Seminar Report: Ordy

Albert Bausili, Noa Yu Ventura, Pol Verdura

March 31, 2023

Upload your report in PDF format.
Use this LaTeX template to format the report.

1 Open questions

Try to answer all the open questions in the assignment. When asked to do so, perform experiments to support your answers.

a) **Are the posts displayed in FIFO, causal, and total order? Justify why.**

FIFO never happens if jitter is greater than 0. See this example with parameters (sleep, jitter, duration) = (100, 1000, 5000) and four processes:

```
P4 RECV( 117) ; From: P4(  36) ; Subject: Re:Re:okcxjbgw
P4 RECV( 118) ; From: P3(  21) ; Subject: Re:mcptlydz
P4 RECV( 119) ; From: P1(  34) ; Subject: Re:Re:jixarshl
P4 RECV( 120) ; From: P1(  29) ; Subject: Re:mdeohbcg
P4 RECV( 121) ; From: P1(  28) ; Subject: wnhvibtq
P4 SEND(  37) ; Subject: Re:wnhvibtq
P4 RECV( 122) ; From: P4(  37) ; Subject: Re:wnhvibtq
P4 RECV( 123) ; From: P2(  33) ; Subject: jwseuphr
```

As we can see here, message 29 from P1 is received by P4 before receiving the message 28 from P1, so FIFO ordering is not respected.

Total order never happens, the parameters do not matter and any execution is valid to prove it:

```
P1 RECV( 117) ; From: P1(  35) ; Subject: Re:Re:hoqcryge
P1 RECV( 118) ; From: P4(  34) ; Subject: ldzsceau
P1 RECV( 119) ; From: P3(  30) ; Subject: zrqaspwe
P1 RECV( 120) ; From: P4(  29) ; Subject: ytpqomju
P1 RECV( 121) ; From: P4(  38) ; Subject: Re:jwseuphr
P1 RECV( 122) ; From: P4(  27) ; Subject: Re:Re:hoqcryge
P1 RECV( 123) ; From: P2(  34) ; Subject: cbwoixkg
```

In this extract of the output of P1 we can see that messages are not ordered, we have number 35 before the 34, the 30, the 29... There is no total ordering.

Casual order might not happen if you receive two related messages, and the last message you receive is the first message. This can happen because there's no message control. This can also be proved by using causal order's definition: it must meet two requirements, and one of them is that it must respect the FIFO order. Since the module basic does not respect FIFO, it does not respect causal either.

b) **Are the posts displayed in FIFO, causal, and total order? Justify why.**

Yes, there is FIFO because if there is causal there is FIFO (by its definition). It is causal because we do not show the messages until the ones that "happened before" are received, which means that the causal order will be maintained. And the total order is not fulfilled, an example of this can be seen in the following extract of an execution:

```
P1:
...
P1 RECV(  69) ; From: P1(  26) ; Subject: gsybtcqv
...
P2:
...
P2 RECV(  69) ; From: P2(  23) ; Subject: dairtxeb
...
P3:
...
P3 RECV(  69) ; From: P3(  24) ; Subject: Re:Re:hjengpil
...
P4:
...
P4 RECV(  69) ; From: P2(  17) ; Subject: Re:hoicsrng
...
```

As it can be seen in this abstract, the same number of received message is not equal in all the processes.

c) **Are the posts displayed in FIFO, causal, and total order? Justify why.**

FIFO-order is not satisfied because, as it's showed in the following lines, from process P1, we receive message 14 before message 15.

```
P2 RECV(  48) ; From: P1(  15) ; Subject: lmzaygnu
P2 RECV(  49) ; From: P4(  16) ; Subject: Re:urahkdve
P2 RECV(  50) ; From: P1(  14) ; Subject: qegfnkyp
```

As FIFO-order is not fulfilled, causal-order won't be satisfied because you can't have causal-order without FIFO-order.

d) **Given a multicast system that is both totally-ordered (using an implementation like ours, which guarantees that the sequence number of any message sent is greater than that of any seen by the sender) and FIFO-ordered, justify whether it is also causally-ordered as a consequence.**

No, it is not causally-ordered as a consequence. Take for example processes A, B and C, and let's say the latency between A and B 150ms, and between A and C 100ms, and between B and C 30ms. If process A sends a message to everyone, it will take longer to arrive to B than to A. After this, C answers to the first message, and arrives to process B. In total, this took 130ms to deliver this message (A->C->B), but then B receives the first message from A (latency is 150ms). Since FIFO is only within the same process, we have a situation where process B received the response from C before receiving the message from A, but all processes respect FIFO ordering. Total ordering is also respected, since everyone receives the response before process B receives the first message.

e) **We have a lot of messages in the system. Derive a theoretical quantification of the number of messages needed to deliver a multicast message (from the sending by a worker to the delivery by all the workers) as a function of the number of workers.**

Having n as the number of workers, we have: $n^2 + 3 \cdot n + 1$

f) **Compare with the basic multicast implementation regarding the number of messages needed.**

Having n as the number of workers, we have: $2 \cdot n + 1$

2 Personal opinion

Give your opinion of the seminar assignment, indicating whether it should be included in next year's course or not.

- **Noa**:
  It helped me a lot to understand what causal order is, since I did not understand it fully in theory class (actually, I realised it thanks to this seminar). What I would improve is that I do not see the point of copying the code from the pdf with the blanks, it was a bit annoying to have to copy all that by hand (some characters are not detected in pdf so copying+pasting it might give errors).

- **Albert**:
  I found this laboratory really interesting and it helped me to fully understand the different types of orderings seen in class as well as knowing some possible implementations. However, I would like to ask for the causal implementation "clues" been directly putted in an erlang file directly as

copying from the pdf was a little bit clunky and to avoid possible mistakes we had to copy it manually.

- **Pol**:
  I think this seminar was more interesting than the last one. Moreover, this lab helps you to understand the differences between multicasts, and it's really cool to see how it's implemented something that we have learned in other courses, in this case multicast.