

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

Introduzione

Server
Registry

Finger Table

Gestione
risorse

Join/Leave

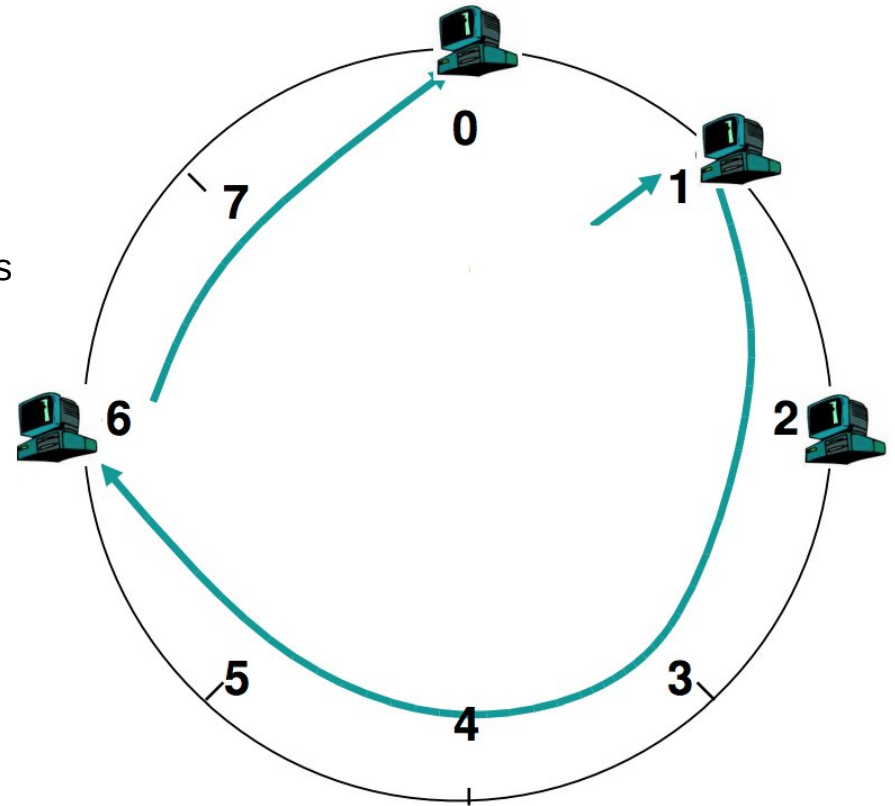
Conclusioni

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



Agenda



Server Registry

- Permette ad un client esterno di interagire col sistema. IP statico e noto a tutti.
- Permette ad un nodo di conoscere i propri nodi adiacenti, con cui instaurare la connessione.
- Mantenendo una lista ordinata dei nodi presenti, favorisce il calcolo delle Finger Table per le varie entità.
- Un nuovo nodo si interfacerà con il Registry. Calcolato l'identificatore di questo nuovo nodo, il Registry fornirà le informazioni dei nodi adiacenti per il corretto posizionamento.



Agenda




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:

$$FT_p[i] = \text{succ}(p + 2^{i-1}) \bmod 2^m$$

Nodo responsabile dell'identificativo posto come argomento.



Finger Table

i	$id + 2^i$	succ
0	2	2
1	3	6
2	5	6

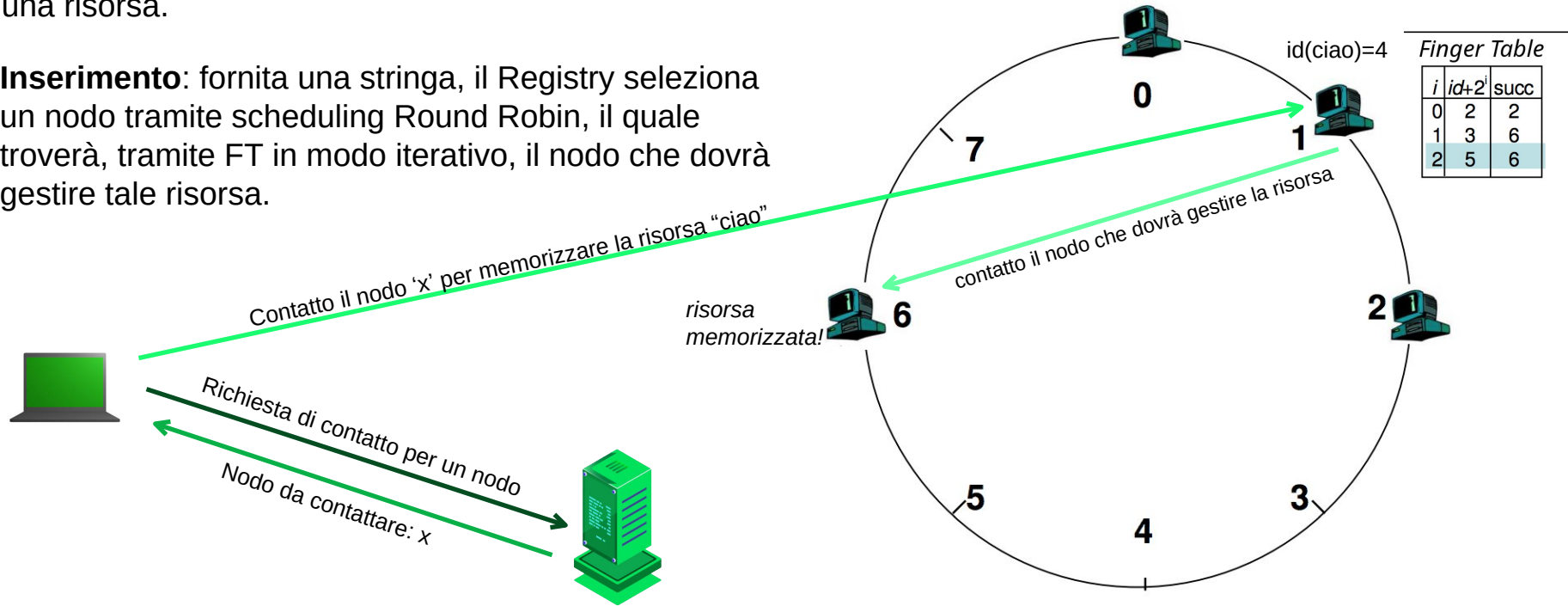
- La Finger Table mantiene una lista di nodi progressivamente distanti.
Fornisce una conoscenza ben definita dei nodi vicini e più approssimata all'aumentare della distanza.
- Ciò consente una ricerca veloce, in $O(\log N)$,
senza interrogare tutto l'anello.

Agenda



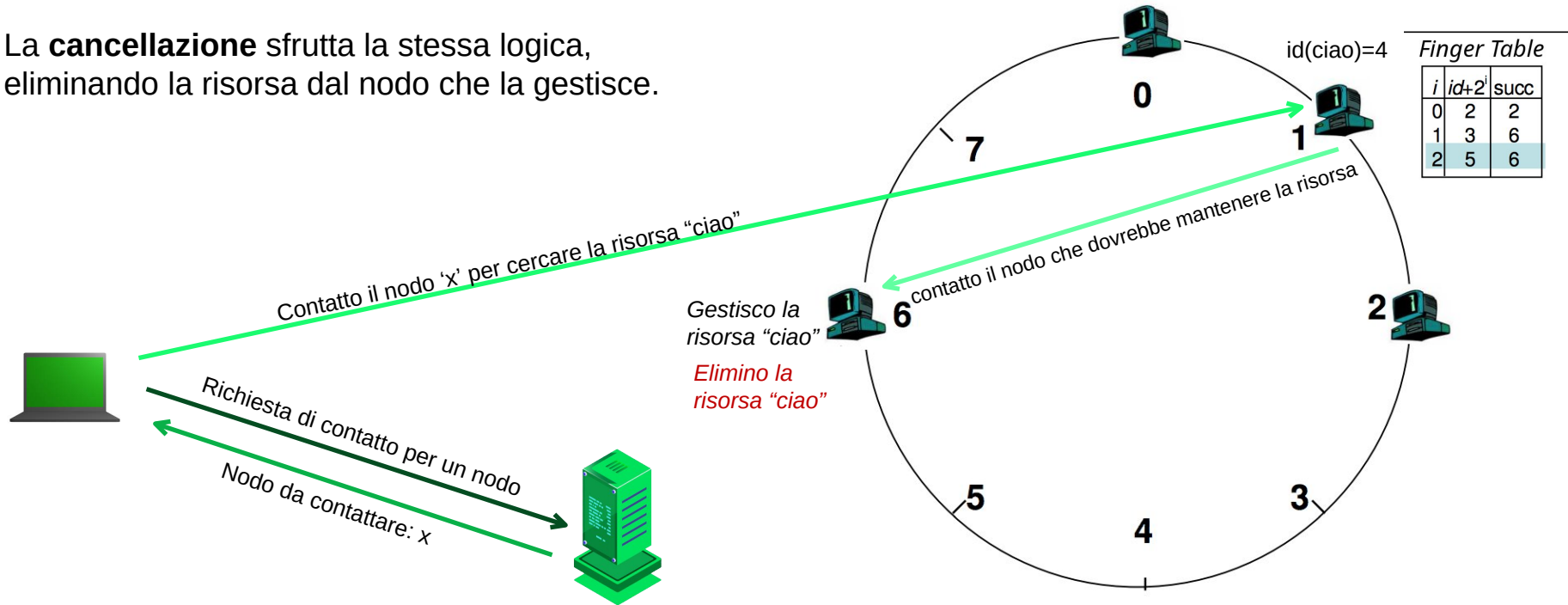
Gestione delle risorse

- 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.

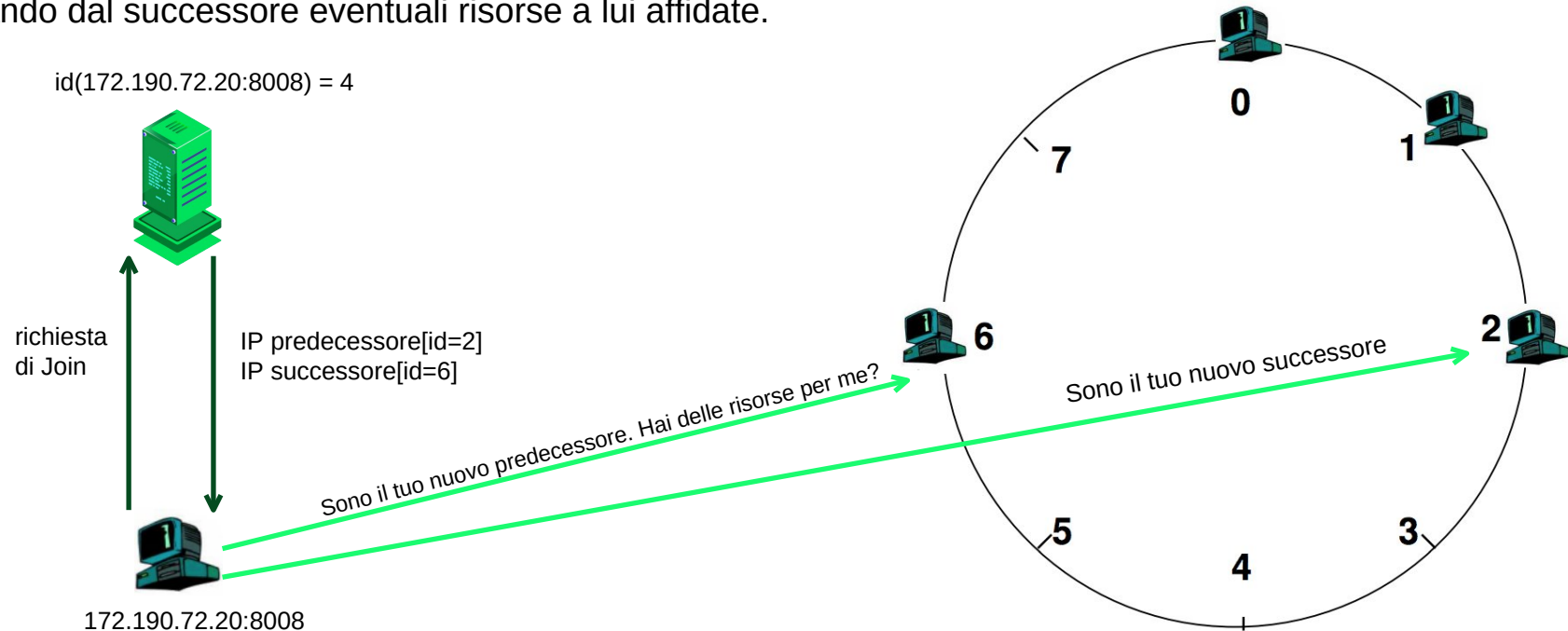


Agenda



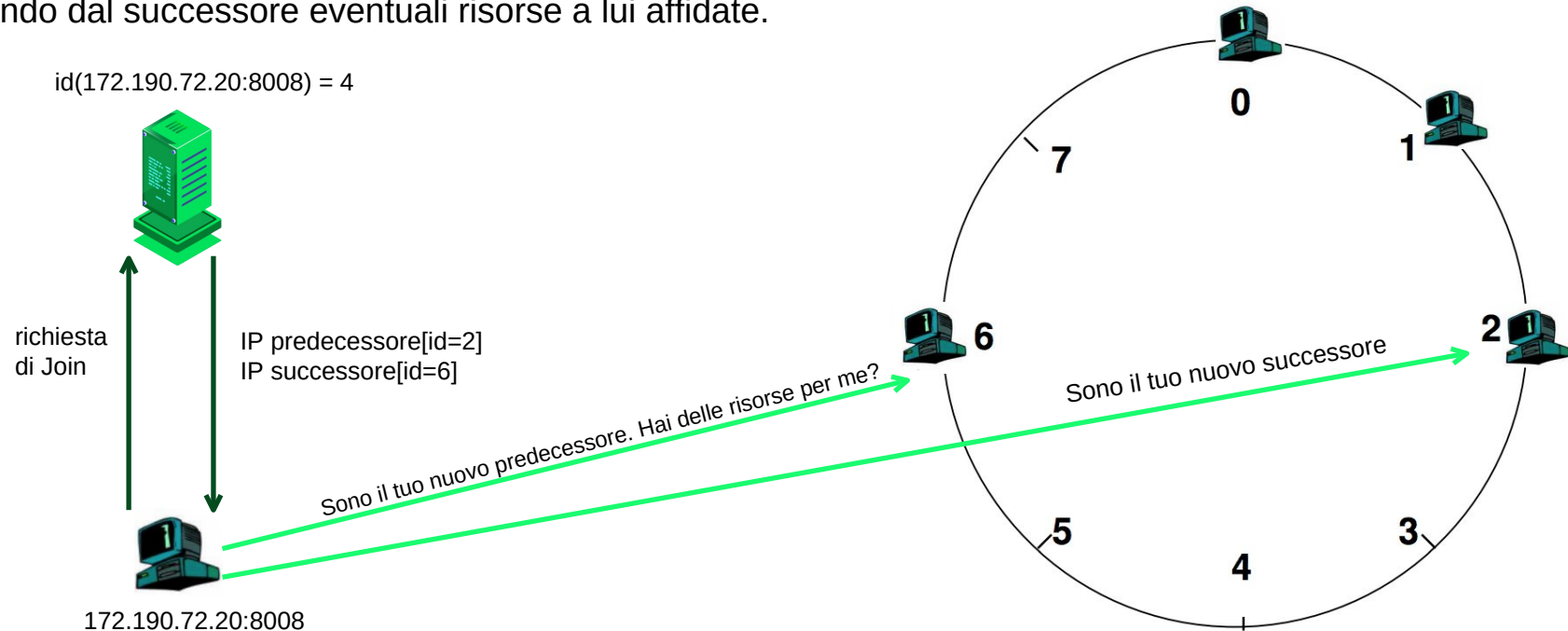
Join/Leave

- 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.



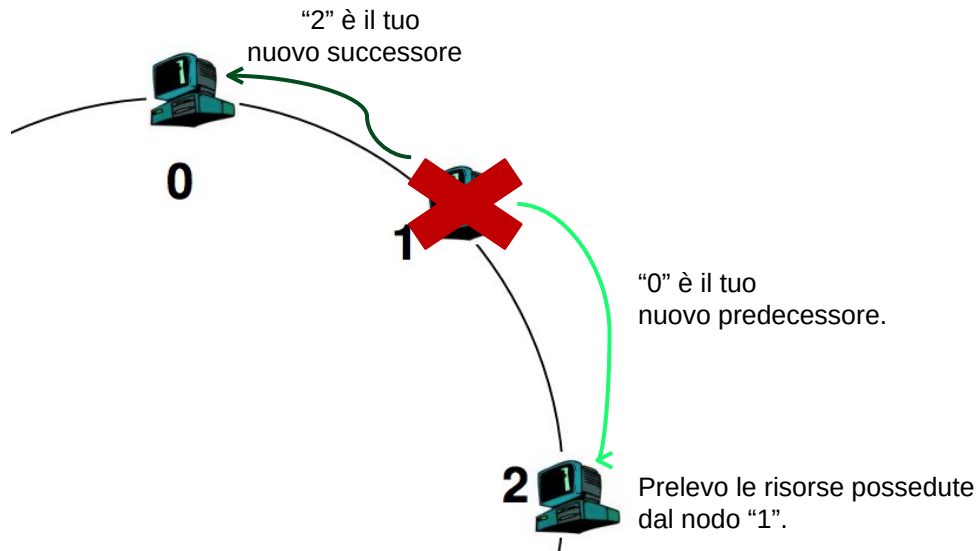
Join/Leave

- 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".

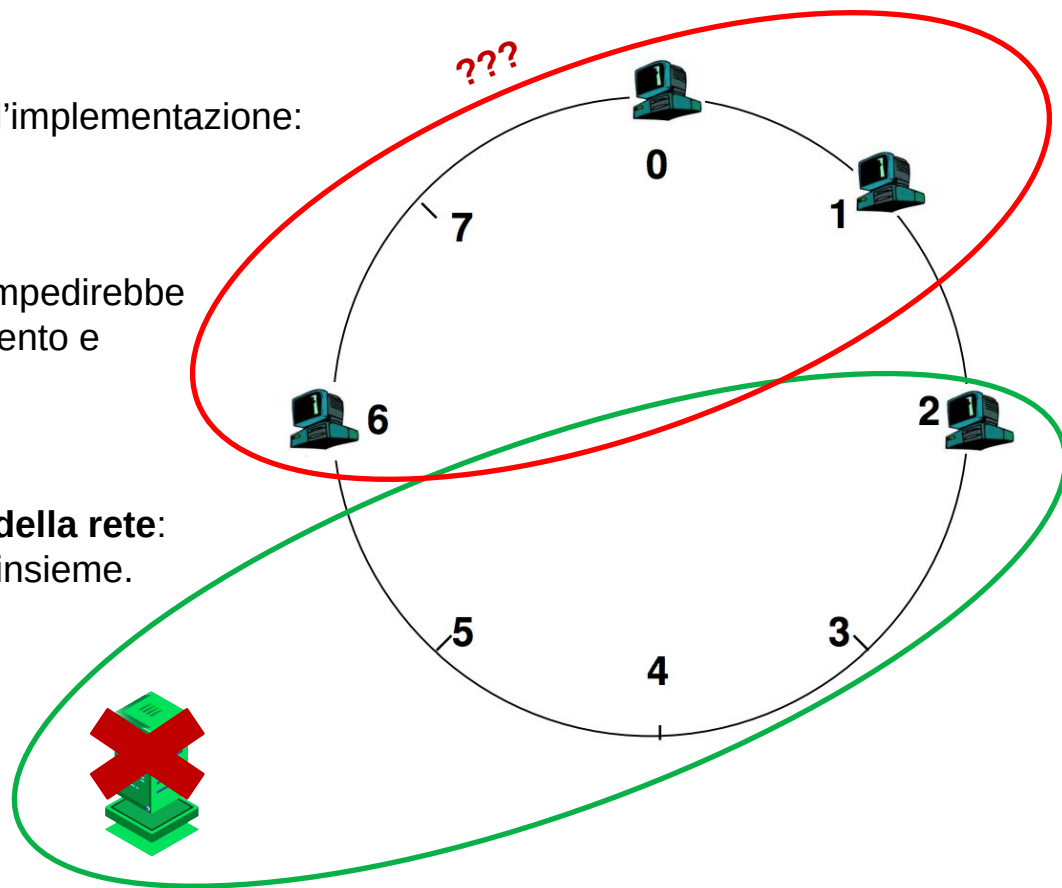


Agenda



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



GitHub

https://github.com/simonefesta/Chord_SDCC

Grazie per l'attenzione!

Simone Festa, mat. 0320408

