PCOMS

PCOMS
Prioritised Choice Over Multiway-Synchronisation

Douglas N. Warren
Computing Laboratory, University of Kent

PCOMS – the plan

- 5 minutes – What?, why?, how? black box
- 20 minutes – Detail
- Pretty(ish) Demo

PCOMS - template

PCOMS – 5 minute explanation

- What is it?

- COMS – Choice Over Multiway Synchronisation
- Prioritised – Ability to select one event over another

- JCSP compatible – written in Java, can be used with the JCSP Alternative class
- Prototype Algorithm – unoptimised, not formally verified, lacking (a few) features.

PCOMS – 5 minute explanation

- What does it do?
- Reliable, generic, atomic message propagation:
  - Pausing, graceful termination
- Anything existing COMS implementations can do:
  - facilitate output guards, broadcast channels

PCOMS – 5 minute explanation

- How do I use it?

- Basically similar to any other communication primitive or Guard.

- Some of the features will be explained in further detail but for now ...

PCOMS - 5 minute explanation

Construction

JCSP code

```
AltableBarrierBase barrier = new AltableBarrierBase();

AltableBarrier bar1 = new AltableBarrier(barrier);

AltableBarrier bar2 =
        new AltableBarrier(barrier, ABConstants.UNPREPARED);
```

occam-pi equivalent

```
MOBILE BARRIER bar:
SEQ
  bar := MOBILE BARRIER
```

There is no real equivalence between the construction of JCSP AltableBarriers and their nearest occam-pi equivalents

PCOMS - 5 minute explanation

GuardGroup

Collection of AltableBarriers considered to be of equal priority but which can belong to a wider priority structure.

JCSP code

```
AltableBarrier left, right;

assignLeftAndRight(); // some method for initialising left & right

GuardGroup group = new GuardGroup(
        new AltableBarrier[] {left, right}
);
```

occam-pi equivalent

```
ALT
  SYNC left
    SKIP
  SYNC right
    SKIP
```

PCOMS - 5 minute explanation

GuardGroup

Collection of AltableBarriers considered to be of equal priority but which can belong to a wider priority structure.

JCSP code

```
AltableBarrier left, right;

assignLeftAndRight(); // some method for initialising left & right

GuardGroup group = new GuardGroup(
        new AltableBarrier[] {left, right}
);
```

occam-pi equivalent

```
ALT
  SYNC left
    SKIP
  SYNC right
    SKIP
```

PCOMS - 5 minute explanation

Alternative

This is how several GuardGroup objects can be made to fit in a wider priority structure.

JCSP code

```
AltableBarrier left, right, middle, high;
assignBarriers(); // some method for initialising barriers

GuardGroup low =
      new GuardGroup(new AltableBarrier[] {left, right});
GuardGroup mid =
      new GuardGroup(new AltableBarrier[]{middle});
GuardGroup hi =
      new GuardGroup(new AltableBarrier[]{high});

Guard[] guards = new Guard[]{hi, mid, low};
Alternative alt = new Alternative(guards);
```

occam-pi equivalent

```
PRI ALT
  SYNC high
    SKIP
  SYNC middle
    SKIP
  ALT
    SYNC left
      SKIP
    SYNC right
      SKIP
```

PCOMS - 5 minute explanation

Resolution

Evaluating the Alternative is easy, discovering which Barrier was selected is slightly more difficult:

JCSP code

```
int index = alt.priSelect();

GuardGroup group = (GuardGroup) guards[index];
AltableBarrier selected = group.lastSynchronised();

if (selected == left) {
        doSomething();
} else {
        doSomethingElse();
}
```

occam-pi equivalent

```
PRI ALT
   SYNC left
      do.something()
   SYNC right
      do.something.else()
```

PCOMS – other COMS algorithms

- 2 and 3 phase commit protocols.
- Alistair McEwan's thesis.
- (just now) Gavin Lowe's algorithm
- The 'oracle' method as implemented in the JCSP AltingBarrier class

PICOMS - Oracle Method

- Grab a global lock when reading/writing any barrier data

- Offer to synchronise on barrier when encountered

  - Offer remains until withdrawn

PICOMS – Oracle Method

- First barrier to receive offers from all enrolled processes wins

- If barrier picked, suppress any other ready guards and report that the barrier was picked.

- Very simple and efficient

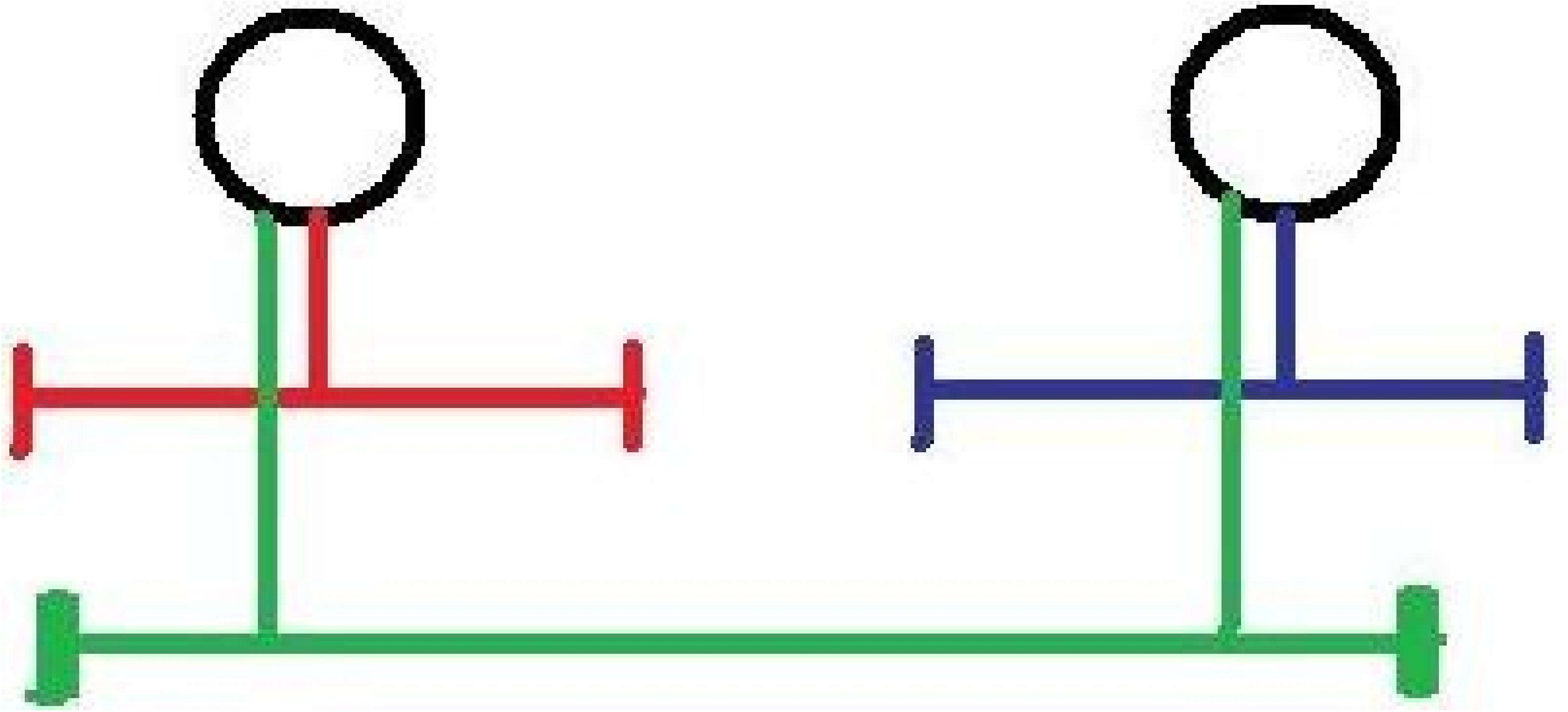
PICOMS – Problems with Oracle

- Oracle is incompatible with meaningful priority

- Certain event / process combinations render some events unselectable.

```
PAR
    ALT                 ALT
        SYNC a              SYNC b
            SKIP                SKIP
        SYNC c              SYNC c
            SKIP                SKIP
```

PICOMS – Problems with Oracle



PCOMS – Problems with Oracle

- If the left-hand process runs first, A is picked.

- If the right-hand process runs first B is picked.

- When barrier C's set of enrolled processes is a super-set of barriers A and B (where A ∩ B is { }) it is impossible to select C in preference to A or B.

- Impossible to pick large global barriers in preference to small local ones.

PCOMS - Priority

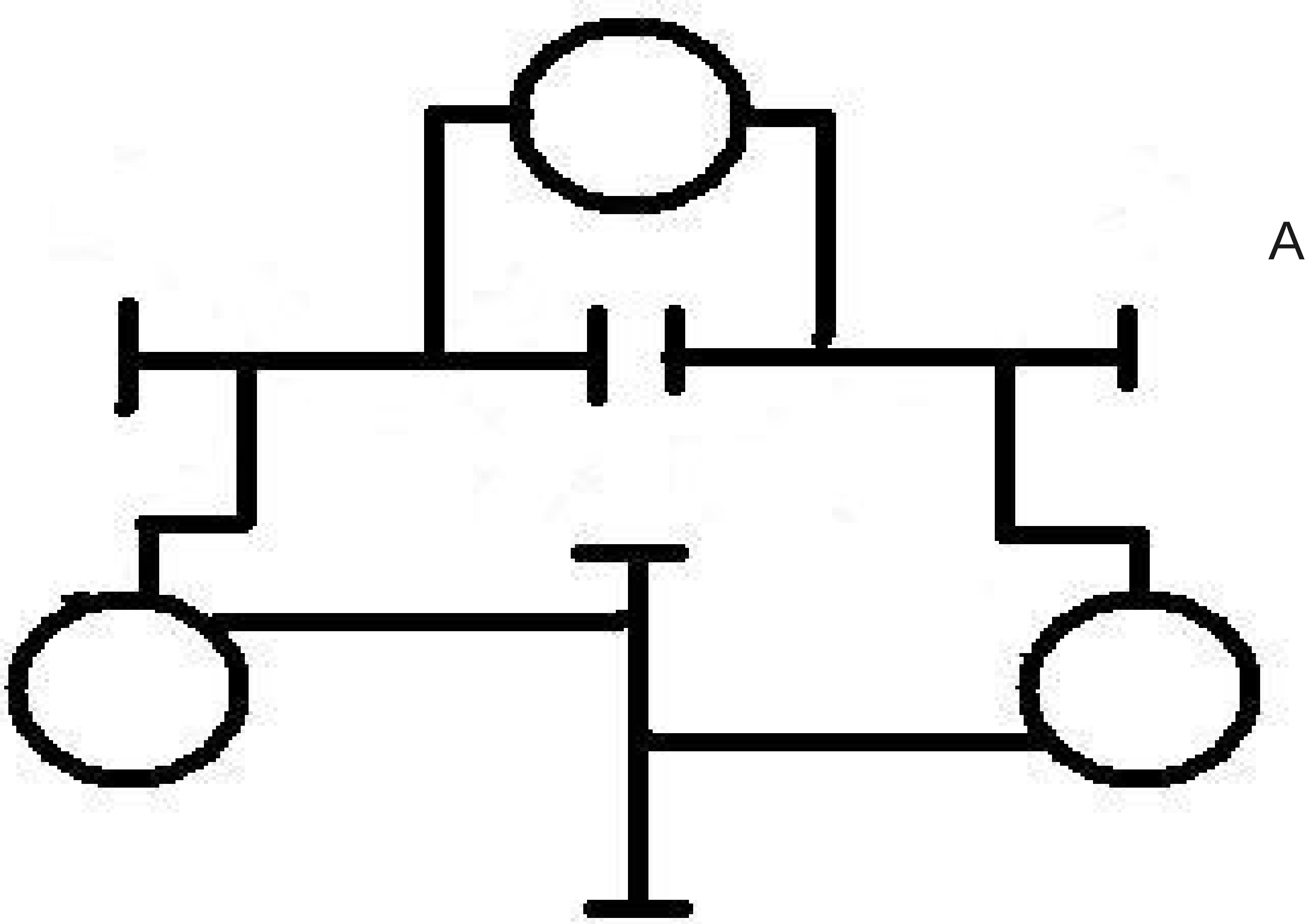
- First-come-first-served is not compatible with priority.
- Conjecture: This can be overcome by giving events the benefit of the doubt. Pre-emptively waiting for events to complete.
- This allows for false positives and negative.
- This is less a redefinition of what 'priority' means and is more a redefinition of 'ready'.

PICOMS - Need for Nesting

- Sometimes the absence of priority between barriers is a good thing.

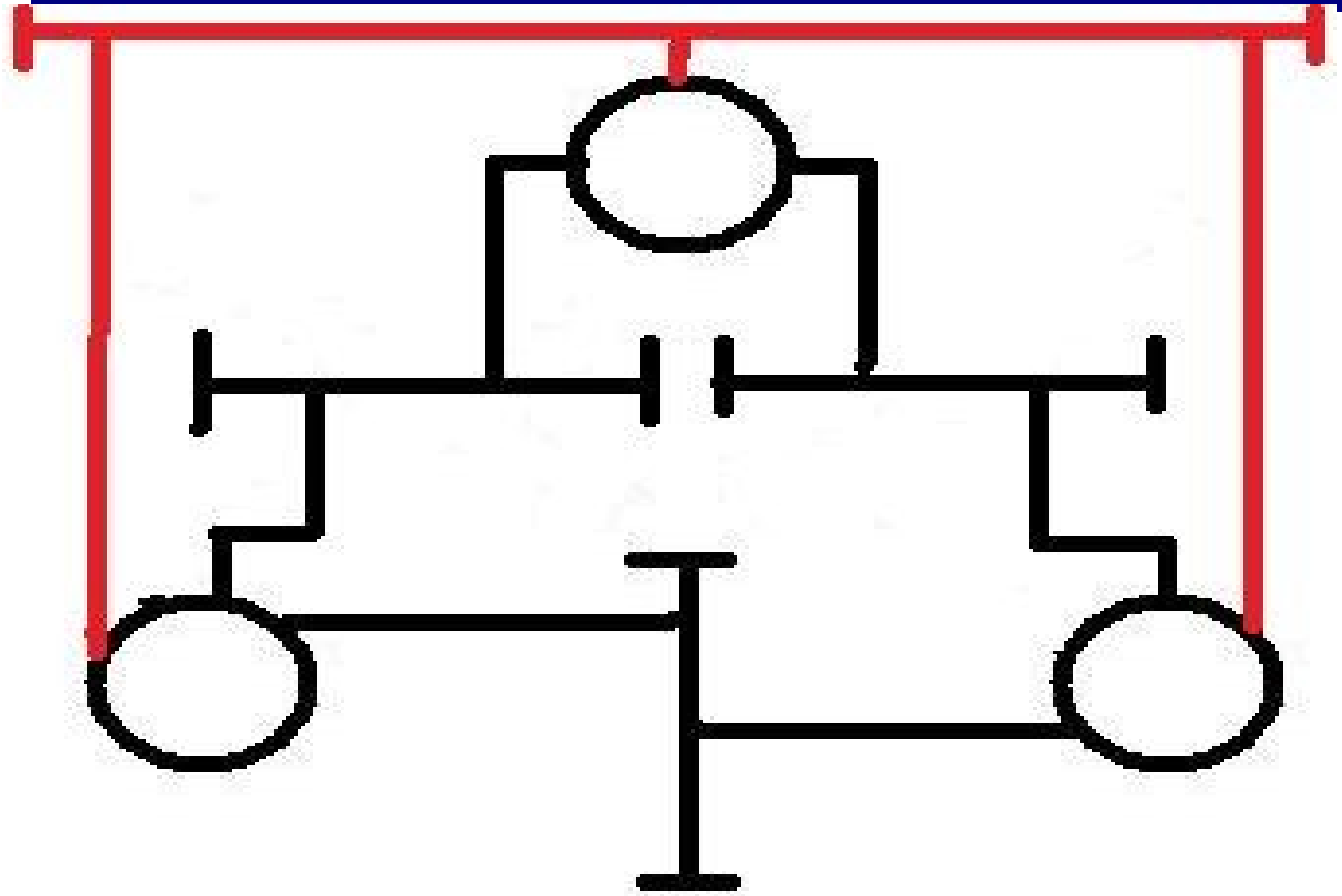
- When adding a high priority barrier to an existing choice, it may be useful to NOT change the relative priorities of the existing barriers.

PICOMS – Need for Nesting



```
ALT
  SYNC anti.clockwise
    SKIP
  SYNC clockwise
    SKIP
```

PICOMS – Need for Nesting



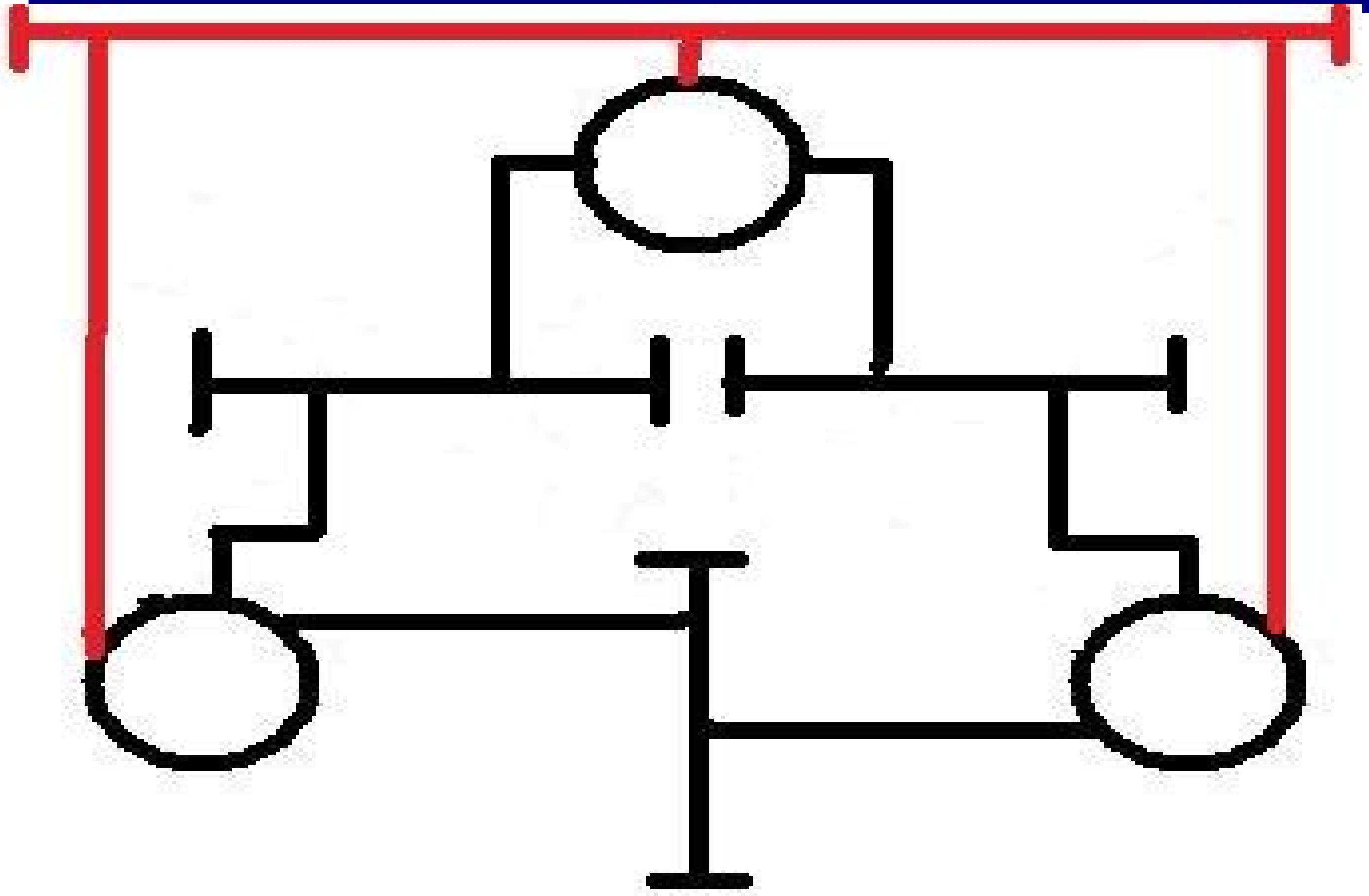
```
PRI ALT
  SYNC pause
    SKIP
  SYNC anti.clock
    SKIP
  SYNC clock
    SKIP
```

PICOMS – Need for Nesting

- Example: where the choice of barriers in one process partially overlap those of another process, introducing priority may cause priority conflict.

- Therefore there needs to be a means of having a number of barriers have no priority among themselves but still fit in a wider priority structure.

PICOMS – Need for Nesting



```
PRI ALT
    SYNC pause
        SKIP
    ALT
        SYNC anti.clock
            SKIP
        SYNC clock
            SKIP
```

PICOMS - Glossary

- AltableBarrier: The object that processes use to interact with a barrier. One object per process for each barrier the process is enrolled on

- AltableBarrierBase: The object to representing the barrier itself and which all AltableBarriers talk to.

- GuardGroup: acts as a collection of AltableBarriers at the same priority and which does extend Guard.

PICOMS - Glossary

- UNPREPARED/PREPARED: A (possibly false) assertion that the process will offer to synchronise on this barrier in the near future. Processes which regularly ALT on inputs (such as server processes) should default to PREPARED.

- PROBABLY_READY: an AltableBarrierBase is considered PROBABLY_READY if all enrolled processes are PREPARED to synchronise.

PICOMS - Specifics

- When evaluating an ALT and a GuardGroup is encountered

- Phase 1: select a barrier

- Claim global lock

- Tell barriers you are PREPARED to synchronise
  - This should be done for all barriers in the current GuardGroup as well as all previously encountered barriers

PICOMS - Specifics

- Select a barrier – Do this for all of the barriers in the current GuardGroup AS WELL AS those previously evaluated in this ALT.

- To be done in priority order.

  - Are any barriers PROBABLY_READY?

  - If none have been selected by other processes, select arbitrarily.

  - Otherwise pick a barrier which has already been selected.

PICOMS - Specifics

- Phase 2: attempt synchronisation
  - 'Steal' other processes enrolled on the barrier
    - If other process is waiting on another barrier transfer it to this one (as long as it is of an equal or lower priority).
    - If not ignore it. Those processes will eventually turn up.

PICOMS - Specifics

- If this is the first process to select the barrier, start a time-out.

- If the time-out elapses before the barrier completes wake everyone up and let them know the synchronisation attempt failed.

- For all processes which failed to turn up before the time-out, set their status flag to UNPREPARED.

PICOMS - Specifics

- Claim a local lock and release the global lock.

- Next wait on the local lock for one of the following to happen:

    - The synchronisation attempt succeeds

    - One of the enrolled processes to set its status flag to UNPREPARED, thus aborting the sync attempt

    - The time-out , thus aborting the sync attempt

PICOMS - Specifics

- when woken release local lock and reclaim global one

- check to see if synchronisation was successful and if it was which barrier completed (the process may have been 'stolen' by another barrier while it waited).

PCOMS – Skipped over

- Some detail missing, see the paper

- Phase 3: Involves making sure that once a synchronisation is successful that it is accurately reported.

- No guards were initially ready … waiting on the 'altmonitor'

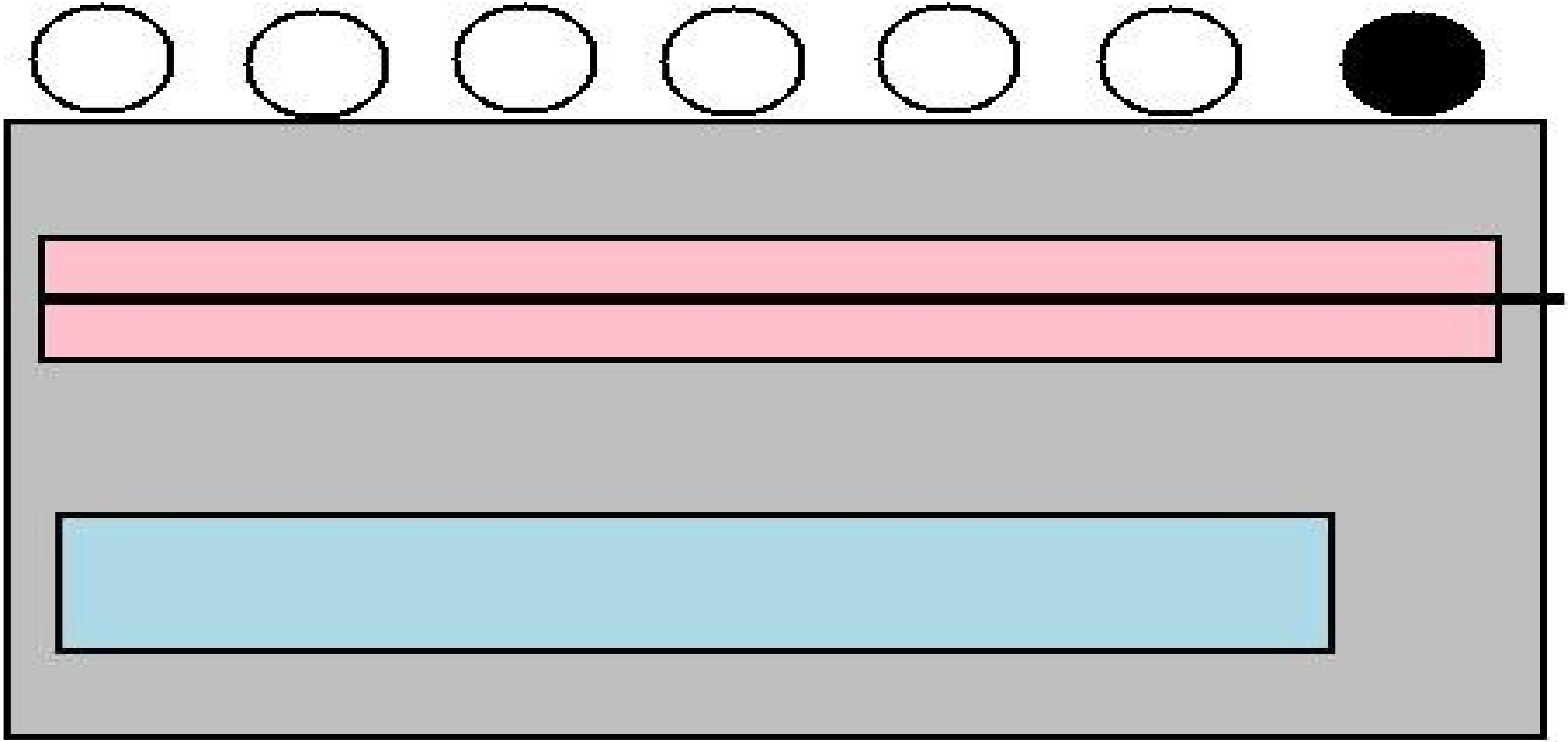
PCOMS – diagram key

- Grey box = ALT
- Pink box = barrier A
- Blue box = barrier B

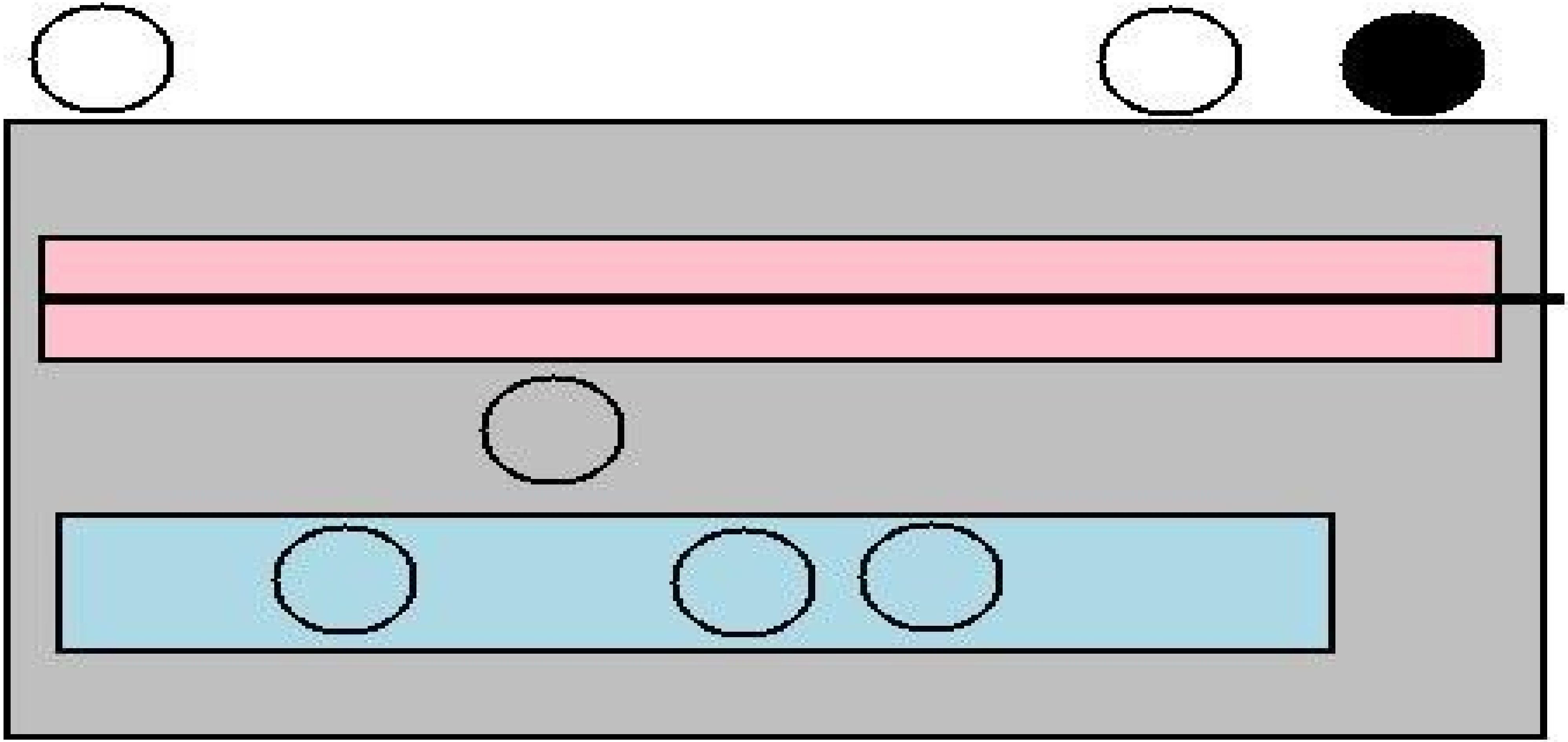
- Clear circle = process PREPARED to synchronise on A
- Black circle = process UNPREPARED to synchronise on A
- All processes are PREPARED to synchronise on B

- Black line = barrier is currently not PROBABLY_READY

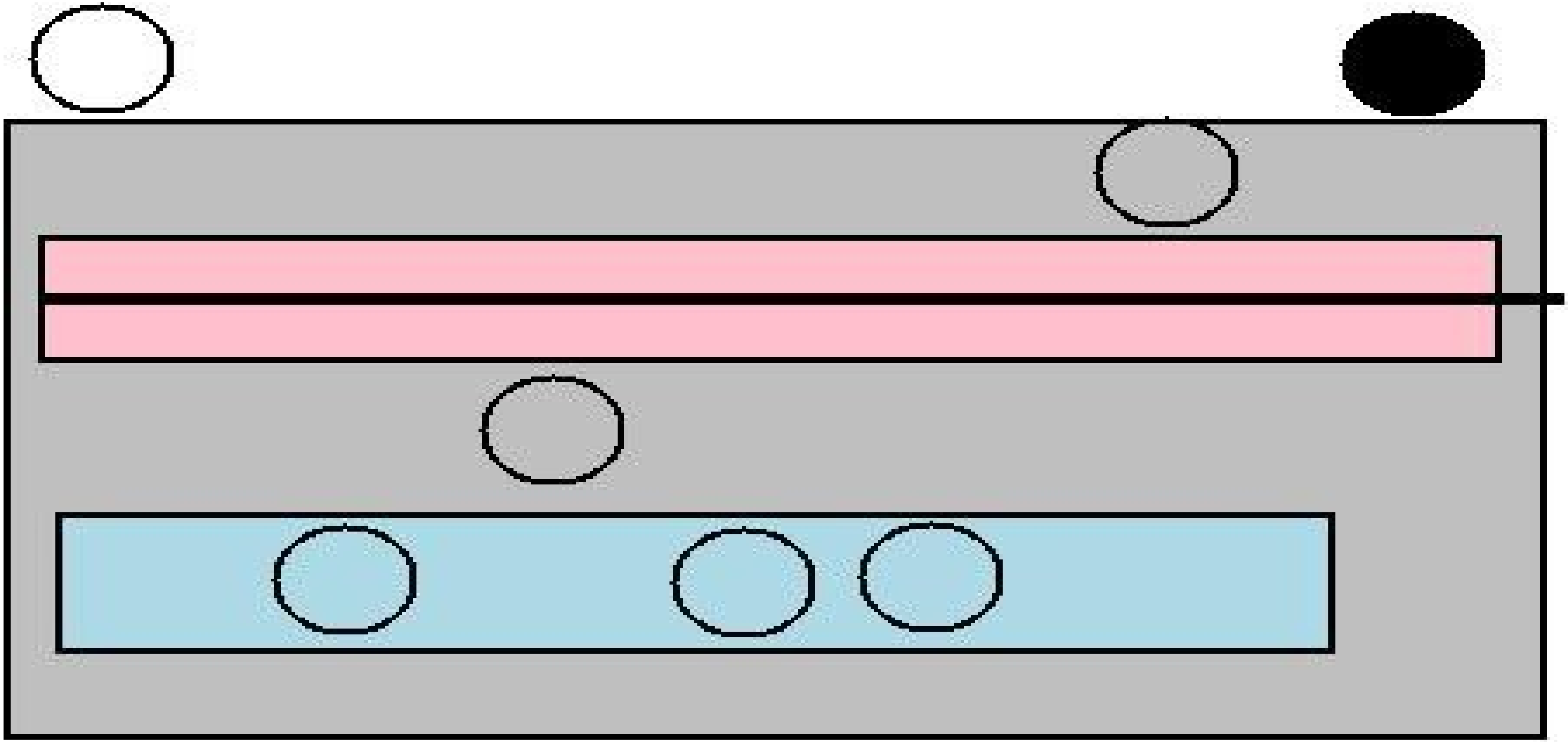
PCOMS - example



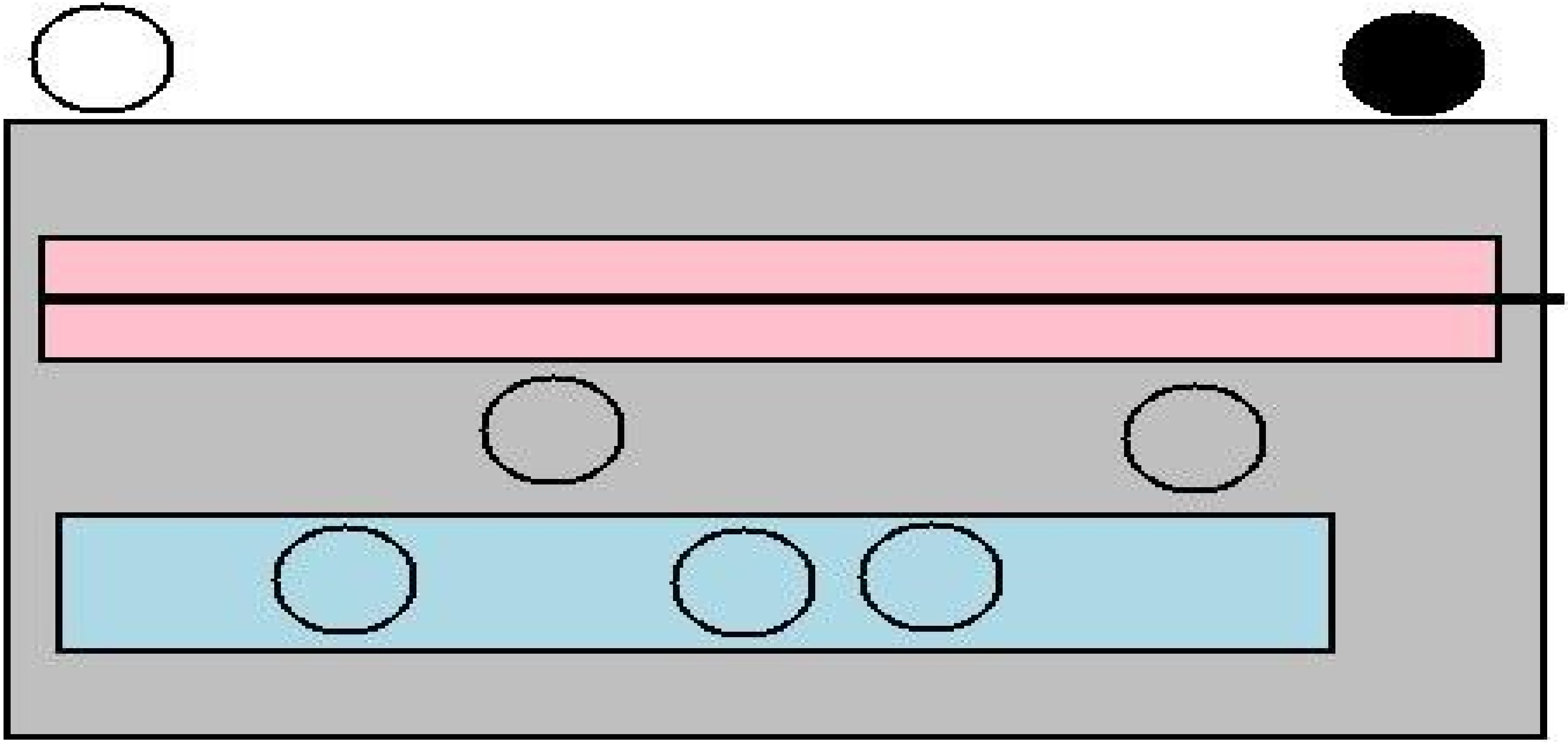
PCOMS - example



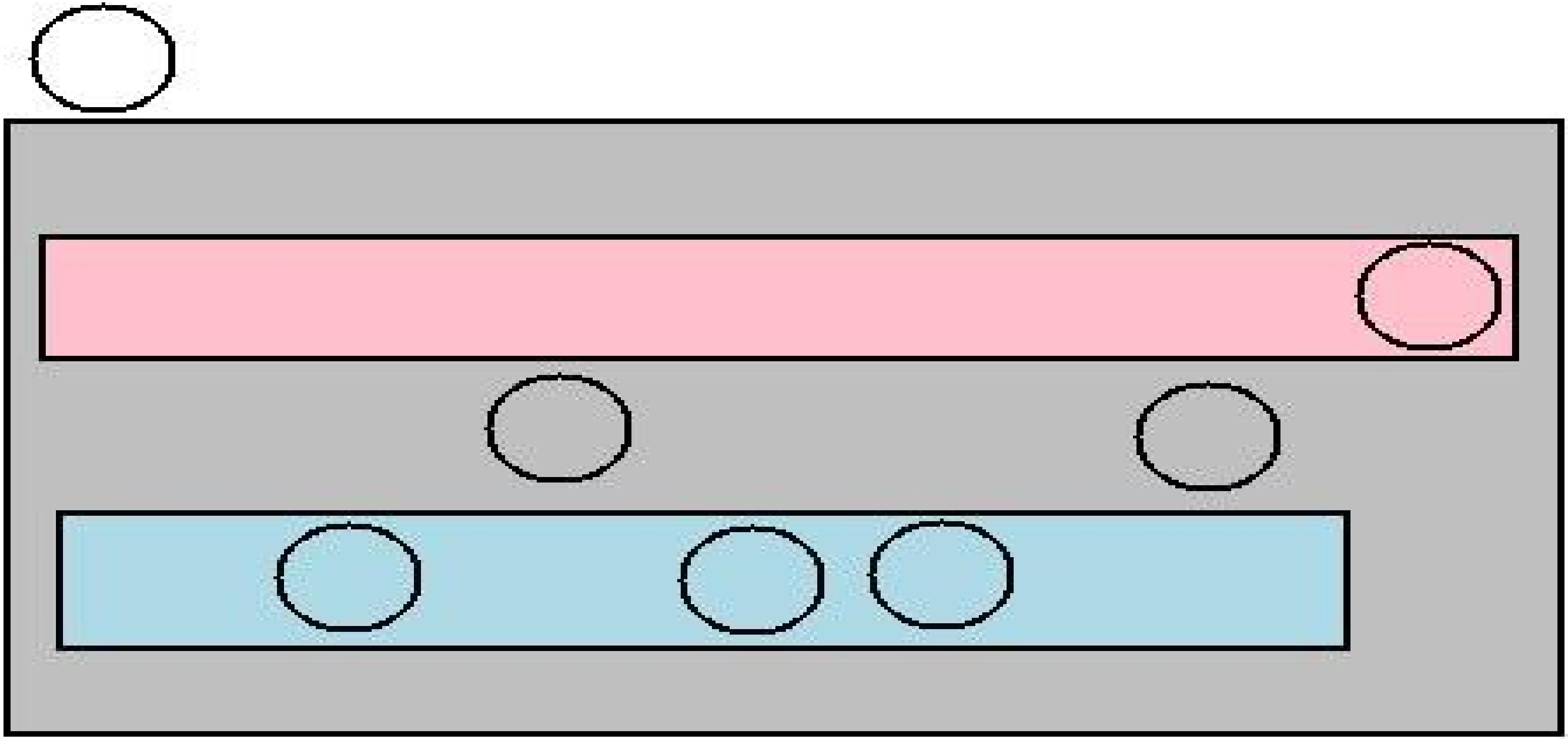
PCOMS - example



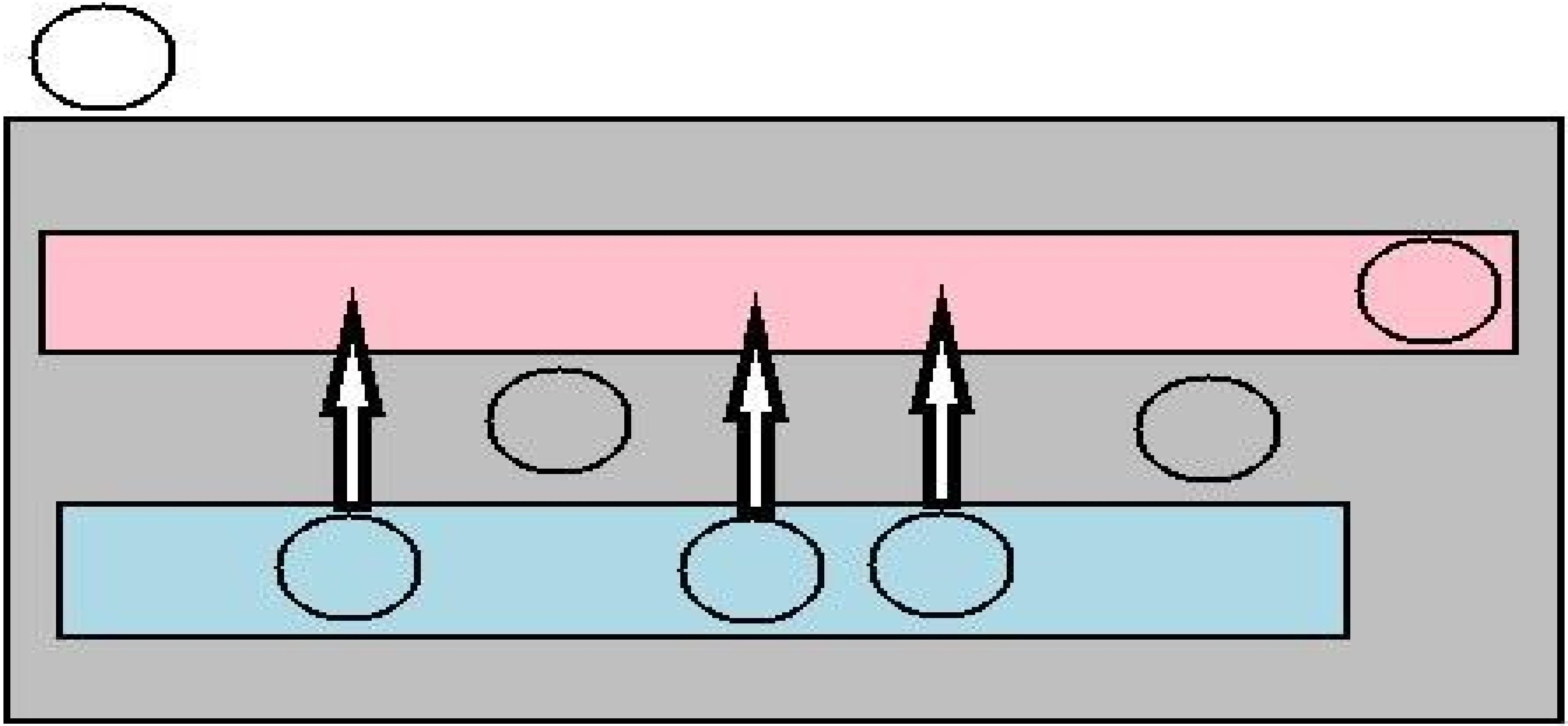
PCOMS - example



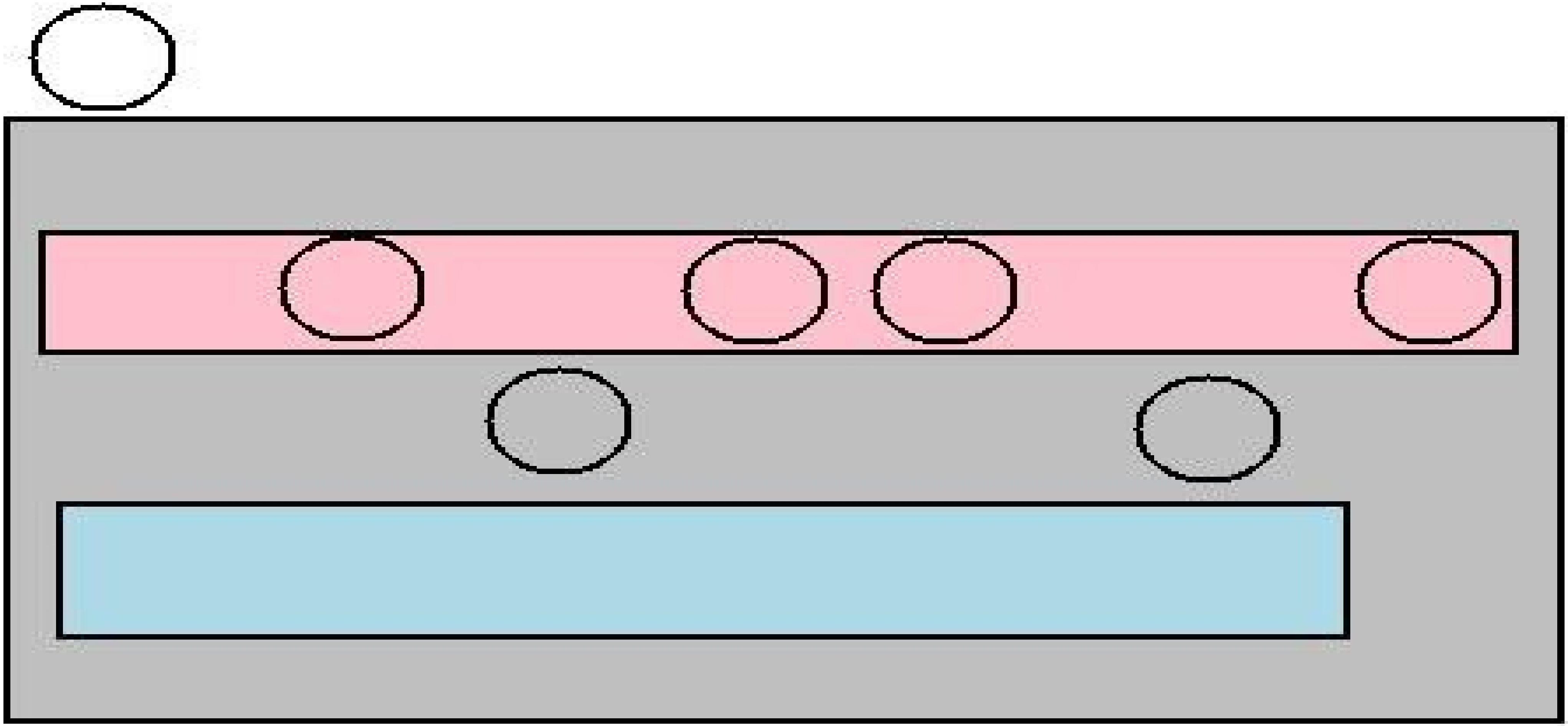
PCOMS - example



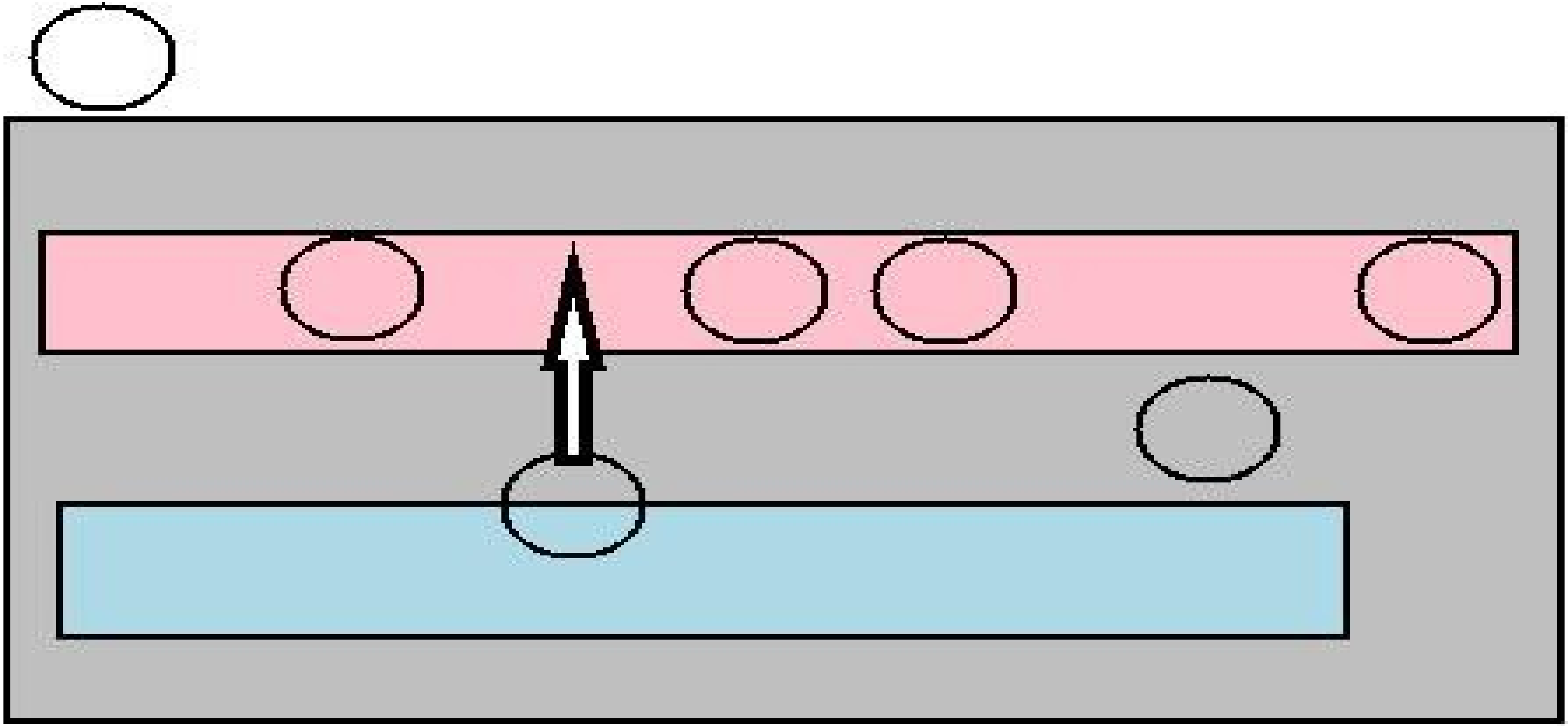
PCOMS - example



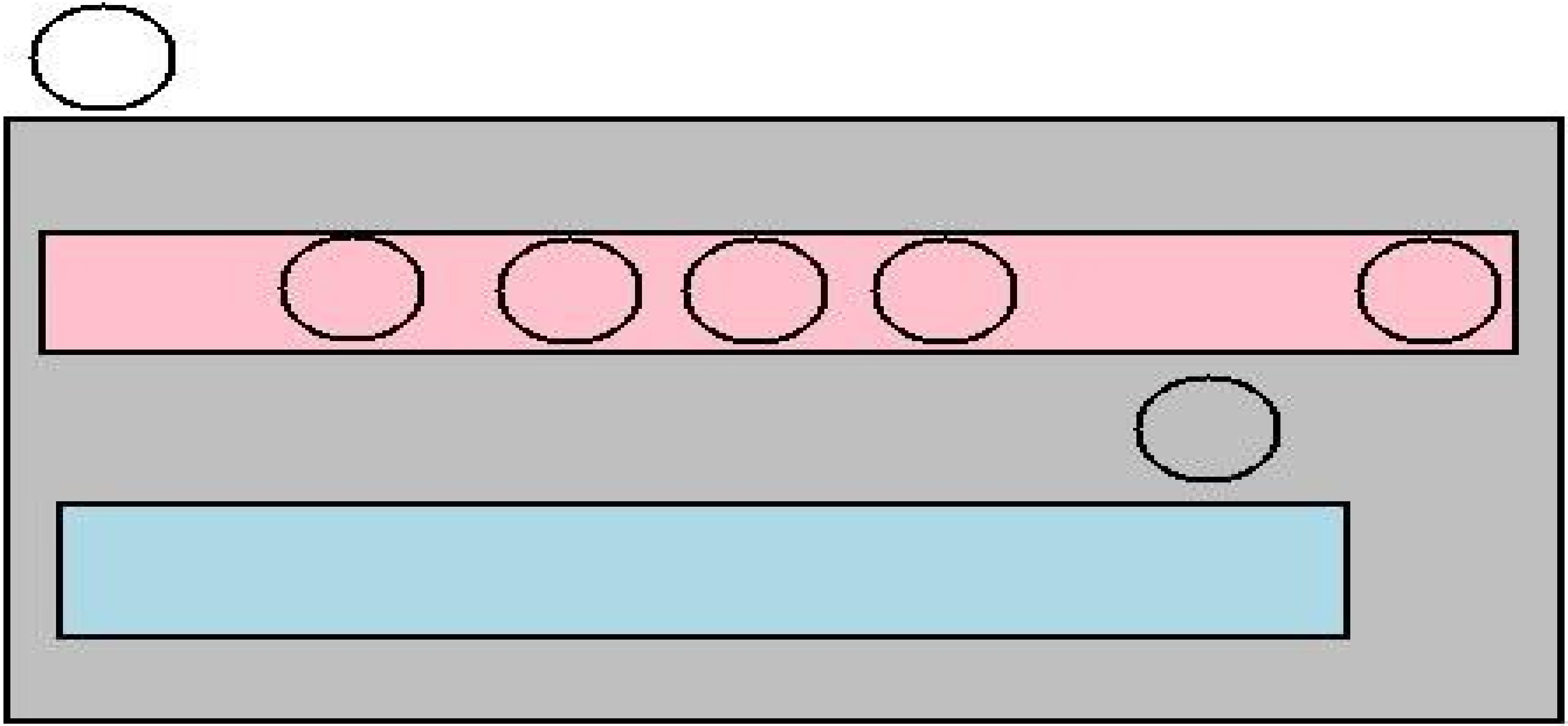
PCOMS - template



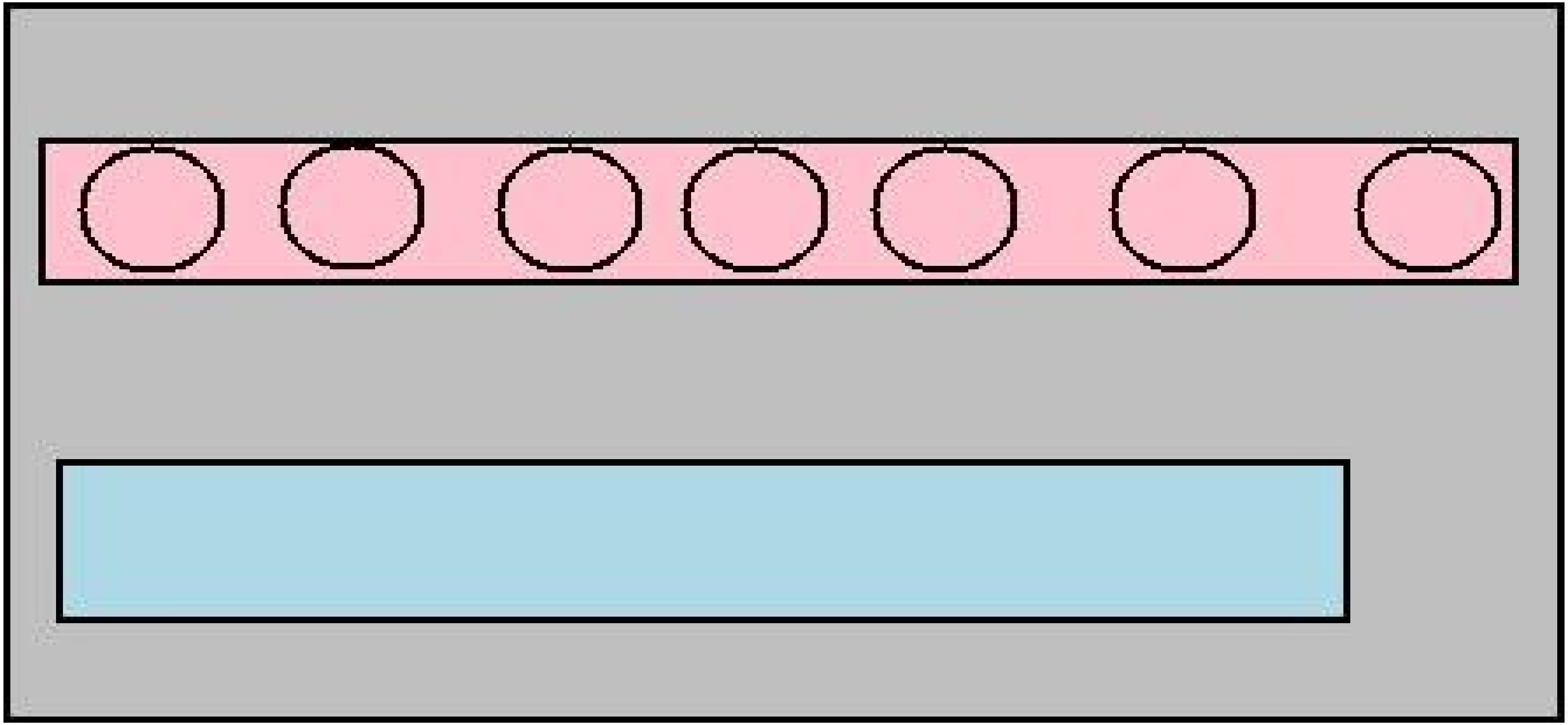
PCOMS - example



PCOMS - template



PCOMS - example



PCOMS – performance testing

- Comparison with the existing AltingBarrier class.
- More in the paper
- Ring of 50 processes connected to their 2 neighbours.
- Time to complete 100 synchronisations

```
WHILE TRUE
  ALT
    SYNC left
      SKIP
    SYNC right
      SKIP
```

PCOMS – performance testing

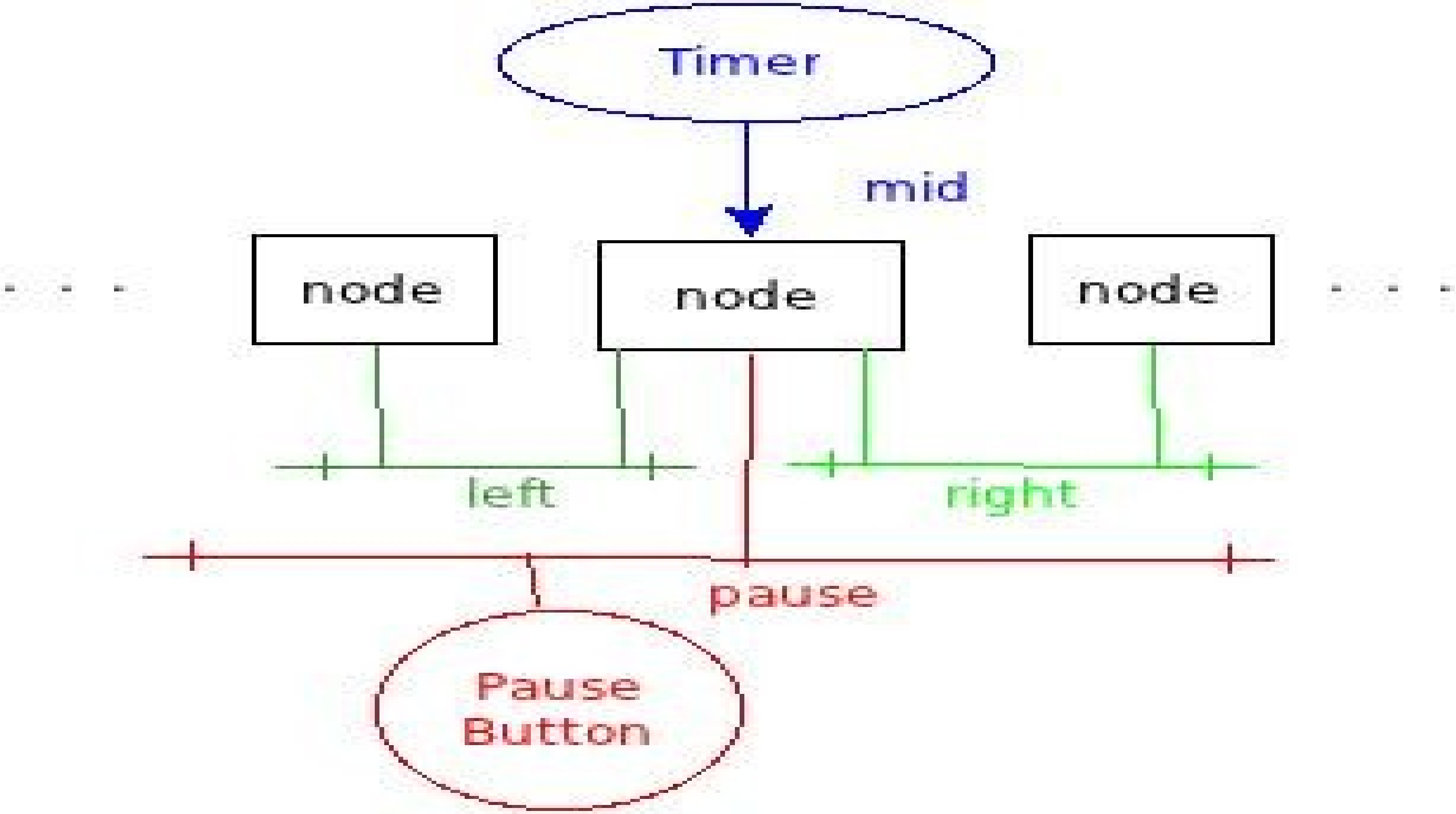
- Using AltingBarrier class finishes in 111ms.
- AltableBarrier class (where all processes are PREPARED) takes 11066 ms.
- AltableBarrier class (where all processes are UNPREPARED) takes 28545 ms.
- in general the AltableBarrier class is 2 orders of magnitude slower than the Alting barrier class.

PCOMS - testing

- Program demonstrates priority
- Compatibility with existing channel guards
- Nested priority

```
WHILE TRUE
  PRI ALT
    SYNC pause
      SYNC pause
    mid ? any
      SKIP
    ALT
      SYNC right
        SKIP
      SYNC left
        SKIP
```

PCOMS - testing



PCOMS – testing

Show demo

PCOMS – future work

- Tidying up, new features, optimisations.
- Distribution over networks.
- Trying out some untested ideas such as fair-alting and 'partial priority'.

PCOMS – sum up

- Prototype algorithm allowing PCOMS.
- (fairly) straightforward to use in JCSP.
- Allows pausing, graceful termination and can be used to underpin output guards and broadcast channels.

PCOMS – any questions

any questions?