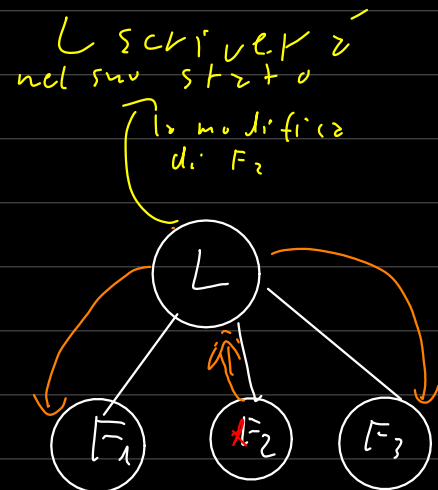
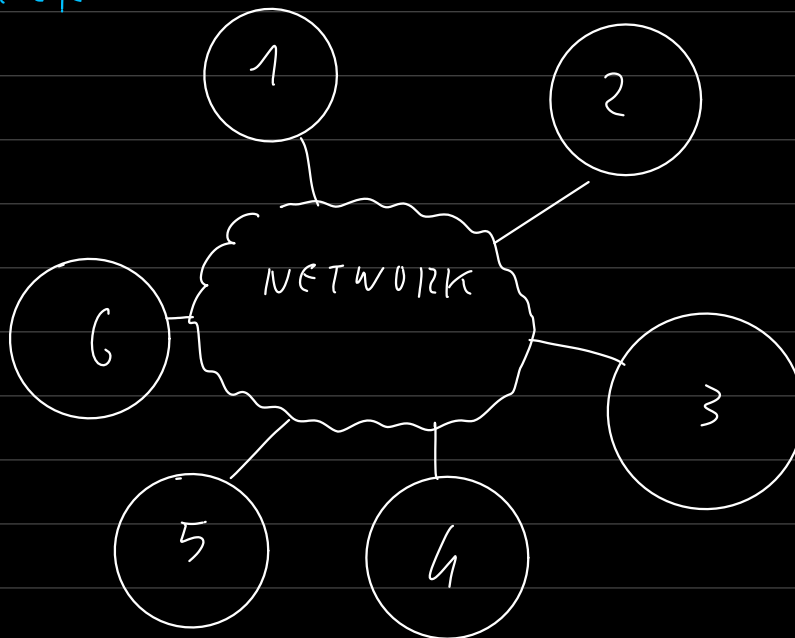


RAFT - ARCHIVIAZIONE

OBBIETTIVI:

- 1- CREAZIONE SISTEMA DI ARCHIVIAZIONE
DISTRIBUITO SENZA CENTRALIZZAZIONE (CLUSTER)
 - 2- UPLOAD FILES (write)
 - 3- DOWNLOAD FILES (read)
 - 4- DELETE FILES (delete)
-

1) CLUSTER

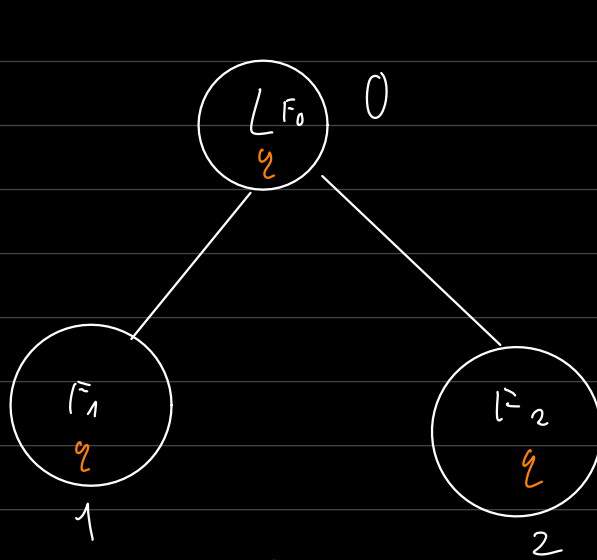


INVARIANTI:

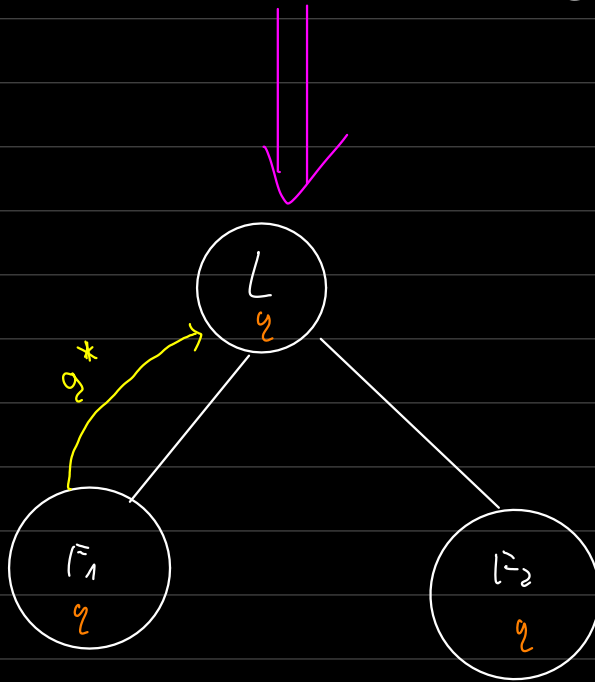
- 1- Tutti i nodi convergono allo stesso stato (a un certo punto avranno gli stessi dati)

⇒ RAFT

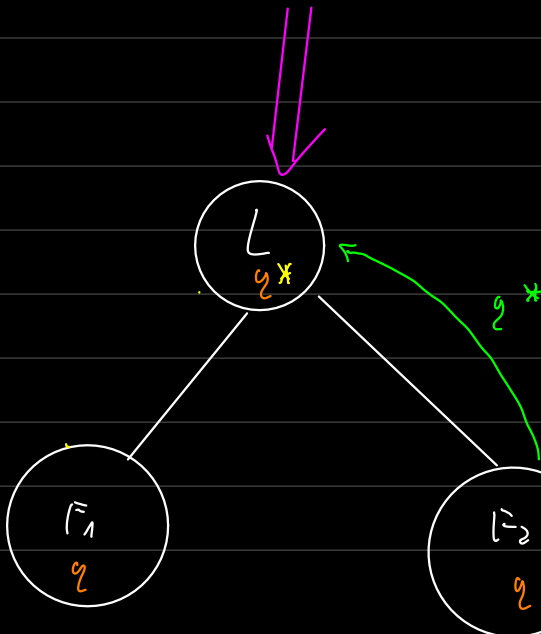
CRITICITY



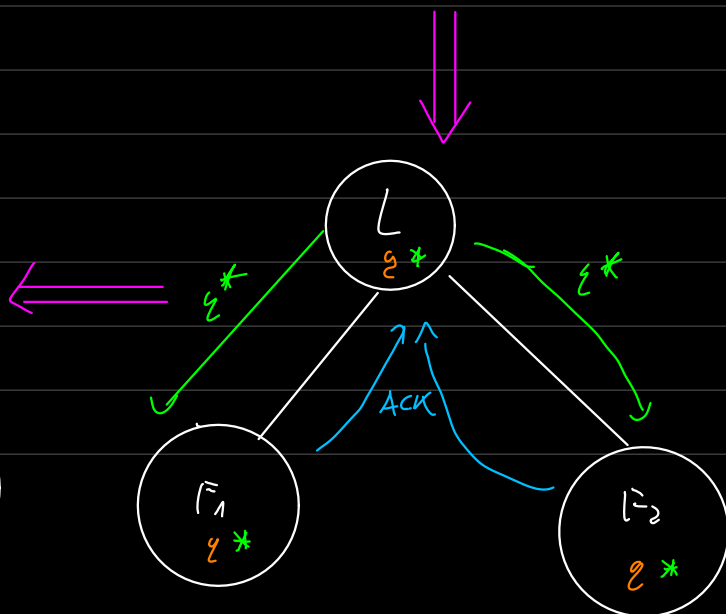
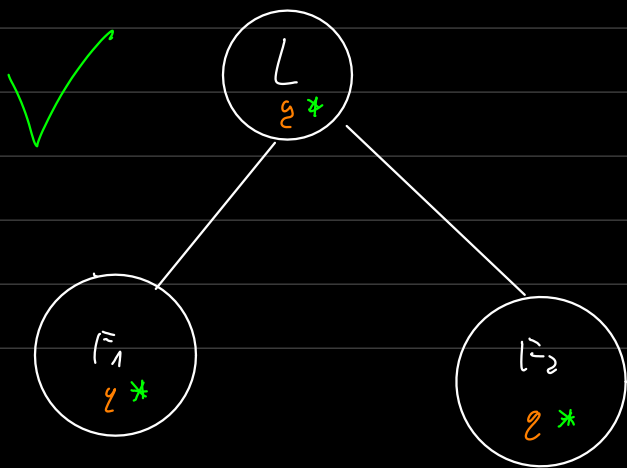
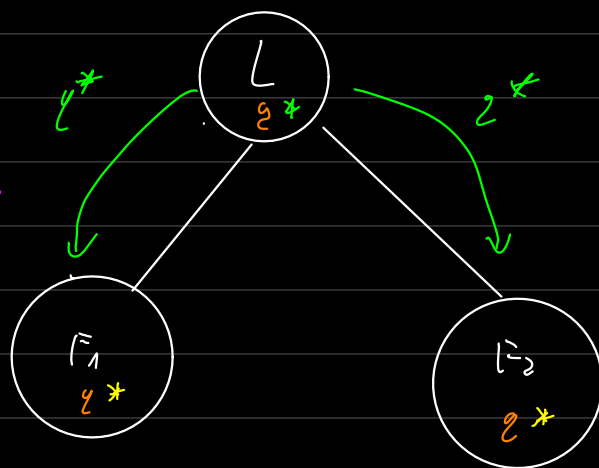
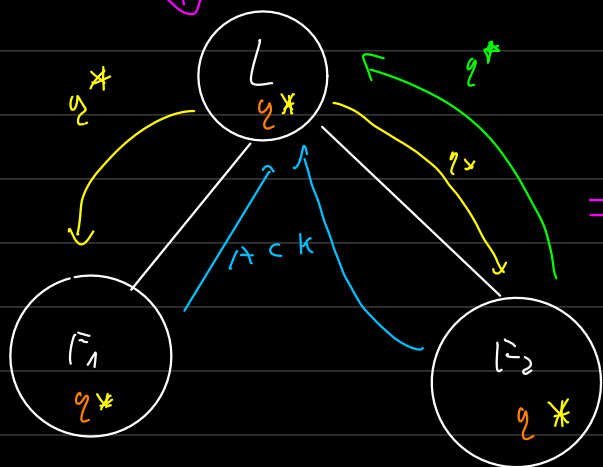
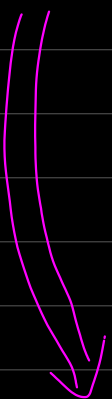
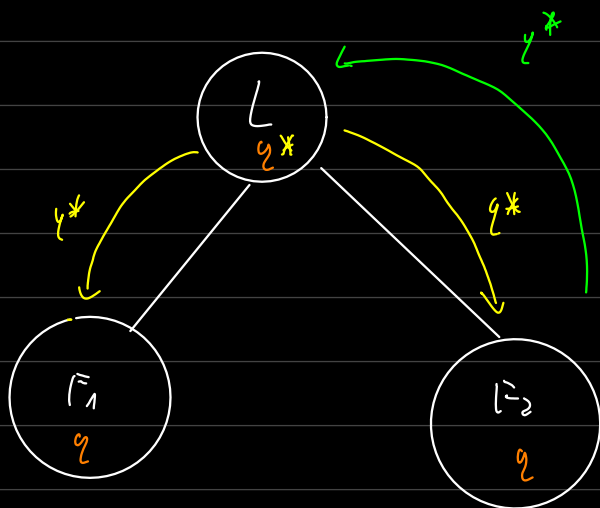
$l_1 l_2 = 1 = ?$
= "hello world"



$g^* = \text{"hello pluto"}$



$g^* = \text{"hello pettingo"}$



Definizione dei nodi:

- VIRTUAL MACHINE:

Mantengo forte isolamento dei nodi, mantengo indipendenza dei nodi, mantengo potenziali grosse differenze tra i nodi, non ho, possibilmente, dipendenze comuni tra i nodi.

Comunicazione:

- Tramite HTTP:

- formato invio dati: **JSON**

STORAGE:

- NO FILESYSTEM DISTRIBUITO

- ogni nodo, internamente, deve avere un filesystem per funzionare

- lo STATO interno di ogni nodo

è una directory con all'interno

tutti i file dove ogni file

avrà un **ID** che verrà generato

a partire dalla hash di:

virtual directory + nome del
di storage file

LEADER ELECTION:

Ogni nodo ha un ID

⇒ evoluzione di Bully algorithm:

ogni nodo ha due timeout:

- 1- timeout heartbeat leader: unico per tutti
- 2- timeout invio richiesta elezione: random su ogni nodo.
(indica quanto tempo un nodo deve attendere prima di inviare una richiesta di elezione)

Poi ogni nodo:

- se non ha indetto un'elezione accetta e invia a tutti l'ID della persona che vota. Ogni nodo conta chi ha più voti e quando c'è una maggioranza il nuovo leader è eletto
- Per essere eletto ogni nodo deve essere in uno stato consistente. Per verificare lo stato il candidato, al momento della candidatura, deve essere accettato da una maggioranza di nodi i quali dovranno verificare che lo stato del candidato sia aggiornato almeno quanto il loro.

~ MESSAGGIO DA INVIARE:

TERM : indice numero

init = 0,

increments monotonicamente

si incrementa nel tempo

(TERM_c must be \geq TERM_G)

ID : candidato id

LOGICAL
CLOCK

||
✓

serve

→ individuare
info
obsolete
(ex. leader
morti).

LAST

LONG(LLI) : indice dell'ultima entry
del candidato

LAST

LOG (LLT)

TERM

TERM dell'ultima
entry del candidato

in tero:
come work lo
→ solo 010

LEADER (già presente):

situazioni di operatività del cluster:

cluster stabile: nessun nodo del cluster vuole modificare lo stato del cluster e ogni nodo ha lo stato conforme

⇒ invia heartbeat
(append entry with no data)
invia messaggio di
aggiornamento dello
stato senza dati da
modificare.

(Lo fa in periodi morti
quando quindi non c'è
attività nel cluster)

cluster instabile: a) il leader ha modifiche di stato da propagare
gli altri nodi sono stabili.

⇒ invoca RPC append
entry e allega alla
chiamata la entry da
replicare. il suo stato
è per ora volatile.

Una volta che la maggioranza dei
follower hanno ricevuto la modifica
il leader aggiorna il suo stato
persistente.

come faccio
a sapere
se hanno
ricevuto?

↳ RPC `append entry` ha un valore di ritorno:

{

TERM_FOLLOWER: CURRENT TERM del follower che il leader vorrebbe aggiornare se stesso.

BOOL SUCCESS: TRUE/FALSE

}

il TERM value nella RPC `APPEND ENTRY` è usato sia in input sia in output e viene usato dai nodi per aggiornare il proprio TERM se quello ricevuto è maggiore del proprio

CASI CRITICI:

↳ candidato

1) FOLLOWER CRASH e non fornisce valore di ritorno di RPC:

Leader riprova a mandare la chiamata RPC fino a che la maggioranza dei follower hanno memorizzato la entry.

(assumendo che la configurazione sia stabile, ovvero i nodi follower sono stabili)

(Se dopo K chiamate RPC il follower non risponde, tale nodo è da considerarsi morto, il leader cambia la sua configurazione in `term` e lo invia a tutti.)

2) Un follower si aggiunge durante invio di aggiornamento di stato, Gestito sotto (↓).

3) casi di cambio di configuration (↓↓)


4) Il leader crasha prima/dopo di propagare lo stato il log può restare inconsistente.

Ogni volta che invia append-entry, tramite return of RPC, check di consistenza LOG (next index).

Se presente inconsistenza il leader retrocede fino al log comune e ordina al follower di eliminare le entry in più, alla fine il leader invia al follower le entry del log successive al suo punto in comune.

b) il leader ha modifiche di stato da propagare gli altri nodi sono instabili.

già gestito



CAMBIO CONFIGURAZIONE CLUSTER

due fasi:

Unione delle configurazioni:
vecchia e nuova.

1- TRANSIZIONE (JOINT CONSENSUS):

combinazione delle due configurazioni:

1- Lo stato del sistema è copiato in tutti i nodi.

1.b - configurazione intermedia
commitata;

* Se il vecchio è ancora attivo

rimane leader e invia configurazione nuova commitandola e vecchio viene scartato, i nodi non presenti nella nuova vengono chiesti.

* se il vecchio leader muore

si indice elezione e solo

i nodi che hanno ricevuto

la configurazione intermedia

possono candidarsi. Nuovo

leader invia configurazione

nuova come prima

1. b - configurazione non-committata:

* Il leader vecchio si è morto

prima di committare configurazione,
viene quindi indetta nuova elezione
tra tutti i nodi che hanno:

CONFIGURAZIONE INTERMEDIA

OPPURE (v)

CONFIGURAZIONE VECCHIA

Attivo nuovo leader:

1- leader ha vecchia configurazione,
se riceve da un follower
un aggiornamento di stato
con configurazione intermedia
si riparte da 1-

2- leader ha configurazione intermedia
si riparte da 1-

2 - FASE CONFIGURAZIONE NUOVA:

3 CASI:

* il leader non è parte della
configurazione nuova.

Il leader ritorna a stato

follower, di conseguenza

si indicano elezioni nella nuova
configurazione.

* Un nodo disconnesso nella nuova
configurazione non riceve messaggio
di vita da leader e indice elezioni
disturbando.

Se il leader (nuova conf) è attivo
e non ignora l'indetta di elezione
da nodo anomalo perché
timeout di liveness non è
ancora scaduto.

Lo stesso fanno i follower.

* C'è un nuovo nodo della nuova configurazione
voto. (stato locale errato).

Nodo completamente nuovo ma
con le entry non aggiornate.
compilato stabilizzare lo stato
(tanto tempo richiesto).

Nel mentre che il leader invia
lo stato a tutte le nodi possono
arrivare commit che modificano
il suddetto stato. (molto tempo
richiesto per aggiornare
nuovo nodo.

→ Nuovo nodo inserito
nel cluster come

membro non votante

fino a che non stabilizza
lo stato così da non
poterlo diventare leader

← solo per il
commit
dello stato

