Implementazione del protocollo di Chord in una rete peer2peer

Corso di SISTEMI DISTRIBUITI E CLOUD COMPUTING
Facoltà di INGEGNERIA INFORMATICA

Simone Festa, mat. 0320408

Agenda

1. Introduzione
2. Server Registry
3. Finger Table
4. Gestione risorse
5. Join/Leave
6. Conclusioni

Introduzione

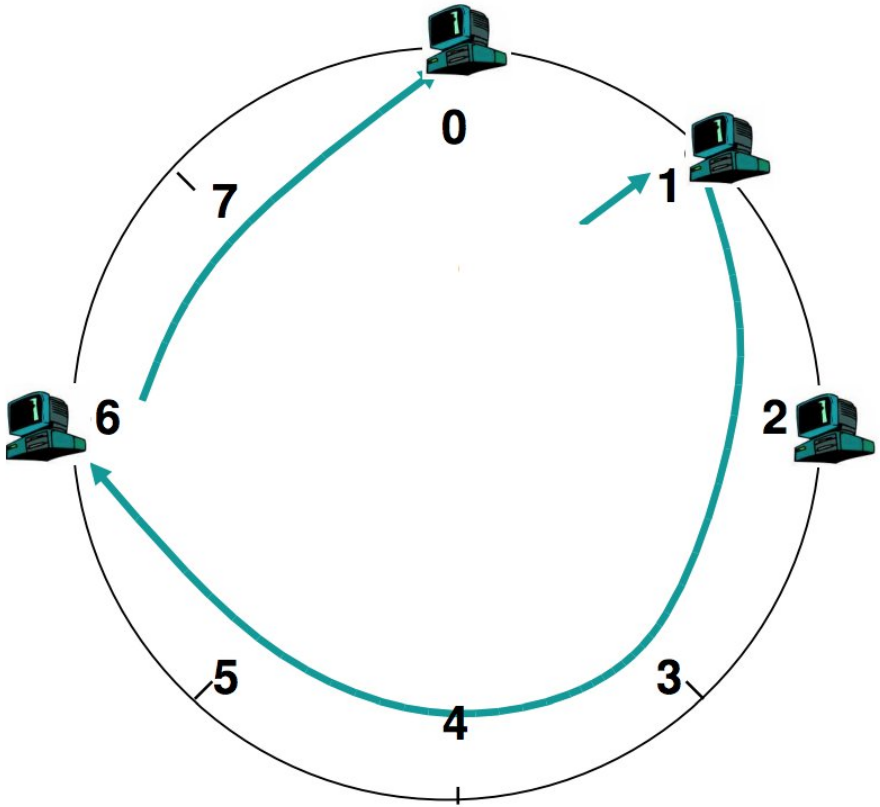
- Lo scopo del progetto è implementare, in un contesto di *overlay network strutturata*, l'algoritmo/protocollo di **Chord**.

- I nodi nell'anello sono in grado di **memorizzare** delle *risorse*.
  Particolare attenzione è data al **come** queste risorse vengono gestite e affidate.
  - Chord ha l'obiettivo di definire queste modalità.

- Di seguito, viene proposta una rappresentazione di una rete ad anello:

Introduzione

- Disposizione ad *anello* a livello *overlay*, non fisico.
- Nodi e Risorse vengono mappate nello stesso *spazio contiguo* **(consistent hashing)**
- Ogni nodo conosce *m* vicini, dove *m* è il numero di bits usati per identificare univocamente un elemento.



- **Come si introduce un nuovo nodo nella rete?**
- **Come comunicano i nodi nella rete?**
- **Come vengono assegnate le risorse?**

Agenda

Introduzione | **Server Registry** | Finger Table | Gestione risorse | Join/Leave | Conclusioni

Server Registry

- Permette ad un *client esterno* di interagire col sistema. IP statico e noto a tutti.

- Fornisce ad un *nuovo* nodo un punto di accesso nel sistema, fornendogli il contatto dei nodi adiacenti con cui instaurare la connessione.

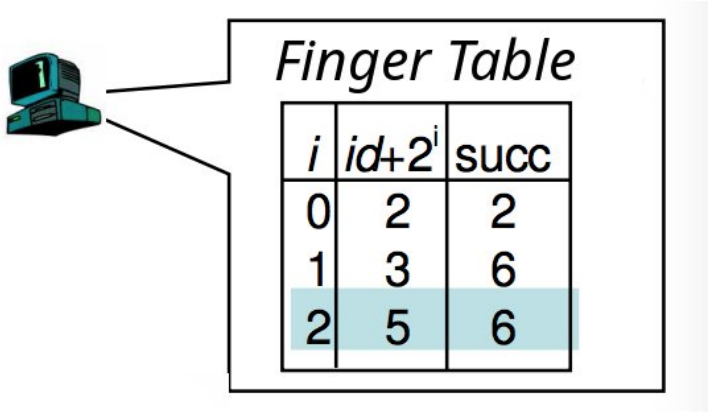
- Se un nodo lascia involontariamente il sistema, il Registry fornisce supporto per la riconfigurazione dei nodi limitrofi.

Agenda

Introduzione | Server Registry | **Finger Table** | Gestione risorse | Join/Leave | Conclusioni

Finger Table

- Ogni nodo possiede una propria FT.
  Tutte le FT hanno lo stesso numero di righe.



- La riga i-esima della FT di un *nodo p*, è così calcolata:

$$\mathbf{FT_p[i]} = \mathbf{succ}\left(\mathbf{p} + \mathbf{2^{i-1}}\right) \mathbf{\ mod\ 2^m}$$

Nodo responsabile dell'identificativo posto come argomento.
Il risultato è ottenuto ricorsivamente, interrogando i nodi successori.

- La Finger Table mantiene una lista di nodi progressivamente distanti.
  Fornisce una conoscenza ben definita dei nodi vicini e più approssimata all'aumentare della distanza.

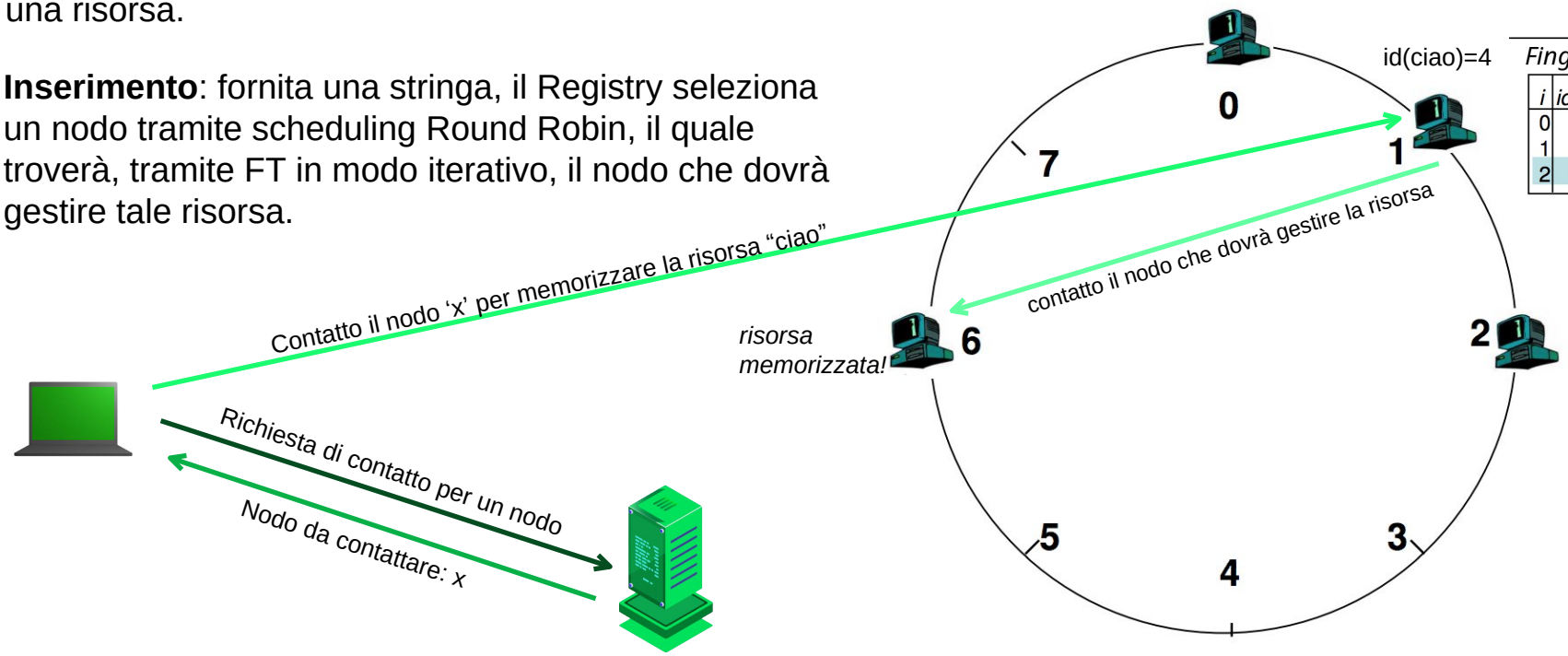
- Ciò consente una ricerca veloce, in $O(\log N)$,
  senza interrogare tutto l'anello.

Agenda

Introduzione | Server Registry | Finger Table | **Gestione risorse** | Join/Leave | Conclusioni

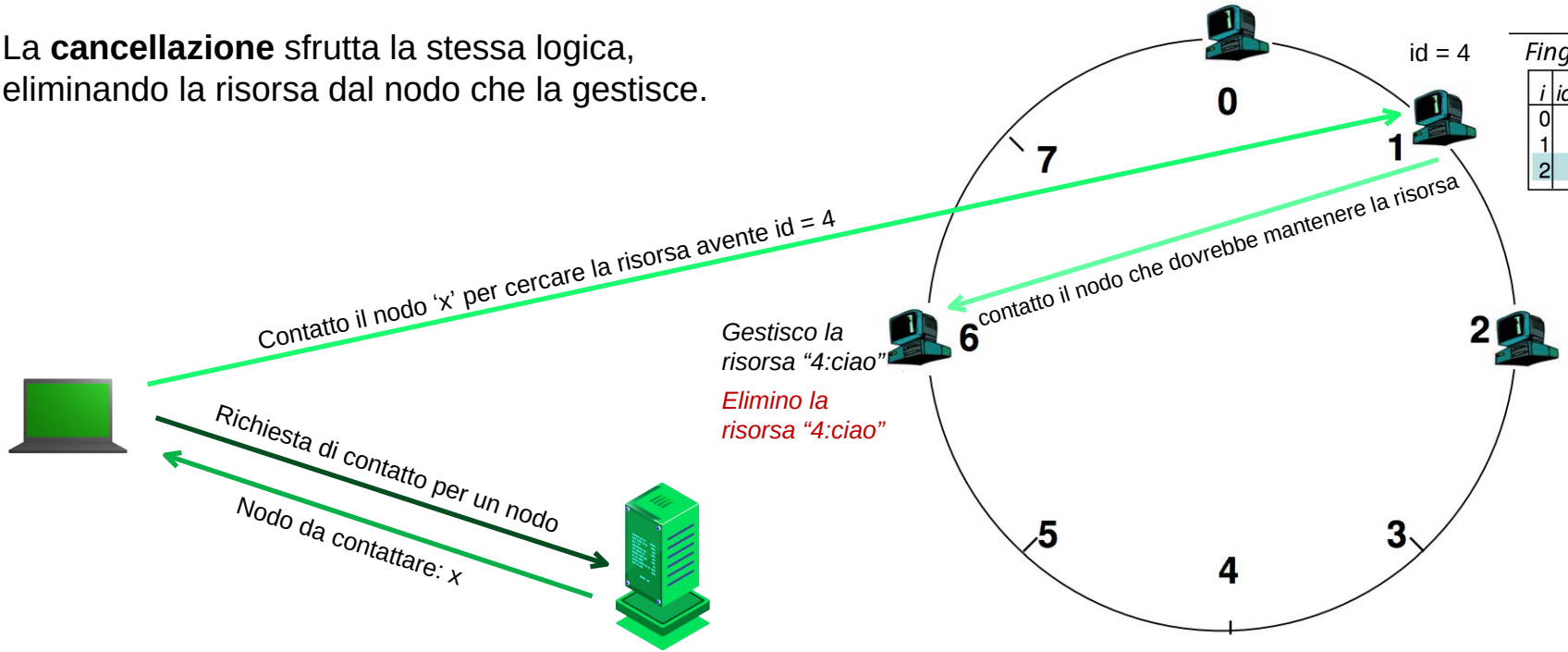
Gestione delle risorse

- Il sistema supporta l'*inserimento*, la *ricerca* e la *cancellazione* di una risorsa.
- **Inserimento**: fornita una stringa, il Registry seleziona un nodo tramite scheduling Round Robin, il quale troverà, tramite FT in modo iterativo, il nodo che dovrà gestire tale risorsa.



Gestione delle risorse

- La **ricerca** di una risorsa è eseguita in modo simile.
- La **cancellazione** sfrutta la stessa logica, eliminando la risorsa dal nodo che la gestisce.

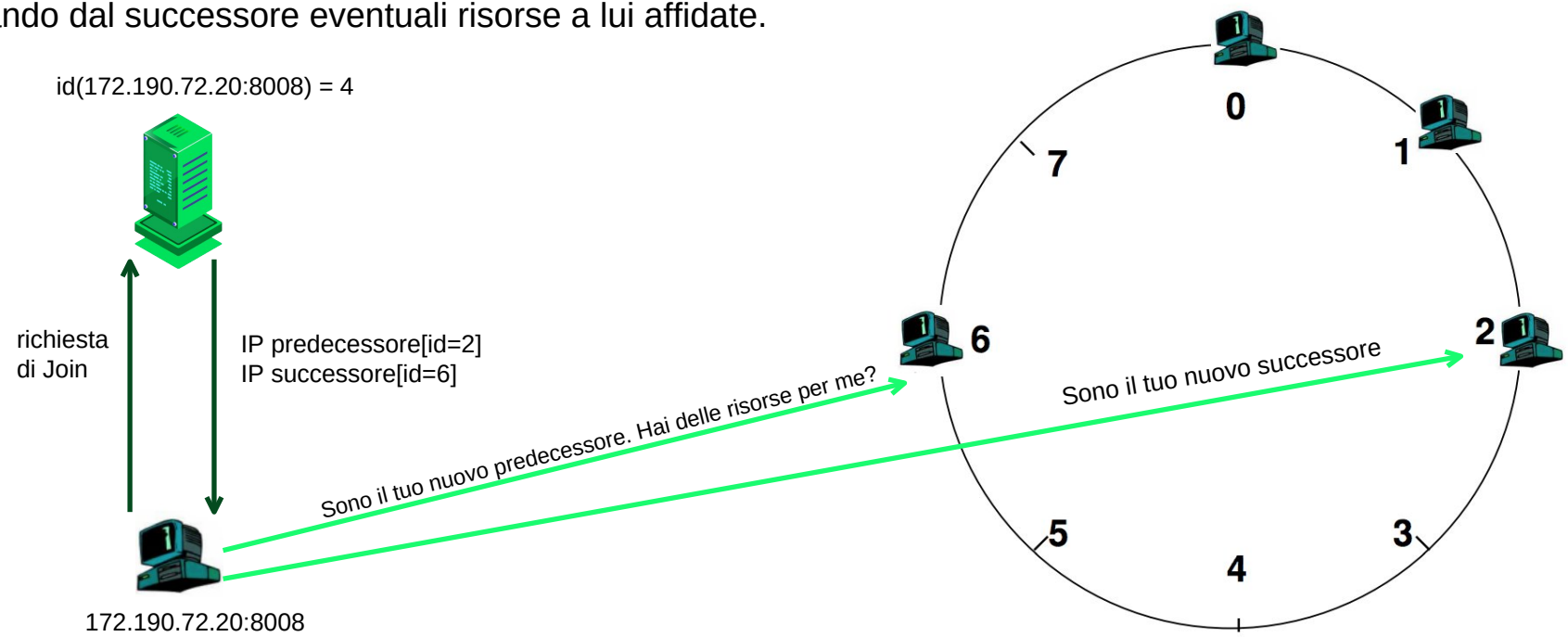


Agenda

Introduzione | Server Registry | Finger Table | Gestione risorse | **Join/Leave** | Conclusioni

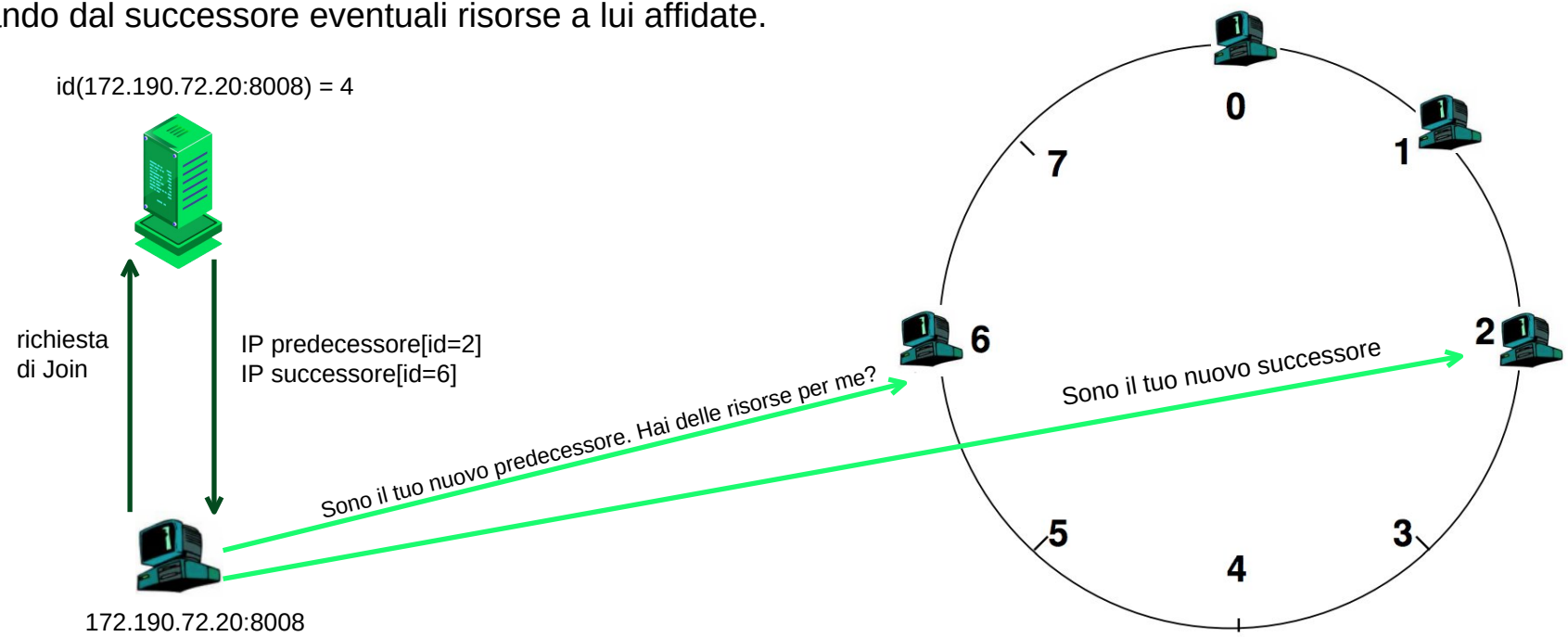
Join/Leave

- Nella fase di **Join**, il nodo entrante comunica col Registry per conoscere i suoi vicini. Li contatterà per inserirsi correttamente, prelevando dal successore eventuali risorse a lui affidate.



Join/Leave

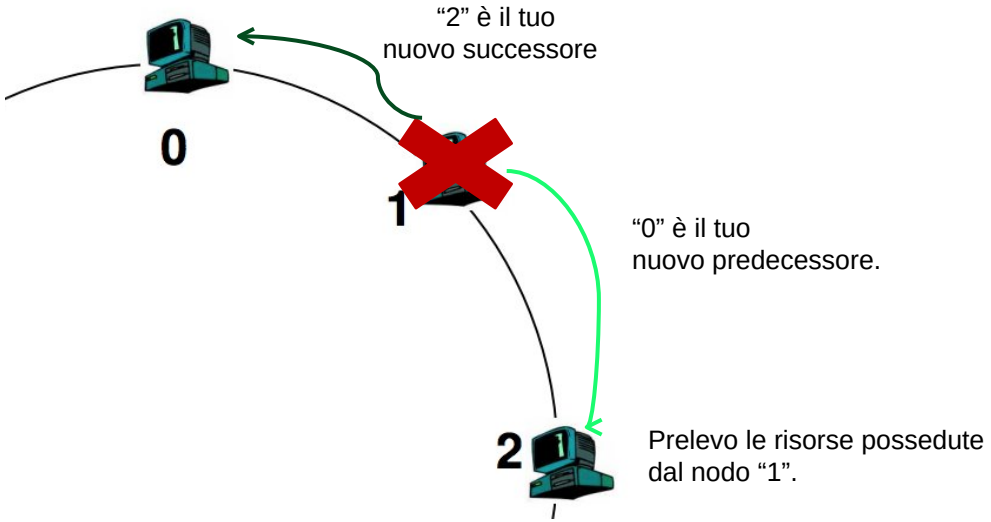
- Nella fase di **Join**, il nodo entrante comunica col Registry per conoscere i suoi vicini. Li contatterà per inserirsi correttamente, prelevando dal successore eventuali risorse a lui affidate.



Join/Leave

- Nella fase di **Leave** (controllata), i nodi adiacenti al nodo da rimuovere verranno contattati per aggiornare la loro conoscenza sui nodi precedenti e successori.

- Esempio: rimozione del nodo “1”.



Agenda

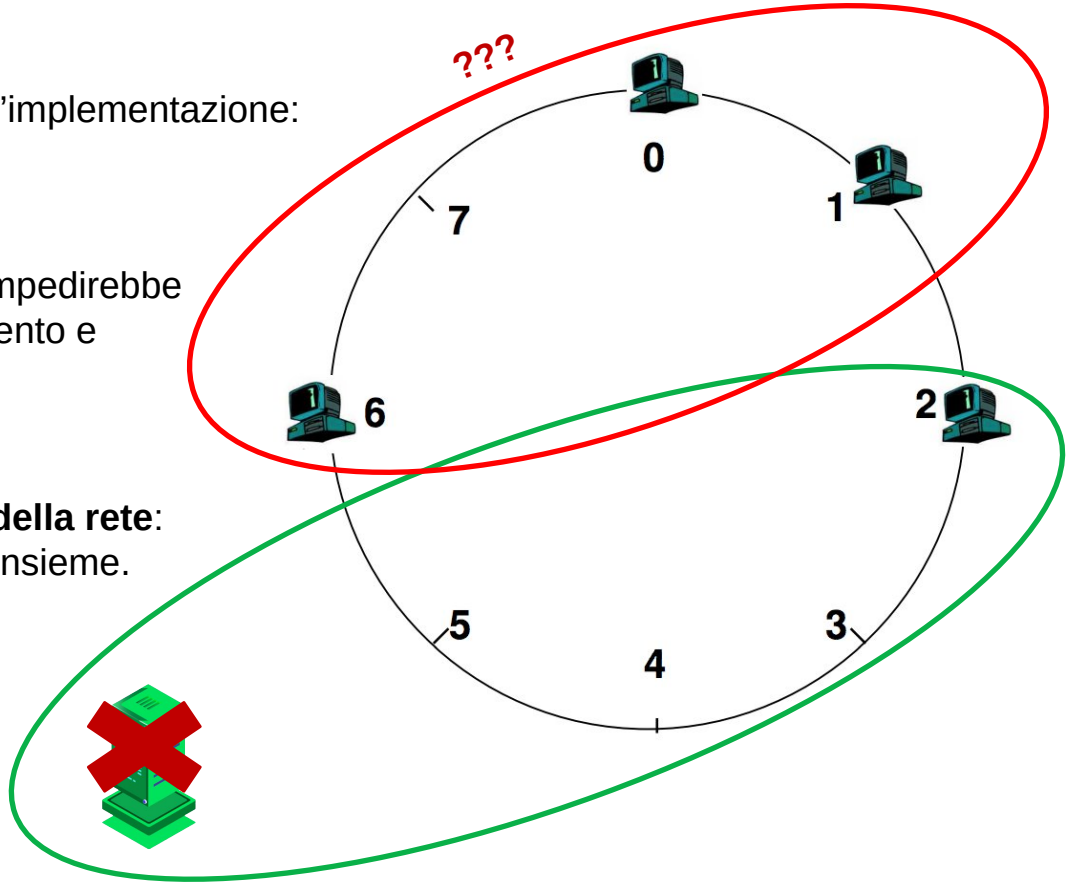
- Introduzione
- Server Registry
- Finger Table
- Gestione risorse
- Join/Leave
- Conclusioni

Conclusioni

- Di seguito, esaminiamo alcune limitazioni dell'implementazione:

- **Server Registry bottleneck**: un suo guasto impedirebbe il corretto svolgimento di operazioni di inserimento e posizionamento dei nodi.

- **Non tolleranza in caso di partizionamento della rete**: Il registry verrebbe associato ad un solo sottoinsieme.



Repository

https://github.com/simonefesta/Chord_SDCC

Grazie per l’attenzione!

Simone Festa, mat. 0320408