



Gestione dei nomi

Anno accademico 2007/8
Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it



Corso di Laurea Specialistica in Informatica, Università di Padova 1/36

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 1

- Le entità di un sistema distribuito devono avere denotazioni che le rendano fruibili
 - Per riferimento, invocazione di servizio, etc.
- Un nome è una stringa usata per riferire entità
- I punti d'accesso (*access point*) sono entità speciali il cui nome è un indirizzo
- Una stessa entità può avere più punti di accesso
 - Più "punti di vista" simultanei o anche diversi nel tempo

Corso di Laurea Specialistica in Informatica, Università di Padova 2/36

 Sistemi distribuiti: gestione dei nomi

Esempio

- Persona = entità
 - Un servizio
- Il suo telefono = l'*access point* dell'entità
 - Il *server* che fornisce quel servizio
- Il numero di quel telefono = l'indirizzo (nome) dell'*access point* di quella entità
 - L'indirizzo di quel *server* è il suo *end point* di livello trasporto

Corso di Laurea Specialistica in Informatica, Università di Padova 3/36

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 2

- Trattare gli indirizzi come nomi speciali
 - Consente di decomporre la corrispondenza <entità-indirizzo> in 2 relazioni legate ma distinte
 - Nome di entità → nomi degli attributi dell'entità (tra cui punto di accesso)
 - Punto di accesso → indirizzo dell'entità
 - Un'entità può cambiare punto di accesso
 - Un punto di accesso può essere riassegnato
 - Una stessa entità può avere più punti di accesso
- Vogliamo nomi di entità indipendenti dai loro indirizzi (*location independence*)



Corso di Laurea Specialistica in Informatica, Università di Padova 4/36

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 3

- È utile che certi nomi possano essere designare entità in modo univoco
 - Per questo usiamo identificatori
 - Non tutti i nomi sono identificatori
 - Alcuni nomi devono essere "trattabili" (*human-friendly*) ma non così gli indirizzi e neppure gli identificatori
- Un vero identificatore
 - Denota al più una singola entità
 - Non è riusabile
 - Una stessa entità non può averne più di uno
 - Così che, con essi, identificatori diversi designano entità diverse
 - Un numero di telefono fisso non un è vero identificatore di una entità "persona" perché può essere riassegnato!
- Nomi diversi per una stessa entità sono detti *alias*

Corso di Laurea Specialistica in Informatica, Università di Padova 5/36

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 4

- Lo spazio dei nomi (*name space*) 1/2
 - Grafo diretto etichettato
 - Gli archi sono etichettati con un nome
 - I nodi sono entità di sistema distribuito e hanno un identificatore
 - **Nodo repertorio** (*directory*) → da cui dipartono archi etichettati con un nome, contiene una tabella di corrispondenze <etichetta d'arco, identificatore di nodo>
 - **Nodo terminale** (*foglia*) → da cui non diparte alcun arco, contiene informazioni sull'entità che esso rappresenta (p.es. l'indirizzo, lo stato)
 - Il nodo con solo archi in uscita è detto **nodo radice** di *name space*
 - Uno spazio dei nomi può ammettere più nodi radice
 - Ogni nodo è un'entità del sistema distribuito ed è associato a un identificatore

Corso di Laurea Specialistica in Informatica, Università di Padova 6/36

Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 11

- **Name resolver**
 - L'entità localmente responsabile per la risoluzione dei nomi
- **Name server**
 - L'entità che gestisce i nomi del proprio repertorio
- **2 tecniche di risoluzione dei nomi**
 - **Iterativa** → ogni *name server* interrogato dal *resolver* fornisce la migliore risposta in suo possesso, senza fare ricerca
 - Il *resolver* determina il nome (cammino) completo dell'entità desiderata e lo fornisce al *name server* radice
 - Il cliente svolge più lavoro del server → non conviene fare *caching* presso il cliente
 - **Ricorsiva** → ogni *name server* interrogato interroga a sua volta i *name server* di livello inferiore per costruire la risposta richiesta
 - Il *name server* radice fornisce al *resolver* il nome (cammino) completo dell'entità richiesta
 - Il server svolge più lavoro del cliente → conviene fare *caching* presso il cliente

Corso di Laurea Specialistica in Informatica, Università di Padova 13/36

Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 12

- **Servizi di repertorio (*directory services*)**
 - Realizzano sistemi di denominazione basati su attributi piuttosto che su nomi strutturati (come il DNS)
 - Dalla combinazione delle due tecniche ha origine LDAP (*lightweight directory access protocol*)
 - Derivante da OSI X.500 con ampi miglioramenti prestazionali
 - Ogni istanza di servizio LDAP gestisce una DIB (*directory information base*) i cui campi hanno attributi dal nome unico
 - L'architettura complessiva LDAP risulta molto simile a quella del DNS però anche più sofisticata e potente

Corso di Laurea Specialistica in Informatica, Università di Padova 14/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 1

- **I nomi dei livelli globale e amministrativo cambiano di rado**
 - Il contenuto dei nodi dei loro grafi è stabile
 - Il *caching* delle relative risoluzioni è conveniente
- **I nomi del livello gestionale cambiano più frequentemente**
 - *Caching* non conveniente
 - Occorre minimizzare i costi di aggiornamento (del contenuto dei nodi) e di *look-up*

Corso di Laurea Specialistica in Informatica, Università di Padova 15/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 2

- **Cambiare i nomi usati come identificatori**
 - Invalida i *symbolic link* che li utilizzano
- **Modificare lo spazio dei nomi del nodo repertorio di origine**
 - Associando un nuovo indirizzo al vecchio nome
 - *Look-up* veloce
 - Ogni successivo aggiornamento (su nodo di origine) ha costo elevato
 - Associando un nuovo nome al valore del vecchio nome
 - = *symbolic link*
 - *Look-up* più lento
 - Facile aggiornamento tramite nuovo *symbolic link* locale al costo di una ulteriore indirizzazione

Corso di Laurea Specialistica in Informatica, Università di Padova 16/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 3

- Entrambe le soluzioni sono inadeguate per sistemi distribuiti a larga scala
- Serve una tecnica che separi i nomi dagli indirizzi
 - Non come nel DNS di *Internet*
- Gli identificatori sono adatti a questo scopo
 - Da nome utente a identificatore → *naming service*
 - Da identificatore a indirizzo → *location service*

Corso di Laurea Specialistica in Informatica, Università di Padova 17/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 4

Corrispondenza **diretta** (1 livello) tra nomi e indirizzi → come nel DNS di *Internet*

Corrispondenza **indiretta** (2 livelli) con uso di identificatori come tramite di risoluzione

Corso di Laurea Specialistica in Informatica, Università di Padova 18/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 5

- Un *location service* accetta un identificatore di entità e ne restituisce l'indirizzo corrente
 - Un indirizzo per ogni copia (o *access point*) dell'entità
- Soluzione brutale: *broadcast*
 - ARP (*Address Resolution Protocol*) restituisce l'indirizzo di livello collegamento dati (rete locale) corrispondente ad un indirizzo IP in ingresso
 - Troppo oneroso per reti vaste
 - Meglio *multicast* verso gruppi specifici (anche dinamici) di nodi coinvolti

Corso di Laurea Specialistica in Informatica, Università di Padova 19/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 6

- Soluzione meno brutale: *forwarding*
 - Simile a una catena di *symbolic link* da ogni locazione precedente verso la successiva
 - 3 importanti difetti
 - Il costo di localizzazione di un'entità può diventare proibitivo
 - Tutte le "vecchie" locazioni devono conservare informazione relativa all'entità
 - Basta un collegamento interrotto o corrotto per rendere l'entità irraggiungibile
 - Realizzato da una catena di SSP, coppie *<proxy, skeleton>*, in un sistema a oggetti distribuiti
 - *Proxy (stub)* → riferimento allo *skeleton* finale o intermedio
 - *Skeleton (scion)* → riferimento locale all'oggetto oppure al suo *proxy* locale
 - *Scion* = discendente in linea ereditaria

Corso di Laurea Specialistica in Informatica, Università di Padova 20/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 7

Corso di Laurea Specialistica in Informatica, Università di Padova 21/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 8

- Il sistema SSP si presta a 2 ottimizzazioni
 - La richiesta include l'identificatore del *proxy* iniziale
 - La risposta include l'indirizzo attuale dell'oggetto
 - Questo consente di creare un cortocircuito tra cliente e oggetto remoto

Corso di Laurea Specialistica in Informatica, Università di Padova 22/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 9

- Soluzione migliore: *home location*
 - Tecnica utilizzata per supportare indirizzi IP mobili
 - Ogni utente mobile ha un IP fisso al quale corrisponde un *home agent*
 - Quando l'utente si sposta su un'altra rete riceve un nuovo indirizzo temporaneo (*care-of*) che viene registrato presso l'*home agent*
 - L'*home agent* gira ogni pacchetto indirizzato all'utente verso il suo indirizzo *care-of* e ne informa il mittente
 - Grande trasparenza al costo di un contatto obbligatorio con la *home location* del destinatario

Corso di Laurea Specialistica in Informatica, Università di Padova 23/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 10

- Soluzione ad approccio gerarchico 1/4
 - La rete di interconnessione è suddivisa in una collezione di domini (similmente all'organizzazione del DNS)
 - Un singolo dominio copre l'intera rete
 - Ogni dominio può essere decomposto in sotto-domini
 - Un dominio non decomposto (terminale) è detto *foglia* e corrisponde a un ben definito aggregato (p.es. una rete locale, una cella)
 - Ogni dominio ha un nodo repertorio che conosce tutte le entità presenti in esso
 - Il nodo repertorio del dominio di livello più alto è detto *nodo radice* e conosce tutte le entità dell'intero sistema

Corso di Laurea Specialistica in Informatica, Università di Padova 24/36

Sistemi distribuiti: gestione dei nomi

Sistemi distribuiti: gestione dei nomi

Entità mobili – 11

□ Soluzione ad approccio gerarchico 2/4

- Ogni entità E presente in un dominio D è rappresentata da una voce (*location record*) nel nodo repertorio N di D
 - La voce di E nel dominio foglia di appartenenza contiene l'indirizzo di E in D
 - Alla voce E nel dominio di livello immediatamente superiore a D (che contiene D) corrisponde solo un riferimento puntatore a N
- Il nodo radice conterrà una voce (*name record*) per ogni entità del sistema, la quale conterrà l'inizio di una catena di puntatori a nodi repertori fino a quello una cui voce (*location record*) contiene l'indirizzo dell'entità

Corso di Laurea Specialistica in Informatica, Università di Padova 25/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 12

□ Soluzione ad approccio gerarchico 3/4

- Il *look-up* di E ha inizio nel nodo repertorio del dominio D_0 di residenza del cliente che lo richiede
 - Se D_0 non contiene una voce per E allora E non si trova in D_0
 - In quel caso la richiesta viene inviata al nodo repertorio del dominio D_1 "genitore" di D_0 , che per definizione è più ampio di D_0 (e quindi conosce più entità di D_0) e così via fino a che una voce per E venga trovata nel nodo repertorio del dominio D_N
 - Ciò significa che E si trova in D_N dove verrà localizzato seguendo a ritroso la catena di riferimento, fino all'indirizzo di E nel suo dominio foglia
- In questo modo il *look-up* sfrutta la località dei riferimenti
 - La fase di "risalita" della ricerca avviene in domini concentricamente più ampi
 - Nel caso peggiore fino al nodo radice, per poi da lì ridiscendere

Corso di Laurea Specialistica in Informatica, Università di Padova 26/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 13

Node has no record for E, so that request is forwarded to parent

Node knows about E, so request is forwarded to child

Look-up request

Domain D

La modalità di *look-up* concentrica, ascendente e discendente

Corso di Laurea Specialistica in Informatica, Università di Padova 27/36

Sistemi distribuiti: gestione dei nomi

Entità mobili – 14

□ Soluzione ad approccio gerarchico 4/4

- Concentrare tutte le voci nel nodo radice pone gravi problemi di scalabilità
 - Rendendolo collo di bottiglia del sistema
- Convienne partizionarne la conoscenza ma il problema diventa decidere dove localizzarne le parti
 - L'uso di calcolo parallelo o distribuito trasferisce il collo di bottiglia dall'elaborazione alla comunicazione
 - Meglio usare più (sotto-)nodi sparsi uniformemente in domini dell'intera rete
 - In questo caso il problema diventa la determinazione dei cammini minimi per localizzare il (sotto-)nodo responsabile dell'informazione richiesta

Corso di Laurea Specialistica in Informatica, Università di Padova 28/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 1

□ Un'entità che non possa (più) essere riferita è inutile al sistema e dovrebbe essere rimossa

- *Garbage collection*

□ La rimozione può (talvolta) essere esplicita

- "L'ultimo" processo a usare una data entità può rimuoverla
- Come individuare "l'ultimo" processo in un sistema distribuito?

Corso di Laurea Specialistica in Informatica, Università di Padova 29/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 2

□ L'insieme dei riferimenti tra oggetti è un grafo diretto nel quale gli oggetti sono nodi e gli archi sono riferimenti

□ Uno specifico sottoinsieme di nodi non riferiti ma capaci di riferire è detto *root set*

- Esempio: servizi di sistema, utenti

□ Gli oggetti non riferiti direttamente o indirettamente dal *root set* vanno rimossi

Corso di Laurea Specialistica in Informatica, Università di Padova 30/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 3

Entità radice, raggiungibili e non raggiungibili

Unreachable entity from the root set

Corso di Laurea Specialistica in Informatica, Università di Padova 31/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 4

- Tecniche di rimozione: contatore dei riferimenti
 - Funziona bene in sistemi monoprocesore ma è ostacolata dai possibili errori di comunicazione nei sistemi distribuiti
 - L'oggetto distribuito mantiene il proprio contatore dei riferimenti nel suo *skeleton*
 - Ogni *proxy* creato presso nodi utente invia una notifica di incremento allo *skeleton* che ne conferma la ricezione
 - La mancata ricezione di conferma (*acknowledgement*) non implica il mancato incremento
 - Rischio di duplicazione
 - Il passaggio di un riferimento remoto da un oggetto all'altro deve causare un incremento
 - Ogni rimozione di *proxy* comporta l'invio di una notifica di decremento allo *skeleton* che può però andare perduta
 - L'oggetto potrebbe restare in vita anche se privo di utenti

Corso di Laurea Specialistica in Informatica, Università di Padova 32/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 5

- Altro grave problema di questa tecnica è causato dalla *race condition* tra incrementi e decrementi
- Il problema sparisce usando solo decrementi
 - Alla creazione dell'oggetto remoto, al suo *skeleton* vengono assegnati un peso totale (RC = 2^n) e un peso parziale (WR = 2^{n-1})
 - Inizialmente $n=m$
 - A ogni creazione di un riferimento, $1/2$ WR dello *skeleton* viene ceduto al *proxy* corrispondente, con l'invariante $RC = \sum (WR)$
 - A ogni passaggio di riferimento, il *proxy* emittente cede $1/2$ del suo peso al destinatario
 - A ogni rimozione di un riferimento il peso del corrispondente *proxy* viene sottratto a RC dello *skeleton*
 - Quando RC dello *skeleton* va a 0 l'oggetto viene rimosso

Corso di Laurea Specialistica in Informatica, Università di Padova 33/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 6

Alla creazione dell'oggetto remoto: $RC = 2^n$; $WR(S) = RC$

Alla creazione del primo riferimento: $WR(P) = WR(S)/2$; $WR(S) = WR(S)/2$; $RC(S) = \sum WR$

Al passaggio di riferimento: $WR(P_{i2}) = WR(P_{i1})/2$; $WR(P_{i1}) = WR(P_{i1})/2$; $RC(S) = \sum WR$

Corso di Laurea Specialistica in Informatica, Università di Padova 34/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 7

- Il problema è ora che ogni oggetto remoto consente solo N riferimenti per N fissato
 - Quando WR di *skeleton* e/o di *proxy* non è più dimezzabile (restituendo un intero) non sono più consentiti riferimenti
- Risolto tramite variante del *forwarding*
 - Una cella di indrezione (IC) rileva WR=1 ma riceve RC= 2^n , che ripartisce tra i nuovi riferimenti come se fosse un nuovo oggetto (esempio $n = 3$)

Corso di Laurea Specialistica in Informatica, Università di Padova 35/36

Sistemi distribuiti: gestione dei nomi

Entità non (più) riferibili – 8

- Lista dei riferimenti
 - Usata da Java RMI per GC distribuito
 - Un processo che crea o riceve un riferimento remoto deve prima identificarsi presso lo *skeleton*
 - Ricevuta conferma (*ack*) crea il *proxy*
 - Lo *skeleton* dell'oggetto remoto mantiene la lista dei riferimenti
 - Riferimento remoto considerato in "affitto" (*leasing*)
 - Rimosso automaticamente in assenza d'uso
 - Prima che il GC locale rimuova un *proxy* ne comunica la cancellazione allo *skeleton* corrispondente
 - Gestione dei riferimenti nello *skeleton* realizzate da RPC con semantica *at-most-once*
- Tecnica più informativa del semplice *reference count* che invece è anonimo
 - Rischio di *race condition* tra inserzioni e rimozioni di riferimenti

Corso di Laurea Specialistica in Informatica, Università di Padova 36/36