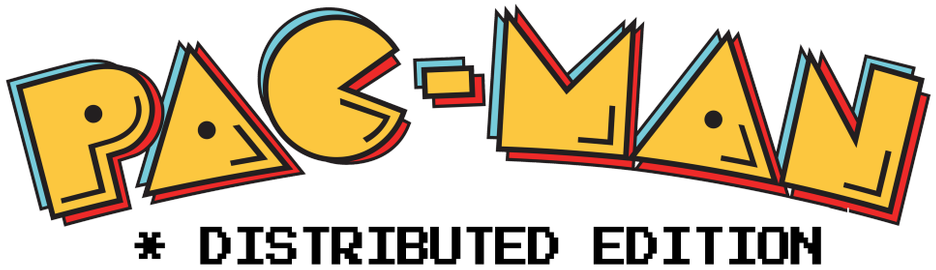
Università di Bologna

DISTRIBUTED SOFTWARE SYSTEM



Giacomo Vallorani

Febbraio 2020

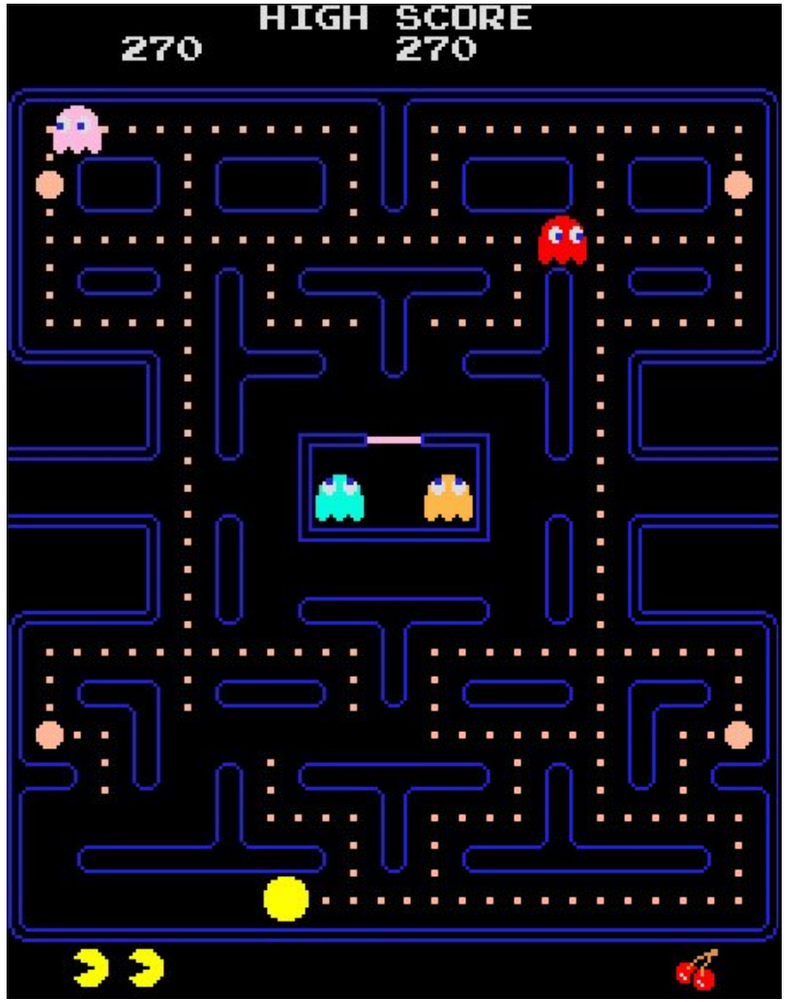
OUTLINE

1. Introduzione
2. Eventual consistency
3. CRDT
4. Software stack
5. Architettura
6. Demo
7. Valutazioni
8. Conclusioni

1. INTRODUZIONE

Originale

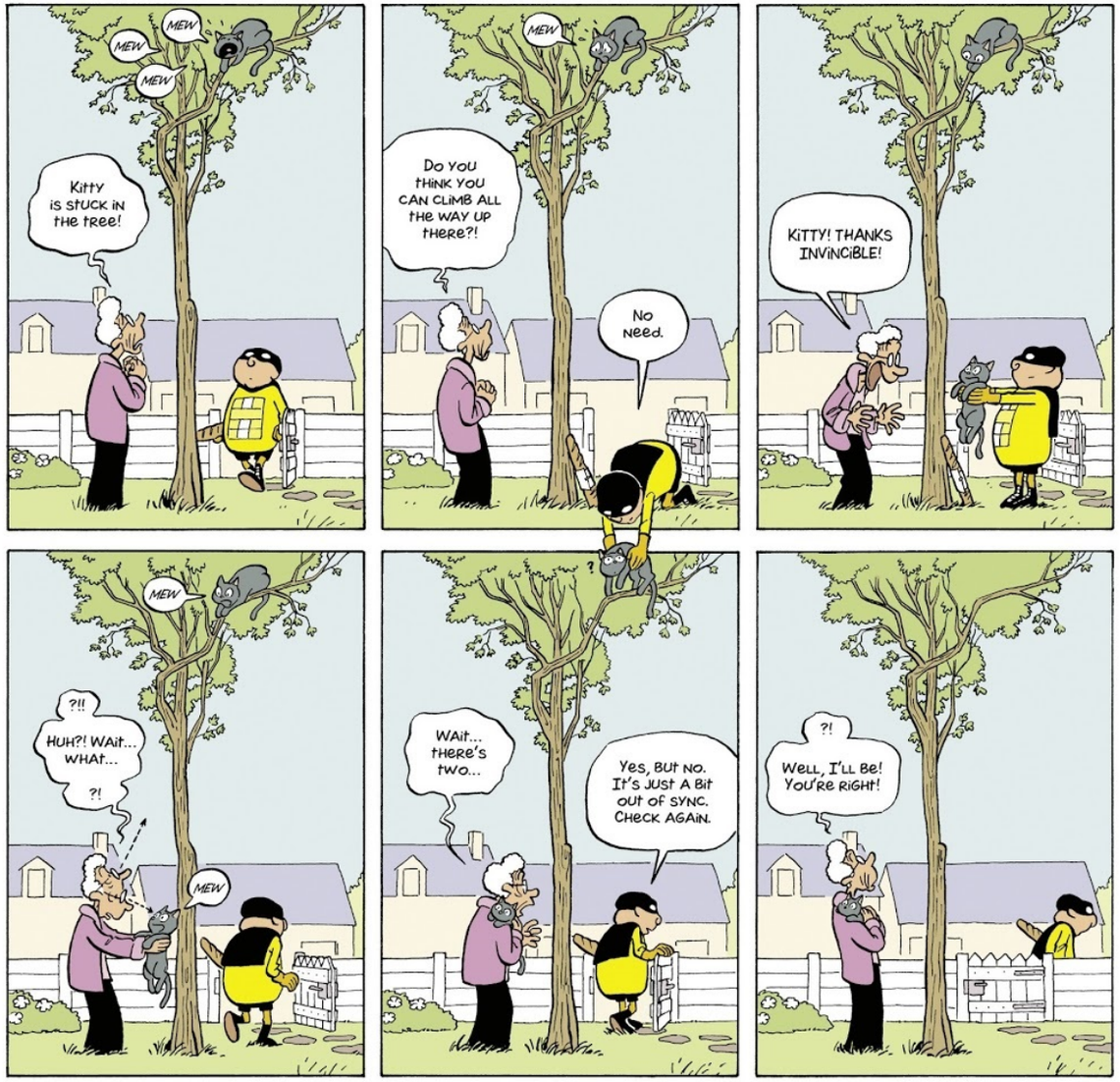
- Prodotto da Namco nel 1980
- **1 Pac-Man**
- 4 fantasmini
- 4 power-pellet
- Scopo del gioco: fagocitare i punti disseminati all'interno del labirinto evitando di scontrarsi con i fantasmini



Multiplayer distribuito

- **(1 - 8) Pac-Man**
- fault tollerant

2. EVENTUAL CONSISTENCY



https://www.europecomics.com/album/1-justice-fresh-vegetables/

2. EVENTUAL CONSISTENCY

> *__CAP__: è impossibile che un servizio distribuito sia coerente, disponibile e tollerante al partizionamento nello stesso istante di tempo [1].*

> *__Eventual Consistency (EC)__: se non vengono apportati nuovi aggiornamenti ad un certo dato, alla fine tutti gli accessi a quel dato restituiranno l'ultimo valore aggiornato [2].*

2. EVENTUAL CONSISTENCY

> ***CAP**: è impossibile che un servizio distribuito sia coerente, disponibile e tollerante al partizionamento nello stesso istante di tempo [1].*

> ***Eventual Consistency (EC)**: se non vengono apportati nuovi aggiornamenti ad un certo dato, alla fine tutti gli accessi a quel dato restituiranno l'ultimo valore aggiornato [2].*

> ***Strong Eventual Consistency (SEC):** tutti i nodi che hanno ricevuto lo stesso insieme di aggiornamenti (non ordinati) in un futuro raggiungeranno lo stesso stato [3].*

3. CRDT

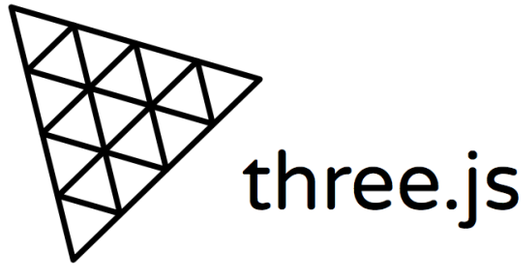
Conflict-free Replicated Data Type (CRT): tipo di dato astratto progettato per essere replicato su più nodi e che presenta le seguenti proprietà:

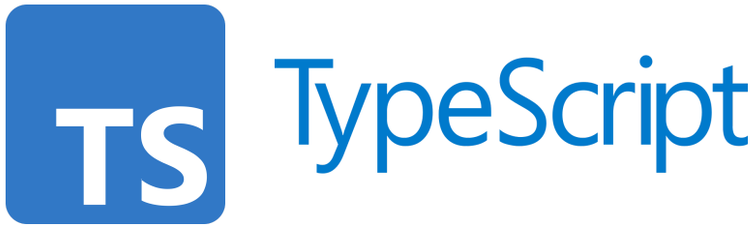
- qualsiasi replica può essere modificata senza coordinarsi con le altre;
- quando due repliche qualsiasi hanno ricevuto lo stesso insieme di aggiornamenti, raggiungono lo stesso stato, deterministicamente, adottando regole matematicamente valide per garantire la convergenza degli stati [4].

Applicazioni enterprise :

- **Riak**, primo database ad aggiungere il supporto ai CRDT nel 2013 [5];
- **Dynomite** *(Netflix)*, sistema di storage, utilizza CRDT internamente [6];
- **CosmosDB** *(Microsoft)*, può risolvere i conflitti utilizzando CRDT [7];
- **Redis Enterprise** utilizza i CRDT per abilitare la replicazione multi-master tra datacenter geograficamente distribuiti [8].

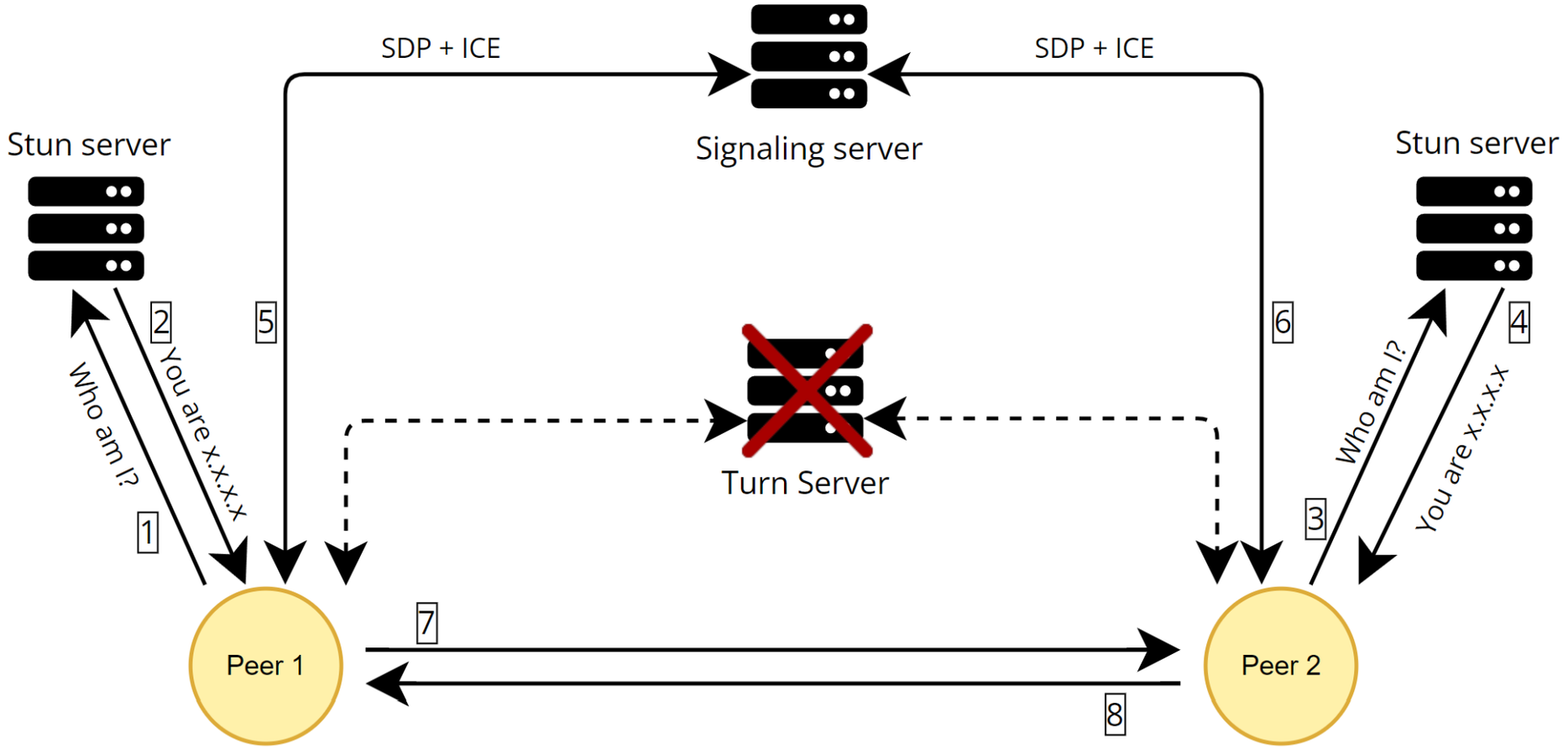
4. SOFTWARE STACK





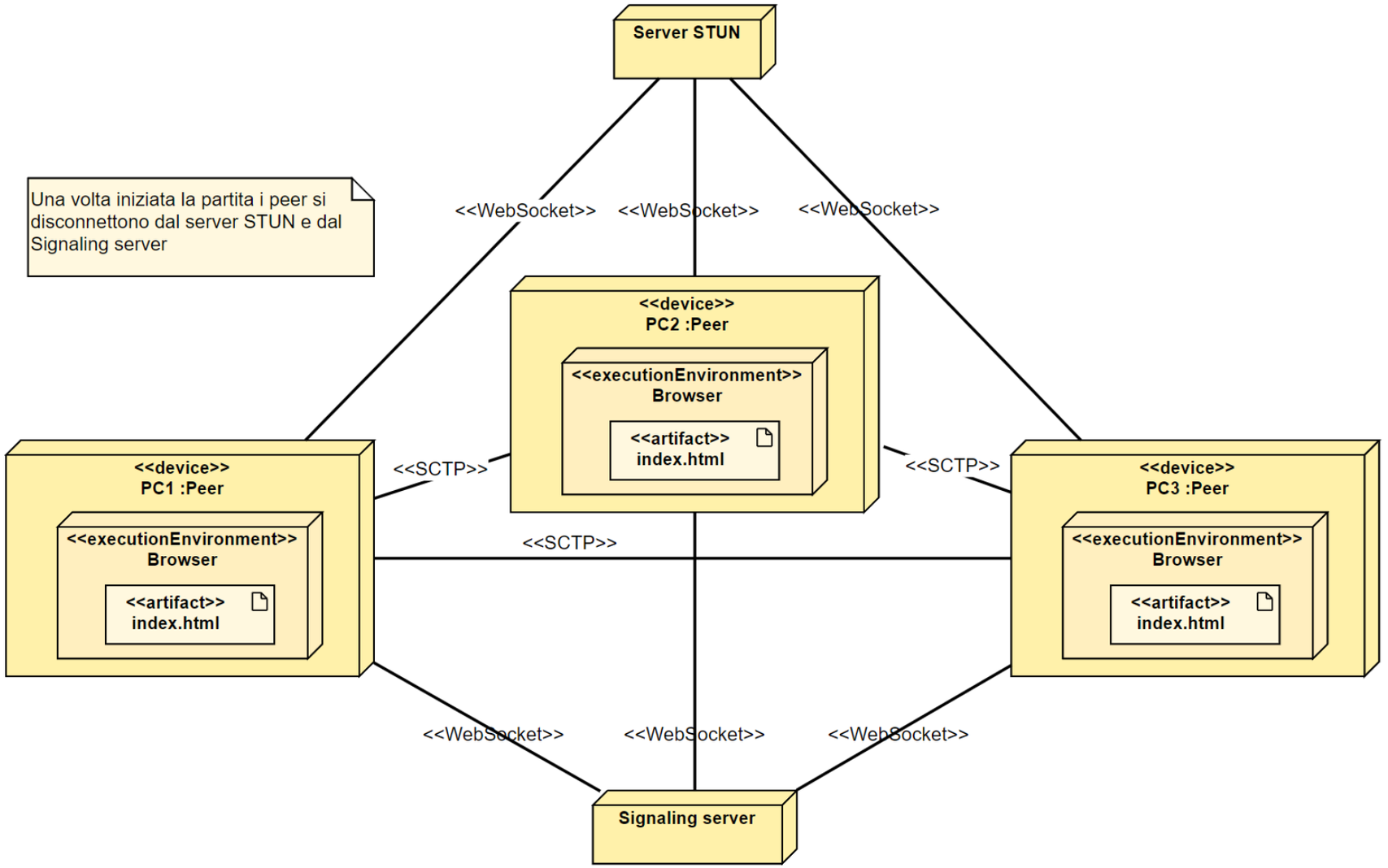
5. ARCHITETTURA

DIAGRAMMA DI COMUNICAZIONE



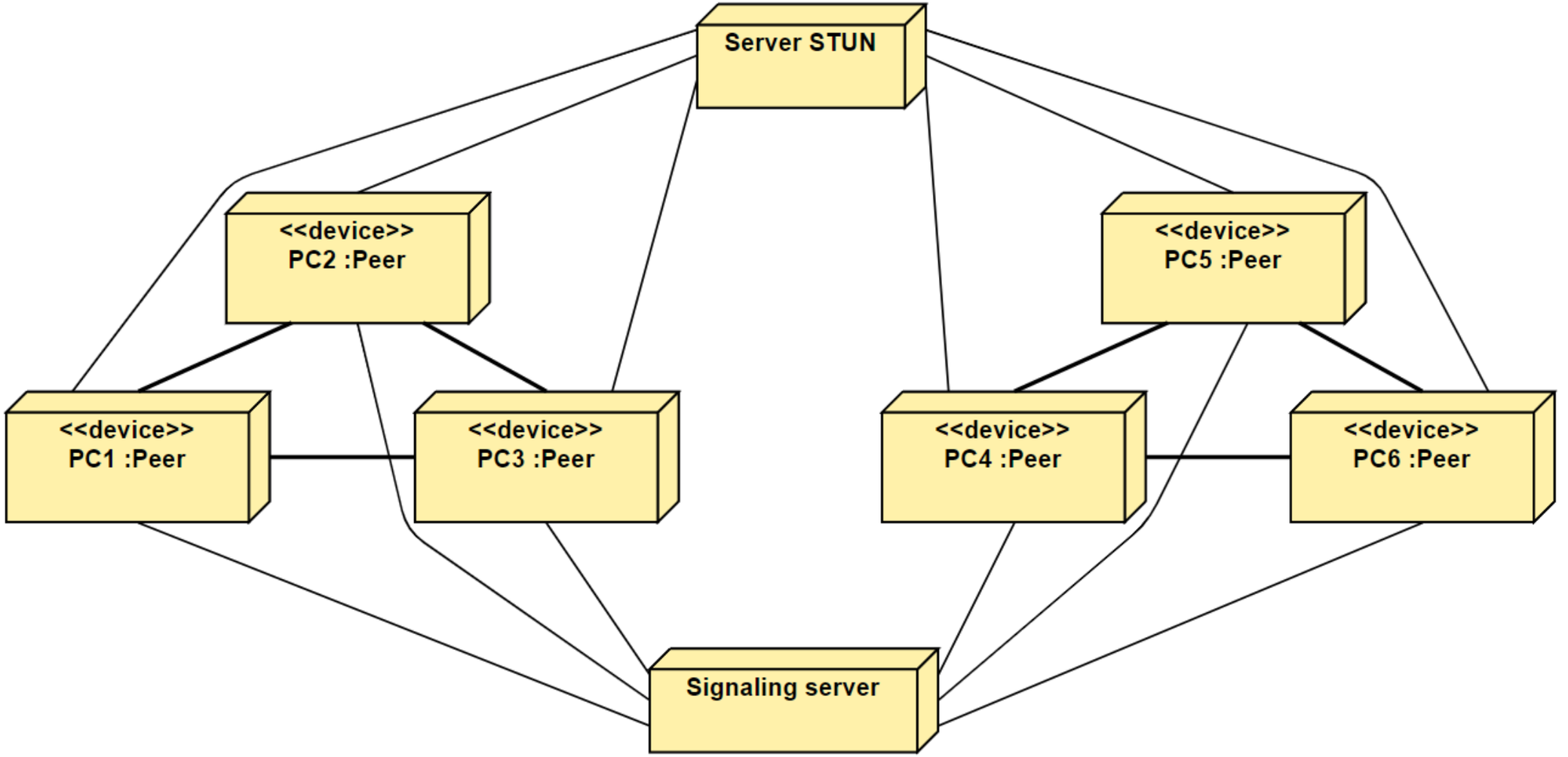
5. ARCHITETTURA

DIAGRAMMA DI DEPLOYMENT



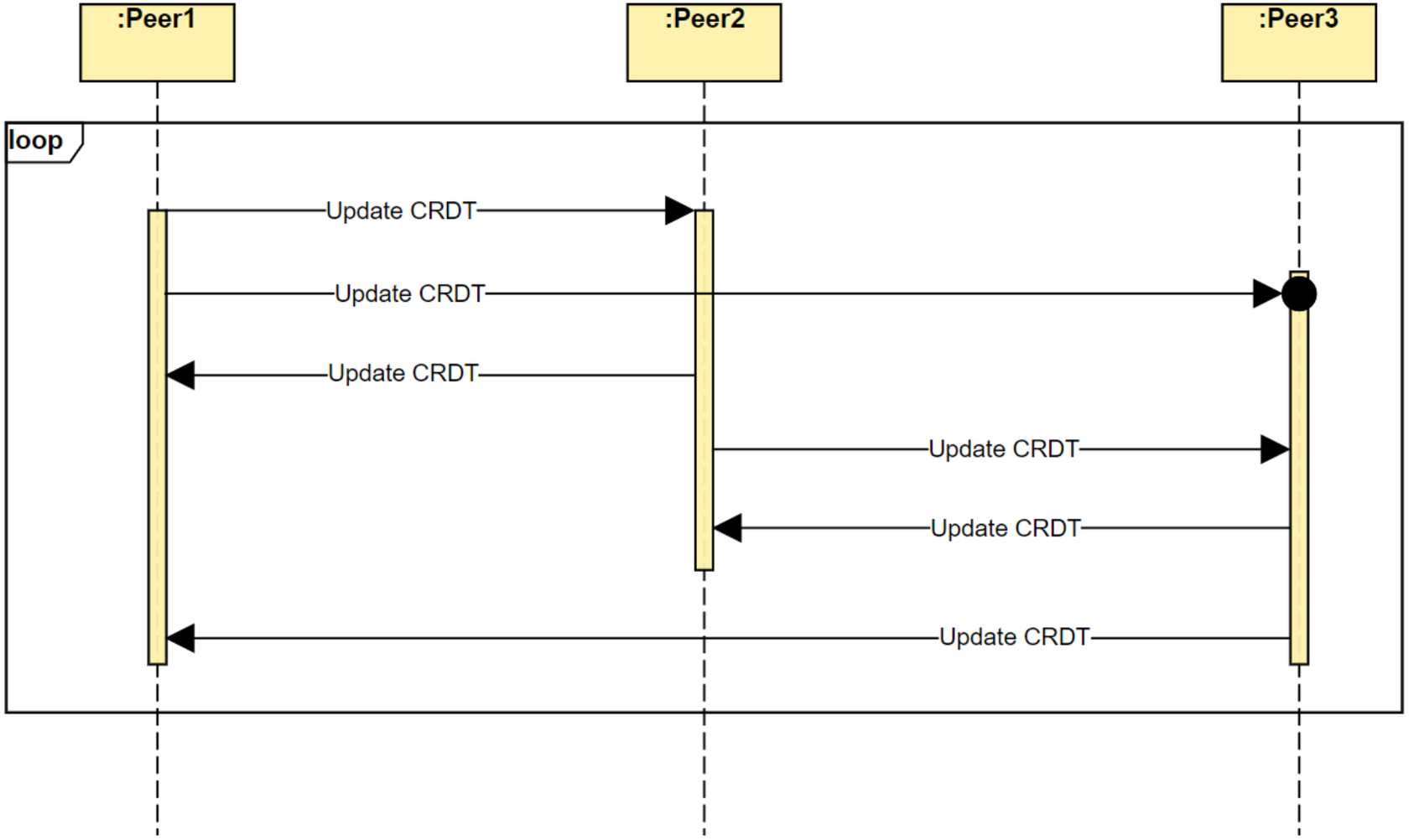
5. ARCHITETTURA

DIAGRAMMA DI DEPLOYMENT



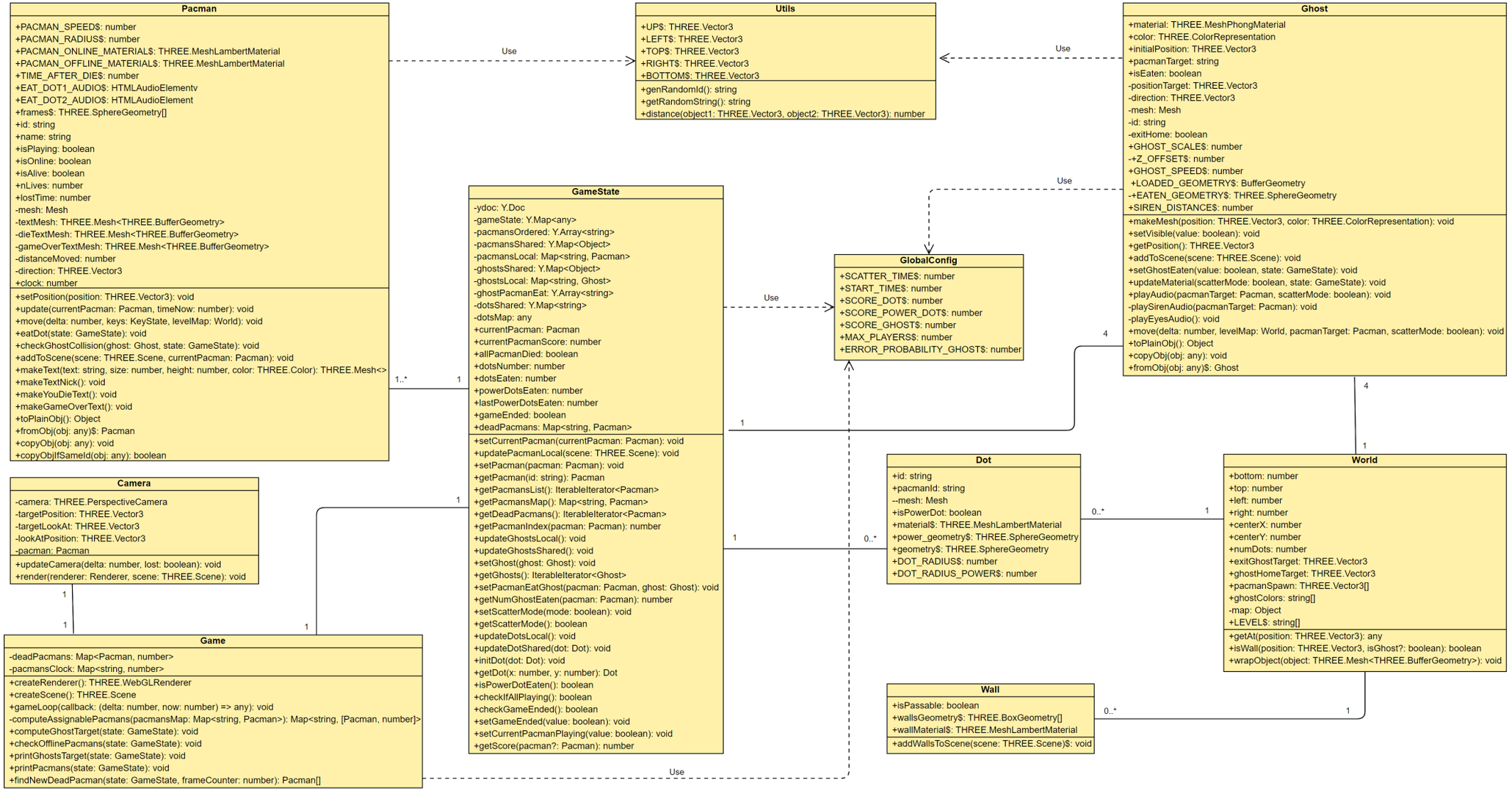
5. ARCHITETTURA

DIAGRAMMA DI SEQUENZA



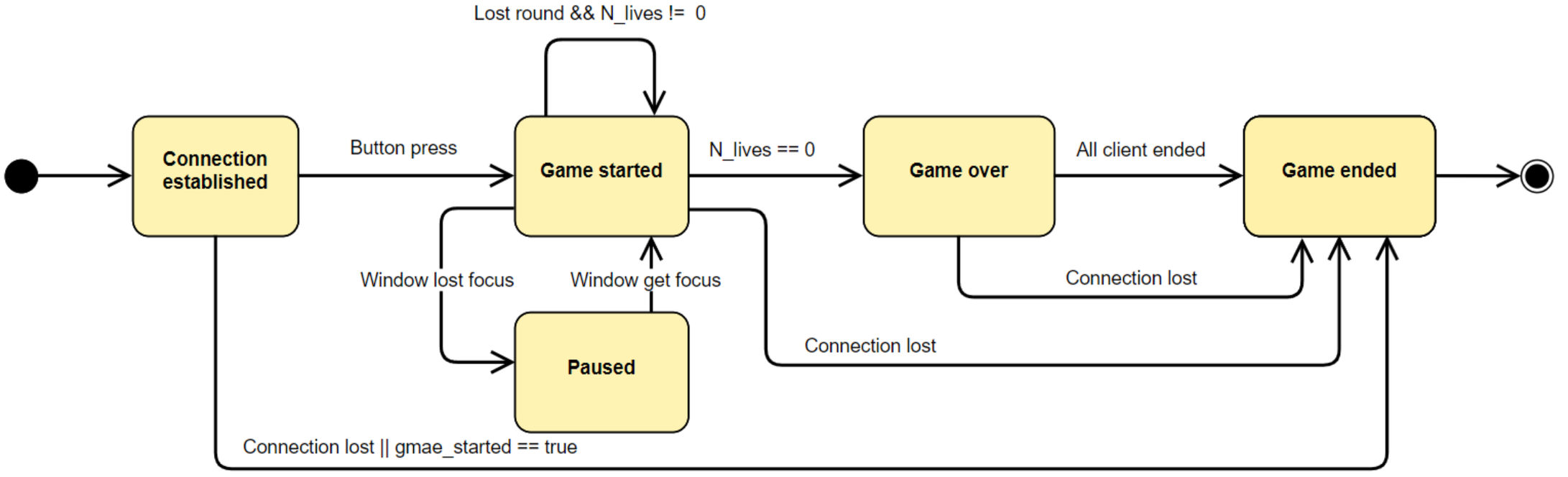
5. ARCHITETTURA

DIAGRAMMA DELLE CLASSI



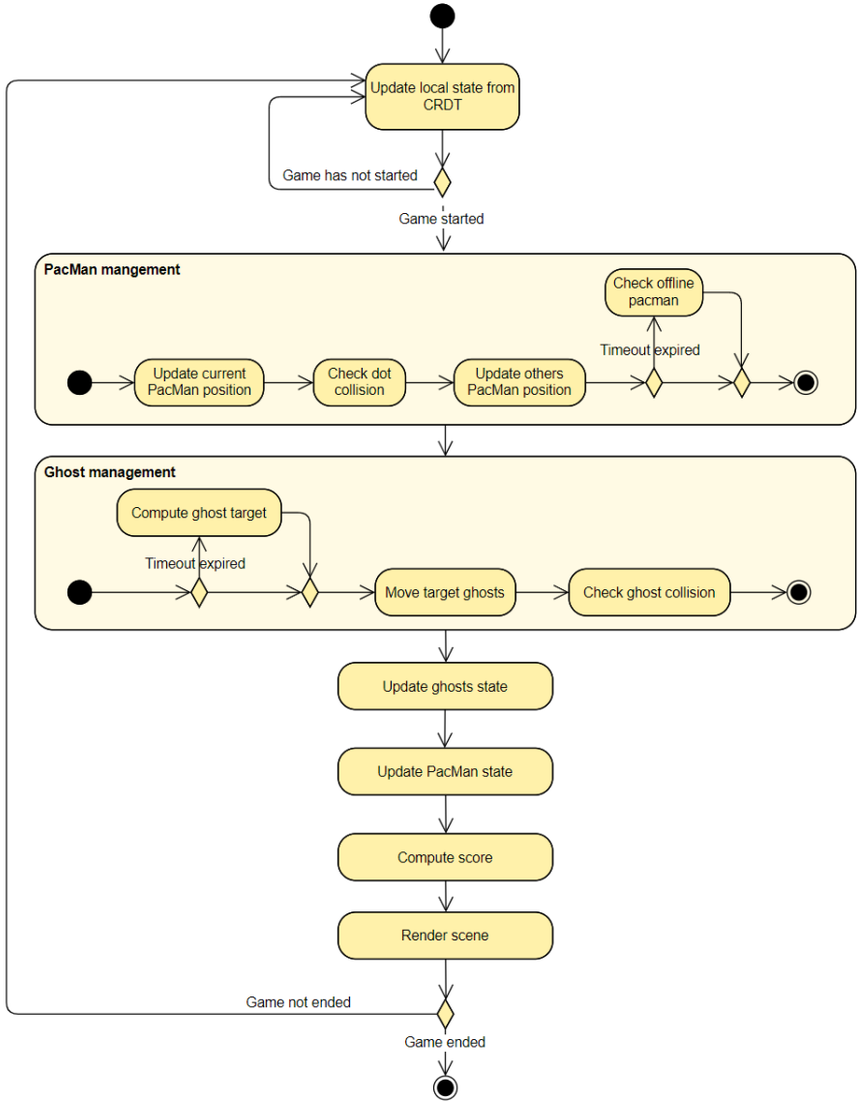
5. ARCHITETTURA

DIAGRAMMA DELLO STATO



5. ARCHITETTURA

DIAGRAMMA DI ATTIVITÀ



5. ARCHITETTURA

STRUTTURA CRDT

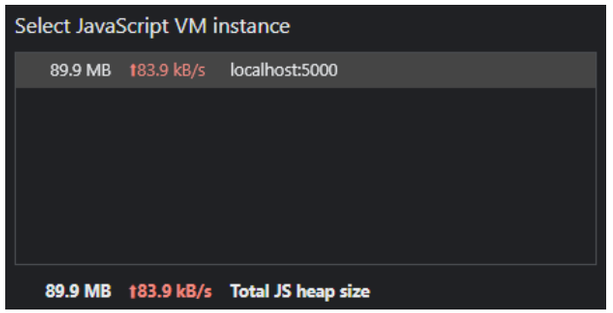
```
// Map<string, string>
let gameState: Y.Map<any>

// Array<pacman_id>
let pacmansOrdered: Y.Array<string>

// Map<pacman_id, pacman_object>
let pacmansShared: Y.Map<Object>

// Map<ghost_id, ghost_object>
let ghostsShared: Y.Map<Object>

// Map<dot_id, pacman_id>
let dotsShared: Y.Map<string>
```

6. DEMO

bit.ly/33c6Htq

7. VALUTAZIONI

- **FPS costante**
- Utilizzo della memoria dinamica **~90MB**
- Sistema altamente fault tollerant
  - Tollerati **n-1 fallimenti** (n = numero di peer)
  - **Caso ottimo:** tutti i giocatori terminano con lo stesso stato.
  - **Caso pessimo:** lo stato finale dei giocatori è diverso, ma comunque coerente nella loro versione locale del gioco.
- > 5 giocatori il sistema subisce dei rallentamenti percepibili



8. CONCLUSIONI

- **~2300** righe di codice
  - 10 classi
  - 5 componenti Svelte

- Approccio CRDT adatto a giochi realtime
- Il sistema rallenta all'aumentare dei giocatori (rete full-mesh)

Sviluppi futuri:

- Render della scena su thread separato (web-worker)
- Implementare meccanismi anti-cheating
- Migliorare la logica dei fantasmini

https://github.com/Vallasc/Distributed-PacMan

BIBLIOGRAFIA

[1] Eric A Brewer. Towards robust distributed systems. In PODC, volume 7, pages 343477–343502. Portland, OR, 2000.

[2] Werner Vogels. Eventually consistent. Commun. ACM, 52(1):40–44, jan 2009.

[3] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free Replicated Data Types. In Xavier Défago, Franck Petit, and Vincent Villain, editors, SSS 2011 - 13th International Symposium Stabilization, Safety, and Security of Distributed Systems, volume 6976 of Lecture Notes in Computer Science, pages 386–400, Grenoble, France, October 2011. Springer.

[4] Nuno Preguiça. Conflict-free replicated data types: An overview, 2018.

[5] Developing with riak kv. https://docs.riak.com/riak/kv/ latest/developing/data-types/index.html, (ultimo accesso 1 Febbraio, 2022).

[6] Anti-entropy using crdts on ha datastores. https: //archive.qconsf.com/sf2019/presentation/modern-cs, 2019 (ultimo accesso 1 Febbraio, 2022).

[7] Azure cosmos db: Pushing the frontier of globally distributed databases. https://azure.microsoft.com/en-us/blog/azurecosmos-db-pushing-the-frontier-of-globally-distributeddatabases/, 2018 (ultimo accesso 1 Febbraio, 2022).

[8] Active-active geo-distribution (crdts-based). https: //redis.com/redis-enterprise/technology/active-activegeo-distribution/, (ultimo accesso 1 Febbraio, 2022).