UNIVESITÁ DEGLI STUDI DI MILANO
LAUREA TRIENNALE IN COMUNICAZIONE DIGITALE
PROGETTO LABORATORIO DI RETI DI CALCOLATORI

Contagion: il tuo amico è il vero nemico ...etchi

Matteo Zignani

4 gennaio 2014

1 PRESENTAZIONE DEL PROBLEMA

Alzi la mano chi non ha mai sognato di infettare i propri amici con virus più o meno letali. Spero nessuno, anche se il progetto di questo appello richiede di sviluppare un servizio social che permette di simulare un contagio mirato di alcuni virus tra i propri amici. Ogni utente può scegliere di infettare un amico alla volta scegliendo fra un insieme di virus più o meno letali. Naturalmente ogni utente è vaccinato oppure è resistente ad un sottoinsieme di virus. Tale informazione è nota solo all'utente stesso e non ai suoi amici. All'inizio del gioco ogni utente contrae un determinato virus e durante la sua vita può diffondere solo virus che prima a contratto. Nelle successive sezioni verranno speigate in dettaglio tutte le regole per contagiare e quali risorse vengono messe a disposizione per poter giocare a Contagion: il tuo amico è il vero nemico ...etchi
Il servizio che lo studente deve implementare si basa su un'architettura client/server. Il server deve gestire la rete sociale tra gli utenti iscritti e deve reperire le informazioni circa la rete sociale e i virus da una fonte specifica che verrà illustrata nelle prossime sezioni. Il server, inoltre, mette a disposizione delle risorse richiedibili utilizzando il protocollo HTTP che permettono la gestione e la diffusione del contagio e la pubblicazione dello stato di salute degli utenti, secondo i vincoli specificati in seguito. Per agevolare la pubblicazione dei post su Facebook è stato creato un utente fittizio di nome Giovanni Algoritmo sul cui profilo è possibili postare gli stati di salute e i commenti.
Lo studente deve sviluppare solamente la parte server dell'architettura descritta. Per quanto riguarda il client si possono utilizzare vari programmi per la fase di test: telnet (Linux e Mac), putty () o un qualsiasi browser facendo richieste a localhost:<portaServerInAscolto>. Il server DEVE essere MULTITHREAD.

2 API

Nelle seguenti sottosezioni vengono descritti le componenti del server e le risorse che esso mette a disposizione.

2.1 Formato della richiesta

Il server riconosce come valide le richieste che seguono il protocollo HTTP. In particolare il server riconosce solo richieste con metodo di accesso di tipo GET. Per questo motivo lo studente NON deve implementare un server HTTP completo ma un suo piccolo sottoinsieme. Un esempio di possibile richiesta valida può essere:

```
GET <URL Risorsa con eventuali parametri> HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Il server deve verificare la validità della riga di intestazione e ignorare tutti i campi settati nella definizione dell'header. Una volta rilevata la stringa vuota di terminazione della richiesta, il server deve eseguire l'operazione richiesta dall'URL e rispondere secondo il formato HTTP. Si noti che l'URL della riga di intestazione della richiesta deve seguire la sintassi specifica di un URL (vedi slide corso). Se la richiesta è andata a buon fine, il formato della risposta è:

```
HTTP/1.0 200 OK

<messaggio>
```

dove < $messaggio$ > è un messaggio testuale che dipende dalla risorsa invocata. Nel caso di un qualsiasi errore il formato della risposta sarà:

```
HTTP/1.0 400 ERRORE

<messaggio>
```

Anche in questo caso < $messaggio$ > contiene un messaggio di errore dipendente dalla risorsa invocata.

2.2 Creazione della social network

Per quanto riguarda la rete sociale che impone i vincoli sulla diffusione dei virus, essa viene caricata al momento della creazione del servizio. Gli utenti e i relativi nodi non possono essere modificati, mentre è possibile creare degli archi durante l'esecuzione del servizio. Di conseguenza, non si possono aggiungere o eliminare utenti ma è possibile creare nuove relazioni di amicizia tra gli utenti presenti, oltre a quelle presenti al momento dell'inizializzazione del server. Il caricamento avviene per mezzo di un richiesta al web-server 159.149.133.70 in ascolto sulla porta 80. La risorsa da richiedere è $getAmici$. Questa risorsa, che non accetta parametri, restituisce una serie di righe (testo) che seguono il formato

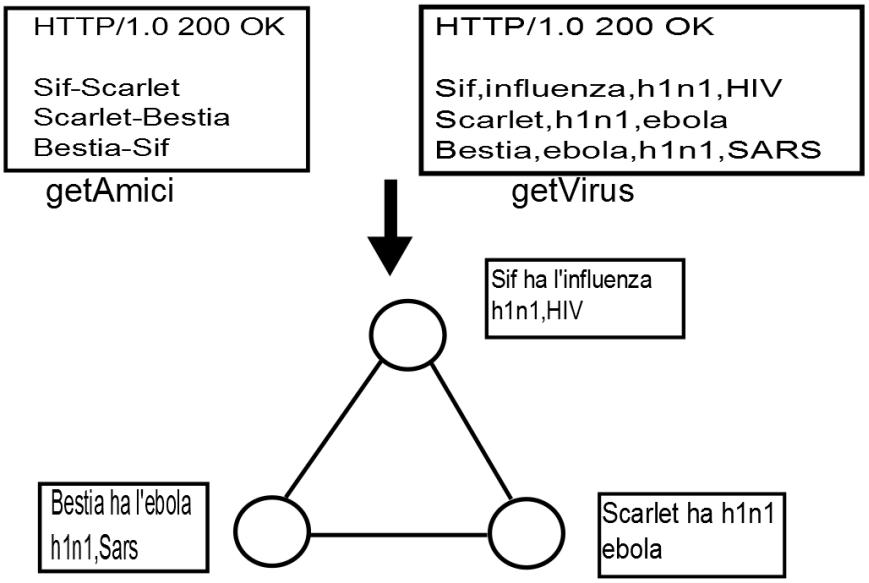

Figura 2.1: Esempio costruzione della rete sociale.

```
<nomeUtente>-<nomeUtente>
```

dove < $nomeUtente$ > rappresenta un segnaposto (< e > non vanno inseriti). La risposta segue il protocollo HTTP. Di seguito viene proposto un esempio completo di risposta nel caso di richiesta corretta.

```
HTTP/1.0 200 OK

Sif-Scarlet
Scarlet-Bestia
Bestia-Sif
```

Una seconda risorsa (*getVirus*) risponde con uno stream testuale che contiene, per ogni utente, il virus con cui è stato inizialmente infettato e una lista di virus a cui è immune. Ogni riga dello stream testuale descrive un utente e segue il formato:

```
<nomeUtente>,<virusIniziale>,<immunità1>,<immunità2>,<immunitàN>
```

Al momento del caricamento del grafo, lo studente deve prevedere anche l'assegnamento delle malattie e delle immunità. Nella figura 2.1 viene presentato un piccolo grafo costruito partendo dalle risposte restituite dalle due risorse.
Tutte le seguenti risorse, parametri e valori sono case-sensitive.

2.3 AGGIUNGERE UNA RELAZIONE DI AMICIZIA

Un utente può aggiungere un amico alla sua cerchia utilizzando la risorsa $aggiungiAmico$. La risorsa *aggiungiAmico* accetta come unici parametri validi $nomeUtente$ e $nomeAmico$, il primo indica l'utente che vuole creare l'amicizia, mentre il secondo indica il nome (stringa) dell'utente verso cui creare una relazione di amicizia. L'arco che viene creato è non orientato e non necessita che l'utente destinatario della richiesta accetti la creazione. Un possibile esempio di richiesta valida è il seguente:

```
GET /aggiungiAmico?nomeAmico=CapitanAmerica&nomeUtente=Sif HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

Amicizia con CapitanAmerica creata con successo
```

Possibili errori da gestire sono l'assenza del nodo di destinazione/sorgente della richiesta o la richiesta di creazione di una relazione che già esiste.

2.4 LISTA DEGLI AMICI

Un utente può visualizzare i propri amici utilizzando la risorsa $vediAmici$. La risorsa necessita dell'argomento $nomeUtente$: l'utente che richiede la lista degli amici. Nel corpo della risposta, $vediAmici$ restituisce un array JSON contenente una serie di oggetti JSON che rappresentano gli amici. Ogni oggetto JSON è costituito da due campi: $idUtente$ contenente il nome utente dell'amico e $malattie$ contenente un array JSON di stringhe, ognuna indicante un virus che attualmente sta infettando l'amico. Un possibile esempio di richiesta valida è la seguente:

```
GET /vediAmici?nomeUtente=Sif HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

[
{"idUtente": "Bestia", "malattie": ["ebola"]},
{"idUtente": "Scarlet", "malattie": ["h1n1" , "SARS"]}},
{"idUtente": "CapitanAmerica", "malattie": ["h1n1" , "ebola"]}}
]
```

Nell'esempio Scarler, amica di Sif, è malata di h1n1 e SARS al momento della richiesta. Possibili errori sono costituiti dall'errata sintassi della risorsa.

2.5 Diffusione di una malattia

Un utente può infettare un determinato amico per mezzo della risorsa $infettaAmico$. La risorsa prevede tre parametri: $nomeUtente$, $nomeAmico$ e $nomeVirus$. Il secondo parametro rappresenta il nome dell'amico a cui $nomeUtente$ vuole inviare il virus indicato dal terzo parametro. Si noti che la risorsa infetta l'amico SOLO se l'amico non è immune al virus, se l'amico non è già stato infettato dallo stesso virus e se $nomeUtente$ ha già contratto $nomeVirus$. Un possibile esempio di richiesta è la seguente:

```
GET /infettaAmico?nomeUtente=Bestia&nomeAmico=Sif&inomeVirus=ebola HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

Sif contagiata con ebola
```

La risorsa, inoltre, pubblica un messaggio e il relativo commento sul profilo di un utente definito dall'id **100007294060146**. Il messaggio ha la forma '< $nomeUtente$ > ha infettato < $nomeAmico$ > con < $nomeVirus$ >' mentre il commento al post pubblicato deve seguire il formato '< $nomeAmico$ >: ho < $listaMalattie$ >. Aiuto!!', dove < $listaMalattie$ > è data dalla concatenazione dei virus che infettano $nomeAmico$ separati da virgola. Per esempio il risultato sul profilo dell'utente alla richiesta precedente è mostrato nella seguente figura:
L'access token da utilizzare per la pubblicazione è il seguente:
CAADaUwpF1ZAEBAF82vZAVpGRmX6eydVUUAp0RGwndROZCmIZBvZC1sPFtviwnwL0d9Fbf3DQfetluTQ3WVZBXqEqxizUZC8BQX6NvpO3uArlGXgzaZBKJVr1GVzelqceKoxOFm3j6yPwU3iRW1n1qi8ZCKQXJTbCmSno3XqYQ3U6NnaEARvBsDYH7KgI84R8BNcsZD Possibili errori possono essere dovuti alla mancata presenza del nome utente, alla mancata relazione di amicizia con l'utente specificato oppure ai vincoli imposti dalla diffusione del virus specificato.

2.6 Richiesta delle malattie in corso

Un utente può richiedere l'elenco delle malattie in corso usando la risorsa $richiediMalattie$. La risorsa richiede il solo parametro $nomeUtente$ e restituisce come risposta un array JSON. Ogni elemento dell'array è un oggetto JSON con due campi: $nomeVirus$ e $infettante$. Il primo campo richiede una stringa rappresentante il virus, mentre il secondo campo indica il primo amico della vittima che ha inviato il virus per contagiarlo. Un possibile esempio di richiesta è la seguente:

Figura 2.2: Nella figura le malattie non sono separate dalla virgola, quindi l'immagine non è da considerarsi conforme alla specifica.

```
GET /richiediMalattie?nomeUtente=Sif HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

[
{"nomeVirus": "ebola", "infettante": "Bestia"},
{"nomeVirus": "SARS", "infettante": "Scarlet"}
]
```

3 Modalità di consegna

Il progetto è individuale e verranno presi provvedimenti nel caso ci sia il sospetto di una eventuale copiatura. In particolare non verranno ammessi alla discussione del progetto e quindi saranno automaticamente esclusi dall'appello quei progetti con almeno un file con indice di similarità superiore al 70% (moss, tool di Stanford). Questo per non incentivare progetti scritti singolarmente ma frutto di brain-storming troppo di gruppo.
Il linguaggio da utilizzare per lo sviluppo è Java. Il codice del progetto e i relativi jar delle librerie utilizzate devono essere inviati al docente in una directory compressa (zip o tar.gz) denominata 'progettoFebbraio2014-2'. La mail deve avere come oggetto 'ProgettoFebbraio2014-2' e contenere nel corpo nome, cognome e numero di matricola dello studente. Il materia-

le deve essere consegnato entro le **23:59:59** del **3 febbraio 2014**. Valgono tassativamente le seguenti regole:

- Il codice che non compila non verrà considerato
- I progetti consegnati dopo la data di consegna non saranno ritenuti validi

Per essere considerato sufficiente e quindi ammissibile per la discussione, il progetto deve implementare correttamente tutte le funzioni precedentemente illustrate. Principali criteri di valutazione del codice:

- Gestione di possibili errori nell'input da parte dell'utente
- Modularità del codice e rispetto del paradigma ad oggetti
- Gestione oculata delle eccezioni
- Uso appropriato dei commenti
- Gestione e risoluzione di eventuali problematiche legate alla concorrenza sia lato client sia lato server

Per chiarimenti, dubbi potete contattare il docente all'indirizzo matteo.zignani@unimi.it, specificando come oggetto obbligatorio 'ProgettoReti'. Eventuali modifiche al testo del progetto saranno comunicate sulla pagina web del corso.