

Bab 6. Name Service

6.1 Pendahuluan

Name Service dalam Sistem Terdistribusi merupakan layanan penamaan yang berfungsi untuk menyimpan *naming context*, yakni kumpulan *binding* nama dengan objek, tugasnya untuk *me-resolve* nama.

Pengaksesan *resource* pada sistem terdistribusi yang memerlukan:

- Nama *resource* (untuk pemanggilan),
- Alamat (lokasi *resource* tsb),
- Rute (bagaimana mencapai lokasi tsb).

Name Service memiliki konsentrasi pada aspek penamaan dan pemetaan antara nama & alamat, bukan pada masalah rute, yang dibahas di Jaringan Komputer. *Resource* yang dipakai dalam *Name Service* adalah: komputer, layanan, *remote object*, berkas, pemakai.

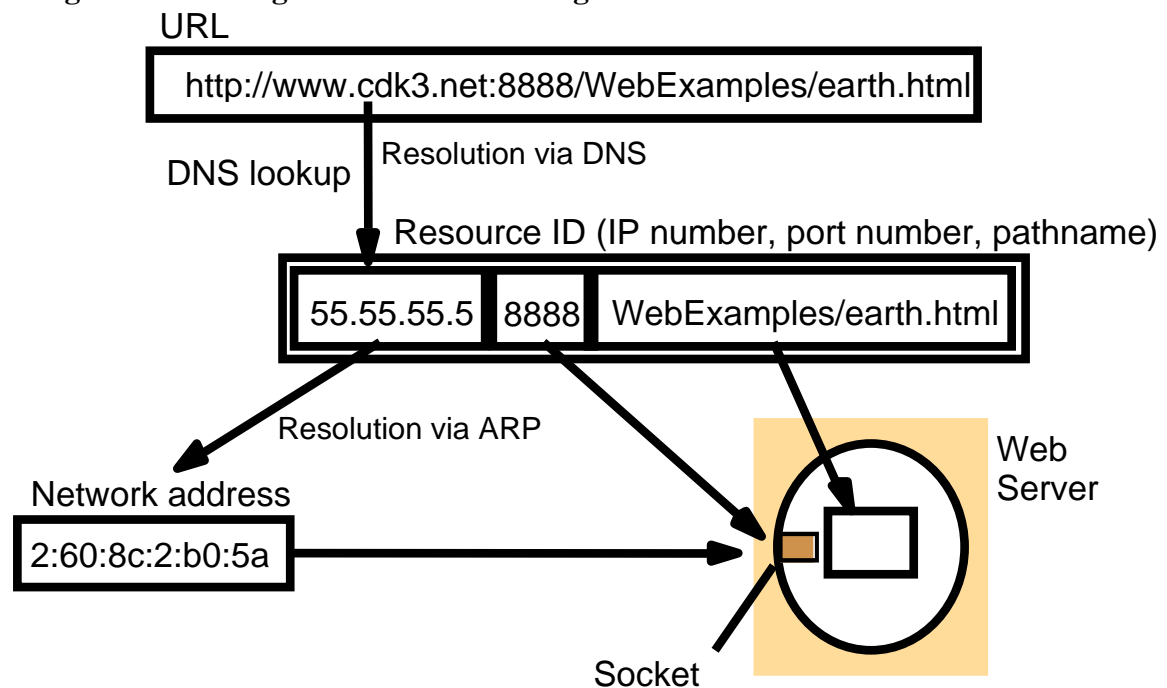
Contoh penamaan pada aplikasi sistem terdistribusi:

- URL untuk mengakses suatu halaman web.
- Alamat e-mail utk komunikasi antar pemakai.

Name Resolution, Binding, Attributes

- ♦ *Name resolution*:
 - Nama ditranslasikan ke data ttg *resource/object* tsb.
- ♦ *Binding*:
 - Asosiasi antara nama & obyek.
 - Biasanya nama diikat (*bound*) ke *attributes* dr suatu obyek.
- ♦ *Address*: atribut kunci dari sebuah entitas dalam sistem terdistribusi
- ♦ *Attribute*: nilai suatu *object property*.

Penguraian Naming Domains untuk mengakses *resource* dari URL



6.2 Tujuan Penamaan

1. Identifikasi
Seorang pemakai menginginkan obyek/layanan A, bukan obyek/layanan B.
2. Memungkinkan terjadinya *sharing*
Lebih dari satu pemakai dapat mengidentifikasi *resource* dengan nama yang sesuai (tidak harus nama yang sama).
3. Memungkinkan *location independence*:
Perubahan lokasi tidak menuntut perubahan nama, asalkan lokasi tidak menjadi bagian dari nama *resource* tsb.
4. Memberikan kemampuan keamanan (*security*)
Jika sebuah nama dipilih secara acak dari himpunan besar *interger*, maka nama tsb hanya bisa diketahui dari *legitimate source*, bukan dari menebak. Jadi jika seseorang mengetahui nama obyek tsb, maka dia memang diberitahu, karena sulit sekali menebak nama tsb.

6.3. Jenis Nama

1. *User names*:
 - Dibuat oleh pemakai (*user*).
 - Merujuk pada suatu obyek atau layanan.
 - Terdiri dari *strings of characters*.
 - Contoh: hp201 untuk pencetak, ~bettyp/tmp/test.c untuk berkas.
2. *System names*:
 - Terdiri dari *bit string*.
 - Internal untuk sistem, tidak ditujukan untuk manusia.
 - Lebih *compact* dari *user names*, sehingga dapat dibandingkan dengan lebih efisien.

6.4. Struktur Nama

1. *Primitive/flat names (Unique Identifiers = UIDs)*
 - Tanpa struktur internal, hanya *string of bits*.
 - Digunakan utk perbandingan dengan UID lain.
 - Tidak membawa informasi lain -> *pure names*.
 - Sangat berguna & banyak digunakan karena:
 - *Location & application independent*, sehingga tidak menjadi masalah bagi mobilitas obyek.
 - Seragam, *fixed size*.
 - *Compact*: mudah disimpan, di-pass, & jika cukup besar menjadi sulit ditebak.
2. *Partitioned Names (PN)*
 - Komposisi dari beberapa nama primitif, biasanya disusun secara hirarkis.
 - Contoh: telaga.cs.ui.ac.id, /cs/docs/akademik/SisDis/naming.ppt.
 - Membawa informasi -> *impure names*.
 - Biasanya tidak secara unik mengidentifikasi obyek, beberapa nama bisa dipetakan ke satu obyek (e.g. UNIX file links).
3. *Descriptive names (DN)*
 - Daftar atribut yang secara bersama-sama mengidentifikasi obyek secara unik.
 - Membawa informasi -> *impure names*.
 - DN adalah superset dari PN.

- E.g. OSI X.500 directory service.
 - *Directory Information Tree (DIT) X.500 name tree*
 - *Directory Information Base (DIB):*
The entire directory structure, including the data associated with the nodes.

6.5. Name Context

Nama selalu diasosiasikan dengan konteks, yang mendefinisikan di mana nama tsb *valid*. Ada 2 macam konteks:

- *Universal context:*
 - Di manapun nama digunakan, nama di-*resolved* dengan cara yang sama.
 - Dapat disalin dari mesin ke mesin dengan bebas.
 - Contoh: <http://www.cs.ui.ac.id/index.html>.
- *Relative context:*
 - *Context dependent.*
 - Contoh: 'a/b/c', 'b/c' *resolvable* pada konteks 'a'. Sedangkan pada node yang berbeda, 'a/b/c' dapat merujuk pada hal yang berbeda pula.

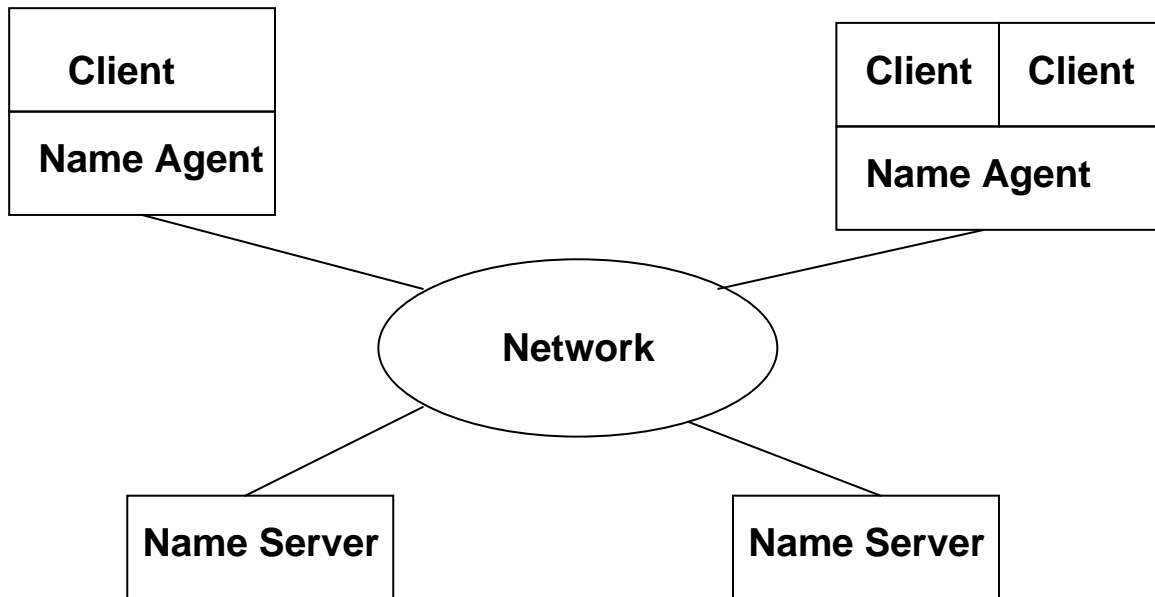
6.6. Sasaran Fasilitas Penamaan

1. Efisien, karena fasilitas penamaan merupakan dasar pada sistem terdistribusi & digunakan secara terus menerus.
2. Terdistribusi. Renungkan jika UIDs dibangkitkan oleh *centralized generator*.
 - *Bottleneck.*
 - Node tempat *generator* tsb mengalami kegagalan.
3. Tampak seperti *global space*, tidak tergantung konektifitas, topologi, dan lokasi obyek.
4. Mendukung pemetaan *1:many* antara nama & obyek, untuk memungkinkan *multicast*.
5. Mendukung *dynamic relocation of objects*, jika obyek/proses potensial untuk *mobile* (berpindah-pindah). Jadi diperlukan *dynamic binding* antara nama & alamat, juga antara alamat & rute.
6. Memungkinkan *local aliases*, shg pemakai dapat mengekspresikan interpretasi semantik mereka thdp suatu obyek. Tentu saja diperlukan pemetaan antara *aliases* dan *full names*.

6.7. Name List

Name Lists terdiri dari 2 komponen yaitu:

1. *Name agents*
2. *Name servers*



6.7.1. Name Agent

Name agents berada di *client*, bisa 1 *name agent* per *client* atau 1 *name agent* digunakan oleh beberapa *clients*. Name agents menjadi perantara antara *client* dan *name server*. Contoh: *resolver* pada *Domain Name Service* (DNS).

Fungsi *Name Agents*:

- memastikan bahwa lokasi *name servers* terlihat transparan bagi *client programs* (menyembunyikan lokasi *name server*).
- ‘berbicara’ dalam protokol komunikasi yang dimengerti *name server*.
- mengetahui bagaimana *name space* diatur, sehingga tahu kemana suatu *request* harus dikirim untuk memperoleh informasi lokasi.
- melakukan negosiasi kompatibilitas atau *availability* sumber daya (*resource*), berdasarkan atribut.

6.7.2. Name Server

Name server menggunakan basis data terdistribusi yang terdiri dari *tuples* <nama, lokasi, [atribut]>. Contoh atribut: jika *resource* adalah *printer*, maka atribut dapat menyatakan apakah obyek dapat melakukan pencetakan *postscripts* atau tidak.

Hal penting yang harus dimiliki:

- *Availability*,
- *Resilience to failure*,
- Konsistensi,
- Kecepatan menerima pengaruh perubahan *name lists*,
- Kemudahan mengkompilasi *list of objects* (*resources*).

Ada 2 jenis *Name Server*:

A. Primary Name Server

- Mendapatkan data dengan membaca file di storage
- Lebih dikenal dengan File Zone

B. Secondary Name Server

- Mendapatkan data dengan mereplikasikan data yang ada di primary server

- Lebih dikenal dengan Transfer Zone

6.7.3. Bentuk Name List

A. Name List Tersentralisasi

Adalah *Name list* yang berada pada satu mesin.

- **Kelebihan:**
 - Layanan cukup dilakukan dengan melihat *name lists*.
 - Waktu yang dibutuhkan antara registrasi obyek & saat obyek tsb dapat diakses, sangat singkat.
 - Mudah untuk memperoleh daftar obyek aktif.
- **Kekurangan:**
 - *Poor resilience*: jika node *crash*, terjadilah malapetaka.
 - Kemacetan (*congestion*) membatasi *availability*.

B. Name List Tereplikasi Penuh

Digunakan untuk mengatasi kekurangan *name list* tersentralisasi.

- **Masalah:**
 - WRITE:
 - Untuk menjaga konsistensi, jika *name list* direplikasi, maka setiap perubahan harus terefleksi di semua *copy*.
 - Bagaimana jika saat perubahan dicatat, ada sebagian replika yang tidak dapat dihubungi (*link or node failures*)?
 - READ:
 - Bagaimana jika informasi yang diperoleh ternyata sudah usang, atau ada beberapa replika yang tidak dapat diakses?
- **Solusi:**
 - Sebuah *name server* dipilih sebagai *master*, dan selalu merefleksikan secara akurat *state of the world*.
 - *Name servers* lainnya bertindak sbg pemberi petunjuk (*hint*), yang belum tentu benar.
 - Propagasi informasi antara master dan replika dilakukan saat 'sepi'.
- **Diperlukan beberapa asumsi yaitu**
 - Data penamaan tidak sering berubah, sehingga ketidakkonsistenan relatif jarang terjadi. Tergantung dari aplikasi, cukup akurat untuk *mail system* tapi tidak untuk sistem berbasis obyek yang sangat dinamis.
 - Jika dipakai data yang usang, maka akan terjadi *error* yang dapat diatasi. Contoh: Buku telepon yang memuat nomor telepon yang tidak terpakai lagi.
 - Tidak ada masalah jika dipakai data usang. Contoh: forward pada alamat *e-mail* yang lama.
- **Kelebihan:**
 - Tidak perlu suatu *central name server*, di mana seluruh *station* tergantung pada *name server* tsb.
 - Masih relatif mudah memperoleh daftar obyek dalam suatu jaringan, di mana suatu *name list* berisi informasi yang dibutuhkan.
 - *Availability* meningkat, shg lokasi obyek dapat ditemukan lebih cepat dari *name list* tersentralisasi.

- **Kekurangan:**
 - Menggunakan lebih banyak memori.
 - Potensial timbul masalah ketidakkonsistenan.
Pada beberapa jaringan, broadcast packet ke replika sangat meningkatkan *overhead* jaringan.

C. Name List Tereplikasi Sebagian

- Sebagian *name lists* disimpan dalam *cache* setiap mesin.
- Memerlukan mekanisme petunjuk (*hint*), yang biasanya benar.
- Tidak ada *master copy*, shg dapat timbul masalah:
 - Seberapa besar *cache*?
 - Manakah nama yang harus dihapus dari *cache* untuk menjaga konsistensi.
- Umum digunakan pada sistem berbasis obyek.
- UID merupakan nama obyek.
- Petunjuk lokasi disimpan dalam nama tsb, untuk menghindari seringnya berkonsultasi dengan *name server*.
- Petunjuk harus dapat diandalkan.
- Jika sebuah obyek berpindah, maka setiap *reference* harus diubah satu persatu.

6.8. Contoh Name Service

1. DNS (Domain Name Service)
 - memetakan nama domain ke alamat
2. GNS (Global Name Service)
 - memetakan global name ke atribut-atribut
 - skalabilitas, dapat menangani perubahan
3. X500 directory service
 - memetakan nama orang ke dalam alamat suatu e-mail dan nomor telepon
4. Jini discovery service
 - mencari objek sesuai dengan atribut yang ada