INFORMATICA 2/89

UPRAVLJANJE Z IMENI V PORAZDELJENIH SISTEMIH

Keywords: management by naming, distributed systems

Jože Rugelj
Institut Jožef Stefan, Ljubljana

POVZETEK Članek podaja pregled in definicije osnovnih pojmov, ki so povezani z imenovanjem objektov v računalniških sistemih. Imenske strukture pomagajo premagovati razlike med potrebami računalnika in željami njegovega uporabnika. Pomen imenskih struktur se veča s kompleksnostjo sistemov in njihov procesno močjo.

ABSTRACT This article gives an overview and definitions of the basic terms concerning naming of the objects in computer systems. Naming structures can reduce the differences between computer and its user. The importance of naming structures is growing with the complexity of systems and with their process power.

Imena imajo v računalniških sistemih zelo pomembno vlogo, saj z njihovo pomočjo identificiramo izbrane objekte na vseh nivojih abstrakcije. Imena uporabljamo na različnih področjih pri upravljanju, npr. pri zaščiti, nadzoru napak in pri upravljanju objektov ter pri njihovem lociranju in delitvi med več uporabnikov. V porazdeljenih tesno sklopljenih računalniških sistemih je ta vloga še pomembnejša in bolj kompleksna zaradi dovoljene raznolikosti posameznih elementov sistema in njihove porazdeljenosti ter želje, da bi uporabnik videl sistem kot enovito celoto [Ruge88].

1 DEFINICIJE OSNOVNIH POJMOV

Najbolj splošna definicija za ime bi bila:

Ime je niz znakov iz neke abecede, ki označuje neko množico objektov.

Vsa imena, ki jih na osnovih pravil lahko tvorimo nad dano abcedo, sestavljajo *imenski prostor* . Pravila za tvorbo imen pa so lahko zelo različna in so odvisna od kompleksnosti in strukture sistema, v katerem označujemo objekte s tako dobljenimi imeni.

Najbolj enostavna oblika imenskega prostora je *ploščat* imenski prostor. Vsak imenski strežnik mora poznati vsa imena v sistemu. Imena sama ne izražajo nobene strukture. Pri kompleksnejših sistemih mora biti imenskemu sistemu pridružena neka dodatna možnost strukturiranja, ki navzven ni vidna.

Bolj kompleksne strukture imen pa imenujemo *hierarhične*. V tem primeru je ime sestavljeno iz zaporedja enostavnih imen. Enostavna imena so med seboj povezana z ločilnimi simboli. Posamezne komponente sestavljenega hierarhičnega imena lahko vsebujejo informacije o fizični lokaciji imenovanega objekta ali o njegovi organizacijski pripadnosti. Taki podatki sicer pomagajo pri iskanju objektov, hkrati pa zmanjšujejo prilagodljivost sistema kot celote: onemogočajo premikanje objektov v sistemu, nadomeščanje objektov z drugimi objekti in podvajanje objektov. Vse te zahteve pa so aktualne v sodobnih porazdeljenih sistemih.

Tudi same oznake imen so v različnih virih različne. Med prvimi je poskušal probleme z imenskimi strukturami v porazdeljenih sistemih formalizirati J.F. Shoch [Shoc78], ki razlikuje tri vidike identifikacije objektov: ime, naslov in pot. Razlika med identifikatorji je v tem primeru semantična.

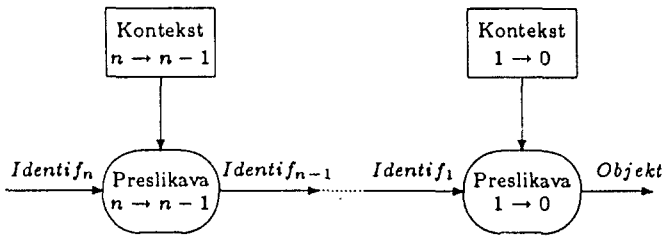
Ime pove kaj iščemo,
naslov kaže kje to je in
pot pokaže kako pridemo tja.

Sintaktična razlika med imeni je v izboru abecede in v pravilih za tvorbo imen, predvsem pri določanju dolžine in pri njeni spremenljivosti. Značilna povezanost semantičnih in sintaktičnih lastnosti identifikatorjev je v tem, da so naslovi običajno fiksne dolžine in sestavljeni iz niza števil, imena pa so spremenljive dolžine in sestavljena iz alfanumeričnih znakov z nekaterimi omejitvami.

J.H.Saltzer v svojem delu [Salt78] obravnava problem imen v centraliziranih operacijskih sistemih, ki se je pojavil zaradi povečevanja računalniških sistemov in združevanja ter prenašanja velikih programskih paketov, vendar v zaključku dela ugotavlja, da so dobljene ideje in ugotovitve uporabne tudi v porazdeljenih sistemih, saj gre pri njih za podobne okoliščine. Za preučevanje imenske problematike uporablja objektni model računalniškega sistema. Računalniški sistem upravlja objekte, ki izvajajo procesiranje. Objekt lahko vsebuje druge objekte, ki jih identificira z imeni. Povezava med imeni in objekti je preslikovalna funkcija, ki je pridružena objektu in jo Saltzer imenuje *kontekst*. Realizirana je kot tabela, ki vsebuje pare $< ime, naslov >$. Tak pristop v različnih oblikah in z različnimi izboljšavami so kasneje privzeli skoraj vsi snovalci imenskih sistemov v porazdeljenih sistemih. Na tak način je dosežena velika fleksibilnost imenske strukture.

Problemi, ki se pojavljajo pri uporabi opisanega imenskega sistema, so imenski konflikti, ki se pojavljajo pri dodajanju neodvisno snovanih objektov in posledice napačno izbranih začetnih kontekstov. Pri bolj kompleksnih sistemih se konteksti pojavljajo na več nivojih zaradi večkratne vsebovanosti objektov. Zaradi fleksibilnosti imenskega sistema in dinamičnega spreminjanja mora uporabnik tudi paziti na konsistentnost danih povezav imen in naslovov. Tudi povezanost imenskega sistema s funkcijami zaščite, upravljanja virov ali upoštevanje faktorjev ekonomičnisti prinaša omejitve in z njimi povezane probleme.

Watson za imena v najširšem smislu pomena uporablja termin $identifikator_{nivo}$, da bi se s tem poenotil imensko hierarhijo in se izognil dvoumnostim (slika 1). Imena so definirana na več logičnih nivojih, ki se ločijo po stopnji abstrakcije pri predstavitvi sistema [Wats81].



Slika 1: Watsonova imenska hierarhija

Ime se interpretira v okviru nekega konteksta. Kontekst je množica povezav imen in objektov. Vsako ime je element vsaj enega konteksta. Postopek iskanja objekta, ki pripada imenu v določenem kontekstu, imenuje *razreševanje imena (resolving)*. Razreševanje imena lahko poteka v več stopnjah. Rezultat razrešitve imena znotraj konteksta lahko namreč da nov kontekst (nižji logični nivo abstrakcije) in šele na najnižjem nivoju dosežemo objekt. Vsak nivo ima svojo imensko strukturo in pravila za dodeljevanje in razreševanje imen. Rezultat razreševanja je običajno identifikator na nižjem logičnem nivoju.

Potreba po več nivojih imen izhaja predvsem iz dveh vzrokov:

- različnosti potreb človeka in računalnika in
- različnosti med lokalnimi in globalnimi potrebami porazdeljenega računalniškega sistema.

Pri izbiri imen se človek odloča za imena, ki so mnemonično uporabna in zato sestavljena iz črk, dolžine imen so različne in zaradi tega, ker jih izbira človek, so lahko tudi dvoumna. Taka imena so za računalnik slabo uporabna, saj zahteva nedvoumna, kratka imena stalne dolžine, nad abecedo z dvema črkama.

Globalna imena so v velikih sistemih lahko zelo dolga in nerodna za uporabo. Zato v lokalnih okoljih lahko uporabimo skrajšana imena ali tako imenovana *domača imena.*

Na izbiro imen pa poleg prej omenjenih faktorjev vplivajo še drugi, predvsem ekonomski, in želja, da imenski sistem neposredno podpira še druge upravljalske funkcije, predvsem zaščito.

Ekonomski faktor običajno vpliva na velikost imenskega prostora, ki je odvisna od velikosti pomnilnika, namenjenega za shranjevanje imenske podatkovne baze in procesorskih zmožnosti, ki morajo biti pri velikem imenskem prostoru in dolgih imenih precej večje. Velik imenski prostor pa ima seveda vrsto prednosti. V njem je praktično neskončno mnogo imen in zato jih po uporabi ni treba ponovno uporabljati, ampak jih enostavno zavržemo. Število imen lahko povečamo tudi tako, da dovolimo različno dolga imena, saj se na ta način zelo poveča število možnih kombinacij znakov iz abecede.

2 STRUKTURA IMENSKEGA PROSTORA

V obstoječih porazdeljenih sistemih je upravljanje z imeni v glavnem odvisno od imenske strukture.

Večina sistemov ima drevesno strukturo imenskega prostora. Veje drevesa imajo oznake, informacija o objektih pa je v listih dreves. Ime objekta predstavlja niz oznak z vej, ki so med korenom drevesa in listi.

Imena v najširšem pomenu besede vsebujejo poleg oznake objekta še dodatne informacije o specifičnih atributih objekta. Primeri takih atributov na uporabniškem nivoju so informacije o upravno-administrativni, geografski ali mrežno topološki pripadnosti objekta [Su83]. Po Shochovi definiciji je ime edini identifikator, ki ne vsebuje nobenih karakterističnih atributov.

Če ime ne vsebuje karakterističnih atributov, ima to svoje prednosti in slabosti. Glavna slabost je v tem, da je razreševanje imena razmeroma kompleksno. Ime ne vsebuje nobenih namigov o lokaciji objekta oziroma o njegovem naslovu. Ravno ta transparentnost lokacije pa je zaželjena pri rekonfiguraciji porazdeljenega sistema, pri združevanju ali cepitvi porazdelejnih sistemov in pri spremembah lokacije objekta. Vse te spremembe vplivajo samo na preslikovalne mehanizme, uporabljene v postopku razreševanja imena.

Kot smo že omenili, imenujemo iskanje povezave imena objekta in njegovega naslova (in s tem lokacije objekta) razreševanje imena. Pravila za imenovanje objektov določajo, v katerem trenutku se izvede razreševanje. Kolikor kasneje se imena preslikajo v naslove, toliko večja je fleksibilnost sistema.

Obravnavo problemov, povezanih s poimenovanjem, lahko poenotimo z vključitvijo v model stranka-strežnik. Vse aktivnosti, ki so potrebne v postopku razreševanje, izvaja imenski strežnik in jih v obliki servisov nudi uporabniku. Imenski strežnik je tesno povezan s pojmom konteksta [Come86]. Kontekst predstavlja tisto particijo imenskega prostora, ki ga upravlja imenski strežnik. Strežniku je tako pridružena preslikovalna funkcija, ki služi za razreševanje imen znotraj particije. Seveda pa je znotraj imenskega prostora lahko več strežnikov in s tem tudi kontekstov.

3 PORAZDELJENO UPRAVLJANJE Z IMENI

Porazdeljeno upravljanje z imeni sestavljajo tri glavne aktivnosti, ki so tesno povezane in sodelujejo med seboj in z uporabniki servisov, ki jih nudijo:

- Dodeljevenje imen
- Porazdeljevanje imenskega prostora
- Razreševanje imen

Dodeljevanje imen

Dodeljevanje imen novim objektom je prepuščeno tistim, ki nov objekt kreirajo. Vsako novo ime pa je treba registrirati pri imenskem strežniku. Le-ta preveri, če tako ime še ne obstaja, da ne bi prišlo do dvoumnosti. V primerih, ko imena, ki ga želi uporabnik registrirati, še ni v sistemu, imenski strežnik tako ime sprejme. Če pa tako ime slučajno že obstaja, strežnik ime zavrne in uporabnik poskuša znova. V porazdeljenih sistemih, kjer je več strežnikov, morajo le-ti komunicirati med seboj pri vsaki registraciji novega imena.

Porazdeljevanje imenskega prostora

Porazdeljevanje imenskega prostora je dodeljevanje odgovornosti za posamezne dele imenskega prostora imenskim strežnikom. Kot smo že prej omenili, je v porazdeljenih sistemih več strežnikov, saj se s tem poveča zanesljivost, hitrost delovanja in prilagodljivost sistema. Zaradi zanesljivosti se lahko imenski prostori posameznih strežnikov tudi prekrivajo, vendar to lahko povzroči težave predvsem pri upravljanju in delitvi odgovornosti

Pooblaščenega imenskega strežnika oziroma več strežnikov določi stranka, ki je objekt kreirala, takoj ko eden od strežnikov potrdi uporabo predlaganega imena. Med delovanjem na podoben način lahko prenesemo pooblastilo drugim strežnikom.

Izbira strežnika je popolnoma neodvisna od imena oziroma oblike imena. Strankam po izbiri strežnika ni treba ohranjati informacije o izbranem pooblaščenem strežniku, saj jo v vsakem trenutku lahko dobi od kateregakoli imenskega strežnika. Vsak strežnik hrani podatke o pooblastilih v *konfiguracijski podatkovni bazi* . V velikih sistemih je taka podatkovna baza prevelika, da bi jo vzdrževal vsak strežnik v celoti, in je porazdeljena. Iskanje določenega podatka tako zahteva sodelovanje med strežniki.

Prostor, ki ga konfiguracijska baza zavzema in čas, ki ga strežnik porabi za njeno vzdrževanje in uporabo, predstavljata ceno za strukturno neodvisno upravljanje z imeni. Pri običajnih imenskih strukturah so namreč vse te informacije vsebovane v samem imenu objekta in v njegovi strukturi.

Skupaj s sporočilom o pooblastilu mora stranka pooblaščenemu strežniku podati tudi atribute imenovanega objekta, ki potem omogočajo strežniku vzdrževanje podatkovne baze z atributi objekta za upravljanje z imeni. Kot smo že poudarili, ime sámo v splošnem ne vsebuje dodatnih informacij o objektu.

Pri izbiri pooblaščenega strežnika oziroma strežnikov je glavno vodilo izpolnitev čimvečjega števila ciljev, ki jih izpolnjujemo v porazdeljenem sistemu. Poudarili bi predvsem zanesljivost delovanja in čim večjo hitrost ter izkoriščenost virov v sistemu. Z izbiro imenskega strežnika, ki je blizu uporabniku ali imenovanim objektom, vplivamo na izpolnitev vseh naštetih ciljev.

Razreševanje imen

Ime lahko razrešimo z zahtevkom poljubnemu imenskemu strežniku v sistemu. Če strežnik ni pooblaščen za delo s podanim imenom, uporabniku vrne ime najbližjega pooblaščenega strežnika ali pa sam posreduje zahtevek le-temu. Šele pooblaščeni strežnik poišče iz podatkovne baze atributov zahtevane podatke. Količina informacije oziroma število atributov v podatkovni bazi je zelo različno, odvisno od potreb in zahtev uporabnikov imenskega servisa.

V primeru, ko so take podatkovne baze replicirane, je treba uporabljati poznane mehanizme za ohranjanje konsistentnosti. Pri tem so zaradi prevladovanja operacije branja in zelo redke uporabe pisanja mehanizmi enostavni.

4 KONTEKSTI - PORAZDELITEV IMENSKEGA PROSTORA

Pri razreševanju imen se se je kot koristna izkazala uporaba *kontekstov* . Ta pojem sta uvedla že Saltzer in Watson [Salt78], [Wats81], dobro pa je obdelana uporaba kontekstov v doktorski tezi [Pete85] in članku [Come86].

Kontekst v splošnem predstavlja razdelitev imenskega prostora na osnovi geografskih, organizacijskih ali funkcionalnih pripadnosti objektov. Bolj določno opredelitev konteksta podaja [Terr86], ki definira kontekst kot zbirko seznamov, ki vsebujejo imena pooblaščenih imenskih strežnikov.

Uporabnik imenujejo objekte v sistemu z imeni, ki so lahko sestavljena iz niza enostavnejših imen, ločenih s posebnimi ločilnimi znaki. Glavna naloga razreševalnega mehanizma je preslikava uporabniških imen v *primitivna imena*, ki imajo zelo enostavno sintakso, so enolična in so fiksne dolžine. To pomeni, da so fizični naslovi objektov ena od oblik primitivnih imen.

Imenski strežniki v porazdeljenih sistemih so po Petersonu tesno povezani s konteksti. Tvorijo jih podatkovne strukture, ki vsebujejo množico povezav in program, ki vrne povezave pridružene danemu imenu. Že Peterson omenja možnost, da interna reprezentacija imenskega strežnika ni nujno identična zunanji predstavitvi imenskega sistema. Tako lahko npr. imenski strežnik hierarhična imena 'stisne' v ploščato strukturo, če mu to olajša delo z imeni. Koncept strukturno neodvisnega upravljanja z imeni je razvijal Terry.

5 STRUKTURNO-NEODVISNO UPRAVLJANJE Z IMENI

D.B. Terry v [Terr86] uvaja pojem strukturno neodvnisnega upravljanja z imeni. Ta je konkretizacija ideje o imenih brez karakterističnih atributov. Pri velikih sistemih, ki se dinamično spreminjajo, je zelo težko ali celo nemogoče vnaprej predvideti vsa stanja sistema in njegove konfiguracije. Dodajanje elementov strojne in programske opreme in priključevanje in odhajanje uporabnikov so aktivnosti, ki se pogosto dogajajo v sistemu, vsako spreminjanje imen pa je drago zaradi možne povezanosti velikega števila objektov. Zato mora biti imenska struktura in delo imenskih strežnikov čimbolj neodvisno od stanja sistema.

5.1 Konteksti v strukturno-neodvisnem upravljanju

Po Terryevi definiciji kontekst predstavlja nedeljivo enoto za shranjevanje podatkov o pooblaščenih strežnikih. Z njim lahko dela več strežnikov. Hkrati ima lahko en strežnik več kontekstov. Imena kontekstov niso pomembna za uporabnika, saj so konteksti skriti v imenskih strežnikih in jih stranke ne vidijo. Konteksti tudi ne vsebujejo nobenih informacij o objektih, katerih imena so povezana z njimi. To pomeni, da sta si drevesna struktura kontekstov ter razreševanje imen in drevo, ki predstavlja poljubno urejenost imenskega prostora, vidno uporabniku, tuja. Tako lahko uporabnik prilagodi imensko strukturo svojim potrebam, s tem pa ne prizadene prilagodljivosti sistema in prednosti strukturno neodvisnega upravljanja z imeni.

Zaradi velike neodvisnosti kontekstov od značilnosti sistema lahko konfiguracijsko podatkovno bazo razstavimo na kontekste z enostavnimi algoritmi, brez posredovanja uporabnikov. Uporabnik lahko vpliva na porazdelitev in dodeljevanje kontekstov strežnikov samo v primeru, če to eksplicitno zahteva. Taki posegi so koristni predvsem pri optimizaciji, izvedemo pa jih z ukazi v t.i. *ortogonalnih jezikih* [Pope85].

Enostaven mehanizem za razdelitev imenskega prostora v kontekste je uporaba *pogoja za razvrščanje v skupine* . Pogoj za razvrščanje imen je enostavna funkcija, ki kot odgovor vrne logično vrednost *PRAVILNO* ali *NEPRAVILNO*, če ji kot argument podamo ime. Imena, pri katerih je odgovor pozitiven, pripadajo kontekstu, katerega last je pogoj za razvrščanje. Razpršilne funkcije so primer takih pogojev za razvrščanje. Pri imenih pa je za razdelitev v kontekste še bolj tipičen način sintaktičnega analiziranja imen in primerjava s podanim vzorcem - nizom znakov iz neke abecede.

Razvrščevalni pogoji so uporabljeni za dodelitev imen kontekstom in za razdelitev kontekstov v manjše kontekste. Postopek začnemo nad celotnim imenskim prostorom in z zaporedno uporabo razvrščevalnih pogojev dobimo željeno velikost kontekstov.

Da bi preverili pripadnost imena nekemu kontekstu je treba nad tem imenom uporabiti vse razvrščevalne pogoje, ki ustrezajo kontekstom, znotraj katerih je ime. Strežnik ob zahtevi za rezrešitev imena najprej pregleda svoj prostor. Če v svojem prostoru ne najde iskanega imena, mora uporabiti dodatne informacije, ki so v imenskem strežniku shranjene v obliki *kontekstnih povezav* . Kontekstne povezave združujejo razvrščevalne

pogoje z imeni kontekstov in lokacijami imenskih strežnikov, ki so pooblaščeni za kontekst. Strežnik tako poišče pooblaščenega strežnika za dani kontekst in mu preda zahtevek. Postopek se nadaljuje, dokler ne dobi zahtevka strežnik, ki ga lahko dokončno razreši, torej vrne iskane atribute imenovanega objekta.

Ob začetku razreševanja je treba podati *začetni kontekst*. Začetni kontekst mora vsebovati vsaj kontekstne povezave za vse objekte v sistemu. V primeru, če začetni kontekst vsebuje kar vse atribute vseh objektov, torej se vsa imena razrešijo v enem koraku, je tak kontekst *globalen*. Globalna so torej tudi vsa imena. V nasprotnem primeru pa imenujemo imena *relativna* glede na začetni kontekst.

Globalna imena so praktično uporabna samo za manjše sisteme s centraliziranim upravljanjem imen. Tipično začetni konteksti vsebujejo samo kontekstne povezave. Kontekstne povezave tvorijo drevo. Samo listi tega drevesa, imenovanega *drevo za razreševanje*, vsebujejo atribute o objektih.

Upravljanje z imeni pomeni vzdrževanje informacij o imenovanih objektih. Te informacije imenujemo atributi, ki niso in ne smejo biti vključene v imenu.

Kot smo že omenili, drevo za razreševanje ni nujno podobno drevesu imenskega prostora. Ravno ta relativna nepovezanost omogoča veliko prilagodljiost imenskih strežnikov spreminjajočim se zahtevam v porazdeljenih sistemih, ne da bi pri tem uporabniki čutili spremembe ali bi celo morali spreminjati imena objektov. Dodajanje novih objektov, ki bi lahko pokvarilo uravnoteženost drevesa, rešimo enostavno tako, da v kritično vozlišče dodamo nove pogoje za razvrščanje. Zelo težko je namreč predvideti, kako velik imenski prostor bo razvijajoč porazdeljen sistem potreboval v prihodnosti. Tudi tako imenovana ploščata imena, ki nimajo posebne strukture, lahko razdelimo na kontekste. Pri tem je odprto vprašanje ustreznih funkcij za razvrščevalne pogoje, ki bi iz na videz nestrukturirane množice poljubno izbranih imen poiskale neko strukturo za razdelitev imenskega prostora v kontekste.

6 UČINKOVITOST SISTEMA ZA UPRAVLJANE Z IMENI IN KVALITETA NJEGOVEGA SERVISA

Učinkovitost sistema za upravljane z imeni lahko merimo s ceno procesiranja in porabo pomnilniškega prostora, ki jo sistem povprečno porabi za razreševanje. Seveda je jasno, da je najbolj učinkovit imenski sistem, kjer bi uporabljali neposredne naslove objektov. Take so bile rešitve v prvih računalnikih. Čimvečji je sistem in čimbolj je kompleksen, večjo ceno so pripravljeni uporabniki plačati za storitve, ki jim pomagajo pri delu. To velja tudi za imenski sistem.

Pomemben faktor kvalitete servisa je število nivojev v imenski strukturi. Zahteve za obliko in vsebino imen za računalnik in človeka so ravno nasprotne. Ker so zahteve računalnika nespremenljive, se mora prilagoditi človek. Čimvečje je število nivojev, tem bolj se lahko imena prilagodijo človekovim zahtevam. Možno je tudi doseganje tranparentnosti imen. Tudi prilagodljivost sistema se na tak način poveča.

Prilagodljivost sistema za dinamične in dolgoročne spremembe pa povečuje tudi strukturna neodvisnost razreševalnega mehanizma od imenske stukture. Pri razreševanju se poveča učinkovitost, če minimiziramo potrebo po prenašanju informacij med procesnimi mesti. Sheltzer v [Shel86] predlaga uporabo *predpomnilnikov (cache)* za zmanjševanje prometa po komunikacijskih kanalih.

7 LITERATURA

[Come86] D.E. Comer, L.L. Peterson:
A Model of Name Resolution in Distributed Systems, IEEE Proc. 6. ICDCS, May 1986

[Pete85] L.L. Peterson:
Defining and Naming the Fundamental Objects in a Distributed Message System,
Ph.D. Thesis, Purdue University, May 1985

[Pope85] G. Popek, B.J. Walker (edts.):
LOCUS Distributed System Architecture,
MIT Press, 1985

[Ruge88] J. Rugelj:
Upravljanje porazdeljenih računalniških sistemov, Magistrsko delo,
Fakulteta za elektrotehniko, Ljubljana 1988

[Salt78] J.H. Saltzer:
Naming and Binding of Objects,
LNCS 60, Springer-Verlag 1978

[Shel86] A.B. Sheltzer, R. Lindell, G.J. Popek:
Name Service Locality and Cache Design in a Distributed Operating Systems,
IEEE Proc.6th ICDCS, May 1986

[Shoc78] J.F. Shoch:
Inter-network Naming, Adressing and Routing,
COMPCON 78, April 1978

[Su83] Z.S. Su:
Identification in Computer Networks,
ACM Computer Comm. Rev. vol.13/10, 1983

[Terr86] D.B. Terry:
Structure-free Name Management for Evolving Distributed Enviroments
IEEE Proc. 6th ICDCS, May 1986

[Wats81] R.W. Watson:
Identifiers in distributed systems,
in *Distributed systems*, LCNS 105,
Springer-Verlag, Berlin 1981