Params.Background_Pri,T_Array(I));
 end loop;
end Group_Handler;
```

Sporadic server

- Server deals with a single task (in the example).
  - The task release mechanism is embedded in the server.
- Group Budget keeps track of group CPU time consumed.
- Multiple Timing Events signals replenishment periods.
  - Each Timing Event must know the amount of budget that must be returned.

Sporadic server
The task

```
task body Sporadic_Task is
begin
 Sporadic_Controller.Register;

 loop
  Sporadic_Controller.Wait_For_Next_Invocation;
  -- undertake the work of the task
 end loop;

end Sporadic_Task;;
```

Sporadic server
Adding a task

```
procedure Register(T : Task_Id := Current_Task) is
begin
 G_Budget.Add_Task(T);

 G_Budget.Add(Params.Budget);
 G_Budget.Set_Handler(Group_Handler’Access);

 Release_Time := Clock;
 Start_Budget := Params.Budget;
end Register;
```

Sporadic server
Releasing task

```
procedure Release_Sporadic is
begin
 Barrier := True;
end Release_Sporadic;


entry Wait_For when Barrier is
begin
 if not G_Budget.Budget_Has_Expired then
  Release_Time := Clock;
  Start_Budget := G_Budget.Budget_Remaining;
  Set_Priority(Params.Foreground_Pri,ID);
 end if;

 Barrier := False;
 Task_Executing := True;
end Wait_For;
```

Sporadic server
Task finishes execution

```
entry Wait_For_Next_Invocation when True is
begin
 -- work out how much budget used, construct
 -- the timing event and set the handler
 Start_Budget := Start_Budget – G_Budget.Budget_Remaining;

 TB_Event := new Budget_Event;
 TB_Event.Bud := Start_Budget;
 TB_Event.Set_Handler(Release_Time+Params.Period,
                      Timer_Handler’Access);

 Task_Executing := False;
 requeue Wait_For with abort;
end Wait_For_Next_Invocation;
```

Sporadic server
Replenishing budget

```
procedure Timer_Handler(E : in out Timing_Event) is
 Bud : Time_Span;
begin
 Bud := Budget_Event(Timing_Event’Class(E)).Bud;

 if G_Budget.Budget_Has_Expired and Task_Executing then
  Release_Time := Clock;
  Start_Budget := Bud;
  G_Budget.Replenish(Bud);
  Set_Priority(Params.Foreground_Pri,ID);
 elsif not G_Budget.Budget_Has_Expired and
       Task_Executing then
  G_Budget.Add(Bud);
  Start_Budget := Start_Budget + Bud;
 else
  G_Budget.Add(Bud);
 end if;
end Timer_Handler;
```

Sporadic server
Managing budget exhaustion

```
procedure Group_Handler(G : in out Group_Budget) is
begin
 -- a replenish event required for the used budget

 TB_Event := new Budget_Event;
 TB_Event.Bud := Start_Budget;
 TB_Event.Set_Handler(Release_Time+Params.Period,
                      Timer_Handler’Access);

 Set_Priority(Params.Background_Pri,ID);

 Start_Budget := Time_Span_Zero;
end Group_Handler;
```

Conclusions

- Ada 2005 includes the mechanisms to build execution-time servers in the standard...
  - … as long as the run-time supports them.

Thanks for your attention!