Lecture 9 :

Priority

plus

ATC in HRT-HOOD

SWINBURNE UNIVERSITY OF TECHNOLOGY

Real-Time Attributes - 1

- The following must be documented for each HRT-HOOD Protected object:
  - ceiling priority (later)
  - operation WCET ("worst case…") for each op
  - operation budget
    - if exceeded the op is supposed to terminate
  -
  - How do we know these?

Real-Time Attributes - 2

- The following must be documented for each cyclic and sporadic object:
  - deadline
  - operation WCET ("worst case…") for each
  - operation budget
  - thread WCET
  - thread budget
  - period (cyclic only)
  - offset (starting delay for cyclic, latency for sporadic)
  - minimum arrival time (or equiv) - sporadic only
  - priority, ceiling priority (later)
  - precedence constraints - what object has to execute before, etc
  - importance (soft or hard)

Latency & Completion

- For a Cyclic (text Fig. 12.1):

  time specified by program

  granularity of clock

  interrupts disabled (or in BIOS)

  Ready (runnable but not running)

  Running : cycle completes

  |xxxxxxxxxxxx|

- For a Sporadic
  - (assuming Worker blocked waiting for start, ie op request):

    Start call

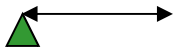
    Start ends

    Worker ready

    Worker running : op completes

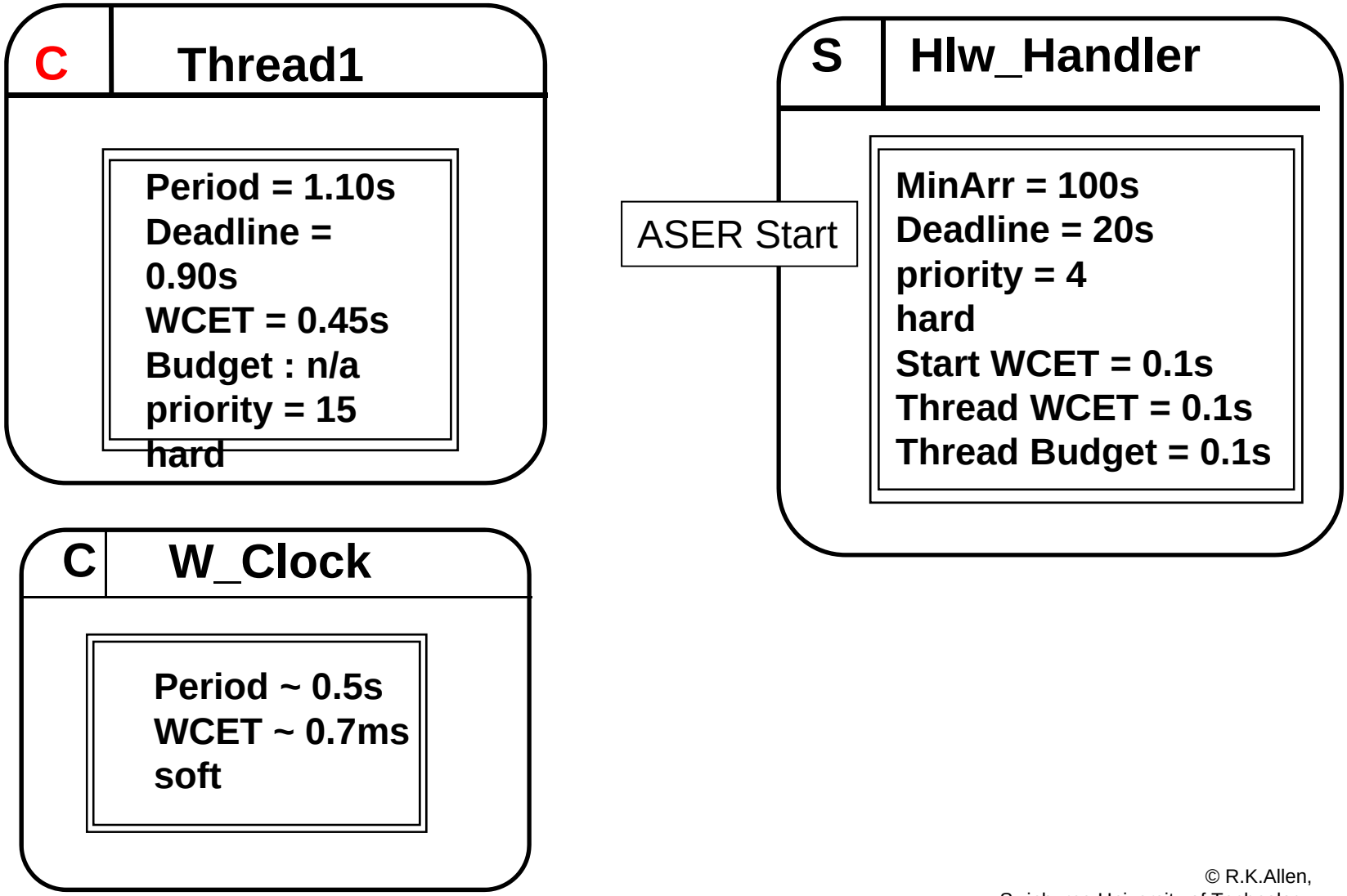
    |xxxxx|

    |xxxxxxxxxxxxxxxxx|



- The Ready time is interference from other tasks.

RT Attributes on Diagrams

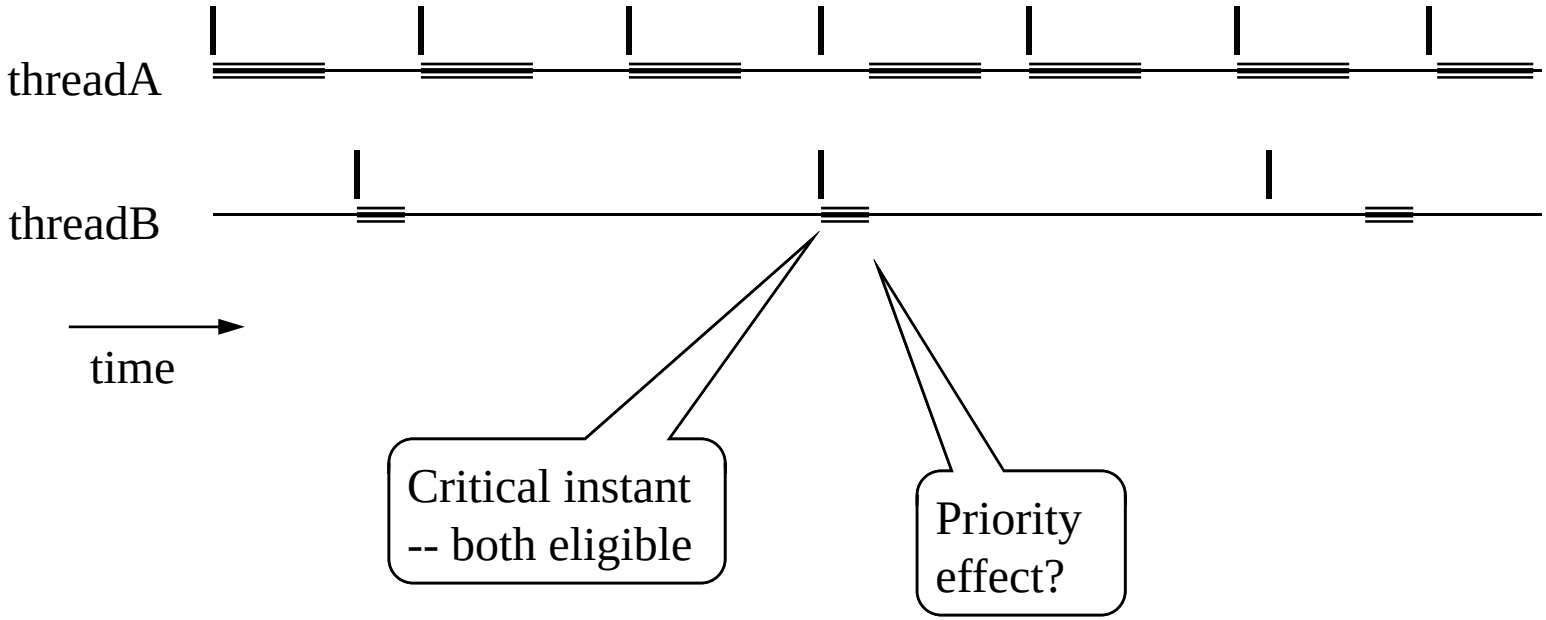


Priority

- A number attached to a task to guide the scheduler
- pragma Priority(9);
  - values platform dependent:
    GNAT:
    - System.Priority is Integer range 0..30
      System.Default_Priority = 15
    - Interrupt priority = 31
  - mapping to smaller sets
  - usually static, but
    - Ada.Dynamic_Priorities has Set_Priority
  - Use within task or protected spec.
- automatic promotion is defined in Ada95

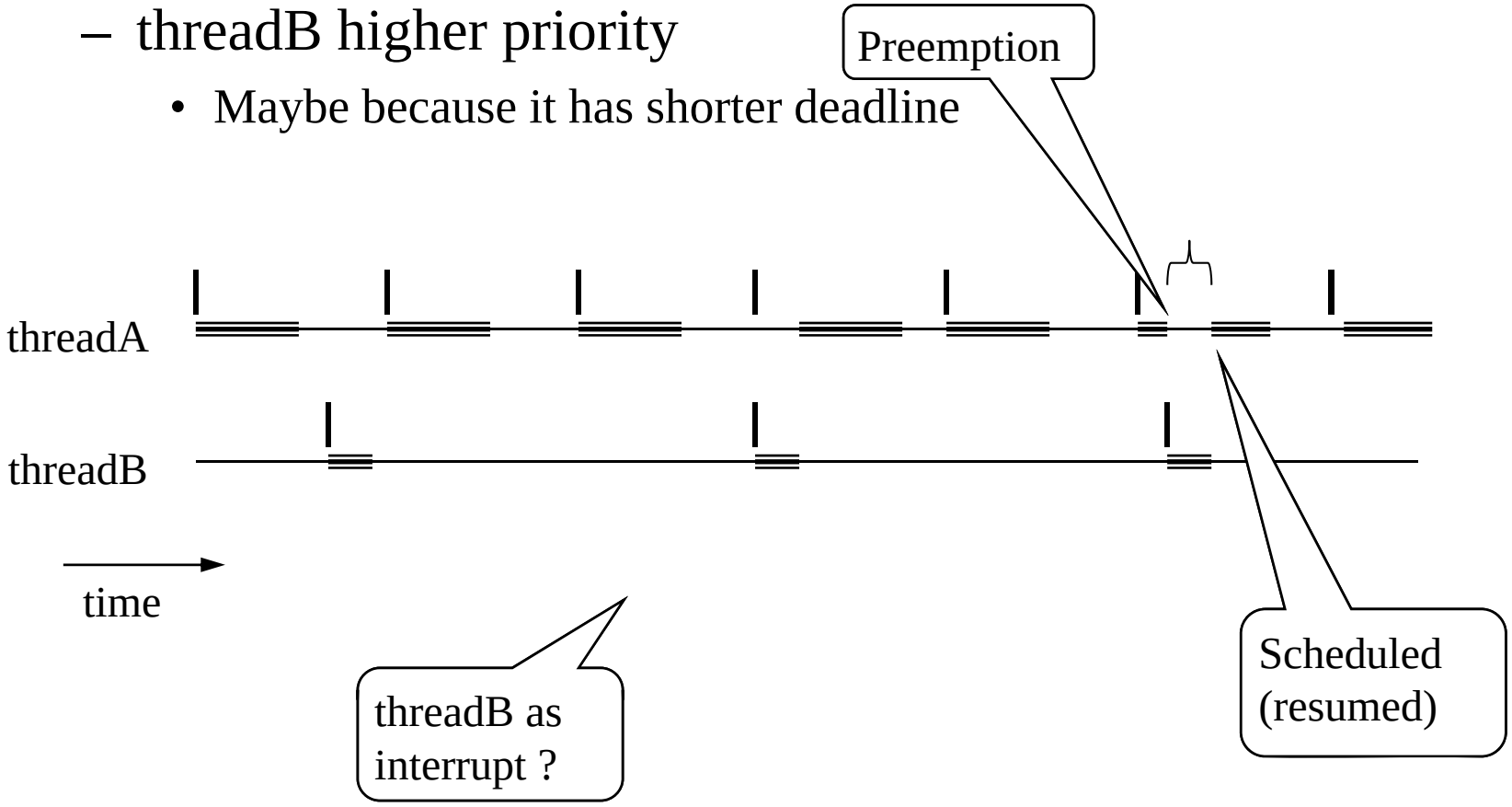
Example

- Two periodic threads
  - periods possibly harmonically related
  - no pre-emption



Example with Preemption

- With pre-emption
  - threadB higher priority
    - Maybe because it has shorter deadline



Priority Inversion

Priority principle:

> If two tasks of different priority on same cpu are both eligible for running then the lower priority task should not be running.

Priority inversion = any violation of this principle.

Good RT design minimises priority inversion.

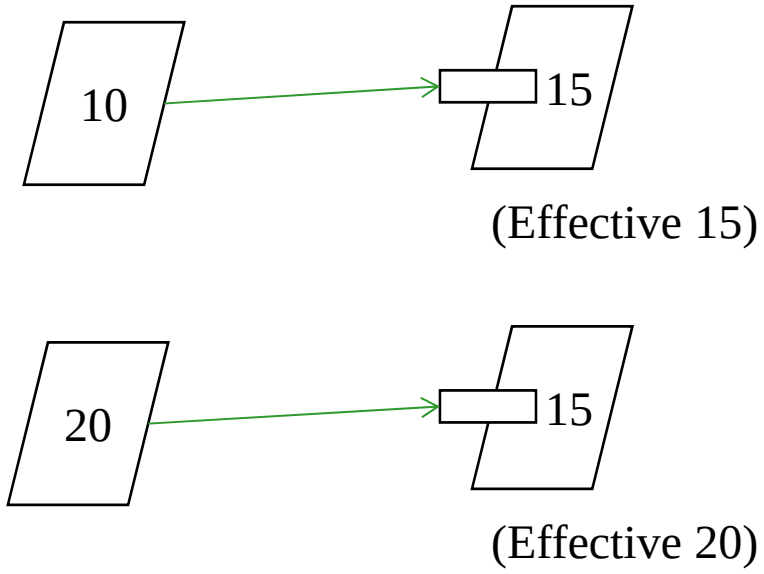
- Number of occurrences
- Duration of each

Examples of inversion:

- in non-pre-emptive system, low priority task runs until blocked
- high priority task behind low in entry queue
- result of priority promotion (in strange cases involving rendezvous)

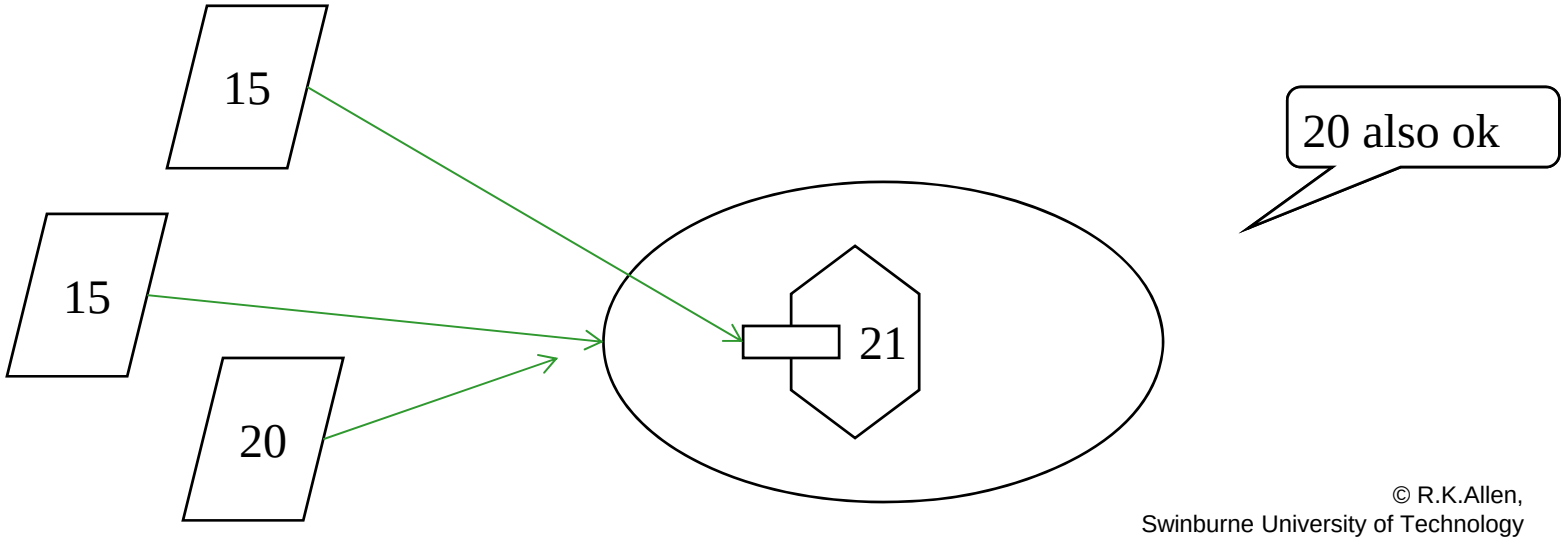
Priority Promotion -1

- This is designed to reduce the duration of inversion
  - to increase throughput, reduce response times.
- The effective priority is temporarily >= the normal priority
- During rendezvous (rdv):
  A server task executes its accept at the higher of its own & the client's priority.



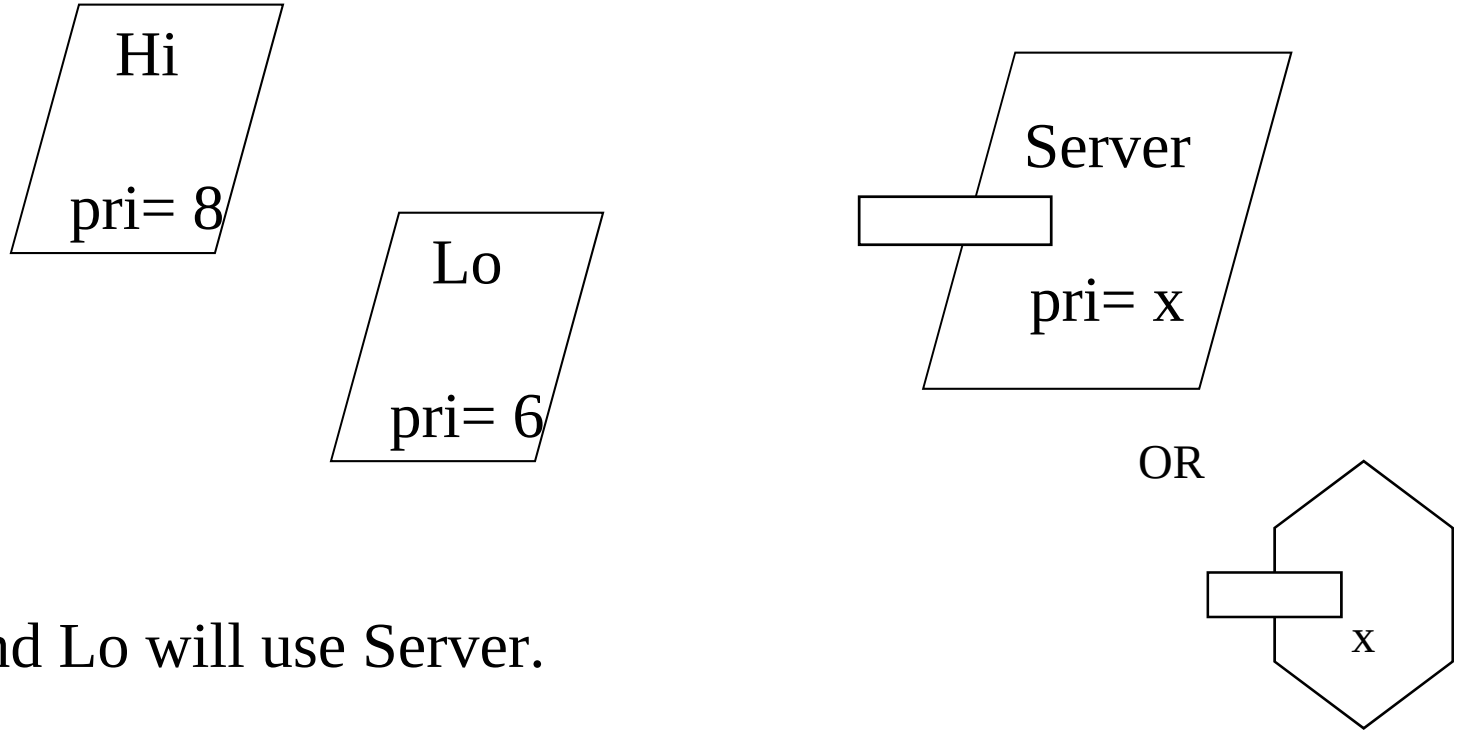
Priority Promotion - 2

- During protected operations:
  The calling thread's priority is increased to the specified "ceiling priority" of the protected object.
  - Normal implementation requires the increase, ie exception raised if the client thread has a current priority > than "ceiling"
  - hence on single-core cpu can have efficient mutual exclusion, no lock bit



Priority Inversion Scenario

- The following examples show that even with promotion, inversion cannot be 100% eliminated:

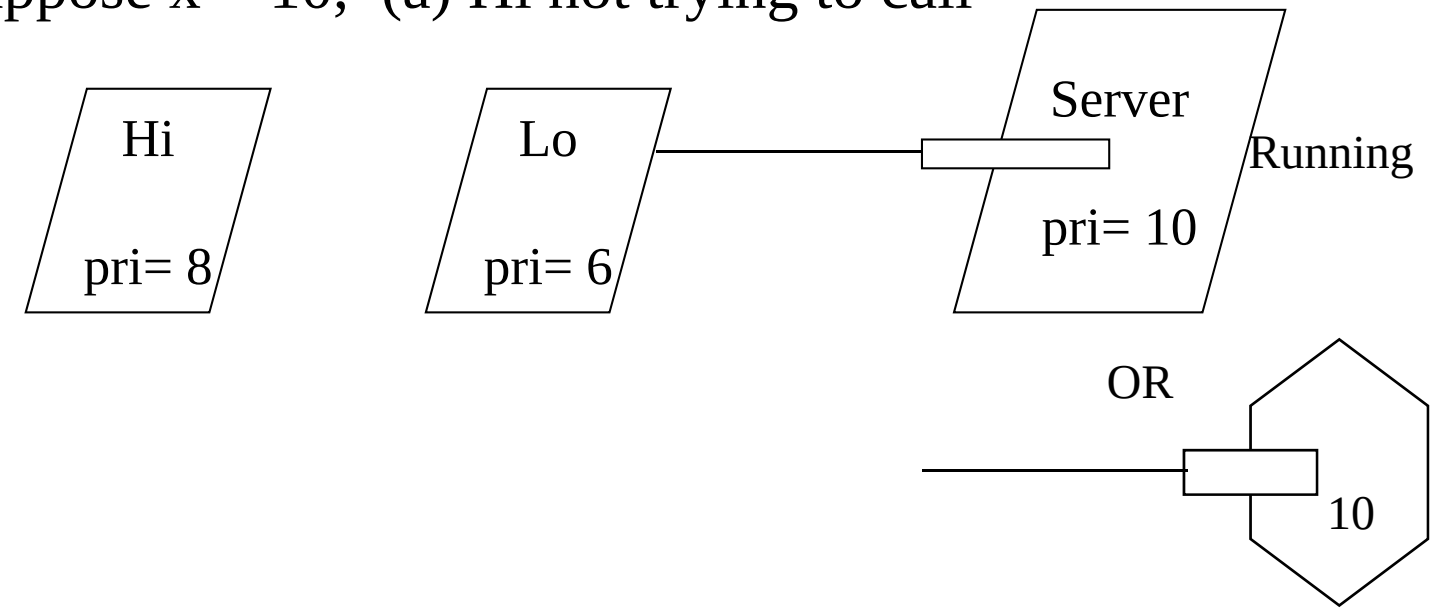


Hi and Lo will use Server.

What priority should Server have?

Priority Inversion Examples - 1

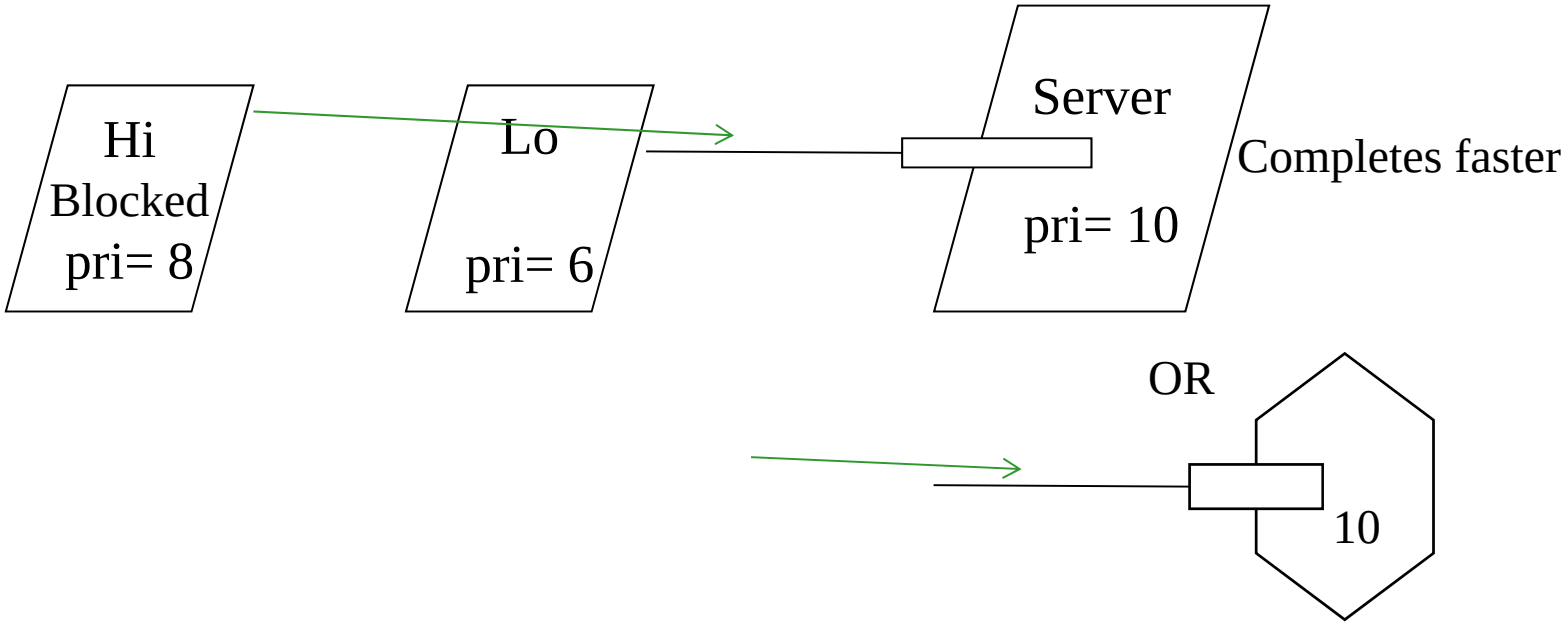
Suppose x = 10, (a) Hi not trying to call



This seems to be ok **but** we should think of Server as being an extension of Lo – Lo's work is being done, Hi's isn't.

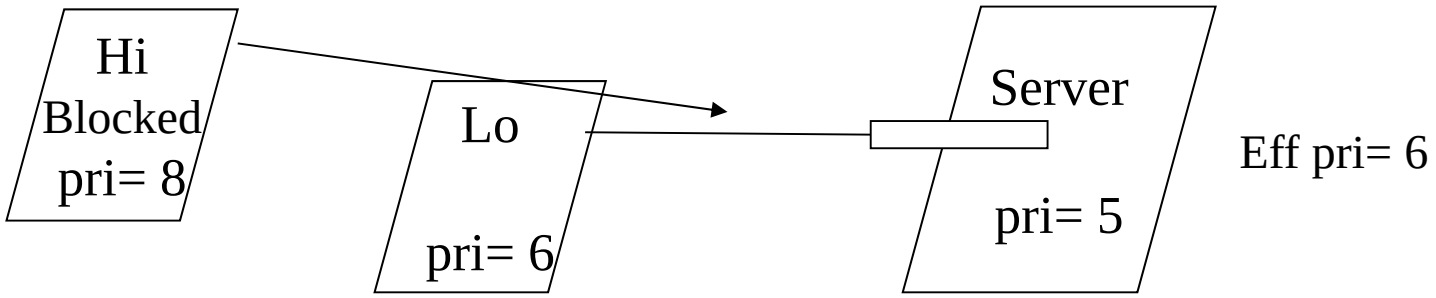
Priority Inversion Examples - 1

(b) Hi now wants to use Server. Again inversion but pri=10 reduces the time.



Priority Inversion Examples - 2

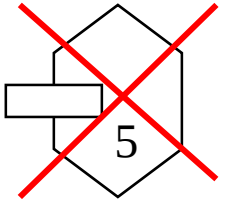
Suppose x = 5 (promotion will apply) Lo starts rdv then Hi makes entry call



Hi is blocked until rdv complete – unavoidable inversion, but likely to be longer (as another task at pri 7 or 8 could run).

[Aside: It would help if the eff priority of Server were increased to 8 when Hi makes its call, but that isn't in standard Ada.]

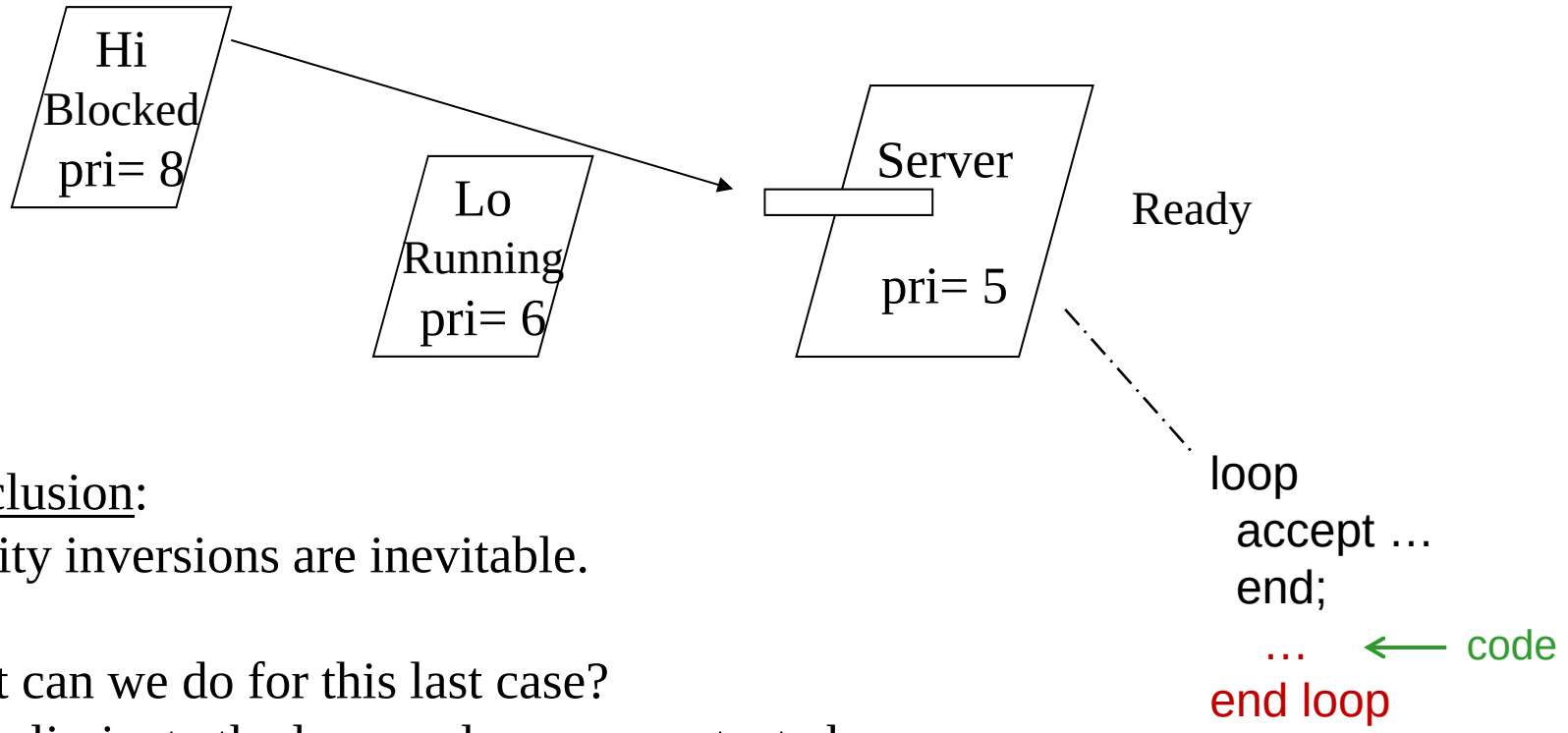
Note: We can't show a protected object on this page because its (ceiling) priority is not allowed to be 5 (below its clients).



Priority Inversion Examples - 3

Suppose rdv now ends: Lo runs in preference

– inversion



Conclusion:
priority inversions are inevitable.

What can we do for this last case?
Ans: eliminate the loop code: use a protected procedure instead and use ceiling priority.

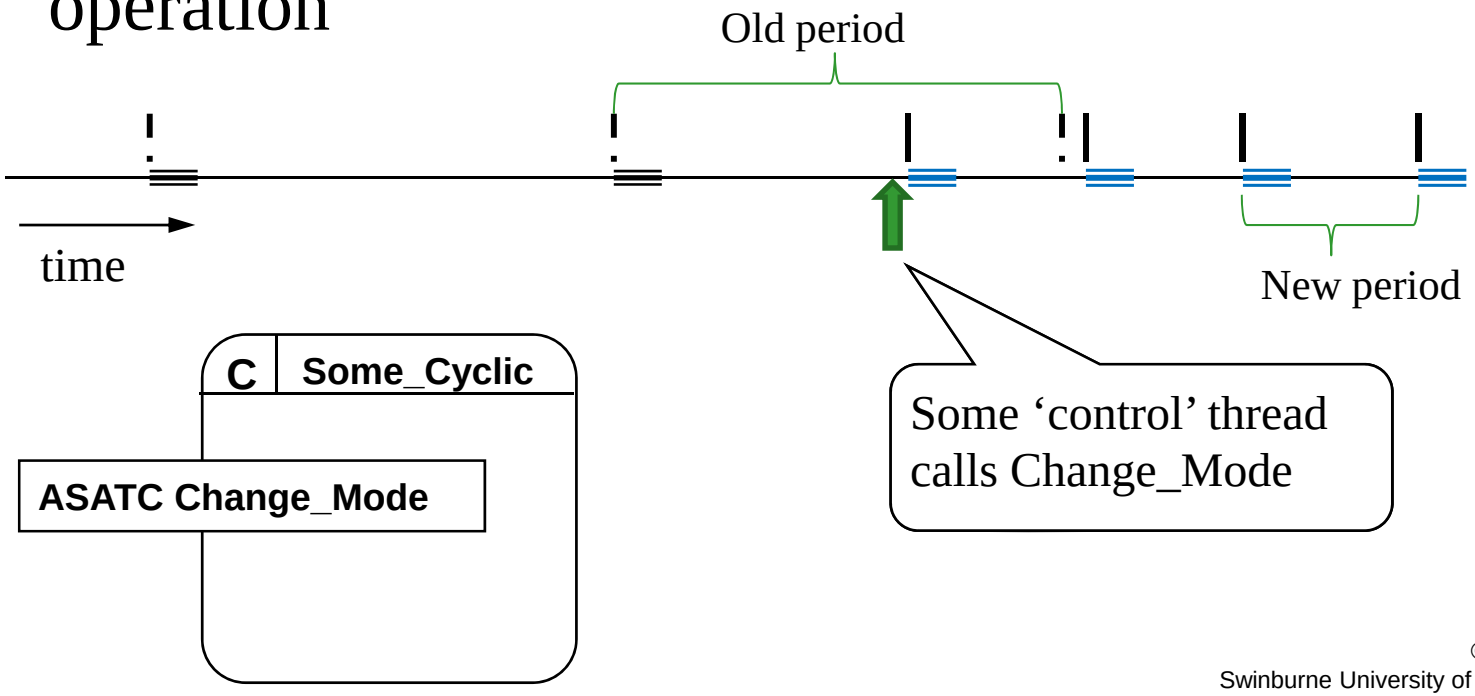
Restrictions of Protected Code

- Try to reduce length of inversion
- no delay
- no rendezvous
- procedures can't call entries
  - Because … (should be obvious)

- BTW:
  - HRT-HOOD overall restriction: no delays except in Cyclics!
  - But in RTP allowed in turnout sporadics

ASATC

- ASynchronous (request for) Asynchronous Transfer of Control
- For mode change (run-time configuration),
- eg a Cyclic may change its period &/or actual operation



How to stop Sleeping

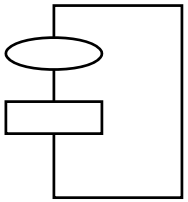
- In Ada, replace delay t; by a *timed entry call* (usually to a protected entry)

Code with delay:	Ada protected:
select	procedure Cancel is … ;
Prot.Wait_Cancel;	entry Wait_Cancel when Cancelled
or	is…
delay	

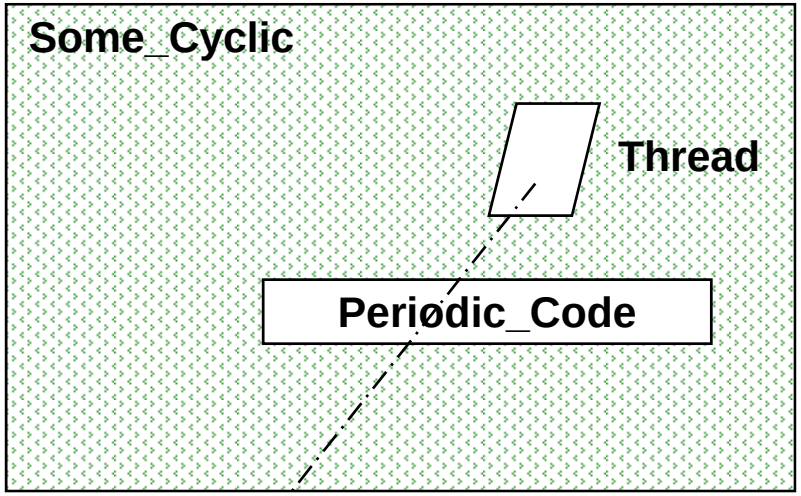
Some 'control' thread calls Cancel

- (Using rendezvous, a *selective wait* statement can combine *accept* with *delay* – you don't need to know.)
- In Java, replace Thread.sleep(t) by wait(t) and arrange the control thread to call notifyAll()

HRT_HOOD Implementation of Cyclic



Some_Rtatt

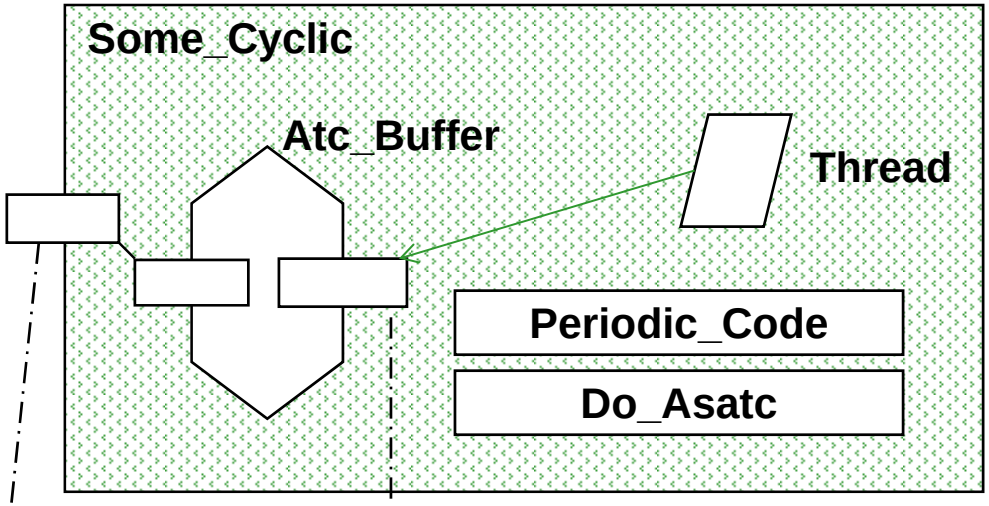



```
type Mode is (Normal, Alert);

Period : constant array (Mode)
of Duration := (5.0,1.5);
-- here in a separate package;
could be in Some_Cyclic
```

```
begin
  loop
    delay until T;
    Periodic_Code;
    T := T + Some_Rtatt.Period(
              Current_Mode);
  end loop;
end Thread;
```

Implementation of ASATC - 1



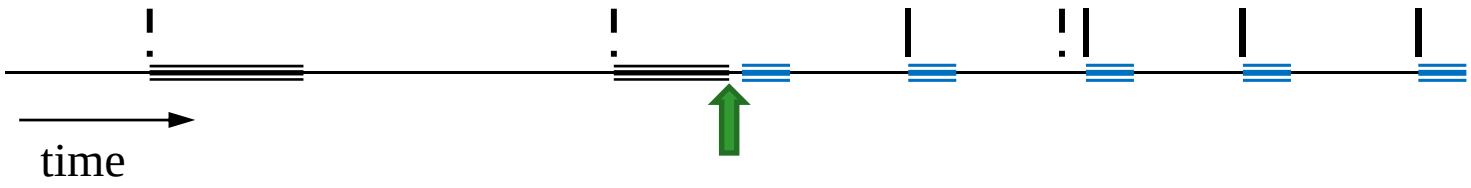
ASATC Change_Mode
(M : in MODE)

```
entry Get_Asatc(M : out Mode)
        when Asatc_Occurred is
begin
            M := Mode_Recvd;
end Get_Asatc;
```

```
   New_M : Mode;
begin
  loop
    select
      Atc_Buffer.Get_Asatc(New_M);
      Do_Asatc(New_M);
    or
      delay until T;
      Periodic_Code;
    end select;
    T := T + Some_Rtatt.Period(
      Current_Mode);
  end loop;
end Thread;
```

Abandoning Periodic_Op

- What if we want a more immediate effect, even if the cyclic operation is underway
- Eg, drawn with a long operation that changes to a shorter one and with a shorter period

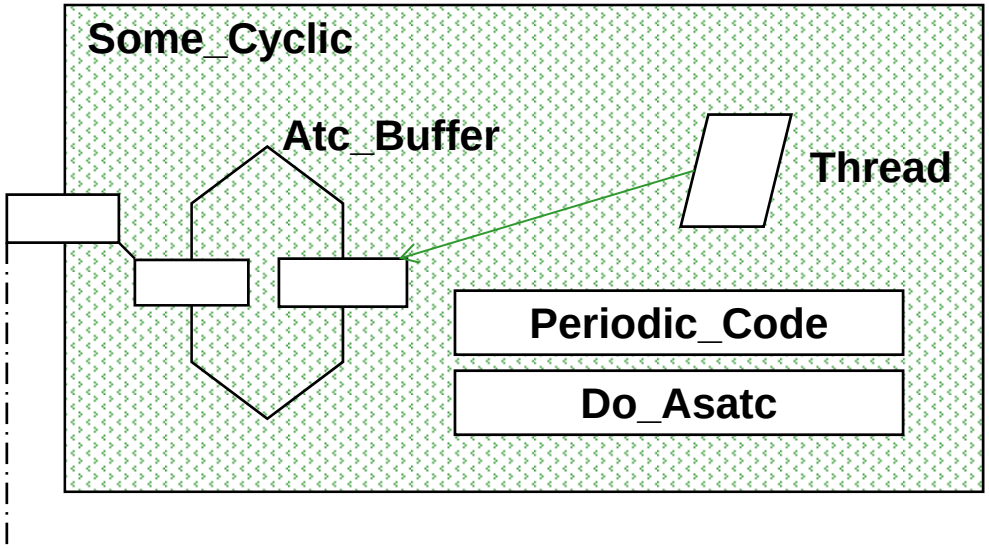


ATC in general

- Asynchronous transfer of control
  - like an interrupt thrown into another thread
  - C POSIX library has signal()
  - Java has interrupt(), suspend().
  - Ada83 has abort T1; (fatal)
  - Ada95 has

```
select
        entry call;
        …
then abort
        statements…
end select;
```

Implementation of ASATC - 2



To fit the previous time diagram, Do_Asatc changes T as well as Current_Mode

```
begin
  loop
    select
          Atc_Buffer.Get_Asatc(New_M);
          Do_Asatc(New_M);
    or
        delay until T;
        select
          Atc_Buffer.Get_Asatc(New_M);
          Do_Asatc(New_M);
        then abort
          Periodic_Code;
        end select;
    end select;
    T := T + Some_Rtatt.Period(
          Current_Mode);
  end loop;
end Thread;
```

Note: this is advanced coding & you won't use it.
(Also too complex for exam.)