UNIVERSITÀ DEGLI STUDI DI BOLOGNA
Dipartimento di Informatica - Scienza e Ingegneria

Corso di Laurea Magistrale in Informatica

Relazione di progetto: simulazione di un ambiente virtuale distribuito

Dott. Fabio Biselli

Anno Accademico 2014/2015

Indice

0.1 Sintesi del sistema originale

Il sistema introdotto nell'articolo ed illustrato in figura 0.1 è così composto:

- 3 server, di cui uno contrassegnato come principale;
- 180 client che controllano un avatar nel mondo virtuale;
- Una rete che connette i client ai server (che sono tra loro interconnessi);
- Un'area di gioco (Virtual Environment);
- Un metodo ed un file di partizionamento che il server principale utilizza per suddividere il carico tra i server.

Il sistema di partizionamento per l'assegnamento dei client ad un server è basato sul concetto di Area d'Interesse di un avatar (AoI). Ovvero se due avatar condividono la medesima AoI dovrebbero essere gestiti dal medesimo server.

All'inizio della simulazione gli avatar (client) sono distribuiti nel'area di gioco in modo uniforme.

La simulazione consiste nel far compiere ad ogni avatar 100 movimenti, uno ogni 2 secondi. Quando l'avatar compie un movimento invia un messaggio di ACK al server associato che lo propaga ai client nella relativa AoI. I client che ricevono l'ACK rispediscono il messaggio al server che

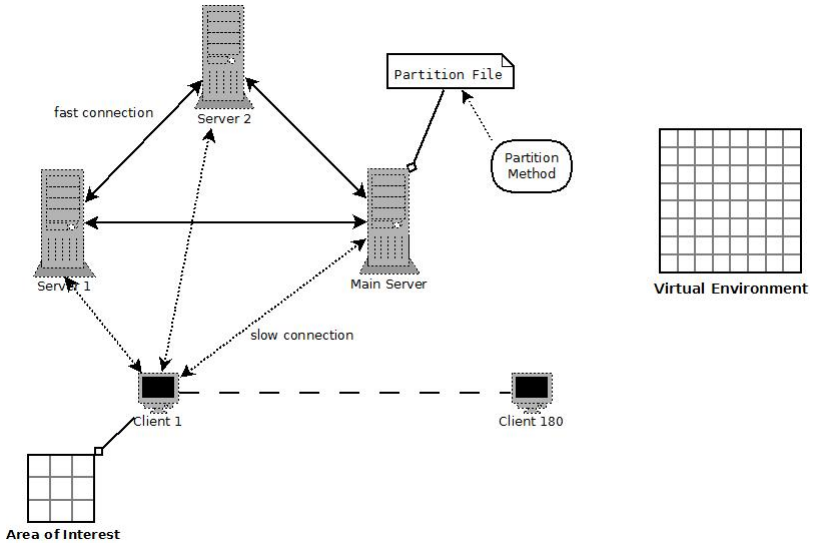

Figura 1: Schema proposto nell'articolo di riferimento.

notifica l'ACK ricevuto al client che ha effettuato lo spostamento. In questo modo è possibile calcolare i tempi di risposta del sistema. Alla fine della simulazione ogni client può calcolare il tempo medio di risposta del sistema.

0.2 Implementazione di un nuovo modello

In questa sezione sono descritte le principali caratteristiche per l'implementazione del nuovo modello proposto. Possiamo schematizzare il sistema descritto in figura 2 nel seguente modo:

- 1 Main Server che si occupa del login dei client e della gestione dell'ambiente simulato;
- $k \in \{1, \ldots, 9\}$ server di partizione che si occupano di gestire i messaggi tra client ed aggiornare il Main Server;
- n client che controllano un distinto avatar nel mondo virtuale;
- una rete WAN simulata che connette i client ai server;
- una rete LAN simulata che connette i server con una struttura ad anello unidirezionale;
- un'area di gioco (`VirtualEnvironment`);
- un metodo di partizionamento statico, dipendente dal numero di server di partizione, che il Main Server utilizza per suddividere il carico.

0.2.1 Impostazioni iniziali

Nell'articolo vengono descritte due diverse simulazioni con numeri di server e client fissi. L'implementazione di questa simulazione, basata su OMNet++, prevede un parametro per i server ed uno per i client in modo tale che l'utente possa specificarne il numero (file `.ini`).

Si suppone che inizialmente gli avatar siano distribuiti nell'area di gioco con una data distribuzione casuale e che non siano già presenti nella suddetta ma debbano effettuare il login sul server principale dopo un tempo casuale dall'inizio della simulazione.

0.2.2 Area di gioco

L'area di gioco è una semplicissima struttura: un array bidimensionale di mappe di Avatar (⟨ `id, VirtualAvatar`⟩). Questa simula un'area di gioco in cui gli avatar possono muoversi liberamente e senza collisioni. Per l'implementazione si utilizzano due classi distinte per gli avatar: `Avatar` utilizzata dai client e `VirtualAvatar` utilizzata dal Main Server.

0.2.3 Avatar

`Avatar` è una semplice classe che rappresenta un client. Ha quattro campi: un ID che si riferisce al client (`getIndex()` in `simpleModule` di OMNet), due interi che rappresentano le coordinate all'interno dell'ambiente virtuale e un vettore di indici che rappresentano gli altri avatar nella propria Area di Interesse.

0.2.4 VirtualAvatar

`VirtualAvatar` è la classe che rappresenta l'avatar all'interno dell'ambiente virtuale gestito dal Main Server. Ha quattro campi: oltre agli interi che rappresentano id e coordinate utilizza un puntatore all'ambiente virtuale.

0.2.5 Gestione area di gioco e movimenti

L'area di gioco viene modificata dal Main Server tramite messaggi da parte dei server di partizione che, grazie alle notifiche dei movimenti da parte dei client, rimuovono l'avatar dalla vecchia cella e lo assegnano a quella di destinazione.

A questo punto vengono aggiornati i client coinvolti, ovvero nel momento in cui un avatar si sposta, il server di partizione:

1. notifica ai vicini che l'avatar lascia la casella;
2. calcola una nuova AoI con i nuovi vicini;

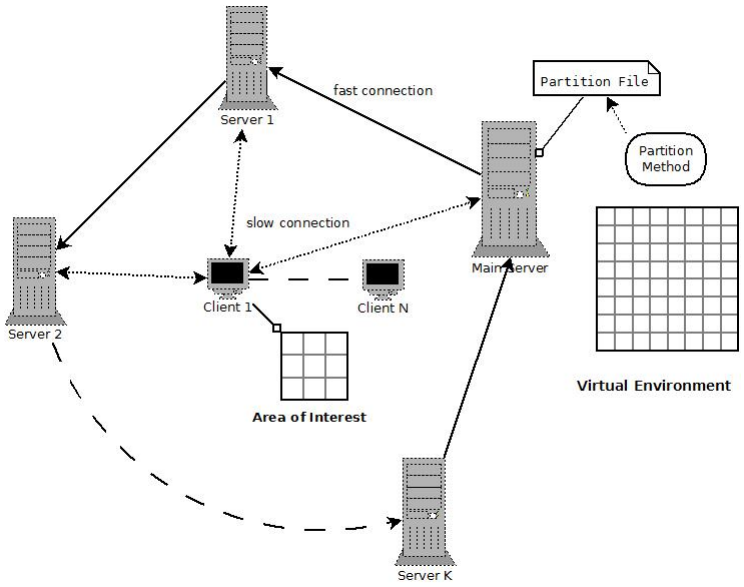

Figura 2: Schema proposto per l'implementazione.

3. notifica ai nuovi vicini l'ingresso dell'avatar nella casella di destinazione ed al Main Server lo spostamento.

I movimenti degli avatar, che avverranno ogni due secondi, avranno come destinazione una delle caselle adiacenti (compresa la casella di partenza, in tal caso nessun messaggio sarà inoltrato nel sistema). Tuttavia, con una piccola probabilità, ogni avatar potrà effettuare un Jump ad una casella casuale nel mondo, come evento eccezionale.

0.2.6 Connessioni

I server sono interconnessi, mediante dei channel con un basso delay per simulare una connessione intranet (LAN) fra loro. Mentre per le connessioni Client-Server i canali hanno una latenza più alta e variabile per simulare una WAN.

0.2.7 Metodo di partizionamento

Il metodo di partizionamento, poiché non è oggetto dello studio, sarà implementato in modo semplificato. Ad ogni Server verrà assegnata una porzione del mondo in modo lineare. Questo introdurrà un numero massimo di server che l'utente potrà specificare all'avvio.

0.2.8 Simulazione di eventi

Il sistema simula l'interazione da parte di utenti (che possono essere giocatori di un gioco distribuito o utenti di un simulatore per l'addestramento militare o civile) con un ambiente virtuale, ogni utente può interagire con il mondo virtuale mediante un client (`DVEClient`). Il client è connesso tramite la rete WAN (internet) al sottosistema server che gestisce il gioco o la sessione di addestramento, ogni qual volta l'utente effettua un movimento che induca un cambio di stato del mondo virtuale questo invia un messaggio di movimento (`MoveMsg`) al server.

Al fine di simulare questi eventi si è scelto di utilizzare il modulo `Source` del pacchetto `queueinglib`. Ogni client ha un'istanza della classe `Source` associata la quale genera un numero stabilito di `Job` a partire da un momento casuale dall'inizio della simulazione, uno alla volta con un intervallo regolare. Entrambi i parametri sono specificati nel file `omnet.ini` e sono quindi configurabili. Il primo evento per ogni client è sempre il login, ovvero viene generato un messaggio di login(`LoginMSG`) da inviare al server principale, i jobs successivi invece sono interpretati come movimenti.