Corso di Reti di Calcolatori aa 2000/01

Franco Morelli 52737

SERVIZIO DI NOMI PARTIZIONATO E REPLICATO

Descrizione

Lo scopo del progetto consiste nella realizzazione di un sistema di nomi che abbia caratteristiche di affidabilità e buona disponibilità grazie a meccanismi di replicazione, unite a una gestione partizionata delle informazioni.

Il sistema si prefigge di rispondere alle esigenze di servizio di un' area locale all'interno della quale sia necessario registrare e reperire i servizi che ciascun host mette a disposizione sulla rete. Il sistema deve perciò mantenere memoria dei riferimenti ai servizi disponibili sulla rete e mettere a disposizione tali riferimenti a chi ne faccia richiesta.
In particolare i riferimenti ai servizi sono costituiti dall' INDIRIZZO IP e dalla PORTA sui quali è disponibile ciascun servizio.
Il cliente che sia interessato a reperire l'indirizzo di un certo servizio, deve formulare una richiesta al sistema presentando il nome logico del servizio ricercato. Si suppone quindi che la chiave di ricerca per l'accesso alle informazioni del sistema sia il NOME LOGICO del servizio (che deve perciò essere unico).
Il sistema deve ovviamente poter rispondere anche a richieste di registrazione di nuovi nomi di servizi, (facendosi carico di mantenere l'unicità dei nomi) oltre che a cancellazione degli stessi.

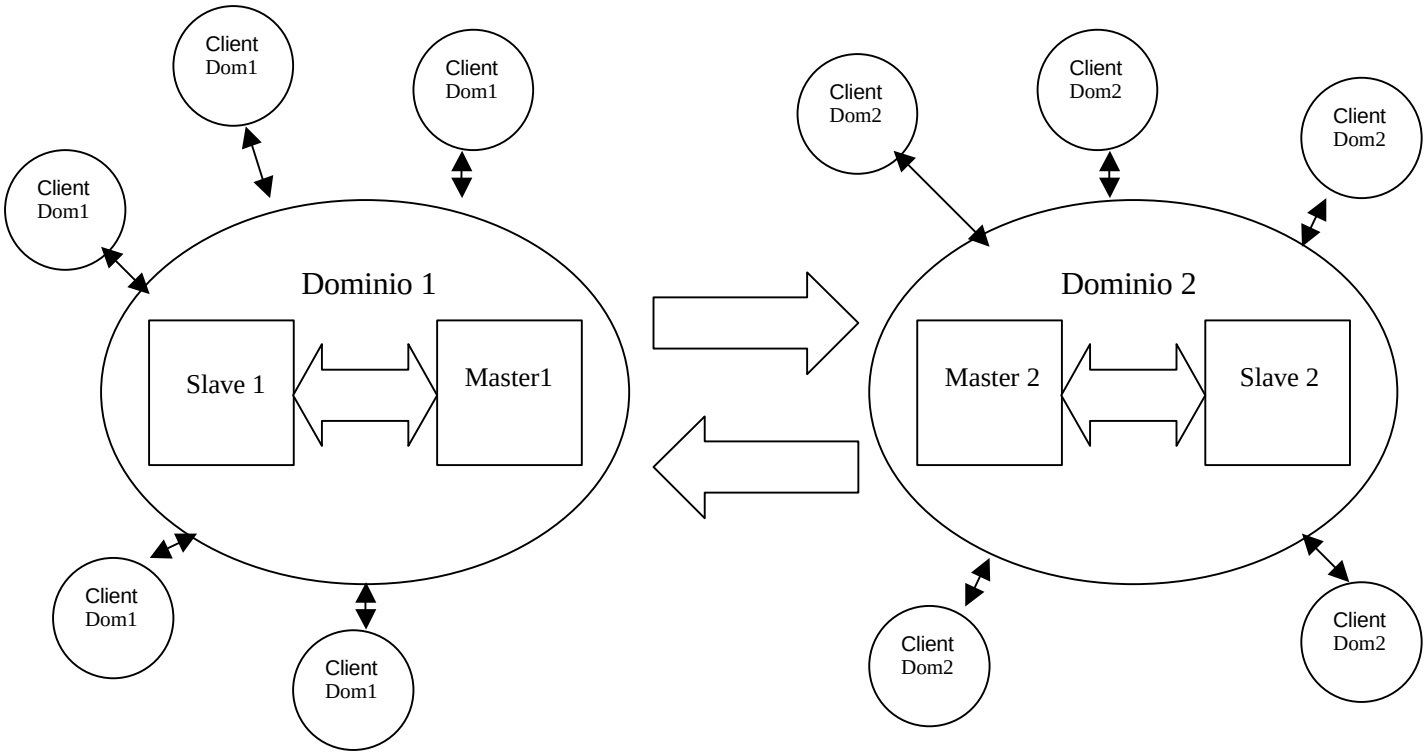
-Partizionamento

Si è fatta l'ipotesi che i clienti che usufruiscono del servizio siano collocabili in due domini e che quindi risulti più efficiente disporre di un punto di accesso al servizio per ciascun dominio.
Si suppone quindi che un cliente appartenente a un dominio faccia richieste al servizio prevalentemente per risolvere nomi appartenenti al proprio dominio e che il servitore del proprio dominio si rivolga al servitore dell'altro dominio quando non sia in grado di risolvere un nome(query ricorsiva).
La tabella dei riferimenti è quindi partizionata e ciascun servitore possiede la porzione di tabella relativa alla propria località.
E' compito del sistema coordinare il comportamento dei servitori delle due località e mantenere l'unicità dei nomi all'interno delle due tabelle ,facendo in modo che lo stesso nome non compaia in tabelle distinte(nomi globali).

-Replicazione

Per garantire un certo grado di affidabilità e disponibilità del servizio, il sistema è stato replicato in modo da poter rispondere alle richieste dei clienti anche in caso di guasto. Così a ciascuno dei

due servitori è stata affiancata una copia calda(ovviamente residente su una macchina distinta) in grado di farne le veci in caso di malfunzionamento.
In questo modo è garantita la possibilità ai clienti di continuare a fare inserimenti, cancellazioni o ricerche di nomi all'interno del registro anche se entrambi i servitori principali sono fuori uso.



Implementazione

Protocollo di comunicazione clienti-servitori

La comunicazione tra clienti e servitori prevede lo scambio di messaggi brevi.
Infatti il cliente può spedire ai server 3 tipi di richiesta: inserimento ,cancellazione o lookup di un nome. Perciò i messaggi contenenti tali richieste, contengono un comando identificativo del tipo di richiesta (di 3 lettere) e un nome logico di servizio.
Dall'altra parte le risposte dei servitori sono altrettanto brevi, infatti esse contengono il risultato delle richieste ,cioè un nome e un indirizzo IP+porta o messaggi di fallimento o successo.
Si capisce dunque perchè il protocollo di comunicazione scelto sia basato su **datagrammi UDP.**
Questa scelta connection-less è particolarmente adatta se si pensa anche all' alto numero di richieste che un servitore può dover soddisfare.
Tale protocollo è però unreliable, perciò è possibile la perdita di messaggi. Dunque le richieste che i clienti formulano ai servitori vengono ripetute in caso di non risposta.
Ho ipotizzato che siano sufficienti 2 ripetizioni di richiesta senza risposta, dopo le quali il cliente evince che il servitore al quale si è rivolto sia caduto.

In particolare, ogni richiesta di un cliente è fatta, in sequenza, due volte al servitore master e due volte al servitore slave.
Il tempo di attesa per la risposta è determinato da un time-out, il quale deve essere superiore al tempo massimo di soddisfacimento del servizio da parte dei servitori.

Il comportamento del cliente è il seguente:

```
send al Master
receive (con timeout) dal Master
se il timeout è scaduto allora {
  send al Master
  receive(con timeout) dal Master
  se il timeout è scaduto allora{
    send allo Slave
    receive(con timeout) dallo Slave
    se il timeout è scaduto allora{
      send allo Slave
      receive(con timeout) dallo Slave
      se il timeout è scaduto allora NESSUNA RISPOSTA
    }
  }
}
```

Da questo schema si nota che, in caso di caduta del Master, il cliente riceve comunque risposta dallo Slave(in modo trasparente), sebbene il tempo necessario a ricevere tale risposta sia superiore.
Nel caso peggiore il cliente è costretto ad attendere un tempo equivalente a 4 time-out.
Adottando questo protocollo, è possibile che la stessa richiesta (dallo stesso cliente) arrivi 2 volte a un servitore . Nel caso si tratti di un *lookup* ,non vi sono problemi di generazione di stati inconsistenti sulla tabella, grazie all'idempotenza della operazione.
Nel caso si tratti di un *inserimento o una cancellazione*, ugualmente non vi sono problemi di inconsistenza , in quanto lo stato prodotto dalle interazioni precedenti è contenuto implicitamente nella tabella(che contiene nomi unici).
L'erronea doppia richiesta formulata dal cliente genera però una doppia risposta da parte del servitore. Questo problema è stato risolto facendo 'buttare' al cliente una delle 2 risposte ricevute.

Protocollo di coordinamento dei 2 domini

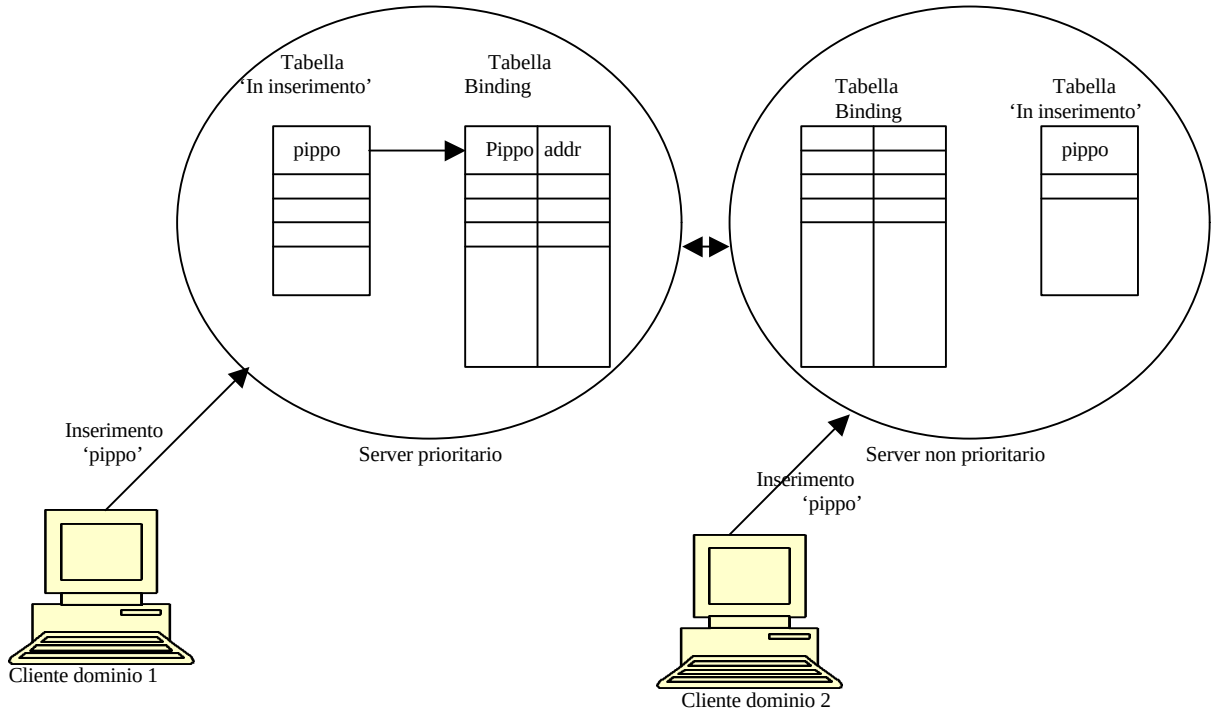
I servitori appartenenti alle due località devono coordinarsi per evitare la presenza di nomi doppi, inoltre devono comunicare per risolvere le interrogazioni dei clienti., infatti se uno dei servitori non conosce un nome richiestogli, esso gira la domanda al servitore dell'altro dominio e riporta il risultato ottenuto al cliente.
Ogni servitore ha perciò conoscenza della coppia di servitori (master e slave) dell'altro dominio.

-COMUNICAZIONE PER RISOLUZIONE NOMI
Come detto, un servitore che ignori un certo nome gira la domanda all'altro servitore.
Esso si rivolge all'altro servitore utilizzando lo stesso protocollo(visto nella sezione precedente) di un qualsiasi cliente. Ovviamente l'altro servitore (master o slave che sia) deve sapere che tale richiesta non proviene da un cliente qualunque, in modo da risolverla in modo non ricorsivo evitando così che i 2 domini se la rimbalzino all'infinito. Quindi, dal punto di vista del servitore,

le richieste provenienti dal server dell'altro dominio sono etichettate in modo diverso da quelle provenienti dai normali clienti.

-COMUNICAZIONE PER IL MANTENIMENTO DELLA GLOBALITA' DEI NOMI

Questo protocollo di coordinamento serve a *gestire l'inserimento di nuovi nomi* all'interno delle tabelle delle 2 località.
I servitori devono garantire la globalità dei nomi, occorre quindi che quando un server riceve una proposta di registrazione da un cliente, interroghi l'altro(facendo un lookup) per sapere se quello in via di inserimento è un nome già presente o meno. Anche in questo caso il protocollo utilizzato è identico a quello di un normale cliente, tranne che per l'etichetta del comando inviato.
Questa fase di coordinamento tra i server termina con l'accettazione o il rifiuto del nuovo inserimento.
Occorre però considerare anche il caso in cui due clienti richiedano contemporaneamente l'inserimento dello stesso nome di servizio sulle due diverse località.
In questo caso ciascun servitore ,non avendo ancora inserito il nome in tabella , richiederebbe all'altro il lookup sul nome in questione, ricevendo risposta negativa(perché effettivamente nessuno dei due ha ancora inserito tale nome). Il risultato sarebbe l'inserimento dello stesso nome nelle due tabelle, violando la globalità dei nomi.
Per risolvere questo problema, è stato imposto che ciascun servitore sia dotato di una tabella ausiliaria nella quale vengono memorizzati i nomi 'in fase di inserimento'.
In questa tabella sono presenti i nomi per i quali è stato richiesto l'inserimento ma che non sono ancora stati inseriti perché non ancora 'verificati'. Così nel caso di inserimento contemporaneo di due nomi uguali sulle 2 località, essi vengono inseriti entrambi nelle tabelle 'in fase di inserimento' dei due servitori, i quali possono così decidere quale dei 2 inserire in base a una priorità statica(assegnata ai servitori stessi in modo arbitrario): il server più prioritario registra il nome, l'altro rinuncia.



Protocollo di coordinamento tra servitore master e copia(slave)

Il servitore master di ciascuna località è replicato per mezzo di una *copia calda.* Ciò significa che le modifiche che avvengono sulla tabella del master, vengono immediatamente propagate allo slave. Tali modifiche (inserimenti e cancellazioni) vengono trasmesse per mezzo di datagrammi UDP, anche in questo caso ripetendo due volte la richiesta in caso di non risposta. Ovviamente i comandi di aggiornamento hanno etichette diverse rispetto a quelli di modifica spediti dai clienti.
Nel funzionamento normale la copia è sempre sincronizzata con il master, quindi può prenderne il posto in qualsiasi momento.

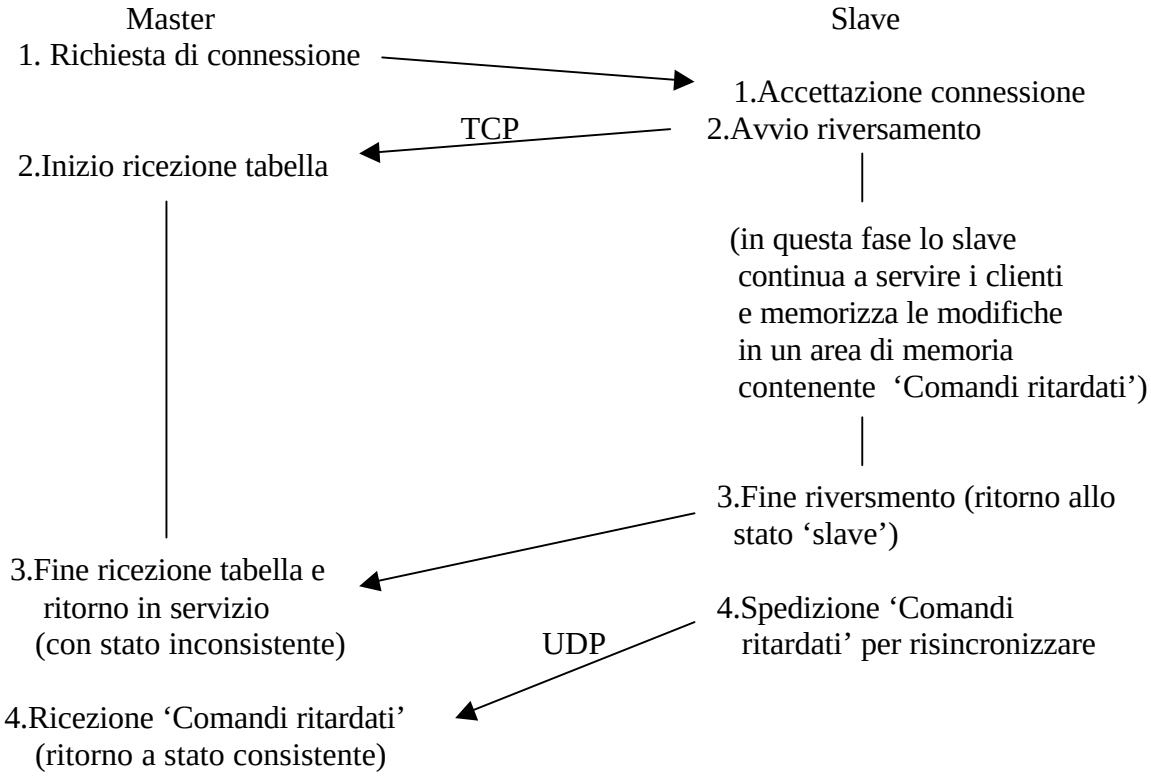
-CADUTA DEL SERVITORE MASTER

La caduta del master fa in modo che i clienti si rivolgano alla copia (grazie al protocollo clienti-servitori visto sopra).La copia normalmente rifiuta le richieste dei clienti (che potrebbero erroneamente arrivare anche quando il master è in servizio) però ogni volta che ne riceve una, si accerta che il master sia effettivamente caduto(check) e in caso di risposta affermativa comincia ad operare in modo perfettamente analogo al master (transita in 'stato master').
Occorre notare che durante il periodo in cui il master è giù, a causa del protocollo clienti-servitori adottato, i clienti ricevono risposta in un tempo leggermente superiore(ma comunque accettabile).

-RIPRESA DEL SERVITORE MASTER

Alla ripartenza il master tenta subito di aggiornarsi rivolgendosi alla copia per ricevere l'intera tabella (tale scelta è stata fatta ipotizzando che la tabella non sia di dimensioni enormi e sia perciò poco oneroso trasmetterla tutta). Questo aggiornamento avviene per mezzo di una connesione TCP, stabilita la quale il master comincia a ricevere la tabella (serializzata).
Durante la fase di riversamento, che potrebbe anche essere lunga, la copia continua a operare sulla tabella facendo nuovi inserimenti e cancellazioni, i quali però non possono essere subito propagati al master, che è impegnato nella ricezione della copia della tabella. Le modifiche che avvengono durante il riversamento vengono perciò applicate subito alla tabella dello slave ma vengono anche memorizzate in un area di memoria separata ('Comandi ritardati') per essere spedite successivamente(alla fine del riversamento) al master attraverso datagrammi UDP. In questo modo la tabella ricevuta dal master viene risincronizzata con quella in possesso dello slave.
Fino a quando non sia finita tale risincronizzazione (che si suppone breve) il master possiede una tabella inconsistente, potendo comunque già rispondere ai clienti. Dunque in questo breve periodo di inconsistenza le richieste di inserimento e cancellazione vengono rifiutate in modo che il cliente sia spinto a riformularle. Si suppone che alla riformulazione il cliente incontri uno stato consistente della tabella e possa perciò ricevere risposta positiva.

Di seguito vengono elencati i passi principali della procedura di restart:



-CADUTA DEL SERVITORE SLAVE

Per l'ipotesi di guasto singolo, la caduta del servitore slave può avvenire solo quando il master è in servizio. Il master si accorge della caduta dello slave semplicemente constatando che esso non risponde più ai comandi di aggiornamento. Dal punto di vista del cliente la caduta dello slave è ovviamente ininfluente.

-RIPRESA DEL SERVITORE SLAVE

La procedura di restart dello slave è analoga a quella del master.
Così lo slave richiede la connessione TCP al master e riceve la tabella corrente. Anche in questo caso la completa risincronizzazione avviene quando anche i 'Comandi ritardati' sono stati spediti.

STRUTTURA E FUNZIONAMENTO DEI SERVITORI

Funzionamento del server master

Il servitore primario è costituito da 2 processi principali:

- PROCESSO DI SERVIZIO AI CLIENTI
- PROCESSO DI AGGIORNAMENTO DELLA COPIA

PROCESSO DI SERVIZIO AI CLIENTI

Il processo di servizio ai clienti è un **servitore parallelo** che rimane in ascolto su una socket datagram in attesa di richieste di servizio. Il verificarsi di una richiesta scatena la generazione di un thread che si occupa della gestione del servizio.
Tali richieste di servizio possono provenire sia dai clienti che dall'altra coppia master-slave che dallo slave associato al master. In particolare i clienti possono fare richieste di INS, DEL o LOOKUP, mentre l'altra coppia master-slave può fare richieste di LOOKUP (per risolvere nomi ad essa ignoti o per coordinarsi in fase di inserimento di un nuovo nome, onde evitare nomi doppi) e la copia associata al master può fare richieste di INS o DEL in seguito alla fase di riversamento della tabella (avvenuta durante il restart del master), per spedire i 'Comandi ritardati'
Il comportamento del servitore varia in base alla provenienza di ciascuna delle richieste:

INS dal cliente — Il thread di servizio si occupa di controllare nella tabella locale la presenza o meno del nome del servizio che il cliente intende registrare.
Se il nome non esiste allora viene inserito in un area di memoria contenente i nomi in fase di inserimento e viene contattata l'altra coppia master-slave(chi dei 2 è attivo) chiedendo un LOOKUP del nome in questione. Se tale nome è presente nella tabella remota, allora il thread di servizio rinuncia a inserirlo in quella locale e notifica il cliente. Se il nome invece è presente tra i nomi in fase di inserimento remoti, allora il thread rinuncia ad inserirlo solo se la coppia master-slave remota è più prioritaria (priorità statica determinata per dirimere le situazioni di inserimento contemporaneo di 2 nomi uguali).
Se, una volta terminata la fase di coordinamento con l'altra coppia, è possibile inserire il nuovo nome nella tabella locale, il thread si comporta in modo differente in base allo stato dell'interazione con lo slave locale. Infatti se lo slave locale è in fase di restart dopo una caduta, allora la tabella sta essendo copiata dal master allo slave e lo slave non è in grado di accettare aggiornamenti. Gli inserimenti che dovrebbero avvenire in questa fase vengono perciò ritardati memorizzando in una area di memoria('Comandi ritardati') stringhe che rappresentino le operazioni da fare successivamente alla fase di restart (es. INS-Pippo-123.345.12.1-2341/ INS-Pluto-145.23.4.32-12313). Tali stringhe di comando verranno poi spedite allo slave dopo il restart, ad opera del processo che si è occupato del riversamento della tabella.
Se invece non si è nella fase di restart dello slave, allora il thread aggiorna la tabella locale, notifica il cliente dell'avvenuto inserimento e aggiorna lo slave per mezzo di un comando di INS.

DEL dal cliente — In questo caso il thread di servizio elimina la entry dalla tabella locale e dalla copia calda. Se la entry non esiste, non viene interrogata l'altra coppia perché si suppone che i clienti di una certa località possano cancellare solo nomi relativi alla propria località.
Come per il caso di INS, se è in atto il restart della copia, la cancellazione effettiva viene ritardata memorizzando una riga di comando da spedire successivamente allo slave (es. DEL Pluto).

LOOKUP dal cliente — Il thread di servizio cerca la entry nella tabella locale, se non la trova invia un pacchetto di lookup all'altra coppia master-slave, la quale cercherà a sua volta nella propria tabella e riporterà un risultato che verrà poi girato al cliente. (Query ricorsiva).

INS dallo slave — Questo tipo di inserimento può avvenire successivamente alla copia della tabella dallo slave al master dopo che esso è ripartito, per eseguire uno dei 'Comandi ritardati' necessari alla completa risincronizzazione.
In questo caso il thread di servizio inserisce la entry in tabella senza interrogare l'altra coppia in quanto i controlli di unicità sono già stati fatti dallo slave e non è possibile fare nuovi inserimenti fino a che non è stato raggiunto uno stato consistente.
Tali comandi di inserimento provenienti dallo slave servono a risincronizzare la tabella del master, perciò si ammette che, fino a quando non siano terminati, il master si trovi in uno stato incosistente.

DEL dallo slave — Anche in questo caso il comando è successivo alla precopia e serve a sincronizzare completamente la tabella del master.

LOOKUP dall'altra coppia Master-Slave — Il thread di servizio controlla la presenza della entry richiesta nella tabella locale e nella zona dei nomi in fase di inserimento, fornendo una risposta diversa nel caso essa si trovi nell'una o nell'altra.

Altri comandi ricevibili dal servitore master, rappresentano mezzi di sincronizzazione con la copia:

CHECK dallo slave — Questo comando viene ricevuto dal master quando lo slave intende accertarsi che sia caduto.

'FINE' dallo slave — Questo comando serve per informare il master della terminazione della risincronizzazione. Il master capisce che la propria tabella è tornata in uno stato consistente (dopo il restart).

PROCESSO DI AGGIORNAMENTO DELLA COPIA

Questo processo rimane sempre in ascolto su una socket stream attendendo richieste di connesione da parte dello slave.
Lo slave richiede una connessione all'atto del restart, quando ha bisogno che il master gli passi la tabella dei riferimenti aggiornata.
Quando sia stabilita la connessione, viene avviato il riversamento della tabella in modo concorrente al normale processo di servizio del master (viene settata una variabile di stato globale che indichi l'avvio di questa fase, per informare il processo di servizio). Come già notato, durante questa fase gli eventuali nuovi inserimenti e cancellazioni vengono rimandati.
Quando sia terminato il riversamento (precopia), il processo di aggiornamento si occupa di effettuare gli aggiornamenti tenuti in sospeso, andando a leggere nella zona di memoria('Comandi ritardati') in cui sono stati memorizzati dal processo di servizio. In particolare questi comandi 'ritardati' vengono spediti allo slave come pacchetti datagram. Questo processo apre perciò anche una socket datagram.

Funzionamento del server slave

Lo slave funziona come copia calda, perciò è mantenuto costantemente aggiornato dal master.
Lo slave può fare le veci del master in caso esso cessi di funzionare. Perciò in ogni istante lo slave può trovarsi o in 'stato slave' o in 'stato master'. Lo stato determina il comportamento dello slave.
Anche questo servitore è costituito da 2 processi principali:

- PROCESSO DI SERVIZIO AI CLIENTI
- PROCESSO DI AGGIORNAMENTO DEL MASTER

PROCESSO DI SERVIZIO AI CLIENTI

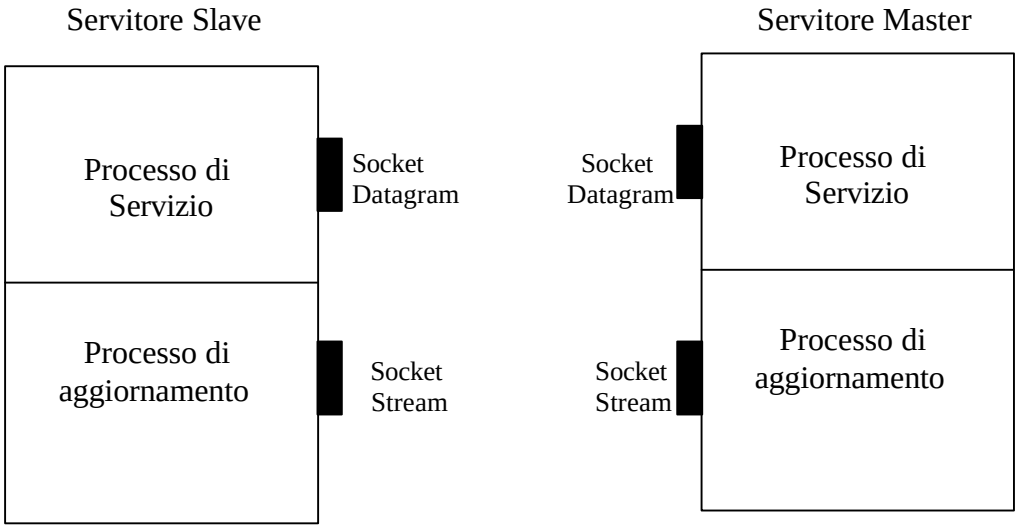
Il processo di servizio funziona in modo analogo a quello del master, in particolare in **'stato master'** accoglie richieste da chiunque mentre in **'stato slave'** accoglie le richieste provenienti dal master per gli aggiornamenti.
Il transito dallo 'stato slave' allo 'stato master' avviene non appena viene ricevuta una richiesta che non provenga dal master. Il verificarsi di questo evento può significare la caduta del master(infatti ogni richiesta formulata alla coppia master-slave è composta di 2 tentativi verso il master e successivi 2 rivolti allo slave) quindi per accertarsene lo slave interroga il master (Check) e se non riceve risposta transita in 'stato master'.
Il transito da 'stato master' a 'stato slave' avviene invece alla fine della fase di restart del master, quando esso può riprendere le proprie funzioni.

INS dal cliente	Se lo stato è 'master' il comportamento è analogo a quello del master. In questo caso però all'atto dell'inserimento (in tabella o nella zona dei comandi 'ritardati') viene controllato anche che sia avvenuta la transizione in stato 'slave', in tal caso l'inserimento viene abortito per evitare inconsistenze (infatti se lo stato è 'slave' allora gli inserimenti devono essere fatti sul master, perché in tale stato gli INS fatti sullo slave non si propagano al master). Se il comando di INS dal cliente perviene quando lo stato è 'slave', allora occorre che il processo che gestisce questa richiesta si accerti che il master sia effettivamente caduto (con una richiesta di LOOKUP). Se il master non risponde allora avviene il transito in stato 'master', diversamente la richiesta viene fatta cadere.
DEL dal cliente	Anche in questo caso, se lo stato è 'master', il comportamento è analogo a quello del master. Se la transizione allo stato 'slave' avviene durante il servizio, tale richiesta viene abortita (in quanto non sarebbe propagata al master). Se la richiesta di DEL perviene quando lo stato è già 'slave', il funzionamento è simile a quanto visto sopra per la richiesta di INS (transizione di stato).
LOOKUP dal cliente	Se lo stato è 'master', il processo di servizio è uguale a quello visto per il server master. Se la transizione allo stato 'slave' avviene durante tale processo, la richiesta viene comunque soddisfatta (perché idempotente). Se lo stato è 'slave', tale richiesta può provocare la transizione di stato.
INS dal Master	Lo slave riceve questo comando per rimanere sincronizzato col master, il comando viene perciò eseguito senza fare controlli.
DEL dal Master	Come sopra.
LOOKUP dall'altra coppia Master-Slave	Se lo stato è 'master', il thread di servizio controlla la presenza della entry richiesta nella tabella locale e nella zona dei nomi in fase di inserimento,

fornendo una risposta diversa nel caso essa si trovi nell'una o nell'altra. Se lo stato è 'slave' la richiesta viene soddisfatta solo dopo l'accertamento della effettiva caduta del master.

PROCESSO DI AGGIORNAMENTO DEL MASTER

Questo processo è analogo al corrispondente processo presente sulla macchina master.
E' questo processo che si occupa di far transistare lo slave dallo stato 'master' allo stato 'slave' dopo avere finito di aggiornare il master. Se per qualche motivo il restart del master dovesse fallire, si ipotizza che ciò sia dovuto a una nuova caduta del master stesso, perciò lo stato del servitore slave viene mantenuto 'master'.



STRUTTURA E FUNZIONAMENTO DEI CLIENTI

I clienti funzionano come applicazioni 'stand alone' e come già detto, ciascun cliente conosce solo il servitore della propria località. Si suppone che ciascuna macchina che usufruisce del servizio abbia un programma cliente installato e settato in modo corretto.
Come già osservato, un cliente può fare lookup, inserimenti e cancellazioni attraverso un protocollo connection-less.
In particolare per gli inserimenti esso deve specificare il *nome* e la *porta* del servizio che intende registrare, per quanto riguarda l'*indirizzo IP*, sarà il servitore a estrarlo dal datargamma e a memorizzarlo insieme alle altre informazioni inviategli.
Per ciò che riguarda le cancellazioni, esse sono possibili solo sul servitore della propria località, in quanto i clienti di una certa località possono effettuare cancellazioni solo di nomi del proprio dominio.
Relativamente a malfunzionamenti del processo cliente, essi non hanno effetti sul servitore in quanto l'interazione avviene senza connessione e senza stato e i servizi richiesti da un cliente vengono comunque portati a termine anche se esso cade.

Prove di funzionamento del sistema

Per effettuare i test di funzionamento, ognuno dei 2 servitori master è stato dotato della possibilità di creare, all'atto del primo avvio, una tabella di prova in memoria centrale di dimensione settabile a piacimento e contenente nomi di fantasia. In questo modo è stato possibile provare l'applicazione con diverse dimensioni della tabella.
Sono stati provate diverse situazioni di caduta e ripresa dei servitori e mi è sembrato di notare un buon comportamento del sistema, sebbene nelle fasi di ripresa sia capitato di non ricevere risposta in qualche occasione.
Per quanto riguarda i timeout, mi è sembrato che il limite inferiore per la comunicazione tra servitori sia 100 millisecondi, di conseguenza per i clienti sia circa 400.
Ho inoltre provato il sistema simulando la presenza di un centinaio di clienti in contemporanea per ciascuna località, ognuno dei quali facente richieste di lookup o inserimento ogni 6 secondi.
Ho notato che i clienti vengono serviti regolarmente, a parte un po' di non risposte nelle fasi di ripresa, e la sincronizzazione e il partizionamento delle tabelle sembra essere conservato(cioè non ci sono nomi doppi, sebbene abbia imposto ai clienti di fare molte richieste di inserimento di nomi uguali).