ESTAT DE L'ART

La computació distribuïda permet la segregació dels recursos computacionals de manera que un sistema pot estar format de diversos components heterogenis que treballen per a un mateix fi, a partir de la compartició dels seus recursos. Aquesta descentralització però, ha de permetre garantir uns llindars d'integritat i estabilitat a partir de serveis de replicació de dades i serveis tolerants a fallades.

Per poder avaluar la xarxa social de mocroblogging descentralitzada Garlanet, es necessita valorar diferents aspectes de la plataforma que tinguin afectació a les característiques principals del sistema distribuït: rendiment, disponibilitat del servei i experiència de l'usuari. Entre aquests aspectes podem trobar: el dinamisme (fallada o desconnexions), la connectivitat (amplada de banda) i els comportaments dels usuaris de la xarxa que puguin condicionar un o més aspectes de la plataforma.

El procés de recerca d'estudis realitzats amb anterioritat s'ha centrat en dos aspectes: l'activitat dels usuaris i el comportament dels repositoris no dedicats aportats voluntàriament.

Els aspectes que s'han avaluat en altres plataformes descentralitzades semblants com Hornet [1], donen una primera proposta de què s'ha de tenir en compte per contrastar el rendiment. En primer lloc, es pot realitzar una estimació de l'ús de les dades que circulen per la plataforma, aquestes es poden tenir diferents orígens en funció de la font que les genera: els usuaris, la lògica de l'aplicació, etc.

El volum de les dades transmeses pel canal de comunicació és molt important, ja que normalment, a més volum més recursos computacionals es necessiten, pel que és un factor que pot repercutir directament al rendiment generalitzat de la plataforma, podent afectar a la seva estabilitat i sent un candidat primordial a avaluar.

No menys important és l'emmagatzemament de les dades, les xarxes socials acostumen a generar un gran volum de dades que han de poder-se emmagatzemar i processar quasi de manera continuada, aquest volum de dades, pot contenir tant la informació facilitada directament per l'usuari en forma de missatges, com informació no tan explícita però important com les seqüències, sessions, etc. que es generen a partir de les interaccions persona-ordinador.

S'ha de tenir en compte que el volum de dades transmeses pel canal de comunicació i el volum de les dades a emmagatzemar i processar no tenen per què ser les mateixes en la seva totalitat, ja que parteixen de bases i naturaleses diferents que fan que tinguin finalitats i continguts diferents.

Els arguments que aporta la proposta de Hornet són molt extrapolables a Garlanet degut que parteixen d'una base similar en quant l'aportació voluntària de recursos a la plataforma i la seva finalitat principal. Així mateix, plataformes com BOINC [2] també contribueixen a ampliar les propostes proporcionant coneixement de com es comporten els recursos no dedicats aportats de manera voluntària, sent un símil en finalitat de com ho permet l'arquitectura de Garlanet amb els seus repositoris. Els usuaris poden dedicar el tems d'inactivitat dels seus recursos per al processament de càlculs (aplicacions, projectes, etc.).

Així doncs, per poder avaluar la plataforma, es necessita poder modelar els aspectes que puguin incidir amb l'enumerat, aquests són: la simulació del comportament dels usuaris i la simulació del comportament dels repositoris.

Simulació dels usuaris:

Interessen els comportaments [3] que puguin interferir en la plataforma, com ho són l'enviament de missatges entre els mateixos usuaris. Aquests missatges són dades que ells aporten a la xarxa social en forma de continguts.

Durant la simulació de comportament dels usuaris és important definir comportaments ben diferenciats, ja que la distribució de missatgeria intercanviada no té per què ser sempre uniforme i equivalent. Per tant, s'han de tenir en compte la diversitat i les diferents tendències:

- Usuaris que envien pocs missatges.
- Usuaris que envien molts missatges.
- Usuaris que es mantenen a dintre d'una quantitat mitjana de missatges enviats.

A més a més, es poden definir tendències globals a partir de les individuals, de manera que:

- La majoria d'usuaris enviïn pocs missatges.
- La majoria d'usuaris enviïn molts missatges.
- Pocs usuaris enviïn molts missatges.
- Molts usuaris només rebin missatges i no n'enviïn.

Després, per definir més paràmetres que permetin caracteritzar l'activitat dels usuaris es poden utilitzar altres aspectes vistos en altres dels estudis cercats, com la mida de la cua de missatges pròpia [4] [5] i la mitjana de missatges rebuts en un temps definit. Aquest darrer permet perfilar el volum de missatgeria que s'envia en funció del temps, ja que no és el mateix enviar una gran quantitat de missatges per un mateix usuari en una línia temporal extensa que fer-ho en un període de temps més reduït.

Per això, amb l'estudi de *Barracudalabs*, es podran definir característiques i parametritzacions per definir l'activitat dels usuaris. Quan es crea un usuari que forma part de la simulació, s'ha de decidir a quin grup formarà part, a partir de les dades extretes de l'informe podem classificar:

Percentatge de classificació segons el nombre de followers:

- 0 followers: 11%
- 1 a 9 followers: 39%
- 10 a 99 followers: 33%
- 100+ followers: 17%

Pel que l'usuari simulat pot decidir-se amb la generació d'un nombre a partir d'una distribució uniforme discreta en funció del percentatge total. Dintre de la classificació, es definiran el nombre d'usuaris que el segueixen dintre del mateix acotament.

Per al cas dels following, es pot emprar el mateix sistema de l'informe.

Amb les gràfiques de *barracudalabs*, podem observar que en funció del nombre de seguidors, es

tendeix a un comportament en nombre de missatges enviats, aquest cas d'estudi també es farà servir a la simulació, de manera que cada usuari es repartirà en funció dels followers i, a més, aquesta classificació definirà el bloc de nombre de missatges a enviar.

També, existeixen els estudis de predicció de personalitat dels usuaris [6] que, tot i no ser aplicables directament, es pot aprofitar per extreure característiques implícites, resultant en el que es coneixen com a relacions entre usuaris:

- Usuaris que són seguits per molts usuaris.
- Usuaris que segueixen a molts usuaris.

Aquestes són dades relacionals, i per tant, forment part d'un altre conjunt d'informació que ha de ser emmagatzemat i processat.

Les relacions de seguiment entre usuaris, es poden interpretar com a grafs dirigits etiquetats [7] [8] [9] [10] on l'atribut relacional de cada usuari pot ser implementat en aquesta estructura de dades.

Atès que els estudis cercats ofereixen unes característiques útils per a la implementació de la simulació dels usuaris, els aspectes que es consideraran per realitzar la implementació són:

- Nombre de missatges enviats.
- Nombre de seguidors.
- Nombre d'usuaris que segueixen un usuari.
- Distribució de missatges enviats i rebuts en un període de temps.
- Efecte que la generació d'un missatge pugui tenir en la generació d'altres missatges (reenviament de missatges per part dels usuaris).
- Distribució de l'activitat durant un temps acotat.
- Possibilitat de tractar la repercussió que puguin tenir les diferents franges horàries sobre el comportament dels usuaris.

Simulació dels repositoris:

Altrament, una altra de les característiques de Garlanet que es vol avaluar són els repositoris. Per això, les traces de diferents sistemes paral·lels i distribuïts [11] que estan disponibles al públic, són fonts d'informació adients que aporten a la decisió del comportament a simular pels repositoris.

Cada repositori es caracteritza i defineix com un tipus diferent a partir del seu comportament, es desitja que durant la validació el sistema tingui la capacitat de ser tolerant a fallades, i per això és imprescindible obtenir les traces adients [12] per comprovar si, un cop avaluat, i amb les dades recollides posteriorment, es pot catalogar el sistema com estable i es pot afirmar que té la capacitat de tolerància a fallades que s'esperava.

Per això s'han definit les característiques següents a implementar en el comportament dels repositoris:

- Repositoris actius.
- Repositoris inactius.
- Fallades de connexió entre repositoris.

Per a l'avaluació del sistema ja existeixen “live systems” tipus PlanetLab [13] que estan designats per a la investigació en les àrees de xarxes i sistemes distribuïts i que proporcionen la capacitat per desplegar aplicacions distribuïdes. Es desplega Garlanet a aquesta plataforma amb les implementacions de simulació realitzades, de manera que els experiments puguin apropar-se més a un sistema real distribuït. Els nodes utilitzats durant la simulació formaran part dels recursos aportats pels usuaris del sistema.

En conseqüència, a partir de les dades obtingudes durant les experimentacions i desplegaments als 'live systems' en forma de traces [14], es quantificarà la disponibilitat de Garlanet.

MODELAT DE TWITTER

- Modelat de Twitter

2.- Modelat del comportament dels usuaris

Per tal de modelar l'activitat dels usuaris utilitzarem l'estudi “Barracuda Labs 2010 Annual Security Report” de Barracuda Labs del 2010 [1].

2.1.- Quantitat de followers

L'estudi de [5] classifica els usuaris de Twitter en les categories següents:

Categories segons el nombre de followers	Percentatge d'usuaris
0	11%
1 – 9	39%
10 – 99	33%
100+	17%

A l'hora de crear els usuaris que simularan l'activitat a la plataforma, es necessita classificar-los segons aquesta taula de relació per nombre de 'followers'. Per decidir a quin grup pertany cada usuari, es farà de la manera següent:

1. Assignar l'usuari en una categoria de nombre de followers.

Es genera un nombre aleatori utilitzant una distribució uniforme discreta [15] en el rang [0,100):

utilitzant el mètode nextInt(int n) de la classe Java java.util.Random, que retorna un valor aleatori uniformement distribuït comprès entre zero (inclòs) i el valor especificat “n” (exclòs).

Segons el nombre aleatori generat l'usuari s'assignarà a una de les categories anteriors:

Nombre aleatòri	Categoria de followers (rang de followers)
0 – 10	0
11 – 49	1 – 9
50 – 82	19 – 99
83 – 99	100+

2. Definir el nombre concret de followers de l'usuari.

Per calcular el nombre exacte de followers que tindrà l'usuari dintre del rang esmentat, generarem el nombre exacte de followers a partir de la taula següent:

Categoria de followers	Nombre de followers
0	0
1 – 9	nombre generat per distribució uniforme discreta en el rang [1, , 9]: `org.apache.commons.math3.distribution.UniformIntegerDistribution(1, 9);`
10 – 99	nombre generat per distribució uniforme discreta en el rang [10, 99]: `org.apache.commons.math3.distribution.UniformIntegerDistribution(10, 99);`
100+	Nombre més gran de 100 generat per distribució geomètrica amb alpha 0,005: `100 + (int) (Math.log(r.nextDouble()) / Math.log(1 - alpha));`

Per fer els càlculs es fa servir la lliberia “Apache Commons Mathematics Library” [20].

2.2.- Generació d'activitat

L'estudi de Barracuda Labs [5] que utilitzem com a base per a simular el comportament dels usuaris no proporciona dades per a poder definir el nivell d'activitat de cada usuari segons el nivell de followers que té. Per aquest motiu hem definit unes taules que permetin definir el nivell d'activitat dels usuaris segons la categoria de followers a la que s'ha assignat l'usuari en el pas 1 de l'apartat anterior.

En la següent taula hem definit quin nivell d'activitat tindrà cada usuari segons la categoria de followers a la que s'ha assignat:

	Categoria followers estudi Barracuda					
	0	**1 – 9**	**10 – 99**	**100+**		
Percentatge d'usuraris estudi Barracuda	11%	39%	33%	17%		
79,4%	10,8%	36,9%	24,9%	6,8%	<1	**Categoria nivells d'activitat (nombre de missatges/dia) estudi Barracuda**
10,9%	0,2%	1,2%	4,5%	5%	1 – 4	
5,4%	0%	0,4%	2%	3%	5 – 9	
3,9%	0%	0,4%	1,5%	2%	10 – 99	
0,4%	0%	0,1%	0,1%	0,2%	100+	

Cada percentatge de color blau estableix quin percentatge d'usuaris hi haurà en cada combinació de categoria de followers i categoria d'activitat d'usuaris. Per exemple, el 36,9% d'usuaris tenen entre 1 i 9 followers i generen menys d'un missatge al dia.

Es continua establint la relació però cada valor de cada fila indica el percentatge dintre de la mateixa columna de followers:
A continuació presentem la mateixa informació però, en lloc de posar el percentatge d'usuaris sobre el nombre total d'usuaris, es posa el percentatge d'usuaris de cada categoria de nivell d'activitat per cada categoria de followers. És a dir, cada fila d'una columna és el percentatge d'usuaris de cada categoria de nivell d'activitat que estan assignats a la categoria de nombre de followers.

		Categoria followers			
		0	**1 – 9**	**10 – 99**	**100+**
Categoria nivells d'activitat (nombre de missatges/dia) estudi Barracuda	**<1**	98,18%	94,62%	75,45%	40%
	1 – 4	1,82%	3,08%	13,64%	29,41%
	5 – 9	0%	1,03%	6,06%	17,65%
	10 – 99	0%	1,03%	4,55%	11,76%
	>100	0%	0,26%	0,3%	1,18%

Els passos per establir el nombre de missatges que enviarà l'usuari per dia es farà de la següent manera:

1. Assignar usuari en una categoria de nivell d'activitat

Es genera un nombre aleatori amb una distribució uniforme [15] entre [0, 100):

```
org.apache.commons.math3.distribution.UniformRealDistribution(0., 100. -
Double.MIN_VALUE);
```

		Categoria de followers			
Categoria nivells d'activitat (nombre de missatges/dia)		**0**	**1 – 9**	**10 – 99**	**100+**
	<1	$0 \leq x < 98{,}18$	$0 \leq x < 94{,}62$	$0 \leq x < 75{,}45$	$0 \leq x < 40$
	1 – 4	$98{,}18 \leq x < 100$	$94{,}62 \leq x < 97{,}71$	$75{,}45 \leq x < 89{,}09$	$40 \leq x < 69{,}41$
	5 – 9	N/A	$97{,}71 \leq x < 98{,}74$	$89{,}09 \leq x < 95{,}15$	$69{,}41 \leq x < 87{,}06$
	10 – 99	N/A	$98{,}74 \leq x < 99{,}77$	$95{,}15 \leq x < 99{,}7$	$87{,}06 \leq x < 98{,}82$
	>100	N/A	$99{,}77 \leq x < 100$	$99{,}71 \leq x < 100$	$98{,}83 \leq x < 100$

2. Determinació nivell d'activitat diària de cada usuari.

Una vegada classificat l'usuari, en una de les categories de nivell d'activitat cal determinar el nombre de missatges que l'usuari enviarà diàriament:

2.1 En el cas de què el nombre de missatges sigui <1, tindrem dos casos diferents segons el nivell d'activitat que vulguem provar:

generarem un nombre aleatori [0,100) amb una distribució uniforme discreta.

Amb: org.apache.commons.math3.distribution.UniformIntegerDistribution(0, 100 – 1).sample();

Cas d'activitat normal: Si aquest nombre és entre [0, 10], llavors s'enviarà un missatge; en cas contrari, no s'enviarà cap missatge.

Cas de molta activitat: Si el nombre és entre [0, 50], llavors s'enviarà un missatge; en cas contrari, no s'enviarà cap missatge.

2.2 Per a la resta de casos:

Categoria 1 – 4: es genera un nombre aleatori [1, 4] amb una distribució uniforme discreta i s'envien la quantitat de missatges que indiqui el nombre. Amb: org.apache.commons.math3.distribution.UniformIntegerDistribution(1,4).sample();

Categoria 5 – 9: es genera un nombre aleatori [5, 9] amb una distribució uniforme discreta i s'envien la quantitat de missatges que indiqui el nombre. Amb: org.apache.commons.math3.distribution.UniformIntegerDistribution(5,9).sample()

Categoria 10 – 99: es genera un nombre aleatori [10, 99] amb una distribució uniforme discreta i s'envien la quantitat de missatges que indiqui el nombre. Amb: org.apache.commons.math3.distribution.UniformIntegerDistribution(10,99).sample();

Categoria +100: es genera un nombre amb una distribució geomètrica [16] amb alpha 0,04 i s'envien la quantitat de missatges que indiqui el nombre.

```
100 + (int)(Math.log(r.nextDouble())/Math.log(1-alpha))
```

Sent r un random number generator.
Aplha: 0,04

2.3.- Quantitat de following

S'ha de decidir cada usuari a qui segueix per tal que cada usuari tingui els followers que s'han calculat en el punt anterior. Per tal de fer una simulació el més realista possible el nombre d'usuaris als quals segueix un usuari s'ajusti a les categories que surten a l'estudi de Barracuda:

- El 16% no segueix a ningú.
- El 27% segueixen entre 1 i 9 usuaris.
- El 40% segueixen entre 10 i 99 usuaris.
- El 17% segueixen 100 o més usuaris.

Aquest pas s'ha realitzat de manera posterior al càlcul dels followers però inmediatament abans d'establir les relacions, s'han definit els quatre blocs amb les quantitats màximes per following. En el moment de repartir tots els followers, es respecten els valors màxims per bloc de following per a cada assignació.

3.- Modelat de comportament dels repositoris

Els repositoris poden estar en un dels dos següents estats connectat o desconnectat. A efectes de l'experiment considerem que desconnectat vol dir en fallada o desconnectat voluntàriament. Els nodes aniran alternant l'estat connectat i desconnectat. Modelarem la durada de la connexió i desconnexió dels repositoris segons [17]:

Els repositoris connectats calcularan el nombre de temps que estaran disponibles a partir de la funció de distribució Weibull [18], següent:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0, \\ 0 & x < 0, \end{cases}$$

S'utilitzarà la classe RandomDataGenerator de l'Apache Commons Math 3 API:

org.apache.commons.math3.random.RandomDataGenerator.nextWeibull(double shape, double scale);

Amb els paràmetres definits a [17]:

λ	κ

2.964	0.393

Transcorregut el temps calculat amb la distribució, el repositori es desconnectarà i estarà indisponible durant un nombre de temps calculat segons la distribució Log-Normal [19] següent:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln(x)-\mu)^2/2\sigma^2}$$

S'utilitzarà la classe LogNormalDistribution de l'Apache Commons Math 3 API:

org.apache.commons.math3.distribution.LogNormalDistribution.sample();

Sent els valors:

μ	**σ**
-0.586	2.844

Els repositoris desconnectats ho faran durant aquest període de temps, després, passaran a estar connectats a partir de la funció de disponibilitat i així successivament de manera cíclica.

ARQUITECTURA

ActivitySimulator: conté tota la lògica extreta a partir de l'estudi del funcionament de Twitter. S'encarrega de fer tots els càlculs per treure els 'description' dels usuaris (following, followers, nombre de missatges, nom de l'usuari, password, keystore, certificats, etc.). Serveix els description als usuaris simulats (UserSimulator).

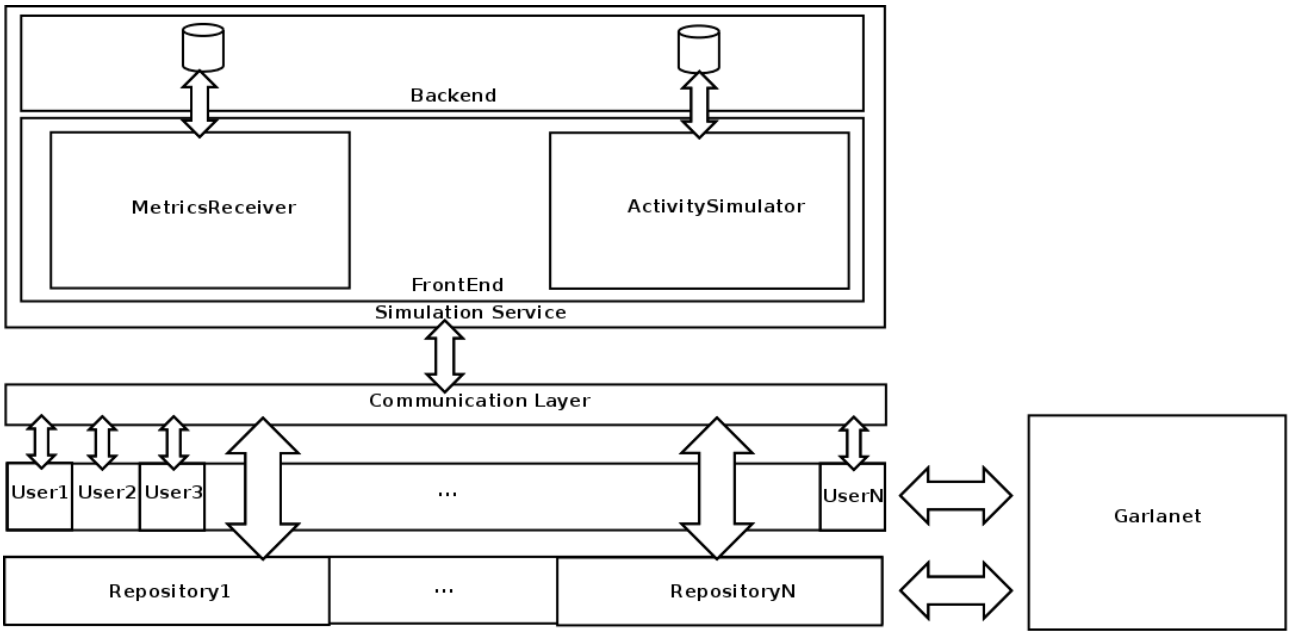
GarlanetSimulator: és el servei que s'encarrega de simular Garlanet, proporciona la capacitat de fer els registres dels usuaris, el login, logout, i l'enviament de missatges als followers dels usuaris.

La darrera part del sistema, *UserSimulator*, representa els clients i és l'encarregat de simular el comportament dels usuaris. Es connecta als dos serveis anteriors, al primer per obtenir el 'description' i al segon per enregistrar-se, loguejar-se, enviar missatges i sortir.

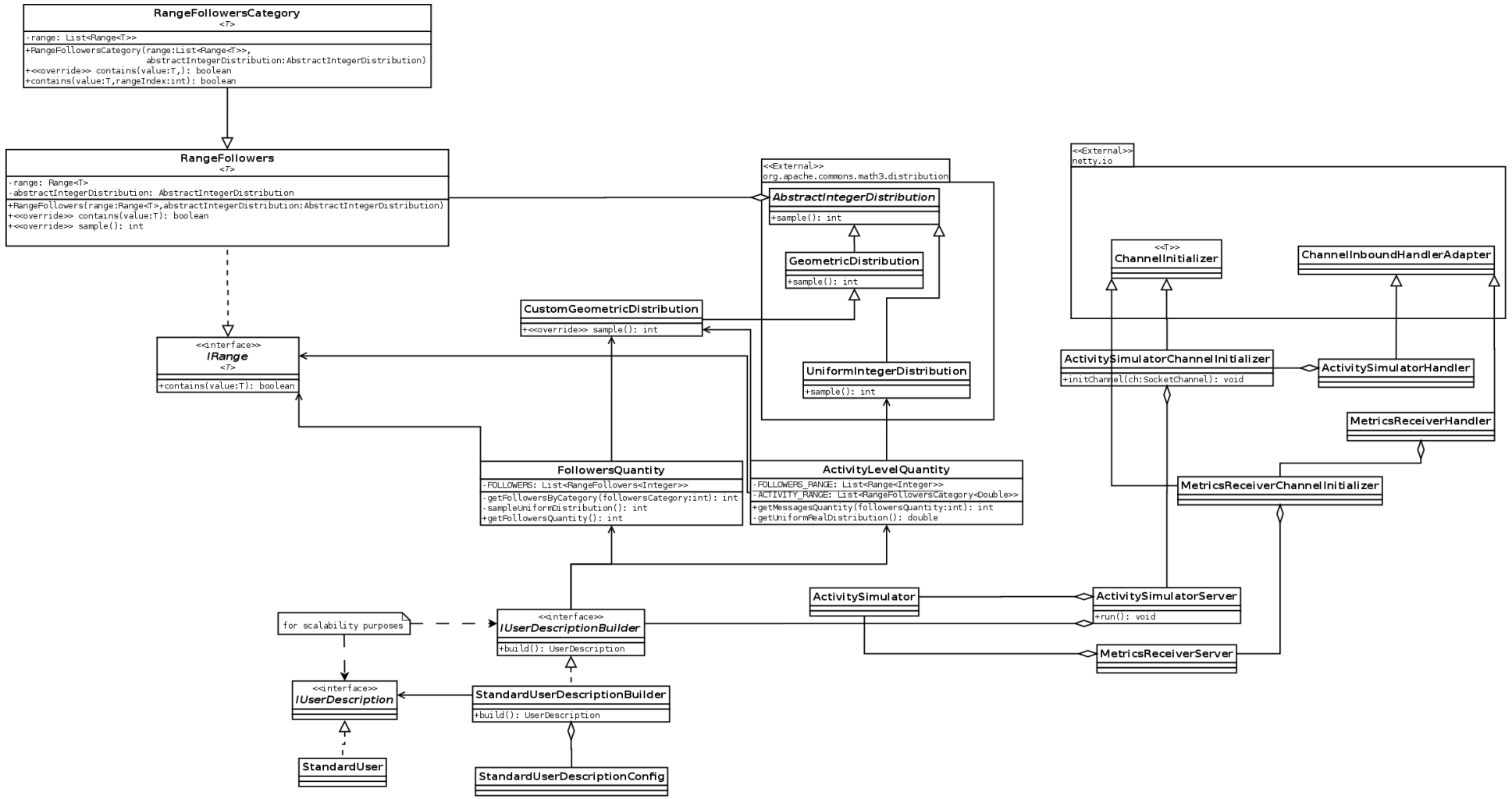
El flux funcional és el següent:

S'engega el servei ActivitySimulator, aquest calcula els 'descriptions' i es queda a l'espera de peticions.
Es posa en funcionament el GarlanetSimulator, que crea els handlers i també resta a l'espera de peticions.
S'aixequen els N usuaris, cada usuari es connecta a l'ActivitySimulator i obté el seu description. Després es connecten al GarlanetSimulator, on s'enregistren, fan login, i comencen a enviar missatges en funció de la seva definició del 'description' i el temps configurat de duració de la simulació. Els missatges son enviats al GarlanetSimulator, el qual distribueix en funció dels

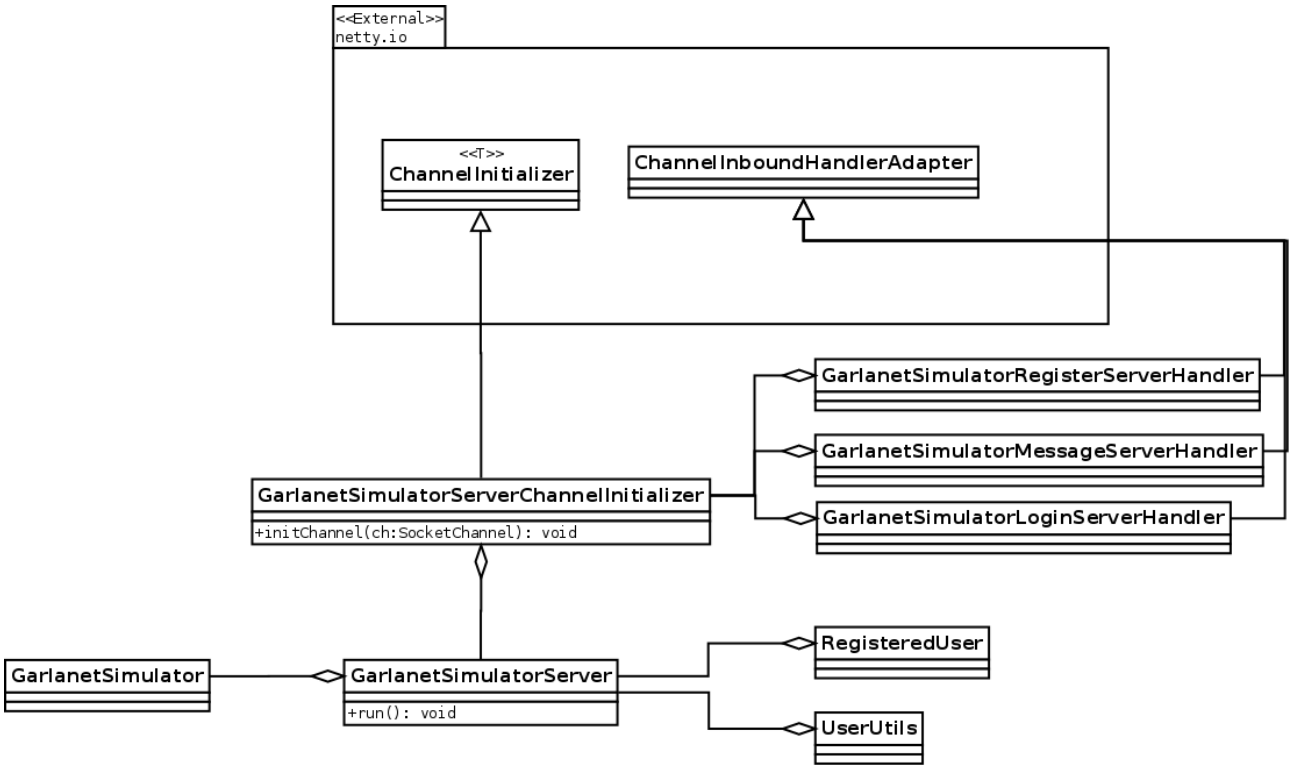
seguidors que tingui l'usuari en concret. Quan s'ha exhaurit el temps, els usuaris acaben de rebre els missatges, fan logout i s'atura la simulació.
Es presenta l'arquitectura, mètodes i diagrames de classes següents:



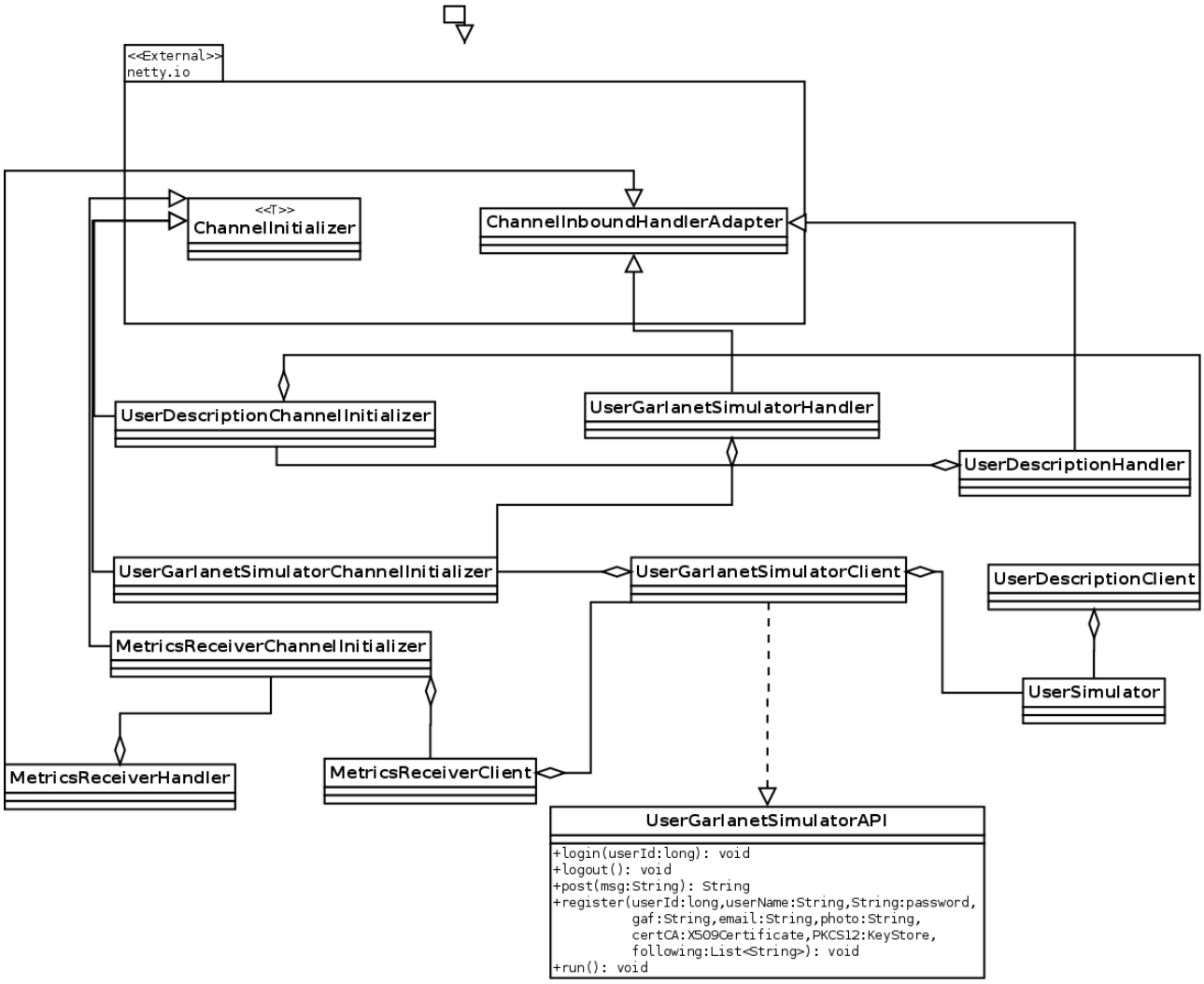
ActivitySimulator (s'obvien les classes creades de suport i testing).



GarlanetSimulator (s'obvien les classes creades de suport i testing).



UserSimulator (s'obvien les classes crades de suport i testing).

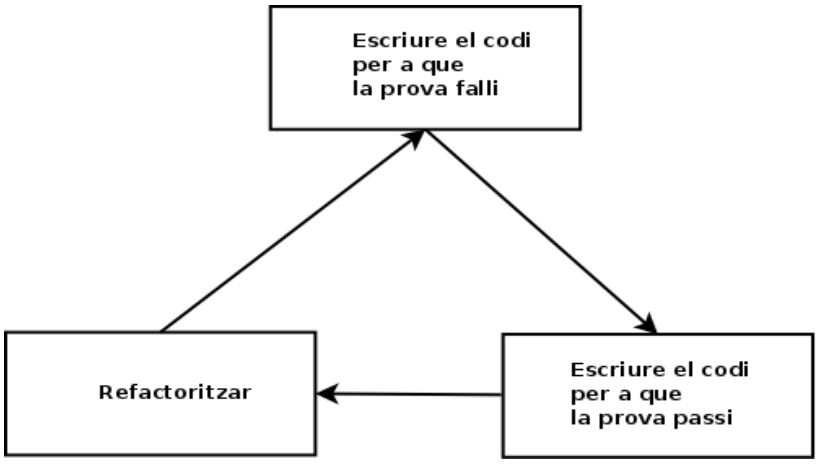


IMPLEMENTACIÓ

El mètode de desenvolupament utilitzada durant la implementació del projecte ha estat el "***Test-driven development (TDD)***". Aquest consisteix a programar primer les proves unitàries i la refactorització del codi, de manera que es pugui provar la funcionalitat i garantir que compleix els requisits.

El cicle de desenvolupament guiat per proves es basa, principalment, en:

- Escriure una prova que falli: a partir dels requeriments que s'hagin d'acomplir, s'escriu el codi necessari per a què la prova falli. Si la prova no falla, pot significar que el requeriment ja estava implementat o que la prova és incorrecta.
- Escriure una prova que passi: igual que l'anterior, a partir dels requeriments que s'hagin d'acomplir i havent verificat que la prova que falla és comporta com s'esperava, s'escriu el codi per a què aquesta prova passi.
- Refactoritzar: un cop comprovat que la prova passa, es refactoritza el codi, eliminant les parts redundants, optimització, etc.

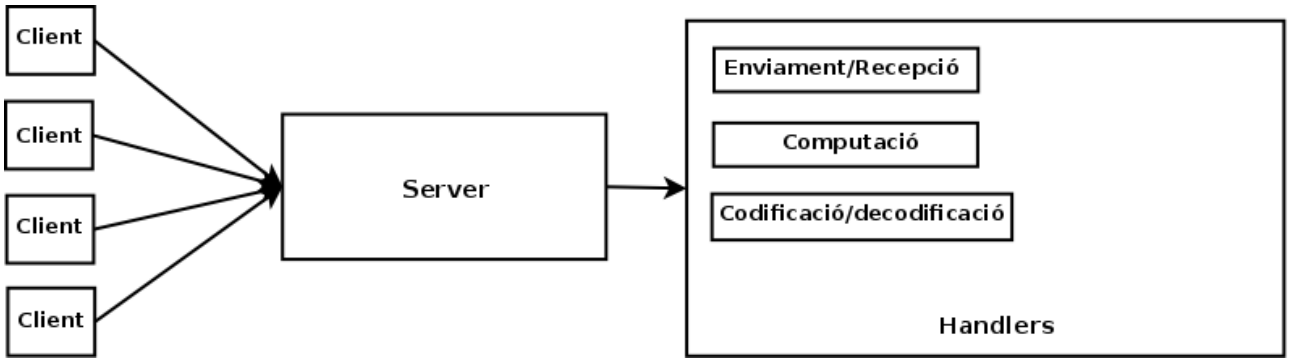


El llenguatge de programació emprat per al desenvolupament del projecte ha estat el Java. La JDK utilitzada ha estat la Oracle JDK 1.7 i el JRE mínim per executar-ho és el 1.7.

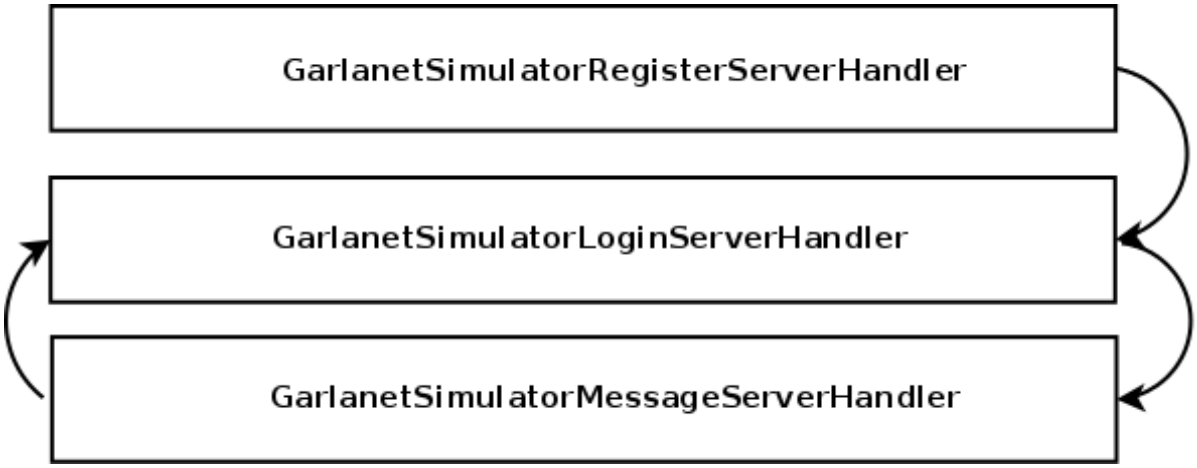
COMUNICACIÓ DE XARXA

Per a la comunicació en xarxa s'ha utilitzat Netty 4.1, un framework NIO client-servidor. En fer servir tecnologies de comunicació de xarxes, s'ha escollit Netty perquè té caràcter asíncron, fet que facilita l'escalabilitat. A més, és el framework que fa servir actualment Twitter [21]. Per a més informació i millor comprensió del codi, es recomana la lectura de la early version del llibre: Netty in Action [25].

S'ha emprat el disseny clàssic per a la comunicació client-servidor:



La capa de handlers que intervenen en els processos d'usuari del GarlanetSimulatorServer i les seves superposicions són:



- Tots els usuaris han d'estar registrats per a poder fer login
- Tots els usuaris han de fer login per a poder enviar missatges
- Els usuaris que han fet login poder fer logout.

El handler GarlanetSimulatorRegisterServerHandler gestiona el procés de registre dels usuaris.
El handler GarlanetSimulatorLoginServerHandler gestiona el procés de login i logout dels usuaris.
El handler GarlanetSimulatorMessageServerHandler gestiona el procés d'enviament de missatges (post).

Threading

Els serveis programats amb netty dos tipus de pool de threads: pool de connection threads i pool de worker threads.

Els connection threads (boss threads): cada ServerSocketChannel té els seu propi boss thread. Els boss threads accepten les connexions entrants, un cop acceptada la connexió, es passa el canal cap al worker thread.

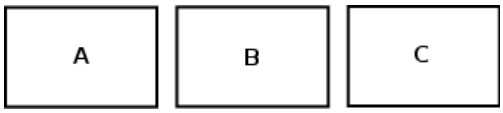
Els worker threads: realitza les operacions d'E/S d'un o més canals de manera no bloquejant.

Data frames

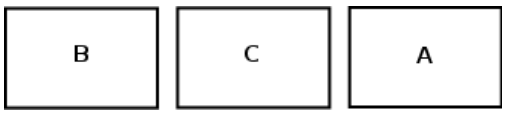
Tant els serveis com els clients, utilitzen frames per a comunicar-se entre ells. Aquests frames són el resultat de la fragmentació dels paquets per a enviar-se a través de la xara.

Per exemple, suposem que tenim el paquet següent, que representa un missatge que volem enviar al servidor Garlanet:

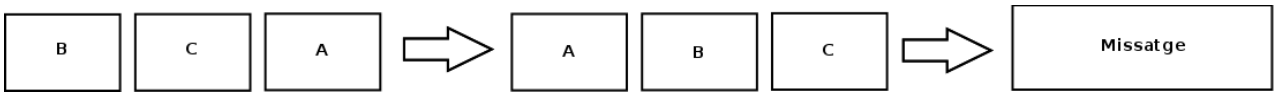
Per a transmetre'l per la xarxa, es divideix en frames:



Que poden arribar a destí de la manera següent:



Netty utilitza els FrameDecoders per a interpretar els frames i entendre'ls correctament, quedant la reconstrucció com l'original:



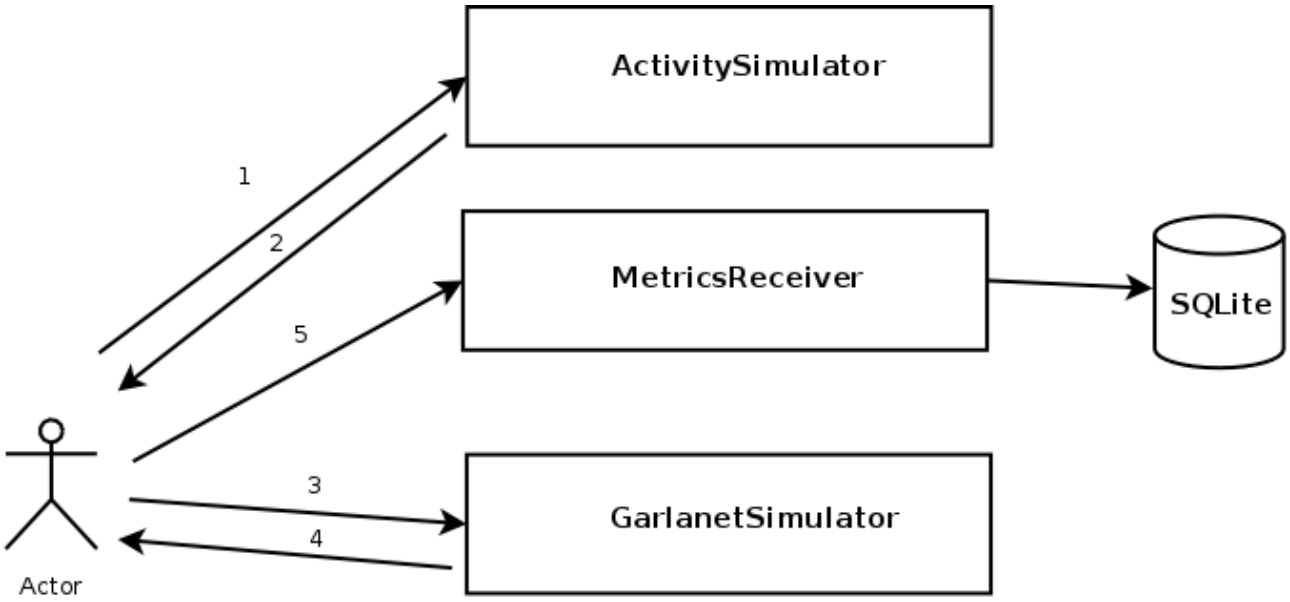
En la transmissió dels missatges via netty s'ha escollit com a FrameDecoders/Encoders els següents:

- StringDecoder
- StringEncoder
- ObjectEncoder
- ObjectDecoder

Els seus noms són descriptius, i s'han fet servir en funció del què s'està enviant en cada cas.

En aquest escenari també intervenen els FrameDelimiters, són els que especifiquen com es truncaran els paquets. S'han limitat l'enviament de strings, amb els delimiters definits amb una mida màxima per línia de text. S'han escollit les mides adients per a cada tipus de servei i client.

FLUX DE LES APLICACIONS



El flux és el següent:

1.- L'usuari simulat demana a l'ActivitySimulator participar en l'experiment. Aquest, el registra.
2.- l'ActivitySimulator li envia el Description de l'usuari.
3.- L'usuari simulat, actua segons el Description rebut: registrant-se, fent login, enviant missatges i fent logout.
4.- Durant la interacció amb GarlanetSimulator, l'usuari rep els missatges dels qui segueix.
5.- Durant l'experiment, l'usuari envia les mètriques al servei MetricsReceiver, que desa aquestes a una base de dades sqlite.

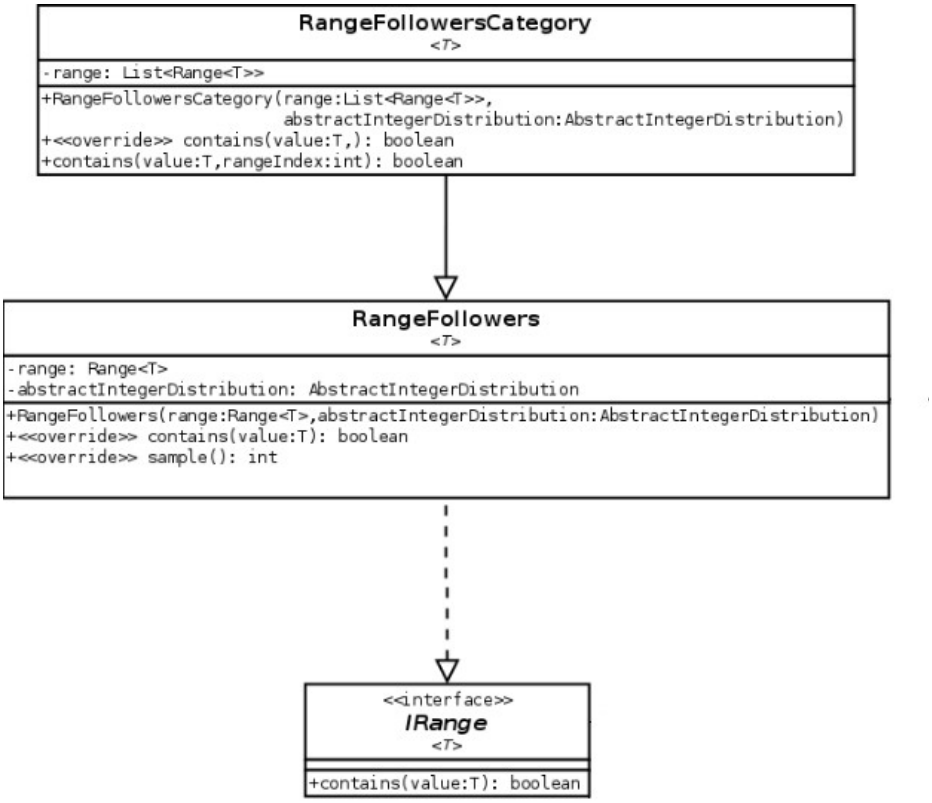
RENDIMENT

A mesura del possible, s'han fet servir les pràctiques recomanades en qüestions de rendiment. El recull utilitzat ha estat de la pàgina discontinuada “Precise Java” [26].

Una de les pràctiques més utilitzada ha estat en l'optimització de bucles, com per exemple, els recorreguts inversos.

JAVA GENERICS

S'han implementat classes que fan ús de la genericitat que proporciona Java a partir de la versió 5. Aquestes classes representen els rangs de followers i els rangs de followers per classificar-se en categories, que s'han representat com a taules anteriorment. Es poden fer ús d'aquests per a qualsevol tipus de dades.



PROVES UNITÀRIES

Per a la realització de les proves unitàries s'ha fet servir:

- JUnit
- PowerMock
- Mockito

Aquestes classes es troben sota el directori src/test/java de cada projecte.

Les proves realitzades són:

ActivityLevelQuantityTest:

- Test del nivell d'activitat per a 0 followers, x = 50. Resultat: menys d'un missatge/dia.
- Test del nivell d'activitat per a 0 followers, x = 99. Resultat: entre 1-4 missatges/dia.
- Test del nivell d'activitat per a 1-9 followers, x = 93,33. Resultat: menys d'un missatge/dia.
- Test del nivell d'activitat per a 1-9 followers, x = 97.70. Resultat: entre 1-4 missatges/dia.

- Test del nivell d'activitat per a 1-9 followers, x = 97.71. Resultat: entre 5-9 missatges/dia.
- Test del nivell d'activitat per a 1-9 followers, x = 99.2. Resultat: entre 10-99 missatges/dia.
- Test del nivell d'activitat per a 1-9 followers, x = 99.99. Resultat: més de 100 missatges/dia.
- Test del nivell d'activitat per a 10-99 followers, x = 70. Resultat: menys d'un missatge/dia.
- Test del nivell d'activitat per a 10-99 followers, x = 77.77. Resultat: entre 1-4 missatges/dia.
- Test del nivell d'activitat per a 10-99 followers, x = 90.01. Resultat: entre 5-9 missatges/dia.
- Test del nivell d'activitat per a 10-99 followers, x = 96. Resultat: entre 10-99 missatges/dia.
- Test del nivell d'activitat per a 10-99 followers, x = 99.77. Resultat: més de 100 missatges/dia.
- Test del nivell d'activitat per a +100 followers, x = 30. Resultat: menys d'un missatge/dia.
- Test del nivell d'activitat per a +100 followers, x = 67.77. Resultat: entre 1-4 missatges/dia.
- Test del nivell d'activitat per a +100 followers, x = 87.01. Resultat: entre 5-9 missatges/dia.
- Test del nivell d'activitat per a +100 followers, x = 96. Resultat: entre 10-99 missatges/dia.
- Test del nivell d'activitat per a +100 followers, x = 99.77. Resultat: més de 100 missatges/dia.

CustomGeometricDistributionTest:

- Comprova que els resultats dels càlculs de nombres aleatoris per a les distribucions geomètriques sigui l'esperat. Fa un test d'1.000.000 mostres, on el resultat ha de ser del 100%.

FollowersQuantityTest:

- Per a un nombre aleatori entre 0– 10: 0 followers.
- Per a un nombre aleatori entre 11 – 49: 1-9 followers.
- Per a un nombre aleatori entre 50 – 82: 10-99 followers.
- Per a un nombre aleatori entre 83 – 99: 100+ followers.

IOUtilsTest:
Tot i que aquest test es “surt” una mica de l'origen dels tests unitaris per a comprovar lògica, s'ha creat aquest test a l'inici del desenvolupament per comprovar que tota la part relacionada amb la generació automàtica de certificats X509, keystores i keypairs funcioni correctament.

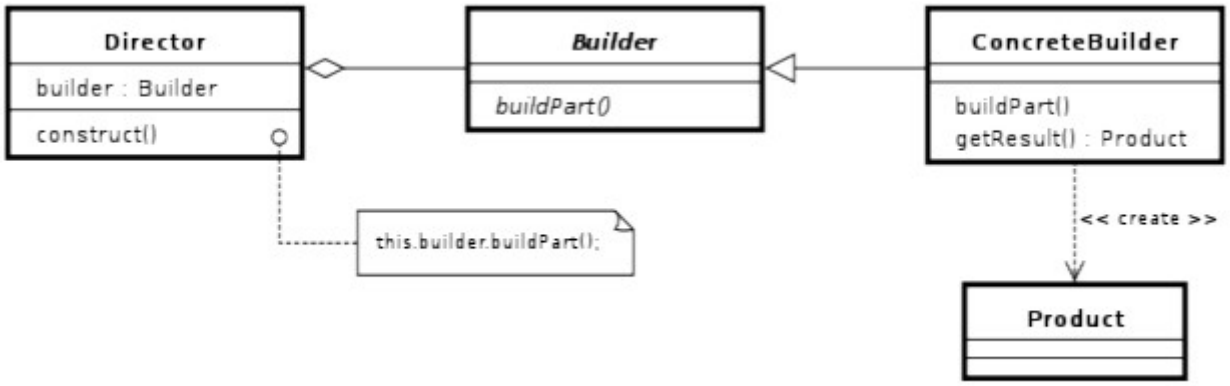
RandomUtilsTest:
Comprova que els randomSet generats a la classe de support RandomUtils siguin correctes.

StandardUserDescriptionBuilderTest:
Comprova que la generació dels Description d'usuaris siguin correctes. Aquest test inclou la prova de tots els aspectes: atributs, followers, nivell d'activitat, etc.

UserUtilsTest:
Comprova la lògica que verifica que un usuari s'ha registrat anteriorment.

PATRONS DE DISSENY

És l'encarregat de generar els "description" dels usuaris. S'ha utilitzat el patró de disseny builder [24] per a crear-los a causa de la naturalesa dels atributs que disposa l'*StandardUserDescription* i aprofitar el polimorfisme.



També es fa ús del patró més comú Factory per al log4j2.

LOG4J2

Els projectes GarlanetSimulator i UserSimulator fan ús opcional de la versió log4j2. S'ha programat un ConnectionFactory que proporciona una connexió a un backend mysql. S'ha configurat al fitxer del log4j2 un appender per a enregistrar els logs a bases de dades mysql.

A més, s'han definit nivells custom de logging amb els seus corresponents nivells de prioritat.

ESTRUCTURA DELS PROJECTES

ActivitySimulator

edu.uoc.activitysimulator.core.ActivitySimulator

Classe principal, és l'encarregada de carregar la configuració *userdescription.properties* i engegar els servidors ActivitySimulatorServer i MetricsReceiverServer.

edu.uoc.activitysimulator.core.StandardUserDescriptionConfig

Classe que representa la configuració carregada del *userdescription.properties*.

edu.uoc.activitysimulator.core.description.IUserDescription

Interfície amb les signatures dels mètodes de les "description" dels usuaris.

edu.uoc.activitysimulator.core.description.StandardUserDescription

Implementa la interfície anterior. Representa les "description" dels usuaris.

edu.uoc.activitysimulator.core.description.StandardUserDescriptionBuilder

És l'encarregat de generar els “description” dels usuaris. S'ha utilitzat el patró de disseny builder [24] per a crear-los a causa de la naturalesa dels atributs que disposa l'*StandardUserDescription* i aprofitar el polimorfisme.
En aquesta classe, i fent ús d'altres, es calcula el nivell d'activitat, followers i following; sent com a resultat una cua de IUserDescription.

edu.uoc.activitysimulator.core.distribution.CustomGeometricDistribution

Implementa la distribució geomètrica utilitzada en la generació de “descriptions”.

edu.uoc.activitysimulator.core.net.ActivitySimulatorServerAPI

Interfície amb les signatures de mètodes per al registre d'usuaris de l'experiment.

edu.uoc.activitysimulator.core.net.ActivitySimulatorChannelInitializer

Classe *Sharable* que hereta de *ChannelInitializer* parametritzada com a *SocketChannel.* És l'encarregada de inicialitzar el canal per al servidor de descriptions. Utilitza el handler *ActivitySimulatorHandler*, un codificador i descodificador d'objectes, ja que és transmetran els *description* per xarxa.

edu.uoc.activitysimulator.core.net.ActivitySimulatorHandler

Sharable handler que hereta de ChannelInboundHandlerAdapter. Implementa la interfície ActivitySimulatorAPI. És el handler encarregat de l'enviament i recepció de missatges (objectes) i d'enregistrar l'usuari a l'experiment.

edu.uoc.activitysimulator.core.net.ActivitySimulatorServer

És el servidor que s'encarregarà d'enviar els descriptions a mesura que rebi les peticions dels clients.

edu.uoc.activitysimulator.core.net.MetricsReceiverChannelInitializer

Classe *Sharable* que hereta de *ChannelInitializer* parametritzada com a *SocketChannel.* És l'encarregada de inicialitzar el canal per al servidor de recollida de traces. Utilitza el handler MetricsReceiverHandler, un codificador i descodificador de Strings.

edu.uoc.activitysimulator.core.net.MetricsReceiverHandler

Sharable handler que hereta de ChannelInboundHandlerAdapter. És el handler encarregat de la recepció de traces.

edu.uoc.activitysimulator.core.net.MetricsReceiverServer

És el servei encarregat d'inserir les traces rebudes a una base de dades sqlite.

edu.uoc.activitysimulator.core.quantity.ActivityLevelQuantity

Calcula el nivell d'activitat dels usuaris.
Es fan servir les llibreries de suport org.apache.commons.math3 i org.apache.commons.lang3, per a les distribucions uniformes i els rangs respectivament.

edu.uoc.activitysimulator.core.quantity.FollowersQuantity

Calcula la quantitat de followers dels usuaris.
Es fan servir les llibreries de suport org.apache.commons.math3 i org.apache.commons.lang3, per a les distribucions uniformes i els rangs respectivament.

edu.uoc.activitysimulator.core.range.IRange

Interfície amb les signatures dels mètodes que representen un rang. S'ha utilitzat els genèrics de Java per a realitzar rangs de tipus qualsevol [28].

edu.uoc.activitysimulator.core.range.RangeFollowers

Representa el rang dels followers. S'ha utilitzat els genèrics de Java per a realitzar rangs de tipus qualsevol. Fa referència a la taula de l'explicació de generació d'activitat (Veure apartats següents).

edu.uoc.activitysimulator.core.range.RangeFollowersCategory

Representa el rang entre els followers i la categoria d'activitat que els hi pertoca. S'ha utilitzat els genèrics de Java per a realitzar rangs de tipus qualsevol. Fa referència a la taula de l'explicació de generació d'activitat (Veure apartats següents).

edu.uoc.activitysimulator.util.IOUtils

Classe de suport que facilita el tancament de streams i escriptura de certificats/keystores a fitxer.

edu.uoc.activitysimulator.util.RandomUtils

Classe de suport que retorna un set d'enters.

edu.uoc.activitysimulator.util.SQLiteUtils

Classe de suport per gestionar l'ús de sqlite. Utilitzada pel MetricsReceiverServer per crear i desar les mètriques rebudes.

edu.uoc.activitysimulator.util.SSLUtils

Classe de suport per generar els certificats, keystore i joc de claus dels usuaris. S'ha fet servir bouncycastle com a security provider.

log4j.properties

Fitxer de configuració del log4j. Es defineixen els appender necessaris.

service.properties

Fitxer properties on s'especifica:

- Port del servei ActivitySimulator: service.port
- Activació de la comunicació SSL del servei ActivitySimulator: service.ssl
- Port del servei de recollida de mètriques MetricsReceiverServer: metrics.port
- Activació de la comunicació SSL del servei MetricsReceiverServer: metris.ssl
- Nom de la base de dades SQLite on es desen les mètriques: sqlite.db

userdescription.properties

- Common Name utilitzat com a issuer per a la generació del certificat X509: ISSUER
- Common Name utilitzat com a subject per a la generació del certificat X509: SUBJECT
- Longitud dels noms d'usuari generats: USERNAME_LENGTH
- Longitud de les paraules de pas dels usuaris generats: PASSWORD_LENGTH
- Longitud del nom de correu dels usuaris generats: EMAIL_LENGTH
- Longitud del domini del correu dels usuaris generats: DOMAIN_LENGTH
- Domini del correu dels usuaris generats: DOMAIN
- Algorisme per a la generació del parell de claus: keyPairAlgorithm
- Algorisme per a la generació dels nombres aleatòris utilitzats en la generació de les claus: KEY_RNGAlgorithm.
- Algorisme utilitzat per la signatura: signatureAlgorithm
- ID d'inici per a la generació d'usuaris: start_uid
- Nombre d'usuaris de l'experiment (nombre de descriptions a crear): users

edu.uoc.activitysimulator.ActivityLevelQuantityTest

Test unitari per a la classe ActivityLevelQuantity.

edu.uoc.activitysimulator.ActivitySimulatorTest

Test unitari per a la classe ActivitySimulator.

edu.uoc.activitysimulator.CustomGeometricDistributionTest

Test unitari per a la classe CustomGeometricDistribution.

edu.uoc.activitysimulator.FollowersQuantityTest

Test unitari per a la classe FollowersQuantity.

edu.uoc.activitysimulator.IOUtilsTest

Test unitari per a la classe IOUtils. Sortint de l'àmbit de la generació de tests unitaris per a comprovar la part funcional, aquesta classe va ser la primera a codificar per comprovar la correcta autogeneració dels certificats X509, parell de claus i keystores.

edu.uoc.activitysimulator.RandomUtilsTest

Test unitari per a la classe RandomUtils

edu.uoc.activitysimulator.StandardUserDescriptionBuilderTest

Test unitari per a la classe StandardUserDescriptionBuilder.

pom.xml

Project Object Model per a Maven.

GarlanetSimulator

edu.uoc.garlanetsimulator.core.logging.DBLoggingConnectionFactory

Connection Factory implementat de l'appender de BDD per a l'ús opcional del log4j2.

edu.uoc.garlanetsimulator.core.net.GarlanetSimulatorLoginServerHandler

Handler per a la gestió dels login dels usuaris.

edu.uoc.garlanetsimulator.core.net.GarlanetSimulatorMessageServerHandler

Handler per a la gestió dels missatges dels usuaris.

edu.uoc.garlanetsimulator.core.net.GarlanetSimulatorRegisterServerHandler

Handler per a la gestió dels registres dels usuaris.

edu.uoc.garlanetsimulator.core.net.GarlanetSimulatorServer

Servei que simula Garlanet.

edu.uoc.garlanetsimulator.core.net.GarlanetSimulatorServerChannelInitializer

Classe *Sharable* que hereta de *ChannelInitializer* parametritzada com a *SocketChannel.* És l'encarregada d'inicialitzar el canal per al servei que simula Garlanet.

edu.uoc.garlanetsimulator.users.RegisteredUser

Classe que representa a l'usuari registrat al GarlanetSimulator.

edu.uoc.garlanetsimulator.users.UserUtils

Classe de suport amb mètodes útils per a la gestió dels usuaris registrats. Comprova registres, logins, Gaf's, i logouts.

edu.uoc.activitysimulator.GarlanetSimulator

Conté el mètode principal, s'encarrega de carregar els properties i engegar els serveis.

edu.uoc.garlanetsimulator.UserUtilsTest

Test unitari per a la classe UserUtils.

log4j2.xml

Fitxer de configuració del log4j2. Es defineixen els appender necessaris. (opcional).

startGarlanetSimulator.sh

Shell script que facilita l'execució de l'aplicació una vegada empaquetada. S'encarrega de validar que estiguin definides les variables d'entorn, la versió mínima de java, i llençar l'aplicació amb els flags correctes.

service.properties

- Port del servei GarlanetSimulator: service.port
- Activació de la comunicació SSL del servei GarlanetSimulator: service.ssl
- Nombre d'usuaris de l'experiment: users
- Nom de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.host
- Port de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.port
- Nom de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.database
- Usuari de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.user
- Clau de pas de l'usuari de la BDD per a l'appender *DBLoggingConnectionFactory* (opcional): db.pwd

create_table.sql

Script sql per a construir la taula de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional).

export_table.sql

Script sql per a exportar a csv la taula de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional).

pom.xml

Project Object Model per a Maven.

UserSimulator

edu.uoc.usersimulator.core.logging.DBLoggingConnectionFactory

Connection Factory implementat de l'appender de BDD per a l'ús opcional del log4j2.

edu.uoc.usersimulator.core.net.MetricsReceiverChannelInitializer

Inicialitza el channel per als MetricsReceiverClient.

edu.uoc.usersimulator.core.net.MetricsReceiverClient

Client que envia les traces cap al MetricsReceiverServer del ActivitySimulator.

edu.uoc.usersimulator.core.net.MetricsReceiverHandler

Handler per al MetricsReceiverClient.

edu.uoc.usersimulator.core.net.UserDescriptionChannelInitializer

Inicialitza el channel per al UserDescriptionClient.

edu.uoc.usersimulator.core.net.UserDescriptionClient

Client que sol·licita el description al ActivitySimulatorServer.

edu.uoc.usersimulator.core.net.UserDescriptionHandler

Handler per al UserDescriptionClient.

edu.uoc.usersimulator.core.net.UserGarlanetSimulatorAPI

Interfície amb la signatura de mètodes que especifica les accions que poden realitzar els usuaris (login, logout, post, register, etc.).

edu.uoc.usersimulator.core.net.UserGarlanetSimulatorChannelInitializer

Inicialitza el channel per al UserGarlanetSimulatorClient.

edu.uoc.usersimulator.core.net.UserGarlanetSimulatorClient

S'encarrega de interaccionar amb el GarlanetSimulatorServer. És qui envia els missatges i les traces.

edu.uoc.usersimulator.core.net.UserGarlanetSimulatorHandler

Handler per al UserGarlanetSimulatorClient.

edu.uoc.usersimulator.core.UserSimulator

La classe principal. Carrega la configuració i inicia els serveis.

log4j2.xml

Fitxer de configuració del log4j2. Es defineixen els appender necessaris. (opcional).

startUserSimulator.sh

Shell script que facilita l'execució de l'aplicació una vegada empaquetada. S'encarrega de validar que estiguin definides les variables d'entorn, la versió mínima de java, i llençar l'aplicació amb els flags correctes.

descriptionService.properties

- Host del servei ActivitySimulatorServer a connectar: server.host
- Port del servei ActivitySimulatorServer a connectar: server.port
- Habilitar SSL si el server ActivitySimulatorServer en fa ús: server.ssl
- Host del servei MetricsReceiverServer a connectar: metrics.host
- Port del server MetricsReceiverServer a connectar: metrics.port

garlanetService.properties

- Host del servei GarlanetSimulatorServer per a connectar: server.host
- Port del servei GarlanetSimulatorServer per a connectar: server.port
- Habilitar SSL si el server GarlanetSimulatorServer en fa ús: server.ssl
- Durada de l'experiment (en milisegons): experiment.time
- Nom de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.host
- Port de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.port
- Nom de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.database
- Usuari de la BDD per a l'appender de *DBLoggingConnectionFactory* (opcional): db.user
- Clau de pas de l'usuari de la BDD per a l'appender *DBLoggingConnectionFactory* (opcional): db.pwd

users.properties

- Usuaris per JVM: jvm.users
- Compressió de temps: time.compression

GESTIÓ DEL TEMPS

L'aplicació permet definir temps real i temps de simulació. Es configura al fitxer users.properties, atribut time.compression, valors possibles: 0 (temps real) o 1 (temps de simulació).

L'equivalència en temps de simulació és de: 1 dia (real) = 1 hora (simulació).

El temps (ms) que es defineix al fitxer users.properties s'especifica en temps real, l'aplicació ja s'encarrega de fer els càlculs necessaris per a transformar-lo en temps de simulació en el cas de què s'hagi habilitat.

Temps real

EXPERIMENT_TIME (ms): carregat del users.properties
MESSAGES_NUMBER: # msg * # dies
TIME_BETWEEN_MSG: 1 day (ms) / #msg

Exemple:

EXPERIMENT_TIME (ms): 8 640 000 (1 dia) ms
MESSAGES_NUMBER: 10 * 1 = 10
TIME_BETWEEN_MSG: 8 640 000 / 10 = 8 640 000 ms

Exemple:

EXPERIMENT_TIME (ms): 216 000 000 (2,5 dies) ms
MESSAGES_NUMBER: 10 * 2.5 = 25
TIME_BETWEEN_MESSAGES: 8 640 000 / 10 = 8 640 000 ms

Temps de simulació

EXPERIMENT_TIME (ms): carregat del users.properties / 24 hores
MESSAGES_NUMBER: # msg * # hores
TIME_BETWEEN_MSG: 1 hora (ms) / # msg

Exemple:

EXPERIMENT_TIME (ms): 8 600 000 / 24 = 3 600 000 ms
MESSAGES_NUMBER: 10 * 1 = 10
TIME_BETWEEN_MSG: 3 600 000 / 10 = 360 000 ms

Exemple:

EXPERIMENT_TIME (ms): 216 000 000 / 24 = 9 000 000 ms
MESSAGES_NUMBER: 10 * 2.5 = 25
TIME_BETWEEN_MESSAGES: 3 600 000 / 10 = 360 000 ms

LIMITS.CONF

Durant la realització dels experiments, s'obren molts de sockets. Aquest fet pot provocar l'error: “Too many open files”. Per solucionar-ho, es pot crear un fitxer del tipus:
/etc/security/limits.d/sim.conf

amb contingut:

```
simuser        hard    nofile  500000
simuser        soft    nofile  500000
```

Els límits dependran molt de l'experiment a realitzar. S'ha comprovat amb lsof que els sockets es tanquen i gestionen correctament per netty. La conclusió és definir un pool de connexions més petit o configurar els limits.

PROCÉS DE DESENVOLUPAMENT

Durant el desenvolupament del projecte, s'ha fet servir Git per al manteniment del codi. Sent un repositori privat amb bitbucket l'origen remot.

S'han definit tres branques:

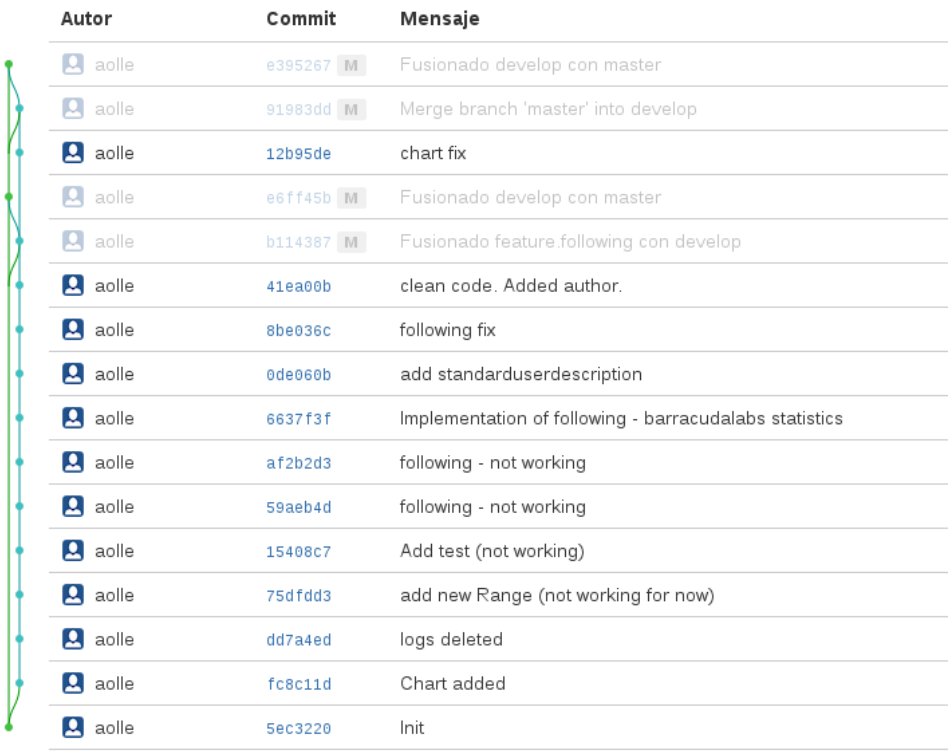
- master
- development
- features

La branca temporal de features ha anat incloent totes les millores que s'han desenvolupat a partir de la branca de development. Aquesta branca features s'ha eliminat al finalitzar aquest TFM. Quedant com a resultats:

- master
- development

Tags:

- 0.1 – Projecte amb totes les funcionalitats implementades. Following distribuïts de manera lineal.
- 0.2 – Projecte amb totes les funcionalitats implementades. Following distribuïts segons estudi de barracudalabs.



Autor	Commit	Mensaje
aolle	e395267 M	Fusionado develop con master
aolle	91983dd M	Merge branch 'master' into develop
aolle	12b95de	chart fix
aolle	e6ff45b M	Fusionado develop con master
aolle	b114387 M	Fusionado feature.following con develop
aolle	41ea00b	clean code. Added author.
aolle	8be036c	following fix
aolle	0de060b	add standarduserdescription
aolle	6637f3f	Implementation of following - barracudalabs statistics
aolle	af2b2d3	following - not working
aolle	59aeb4d	following - not working
aolle	15408c7	Add test (not working)
aolle	75dfdd3	add new Range (not working for now)
aolle	dd7a4ed	logs deleted
aolle	fc8c11d	Chart added
aolle	5ec3220	Init

MAVEN: CONFIGURACIÓ, PLUGINS I DEPENDÈNCIES

Les dependències s'han gestionat amb Maven, tots els projectes inclouen el pom.xml corresponent.

Els projectes tenen les dependències següents:

ActivitySimulator:
- JDK 1.7
- Apache Commons Configuration 1.10
- Apache Commons Lang 3.3.2
- Apache Commons Math3 3.4.1
- Apache Commons IO 2.4
- Apache Log4j RELEASE
- Netty IO 4.0.26 Final
- Bouncycastle 1.47
- JUnit 4.12
- Mockito 1.9.5
- PowerMock 1.6.1
- SQLite 3.8.7

GarlanetSimulator:
- JDK 1.7
- Apache Commons Configuration 1.10
- Apache Commons Lang 3.3.2
- Apache Commons Pool 1.6
- Apache Commons DBCP 1.4
- Apache Log4j2 2.2
- Netty IO 4.0.26 Final
- JUnit 4.12
- Mockito 1.9.5
- PowerMock 1.6.1
- MySQL Connetor 5.1.35

UserSimulator:
- JDK 1.7
- Apache Commons Configuration 1.10
- Apache Commons Pool 1.6
- Apache Commons DBCP 1.4
- Apache Log4j2 2.2
- Netty IO 4.0.26 Final
- Netty IO 4.0.26 Final

UserSimulator també conté com a dependències les classes següents:
- edu.uoc.activitysimulator.core.description.IUserDescription
- edu.uoc.activitysimulator.core.description.StandardUserDescription
- edu.uoc.activitysimulator.util.SSLUtils

Aquestes classes són necesàries perquè IUSerDescription i StandardUserDescription s'envien s'obtenen serialitzades de l'ActivitySimulator i s'utilitza un ObjectDecoder per a la seva descodificació. Si es fa alguna modificació a aquestes classes originals de l'ActivitySimulator, s'ha de tenir en compte l'actualització de la dependència.

Per al goal 'build' s'han inclòs les classes, els properties i els scripts bash de cada projecte.

S'ha utilitzat el plugin maven-shade-plugin (2.3), maven-surefire-plugin (2.12.4) i maven-jar-plugin (2.4), per a la creació de fat JARs, gestió dels includes dintre del JAR i proves unitàries durant el build.
La creació d'un fat JAR és opcional, es poden fer la gestió de dependències de la manera que es vulgui. S'ha de tenir en compte les llicències de les llibreries en cap d'optar pel fat JAR.

SCRIPTS D'ENGEGADA

Cada projecte disposa del seu propi script d'engegada que facilita l'execució de les aplicacions un cop han estat empaquetades com a fitxers JAR (Java ARchive). Aquests són:

- ActivitySimulatorServer: startActivitySimulator.sh
- GarlanetSimulatorServer: startGarlanetSimulator.sh
- UserSimulator: startUserSimulator.sh

Els flags i opcions que aplica l'script a la JVM són:

Script	Flags i opcions
startActivitySimulator.sh	-server -jar -Dlog4j.configuration
startGarlanetSimulator.sh	-server -jar -Dlog4j.configurationFile
startUserSimulator.sh	-jar -Djava.security.egd=file:/dev/./urandom -Dlog4j.configurationFile

Netty utilitza SecureRandom per a la generació de nombres aleatoris (RNG). En sistemes Unix-like, el Oracle JRE utilitza /dev/random per obtenir l'entropia per al SecureRandom. El /dev/random pot bloquejar-se si no té disponible la suficient entropia i causar el warning següent en netty:

WARNING: Failed to generate a seed from SecureRandom within 3 seconds. Not enough entrophy?

Per a solucionar-ho es passa com a opció a la JVM el entropy gathering device. Tal com especifiquen a: https://github.com/netty/netty/issues/3419

En el cas particular, no ha funcionat l'alternativa: -Dio.netty.initialSeedUniquifier.

EMPAQUETAR ELS PROJECTES I EXECUTAR-LOS

Per a fer el build del projecte necessitem Maven. La comanda per fer el build, estant situat a dintre de la carpeta principal del projecte, és:

*mvn clean; mvn package; cd target; chmod +x *.sh ;*

Això ens deixarà al directori target, on tindrem l'script bash per executar el projecte.

Haurem de fer aquest pas per a cadascun dels projectes: ActivitySimulator, GarlanetSimulator i UserSimulator.

L'ordre d'execució és: ActivitySimulator, GarlanetSimulator i UserSimulator. No s'ha d'aixecar el UserSimulator fins que el procés d'ActivitySimulator hagi fet el bind del port (després de calcular els description).

GENERACIÓ DELS RESULTATS

Quan fem el build de l'ActivitySimulator, al directori target se'ns generarà la base de dades (per defecte metrics.db) i disposarem de l'script chart.sh que conté tota la lògica per a la generació de l'html resultant a partir de consultes SQL i càlculs matemàtics. S'ha de disposar de sortida a internet per veure els resultats de manera correcta. El motiu és l'ús de les llibreries dinàmiques javascript de Google Charts.

Per al correcte funcionament, es necessita tenir instal·lada la dependència: sqlite3.

L'script charts.sh ha de ser editat per a introduir les constants del nostre experiment, aquestes són:

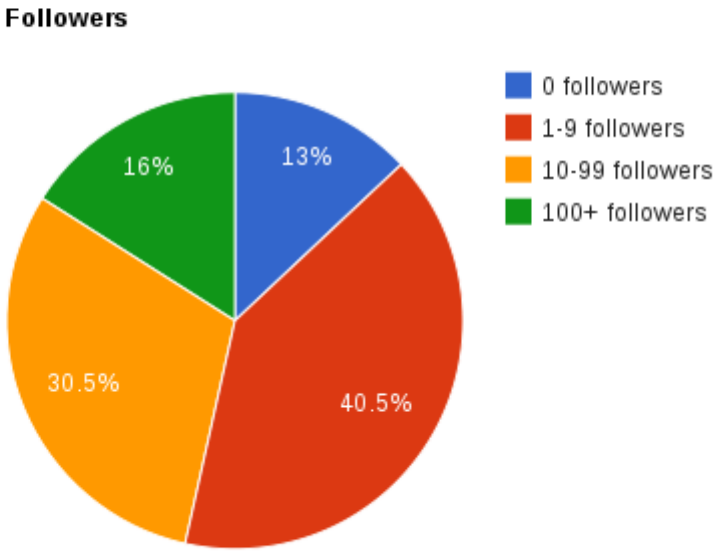
```
UIDINI=500     # ID inicial dels UserDescription
TOTAL=200      # Nombre total d'usuaris
EXPERIMENT_NAME="EXPERIMENT 2"  # Nom de l'experiment
DB=metrics.db  # Base de dades sqlite3
DAYS=1         # Nombre de dies de l'experiment
D3="false"     # Generar un gràfic 3D. Per defecte es generen gràfics en 2D.
```

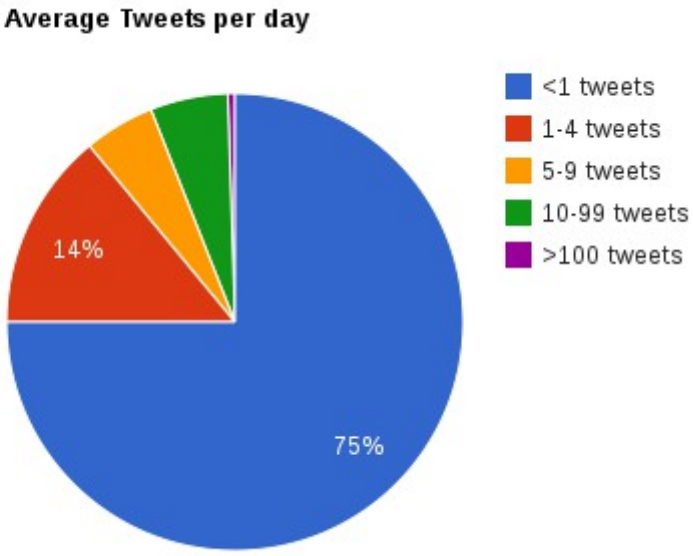
VALIDACIÓ DEL MODELAT DE TWITTER

S'han realitzat diversos experiments dels quals es mostren els resultats propers a l'estudi de barracudalabs. La quantitat d'usuaris durant l'experiment ha estat limitada per les limitacions físiques computacionals des d'on han estat executats.

EXPERIMENT 1

Dies: 1
Usuaris: 200
Missatges totals: 821

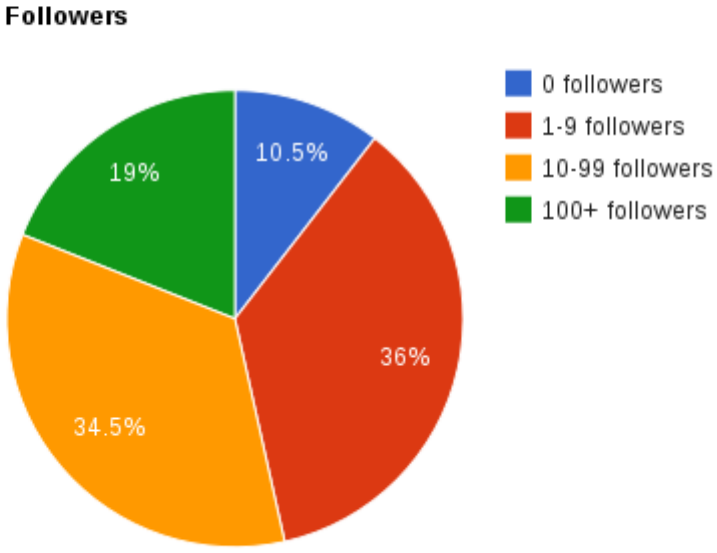


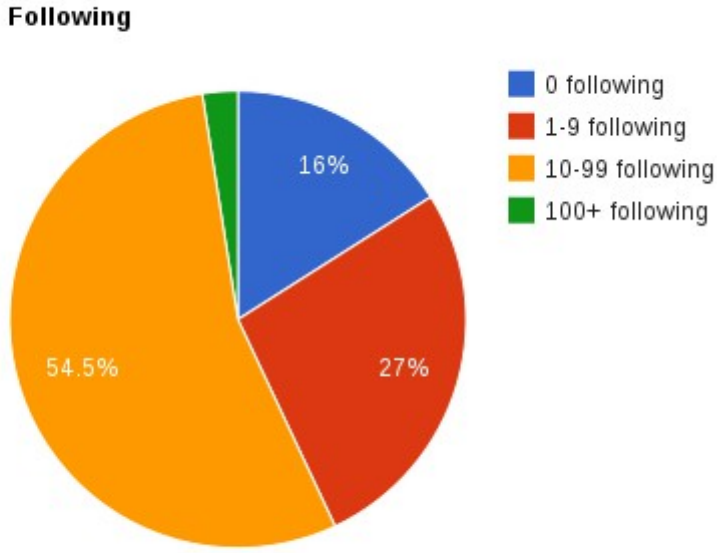


5-9 tweets: 5% ; 10-99 tweets: 5.5%; >100 tweets: 0,5%

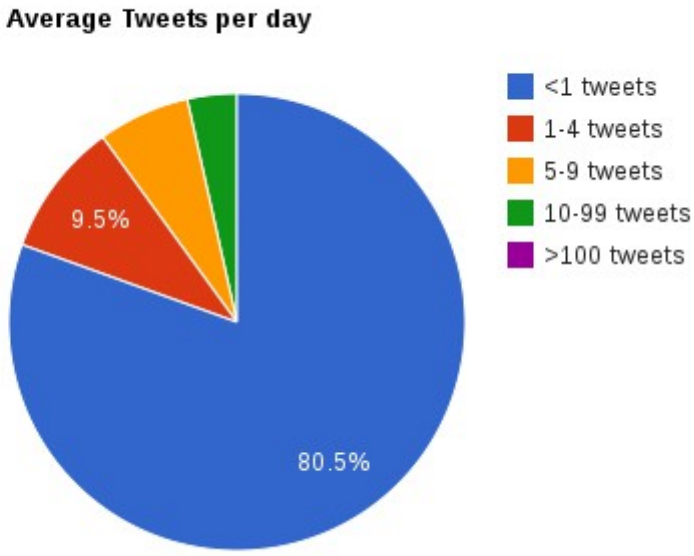
EXPERIMENT 2

Dies: 1
Usuaris: 200
Missatges totals: 552





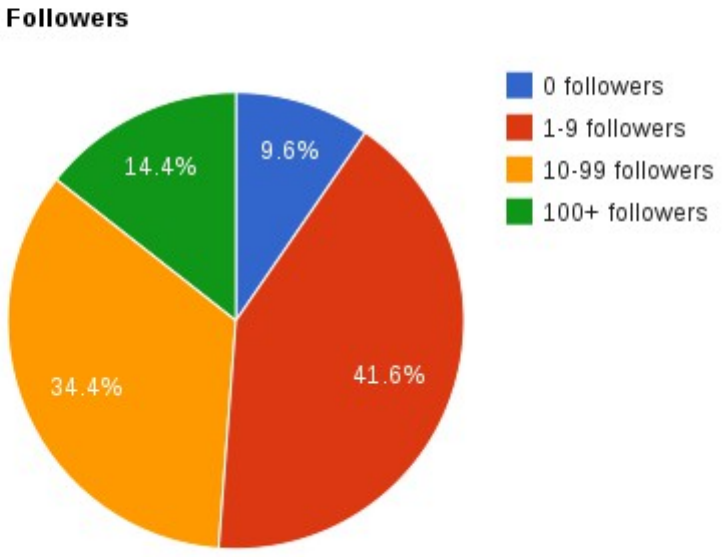
100+ following: 2,5%

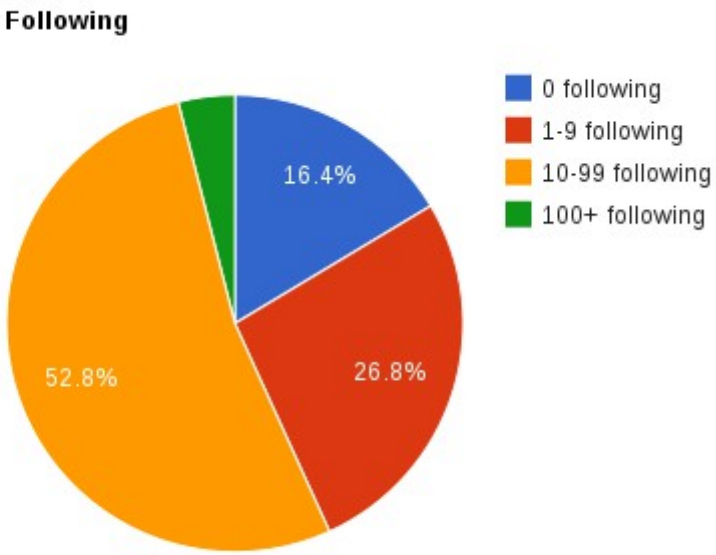


5-9 tweets: 6,5%; 10-99 tweets: 3,5%; >100 tweets: 0%

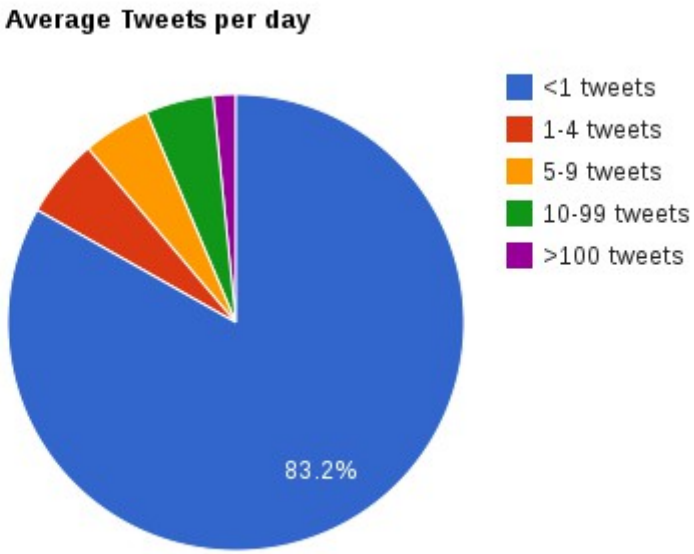
EXPERIMENT 3

Dies: 2
Usuaris: 250
Missatges totals: 1348





100+ following: 4%



1-4 tweets: 5,6%; 5-9 tweets: 4,8%; 10-99 tweets: 4,8%; >100 tweets: 1,6%

EXPERIMENTS DEFINITS PER A GARLANET

Hipòtesis
En els experiments exposats a continuació, es defineixen les dues hipòtesis següents:

- Els usuaris perceben que reben els missatges immediatament després d'enviar-se. Com a immediatesa es coneix com el temps que triga un missatge desque s'envia fins que arriba als seus receptors.
- Els usuaris reben un percentatge dels missatges que se'ls han enviat per part d'altres usuaris.

Experiment 1: mesura de la fiabilitat de la plataforma.

En aquest experiment s'espera mesurar el percentatge dels missatges rebuts per part d'un usuari respecte als missatges que realment hauria d'haver rebut. A continuació es detallen les condicions en les quals es basarà l'experiment.

- ***Quantitat d'usuaris***

Per comprovar l'efecte que té el nombre d'usuaris al comportament de Garlanet, es provarà amb uns nombres d'usuaris de 50, 100, 1000, 10000 i 100000 usuaris.

- ***Valors dels paràmetres de l'aplicació usuari***

S'utilitza l'application amb els valors del parameters.properties següents:

Temps:

Paràmetre	Valor (ms)
time_timeline_session	30000
time_messages_session	30000
time_timeline_update	30000
time_profile_update	30000
time_following_update	3600000
networkTimeuot	1000

Valor de paginació: 20.

Booleans:

Paràmetre	Valor (c/f)
messages_update	Fals
timeline_update	Cert
profile_update	Cert
following_update	Fals

Nombre de repositoris

Es realitzarà l'experiment amb un percentatge diferent d'usuaris que aporten repositoris a Garlanet per veure quin efecte té en el comportament de la plataforma: 5%, 10%, 20%, 50% i 80%.

Valors dels paràmetres dels repositoris

Parámetre	**Valor**
TimeMyServicesCheck	50000
TimeIAmAlive	60000
TimeValidateVersion	604800000
consistency_sessions	60000
MaxtimeoutDSServer	30240000
LookupServiceExpiration	120000
LookupServiceDuration	120000
NetworkTimeout	500
DownloadSpeed	18.48 MB
UploadSpeed	0.98 MB
MaxServices	30

Grau de replicació dels repositoris

Provarem amb diferents graus de replicació per veure quin efecte té en el rendiment de Garlanet i en la disponibilitat dels missatges: 3, 5, 7 i 10.

Durada de la simulació

L'experiment tindrà dues fases:

- Fase 1 (warm up): període de temps que necessita el sistema per estar en un estat realista. 1 dia.
- Fase 2: es prenen mesures. 7 dies.

Compressió del temps: Sí.

Nombre de mostres

L'experiment es reproduirà 100 cops.

Entorn d'execució

L'experiment es farà en diferents entorns:

- local (clúster)
- testbed Internet (PlanetLab)
- testbed xarxa comunitària (CommunityLab)

Mesures

En aquest experiment es mesurarà que els missatges enviats pels usuaris als seus seguidors, arribin als seus destinataris correctament.

Com es mesurarà

Quan un usuari del sistema enviï un missatge, es generarà un registre al log aplicatiu de la forma:

Usuari origen – usuari destí – Identificador del missatge – Tipus (enviament/recepció)

De la mateixa manera, quan un usuari rebi un missatge, es generarà un registre al log aplicatiu amb la mateixa forma que l'anterior.

D'aquesta manera, mitjançant una anàlisi dels logs, es poden emparellar les traces que tinguin el mateix identificador de missatge, de manera que si alguna traça del tipus enviament, no es troba la traça de tipus recepció, significaria que el missatge no s'ha rebut correctament pel seu destinatari.

Es generarà una traça per cada enviament, independentment de què el missatge sigui el mateix, és a dir, si l'usuari envia el mateix missatge a tres persones, es generen tres traces per poder-les emparellar amb les traces de recepció.

Experiment 2: Immediatesa: temps que triga un missatge desque s'envia fins que arriba a tots els destinataris.

En aquest experiment s'espera mesurar el temps que triga un missatge a ser rebut pel destinatari des del moment en què és enviat per l'emissor.

Quantitat d'usuaris

Per comprovar l'efecte que té el nombre d'usuaris al comportament de Garlanet, es provarà amb uns nombres d'usuaris de 50, 100, 1000, 10000 i 100000 usuaris.

Valors dels paràmetres de l'aplicació usuari

S'utilitza l'application amb els valors del parameters.properties següents:

Temps:

Paràmetre	Valor (ms)

time_timeline_session	30000
time_messages_session	30000
time_timeline_update	30000
time_profile_update	30000
time_following_update	3600000
networkTimeuot	1000

Valor de paginació: 20.

Booleans:

Paràmetre	**Valor (c/f)**
messages_update	Fals
timeline_update	Cert
profile_update	Cert
following_update	Fals

Nombre de repositoris

Es realitzarà l'experiment amb un percentatge diferent d'usuaris que aporten repositoris a Garlanet per veure quin efecte té en el comportament de la plataforma: 5%, 10%, 20%, 50% i 80%.

Valors dels paràmetres dels repositoris

Parámetre	**Valor**
TimeMyServicesCheck	50000
TimeIAmAlive	60000
TimeValidateVersion	604800000
consistency_sessions	60000
MaxtimeoutDSServer	30240000
LookupServiceExpiration	120000
LookupServiceDuration	120000
NetworkTimeout	500
DownloadSpeed	18.48 MB
UploadSpeed	0.98 MB
MaxServices	30

Grau de replicació dels repositoris

Provarem amb diferents graus de replicació per veure quin efecte té en el rendiment de Garlanet i en la disponibilitat dels missatges: 3, 5, 7 i 10.

Durada de la simulació

L'experiment tindrà dues fases:

- Fase 1 (warm up): període de temps que necessita el sistema per estar en un estat realista. 1 dia.
- Fase 2: es prenen mesures. 7 dies.

Compressió del temps: Sí.

Nombre de mostres

L'experiment es reproduirà 100 cops.

Entorn d'execució

L'experiment es farà en diferents entorns:
- local (clúster)
- testbed Internet (PlanetLab)
- testbed xarxa comunitària (CommunityLab)

Mesures

En aquest experiment es mesura el temps que triga un missatge a ser rebut pel destinatari.

Com es mesurarà

Quan un usuari del sistema enviï un missatge, es generarà un registre al log aplicatiu de la forma:

Usuari origen – usuari destí – Identificador del missatge – Tipus (enviament/recepció) - Timestamp

De la mateixa manera, quan un usuari rebi un missatge, es generarà un registre al log aplicatiu amb la mateixa forma que l'anterior.

D'aquesta manera, mitjançant una anàlisi dels logs, es poden emparellar les traces que tinguin el mateix identificador de missatge i es poden resoldre els temps a partir de la diferència dels timestamp.

MILLORES I TREBALL FUTUR

És plantegen els següents aspectes com a possibles millores, treball futur i/o idees que es poden considerar:

- Optimitzar el rendiment del MetricsReceiver. Actualment, té un retard bastant important en escriure a la base de dades sqlite a mesura que van arribant els missatges. Aquesta base de dades pot ser llegida per múltiples fils però només pot ser escrita per un fil a la vegada.
Aquest fet fa que s'encuen peticions d'inserció a la base de dades de missatges enviats.
Podria ser que Garlanet, en fer ús d'aquest tipus de base de dades, pugui tenir un coll d'ampolla en un futur que suposi un volum d'usuaris importants del sistema.

- Segons referència [21] del "Twitter Engineering Blog", la implementació de Netty en les comunicacions es tradueix a una comunicació molt més ràpida que en Apache MINA. Es pot considerar implementar la communication layer amb aquest framework.

- El projecte està preparat per acceptar implementar la comunicació via SSL. No està provat, però si es considera que a nivell de capa de comunicació estaria bé afegir seguretat, es pot implementar.

- Per a l'ús de claus de pas en aplicacions Java, es recomana l'ús de char[] en lloc de Strings. Això pot aplicar en les aplicacions implementades en aquest projecte i al mateix Garlanet. És una de les moltes recomanacions de la guia de referència JCA [22]. Entre altres [23].

- Refactoritzar el codi. Si és possible.

- Millorar el rendiment, detectar colls d'ampolla i solucionar-los.

- Implementar la simulació de repositoris.

- Desplegaments a màquines més potents i/o distribuïdes per a la realització d'experiments més grans. Execució de testbeds a PlanetLab i communityLab.

- Utilitzar l'ActivitySimulator i l'UserSimulator directament amb Garlanet.

REFERÈNCIES

[1] D. Lazaro, J.M. Marques, G. Cabrera, E. Rifa-Pous, A. Montane, "HorNet: Microblogging for a Contributory Social Network," *IEEE Internet Computing* vol.16 N.3 Pg.37-44 , Jun. 2012.

[2] BOINC, "Berkeley Open Infrastructure for Network Computing," http://boinc.berkeley.edu/ , 2002-2014.

[3] R.D.W. Perera, S. Anand, K.P. Subbalakshmi, R. Chandramouli, "Twitter Anallytics: Architecture, Tools and Anallysis," *MILCOM Military Communications Conference*. 2010.

[4] B. Goncalves, N. Perra, A. Vespignani, "Modeling User's Activity on Twitter Networks: Validation of Dunbar's Number," *PLoS ONE*. Aug. 2011.

[5] Barracuda Labs, "Barracuda Labs 2010 Annual Security Report," *barracudalabs.com*. 2010.

[6] D. Quercia, M. Kosinski, D. Stillwell, J. Crowcroft, "Our Twitter Profiles, Our Selves: Predicting Personality with Twitter," *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International*

Conference. 2011.

[7] T. Yamashita, H. Sato, S. Oyama, M. Kurihara, "Classification of Twitter Users Based on Following Relations," *International MultiConference of Engineers and Computer Scientists 2013*. 2013.

[8] P. Pena, R. del Hoyo, J. Vea-Murguia, C. Gonzalez, S. Mayo, "Collective Knowledge Ontology User Profiling for Twitter," *12th IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT).* 2013.

[9] S. Tramp, P. Frischmuth, T. Ermilov, S. Shekarpour, S. Auer, "An Architecture of a Distributed Semantic Social Network," *IOS Press.* 2012.

[10] K. Tao, F. Abel, Q. Gao, G. Houben, "TUMS: Twitter-based User Modeling Service," *8th Extended Semantic Web Conference (ESWC 2011).* 2011.

[11] D. Kondo, B. Javadi, A. Iosup, D. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid).* 2010.
[12] Delft University, INRIA, University of Western Sydney, The University of Heidelberg, "Failure Trace Archive," *http://fta.scem.uws.edu.au* . 2009-2014.

[13] M. Quinson, A. Legrand, "Modeling Large Scale Systems and Validating their Simulators," *grid5000.fr.* 2013.

[14] D. Lazaro, D. Kondo, J.M. Marques, "Long-term availability prediction for groups of volunteer resources," *Journal of Parallel and distributed computing – ACADEMIC PRESS INC ELSEVIER SCIENCE*. 2012.

[15] Wikipedia, "Uniform distribution", http://en.wikipedia.org/wiki/Uniform_distribution_%28continuous%29, 2015

[16] Wikipedia, "Geometric Distribution", http://en.wikipedia.org/wiki/Geometric_distribution, 2015

[17] Javadi B, Kondo D, Vincent JM, Anderson D.P, "Discovering statistical models of availability in large distributed systems: An empirical study of SETI@home", IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 11, November 2011.

[18] Wikipedia, "Weibull distribution", http://en.wikipedia.org/wiki/Weibull_distribution, 2015

[19] Wikipedia, "Log-normal distribution", http://en.wikipedia.org/wiki/Log-normal_distribution, 2015.

[20] Apache Commons, "Apache Commons Mathematics Library", http://commons.apache.org/proper/commons-math/, 2015.

[21] Twitter Engineering blog, "Twitter Search is Now 3x Faster", https://blog.twitter.com/2011/twitter-search-now-3x-faster

[22] Oracle Java SE Documentation, “Java Cryptography Architecture (JCA) Reference Guide”, http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html

[23] Javarevisited, “Best Practices while dealing with Password in Java”, http://javarevisited.blogspot.com.es/2012/05/best-practices-while-dealing-with.html

[24] Wikipedia, “Builder pattern”, http://en.wikipedia.org/wiki/Builder_pattern

[25] Manning, “Netty in Action”, http://www.manning.com/maurer/

[26] Precise Java, “Best Practices to improve Performance in J2SE”, http://www.precisejava.com/

[27]

[28] Oracle Java Documentation, “The Java Tutorials – Lesson: Generics”, https://docs.oracle.com/javase/tutorial/extra/generics/index.html