

Seminar Report: Ordy

Daniel García, Aitor Ardila, Oscar Hernández
April 17, 2016

1 - Introduction

In this seminar we are facing sorting problems using multicast messages between workers. Our job is to implement two algorithms including 'causal' ordering and 'total' ordering.

2 - Work done

In the first part of this seminar we had 'basic' multicast algorithm that sends to the rest of peer multicast messages with some Jitter achieving some interleave order.

(1) BASIC

```
(p1@127.0.0.1)13> ordy:start(basic,100,2000,10000).█
```

p1 window

```
...
(p1@127.0.0.1)13> P1 SEND( 304) ; Subject: Re:Re:ukwvrlx
P1 RECU(1093) ; From: P1( 304) ; Subject: Re:Re:ukwvrlx
..
```

p3 window

```
P3 SEND( 304) ; Subject: Re:sbxdzfuu
P3 RECU(1062) ; From: P1( 262) ; Subject: Re:ukwvrlx
P3 RECU(1063) ; From: P2( 217) ; Subject: apkswxfl
P3 RECU(1064) ; From: P3( 304) ; Subject: Re:sbxdzfuu
```

Secondly, we needed to implement some causal sort algorithm in order to eliminate unsorted replies problems. We did this by using Vector Clocks and tracking VC of each process updating it if was necessary.

(2) CAUSAL

```
(p1@127.0.0.1)16> ordy:start(causal,100,2000,10000).■
```

p1 Window

```
P1 RECV( 372) ; From: P3( 79) ; Subject: Re:afugisjr
P1 SEND( 139) ; Subject: Re:Re:afugisjr
P1 RECV( 373) ; From: P3( 80) ; Subject: Re:uxzvorkj
P1 SEND( 140) ; Subject: Re:Re:uxzvorkj
P1 RECV( 374) ; From: P3( 81) ; Subject: dsqbxyl
P1 RECV( 375) ; From: P3( 82) ; Subject: Re:Re:Re:jxyaic
P1 RECV( 376) ; From: P3( 83) ; Subject: mtzjpowb
P1 RECV( 377) ; From: P3( 84) ; Subject: Re:nzoiyqhs
P1 RECV( 378) ; From: P3( 85) ; Subject: rgjztyhw
P1 RECV( 379) ; From: P3( 86) ; Subject: tewgfjks
P1 RECV( 380) ; From: P1( 139) ; Subject: Re:Re:afugisjr
```

Finally, we implemented total ordering which forces all messages to be received in the same order. We achieved this by using proposal messages.

(3) TOTAL

```
(p1@127.0.0.1)13> stopped
(p1@127.0.0.1)13> ordy:start(total,100,1000,5000).
```

p1 window

```
P1 RECV( 165) ; From: P3( 42) ; Subject: vufduoza
P1 RECV( 166) ; From: P1( 33) ; Subject: gwucpexj
P1 RECV( 167) ; From: P2( 48) ; Subject: tsqfrnuv
P1 SEND( 86) ; Subject: Re:tsqfrnuv
P1 RECV( 168) ; From: P2( 44) ; Subject: ytncefri
P1 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P1 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P1 SEND( 87) ; Subject: Re:ntgrjoyf
P1 RECV( 171) ; From: P3( 52) ; Subject: jgupnhur
```

p2 window

```
P2 RECV( 165) ; From: P3( 42) ; Subject: vufduoza
P2 RECV( 166) ; From: P1( 33) ; Subject: gwucpexj
P2 RECV( 167) ; From: P2( 48) ; Subject: tsqfrnuv
P2 RECV( 168) ; From: P2( 44) ; Subject: ytncefri
P2 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P2 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
```

p3 window

```
P3 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P3 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P3 SEND( 104) ; Subject: hluxrpi
P3 SEND( 105) ; Subject: jdrvfsl
P3 SEND( 106) ; Subject: cxtmplhs
```

p4 window

```

P4 RECV( 167) ; From: P2( 48) ; Subject: tsqFrnuv
P4 RECV( 168) ; From: P2( 44) ; Subject: ytncefri
P4 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkun
P4 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P4 RECV( 171) ; From: P3( 52) ; Subject: jgupnhvr
P4 SEND( 96) ; Subject: Re:jgupnhvr

```

3 - Experiments

3.1 - Basic algorithm

Basic algorithm starts to unsort messages while Jitter increases as you can see in the figures below:

BASIC 100ms Sleep, 0 Jitter, 3 seconds executing

```

(p1@127.0.0.1)17> ordy:start(basic,100,0,3000).
(p1@127.0.0.1)18> P1 SEND( 1) ; Subject: enpuxdkg
P1 RECV( 1) ; From: P1( 1) ; Subject: enpuxdkg
P1 RECV( 2) ; From: P3( 1) ; Subject: qijwhzlp
P1 SEND( 2) ; Subject: efprthcl
P1 RECV( 3) ; From: P4( 1) ; Subject: Re:qijwhzlp
P1 RECV( 4) ; From: P4( 2) ; Subject: nskfamro
P1 RECV( 5) ; From: P2( 1) ; Subject: Re:Re:qijwhzlp
P1 RECV( 6) ; From: P2( 2) ; Subject: qdksgcpw
P1 RECV( 7) ; From: P1( 2) ; Subject: efprthcl

```

As you can see, everything is sorted in this environment

BASIC 300ms Sleep, 3 seconds Jitter, 20 seconds executing

```

(p1@127.0.0.1)23> ordy:start(basic,300,3000,20000).
P1 RECV( 874) ; From: P3( 219) ; Subject: Re:rubzardl
P1 RECV( 875) ; From: P4( 213) ; Subject: Re:Re:ycarvepl
P1 SEND( 231) ; Subject: Re:Re:Re:ycarvepl
P1 RECV( 876) ; From: P1( 231) ; Subject: Re:Re:Re:ycarvepl
P1 SEND( 232) ; Subject: gaehbmzt
P1 RECV( 877) ; From: P1( 232) ; Subject: gaehbmzt
P1 RECV( 878) ; From: P2( 217) ; Subject: xdgwitck
P1 RECV( 879) ; From: P2( 239) ; Subject: Re:Re:gtcwpuea
P1 RECV( 880) ; From: P2( 209) ; Subject: Re:heoucpsz
P1 RECV( 881) ; From: P3( 204) ; Subject: Re:ycarvepl
P1 SEND( 233) ; Subject: Re:Re:ycarvepl
P1 RECV( 882) ; From: P1( 233) ; Subject: Re:Re:ycarvepl

```

As you can see, we are receiving replies from a message before the original message.

3.2 - Causal algorithm

CAUSAL 100ms Sleep, 0 Jitter, 3 seconds executing

```
-----  
(p1@127.0.0.1)23> ordy:start(causal,100,0,3000).  
P1 RECV( 103) ; From: P3( 26) ; Subject: edzapjur  
P1 RECV( 104) ; From: P3( 27) ; Subject: rcywjhzt  
P1 SEND( 28) ; Subject: Re:rcywjhzt  
P1 RECV( 105) ; From: P3( 28) ; Subject: Re:Re:Re:upbcfdjq  
P1 RECV( 106) ; From: P3( 29) ; Subject: Re:Re:Re:ocgmftaq  
P1 RECV( 107) ; From: P2( 29) ; Subject: Re:edzapjur  
P1 RECV( 108) ; From: P4( 25) ; Subject: Re:Re:Re:upbcfdjq  
P1 RECV( 109) ; From: P4( 26) ; Subject: aoxkzicl  
P1 RECV( 110) ; From: P4( 27) ; Subject: Re:edzapjur  
P1 RECV( 111) ; From: P2( 30) ; Subject: tzmrioaf  
P1 RECV( 112) ; From: P2( 31) ; Subject: Re:Re:edzapjur  
P1 SEND( 29) ; Subject: Re:Re:Re:edzapjur  
P1 RECV( 113) ; From: P2( 32) ; Subject: liphvfzm  
P1 RECV( 114) ; From: P4( 28) ; Subject: cugktibd  
P1 SEND( 30) ; Subject: Re:cugktibd  
P1 RECV( 115) ; From: P4( 29) ; Subject: Re:Re:Re:edzapjur
```

Messages still have causal order in this scenario

CAUSAL 300ms Sleep, 3 seconds Jitter, 20 seconds executing

```
(p1@127.0.0.1)25> ordy:start(causal,300,3000,20000).█  
417 P1 RECV( 456) ; From: P3( 100) ; Subject: Re:Re:vtaglkme  
418 P1 RECV( 457) ; From: P3( 101) ; Subject: bdcwgqnp  
419 P1 RECV( 458) ; From: P3( 102) ; Subject: Re:chopwvi  
420 P1 RECV( 459) ; From: P3( 103) ; Subject: riaodtbs  
421 P1 RECV( 460) ; From: P3( 104) ; Subject: iuyvlwet  
422 P1 RECV( 461) ; From: P3( 105) ; Subject: Re:yhvokedf  
423 P1 SEND( 144) ; Subject: Re:Re:yhvokedf  
424 P1 RECV( 462) ; From: P3( 106) ; Subject: Re:Re:jodzgvul  
425 P1 RECV( 463) ; From: P3( 107) ; Subject: Re:Re:vtaglkme  
426 P1 SEND( 145) ; Subject: Re:Re:Re:vtaglkme  
427 P1 RECV( 464) ; From: P1( 144) ; Subject: Re:Re:yhvokedf  
428 P1 RECV( 465) ; From: P4( 111) ; Subject: Re:agtdhez  
429 P1 RECV( 466) ; From: P4( 112) ; Subject: nbvpgqiw  
430 P1 RECV( 467) ; From: P1( 145) ; Subject: Re:Re:Re:vtaglkme
```

Messages have now causal order in this scenario thanks to vector clocks, even if jitter is extremely high.

3.3 - Total algorithm

TOTAL 100ms Sleep, 0 Jitter, 3 seconds executing

```
(p1@127.0.0.1)15> P1 RECV( 461) ; From: P3( 123) ; Subject: uzpčxóvs  
(p1@127.0.0.1)15> P1 SEND( 111) ; Subject: Re:uzpcxovs  
(p1@127.0.0.1)15> P1 RECV( 462) ; From: P1( 111) ; Subject: Re:uzpcxovs  
(p1@127.0.0.1)15> P1 RECV( 463) ; From: P2( 122) ; Subject: xbinjlgk  
(p1@127.0.0.1)15> P1 RECV( 464) ; From: P4( 108) ; Subject: lwjadohz  
(p1@127.0.0.1)15> P1 RECV( 465) ; From: P3( 124) ; Subject: Re:lwjadohz  
(p1@127.0.0.1)15> P1 RECV( 466) ; From: P2( 123) ; Subject: Re:Re:lwjadohz  
(p1@127.0.0.1)15> P1 RECV( 467) ; From: P4( 109) ; Subject: Re:Re:Re:lwjadohz  
(p1@127.0.0.1)15> P1 RECV( 468) ; From: P3( 125) ; Subject: bcmvihso  
(p1@127.0.0.1)15> P1 RECV( 469) ; From: P2( 124) ; Subject: becfwlsq  
(p1@127.0.0.1)15> P1 SEND( 112) ; Subject: Re:becfwlsq  
(p1@127.0.0.1)15> P1 RECV( 470) ; From: P1( 112) ; Subject: Re:becfwlsq  
(p1@127.0.0.1)15> P1 SEND( 113) ; Subject: ealnxdis  
(p1@127.0.0.1)15> P1 RECV( 471) ; From: P4( 110) ; Subject: Re:Re:becfwlsq
```

As we can see, without jitter all messages are received ordered (same as with the other algorithms).

TOTAL 300ms Sleep, 3 seconds Jitter, 20 seconds executing

The screenshot shows four Erlang console windows, each displaying a log of messages. The messages are ordered by sequence number across all processes, demonstrating total ordering. The logs are as follows:

Window 1 (Top Left):

```
P1 RECV( 150) ; From: P4( 35) ; Subject: dgyssuock
P1 RECV( 151) ; From: P2( 42) ; Subject: bgpfczts
P1 RECV( 152) ; From: P1( 27) ; Subject: xgshkure
P1 RECV( 153) ; From: P3( 36) ; Subject: Re:zuebgycu
P1 RECV( 154) ; From: P1( 31) ; Subject: vhmteaju
P1 RECV( 155) ; From: P1( 29) ; Subject: hmlckcw
P1 RECV( 156) ; From: P1( 28) ; Subject: lxdvgjqu
P1 RECV( 157) ; From: P1( 32) ; Subject: zrtuxkfg
P1 SEND( 81) ; Subject: jdhepcux
P1 SEND( 82) ; Subject: zedslovh
P1 SEND( 83) ; Subject: kwrfnd
P1 RECV( 158) ; From: P2( 43) ; Subject: fmpagibn
P1 RECV( 159) ; From: P3( 37) ; Subject: Re:uhnafoqi
P1 RECV( 160) ; From: P4( 41) ; Subject: adirhevu
P1 RECV( 161) ; From: P3( 53) ; Subject: empclngj
P1 SEND( 84) ; Subject: ojziahir
P1 SEND( 85) ; Subject: gypdfus
P1 RECV( 162) ; From: P4( 40) ; Subject: najovngf
P1 RECV( 163) ; From: P1( 50) ; Subject: Re:hgxtnonc
P1 RECV( 164) ; From: P4( 42) ; Subject: bhorciak
P1 RECV( 165) ; From: P3( 42) ; Subject: vufduoza
P1 RECV( 166) ; From: P1( 33) ; Subject: gwucpej
P1 RECV( 167) ; From: P2( 48) ; Subject: tsqfrmw
P1 SEND( 86) ; Subject: Re:tsqfrmw
P1 RECV( 168) ; From: P2( 44) ; Subject: ytmcefri
P1 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P1 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P1 SEND( 87) ; Subject: Re:ntgrjoyf
P1 RECV( 171) ; From: P3( 52) ; Subject: jgupnhur
```

Window 2 (Top Right):

```
P2 RECV( 148) ; From: P4( 45) ; Subject: Re:upshkjev
P2 RECV( 149) ; From: P4( 49) ; Subject: Re:atshbeux
P2 RECV( 150) ; From: P4( 35) ; Subject: dgyssuock
P2 RECV( 151) ; From: P2( 42) ; Subject: bgpfczts
P2 RECV( 152) ; From: P1( 27) ; Subject: xgshkure
P2 RECV( 153) ; From: P3( 36) ; Subject: Re:zuebgycu
P2 RECV( 154) ; From: P1( 31) ; Subject: vhmteaju
P2 RECV( 155) ; From: P1( 29) ; Subject: hmlckcw
P2 RECV( 156) ; From: P1( 28) ; Subject: lxdvgjqu
P2 RECV( 157) ; From: P1( 32) ; Subject: zrtuxkfg
P2 RECV( 158) ; From: P2( 43) ; Subject: fmpagibn
P2 RECV( 159) ; From: P3( 37) ; Subject: Re:uhnafoqi
P2 RECV( 160) ; From: P4( 41) ; Subject: adirhevu
P2 SEND( 80) ; Subject: qtbzrjcu
P2 RECV( 161) ; From: P3( 53) ; Subject: empclngj
P2 SEND( 81) ; Subject: eupkzqf
P2 SEND( 82) ; Subject: qbowikte
P2 RECV( 162) ; From: P4( 40) ; Subject: najovngf
P2 SEND( 83) ; Subject: Re:najovngf
P2 RECV( 163) ; From: P1( 50) ; Subject: Re:hgxtnonc
P2 SEND( 84) ; Subject: Re:hgxtnonc
P2 SEND( 85) ; Subject: nxdteliz
P2 RECV( 164) ; From: P4( 42) ; Subject: bhorciak
P2 RECV( 165) ; From: P3( 42) ; Subject: vufduoza
P2 RECV( 166) ; From: P1( 33) ; Subject: gwucpej
P2 RECV( 167) ; From: P2( 48) ; Subject: tsqfrmw
P2 RECV( 168) ; From: P2( 44) ; Subject: ytmcefri
P2 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P2 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
```

Window 3 (Bottom Left):

```
P3 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P3 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P3 SEND( 104) ; Subject: hluxrupi
P3 SEND( 105) ; Subject: jdrufsal
P3 SEND( 106) ; Subject: cxtmplhs
P3 RECV( 171) ; From: P3( 52) ; Subject: jgupnhur
P3 RECV( 172) ; From: P4( 60) ; Subject: Re:luxofbpg
P3 RECV( 173) ; From: P2( 56) ; Subject: Re:lecbvpxr
P3 RECV( 174) ; From: P2( 52) ; Subject: Re:uefizlqx
P3 RECV( 175) ; From: P4( 52) ; Subject: nqubajjt
P3 RECV( 176) ; From: P4( 48) ; Subject: Re:hbugarxs
P3 SEND( 107) ; Subject: Re:Re:hbugarxs
P3 RECV( 177) ; From: P4( 64) ; Subject: grtbkjna
P3 SEND( 108) ; Subject: Re:grtbkjna
P3 RECV( 178) ; From: P2( 47) ; Subject: nnagfcuy
P3 RECV( 179) ; From: P2( 46) ; Subject: yfauppid
P3 RECV( 180) ; From: P4( 43) ; Subject: voeghtkj
P3 RECV( 181) ; From: P3( 43) ; Subject: Re:upshkjev
P3 RECV( 182) ; From: P3( 44) ; Subject: Re:tnvabazl
P3 SEND( 109) ; Subject: bfuhzngl
P3 SEND( 110) ; Subject: nsjudqvh
P3 SEND( 111) ; Subject: dunsnpgh
P3 RECV( 183) ; From: P1( 47) ; Subject: nlagirxd
P3 SEND( 112) ; Subject: Re:nlagirxd
P3 RECV( 184) ; From: P1( 39) ; Subject: lmvockht
P3 RECV( 185) ; From: P1( 34) ; Subject: Re:blgvaint
P3 RECV( 186) ; From: P4( 44) ; Subject: Re:olubjufy
P3 SEND( 113) ; Subject: Re:Re:olubjufy
P3 RECV( 187) ; From: P4( 47) ; Subject: jiwcbssug
```

Window 4 (Bottom Right):

```
P4 RECV( 164) ; From: P4( 42) ; Subject: bhorciak
P4 RECV( 165) ; From: P3( 42) ; Subject: vufduoza
P4 RECV( 166) ; From: P1( 33) ; Subject: gwucpej
P4 RECV( 167) ; From: P2( 48) ; Subject: tsqfrmw
P4 RECV( 168) ; From: P2( 44) ; Subject: ytmcefri
P4 RECV( 169) ; From: P2( 60) ; Subject: Re:blsdgkvn
P4 RECV( 170) ; From: P3( 40) ; Subject: ntgrjoyf
P4 RECV( 171) ; From: P3( 52) ; Subject: jgupnhur
P4 SEND( 96) ; Subject: Re:jgupnhur
P4 RECV( 172) ; From: P4( 60) ; Subject: Re:luxofbpg
P4 RECV( 173) ; From: P2( 56) ; Subject: Re:lecbvpxr
P4 RECV( 174) ; From: P2( 52) ; Subject: Re:uefizlqx
P4 RECV( 175) ; From: P4( 52) ; Subject: nqubajjt
P4 RECV( 176) ; From: P4( 48) ; Subject: Re:hbugarxs
P4 RECV( 177) ; From: P4( 64) ; Subject: grtbkjna
P4 RECV( 178) ; From: P2( 47) ; Subject: nnagfcuy
P4 SEND( 97) ; Subject: Re:nnagfcuy
P4 RECV( 179) ; From: P2( 46) ; Subject: yfauppid
P4 RECV( 180) ; From: P4( 43) ; Subject: voeghtkj
P4 RECV( 181) ; From: P3( 43) ; Subject: Re:upshkjev
P4 RECV( 182) ; From: P3( 44) ; Subject: Re:tnvabazl
P4 SEND( 98) ; Subject: btprnzxi
P4 RECV( 183) ; From: P1( 47) ; Subject: nlagirxd
P4 SEND( 99) ; Subject: Re:nlagirxd
P4 RECV( 184) ; From: P1( 39) ; Subject: lmvockht
P4 RECV( 185) ; From: P1( 34) ; Subject: Re:blgvaint
P4 RECV( 186) ; From: P4( 44) ; Subject: Re:olubjufy
P4 RECV( 187) ; From: P4( 47) ; Subject: jiwcbssug
P4 SEND( 100) ; Subject: rjquzvos
```

As we can see, message are received ordered (we can check this by the sequence number. In all 4 processes, the messages received are totally ordered.

4 - Open questions

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

- For the BASIC implementation we can reach FIFO order if we set a small jitter. But when the jitter is high the messages will be delivered unsorted.
- For the CAUSAL implementation they are ordered as causal. With jitter, messages are delivered unsorted but reply messages come after original messages.
- For the TOTAL implementation they are ordered totally, even with jitter messages are displayed in the same order for all nodes.

5 - Personal opinion

In our opinion, this seminar is a good way to learn programming with Erlang sort algorithms in a distributed environment. We can force the system to sort the received messages even when there's a high jitter. The output messages help to see clearly whether or not the implemented algorithm is working. We would recommend to use this seminar for future courses.