Università di Urbino

Informatica Applicata

Reti di calcolatori

Relazione

Progetto per la sessione estiva 2015/2016

Studente:
Marco Tamagno
matricola no: 261985

Studente:
Julian Sparber
matricola no: 260324

Professore:
Antonio Della Selva

June 20, 2015

Contents

1 Specifica del Problema

Creare sfruttando il protocollo WebRTC un gioco che prenda spunto al gioco del pong, ma nel quale 4 giocatori possano sfidarsi tra loro. Gli utenti connessi tramite browser si dovranno scambiare informazioni tramite il protocollo.

2 Analisi del Problema

2.1 WebRTC

WebRTC (Web Real-Time Communication) è un API standard creato dal World Wide Web Consortium (W3C) che supporta applicazioni browser-to-browser per chiamate, videochiamate, e P2P file sharing senza il bisogno di programmi esterni.

2.2 Interactive Connectivity Establishment (ICE)

L'ICE viene sfruttato nelle connessioni browser to browser per stabilire una connessione tra due peer.

STUN è l'acronimo di Session Traversal Utilities for NAT (Network Address Translation): si tratta di un protocollo e di un insieme di funzioni che permettono alle applicazioni in esecuzione su un computer di scoprire la presenza ed i tipi di NAT e firewall che si interpongono tra il computer e la rete pubblica.

STUN permette a queste applicazioni di conoscere gli indirizzi IP e le porte con cui il dispositivo NAT li sta rendendo visibili sulla rete pubblica. STUN opera con molti NAT preesistenti e non richiede particolari comportamenti da essi.

Come risultato, STUN assicura ad una grande varietà di applicazioni IP (ad esempio, i telefoni VoIP) di lavorare attraverso le varie strutture NAT preesistenti.

Nella specifica originaria in RFC 3489, STUN era l'acronimo di Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), ma nella specifica aggiornata pubblicata come RFC 5389 il titolo è mutato in Session Traversal Utilities for NAT, mantenendo lo stesso acronimo.

STUN è un protocollo client-server. Un telefono o un software VoIP può includere un client STUN, che invierà una richiesta ad un server STUN. Il server riporterà al client STUN l'indirizzo IP pubblico e la porta UDP che il dispositivo NAT (es. router) sta associando al client per il traffico entrante nella rete.
Le risposte permettono anche al client STUN di determinare che tipo di NAT è in uso.
Ci sono tre tipi di NAT che è possibile attraversare tramite STUN: Full Cone, Restricted Cone e Port Restricted Cone.

STUN non lavora con il quarto tipo di NAT, detto simmetrico o bidirezionale, questo a causa del fatto che i dati trovati dal server STUN non saranno validi per terze parti, in quanto il NAT bidirezionale non permette a terzi di riusare IP e porte abilitate, differenziando le associazioni a seconda dell'host contattato.
Se a causa del NAT non è possibile creare una connessione peer to peer allora la connessione si appoggerà a un TURN server.

2.3 Il gioco

2.3.1 Elementi

Il gioco sarà composto da questi elementi:
una pallina
una racchetta (una per ogni giocatore)
una stanza (dove verrà svolta la partita)
una rete (dove verranno segnati i gol)

2.3.2 Gameplay

Ogni giocatore invierà a ogni altro giocatore la posizione della propia racchetta.

Il giocatore master, in possesso della pallina comunicherà a tutti gli altri giocatori la posizione della pallina.

Il master invierà agli altri giocatori il numero di reti che hanno subito subito.

Il master controllerà l'avvenimento di collisioni della pallina contro qualsiasi oggetto del campo, e ne comunicherà l'avvenuta collisione agli altri giocatori.

Quando la pallina sarà soggetto di una collisione verrà riprodotto un suono, differente a seconda di che cosa è stato colpito.
Ogni stanza ospiterà una partita differente la quale sarà identificata tramite un id.

3 Scelte di Progetto

3.0.3 SimpleWebRTC

Abbiamo deciso di utilzzare la libreria SimpleWebRTC perchè crea un interfaccia semplice e unificata per ogni browser compatibile a webRTC.
La libreria permette di accedere al microfono e alla webcam dell'utente, ma per il nostro gioco non ne necessitiamo, per cui disattiveremo la richiesta a essi e il trasferimento del video e dell'audio.

```
var webrtc = new SimpleWebRTC({
  // we don't do video
  localVideoEl: '',
  remoteVideosEl: '',
  // don't ask for camera access
  autoRequestMedia: false,
  // don't negotiate media
  receiveMedia: {
    mandatory: {
      offerToReceiveAudio: false,
      offerToReceiveVideo: false
    }
  },
  // our own signalserver
  url: 'https://sparber.net:62249/'
});
```

La libreria SimpleWebRTC utilizza per creare le stanze e inizializzare la connessione tra i peer un signal server. Abbiamo scelto come signal server Signalmaster compatibile con la libreria SimpleWebRTC, il quale abbiamo installato e configurato sul server sparber.net

A causa dell impossibilità di installare uno STUN/TURN server sul server sparber.net, abbiamo deciso di utilizzarne uno reso disponibile da xirsys.com, che nell offerta gratuita ci mette a disposizione 10 connessioni possibili al TURN server e 100 MB di traffico mensili, caratteristiche che per una dimostrazione della funzionalità del gioco sono sufficenti.

Modificando il Signalmaster abbiamo fatto in modo che questo distribuisse le credenziali d'accesso al turn server ai vari peer, le quali sono prima state richieste al server di xirsys.com.

3.0.4 Stanze

Per poter creare piú stanze ogni stanza ha un url con una sua query, ad esempio:
https://unicooperative.github.io/network2015/?room5251

Questo sistema permette di poter giocare a piú partite alla volta. Per giocare nella stessa stanza basta mettere la stessa query nella url. Un url privo di query avrà assegnato un id casuale, ad esempio "room5251"

```
var room = "room" + parseInt(Math.random()*10000);
if (history.pushState) {
  var newurl = window.location.protocol +
    "//" +
    window.location.host +
    window.location.pathname +
    '?' +
    room;
    window.history.pushState({path:newurl},'',newurl);
  }
```

3.0.5 Gameplay

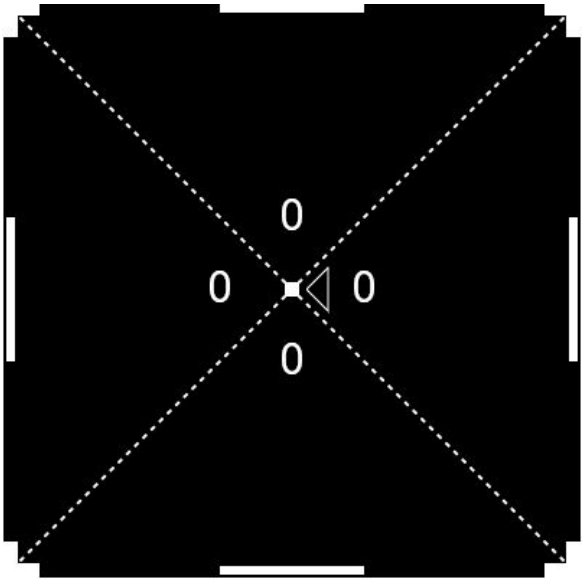
La partita puo' iniziare solo quando sono arrivati tutti e 4 i giocatori..

```
function isReadyToPlay() {
  var allPeers = webrtc.webrtc.peers;
  var readyPlayers = 0;
  if(allPeers.length === 3 && state === "notReady"){
    for(var i = 0; i < allPeers.length; i++) {
      if (allPeers[i].channels.message.readyState === "open") {
        readyPlayers++;
      }
    }
    if(readyPlayers === 3) {
      state = "ready";
      return true;
    }
  }
  return false;
}
```

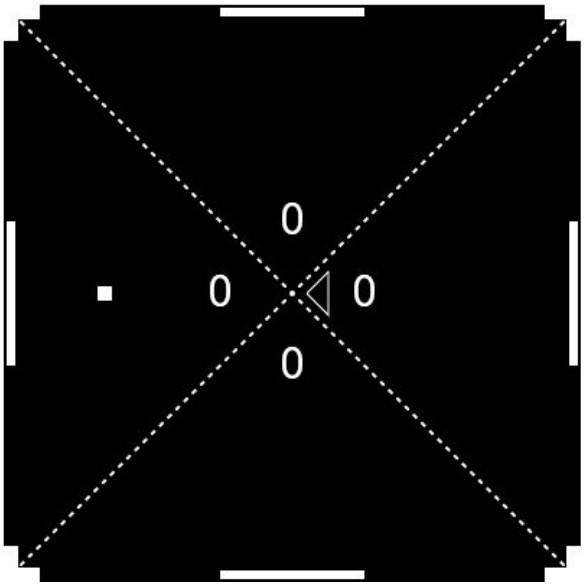
4 Demo dell'Applicazione

All inizio del gioco il giocatore aspetta l'arrivo di 3 giocatori.

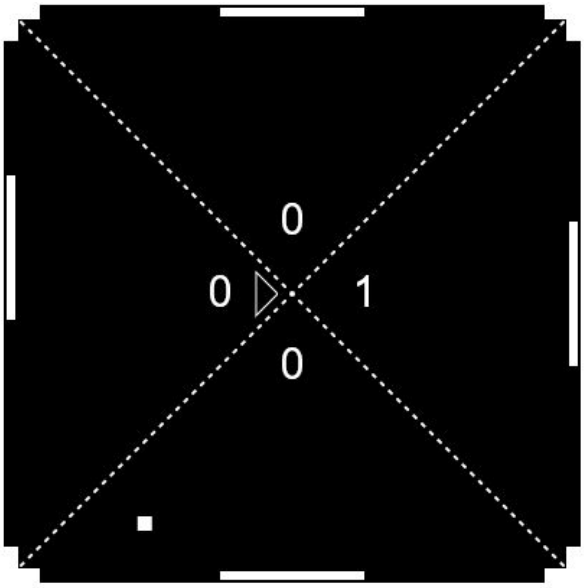
Una volta connessi i 4 giocatori il master invia agli altri giocatori la pallina.



E i suoi relativi spostamenti.



Ogni giocatore invia i gol subiti agli altri giocatori.



Se volete testare il gioco è reso disponibile all'url:
https://unicooperative.github.io/network2015/?room5251