A NEUTRAL MECHANISM FOR THE DETECTION OF DEADLOCKS UNDER PVM

Serge Chaumette & Bruno Teplier

Serge.Chaumette@labri.u-bordeaux.fr

LaBRI,
Université Bordeaux I,
351 Cours de la Libération,
33405 Talence,
FRANCE

October 1993

CONTENTS

Introduction

General goals:

- provide high level tools ...
- ... with a formal foundation

Current topic:

provide a neutral mechanism for collection of distributed, communication related information

Basis:

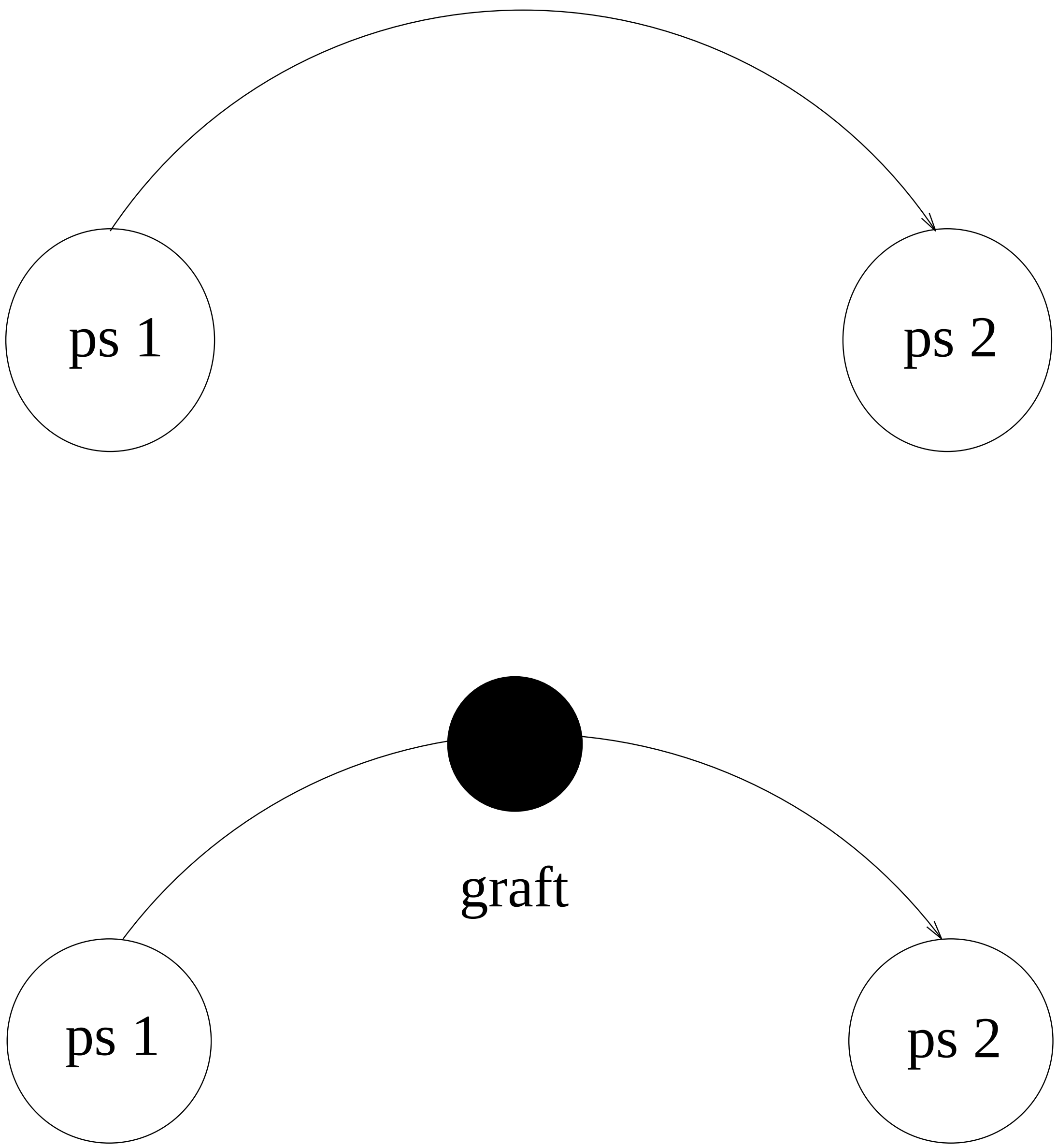
previous work on *grafts* and *spies*

Illustration:

a neutral deadlock detection mechanism

CONTENTS

Principle of the mechanism



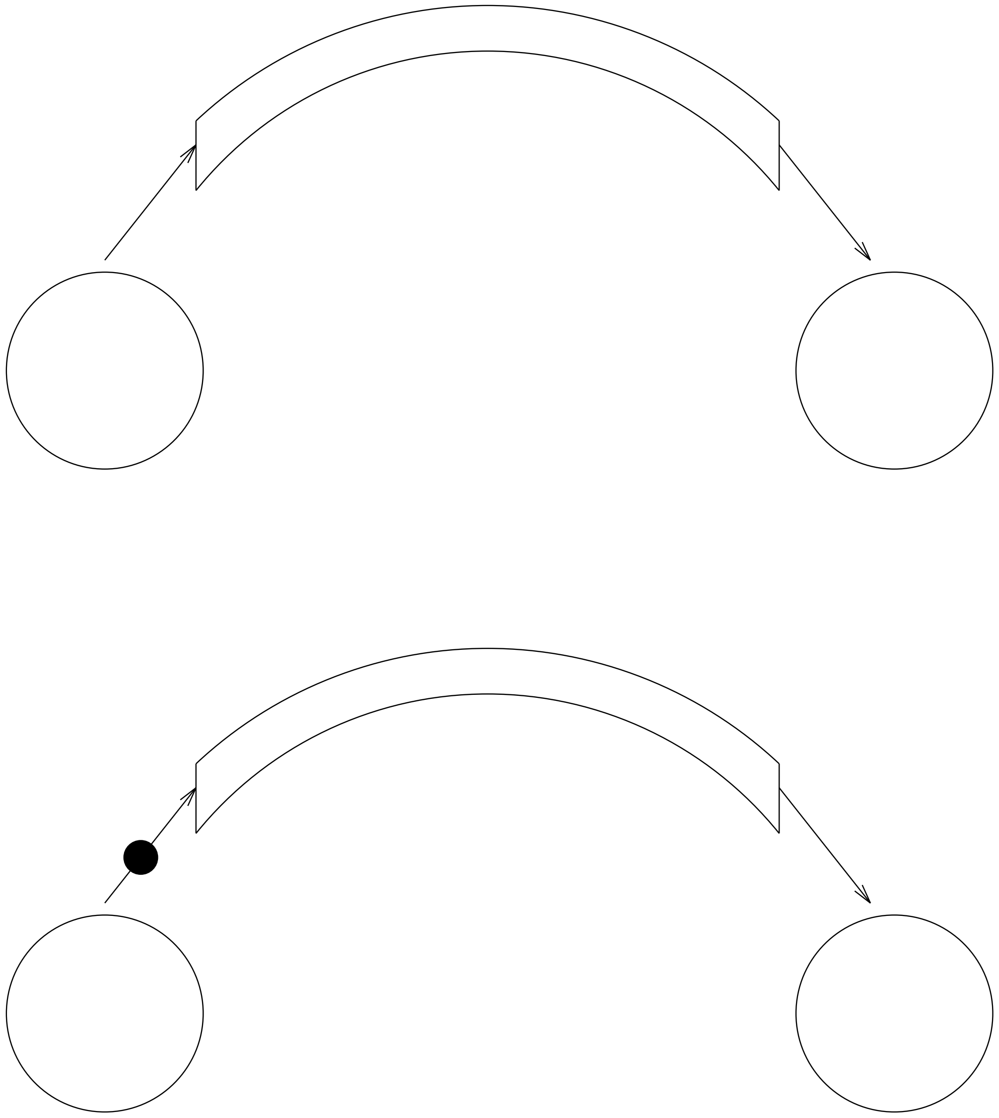
Synchronous framework

Spies are grafts that do not modify the contents of the messages.

→ they are "neutral" (CCS)

[See Parco93]

"Neutralization" of asynchronism

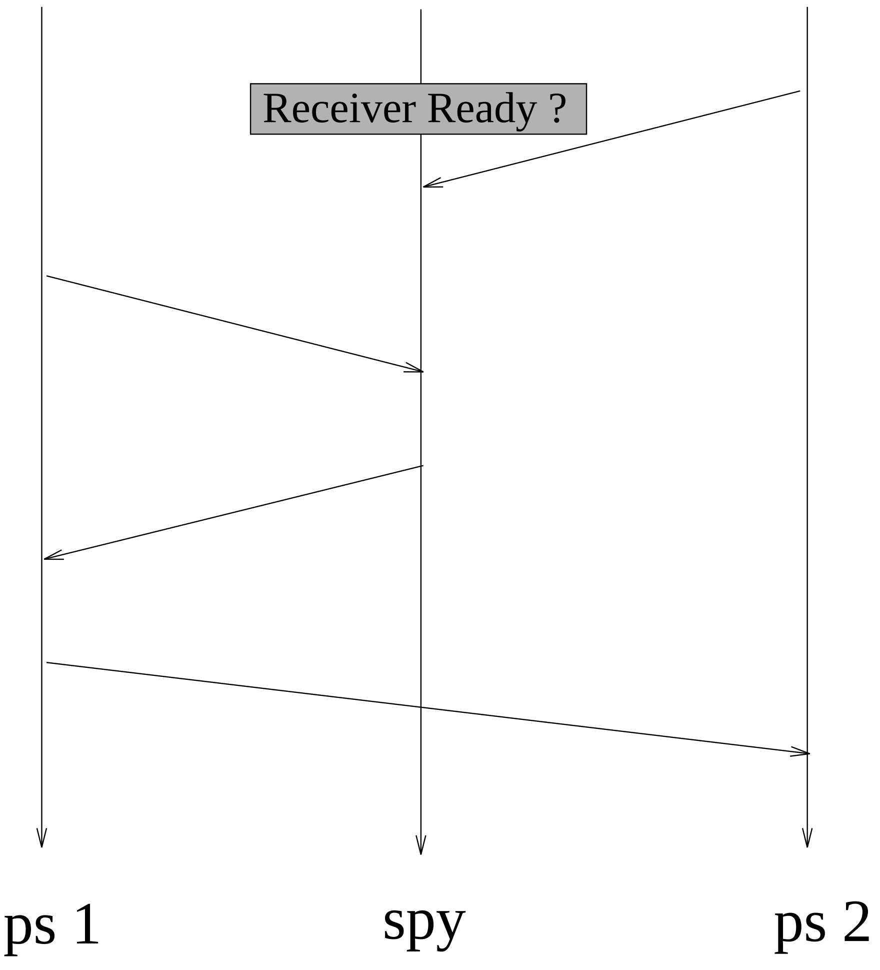


[See PVMUG, Kxv, 94]

Implementation

Pb.: PVM processes communicate using tids

⇒ the message <u>must</u> be sent by
the initial sender process



CONTENTS

$\surd$ • Introduction

$\surd$ • Neutral observation of links

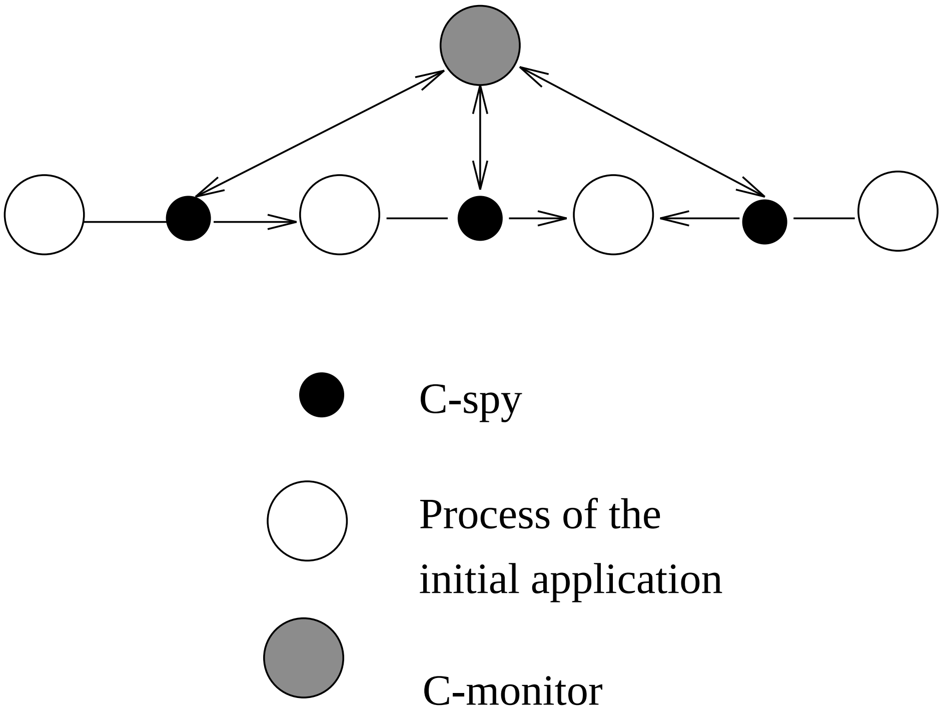
$\Rightarrow$ • Neutral centralization

- Principle of the mechanism
- Implementation

• Use example

• Future work

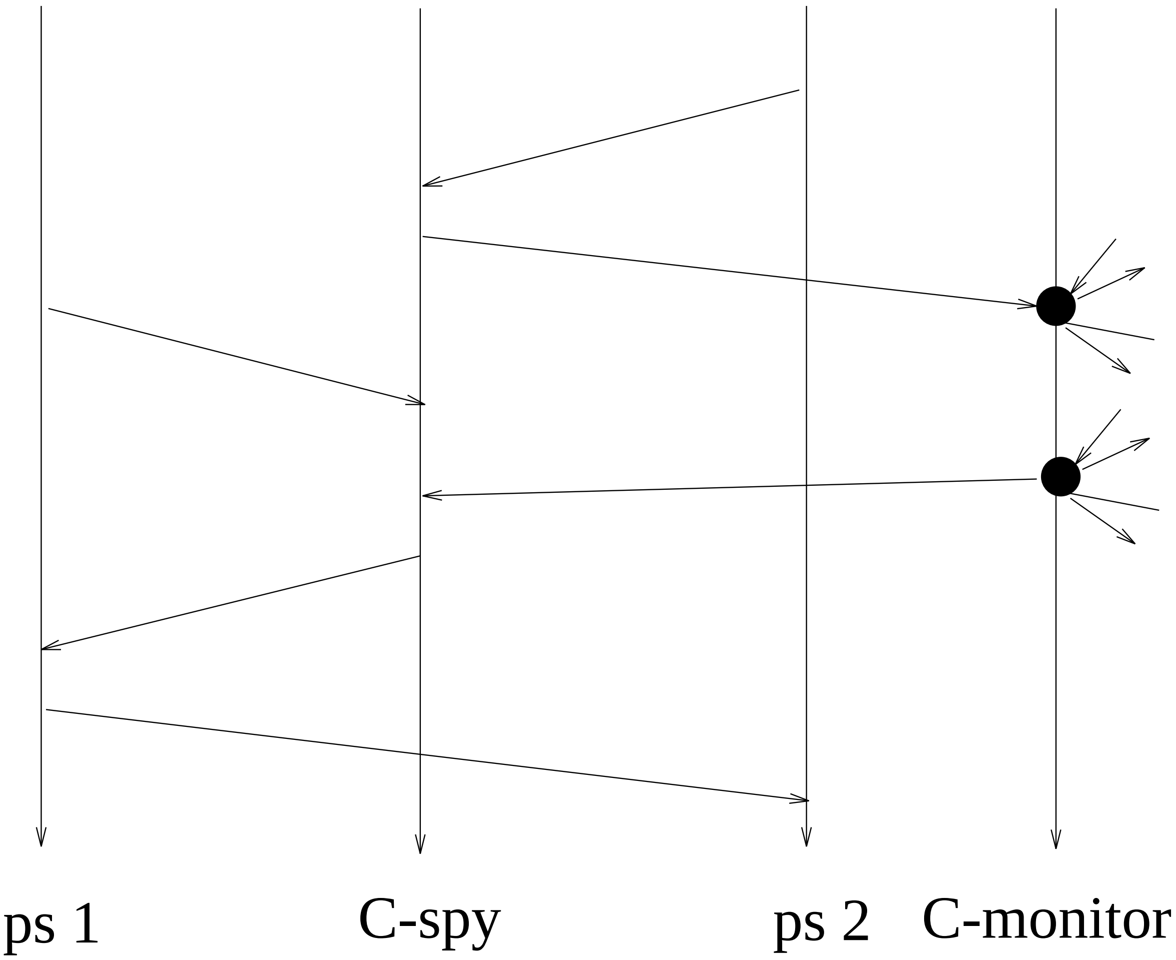
Principle of the mechanism



→ they are “neutral” (CCS)

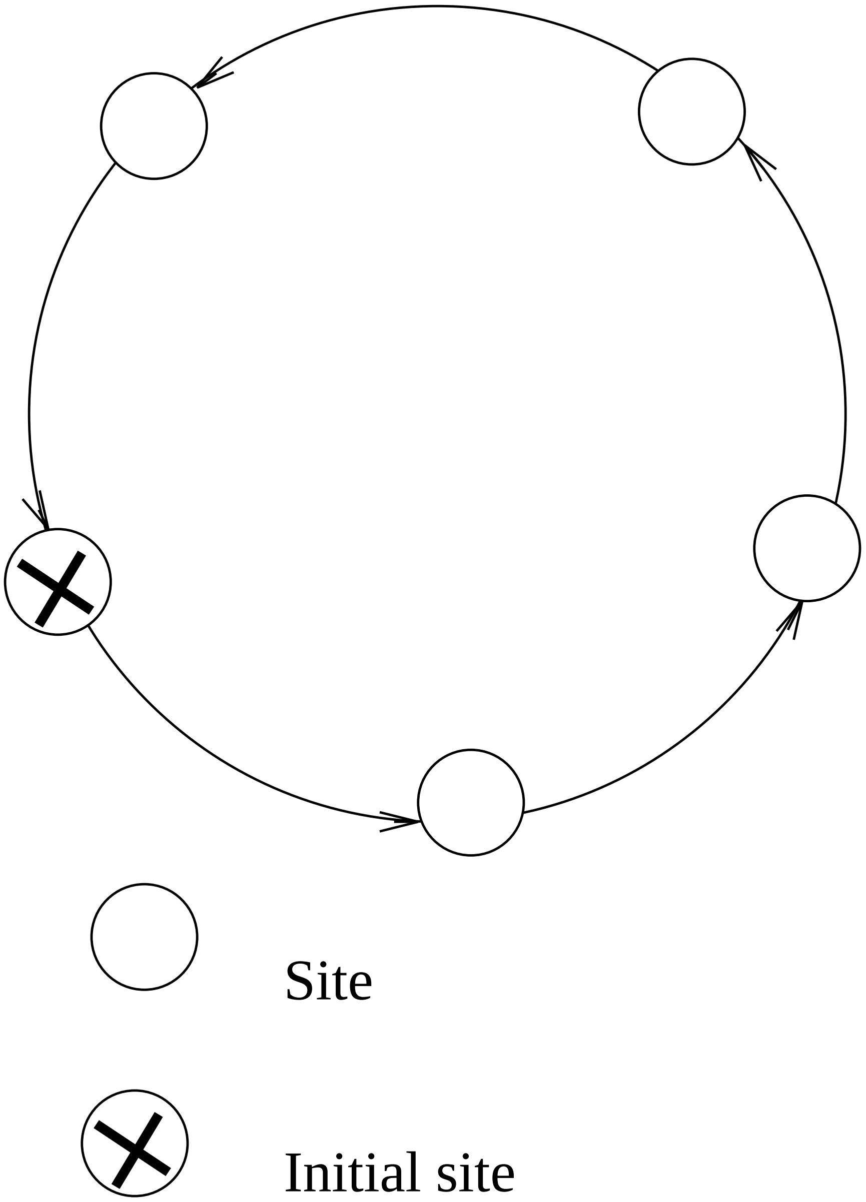
Implementation

$\Rightarrow$ the monitor can receive from and send to any spy in a non-deterministic manner



CONTENTS

Algorithm



— SOURCE 1 : site.c ———

```
[...]
main(int argc, char **argv){

    int tid[2];
    int token_flag=10;
    int loop;

    pvm_mytid();
    enter_ring(tid,argv);

    for (loop=0;loop<=NB_LOOPS;loop++){

        pvm_recv(tid[0],token_flag);
        pvm_initsend(PvmDataDefault);
        pvm_send(tid[1],token_flag);
    }
    pvm_exit();
}
[...]
```

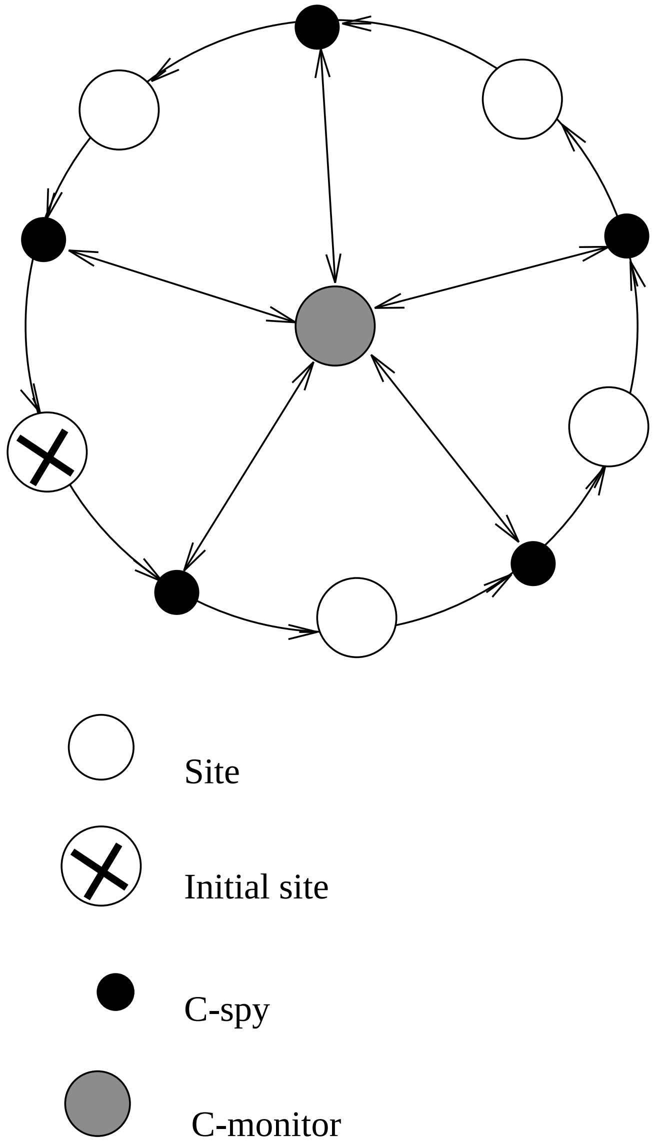
——— SOURCE 1 : site.c —

— Source 2 : init.c —

```
[...]
main(int argc, char **argv){

    int tid[2];
    int loop;
    int token_flag=10;
    int duration[NB_LOOPS];
    double mean_time;

    pvm_mytid();
    enter_ring(tid,argv);

    for (loop=0;loop<=NB_LOOPS;loop++){

        duration[loop]=(int) time(NULL);

        /* Send token to the ring */
        pvm_initsend(PvmDataDefault);
        pvm_send(tid[1],token_flag);

        /* Receive token from the ring */
        pvm_recv(tid[0],token_flag);

        /* Compute duration of this loop */
        [...]
    }
    /* Compute mean duration of a loop */
    [...]
    pvm_exit();
}
[...]
```

— Source 2 : init.c —

Monitoring



— Source 3 : c-spy.c —

```
[...]
main(int argc, char **argv){

    int sender, receiver ,moniteur;

    pvm_mytid();

    _gs_setup(&argc, argv, &sender, &receiver);

    sscanf(argv[1],"%i", &moniteur);

    while (1){

        /* Wait for receiver to become */
        /*Ready to receive             */
        _gs_wait_for_receiver(receiver);

        /* Tell that to the C-monitor */
        msg_moniteur(moniteur, WAITING,
                     sender, receiver);

        /* Receive from the sender */
        _gs_mess_in(sender, receiver);

        /* Forward to the receiver */
        _gs_mess_out(sender, receiver);

        /* Tell that to the C-monitor */
        msg_moniteur(moniteur, DONE,
                     sender, receiver);
    }
}
[...]
```

— Source 3 : c-spy.c —

— Source 4 : c-monitor.c —

```
[...]
main(int argc, char **argv){

    int sender_tid, receiver_tid;
    int communication_status;

    pvm_mytid();

    while(1){

        /* Receive a message from any C-spy */
        pvm_recv(-1,-1);

        pvm_upkint(&sender_tid,1,1);
        pvm_upkint(&receiver_tid,1,1);
        pvm_upkint(&communication_status,1,1);

        /* Store state of communication */
        state_table[rank(sender_tid)]
                   [rank(receiver_tid)]
                       = communication_status;

        /* Is there a deadlock now ? */
        test_for_deadlock();
    }
}
[...]
```

— Source 4 : c-monitor.c —

“Monitored” execution

— Source 5 : Execution example (I of II) –

```
[...]
   $ pvm
   [...]
   pvm> spawn -> init 4
   [...]
   pvm> spawn -> c-monitor
   [...]
   pvm> ps
              HOST      TID      FLAG COMMAND
              isis    40002     0x4/c pvmgs
              isis    40003     0x4/c init
              isis    40004     0x4/c site
              isis    40005     0x4/c site
              isis    40006     0x4/c site
              isis    40007     0x4/c site
              isis    40008     0x4/c c-monitor
   pvm> spawn -> c-spy 0x40003 0x40004 0x40008
   [...]
   pvm> spawn -> c-spy 0x40004 0x40005 0x40008
   [...]
   pvm> spawn -> c-spy 0x40005 0x40006 0x40008
   [...]
   pvm> spawn -> c-spy 0x40006 0x40007 0x40008
   [...]
   pvm> spawn -> c-spy 0x40007 0x40003 0x40008
   [...]
[...]
```

— Source 5 : Execution example (I of II) —

— Source 6 : Execution example (II of II)

```
[...]
   pvm>
   [2:t40003] Loop #0 took 1 second(s)
   [2:t40003] Loop #1 took 2 second(s)
   [2:t40003] Loop #2 took 1 second(s)
   [2:t40003] Loop #3 took 1 second(s)
   [...]
   [2:t40003] Loop #98 took 1 second(s)
   [2:t40003] Loop #99 took 1 second(s)
   [3:t40008] The following deadlock has been detected:
   [3:t40008]
   [3:t40008]  site_40007 ->init_40003 ->site_40004 ->
               site_40005 ->site_40006 ->site_40007
   [...]
   pvm> ps
              HOST      TID      FLAG COMMAND
              isis    40002     0x4/c pvmgs
              isis    40003     0x4/c init
              isis    40004     0x4/c site
              isis    40005     0x4/c site
              isis    40006     0x4/c site
              isis    40007     0x4/c site
              isis    40008     0x4/c c-monitor
              isis    40009     0x4/c c-spy
              isis    4000a     0x4/c c-spy
              isis    4000b     0x4/c c-spy
              isis    4000c     0x4/c c-spy
              isis    4000d     0x4/c c-spy
   pvm>
[...]
```

— Source 6 : Execution example (II of II) —

$\boxed{\text{CONTENTS}}$

- $\surd$ • Introduction
- $\surd$ • Neutral observation of links
- $\surd$ • Neutral centralization
- $\surd$ • Use example
- $\Rightarrow$ • Future work

Future work

Use a monitor for

$\rightarrow$ traces collection $\rightarrow$ replay
$\rightarrow$ tuning

$\rightarrow$ visualization

$\rightarrow$ ...