



University of Twente

*The Netherlands*

# On Combining Functional Verification and Performance Evaluation using CADP

Hubert Garavel

VASY  
INRIA Rhône-Alpes  
France

Holger Hermanns

Faculty of Computer Science  
University of Twente  
The Netherlands



# Overview

- ▶ First the case study
- Then the success story
- And then a look behind the scene  
(in case you want to know why this works, ... and how)
  - The challenge of specification
  - The challenge of timing
  - The challenge of driving CADP to estimate performance



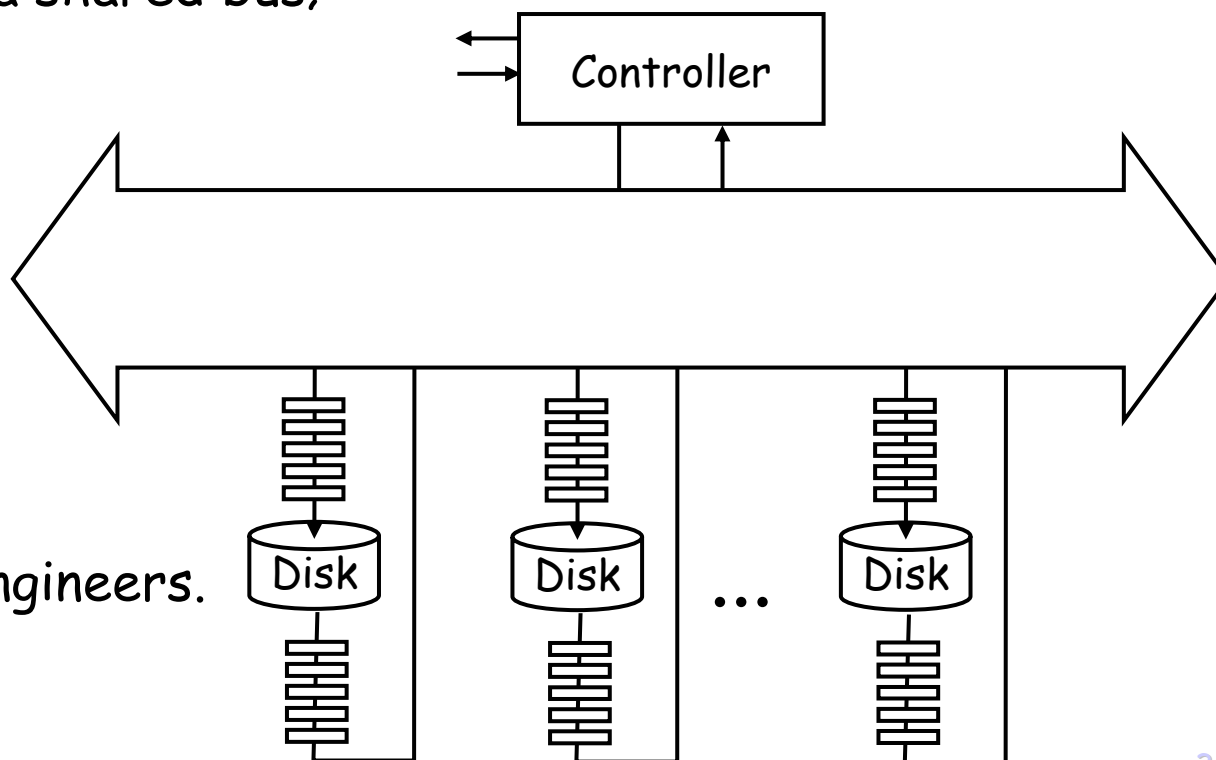
# Case Study

## SCSI-2: Small Computer System Interface

- brought to our attention by Bull SA, Italy;
- designed to provide fast access to multiple storage devices, via a shared bus;

- up to 7 devices (disks, in the sequel) and 1 controller;

- 'starvation problem' discovered by Bull engineers.



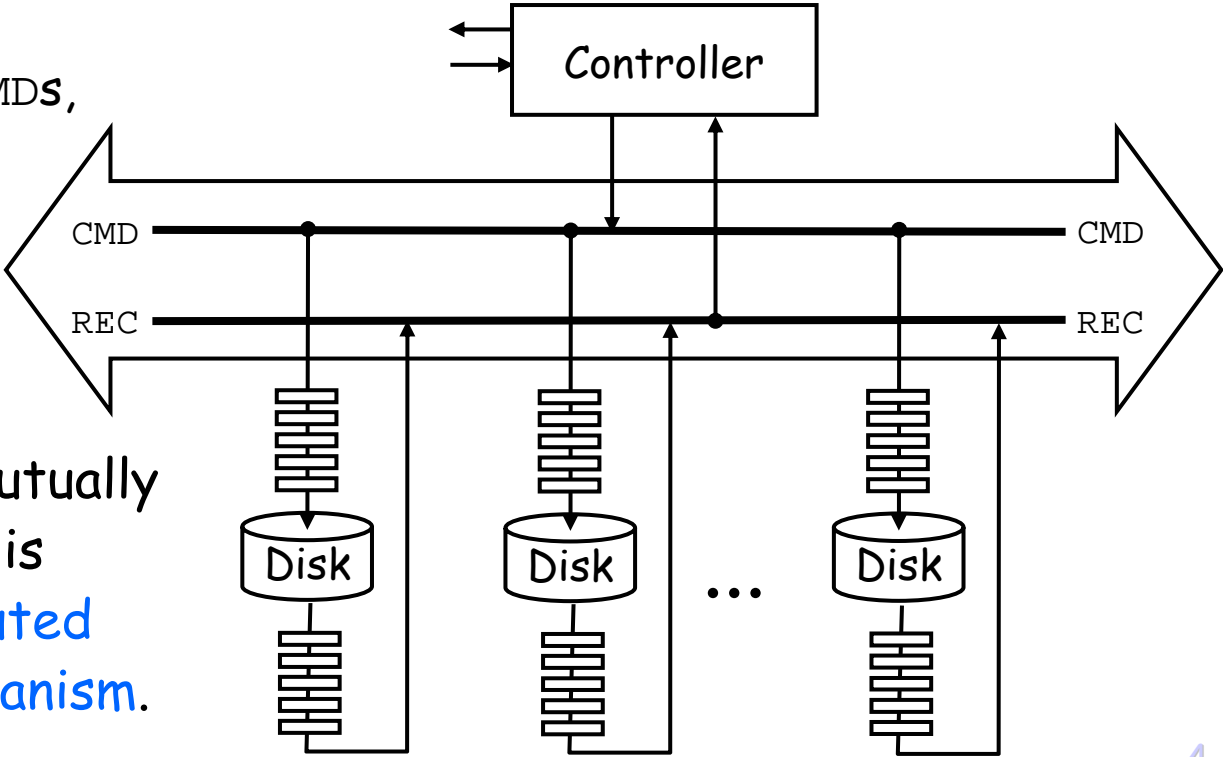


# SCSI-2 Bus Architecture

- Controller
  - handles (OS level) requests
  - passes read/write requests to the designated disk (CMD)
  - passes results back to the OS (REC)
  - provides flow control to prevent disk flooding,

- Disks
  - process incoming CMDs,
  - send back results by REC,

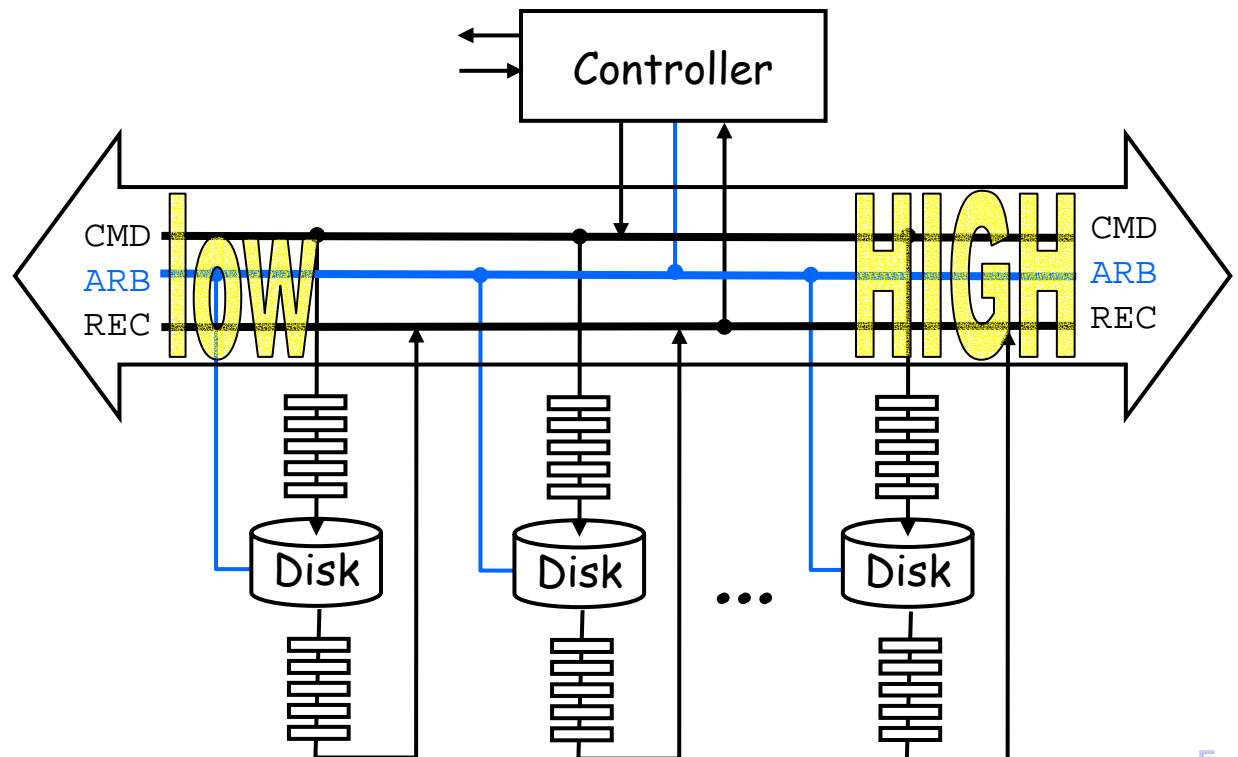
- Disks and Controller share the bus, but mutually exclusive bus access is granted by a **distributed bus arbitration mechanism.**





# SCSI 2 Bus Arbitration

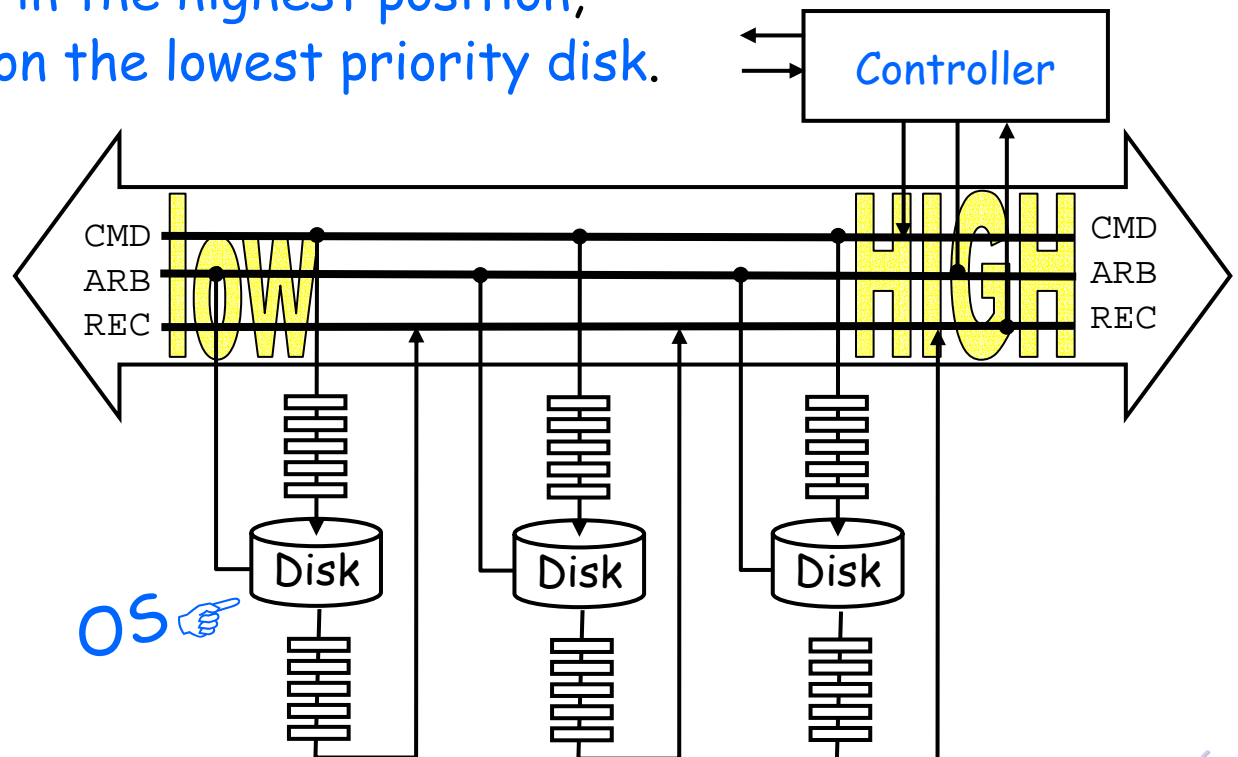
- Prioritized, based on static IDs on bus;
- Any bus access is preceded by a scan ensuring that no higher priority device requires the bus;
- Realized through a mesh of dedicated wires.





# Starvation and how it was fixed

- The Bull engineers observed 'starvation' of applications for some specific configurations, dependent on the position of the controller on the bus.
- They observed that this problem was absent if the controller was in the highest position, and the OS was put on the lowest priority disk.





# So, what's the success story then?

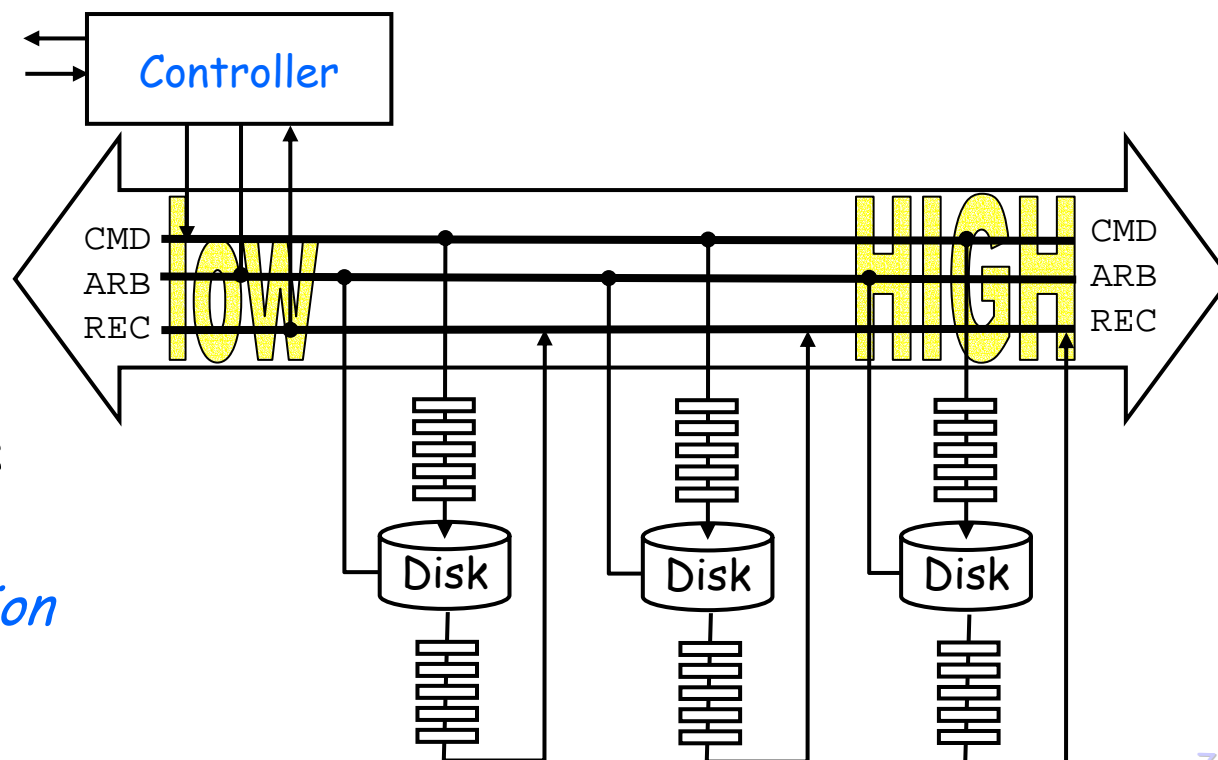
- Verification (model checking) with CADP revealed the starvation problem, and its cause. [Garavel/Mateescu]

(a livelock preventing lower priority disks to get the bus granted)

- Our performance studies (on top of the verified model) showed that the 'problem fix' is (mildly speaking) suboptimal.

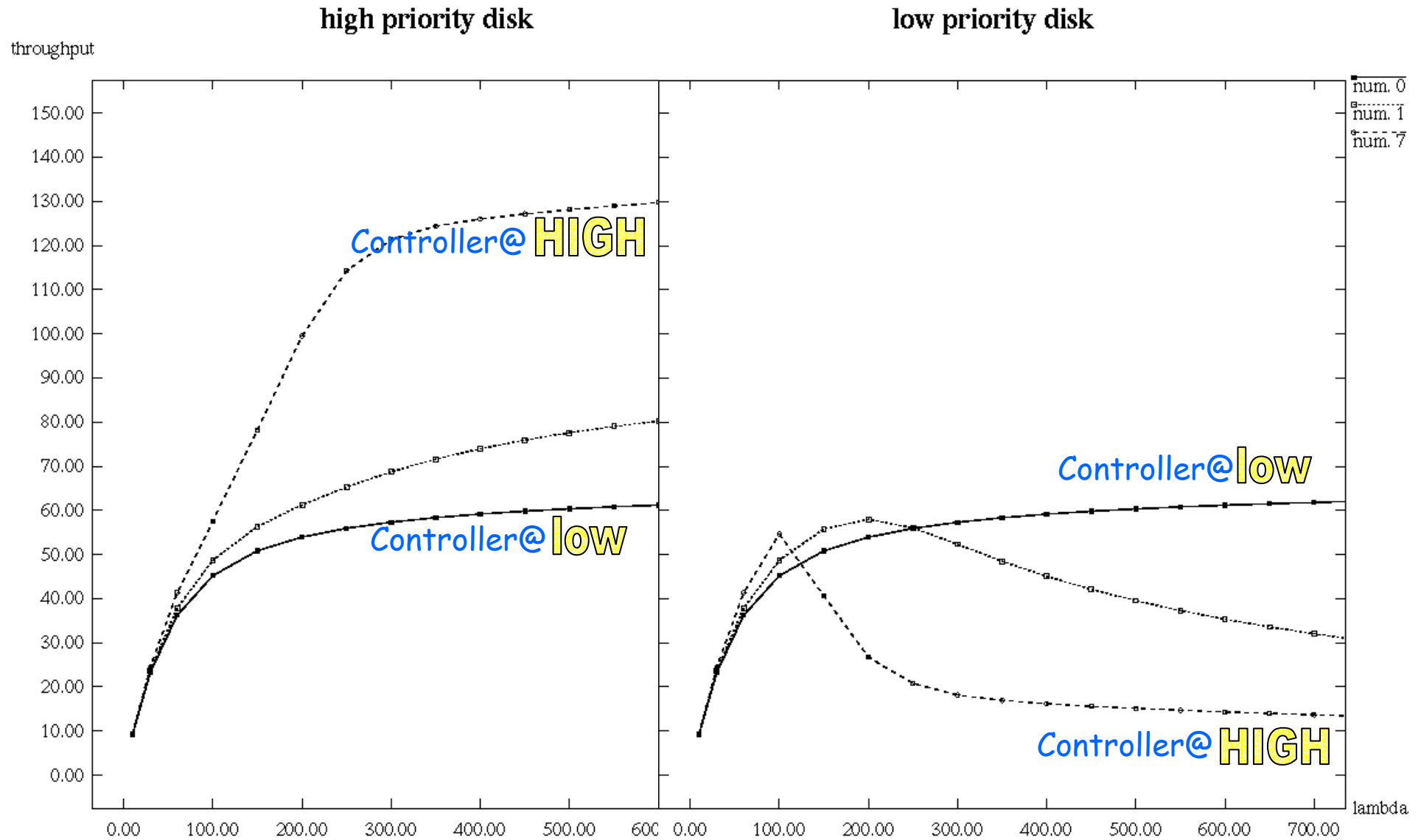
- Another solution avoids livelocks, and is superior in the spectrum of scenarios considered:

*put the controller in lowest-priority position*





# Influence of the controller position







# Overview

- First the case study

- Then the success story

- ▶ ● And then a look behind the scene  
(in case you want to know why this works, ... and how)

- The challenge of specification

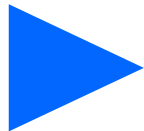
- The challenge of timing

- The challenge of driving CADP to estimate performance



# Overview

- First the case study
- Then the success story
- And then a look behind the scene  
(in case you want to know why this works, ... and how)



- The challenge of specification
- The challenge of timing
- The challenge of driving CADP to estimate performance

# The challenge of specification

- To capture the bus arbitration mechanism (distributed, virtually synchronous) in the abstract specification is nontrivial.
- We use a feature of Lotos, multiway rendezvous with value negotiation.

```
(
DISK [ARB, CMD, REC, MU] (0, 0, false)
|[ARB]|
DISK [ARB, CMD, REC, MU] (1, 0, false)
|[ARB]|
...
|[ARB]|
DISK [ARB, CMD, REC, MU] (6, 0, false)
)
|[ARB, CMD, REC]|
CONTROLLER [ARB, CMD, REC, LAMBDA]
```

an eight-tuple of booleans

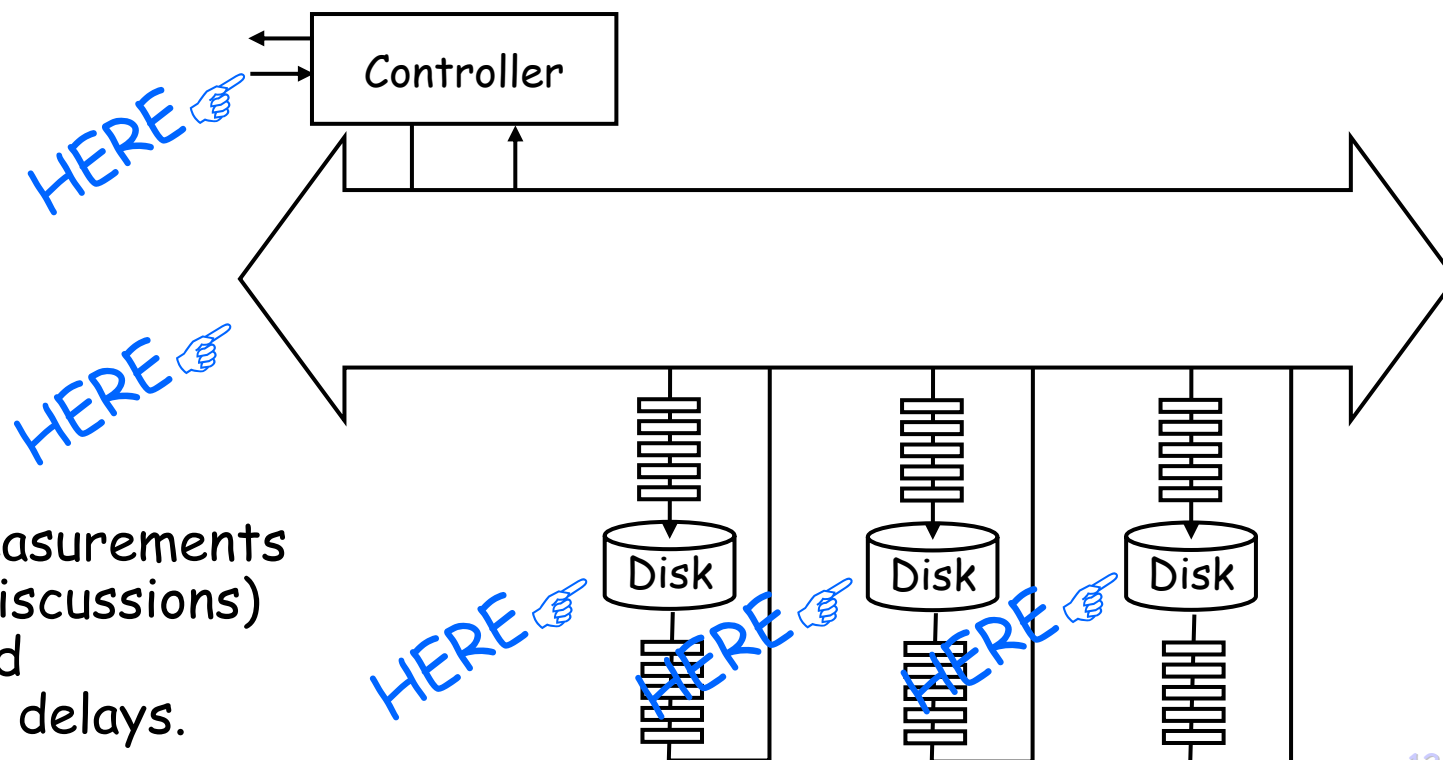
```
process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, READY:BO
  CMD !N;
  DISK [ARB, CMD, REC, MU] (N, L+1, READY)
  []
  ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
  DISK [ARB, CMD, REC, MU] (N, L, READY)
  []
  [not (READY) and (L > 0)] ->
  MU !N;
  DISK [ARB, CMD, REC, MU] (N, L-1, true)
  []
  ARB ?W:WIRE [READY and C_LOSS (W, N)];
  DISK [ARB, CMD, REC, MU] (N, L, READY)
  []
  ARB ?W:WIRE [READY and C_WIN (W, N)];
  REC !N;
  DISK [ARB, CMD, REC, MU] (N, L, false)
```

'pattern matcher'  
'pattern matcher'  
'pattern matcher'



# The challenge of timing

- To estimate system performance, some measurements (or, at least, educated guesses) are needed
  - where relevant delays occur;
  - what characteristics they have.



- Based on measurements (as well as discussions) we identified the relevant delays.

# The challenge of timing (2)

- How are the timing characteristics incorporated into the specification?
- For the moment: just insert **at the right place** in the specification.

Proof obligation: either

→ repeat model checking,

or

→ show that modified specification is (branching) equivalent to original one, if Markov delays are considered as internal ( $\tau$ ) steps.

```

process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, READY:BO
  CMD !N;
    DISK [ARB, CMD, REC, MU] (N, L+1, READY)
  []
  ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
    DISK [ARB, CMD, REC, MU] (N, L, READY)
  []
  [not (READY) and (L > 0)] ->
    MU !N; (* Markov delay inserted here *)
    DISK [ARB, CMD, REC, MU] (N, L-1, true)
  []
  ARB ?W:WIRE [READY and C_LOSS (W, N)];
    DISK [ARB, CMD, REC, MU] (N, L, READY)
  []
  ARB ?W:WIRE [READY and C_WIN (W, N)];
    REC !N;
    DISK [ARB, CMD, REC, MU] (N, L, false)
endproc
  
```

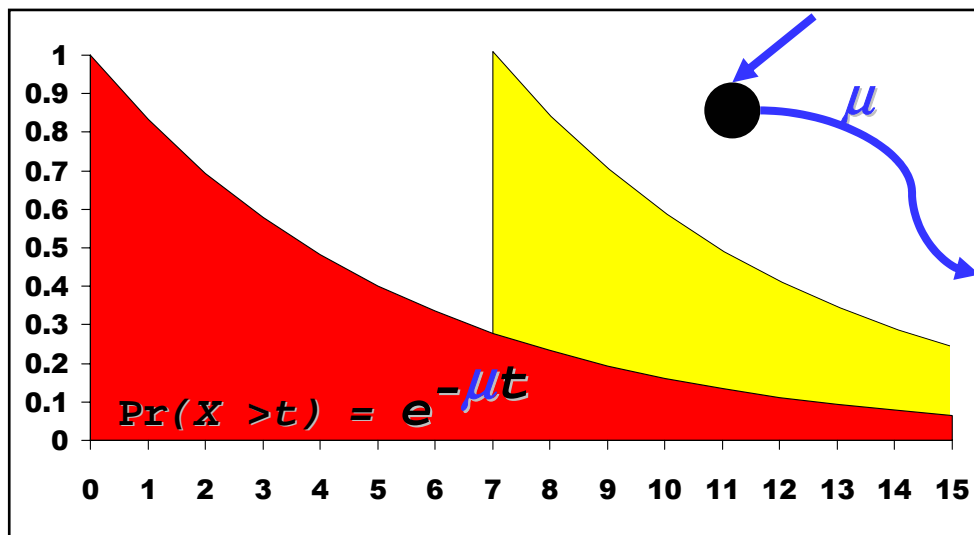
## Intermezzo: Markov delays

- ▶ A far too rapid introduction into Markov models
  - How this combines with labelled transition systems
  - How this is supported in CADP



## Continuous-time Markov chains (CTMCs)

- (finite state) automata,
- all times are *exponentially distributed*,

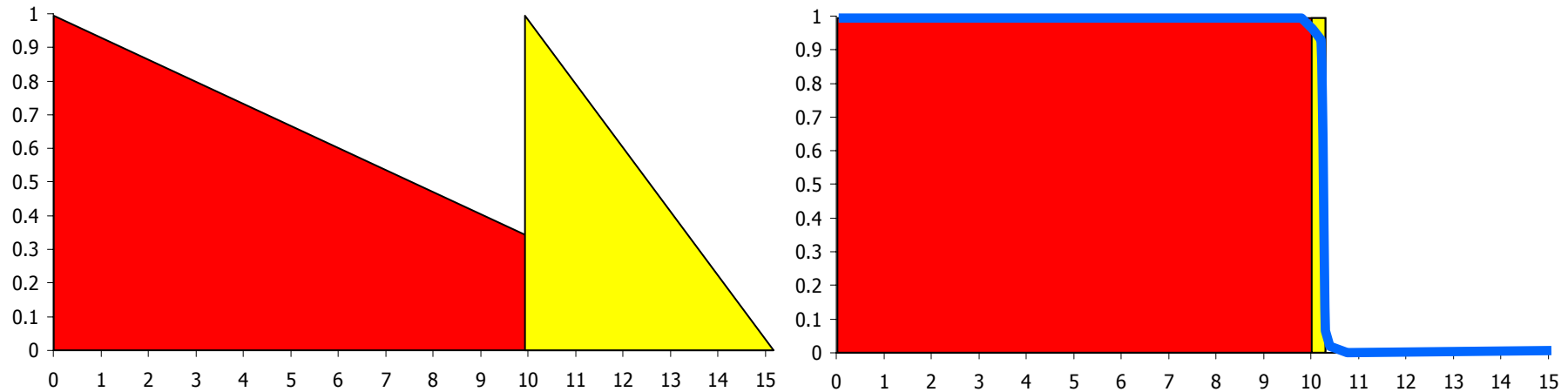


- sojourn time in states are *memory-less*,

- very well investigated class of stochastic processes,
- widely used in practice,
- best guess, if only mean values are known,
- *efficient* and *numerically stable* algorithms for *stationary* and *transient* analysis are available.



Well, this sounds quite restrictive!



*Absence of memory is rare.*

But:

Superpositions of exponential phases allows one to approximate *arbitrary distributions* within the CTMC framework.





## Intermezzo: Markov delays

- A far too rapid introduction into Markov models



- ▶ ● How this combines with labelled transition systems

- How this is supported in CADP



# Interactive Markov chains

## Model level

- An orthogonal extension
  - of labelled transition systems
  - of CTMCs
- two types of transitions
  - 
  - in the state space
- equipped with property-preserving minimisation algorithms

## Syntax level

- A super-algebra
  - of standard process algebra
  - of CTMCs
- two types of 'actions' (gates)
  - actions
  - Markov delaysin the specification
- equipped with the necessary compositional theory



## Intermezzo: Markov delays

- A far too rapid introduction into Markov models
- How this combines with labelled transition systems

▶ How this is supported in CADP

# Interactive Markov Chains in CADP

## Model level

- two types of transitions
  - $\xrightarrow{\text{CMD}}$
  - $\xrightarrow{\text{rate } 24}$
 in the state space representation
- equipped with efficient property-preserving minimisation algorithms: **bcg\_min**
- linked to standard Markov chain solvers

## Syntax level

- pragmatic
- user defined (and user maintained) separation of 'gates' into two types
  - gates
  - Markov delays

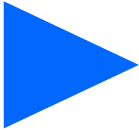
```

process DISK (ARB, CMD, REC, MU) (N:NUM, L:NAT, READY:BOO
  CMD !N;
    DISK [ARB, CMD, REC, MU] (N, L+1, READY)
  []
  ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
    DISK [ARB, CMD, REC, MU] (N, L, READY)
  []
  [not (READY) and (L > 0)] ->
    MU !N; (* Markov delay inserted here *)
    DISK [ARB, CMD, REC, MU] (N, L-1, true)
  []
  
```



# Overview

- First the case study
- Then the success story
- And then a look behind the scene  
(in case you want to know why this works, ... and how)
  - The challenge of specification
  - The challenge of timing
  - The challenge of driving CADP to estimate performance





## The challenge of timing (3)

- How are the timing characteristics incorporated into the specification?
- For the moment: just insert at the right place in the specification.

Proof obligation: either

- repeat model checking, or
- show that modified specification is (branching) equivalent to original one, if Markov delays are considered as internal ( $\tau$ ) steps.

● **Better:**  
Use a compositional, constraint-oriented style (call it aspect-orientation if you like):

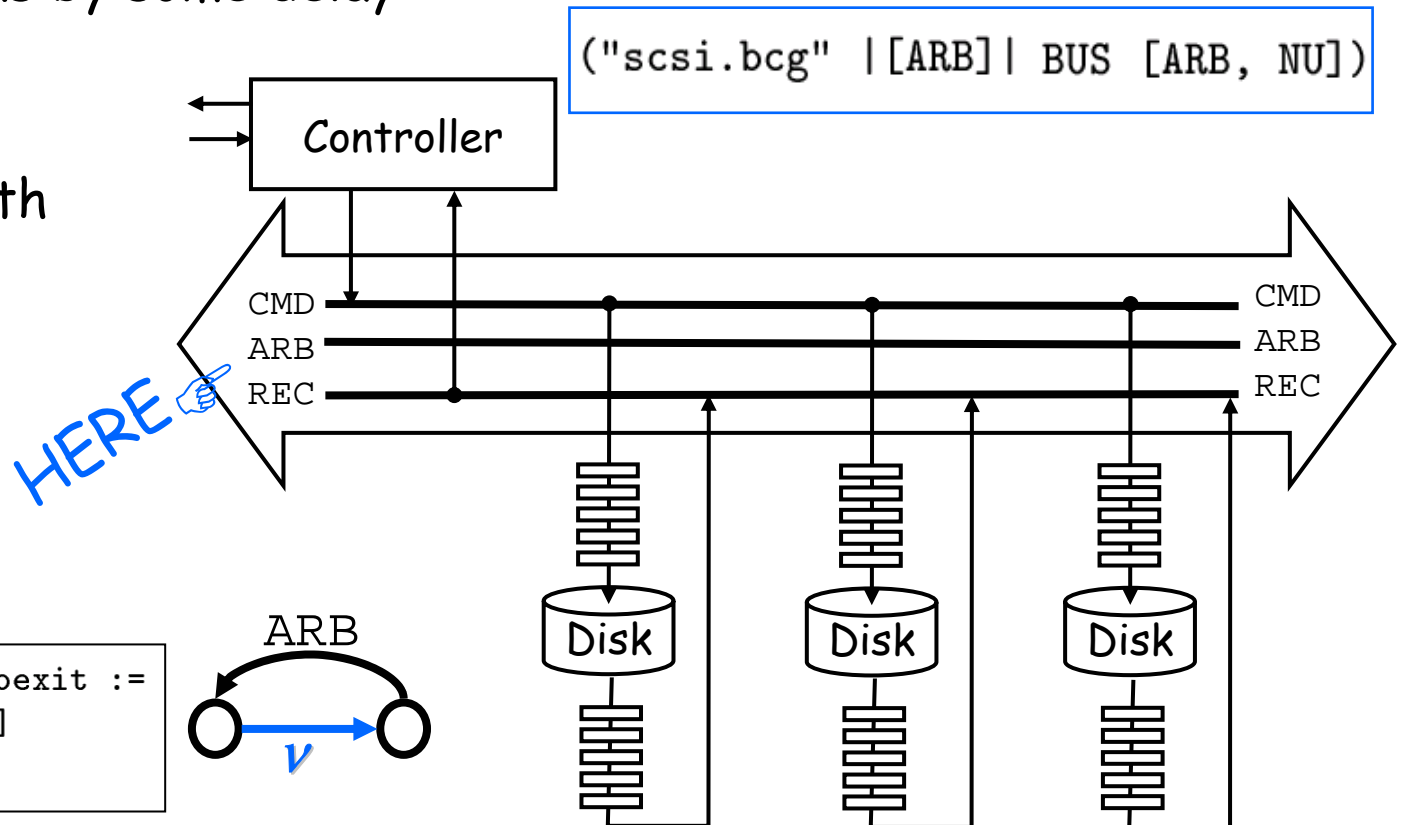
- identify specific actions that
  - are to be delayed
  - initialize a delay
  - may interrupt a delay
- Use composition to insert delays
- No proof obligation



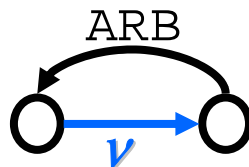
# Time constraints via composition

- Let's incorporate a *bus delay*, by separating any two consecutive bus arbitrations by some delay

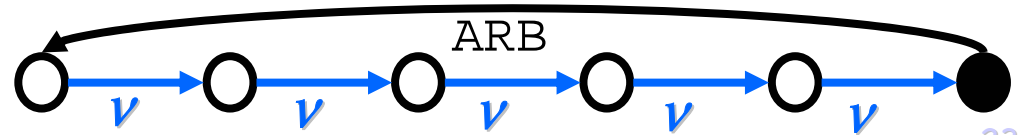
- We do this via composition with an appropriate 'constraint'



```
process BUS [ARB, NU]:noexit :=
  ARB; NU; BUS [ARB, NU]
endproc
```



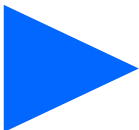
```
process BUS_5 [ARB, NU]:noexit :=
  ARB; NU; NU; NU; NU; BUS_5 [ARB, NU]
endproc
```





# Overview

- First the case study
- Then the success story
- And then a look behind the scene  
(in case you want to know why this works, ... and how)
  - The challenge of specification
  - The challenge of timing
  - The challenge of driving CADP to estimate performance







# Driving the Analysis: SVL

```
"scsi.bcg" = branching reduction of
    total rename "ARB !.*" -> ARB in
    hide CMD, REC in
    "scsi.lotos";

"model.bcg" = hide all but LAMBDA, MU, NU in
    ("scsi.bcg" |[ARB]| "erlang.lotos":BUS [ARB, NU])

% for SPEED in .4 2 4 40 400
% do
    % for LOAD in .01 .03 .06 .1 .15 .2 .25 .3 .35 \
    %             .4 .45 .5 .55 .6 .65 .7 .75 .8
    % do
        % BCG_MIN_OPTIONS="-rate"
        "res-$$SPEED.bcg" = branching reduction with bcg_min of
            total rename "NU" -> "rate $$SPEED",
            "MU !0" -> "DISK_L; rate .4",
            "MU !1" -> "DISK_M; rate .4",
            "MU !2" -> "DISK_H; rate .4",
            "LAMBDA !.*" -> "rate $$LOAD" in
            "model.bcg";
        % seidel -v $$LOAD "res-$$SPEED.bcg"
    % done
% done
```



## Summary?

- CADP (one of the main functional verification tools) has been extended towards performance evaluation;
- Formal basis: Interactive Markov Chains;
- Pragmatic integration into CADP syntax;
- Main effort is in the minimizer `bcg_min`;
- Application to the SCSI-2 case.