

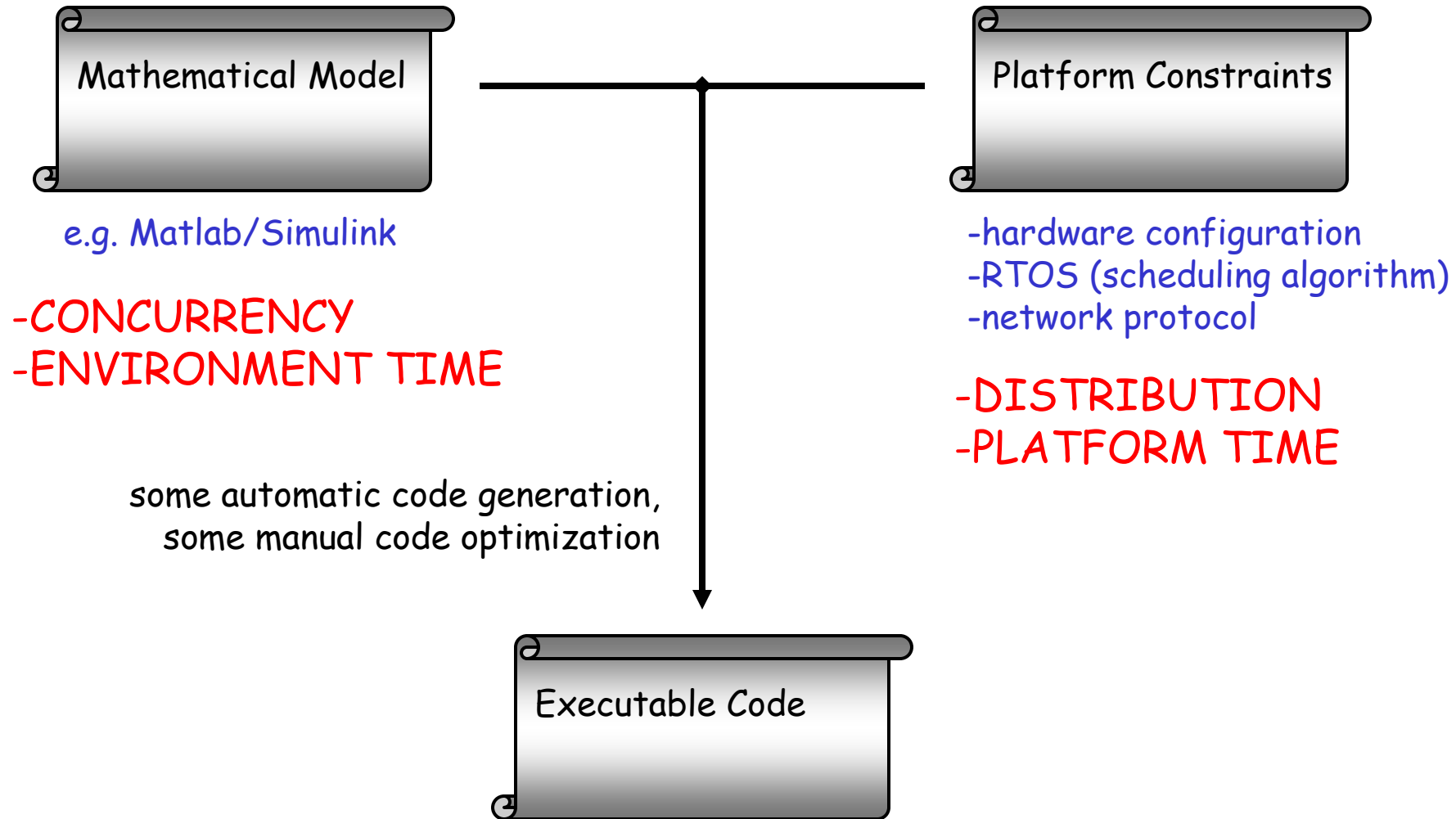
Giotto

Thomas A. Henzinger, Benjamin Horowitz, Christoph Kirsch

UC Berkeley

www.eecs.berkeley.edu/~fresco/giotto

Control Software Development



Control Software Development

Mathematical Model

e.g. Matlab/Simulink

Platform Constraints

-hardware configuration
-RTOS (scheduling algorithm)
-network protocol

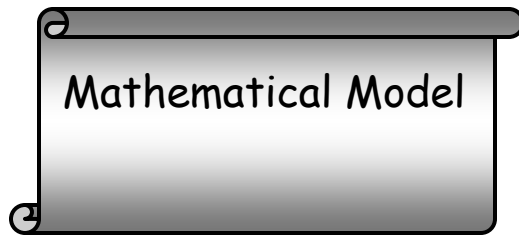
Problems:

- close correspondence between model and code is lost with code optimization
- if either model or platform changes, the entire process needs to be repeated

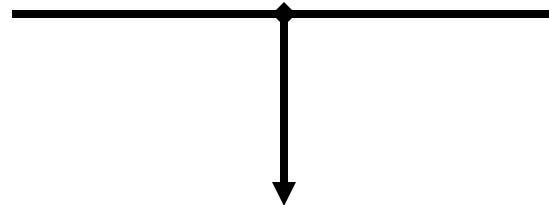
some automatic code generation,
some manual code optimization

Executable Code

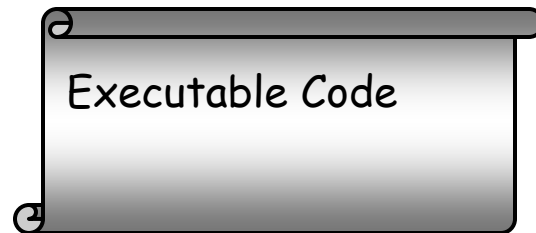
Control Software Development



e.g.
What is the control equation?
What is the sampling rate?



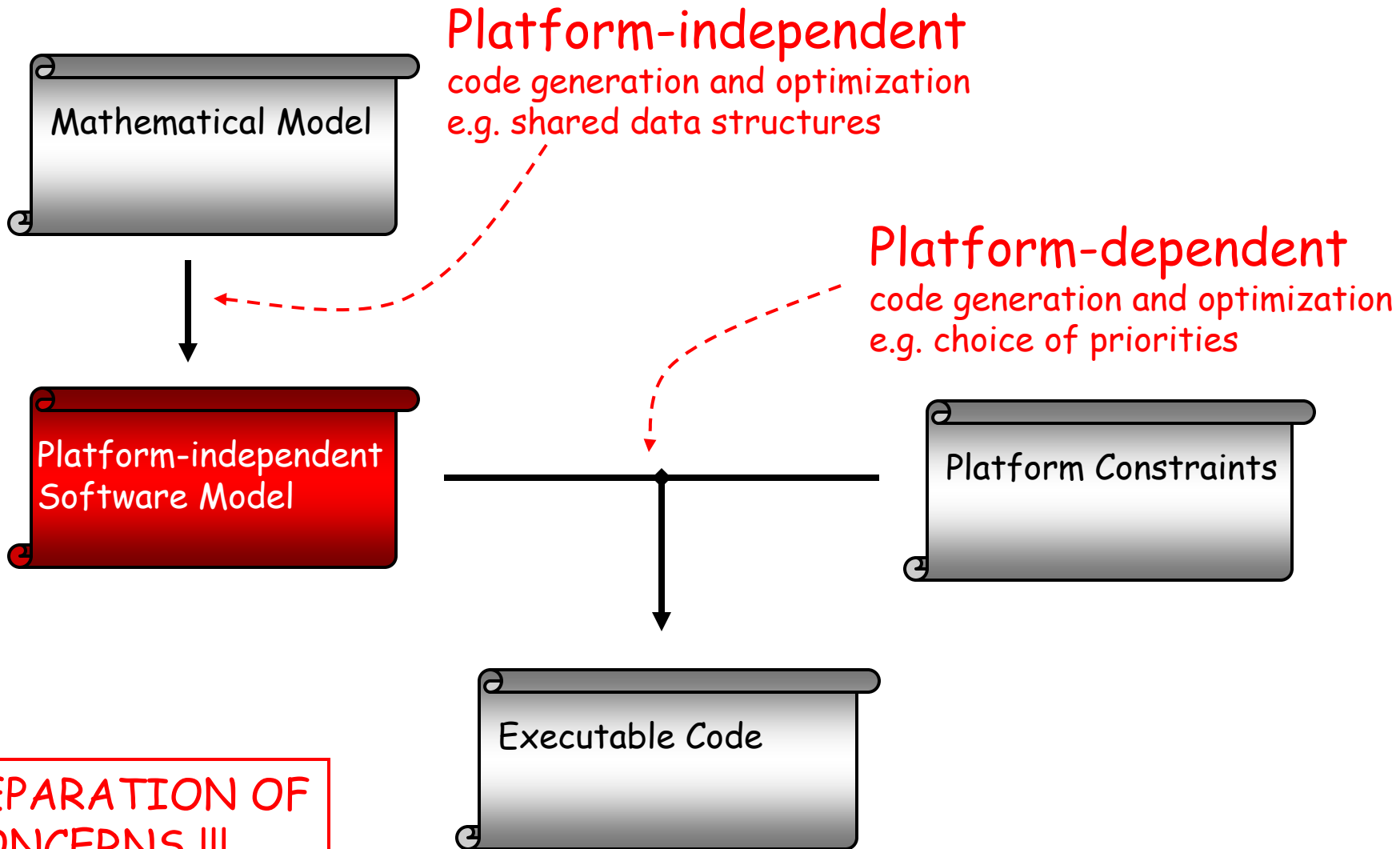
e.g.
Which CPU executes the control procedure?
What priority has the execution?



e.g.
Which procedure computes the control equation?
Which (clock) event triggers the computation?

-still CONCURRENCY
-still ENVIRONMENT TIME

Control Software Development



Platform-independent
code generation and optimization
e.g. shared data structures

Platform-dependent
code generation and optimization
e.g. choice of priorities

SEPARATION OF CONCERNS !!!

Motivation: Flight Control Software



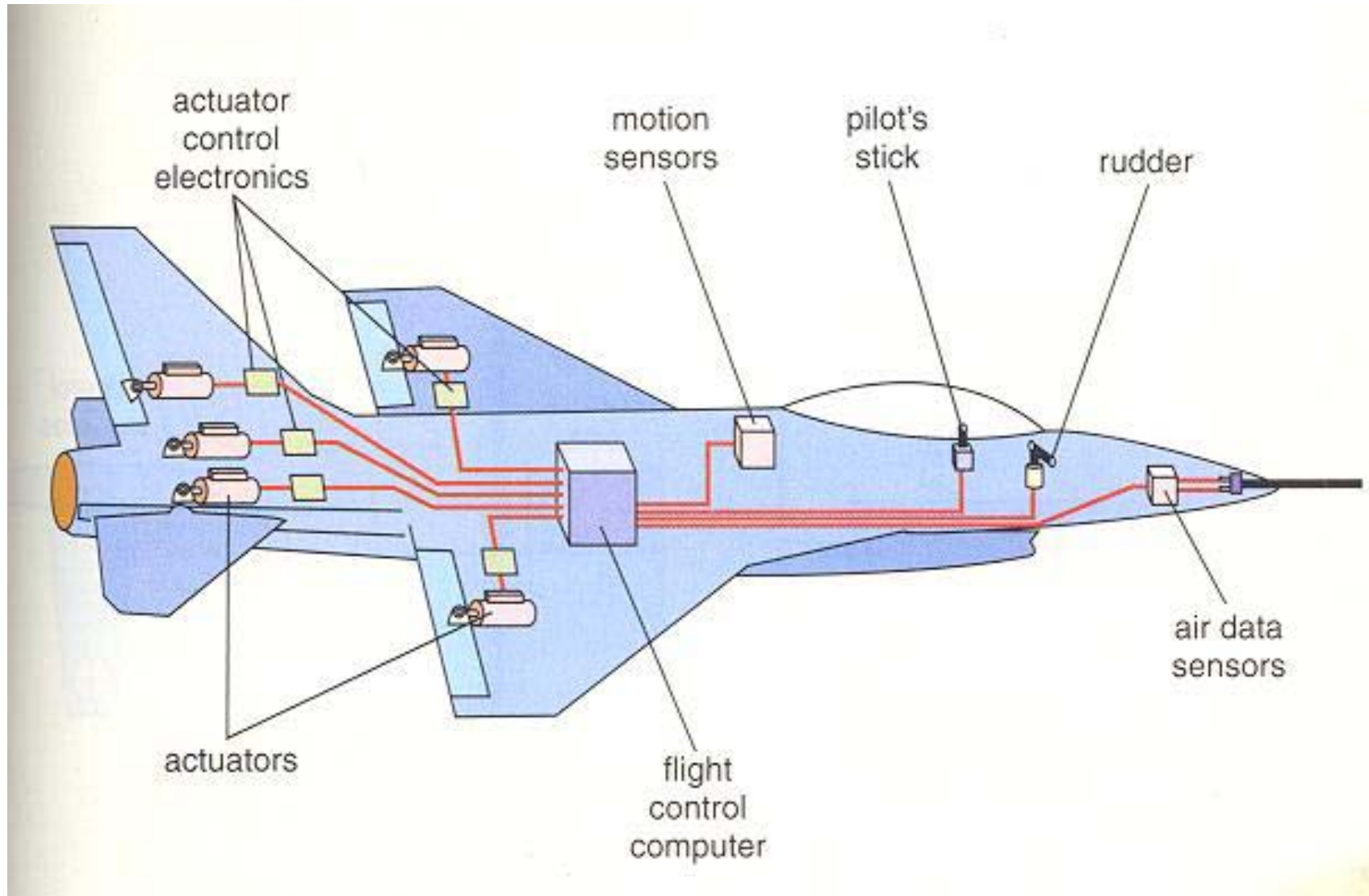
Single CPU.

Motivation: Flight Control Software

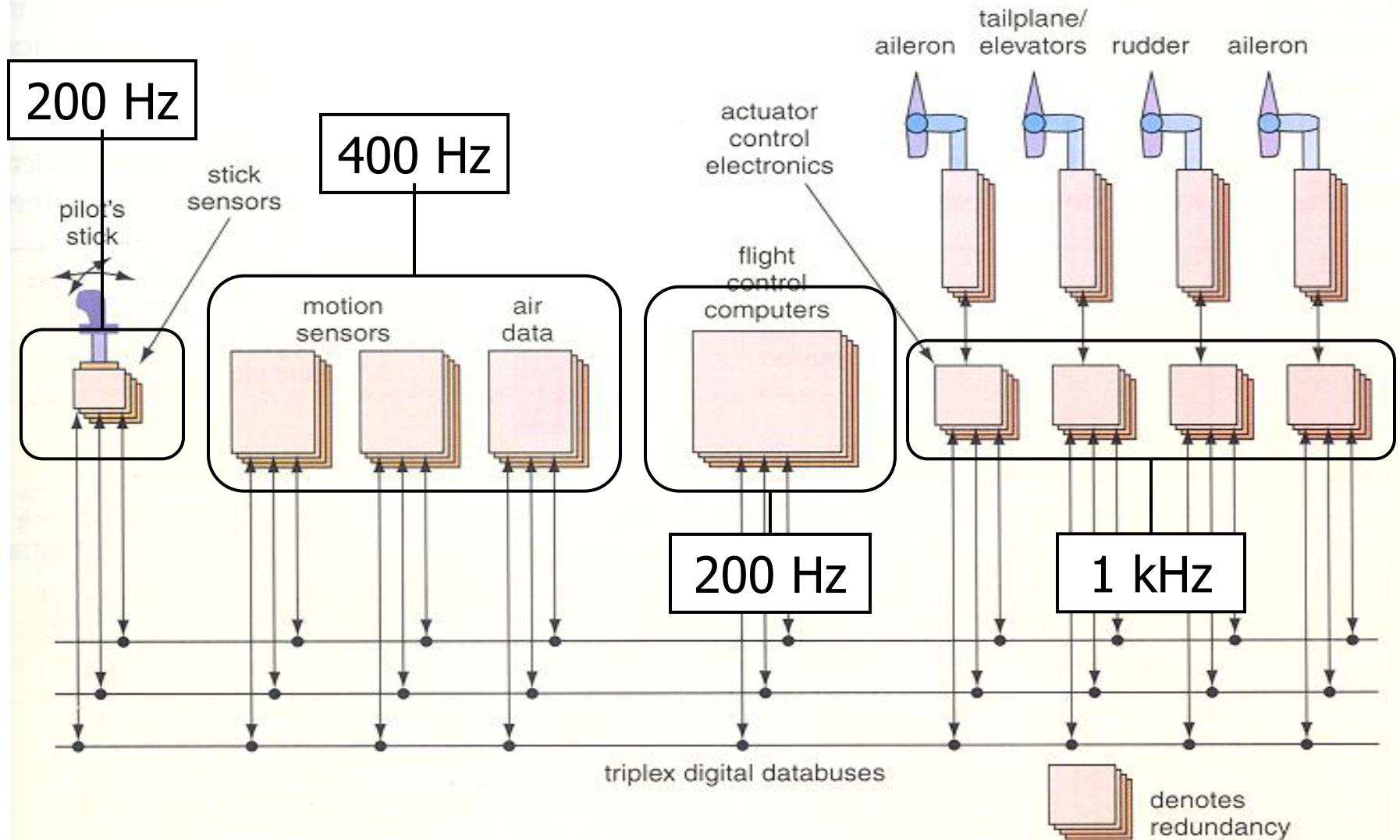


Two or three connected CPUs.

Motivation: Flight Control Software



Motivation: Flight Control Software



Platform-independent Software Model

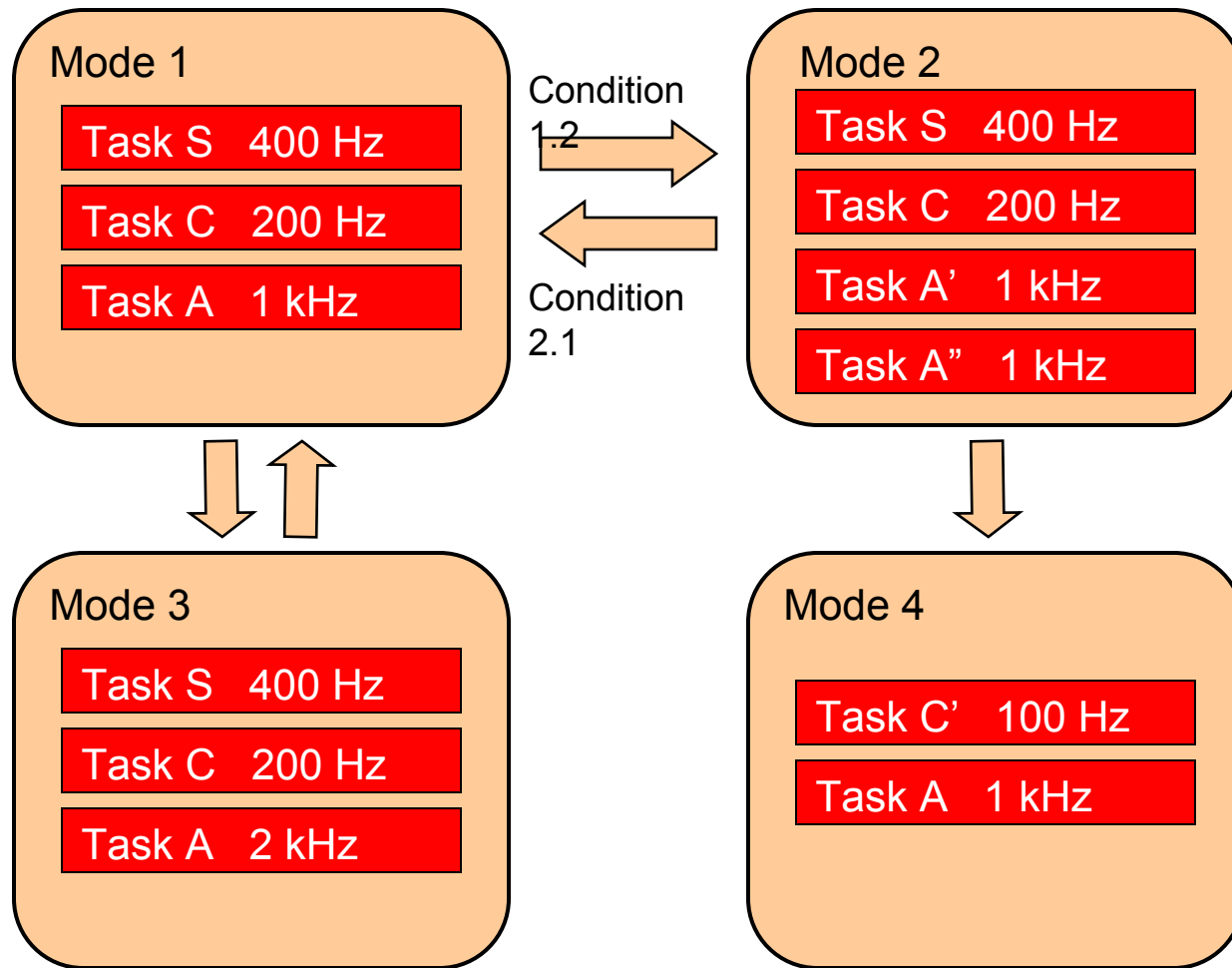
1. Concurrent periodic tasks:

- sensing
- control law computation
- actuating

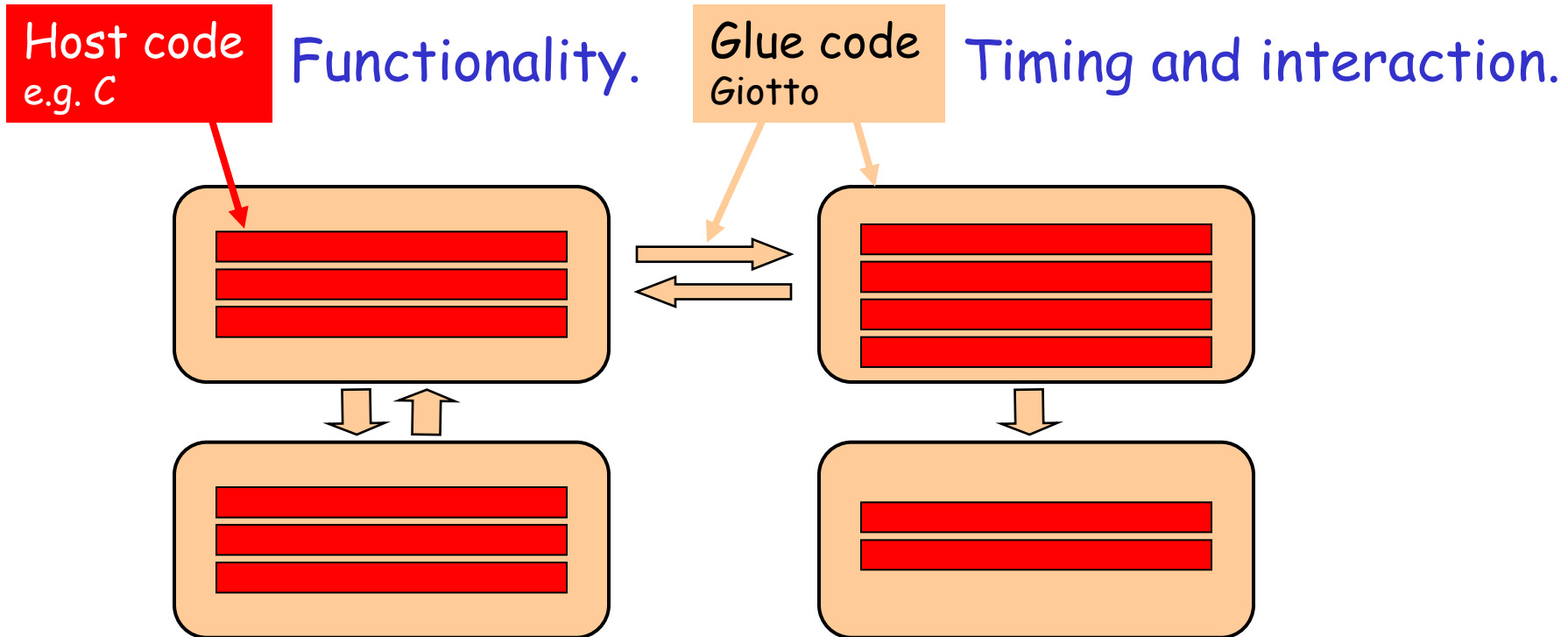
2. Multiple modes of operation:

- navigational modes (autopilot, manual, etc.)
- maneuver modes (taxi, takeoff, cruise, etc.)
- degraded modes (sensor, actuator, CPU failures)

Platform-independent Software Model

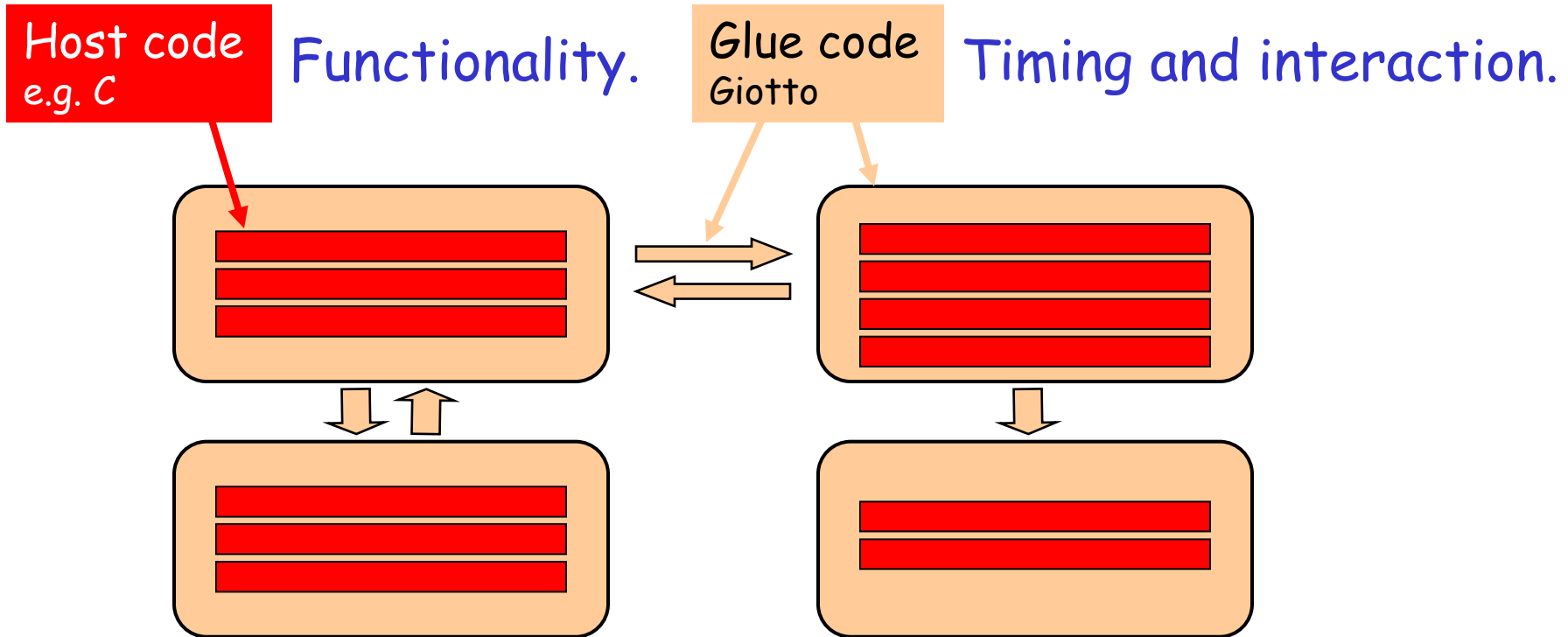


Platform-independent Software Model



- Concurrency,
not distribution.
- Environment time,
not platform time.

Platform-independent Software Model



This kind of software is understood:
Host code may be generated automatically.

The software complexity lies in the glue code:
Giotto enables requirements-driven rather than platform-driven glue-code programming.

The Giotto Programmer's Model

Programming in terms of environment time:

- time-triggered task invocation
- tasks have fixed duration (\geq WCET)
- tasks are not preemptable

Implementation in terms of platform time:

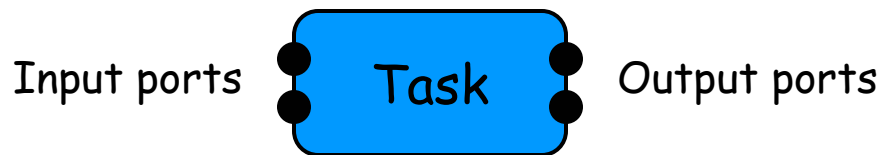
- need access to (logical) global time,
no other platform requirements
- tasks may finish early,
but outputs cannot be observed early
- tasks may be preempted

Similar to the synchronous programmer's model,
only simpler (no fixpoint issues).

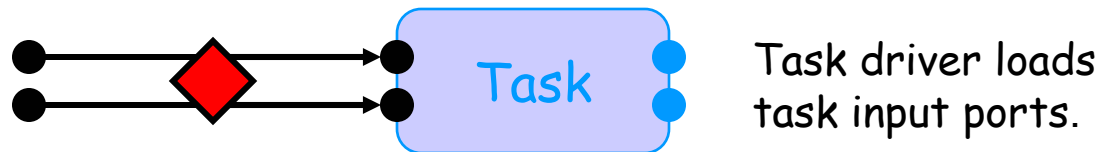
The Giotto Programmer's Model

Given:

1. Units of scheduled host code (application-level tasks).
e.g. control law computation



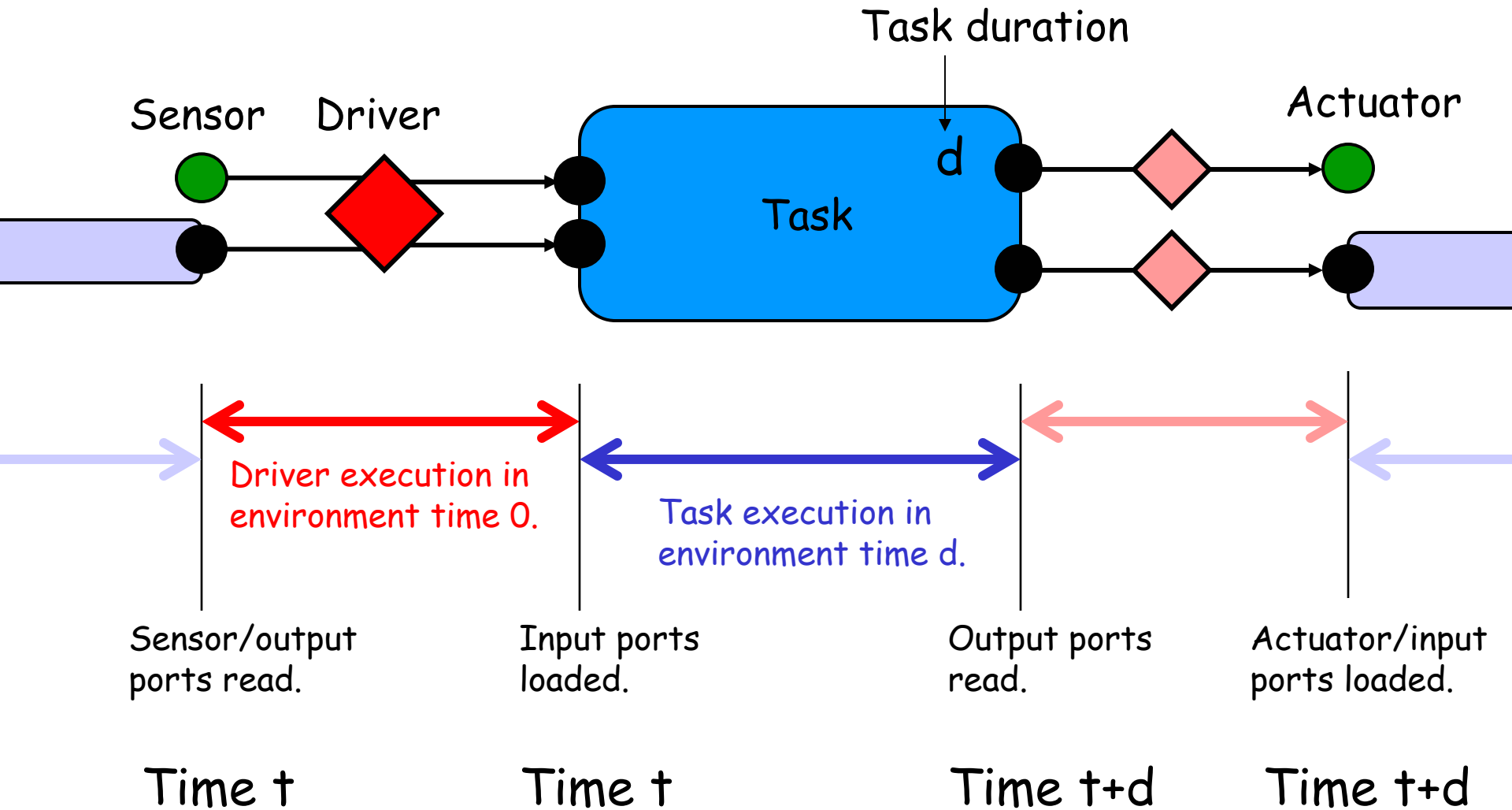
2. Units of synchronous host code (system-level drivers).
e.g. device drivers



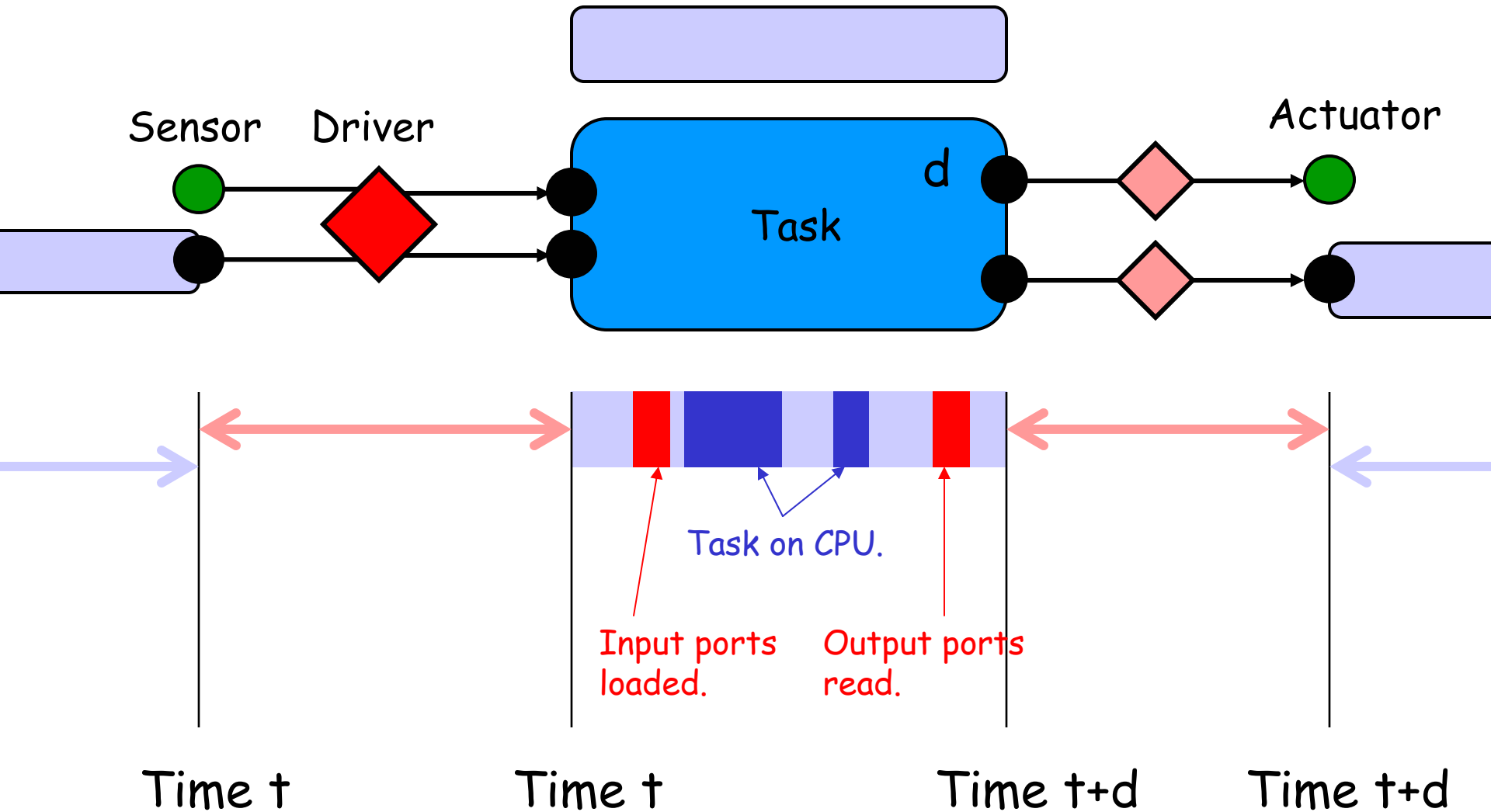
3. Real-time requirements and data flow between tasks.

Giotto: Glue code that calls 1. and 2. in order to realize 3.

Environment Timeline (defined by Giotto semantics)



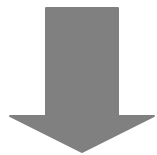
Platform Timeline (chosen by Giotto compiler)



Platform Independence ensures Predictability

Input Determinism:

The Giotto compiler chooses for a given platform a platform timeline that is value equivalent to the environment timeline defined by the Giotto semantics.

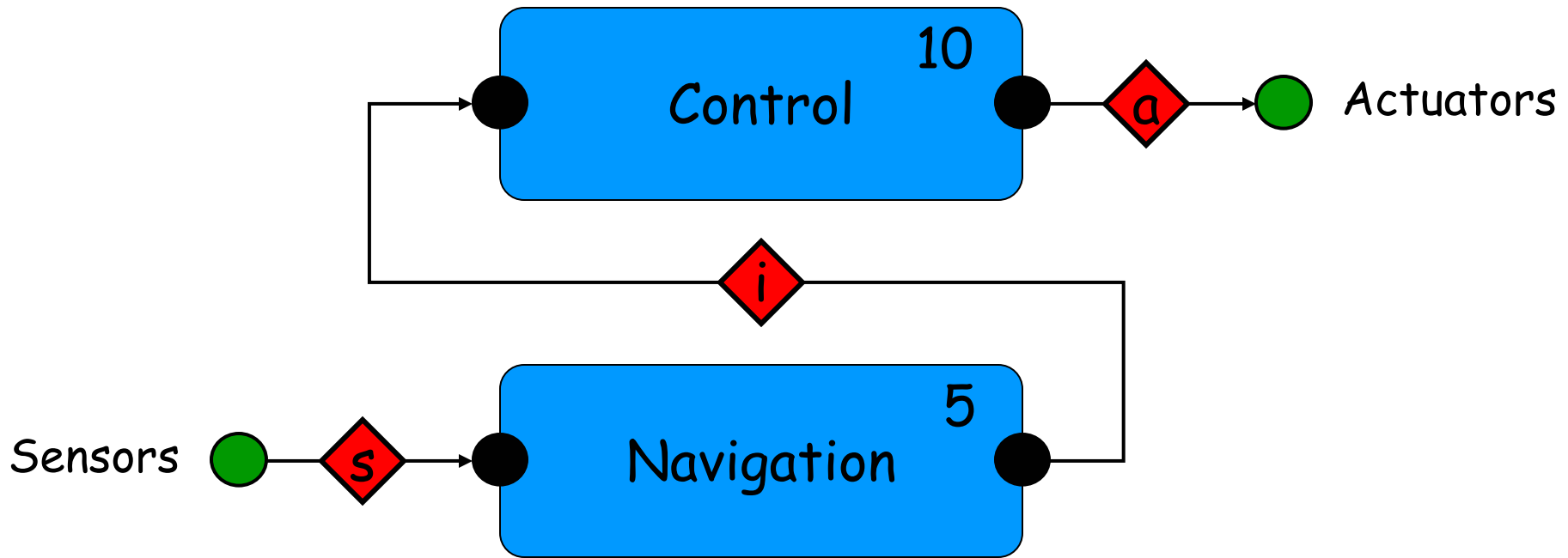


implies

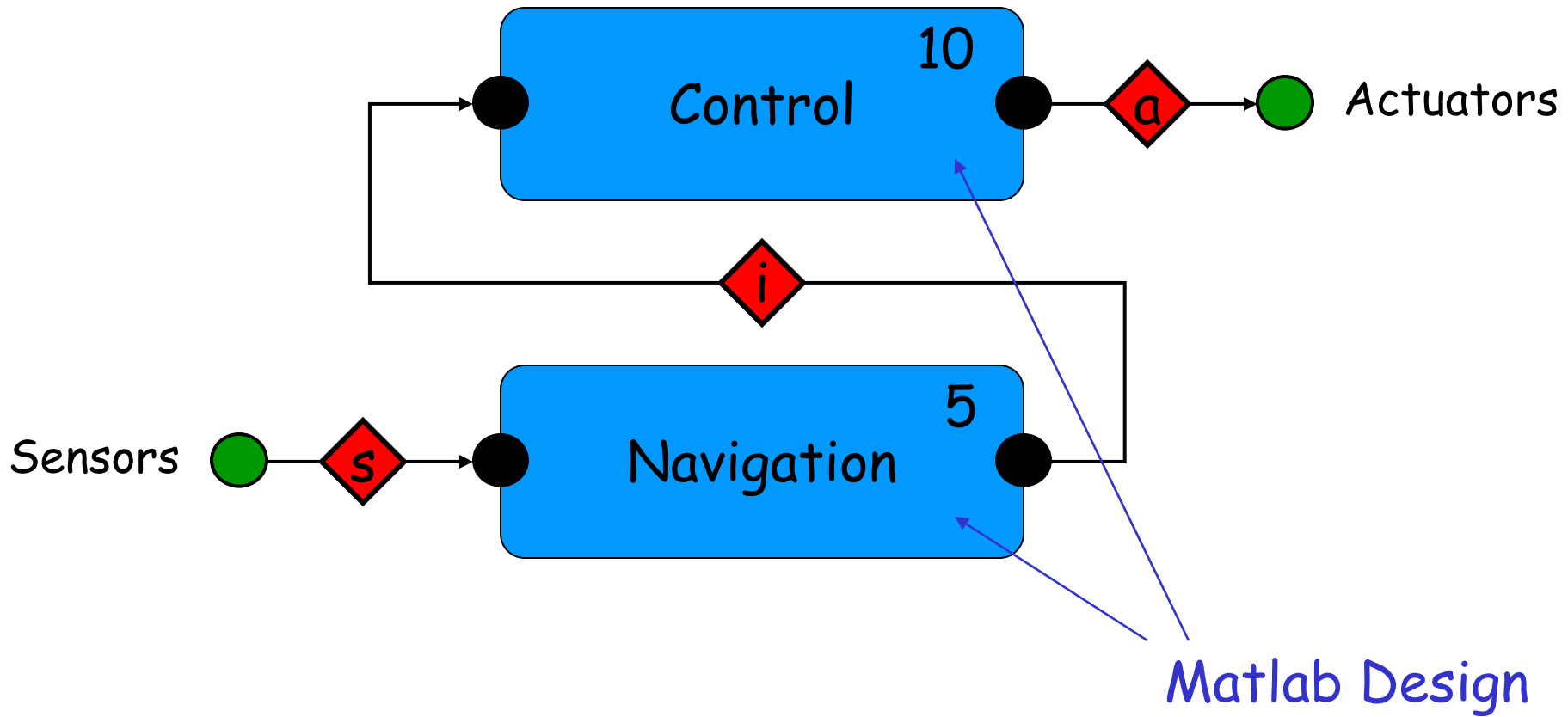
Time Determinism:

For a given time-triggered sequence of sensor readings, the corresponding time-triggered sequence of actuator settings is uniquely determined (i.e., there are no race conditions).

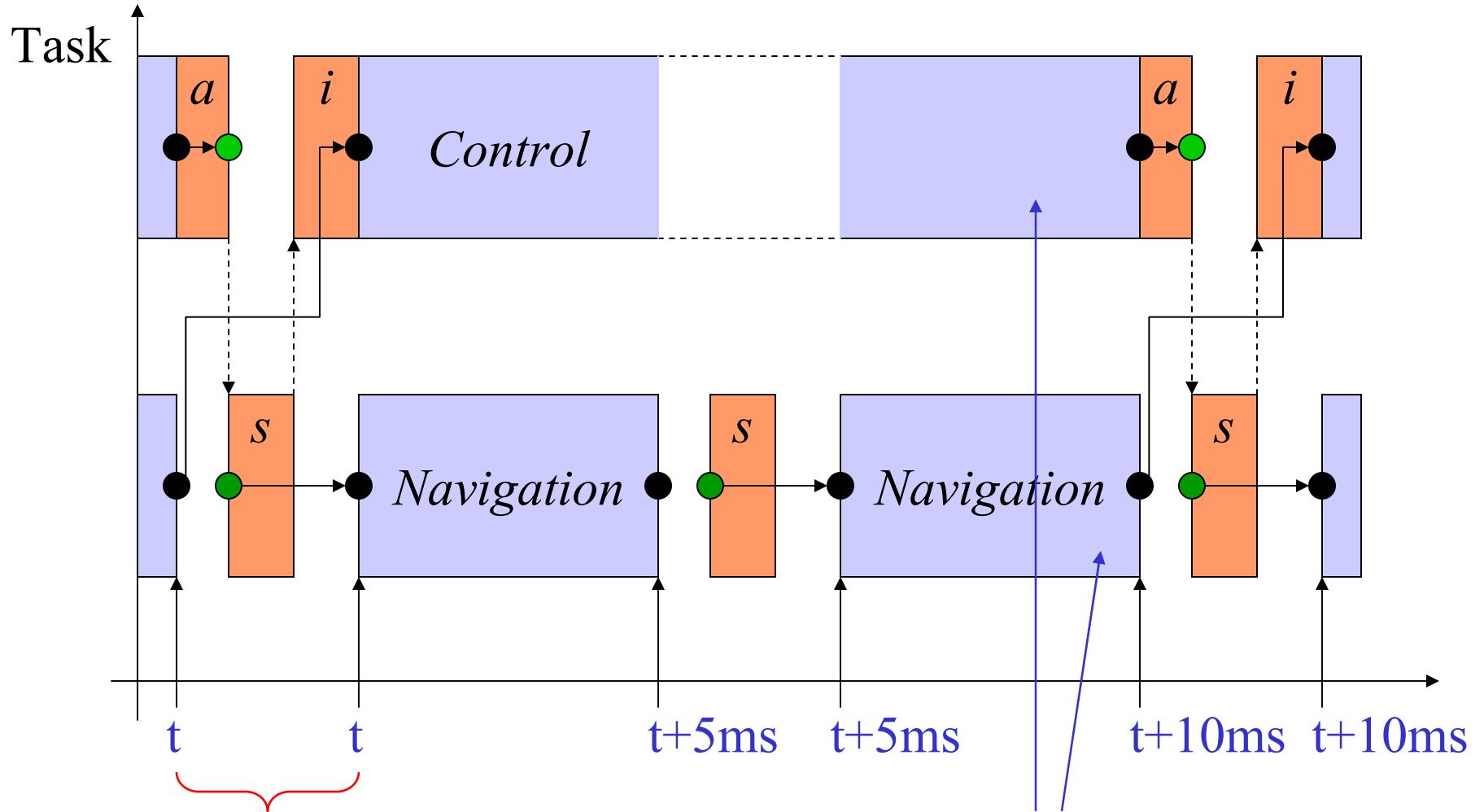
Helicopter Software



Helicopter Software



Helicopter Software: Environment Timeline



Block of synchronous code
(nonpreemptible)

Scheduled tasks
(preemptible)

Helicopter Software: Giotto Syntax (Functionality)

```
sensor gps_type GPS uses c_gps_device ;  
actuator servo_type Servo := c_servo_init  
    uses c_servo_device ;
```

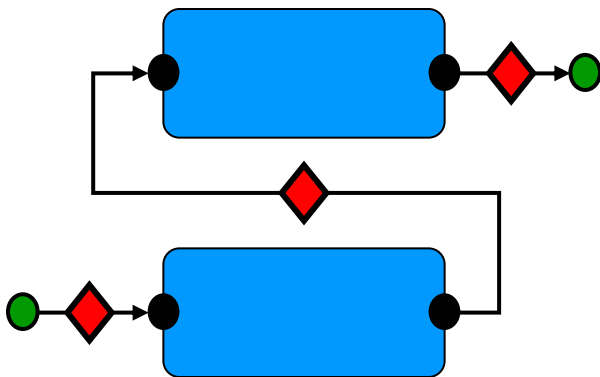
output

```
ctr_type CtrOutput := c_ctr_init ;  
nav_type NavOutput := c_nav_init ;
```

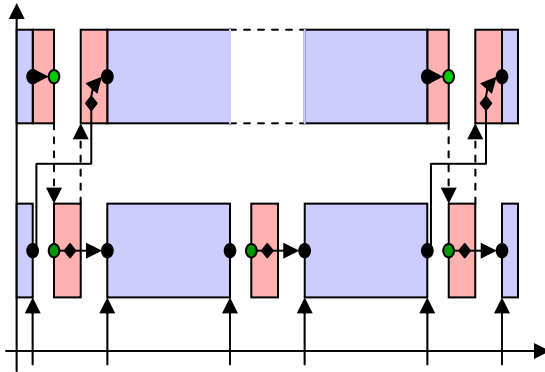
```
driver sensing (GPS) output (gps_type gps)  
{ c_gps_pre_processing ( GPS, gps ) }
```

```
task Navigation (gps_type gps) output (NavOutput)  
{ c_matlab_navigation_code ( gps, NavOutput ) }
```

...



Helicopter Software: Giotto Syntax (Timing)



...

```
mode Flight ( ) period 10ms
```

```
{
```

```
actfreq 1 do Actuator ( actuating ) ;
```

```
taskfreq 1 do Control ( input ) ;
```

```
taskfreq 2 do Navigation ( sensing ) ;
```

```
}
```

...

The Giotto Compiler

Functionality Native Code for tasks and drivers

Timing
Interaction Giotto Program

Hardware Giotto-H hardware specification

- topology (CPUs, nets)
- performance (WCET, latency)

Giotto Compiler



Executables or

"Failure"

either Giotto-H overconstrained,
or compiler not smart enough
(distributed scheduling problem!)

Closing the Gap: Annotated Giotto

Functionality	Native Code	for tasks and drivers
Timing Interaction	Giotto Program	
Hardware	Giotto-H	-topology (CPUs, nets) -performance (WCET, latency)
Map	Giotto-HM	-assign tasks to CPUs -assign connections to nets

Giotto Compiler



Executables

or

"Failure"

either Giotto-HM overconstrained,
or compiler not smart enough
(local scheduling problems)

Closing the Gap: Annotated Giotto

Functionality	Native Code	for tasks and drivers
Timing Interaction	Giotto Program	
Hardware	Giotto-H	-topology (CPUs, nets) -performance (WCET, latency)
Map	Giotto-HM	-assign tasks to CPUs -assign connections to nets
Schedule	Giotto-HMS	-assign tasks to priorities (say) -assign connections to TDMA slots (say)

Giotto Compiler



Executables

or

"Failure"

Giotto-HMS overconstrained

Single-CPU Helicopter: Annotated Giotto

```
[ host Heli address 192.168.0.1 ] // Giotto-H Annotation
```

```
...
```

```
mode Flight ( ) period 10ms
```

```
{
```

```
    actfreq 1 do Actuator ( actuating ) ;
```

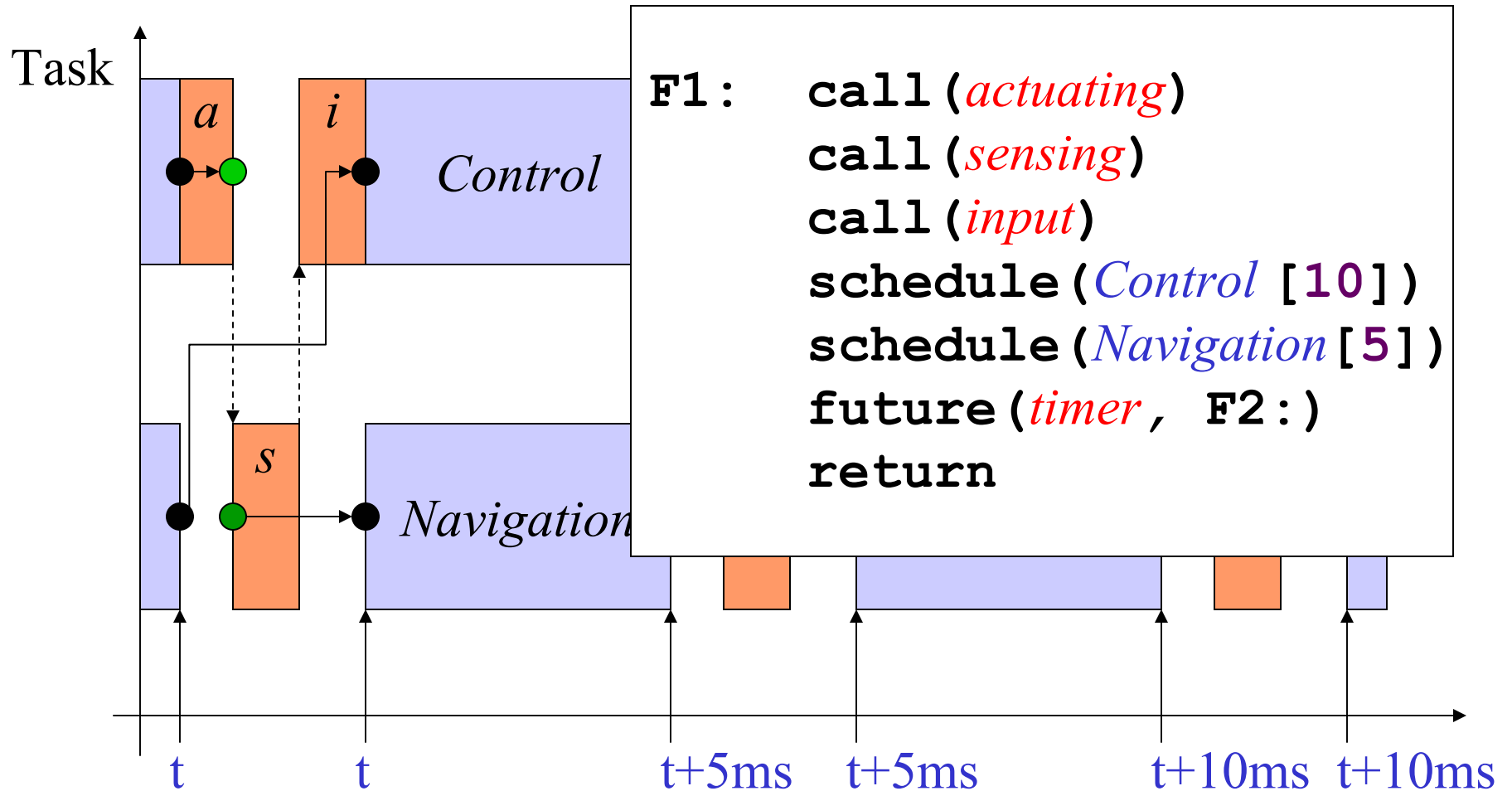
```
    taskfreq 1 do Control ( input ) [ host Heli deadline 10 ] ; // Giotto-MS
```

```
    taskfreq 2 do Navigation ( sensing ) [host Heli deadline 5 ] ;
```

```
}
```

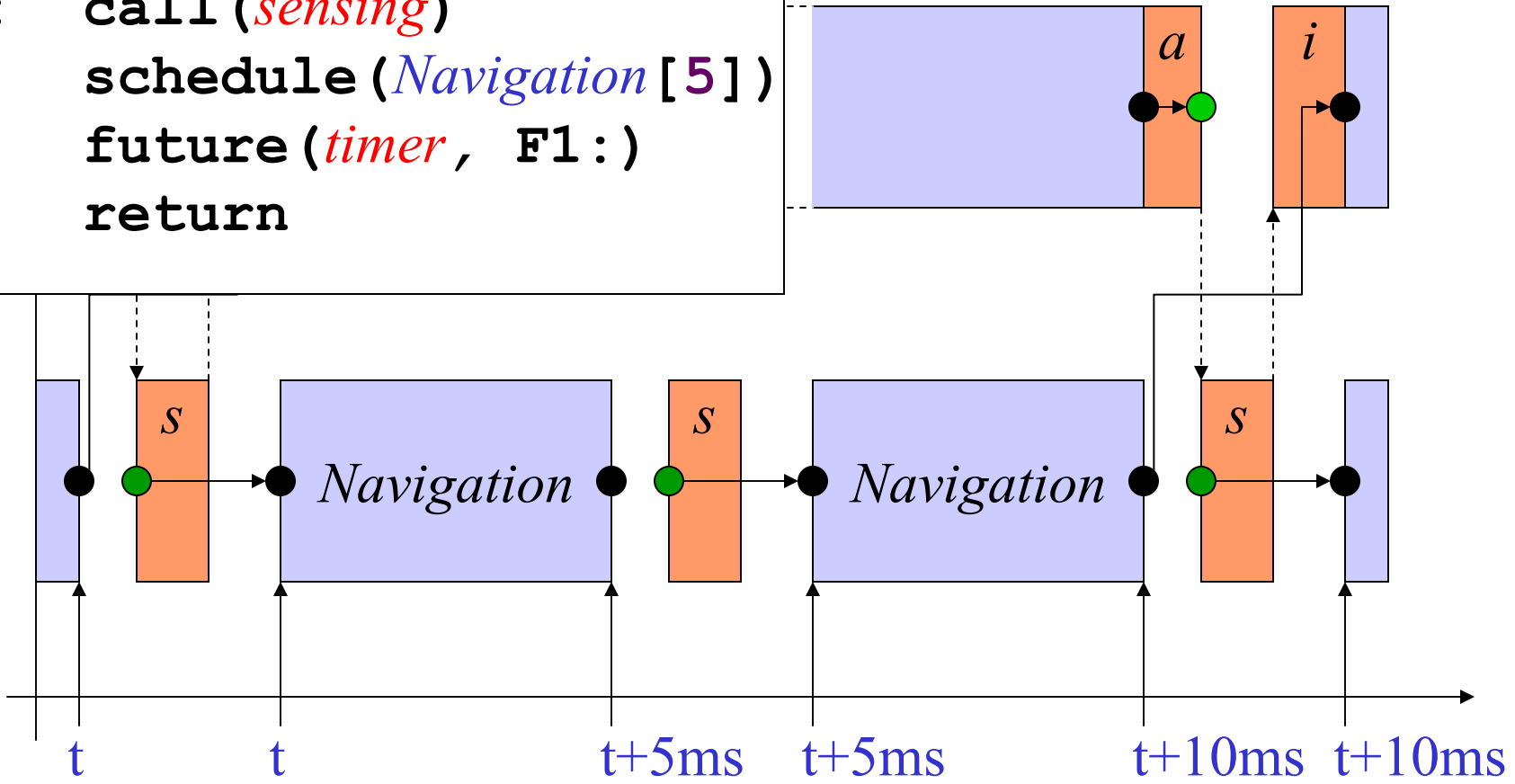
```
...
```


Code Generation



Code Generation

```
F2:  call (sensing)  
     schedule (Navigation [5])  
     future (timer, F1:)  
     return
```

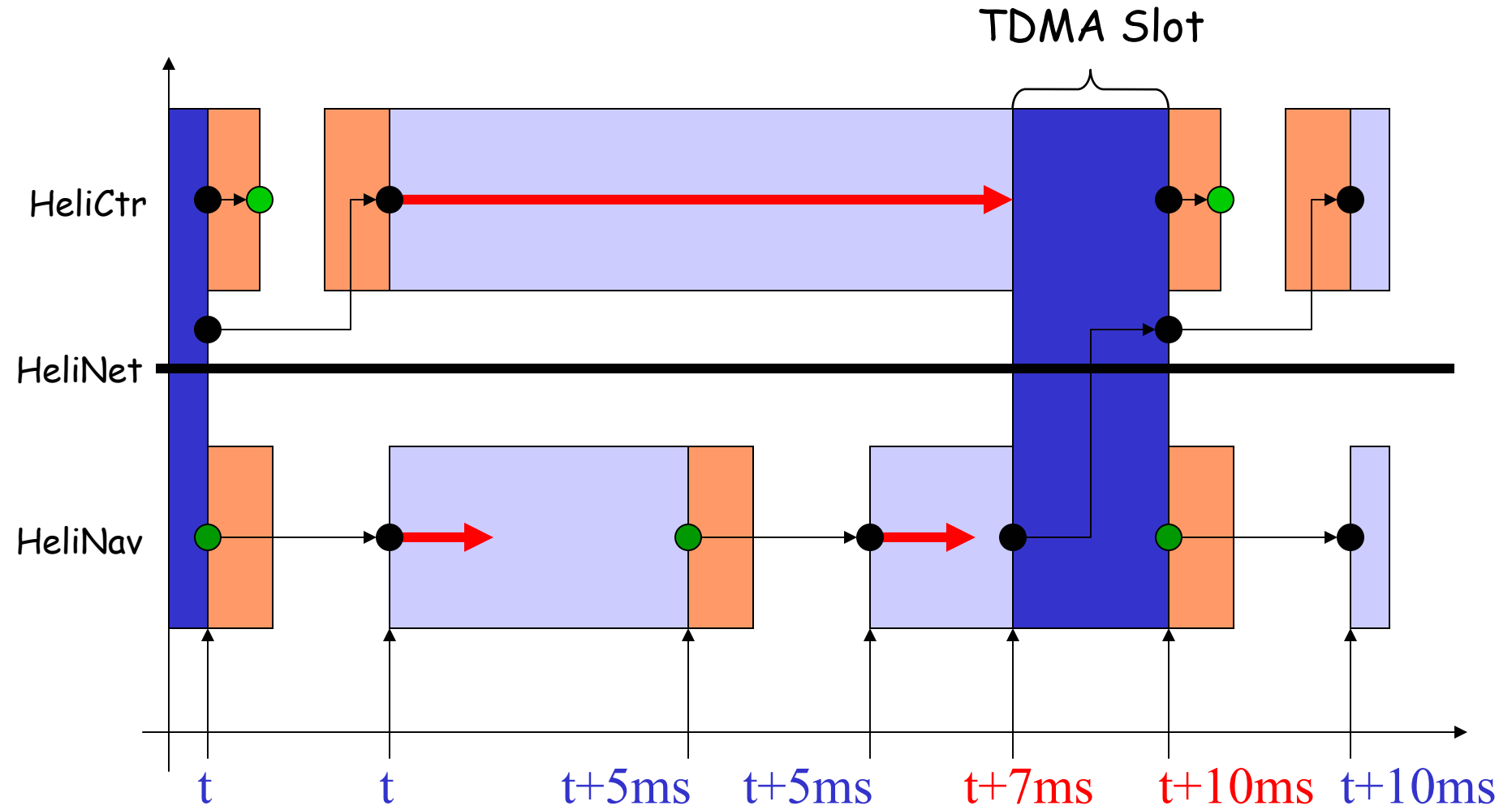


Two-CPU Helicopter: Annotated Giotto (Time-triggered Communication)

```
[ host HeliCtr address 192.168.0.1;
  host HeliNav address 192.168.0.2;
  network HeliNet address 192.168.0.0 connects HeliCtr, HeliNav ]
...
mode Flight ( ) period 10ms
{
  actfreq 1 do Actuator ( actuating ) ;

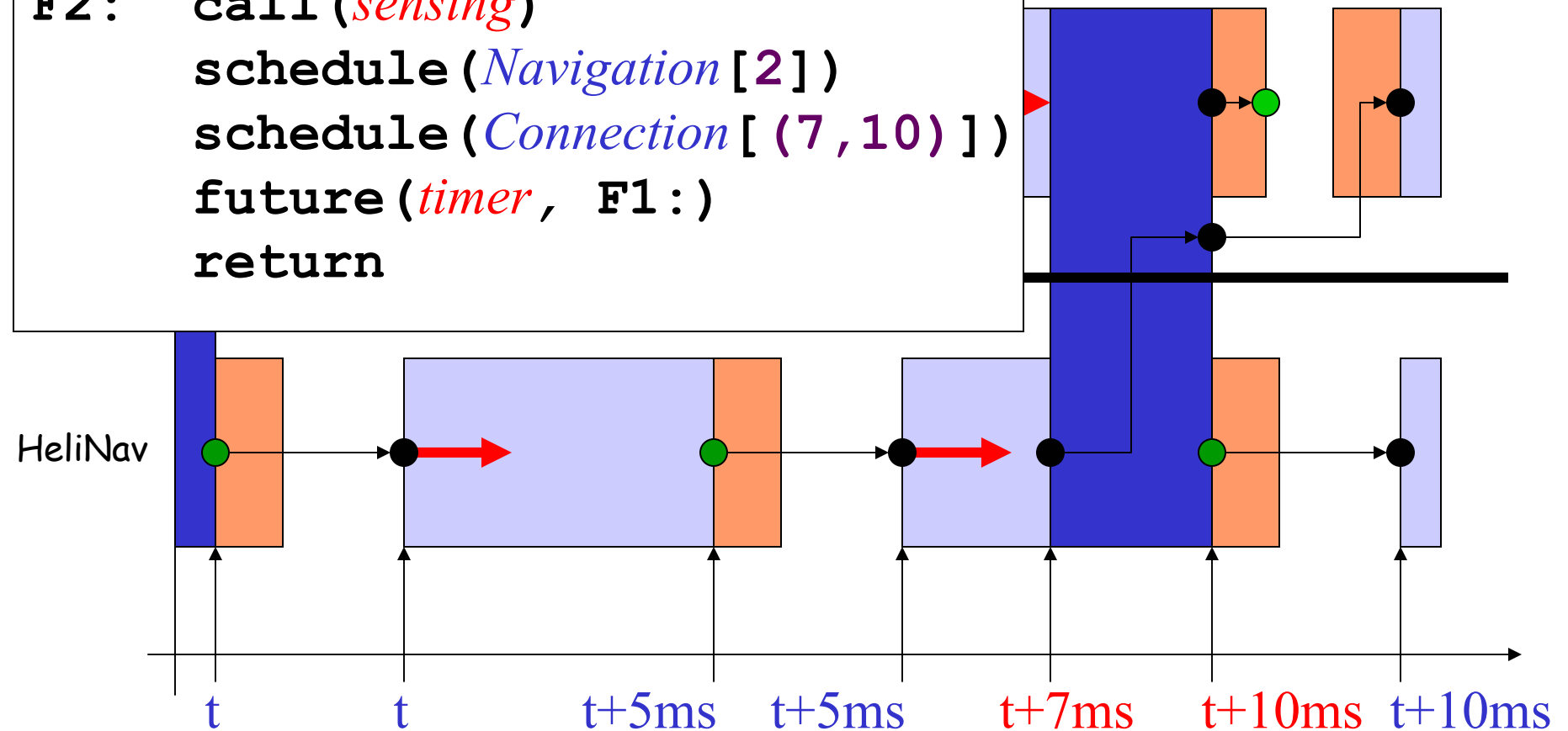
  taskfreq 1 do Control ( input ) [ host HeliCtr deadline 7 ] ;
  taskfreq 2 do Navigation ( sensing ) [host HeliNav deadline 2 ;
  push ( NavOutput ) to ( HeliCtr ) in HeliNet slots (7,10) ] ;
}
...
```

Two-CPU Helicopter: Platform Timeline (Time-triggered Communication)



Code Generation for HeliNav

```
F2:  call(sensing)
      schedule(Navigation[2])
      schedule(Connection[(7,10)])
      future(timer, F1:)
      return
```

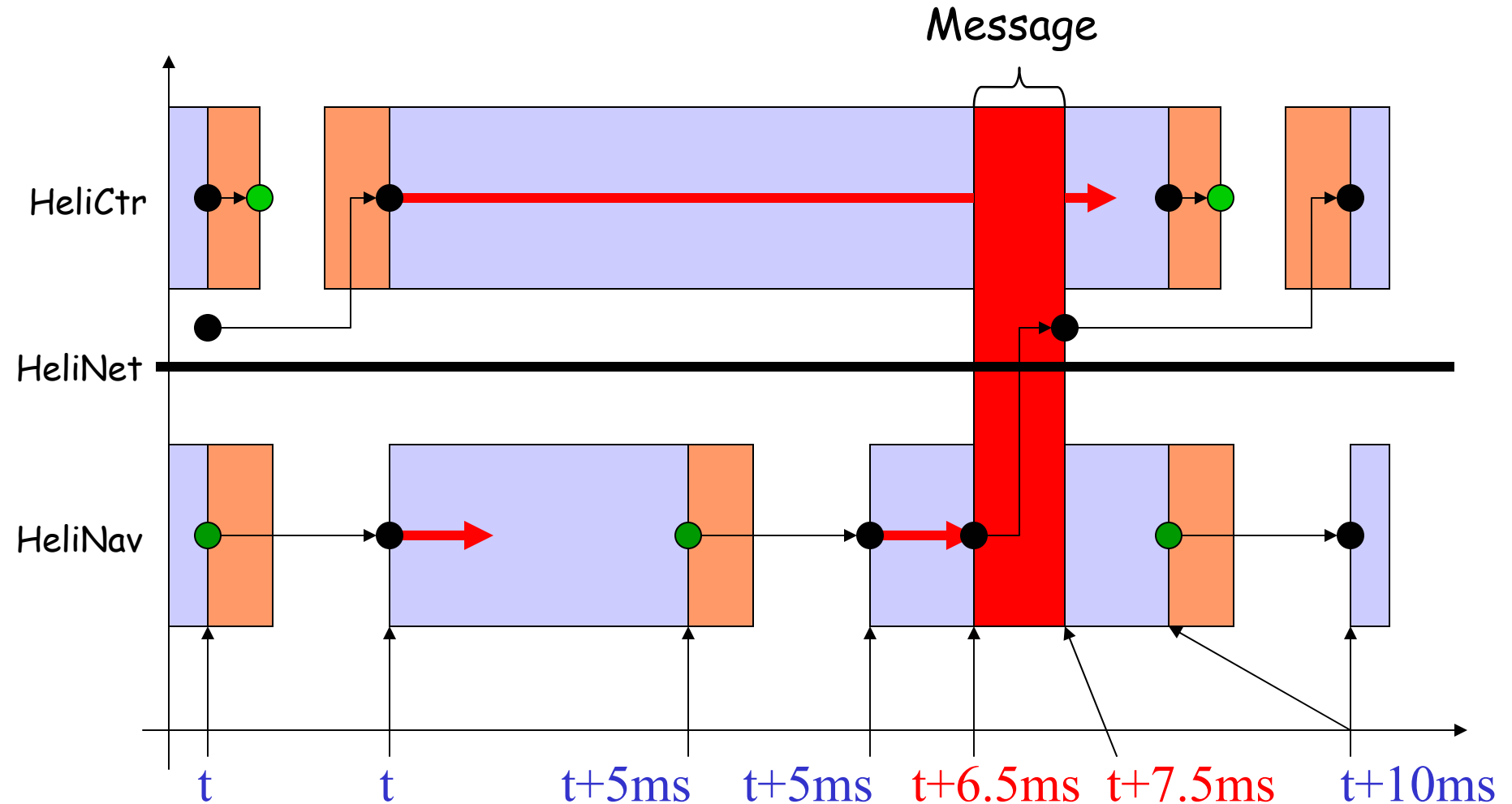


Two-CPU Helicopter: Annotated Giotto (Event-triggered Communication)

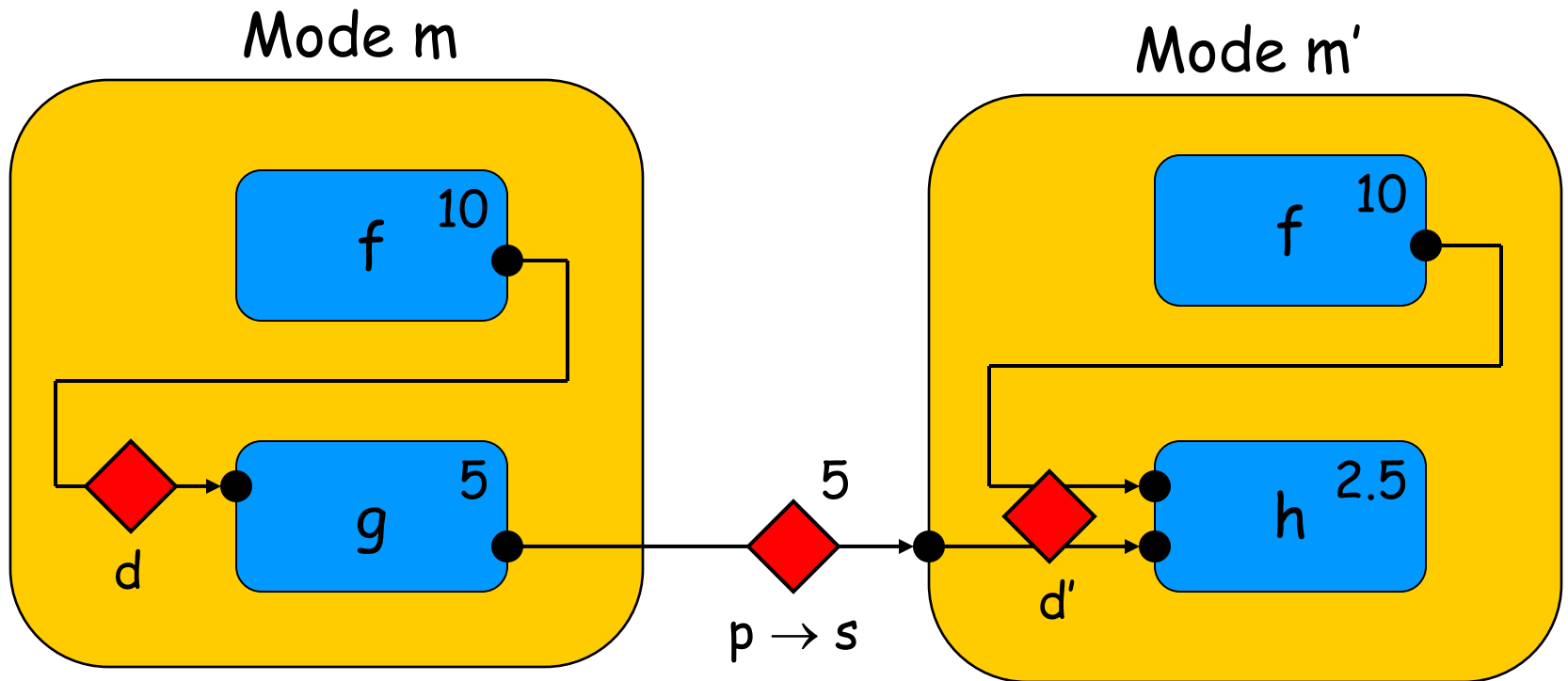
```
[ host HeliCtr address 192.168.0.1;
  host HeliNav address 192.168.0.2;
  network HeliNet address 192.168.0.0 connects HeliCtr, HeliNav ]
...
mode Flight ( ) period 10ms
{
  actfreq 1 do Actuator ( actuating ) ;

  taskfreq 1 do Control ( input ) [ host HeliCtr deadline 10 ] ;
  taskfreq 2 do Navigation ( sensing ) [host HeliNav deadline 2;
  push ( NavOutput ) to ( HeliCtr ) in HeliNet deadline 3 ] ;
}
...
```

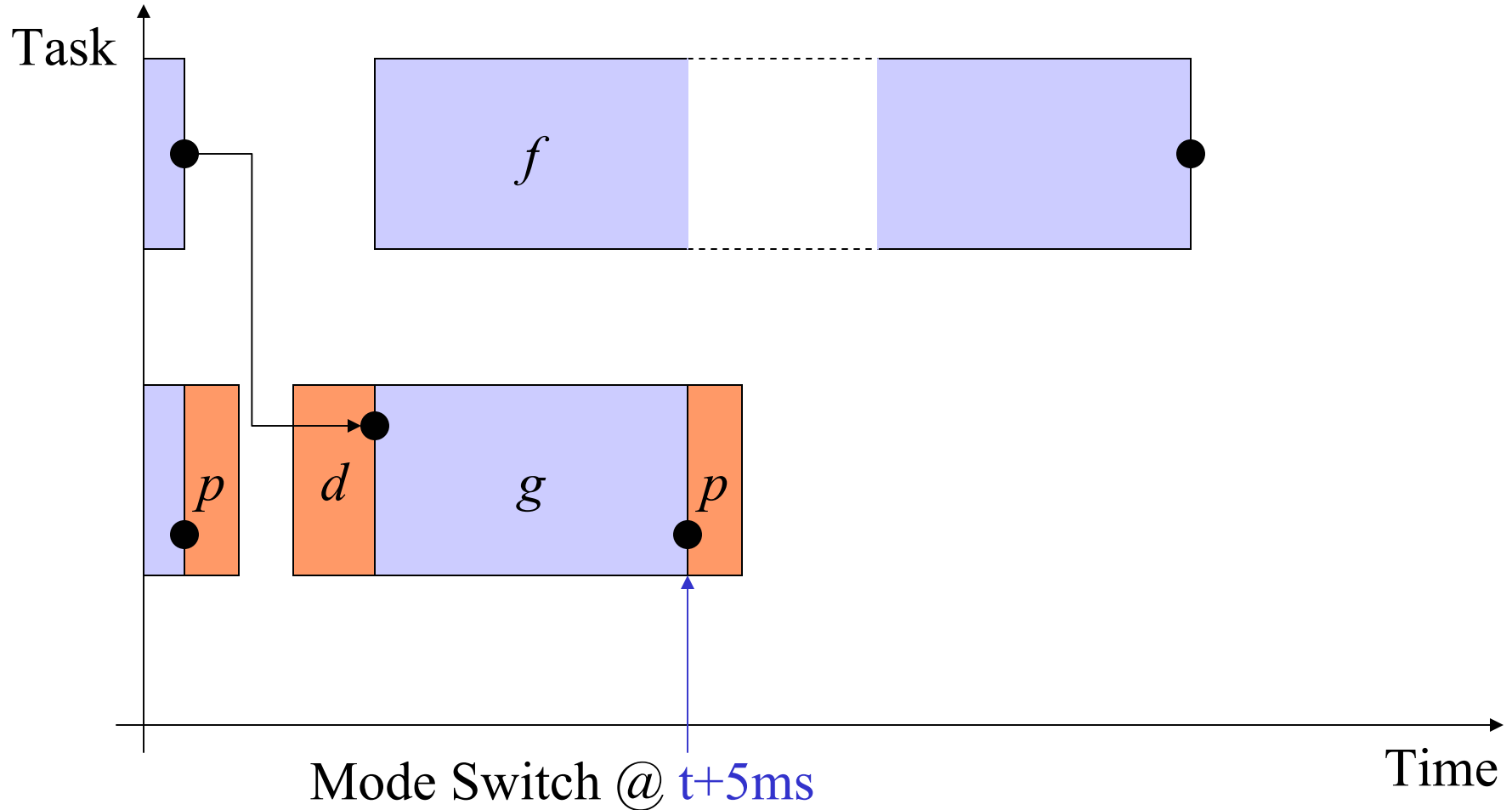
Two-CPU Helicopter: Platform Timeline (Event-triggered Communication)



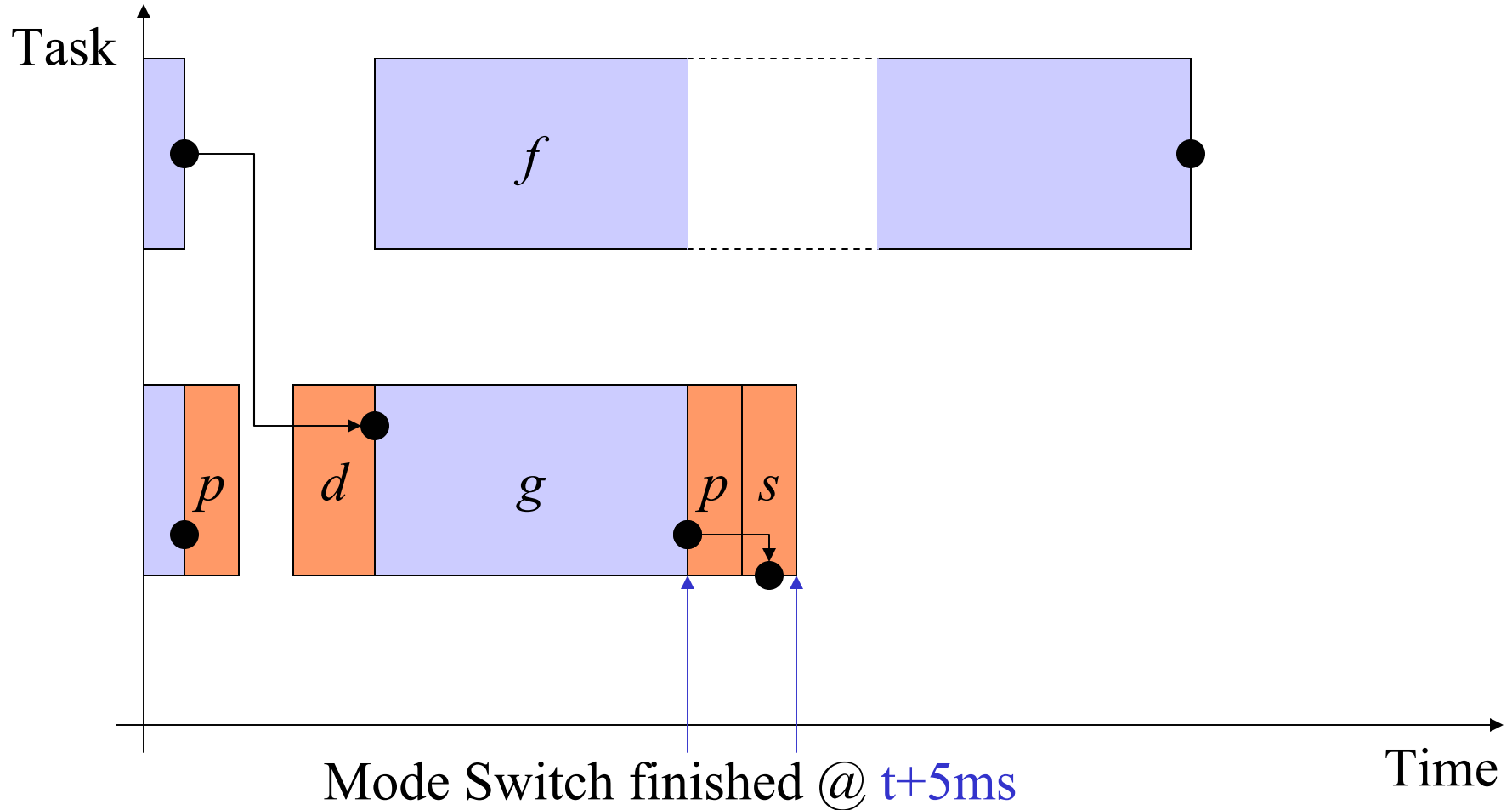
Mode Switch



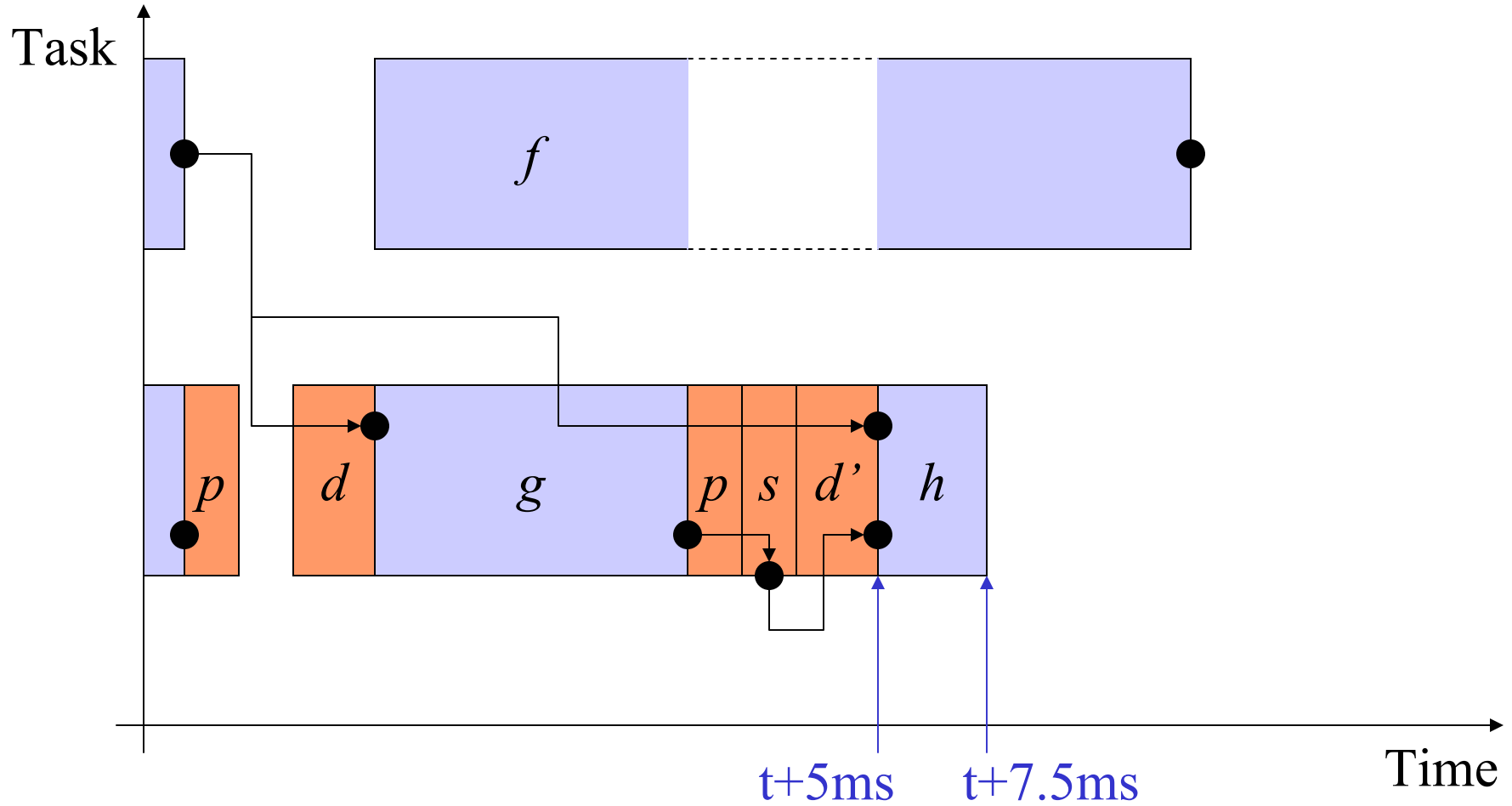
Mode Switch: Environment Timeline



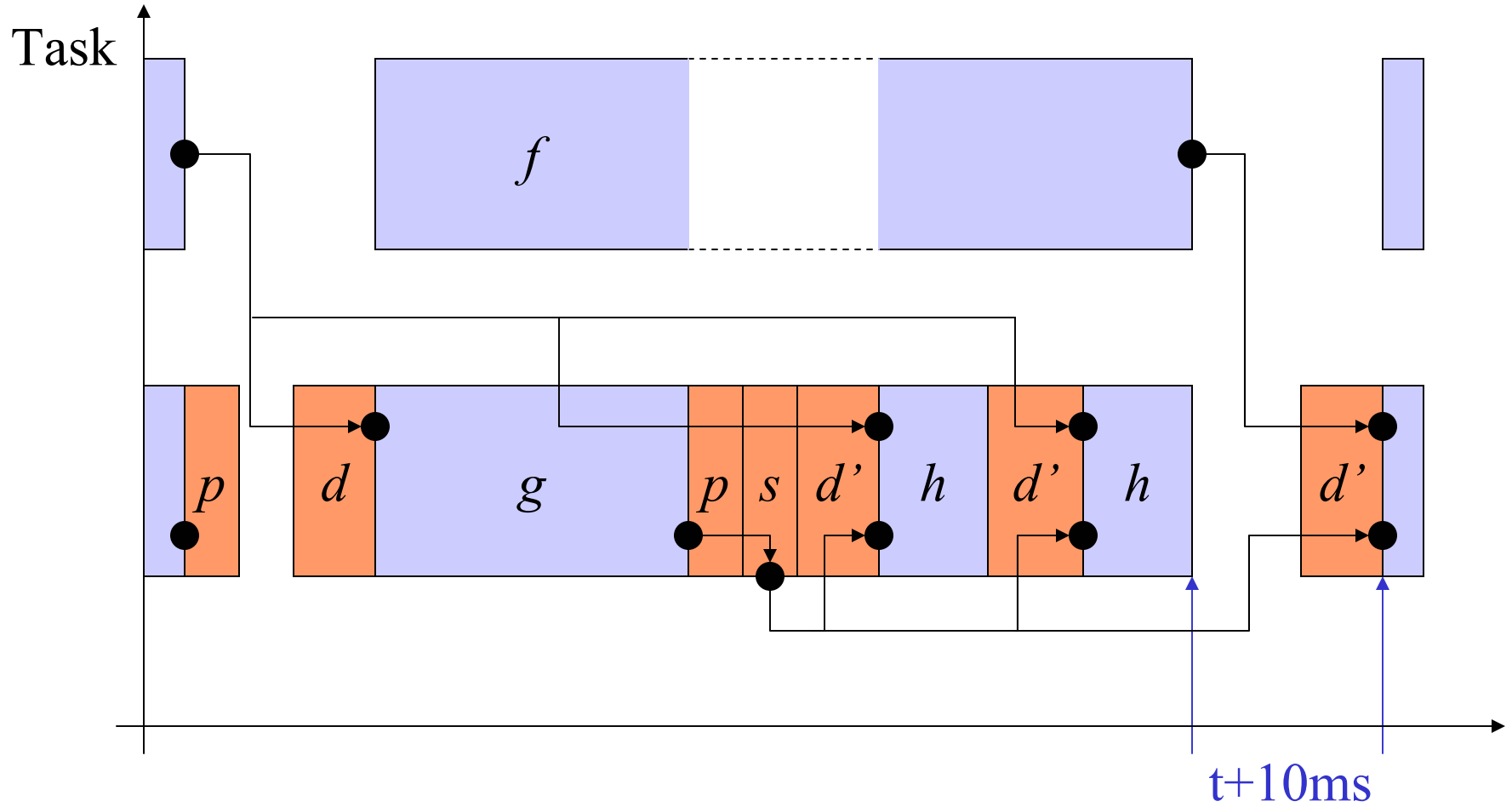
Mode Switch: Environment Timeline



Mode Switch: Environment Timeline



Mode Switch: Environment Timeline



Try it out!

www.eecs.berkeley.edu/~fresco/giotto