

Seminar Report: Ordy

Marc Ortiz
Omais Iqbal
V ctor Gonz lez

April 5, 2017

1 Introduction

Aquesta practica consisteix en implementar versions d'un multicast reliable service que suporta diferents ordenacions de missatges.

2 Work done

Els fitxers font es troben a la carpeta codes i les imatges la carpeta figures per si no es veuen prou b .

3 Experiments

3.1 Basic multicast

En la primera implementaci  b sica cada vegada que un node rep un missatge, aquest s'entrega a tots els altres nodes un per un, sense cap ordre i sense cap tipus de prioritat.

1. *Experiment amb Sleep time molt m s gran que Jitter time*

Hip tesis: En aquest cas hi haur  molta menys probabilitat de que en un moment donat un node faci un multicast, el temps de Jitter d'un missatge sigui mes gran que la suma de Sleep+Jitter+execucio del programa del seguent multicast, tot aix  sense tenir en compte les respostes dels altres missatges. Si les tingu ssim en compte augmentaria la probabilitat de desordre FIFO.

Experimentació:

The figure displays four terminal windows, each showing a log of network traffic for a process named 'SDX: beam.smp'. The logs are organized into a grid. Each window shows a sequence of messages, including 'SEND' and 'RECV' (receive) operations, with details about the source and destination nodes (P1, P2, P3, P4) and the subject of the message. Red arrows in the top-right and bottom-right windows point to specific 'RECV' messages: 'Re:anfsqjxq' and 'Re:dxuqifc' respectively, highlighting a deviation from the expected FIFO order.

Figure 1: Sleep time = 1000, Jitter time = 100

Conclusions: Com hem dit es mes probable que no es respecti l'ordre FIFO, tot i així en l'experiment s'ha trobat que un cas després d'uns quants multicasts ordenats en que no s'ha respectat l'ordre FIFO (marcat amb vermell en la figure 1). En el node emissor apareix "Recv.." en ordre ja que a ell no se li aplica el Jitter, ens hem de fixar en el node 3 que es troba a la finestra superior dreta. Es node 3 rep abans el missatge "anfsqjxq" que el "dxuqifc" encara que el node 4 els envia en ordre invers tal i com es pot veure en la finestra inferior dreta.

2. Experiment amb temps de Jitter molt mes gran que temps de Sleep

Hipòtesis: En aquesta cas es bastant probable que no es respecti l'ordre FIFO en l'interval de temps en que es fa multicast. Si anem decremantant l'Sleep time, el conjunt de missatges que intenten ser enviats mentres un esta esperant (en el Jitter) es molt elevat, es facil intuir doncs que com mes gran sigui aquest conjunt mes facil es trobar un missatge que s'envii abans i no respectar l'ordre FIFO.

Experimentació:

The figure displays four terminal windows showing network traffic logs. Red arrows point to specific lines in the logs, indicating message order anomalies. The logs show a sequence of SEND and RECV messages between nodes P1, P2, P3, and P4. The anomalies are marked with red arrows and a red question mark.

```

[hi] [kernel-poll:false]
Eshell V7.3.1.2 (abort with ^G)
(p2b127.0.0.1)> ordy:start(basic, 100, 1000, 350).
From
(p1b127.0.0.1)> P1 RECV( 1) ; From: P4( 1) ; Subject: yocfeipv
P1 SEND( 1) ; Subject: wfsusknx
P1 RECV( 2) ; From: P1( 1) ; Subject: wfsusknx
P1 SEND( 2) ; Subject: odgsipel
P1 RECV( 3) ; From: P1( 2) ; Subject: odgsipel
P1 SEND( 3) ; Subject: nturojyb
P1 RECV( 4) ; From: P1( 3) ; Subject: nturojyb
P1 SEND( 5) ; From: P1( 3) ; Subject: Re:facjwdk
P1 RECV( 4) ; Subject: ghzanvao
P1 SEND( 6) ; From: P1( 4) ; Subject: ghzanvao
P1 RECV( 7) ; From: P4( 2) ; Subject: dkqpenl
P1 RECV( 8) ; From: P2( 6) ; Subject: dmcorgu
P1 SEND( 5) ; Subject: ewsqztn
P1 RECV( 9) ; From: P1( 5) ; Subject: ewsqztn
P1 SEND( 6) ; Subject: lbnhwkf
P1 RECV( 10) ; From: P1( 6) ; Subject: lbnhwkf
P1 SEND( 7) ; Subject: xflayabp
P1 RECV( 11) ; From: P1( 7) ; Subject: xflayabp
P1 RECV( 12) ; From: P2( 2) ; Subject: anjrykgl
P1 RECV( 13) ; From: P3( 4) ; Subject: vetsofn
Stopping...
Stopped.
(p1b127.0.0.1)>

[hi] [kernel-poll:false]
Eshell V7.3.1.2 (abort with ^G)
(p3b127.0.0.1)> P3 SEND( 1) ; Subject: zafuinjk
P3 RECV( 1) ; From: P3( 1) ; Subject: zafuinjk
P3 SEND( 2) ; Subject: vzgiwayl
P3 RECV( 2) ; From: P3( 2) ; Subject: vzgiwayl
P3 SEND( 3) ; From: P2( 1) ; Subject: fncjwdk
P3 SEND( 3) ; Subject: Re:facjwdk
P3 RECV( 4) ; From: P3( 3) ; Subject: Re:facjwdk
P3 SEND( 4) ; Subject: vetsofn
P3 RECV( 5) ; From: P3( 4) ; Subject: vetsofn
P3 SEND( 5) ; Subject: cfaxnlry
P3 RECV( 6) ; From: P3( 5) ; Subject: cfaxnlry
P3 RECV( 7) ; From: P4( 1) ; Subject: yocfeipv
P3 SEND( 6) ; Subject: Re:yocfeipv
P3 RECV( 8) ; From: P3( 6) ; Subject: Re:yocfeipv
P3 SEND( 7) ; Subject: ltohzaix
P3 RECV( 9) ; From: P3( 7) ; Subject: ltohzaix
P3 RECV( 10) ; From: P1( 2) ; Subject: odgsipel
[]

[hi] [kernel-poll:false]
Eshell V7.3.1.2 (abort with ^G)
(p4b127.0.0.1)> P4 SEND( 1) ; Subject: yocfeipv
P4 RECV( 1) ; From: P4( 1) ; Subject: yocfeipv
P4 SEND( 2) ; Subject: dkqpenl
P4 RECV( 2) ; From: P4( 2) ; Subject: dkqpenl
P4 SEND( 3) ; Subject: vxuhtns
P4 RECV( 3) ; From: P4( 3) ; Subject: vxuhtns
P4 SEND( 4) ; From: P4( 4) ; Subject: qtytskfc
P4 RECV( 4) ; From: P4( 4) ; Subject: qtytskfc
P4 RECV( 5) ; From: P1( 1) ; Subject: wfsusknx
P4 SEND( 5) ; Subject: uypombfq
P4 RECV( 6) ; From: P4( 5) ; Subject: uypombfq
P4 SEND( 6) ; Subject: wqzvrjc
P4 RECV( 7) ; From: P4( 6) ; Subject: wqzvrjc
P4 SEND( 7) ; Subject: bktwdjrf
P4 RECV( 8) ; From: P4( 7) ; Subject: bktwdjrf
P4 SEND( 8) ; Subject: rytwlwbk
P4 RECV( 9) ; From: P4( 8) ; Subject: rytwlwbk
P4 RECV( 10) ; From: P1( 2) ; Subject: odgsipel
P4 SEND( 9) ; Subject: Re:odgsipel
P4 RECV( 11) ; From: P4( 9) ; Subject: Re:odgsipel
P4 RECV( 12) ; From: P2( 1) ; Subject: fncjwdk
P4 RECV( 13) ; From: P3( 2) ; Subject: vzgiwayl
P4 SEND( 10) ; Subject: Re:vzgiwayl
P4 RECV( 14) ; From: P4( 10) ; Subject: Re:vzgiwayl
[]

```

Figure 2: Sleep time = 100, Jitter time = 1000

Conclusions: Podem veure amb la experimentació que s'ha respectat menys l'ordre FIFO en l'enviament de missatges. Les fletxes vermelles ens indiquen desordres que s'han detectat. També cal dir que hi ha missatges que encara no han arribat als emissors degut al elevat temps de Jitter, per això podem veure que el node 4 té molt més contingut (ha fet multicast de molts missatges) i els altres en tenen menys perquè entre altres coses encara no han arribat tots els missatges del node 4. Gràcies a que el node 4 ha efectuat molts multicasts (temps sleep petit) i que el Jitter era elevat hem pogut distingir fàcilment els desordres.

3.2 Causal order multicast

En la segona versió modificada s'implementa un causal order multicast amb vector clocks on es garanteix que en un node la pregunta (missatge original) sempre arriba abans que la resposta.

1. Experiment amb sleep time molt més gran que jitter time

Hipotesis: amb aquesta versio s'espera que es respecti l'ordre FIFO, ja que abans d'enviar un missatge es comproven dues condicions: el missatge que es rep es el següent que s'espera i que ja s'han rebut tots els missatges anteriors del proces que ha enviat el missatge. I que les respostes no es rebran abans que els missatges originals.

Experimentació:

The figure displays four terminal windows showing network traffic logs for a causal order multicast experiment. The logs are organized into four quadrants, each representing a different node or process. The top-left window shows messages from p1 to p3, p1 to p4, and p1 to p2. The top-right window shows messages from p3 to p1, p3 to p4, and p3 to p2. The bottom-left window shows messages from p2 to p1, p2 to p3, and p2 to p4. The bottom-right window shows messages from p4 to p1, p4 to p3, and p4 to p2. A red arrow in the top-left window points to a message from p3 to p1, indicating a specific point in the sequence. The logs show a sequence of messages between nodes p1, p2, p3, and p4, with each message containing a subject and a vector clock. The messages are ordered chronologically, showing the flow of information between the nodes.

Figure 3: Sleep time = 1000, Jitter time = 100

Conclusions: La primera característica que podem observar es que sí es respecta l'ordre FIFO, cada procés rep els missatges enviats per l'emissor en l'ordre en que ell els ha enviat. Podem veure també que es manté la causalitat veient les respostes als missatges, no hi ha cap resposta abans de la pregunta (missatge original). Finalment podem observar que l'ordre total no es manté. A l'experiment s'ha assenyalat amb una fletxa un possible desordre (tot i que es faclment observable a partir de les primeres 4 instancies).

2. Experiment amb Sleep time molt mes petit que jitter time

Hipotesis: L'augment del jitter time no hauria s'afectar a l'ordre dels missatges sinó que nomes pot provocar un retard mes gran per que els missatges no son entregats fins que els anteriors ja ho han sigut.

Experimentació:

```
* exception error: bad argument
  in function register/2
    called as register(ordy,<0.135,0>)
  in call from ordystart/4 (ordy.erl line 7)
(p10127.0.0.1)8- P1 SEND( 1) : Subject: syvacptj
(p10127.0.0.1)8- P1 RECV( 1) : From: P1( 1) ; Subject: syvacptj
(p10127.0.0.1)8- P1 SEND( 2) : Subject: ukhtxejl
(p10127.0.0.1)8- P1 RECV( 2) : From: P1( 2) ; Subject: ukhtxejl
(p10127.0.0.1)8- P1 SEND( 3) : Subject: gwpfxkhy
(p10127.0.0.1)8- P1 RECV( 3) : From: P1( 3) ; Subject: gwpfxkhy
(p10127.0.0.1)8- P1 SEND( 4) : Subject: smatfwdd
(p10127.0.0.1)8- P1 RECV( 4) : From: P1( 4) ; Subject: smatfwdd
(p10127.0.0.1)8- P1 SEND( 5) : Subject: aripmtvz
(p10127.0.0.1)8- P1 RECV( 5) : From: P1( 5) ; Subject: aripmtvz
(p10127.0.0.1)8- P1 SEND( 6) : Subject: dqcmwury
(p10127.0.0.1)8- P1 RECV( 6) : From: P1( 6) ; Subject: dqcmwury
(p10127.0.0.1)8- P1 RECV( 7) : From: P2( 1) ; Subject: yjdaegiwi
(p10127.0.0.1)8- P1 RECV( 8) : From: P2( 2) ; Subject: onhybkcg
(p10127.0.0.1)8- P1 SEND( 7) : Subject: symweztu
(p10127.0.0.1)8- P1 RECV( 9) : From: P1( 7) ; Subject: symweztu
(p10127.0.0.1)8- P1 SEND( 8) : Subject: vckthadv
(p20127.0.0.1)15- P2 SEND( 49) : Subject: gpccajkye
(p20127.0.0.1)15-
Eshell V8.2 (abort with ^C)
C:\Users\Marc\Documents\SDX\Erlang\ordy-source-files\erl -name p20127.0.0.1
(p20127.0.0.1)15- P2 SEND( 1) : Subject: yjdaegiwi
(p20127.0.0.1)15- P2 RECV( 1) : From: P2( 1) ; Subject: yjdaegiwi
(p20127.0.0.1)15- P2 SEND( 2) : Subject: onhybkcg
(p20127.0.0.1)15- P2 RECV( 2) : From: P2( 2) ; Subject: onhybkcg
(p20127.0.0.1)15- P2 SEND( 3) : Subject: fbjpxaul
(p20127.0.0.1)15- P2 RECV( 3) : Subject: tbxfdekj
(p20127.0.0.1)15- P2 RECV( 4) : From: P2( 3) ; Subject: tbxfdekj
(p20127.0.0.1)15- P2 SEND( 4) : Subject: stjiclx
(p20127.0.0.1)15- P2 RECV( 5) : From: P2( 4) ; Subject: stjiclx
(p20127.0.0.1)15- P2 SEND( 5) : Subject: kpqalrgd
(p20127.0.0.1)15- P2 RECV( 6) : From: P2( 5) ; Subject: kpqalrgd
(p20127.0.0.1)15- P2 SEND( 6) : Subject: gkrsnbtu
(p20127.0.0.1)15- P2 RECV( 7) : From: P2( 6) ; Subject: gkrsnbtu
(p20127.0.0.1)15- P2 SEND( 7) : Subject: mdkaxyqs
(p20127.0.0.1)15- P2 RECV( 8) : From: P2( 7) ; Subject: mdkaxyqs
(p20127.0.0.1)15- P2 SEND( 8) : Subject: nvxjiucr
(p20127.0.0.1)15- P2 RECV( 9) : From: P2( 8) ; Subject: nvxjiucr
C:\Users\Marc\Documents\SDX\Erlang\ordy-source-files\erl -name p30127.0.0.1
Eshell V8.2 (abort with ^C)
(p30127.0.0.1)15- P3 SEND( 1) : Subject: yfoquxtd
(p30127.0.0.1)15- P3 RECV( 1) : From: P3( 1) ; Subject: yfoquxtd
(p30127.0.0.1)15- P3 RECV( 2) : From: P1( 1) ; Subject: syvacptj
(p30127.0.0.1)15- P3 SEND( 2) : Subject: vdkxglci
(p30127.0.0.1)15- P3 RECV( 3) : From: P3( 2) ; Subject: vdkxglci
(p30127.0.0.1)15- P3 SEND( 3) : Subject: tlrgazod
(p30127.0.0.1)15- P3 RECV( 4) : From: P3( 3) ; Subject: tlrgazod
(p30127.0.0.1)15- P3 RECV( 5) : From: P2( 1) ; Subject: yjdaegiwi
(p30127.0.0.1)15- P3 RECV( 6) : From: P4( 1) ; Subject: fyjpxaul
(p30127.0.0.1)15- P3 SEND( 4) : Subject: rjmdxhsq
(p30127.0.0.1)15- P3 RECV( 7) : From: P3( 4) ; Subject: rjmdxhsq
(p30127.0.0.1)15- P3 RECV( 8) : Subject: bxrkghet
(p30127.0.0.1)15- P3 RECV( 9) : From: P3( 5) ; Subject: bxrkghet
(p30127.0.0.1)15- P3 SEND( 6) : Subject: wzkmpnif
(p30127.0.0.1)15- P3 RECV( 10) : From: P3( 6) ; Subject: wzkmpnif
(p30127.0.0.1)15- P3 RECV( 11) : Subject: fhbwnplk
(p30127.0.0.1)15- P3 RECV( 12) : From: P3( 7) ; Subject: fhbwnplk
(p30127.0.0.1)15- P3 SEND( 8) : Subject: ufihgmo
(p30127.0.0.1)15- P3 RECV( 13) : From: P3( 8) ; Subject: ufihgmo
C:\Users\Marc\Documents\SDX\Erlang\ordy-source-files\erl -name p40127.0.0.1
Eshell V8.2 (abort with ^C)
(p40127.0.0.1)15- P4 SEND( 1) : Subject: fyjpxaul
(p40127.0.0.1)15- P4 RECV( 1) : From: P4( 1) ; Subject: fyjpxaul
(p40127.0.0.1)15- P4 RECV( 2) : From: P1( 1) ; Subject: syvacptj
(p40127.0.0.1)15- P4 SEND( 2) : Subject: omwqcnb
(p40127.0.0.1)15- P4 RECV( 3) : From: P4( 2) ; Subject: omwqcnb
(p40127.0.0.1)15- P4 RECV( 4) : From: P2( 1) ; Subject: yjdaegiwi
(p40127.0.0.1)15- P4 SEND( 3) : Subject: jzibwgay
(p40127.0.0.1)15- P4 RECV( 5) : From: P4( 3) ; Subject: jzibwgay
(p40127.0.0.1)15- P4 RECV( 6) : From: P3( 3) ; Subject: yfoquxtd
(p40127.0.0.1)15- P4 SEND( 4) : Subject: pfmzgtbk
(p40127.0.0.1)15- P4 RECV( 7) : From: P4( 4) ; Subject: pfmzgtbk
(p40127.0.0.1)15- P4 SEND( 5) : Subject: alvdfnik
(p40127.0.0.1)15- P4 RECV( 8) : From: P4( 5) ; Subject: alvdfnik
(p40127.0.0.1)15- P4 SEND( 6) : Subject: xrlpjtyk
(p40127.0.0.1)15- P4 RECV( 9) : From: P4( 6) ; Subject: xrlpjtyk
(p40127.0.0.1)15- P4 SEND( 7) : Subject: udoznegi
(p40127.0.0.1)15- P4 RECV( 10) : From: P4( 7) ; Subject: udoznegi
(p40127.0.0.1)15- P4 SEND( 8) : Subject: kpqqualz
(p40127.0.0.1)15- P4 RECV( 11) : From: P4( 8) ; Subject: kpqqualz
```

Figure 4: Sleep time = 100, Jitter time = 1000

Conclusions: Com s'ha comentat abans, al contrari del que passava anteriorment l'augment del jitter no provoca un augment del desordre degut a que els vector clocks de cada missatge (timestamp) fan que al arribar el receptor els pugui ordenar i entregar als següents nodes de manera ordenada. Aquest augment del jitter tal i com s'ha especulat en la hipotesis nomes pot provocar que el delivery dels missatges tarda més degut a que els missatges que arriben no son entregats fins que els anteriors (en temps de vector clock) ja han sigut entregats. Finalment dir que un temps d'Sleep petit provoca molts enviaments com veiem en les figures.

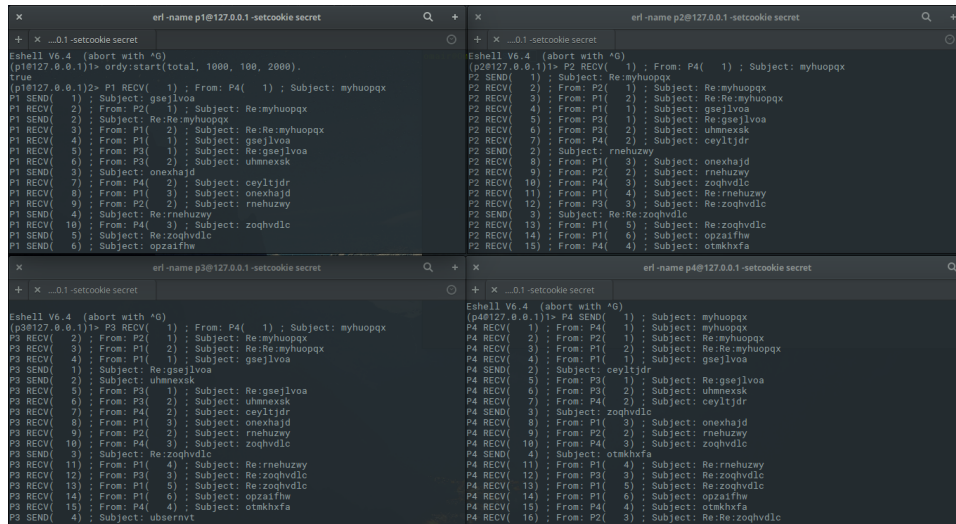
3.3 Total order multicast

En aquesta ultima implementacio l'objectiu es que els missatges no s'enviïn en desordre per tant s'implementa un algorisme que s'usa en el ISIS system i que consisteix en sol·licitar propostes de tot els nodes en un grup.

1. *Experiment amb sleep time molt més gran que jitter time*

Hipotesis: Ni el temps d'Sleep ni el de Jitter haurien de provocar un desordre, els missatges s'entreguen en ordre de nSeq ja que aquests estan ordenats per ordre de minim nSeq en la hold-back queue, donat l'alt temps de sleep s'enviaran pocs missatges.

Experimentacio:



```
erl-name p1@127.0.0.1-setcookie secret
Eshell V6.4 (abort with ^G)
(p1@127.0.0.1)> ordy:start(total, 1000, 100, 2000).
true
(p1@127.0.0.1)> P1 RECV( 1) : From: P4( 1) ; Subject: myhuopqx
P1 SEND( 1) : Subject: gsejlvaa
P1 RECV( 2) : From: P2( 1) ; Subject: Re:myhuopqx
P1 SEND( 2) : Subject: Re:myhuopqx
P1 RECV( 3) : From: P1( 2) ; Subject: Re:myhuopqx
P1 RECV( 4) : From: P1( 1) ; Subject: gsejlvaa
P1 RECV( 5) : From: P3( 1) ; Subject: Re:gsejlvaa
P1 RECV( 6) : From: P3( 2) ; Subject: uhmexsk
P1 SEND( 3) : Subject: onexhjd
P1 RECV( 7) : From: P4( 2) ; Subject: ceyltjdr
P1 RECV( 8) : From: P1( 3) ; Subject: onexhjd
P1 RECV( 9) : From: P2( 2) ; Subject: rnehuzay
P1 SEND( 4) : Subject: Re:rnehuzay
P1 RECV( 10) : From: P4( 3) ; Subject: zoqhvdic
P1 SEND( 5) : Subject: Re:zoqhvdic
P1 SEND( 6) : Subject: opzaifhw

erl-name p2@127.0.0.1-setcookie secret
Eshell V6.4 (abort with ^G)
(p2@127.0.0.1)> P2 RECV( 1) : From: P4( 1) ; Subject: myhuopqx
P2 SEND( 1) : Subject: Re:myhuopqx
P2 RECV( 2) : From: P2( 1) ; Subject: Re:myhuopqx
P2 RECV( 3) : From: P1( 2) ; Subject: Re:myhuopqx
P2 RECV( 4) : From: P1( 1) ; Subject: gsejlvaa
P2 RECV( 5) : From: P3( 1) ; Subject: Re:gsejlvaa
P2 RECV( 6) : From: P3( 2) ; Subject: uhmexsk
P2 RECV( 7) : From: P4( 2) ; Subject: ceyltjdr
P2 SEND( 2) : Subject: rnehuzay
P2 RECV( 8) : From: P1( 3) ; Subject: onexhjd
P2 RECV( 9) : From: P2( 2) ; Subject: rnehuzay
P2 RECV( 10) : From: P4( 3) ; Subject: zoqhvdic
P2 RECV( 11) : From: P1( 4) ; Subject: Re:rnehuzay
P2 RECV( 12) : From: P2( 3) ; Subject: Re:zoqhvdic
P2 SEND( 3) : Subject: Re:zoqhvdic
P2 RECV( 13) : From: P1( 5) ; Subject: Re:zoqhvdic
P2 RECV( 14) : From: P1( 6) ; Subject: opzaifhw
P2 RECV( 15) : From: P4( 4) ; Subject: otakhfwa

erl-name p3@127.0.0.1-setcookie secret
Eshell V6.4 (abort with ^G)
(p3@127.0.0.1)> P3 RECV( 1) : From: P4( 1) ; Subject: myhuopqx
P3 RECV( 2) : From: P2( 1) ; Subject: Re:myhuopqx
P3 RECV( 3) : From: P1( 2) ; Subject: Re:myhuopqx
P3 RECV( 4) : From: P1( 1) ; Subject: gsejlvaa
P3 SEND( 1) : Subject: Re:gsejlvaa
P3 SEND( 2) : Subject: uhmexsk
P3 RECV( 5) : From: P3( 1) ; Subject: Re:gsejlvaa
P3 RECV( 6) : From: P3( 2) ; Subject: uhmexsk
P3 RECV( 7) : From: P4( 2) ; Subject: ceyltjdr
P3 RECV( 8) : From: P1( 3) ; Subject: onexhjd
P3 RECV( 9) : From: P2( 2) ; Subject: rnehuzay
P3 RECV( 10) : From: P4( 3) ; Subject: zoqhvdic
P3 SEND( 3) : Subject: Re:zoqhvdic
P3 RECV( 11) : From: P1( 4) ; Subject: Re:rnehuzay
P3 RECV( 12) : From: P2( 3) ; Subject: Re:zoqhvdic
P3 RECV( 13) : From: P1( 5) ; Subject: Re:zoqhvdic
P3 RECV( 14) : From: P1( 6) ; Subject: opzaifhw
P3 RECV( 15) : From: P4( 4) ; Subject: otakhfwa
P3 SEND( 4) : Subject: ubsernvt

erl-name p4@127.0.0.1-setcookie secret
Eshell V6.4 (abort with ^G)
(p4@127.0.0.1)> P4 SEND( 1) : Subject: myhuopqx
P4 RECV( 1) : From: P4( 1) ; Subject: myhuopqx
P4 RECV( 2) : From: P2( 1) ; Subject: Re:myhuopqx
P4 RECV( 3) : From: P1( 2) ; Subject: Re:myhuopqx
P4 RECV( 4) : From: P1( 1) ; Subject: gsejlvaa
P4 SEND( 2) : Subject: ceyltjdr
P4 RECV( 5) : From: P2( 1) ; Subject: Re:gsejlvaa
P4 RECV( 6) : From: P3( 1) ; Subject: uhmexsk
P4 RECV( 7) : From: P4( 2) ; Subject: ceyltjdr
P4 SEND( 3) : Subject: zoqhvdic
P4 RECV( 8) : From: P1( 3) ; Subject: onexhjd
P4 RECV( 9) : From: P2( 2) ; Subject: rnehuzay
P4 RECV( 10) : From: P4( 3) ; Subject: zoqhvdic
P4 SEND( 4) : Subject: otakhfwa
P4 RECV( 11) : From: P1( 4) ; Subject: Re:rnehuzay
P4 RECV( 12) : From: P2( 3) ; Subject: Re:zoqhvdic
P4 RECV( 13) : From: P1( 5) ; Subject: Re:zoqhvdic
P4 RECV( 14) : From: P1( 6) ; Subject: opzaifhw
P4 RECV( 15) : From: P4( 4) ; Subject: otakhfwa
P4 RECV( 16) : From: P2( 3) ; Subject: Re:zoqhvdic
```

Figure 5: Sleep time = 1000, Jitter time = 100

Conclusions: Es pot observar que tots el processos reben la mateixa sequencia de missatges i no s'observa cap tipus de desordre entre ells.

2. *Experiment amb Sleep time mes petit que jitter time*

Hipotesis: Igual que en el experiment anterior, tots els processos rebran la mateixa sequencia de missatges.

Demostració:

Conclusions: tots els processos reben la mateixa sequencia de missatges nomes que ara cada proces envia un nombre mes gran de missatges degut al baix temps de Sleep.

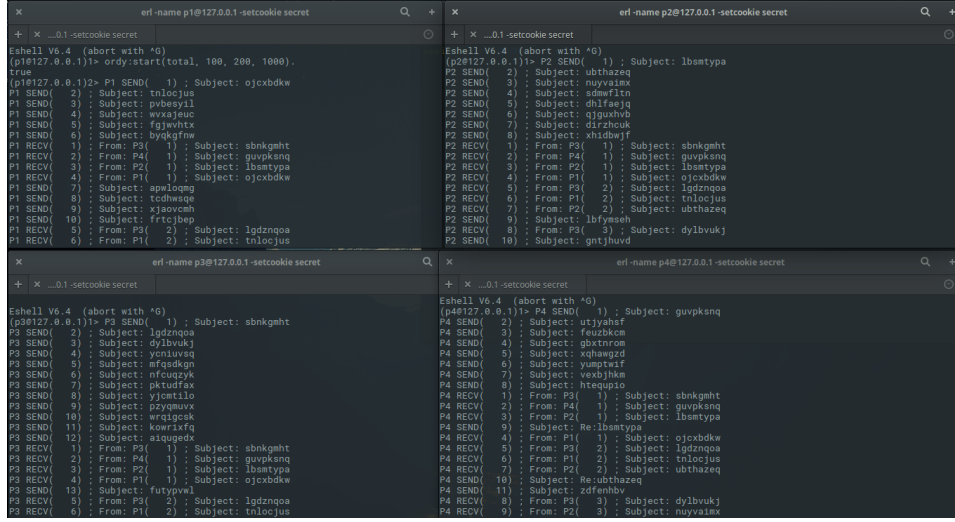


Figure 6: Sleep time = 100 , jitter = 200

3. We have a lot of messages in the system. Derive a theoretical quantification of the number of messages needed to deliver a multicast message as a function of the number of workers and check experimentally that your formulation is correct

totalMessages($n = n^o$ of workers)

Basic = $1 * \text{send} + n * \text{multicast} + n * \text{deliver} = 2*n + 1$

Total = $1 * \text{send} + n * \text{request} + n * \text{proposal} + n * \text{agreed} + n * \text{deliver} = 4*n + 1$

4 Open questions

4.1 Basic multicast

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

Aquesta primera implementacio de multicast no segueix cap dels tres ordres, Erlang ens garanteix que si un procés envia una secuencia de missatges, es rebran en aquella secuencia als altres nodes. En el nostre modul basic utilitzem la funcio `send after()` que fa que els missatges s'enviïn al cap d'un temps random que depen del parametre jitter, això pot provocar que per exemple si el temps d'un worker de contestar o tornar a enviar un missatge multicast es prou petit, pot provocar que el segon multicast del worker hi hagin missatges que s'enviïn abans del primer multicast. Es veura en els proxims experiments la influencia dels parametres de Sleep i Jitter en el multicast basic que s'ha donat implementat.

4.2 Causal order multicast

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

L'ordre causal si que es respecta usant els vector clocks, ja hem vist a teoria com els vector clocks ens ajuden a introduir una relacio de happened-before als missatges. Aquest ús dels vector clocks no ens garanteix l'ordre total degut a que per exemple dos missatges enviats al mateix instant (temps de vector clock) no es pot diferenciar quin a anat abans o despres degut a que no tenen relacio de causalitat. El que si ens garanteix l'ús de vector clocks es l'ordre FIFO degut a que dos missatges enviats per el mateix proces si que tenen una relacio d'ordre. En aquest algorisme el parametre de Jitter no influix en la perdua d'ordre causal o FIFO gracies als vector clocks i la cua ordenada per mantenir els missatges abans de entregar-los.

4.3 Total order multicast

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

No es compleix l'ordre FIFO per que la posicio de cada missatge depen del jitter amb que arriba als processos. Els processos no tenen una ordenacio interna dels missatges que envien, a diferencia del que es tenia en la versio causal, on estava implicit en els timestamps de cada missatge.

Tampoc es compleix l'ordre causal, no hem establert cap relacio de

happened-before entre els missatges (com ho poden ser els VC de la versio causal). Tampoc apareix com a efecte colateral de l'utilitzacio del numeros de sequencia tal i com s'explica a continuacio.

Si es repecta l'ordre total ja que tots els processos reben la mateixa sequencia de missatges que es pot demostrar a partir de dues propietats:

1) El numero de sequencia (nSeq en endavant) d'un missatge es únic en el sistema i identic en tots el processos. Si un proces ja ha proposat un nSeq ns, els següents que proposi seran com a minim a a partir de ns+1. Es per assegurar aquest punt que cada proces porta el compte de quin es el maxim nSeq que s'ha proposat fins al moment. A més, com dos procesos poden proposar el mateix nSeq per a dos missatges diferents, s'afegeix l'identificador del proces en els nSeq de manera que cada parell de nSeq es unic en el sistema. Com que un proces no proposara el mateix nSeq dos cops, llavors tenim unicitat en els nSeq. La segona part, que el nSeq d'un missatge és idèntic en tots els processos, és deriva del missatge d'agreement que el emissor envia a tots els processos, comunicant el nSeq acordat.

2) Els missatges s'entregen en ordre de nSeq. Aquesta propietat es compleix per la manera en que s'ordena la hold-back queue. Si un missatge m1 amb nseq a1 acordat precedeix un altre missatge m2, llavors:

a) si m2 te un nSeq acordat a2, llavors $a1 < a2$ per l'ordenacio de la cua.

b) Si m2 no te un nSeq acordat, llavors te un nSeq proposat p2 tal que $a1 < p2$. Per la manera com decidim el nSeq acordat (maxim dels proposats) sabem que m2 pendra un nSeq $a2 \geq p2$ i per tant, $a1 < a2$. La cua nomes es pot buidar treient el primer element si aquest te un nSeq acordat. Com els missatges estan ordenats en ordre de minin nSeq prospectiu, llavors els missatges s'entreguen en ordre de nSeq.

5 Personal opinion

Aquest lab ens ha ajudat ha comprendre millor els conceptes estudiats a teoria aixi com entendre com funciona un dels protocols amb mes exit avui en dia.