**Nama : Habib Maulana Shidiq**

**Nim : 2111082016**

**Kelas : 3A TRPL**

**RESUME PENAMAAN**

## **Penamaan(naming)**

1. Penamaan digunakan untuk mengidentifikasi entitas dalam sistem terdistribusi.
2. Ada tiga jenis referensi untuk mengidentifikasi entitas: Nama, Alamat, dan Pengenal.
3. Nama adalah sekelompok karakter atau bit yang mengidentifikasi entitas dan bisa bersifat human-friendly atau tidak.
4. Alamat adalah lokasi entitas yang bisa bersifat tergantung pada lokasi atau tidak.
5. Pengenal adalah nama yang unik yang mengidentifikasi entitas dan harus memenuhi properti tertentu.

## **Nama, Alamat & Pengenal**

1. Suatu entitas dapat dikenali dengan tiga jenis referensi:
2. **Nama**

Nama adalah sehimpunan bit atau karakter yang mereferensi/mengacu suatu entitas

Nama dapat bersifat human-friendly (atau tidak)

1. **Alamat (Address)**

Setiap entitas terletak pada suatu access point (titik akses), dan access point itu mempunyai alamat (address)

Address dapat bersifat location-dependent (atau tidak)

Contoh: IP Address + Port

1. **Pengenal (Identifier)**

Identifier adalah nama yang secara unik mengenali entitas

Suatu identifier sejati adalah nama dengan properti-properti berikut:

* Suatu identifier mengacu ke paling banyak satu entitas
* Setiap entitas dirujuk oleh paling banyak satu identifier
* Suatu identifier selalu merujuk ke entitas yang sama (tidak pernah digunakan-ulang)

## **Sistem Penamaan**

Sistem penamaan adalah middleware yang membantu dalam resolusi nama.

Dibagi menjadi tiga kelas: Penamaan Flat, Penamaan Terstruktur, dan Penamaan Berbasis Atribut.

## **Kelas – kelas penamaan**

* **Penamaan Flat** menggunakan identifier acak, dan ada empat mekanisme resolusi nama, termasuk broadcasting dan Distributed Hash Tables (DHTs).
* **Broadcasting** mengirimkan nama/alamat ke seluruh jaringan, yang memungkinkan entitas yang terkait dengan nama tersebut merespons.
* **Forwarding Pointers** memungkinkan penemuan entitas mobile dengan rantai pointer.
* **Pendekatan berbasis Home** memberikan setiap entitas sebuah node home dengan alamat tetap, dan entitas mobile memelihara alamat terkininya di sana.
* **Distributed Hash Tables (DHTs)** adalah sistem yang menyediakan layanan pencarian seperti hash table.

## **Chord:**

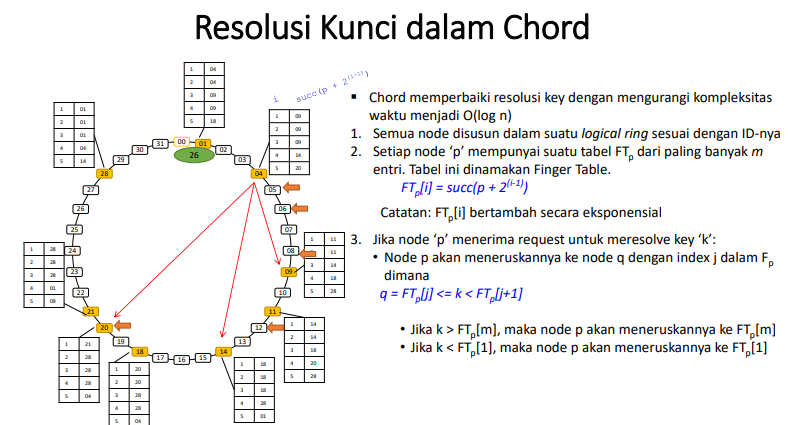
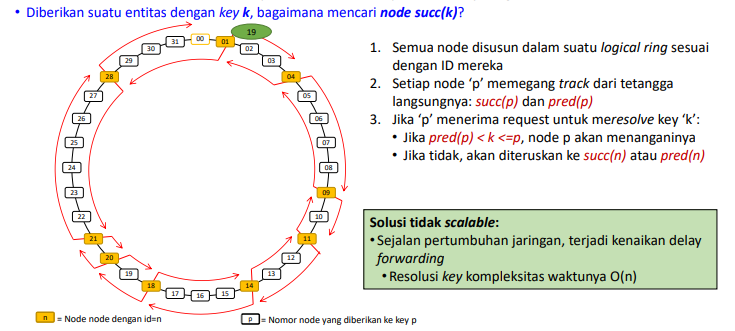
Chord menyematkan suatu m-bit identifier (dipilih secara acak) kepada setiap node

* Suatu node dapat dihubungi melalui alamat jaringannya
* ia memetakan setiap entitas ke suatu node
* Entitas dapat berupa proses, file, dll.,
* Pemetaan entitas ke node
* Setiap node bertanggungjawab untuk sehimpunan entitas
* Suatu entitas dengan key k jatuh di bawah dari node dengan pengenal terkecil id >= k. Node ini dikenal sebagai successor dari k, dan dinotasikan dengan succ(k

## **Algoritma Resolusi Kunci Naïve**

Salah satu pendekatan dalam kriptografi kunci publik yang digunakan untuk pertukaran kunci aman antara dua pihak. Pendekatan ini termasuk dalam kategori kriptografi kunci publik dan merupakan langkah awal dalam pemahaman lebih mendalam tentang algoritma pertukaran kunci yang lebih kompleks, seperti Diffie-Hellman atau RSA.

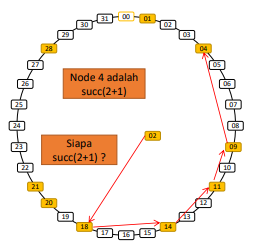
* Isu utama dalam DHT adalah efisiensi pemetaan suatu key k ke lokasi jaringan dari succ(k)
* Diberikan suatu entitas dengan key k, bagaimana mencari node succ(k)?



## **Chord: Protokol Join & Leave**

Chord adalah protokol overlay P2P yang memungkinkan node bergabung (Join) dan keluar (Leave). Dalam Join, node baru mencari entry node, mendapatkan tugas, dan merespons permintaan data. Dalam Leave, node yang keluar memberi tahu tetangga, meneruskan kunci, mempertahankan konsistensi, dan jaringan menyesuaikan diri. Ini memungkinkan pertumbuhan dan pengurangan jaringan sambil menjaga konsistensi dan efisiensi pencarian data.

* Jika suatu node p ingin bergabung (join): • Ia menghubungi arbitrary node, mencarikan succ(p+1), dan menyisipkan dirinya ke dalam ring tersebut
* Jika node p ingin meninggalkan (leave): • Ia menghubungi pred(p) dan succ(p+1) dan mengupdatenya.



## **Chord: Protokol Update Finger Table**

Chord: Protokol Update Finger Table adalah salah satu aspek penting dalam jaringan overlay seperti Chord, yang digunakan untuk mengoptimalkan pencarian data dalam lingkungan distribusi peer-to-peer. Berikut adalah intisari tentang protokol ini

1. **Finger Table**: Tabel jari yang mempercepat pencarian. Setiap simpul memiliki entri untuk simpul lain.
2. **Update Finger Table**: Mengupdate Finger Table saat simpul bergabung, keluar, atau menerima pembaruan.
3. **Join Operation**: Bergabungnya simpul baru dengan mengupdate Finger Table.
4. **Stabilisasi**: Menjaga konsistensi dan stabilitas jaringan dengan operasi berkala.
5. **Failure Handling**: Mengidentifikasi dan menghapus entri tidak valid saat simpul mengalami kegagalan.

Protokol ini penting untuk menjaga akurasi Finger Table dan efisiensi pencarian dalam jaringan Chord bahkan saat ada perubahan dinamis.

* Untuk suatu node q, FTq[1] harus up-to-date
* Ia merujuk ke next node dalam ring tersebut
* Protokol:
* Secara berkala, request succ(q+1) untuk return pred(succ(q+1))
* Jika q = pred(succ(q+1)), maka informasinya up-to-date
* Jika tidak, suatu node baru p telah ditambahkan ke ring sehingga q < p < succ(q+1)
* FTq[1] = p
* Request p untuk update pred(p) = q
* Dengan cara serupa, node p mengupdate setiap entri i dengan mencari

succ(p + 2(i-1))

## **Kedekatan Jaringan dalam Chord**

Organisasi logis dari node-node dalam jaringan overlay dapat mengakibatkan ketidak-efisienan transfer message

* Node k dan node succ(k +1) mungkin terpisah jauh

Chord dapat dioptimalkan dengan mempertimbangkan lokasi jaringan dari node-node

1. Penugasan Node Sadar Topologi

Dua node berdekatan memperoleh identifiers yang saling berdekatan

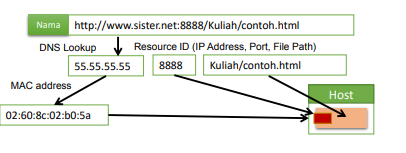
1. Routing kedekatan

* Setiap node q memelihara ‘r’ suksesor untuk entri ke-i dalam finger table
* FTq[i] skrg merujuk ke sebanyak r node suksesor di dlm rentang [p + 2(i-1), p + 2i-1)
* Untuk meneruskan request lookup, ambil 1 dari r suksesor terdekat ke node q tersebut.

## **Penamaan**

Nama digunakan untuk secara unik mengidentifikasi entitas-entitas di dalam sistem terdistribusi

* Entitas dapat berupa proses, remote objects, newsgroups, dll.,
* Nama dipetakan ke lokasi entitas menggunakan resolusi (name resolution)
* Contoh resolusi nama:

****

### **Penamaan Terstruktur**

Nama terstruktur tersusun dari nama-nama human-readable sederhana

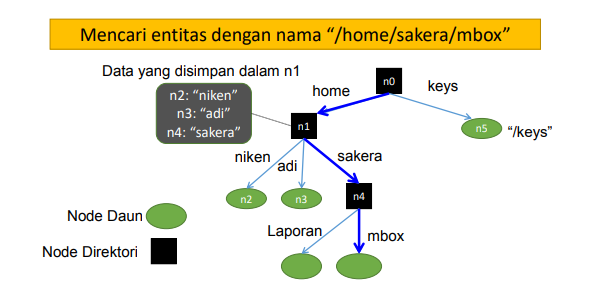
* Nama-nama diatur dalam suatu struktur tertentu
* Contoh:
* File-systems menggunakan nama terstruktur untuk mengidentifikasi file-file
* /home/userid/work/dist-systems/naming.txt
* Websites dapat diakses melalui nama terstruktur
* Husni.trunojoyo.ac.id

### **Ruang Nama**

Nama terstruktur diorganisir dalam ruang nama (name spaces). Name space adalah suatu directed graph yang terdiri dari:

* **Node daun (leaf)**
* Setiap node daun merepresentasikan suatu entitas
* Node daun biasanya menyimpan address dari suatu entitas (misalnya dalam DNS), atau state dari (atau path menuju) suatu entitas (misalnya di dalam file systems)
* **Node direktori**
* Node direktori merujuk ke node leaf atau direktori lain
* Setiap outgoing edge diwakili oleh (edge label, node identifier)
* **Setiap node dapat menyimpan tipe data tertentu**
* Yaitu State dan/atau address (misalnya untuk suatu mesin) dan/atau path.

Ruang nama terdistribusi memiliki tiga lapisan: global, administrasional, dan manajerial.



### **Resolusi Name**

Proses pencarian nama disebut resolusi (name resolution)

* Mekanisme pengakhiran (closure):
* Resolusi nama tidak dapat diselesaikan tanpa suatu initial directory node
* Mekanisme closure memilih konteks implisit dari mana memulai resolusi nama
* Contoh:
* husni.trunojoyo.ac.id: mulai pada DNS Server
* /home/sakera/mbox: mulai pada root dari file-system

### **Pengaitan Nama**

Pengaitan nama melibatkan hard links dan symbolic links untuk menghubungkan dua entitas yang berbeda dalam ruang nama.

* Dua jenis link dapat hadir antara node-node:

1. **Hard Links**

Hard links adalah beberapa entitas file yang berbagi data fisik yang sama di dalam sistem file

* Ada suatu directed link dari hard link (nama link) ke actual node (node sebenarnya)
* Resolusi nama:
  + Serupa dengan resolusi nama
* Aturan:
  + Harus tidak ada siklus didalam graf.

1. **Symbolic Links**

Symbolic Links (disingkat symlink) adalah jenis tautan dalam sistem berkas yang digunakan untuk membuat pintasan atau referensi ke file atau direktori lain.

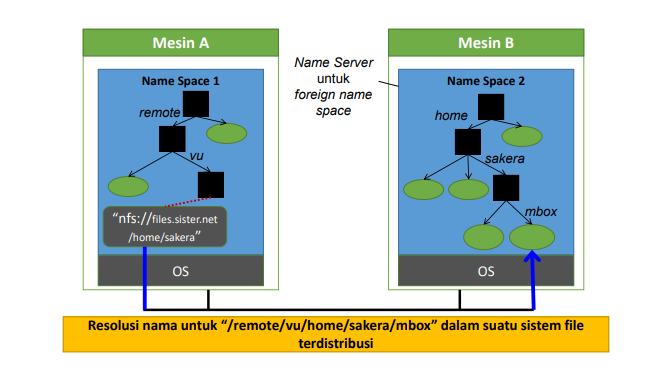
* Symbolic link menyimpan nama dari node asli sebagai data
* Resolusi nama bagi suatu symbolic link SL
  + Pertama, resolve nama SL
  + Baca isi dari SL
  + Resolusi nama berlanjut dengan isi dari SL
* Aturan:
  + Harus tidak muncul siklus referensi

### **Penggabungan Ruang Nama**

Ruang nama dapat digabungkan menggunakan teknik mounting, seperti yang dilakukan dalam Network File System (NFS).

* Network File System (NFS) adalah contoh dimana ruang nama berbeda digabungkan (mounted)
* NFS memungkinkan akses transparan ke file-file remote.

**Contoh Mounting Ruang Nama dalam NFS**



### **Ruang Nama Terdistribusi**

Ruang nama terdistribusi memiliki tiga lapisan: global, administrasional, dan manajerial.

* Mendistribusikan node-node dari graf penamaan
* Mendistribusikan manajemen ruang nama tersebut
* Mendistribusikan mekanisme resolusi nama.

1. **Layer dalam Ruang Nama Terdistribusi**

Ruang nama terdistribusi dapat dibagi ke dalam 3 (tiga) layer:

1. **Lapisan Global**

* Terdiri dari node-node direktori level tinggi
* Node-node direktori bersama-sama dikelola oleh administrasi berbeda

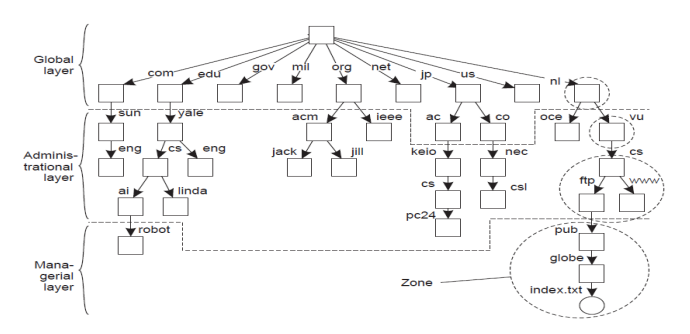
1. **Lapisan Administrasional**

* Mengandung node-node direktori level tengah.
* Node-node direktori dikelompokkan bersama-sama dengan cara tertentu yang setiap grup dikelola oleh suatu administrasi

1. **Lapisan Managerial**

* Berisi node-node direktori level bawah (rendah) di dalam suatu administrasi tunggal
* Persoalan utama adalah bagaimana secara efisien memetakan node-node direktori ke server nama lokal.

### **Ruang Nama Terdistribusi: Contoh**



## **Perbandingan Server Nama pada Layer Berbeda**



### **Resolusi Nama Terdistribusi**

Resolusi nama terdistribusi melibatkan mekanisme resolusi nama iteratif dan rekursif.

Resolusi nama terdistribusi bertanggungjawab untuk memetakan nama-nama ke alamat dalam suatu sistem dimana:

1. Server-server nama didistribusikan antar node-node yang berpartisipasi
2. Setiap server nama mempunyai suatu name resolver lokal.

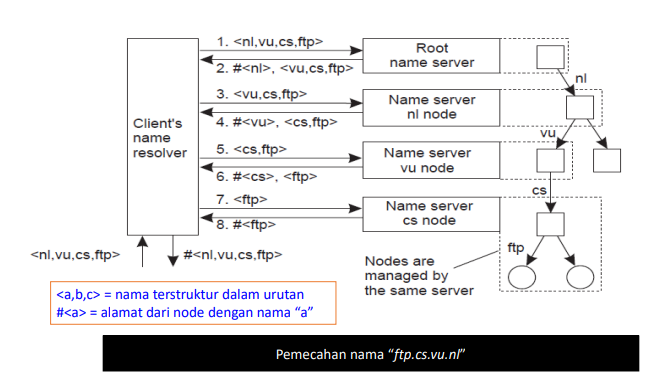
Akan dibahas dua algoritma resolusi nama terdistribusi:

* Resolusi nama iteratif
* Resolusi nama rekursif

### **Resolusi Nama Iteratif**

1. Client menyerahkan nama lengkap yang akan dipecahkan ke root name server
2. Root name server memecahkan (resolve) nama sejauh kemampuannya dan mengembalikan hasilnya kepada client
3. Root name server juga mengembalikan alamat dari server nama level selanjutnya (disingkat NLNS) jika alamat tidak terpecahkan secara lengkap
4. Client melewatkan bagian yang tidak terpecahkan dari nama ke NLNS
5. NLNS memecahkan nama sejauh kemampuannya dan mengembalikan hasilnya kepada client (bersama dengan next-level name server-nya)
6. Proses ini berlanjut sampai nama lengkap terpecahkan.

### **Resolusi Nama Iteratif: Contoh**



### **Resolusi Nama Rekursif**

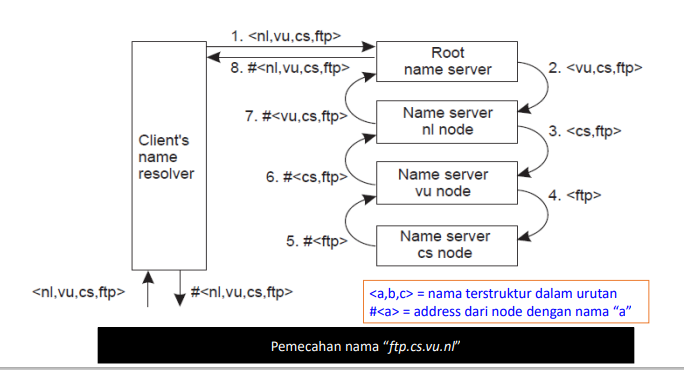
1. **Pendekatan:**

* Client menyerahkan nama yang akan dipecahkan kepada root name server
* Root name server tersebut melewatkan hasilnya ke next name server yang ditemukannya
* Proses berlanjut sampai nama tersebut secara lengkap terpecahkan

1. **Kekurangan:**

* Biaya besar pada server nama (terutama pada server nama tingkat tinggi)

### **Resolusi Nama Rekursif: Contoh**



## **Penamaan Berbasis Atribut**

Penamaan berbasis atribut menggunakan atribut-atribut entitas untuk mencari entitas. LDAP (Light-weight Directory Access Protocol) adalah contoh sistem yang menggunakan penamaan berbasis atribut. Dalam banyak kasus, adalah jauh lebih tepat menamai dan mencari entitas berdasarkan arti dari atribut-atributnya

* Mirip dengan layanan direktori tradisional (misal: yellow pages)
* Namun, operasi pencarian (look-up) dapat menjadi sangat mahal
* Harus mencocokkan nilai-nilai atribut yang direquest terhadap nilai-nilai atribut aktual yang mungkin memerlukan pemeriksaan semua entitas
* Solusi: Implementasikan layanan direktori dasar sebagai sebuah database, dan kombinasikan itu dengan sistem penamaan terstruktur tradisional
* Akan dibahas Light-weight Directory Access Protocol (LDAP); suatu sistem contoh yang menggunakan penamaan berbasis atribut

## **Light-weight Directory Access Protocol (LDAP)**

LDAP adalah protokol standar yang digunakan untuk mengakses dan mengelola informasi direktori. Direktori adalah kumpulan data terstruktur yang mengandung informasi seperti informasi pengguna, grup, perangkat, dan sumber daya lainnya

1. Layanan direktori LDAP terdiri dari sejumlah record bernama “directory entries”

* Setiap record tersusun dari pasangan (attribute, value)
* Standard LDAP menetapkan lima atribut untuk setiap record

1. Directory Information Base (DIB) adalah koleksi semua directory entries

* Setiap record dalam DIB bersifat unik
* Setiap record direpresentasikan oleh suatu nama yang membedakan

Misal: /C=NL/O=Vrije Universiteit/OU=Comp. Sc.

## **Pohon Informasi Direktori dalam LDAP**

* Semua record dalam DIB dapat ditata ke dalam suatu pohon hirarkis bernama Directory Information Tree (DIT)
* LDAP menyediakan mekanisme pencarian lanjut berdasarkan pada atribut dengan melintasi DIT tersebut
* Sintaks contoh untuk pencarian semua Main\_Servers di dalam Vrije Universiteit:

search("&(C = NL) (O = Vrije Universiteit) (OU = \*) (CN = Main server)")

