



Sistemi distribuiti: gestione dei nomi

Gestione dei nomi



Anno accademico 2014/15
Sistemi Concorrenti e Distribuiti

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

Laurea Magistrale in Informatica, Università di Padova 1/45



Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 2

- ❑ I punti d'accesso (*access point*) sono entità speciali il cui nome è un indirizzo
 - Un nome di entità che non varia con l'indirizzo è detto *location-independent*
 - La *location independence* è una proprietà desiderabile
- ❑ Una medesima entità può offrire più punti di accesso
 - Più "punti di vista" simultanei o anche in momenti diversi
 - Necessario se vogliamo distribuzione orizzontale!

Laurea Magistrale in Informatica, Università di Padova 3/45



Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 1

- ❑ Le entità di un sistema distribuito devono avere denotazioni che le rendano note
 - Per riferimento, invocazione di servizio, ecc.
 - Nel *middleware* di un sistema distribuito vi sono molte entità che vanno denotate
- ❑ Un nome è una stringa che riferisce entità

Laurea Magistrale in Informatica, Università di Padova 2/45



Sistemi distribuiti: gestione dei nomi

Esempio

- ❑ Persona = entità di sistema
 - Il suo ruolo pubblico è offrire servizi
- ❑ Telefono = *access point* di quei servizi
 - Ciò che va contattato per ottenere un servizio
 - Quindi ne è il *server* che è una entità di livello *middleware*
- ❑ Numero di telefono = nome (indirizzo) dell'*access point* di quel servizio
 - L'indirizzo del *server* è il suo *end point* di livello trasporto

Laurea Magistrale in Informatica, Università di Padova 4/45

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 3

Gli indirizzi sono nomi speciali

- Conviene decomporre la corrispondenza <entità-indirizzo> in due relazioni 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*

Laurea Magistrale in Informatica, Università di Padova 5/45

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 4

Gli identificatori sono nomi che designano entità in modo univoco

- Non tutti i nomi sono identificatori**
- Alcuni nomi devono essere *human-friendly* ma per esserlo non possono essere indirizzi e identificatori**

Laurea Magistrale in Informatica, Università di Padova 7/45

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 4

Il «sistema di *naming*» si occupa del *name-to-address binding*

- La sua forma più semplice come tabella di coppie <nome, indirizzo> non scala bene in distribuito**
- In genere allora il nome viene strutturato in parti e la risoluzione (*binding*) viene effettuata in modo ricorsivo**
 - Nessun elemento del sistema ha tutta l'informazione necessaria
 - Ma esiste una struttura logica di ripartizione della conoscenza
- Per esempio l'indirizzo del *server* di nome `smtp.math.unipd.it` viene determinato tramite richiesta ricorsiva a partire da `.it`**

Laurea Magistrale in Informatica, Università di Padova 6/45

 Sistemi distribuiti: gestione dei nomi

Denominazione di entità – 5

Un vero identificatore

- Denota al più una singola entità**
- Non è riusabile**
- Una stessa entità non può averne più di uno**
 - Identificatori diversi designano entità diverse
 - Un numero di telefono fisso non è un vero identificatore di una entità "persona" perché può essere riassegnato!

Nomi diversi per una stessa entità sono detti *alias*

Laurea Magistrale in Informatica, Università di Padova 8/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 1

❑ **Name space come grafo diretto etichettato**

- **Gli archi sono etichettati con un nome**
 - I nomi servono per raggiungere informazione
- **I nodi sono entità e hanno un identificatore**
 - Le entità erogano servizi (inclusi quelli informativi)
- **Nodo directory**
 - Ne dipartono archi etichettati con un nome e contiene una tabella di *binding* <etichetta, identificatore>
 - **Il nodo con solo archi in uscita è detto «radice del name space»**
 - Un *name space* può ammettere più nodi radice
- **Nodo foglia**
 - Non ne diparte alcun arco e contiene informazioni sull'entità che esso rappresenta (p.es. l'indirizzo, lo stato)

Laurea Magistrale in Informatica, Università di Padova

9/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 3

❑ **Risoluzione dei nomi**

- **Closure determina il nodo dal quale la risoluzione ha inizio**
 - **Esempio:** nel grafo dei nomi di un *file system* in UNIX il nodo radice è sempre il primo *i-node* della partizione che rappresenta il *file system*
- **Look-up restituisce l'identificatore del nodo da cui proseguire la risoluzione**
 - **Esempio:** nel grafo dei nomi di un *file system* UNIX l'identificatore di nodo è l'indice di un particolare *i-node* che ci fornisce un indirizzo su disco dove si trova il *name space* da cui proseguire la risoluzione
- **Alias**
 - **Hard link:** un cammino assoluto che denota una entità (nodo)
 - **Symbolic link:** un attributo del nodo (entità) denota un cammino assoluto

Laurea Magistrale in Informatica, Università di Padova

11/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 2

❑ **Ogni cammino sul grafo è un *path name* denotato dalle etichette degli archi percorsi**

- Cammino assoluto se l'origine è la radice del grafo
- Cammino relativo altrimenti

❑ **Un nome è sempre definito in relazione a un nodo *directory***

- Un nome globale viene interpretato sempre rispetto alla stessa *directory*
- Per un nome locale la *directory* di risoluzione è variabile

❑ **I grafi dei nomi sono spesso aciclici ma non tutti**

Laurea Magistrale in Informatica, Università di Padova

10/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 4

Grafo dei nomi con nodo radice unico e di nome implicito

Contenuto del nodo (directory) n1

- n2: "elke"
- n3: "max"
- n4: "steen"

Leaf node ○

Directory node □

Un cammino nomina archi e raggiunge nodi

Laurea Magistrale in Informatica, Università di Padova

12/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 5

Grafo dei nomi con nodo radice unico e di nome implicito

Contenuto del nodo (directory) n1
 n2: "elke"
 n3: "max"
 n4: "steen"

Data stored in n6
 n6 è un sym link verso n5

Leaf node ○

Directory node □

Laurea Magistrale in Informatica, Università di Padova
13/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 7

□ La realizzazione di uno spazio dei nomi può avere 3 livelli gerarchici

- **Livello globale** → i nodi di più alto livello
 - Radici e loro figli (informazione generalmente stabile)
- **Livello amministrativo** → i nodi directory amministrati da una singola autorità
 - Ciascun nodo *directory* rappresenta gruppi di entità appartenenti alla stessa organizzazione o unità amministrativa
- **Livello gestionale** → i nodi che possono cambiare frequentemente
 - Ciascun nodo gestito dall'utente e dall'amministratore di sistema

Laurea Magistrale in Informatica, Università di Padova
15/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 6

□ La risoluzione dei nomi è possibile anche su più *name space* uniti trasparentemente

- Analogamente al principio del *mount* su *file system*
- Dove un nodo *directory* di un *name space* diventa la radice di un *name space* esterno importato

□ Per accedere al *name space* esterno serve conoscere

- Il protocollo di accesso al nodo esterno che lo ospita
- Il nome del *server* che lo gestisce
- Il nome del nodo radice nel *name space* esterno

Laurea Magistrale in Informatica, Università di Padova
14/45

Sistemi distribuiti: gestione dei nomi

Lo spazio dei nomi – 8

□ I 3 livelli nello spazio dei nomi realizzato utilizzando il DNS di Internet

Laurea Magistrale in Informatica, Università di Padova
16/45

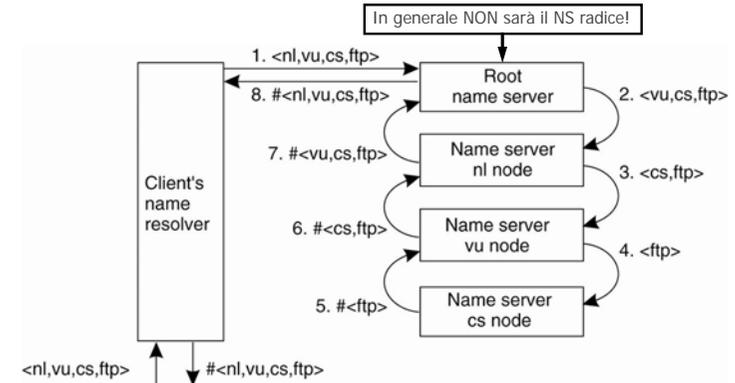


Lo spazio dei nomi – 9

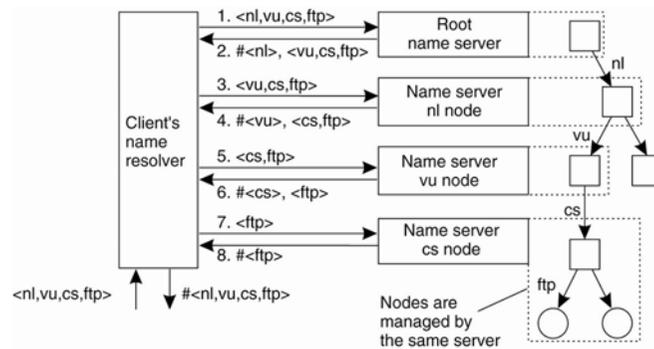
- ❑ **Name resolver**
 - L'entità localmente responsabile per la risoluzione dei nomi
- ❑ **Name server**
 - L'entità che gestisce i nomi della propria *directory*
- ❑ **2 tecniche di risoluzione dei nomi**
 - **Iterativa** → il *name server* interrogato (quale?) fornisce la migliore risposta in suo possesso senza fare interrogazioni
 - Il *resolver* determina il nome completo dell'entità e lo fornisce al *name server* radice
 - Il cliente svolge più lavoro del servente → non conviene fare *caching* presso il cliente
 - **Ricorsiva** → il *name server* interrogato (quale?) interroga altri *name server* per costruire la risposta
 - Il *name server* interrogato riceve il nome completo e fornisce la risposta finale
 - Il servente svolge più lavoro del cliente → conviene fare *caching* presso il cliente



Risoluzione ricorsiva



Risoluzione iterativa



Lo spazio dei nomi – 10

- ❑ **Directory service**
 - Realizzano sistemi di *naming* basati su attributi invece che su nomi strutturati
 - Come fa invece il DNS per il quale si parla di *naming service*
 - Fondamentalmente un protocollo di accesso (:389) e uno schema dati
 - 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*) con campi con attributi dal nome unico
 - L'architettura LDAP è molto simile a quella DNS ma più sofisticata e potente



Sistemi distribuiti: gestione dei nomi

LDAP *directory entry*

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
CommonName	CN	Main server
Mail_Servers	—	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	—	130.37.20.20
WWW_Server	—	130.37.20.20

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc

Laurea Magistrale in Informatica, Università di Padova
21/45



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*

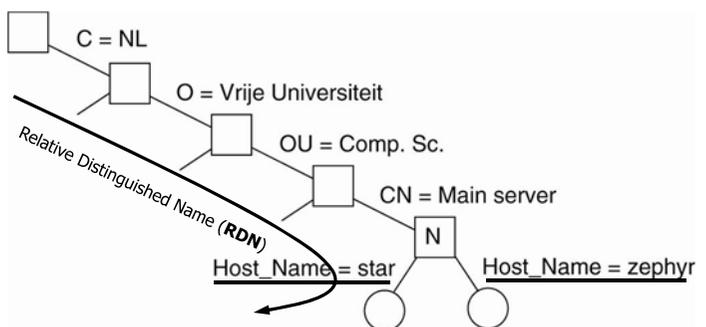
Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc

Laurea Magistrale in Informatica, Università di Padova
23/45



Sistemi distribuiti: gestione dei nomi

LDAP *directory information tree*



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc

Laurea Magistrale in Informatica, Università di Padova
22/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 2

- ❑ Cambiare i nomi usati come identificatori invalida i *symbolic link* che li utilizzano
- ❑ Come modificare lo spazio dei nomi del nodo *directory* di origine
 - Soluzione 1: associare nuovo indirizzo a vecchio nome
 - *Look-up* veloce
 - Ma ogni successivo aggiornamento (su nodo di origine) ha costo elevato
 - Soluzione 2: associare nuovo nome a valore del vecchio nome
 - = *symbolic link*
 - *Look-up* più lento
 - Facile aggiornamento tramite nuovo *symbolic link* locale al costo di ulteriore indirezione

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc

Laurea Magistrale in Informatica, Università di Padova
24/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 3

- ❑ Entrambe le soluzioni sono inadeguate per sistemi distribuiti a larga scala
- ❑ Conviene separare 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*

Laurea Magistrale in Informatica, Università di Padova

25/45



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 semplicistica: *broadcasting*
 - ARP (*Address Resolution Protocol*) restituisce l'indirizzo di livello collegamento dati in rete locale corrispondente a un indirizzo IP in ingresso
 - Troppo oneroso per reti vaste
 - Meglio *multicast* verso gruppi specifici (anche dinamici) di nodi coinvolti

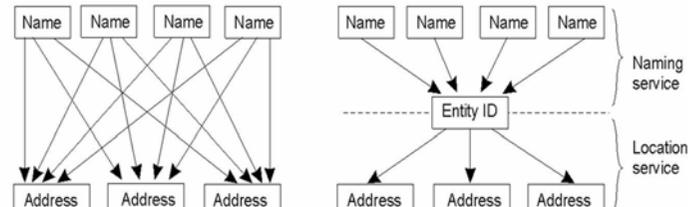
Laurea Magistrale in Informatica, Università di Padova

27/45



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

Laurea Magistrale in Informatica, Università di Padova

26/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 6

- ❑ Soluzione migliorativa: *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
 - Nei sistemi a oggetti distribuiti può essere realizzato con una catena di coppie SSP <*proxy, skeleton*>
 - *Proxy (stub)* → riferimento allo *skeleton* finale o intermedio
 - *Skeleton (scion)* → riferimento locale all'oggetto oppure al suo *proxy* locale
 - Scion = discendente in linea ereditaria

Laurea Magistrale in Informatica, Università di Padova

28/45

Sistemi distribuiti: gestione dei nomi

Entità mobili – 7

Un oggetto che cambia locazione lascia un *proxy* al suo vecchio indirizzo e installa uno *skeleton* che punta a se nella nuova locazione

Questa migrazione è del tutto trasparente al cliente

La richiesta viaggia verso l'oggetto, non l'indirizzo verso il cliente!

Il sistema SSP (*stub-scion-pairs*) di forwarding

Laurea Magistrale in Informatica, Università di Padova

29/45

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 cui 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

Laurea Magistrale in Informatica, Università di Padova

31/45

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
- Ciò crea un cortocircuito tra cliente e oggetto remoto

Laurea Magistrale in Informatica, Università di Padova

30/45

Sistemi distribuiti: gestione dei nomi

Entità mobili – 10

□ **Soluzione ad approccio gerarchico /I**

- **Rete di interconnessione suddivisa in una collezione di domini (similmente al 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 *directory* che conosce tutte le entità presenti in esso**
 - Il nodo *directory* del dominio di livello più alto è detto nodo radice e conosce tutte le entità dell'intero sistema

Laurea Magistrale in Informatica, Università di Padova

32/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 11

□ **Soluzione ad approccio gerarchico /II**

- Ogni entità E di dominio D è rappresentata da un *location record* nel nodo *directory* N di D
 - Il *location record* di E in N 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 un *name record* per ogni entità di sistema
 - Il *name record* conterrà l'inizio di una catena di puntatori a nodi *directory* fino a quello a cui *location record* contiene l'indirizzo dell'entità

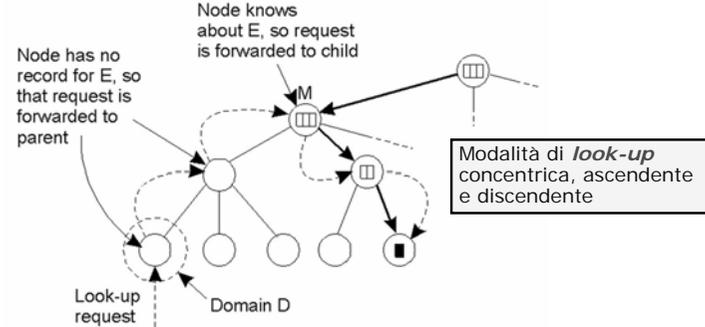
Laurea Magistrale in Informatica, Università di Padova

33/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 13



Laurea Magistrale in Informatica, Università di Padova

35/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 12

□ **Soluzione ad approccio gerarchico /III**

- Il *look-up* di E ha inizio nel nodo *directory* del dominio D_0 di residenza del cliente che lo richiede
 - Se D_0 non contiene informazioni per E allora E non si trova in D_0
 - La richiesta viene inviata al nodo *directory* del dominio D_1 "genitore" di D_0 (che per definizione conosce più entità di D_0) e così via fino a che un *name record* per E venga trovato nel nodo *directory* del dominio D_N
 - Allora 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

Laurea Magistrale in Informatica, Università di Padova

34/45



Sistemi distribuiti: gestione dei nomi

Entità mobili – 14

□ **Soluzione ad approccio gerarchico /IV**

- Concentrare tutti i *name record* nel nodo radice pone gravi problemi di scalabilità
 - Centralizzazione → collo di bottiglia
- Meglio partizionare la conoscenza
- 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

Laurea Magistrale in Informatica, Università di Padova

36/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 1

- Un'entità che non possa essere riferita è inutile al sistema e va rimossa
 - *Garbage collection*
- La rimozione può essere esplicita
 - "L'ultimo" processo a usare una data entità può rimuoverla
 - Come individuare "l'ultimo" processo in un sistema distribuito?

Laurea Magistrale in Informatica, Università di Padova 37/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 3

Laurea Magistrale in Informatica, Università di Padova 39/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non 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

Laurea Magistrale in Informatica, Università di Padova 38/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 4

- Tecniche di rimozione: *reference counting*
 - Adatta ai sistemi centralizzati ma ostacolata dai possibili errori di comunicazione nei sistemi distribuiti
 - L'oggetto remoto mantiene nel suo *skeleton* un contatore dei riferimenti
 - Ogni *proxy* creato presso nodi cliente invia una notifica di incremento allo *skeleton* con conferma di 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 invio di notifica di decremento allo *skeleton* che può però andare perduta
 - L'oggetto potrebbe restare in vita anche se privo di utenti

Laurea Magistrale in Informatica, Università di Padova 40/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 5

- ❑ Grave problema del *reference counting* è la *data race* tra incrementi e decrementi
- ❑ Meglio usare solo decrementi
 - Creazione oggetto remoto: allo *skeleton* vengono assegnati un peso totale ($RC = 2^n$) e un peso parziale ($WR = 2^m$)
 - Inizialmente $n = m$
 - Creazione riferimento: $\frac{WR}{2}$ dello *skeleton* viene ceduto al *proxy* corrispondente, con invariante $RC = \sum_i WR_i$
 - A ogni passaggio di riferimento, il *proxy* emittente cede $\frac{1}{2}$ del suo peso al destinatario
 - Rimozione di riferimento: il *WR* del corrispondente *proxy* viene sottratto a *RC* dello *skeleton*

Laurea Magistrale in Informatica, Università di Padova

41/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 7

Al passaggio (copia) del riferimento:
 $WR(P_{P2}) = \frac{WR(P_{P1})}{2}, WR(P_{P1}) = \frac{WR(P_{S1})}{2}, RC(S) = \sum_i WR_i$

Laurea Magistrale in Informatica, Università di Padova

43/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 6

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

Ala creazione del primo riferimento:
 $WR(P) = \frac{WR(S)}{2}, WR(S) = \frac{WR(S)}{2}, RC(S) = \sum_i WR_i$

Laurea Magistrale in Informatica, Università di Padova

42/45

Sistemi distribuiti: gestione dei nomi

Rimozione entità non riferibili – 8

- ❑ Ma in questo modo ogni oggetto remoto consente solo N riferimenti per N fissato
 - Per *WR* (di *skeleton* e/o di *proxy*) non più dimezzabile restituendo un intero non sono più consentiti riferimenti
- ❑ Risolto tramite variante del *forwarding*
 - Una cella di indirectione (*IC*) rileva $WR = 1$ ma riceve $RC = 2^n$ che ripartisce tra i nuovi riferimenti come se fosse un nuovo oggetto

Laurea Magistrale in Informatica, Università di Padova

44/45



Rimozione entità non riferibili – 9

□ 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
 - Il riferimento remoto viene considerato in *leasing* e rimosso in assenza d'uso
- Prima che il GC locale rimuova un *proxy* ne comunica la cancellazione allo *skeleton* corrispondente
 - Gestione dei riferimenti nello *skeleton* via RPC con semantica *at-most-once*
- Più informativa del *reference counting* che è anonimo
 - Ma con rischio di *data race* tra inserzioni e rimozioni di riferimenti