Progetto di Distributed Software System: Quick, DrawRTC!

Mario Sessa
mariosessa@studio.unibo.it

A.A. 2021/22

Indice

1. Introduzione

- **Quick, Draw!** è un gioco sviluppato dalla Google

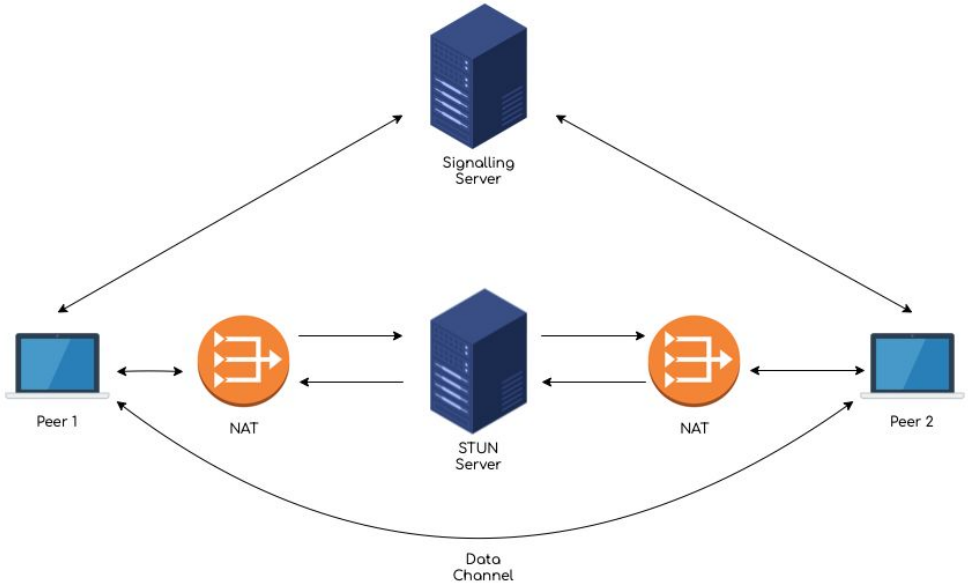
- Lo scopo del gioco è far riconoscere uno scarabocchio ad un agente intelligente e acquisire nuovi dati di training.

Versione distribuita:

- Definire un gioco decentralizzato in modo da rendere il gioco fruibile per più giocatori, tollerante agli errori e con stati consistenti.

- Ogni game ha un numero minimo di 3 giocatori e massimo 8.

2. Pattern Architetturale



L'architettura di base è di tipo **P2P** in full mesh.

- Si utilizza un **server STUN** per riconoscere il proprio IP pubblico attraverso il **NAT**.
- Si stabilisce una comunicazione **SDP+ICE** con il signalling server per creare un **Data Channel** tra i peer.

3. Framework Utilizzati

PeerJS è una libreria per la comunicazione P2P basata su **WebRTC**.

- La comunicazione con il server di signaling utilizza **Socket.io**
- Il server di signaling è implementato in **express.js** integrato con la libreria di **PeerJS Server.**

https://github.com/peers/peerjs-server

https://github.com/peers/peerjs

4. Casual Consistency

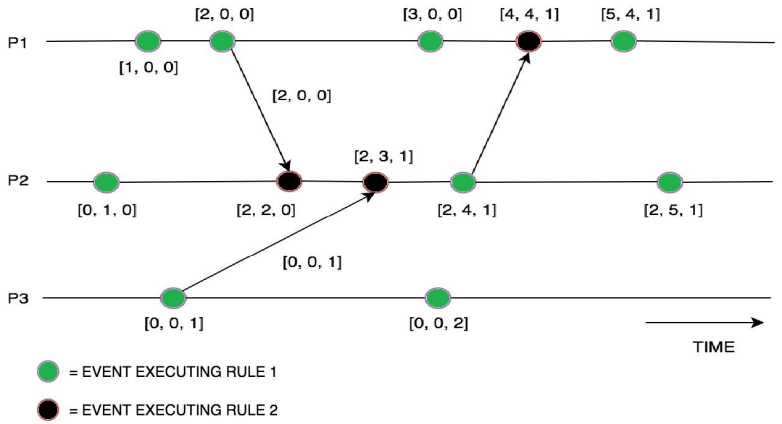
Causal Consistency: Rappresenta una forma di consistenza che preserva parzialmente l'ordine delle operazioni. Essa basa il rispetto dell'ordinamento degli effetti secondo un principio di causalità dettato da relazioni **happens-before.**

- **Ordinamento parziale:** Si rispetta l'ordine delle operazioni solamente in casi in cui le operazioni sono correlate.
- **Violazione di causalità:** Se A causa B ma B viene eseguito prima di A, allora il risultato atteso è differente rispetto al risultato ottenuto.

P1: W(x)a			
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)a	R(x)b

5. Vector Clock

- **Definizione:** Algoritmo scalare in grado di definire un ordinamento parziale andando a rispettare il principio di causalità secondo relazioni **happens-before.**
- **Descrizione ad alto livello**:
  - Inizialmente ogni elemento del vettore è impostato a 0.
  - Ad ogni evento interno si aggiorna il proprio clock logico.
  - Ad ogni evento esterno si aggiorna il clock logico del mandante prendendo il massimo tra il valore locale con quello ricevuto.



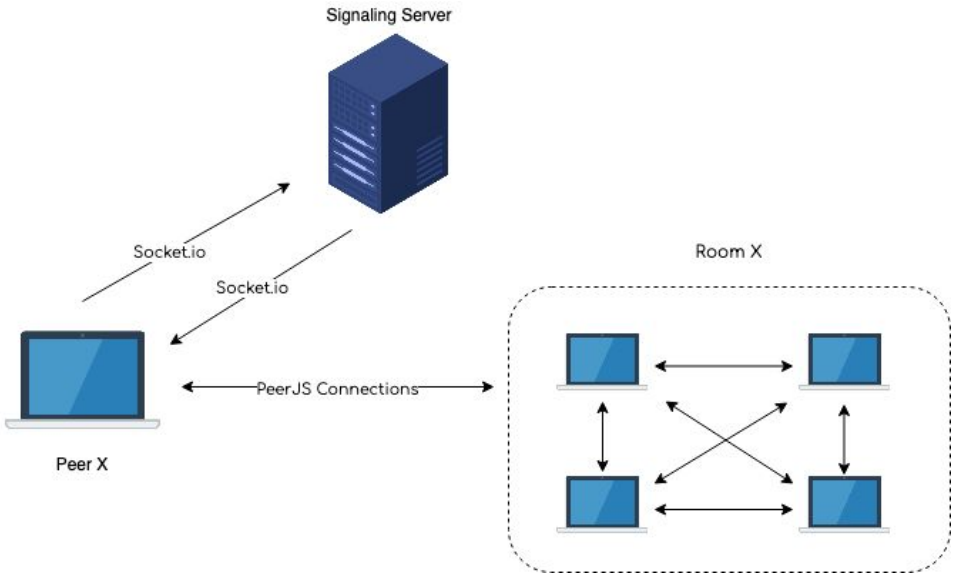
6. Estensione del signaling server

Esso è responsabile della:

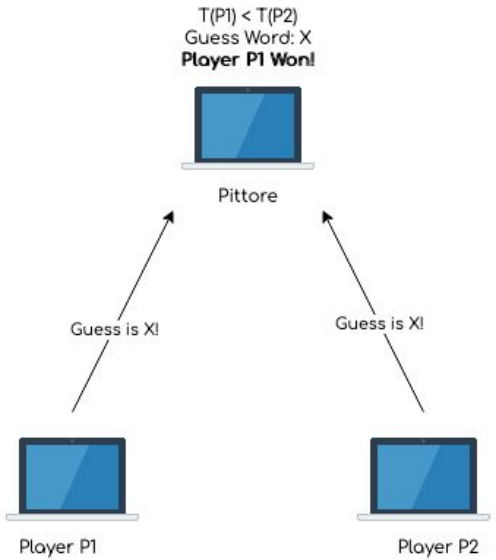
1. Gestione del sistema multi-lobby
2. Gestione delle connessioni tra i peer

Ogni lobby rappresenta una rete mesh indipendente in cui poter connettersi fino all'ingresso in gioco di uno dei player.

Il crash del signaling server impedisce solamente l'aggiunta di nuovi peer ad una rete.



7. Logica di gioco



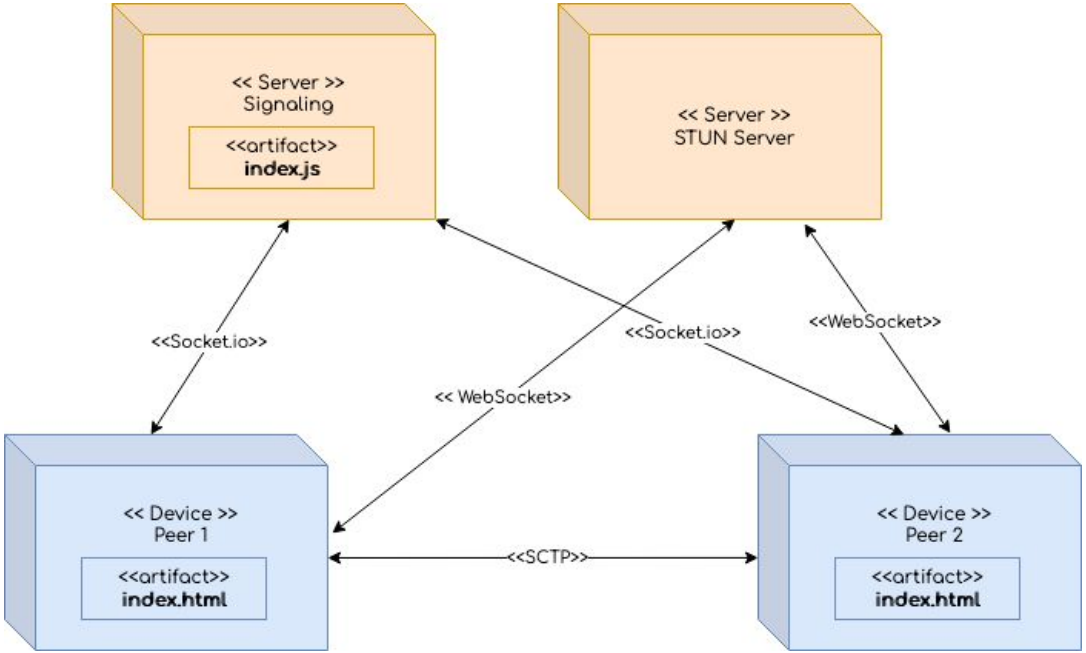
- Limitazione della condivisione della guess word per evitare cheating.
- L'ordinamento causale viene utilizzato per rimuovere criticità happens-before relativo a:
  - Aggiornamento di punteggi dei giocatori
  - Verifica dello stato di vincita o di perdita

8. Sistema di selezione su priorità

Per poter scegliere un pittore tra i peer connessi al gioco:

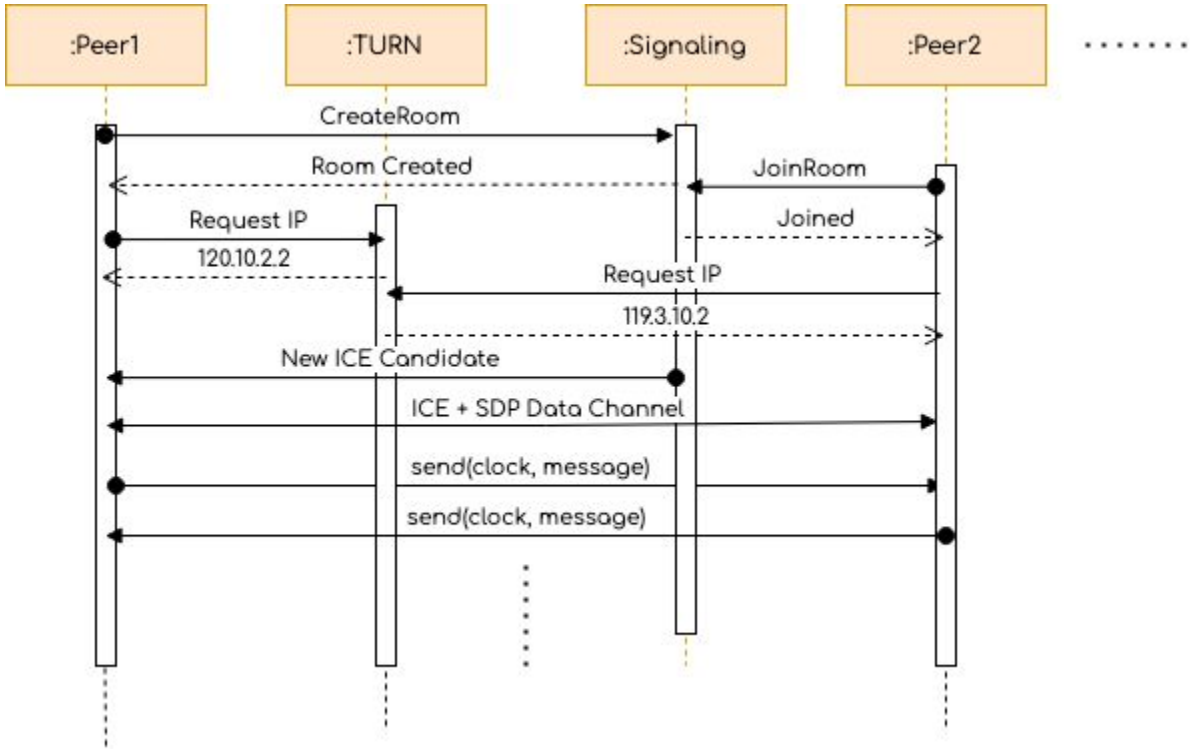
- All'inizializzazione, i peer otterranno una priorità randomica univoca rispetto ai valori presenti sugli altri nodi di rete.
- Prima dell'avvio del gioco, ogni utente seleziona un peer randomicamente come candidato al ruolo di pittore.
- Il peer con la priorità più bassa tra i candidati viene eletto come pittore, i restanti saranno i partecipanti al gioco.

8. Architettura: Deployment Diagram

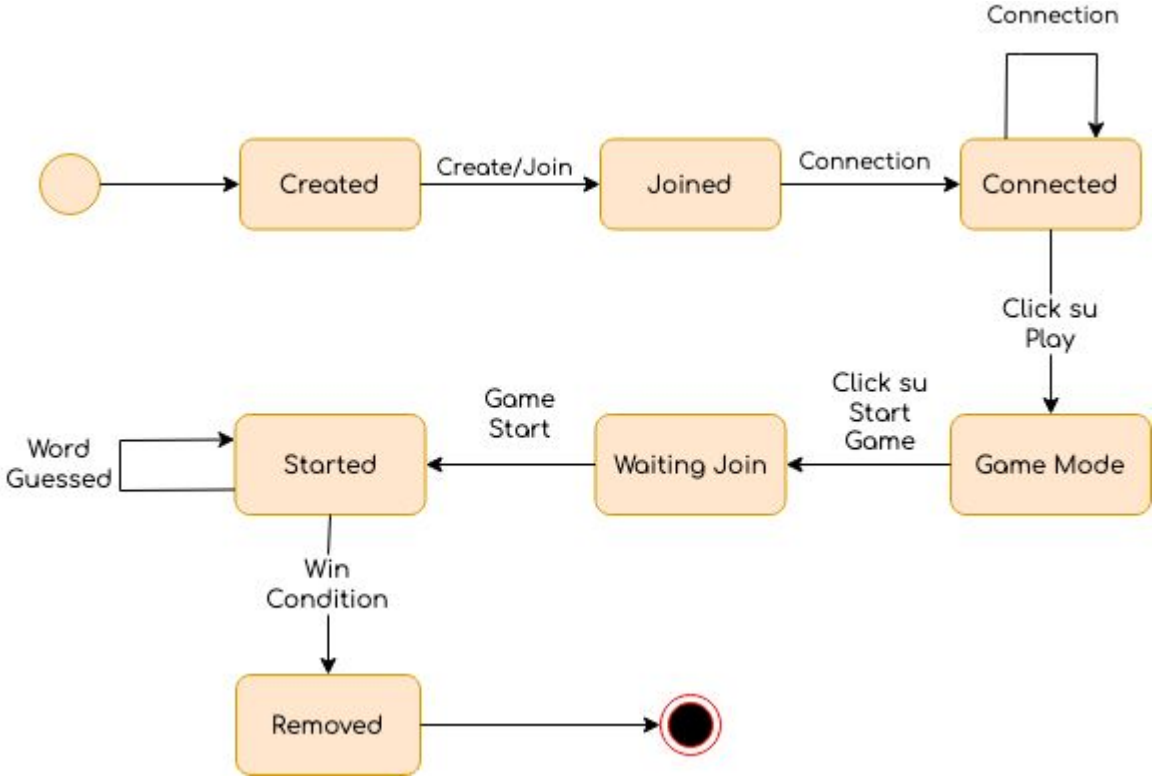


- Il server di signaling e il server STUN non sono utilizzati in seguito alle connessioni dei peer.
- Questo fa sì che una rete P2P precedentemente creata, possa elaborare la comunicazione in maniera indipendente da tali componenti.
- Un loro crash non prevede la terminazione del gioco ma impossibilità l'istanziazione di nuovi sessioni di gioco.

8. Architettura: Sequence Diagram



8. Architettura: Statechart Diagram



8. Stato di un Peer

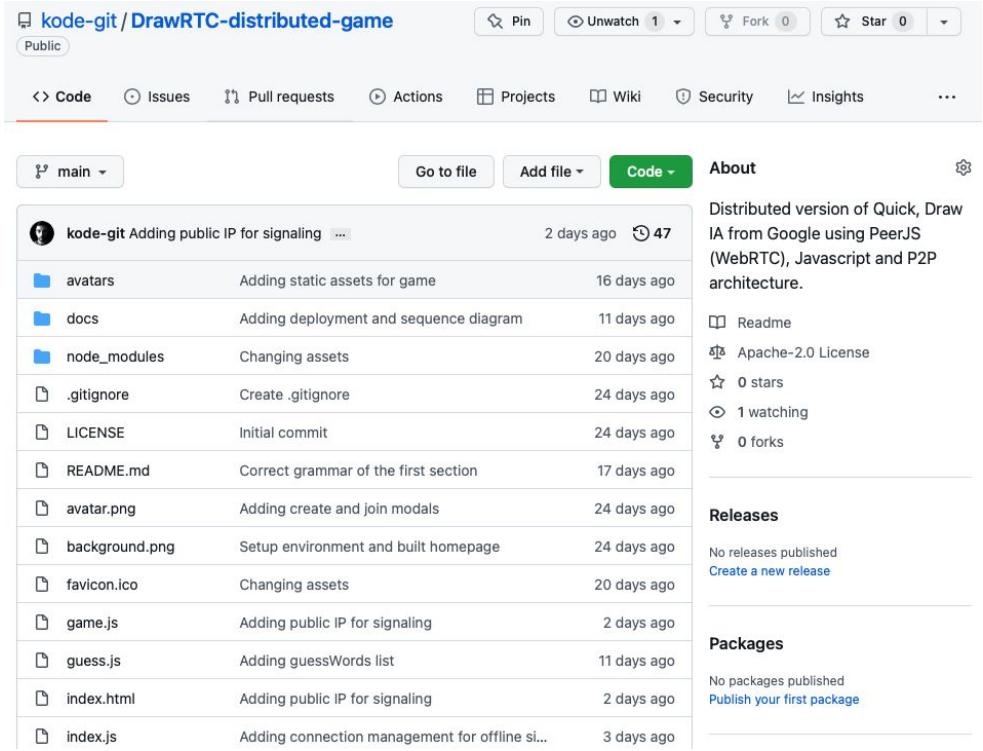
Lo stato di un peer è raffigurato da un insieme di dati logici mantenuti in sessione o in coda dei messaggi categorizzati come segue:

- **Informazione di connessioni:** Dati come l'id della stanza, la collezione degli id propri e dei peer connessi, l'insieme delle connessioni, stato di sessione, vettore dei clock.
- **Dati del sistema di votazione:** Dati utilizzati per l'elezione di un pittore tra i peer entrati in gioco; come l'insieme dei voti, il numero dei voti e lo stato di votazione.
- **Dati di gioco:** Lista dei punteggi, stato di gioco, guess word.
- **Dati di priorità:** Valore proprio e lista di priorità dei peer connessi per la gestione della votazione su priorità.

9. Implementazioni future

- Miglioramento delle performance di gioco
- Rendere l'interfaccia responsive per dispositivi mobili
- Supporto al touchscreen per il disegno
- Inserimento di giocatori automatici basati sul riconoscimento simbolico di immagini allenata su rete neurale.
- Miglioramento dell'usabilità dell'interfaccia di gioco

Github e link utili



https://github.com/kode-git/DrawRTC-distributed-game

Github: L'intero progetto è stato caricato in una repository pubblica su Github.

Riferimenti utilizzati:

- https://peerjs.com
- https://webrtc.org

Riferimenti

1. Matthieu Perrin, Achour Mostefaoui, and Claude Jard. 2016. Causal consistency: beyond memory. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '16)*. Association for Computing Machinery, New York, NY, USA, Article 26, 1–12.

2. Yen, Li-Hsing & Huang, Ting-Lu. (2003). Resetting Vector Clocks in Distributed Systems. Journal of Parallel and Distributed Computing. 43. 10.1006/jpdc.1997.1330.

3. Fox, Geoffrey. (2001). Peer-to-Peer Networks. Computing in Science & Engineering. 3. 75 - 77. 10.1109/5992.919270.

4. Rabab Hayek, Guillaume Raschia, Patrick Valduriez, Noureddine Mouaddib. Data Sharing in P2P Systems. *Handbook of Peer-to-Peer Networking*, Springer US, pp.531-570, 2010. ⟨hal-00379832⟩

Demo!