MODUL PRAKTIKUM



PEMROGRAMAN JARINGAN

UNIVERSITAS TEKNOLOGI MATARAM

2019

Jl. Kampus Universitas Teknologi Mataram, Kekalik, Mataram

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Konsep Dasar Jaringan

Abstract

Modul ini akan menjelaskan tentang konsep dasar jaringan.

Kompetensi

Mahasiswa dapat memahami konsep dasar jaringan.

Konsep Dasar Jaringan

Seringkali kita mendengar kata internet, sekilas mungkin kita akan berpikir bahwa yang namanya internet merupakan sebuah jaringan yang sangat besar dan terdiri dari banyak kompuer. Atau bahkan bagi orang yang awam internet sering diartikan sebagai browsing, chatting, dan lain-lain.

Pengertian ini merupakan sebuah pandangan yang kurang benar. Karena sebenarnya internet adalah kumpulan dari jaringan-jaringan kecil dan besar yang saling terhubung secara *real-time* atau terus menerus di seluruh dunia.

Dalam suatu sistem jaringan, dimana seluruh komputer saling berbagi data dan resources satu sama lain sehingga tercapai efisiensi dalam pemanfaatan teknologi, amat dibutuhkan perangkat-perangkat khusus dan instalasi tertentu.

Pada bab ini akan dijelaskan beberapa peralatan yang digunakan dalam sistem jaringan serta pengaturan TCP/IP pada sistem operasi Windows.

Topologi Jaringan

Tujuan dari suatu jaringan adalah menghubungkan jaringan-jaringan yang telah ada dalam jaringan tersebut sehingga informasi dapat ditransfer dari satu lokawi ke lokasi yang lain. Karena suat perusahaan memuliki keinginan atau kebutuhan yang berbeda-beda maka terdapat berbagai cara jaringan terminal-terminal dapat dihubungkan. Struktur Geometric ini disebut dengan LAN Topologies.

Terdapat 6 jenis topologi yaitu :

- Bus
- Ring
- Star
- Extended Star
- hierarchical topology
- Mesh

Setiap topologi memuliki karakteristik yang berdeda-beda dan masing-masing juga memiliki keuntungan dan kerugian. Topologi tidak tergantung kepada medianya dan setiap topologi biasanya menggunakan media sebagai berikut :

Jenis-jenis Media yaitu :

- Twisted Pair

- Coaxial Cable
- Optical Cable
- Wireless

Topologi dibagi menjadi dua jenis yaitu Physical Topology dan Logical Topologi. Dibawah ini adalah jenis-jenis Physical Topologi.

Topologi Bus atau Daisy Chain

Topologi ini memiliki karakteristik sebagai berikut:

• merupakan satu kabel yang kedua ujung nya ditutup, dimana sepanjang kabel terdapat node-node

• umum digunakan karena sederhana dalam instalasi

• signal melewati kabel dalam dua arah dan mungkin terjadi *collision*

• problem terbesar pada saat kabel putus. Jika salah satu segmen kabel putus, maka seluruh jaringan akan terhenti.

2. Topologi Ring

Topologi ini mempuyai karakteristik sebagai berikut:

• lingkaran tertutup yang berisi node-node

• sederhana dalam layout

• signal mengalir dalam satu arah, sehingga dapat menghindarkan terjadinya *collision* (dua paket data bercampur), sehingga memungkinkan pergerakan data yang cepat dan *collision detection* yang lebih sederhana

• problem: sama dengan topologi bus

• biasanya topologi ring tidak dibuat secara fisik melainkan direalisasikan dengan sebuah *consentrator* dan kelihatan seperti topologi star.

3. Topologi Star

Topologi ini mempunyai karakteristik sebagai berikut:

• setiap node berkomunikasi langsung dengan *central node*, traffic data mengalir dari node ke *central node* dan kembali lagi.

• mudah dikembangkan, karena setiap node hanya memiliki kabel yang langsung terhubung ke *central node.*

• keunggulannya adalah jika satu kabel node terputus yang lainnya tidak terganggu.

• dapat digunakan kabel yang "*lower grade*" karena hanya menghandel satu traffic node, biasanya digunakan kabel UTP.

Topologi Extended Star

Topologi Extended Star merupakan perkembangan lanjutan dari topologi star dimana karakteristiknya tidak jauh berbeda dengan topologi star yaitu :

• setiap *node* berkomunikasi langsung dengan *sub node*, sedangkan *sub node* berkomunikasi dengan *central node*. traffic data mengalir dari *node* ke *sub node* lalu diteruskan ke *central node* dan kembali lagi.

• Digunakan pada jaringan yang besar dan membutuhkan penghubung yang banyak atau melebihi dari kapasitas maksimal penghubung.

• keunggulan : jika satu kabel *sub node* terputus maka s*ub node* yang lainnya tidak terganggu, tetapi apabila *central node* terputus maka semua node disetiap *sub node* akan terputus

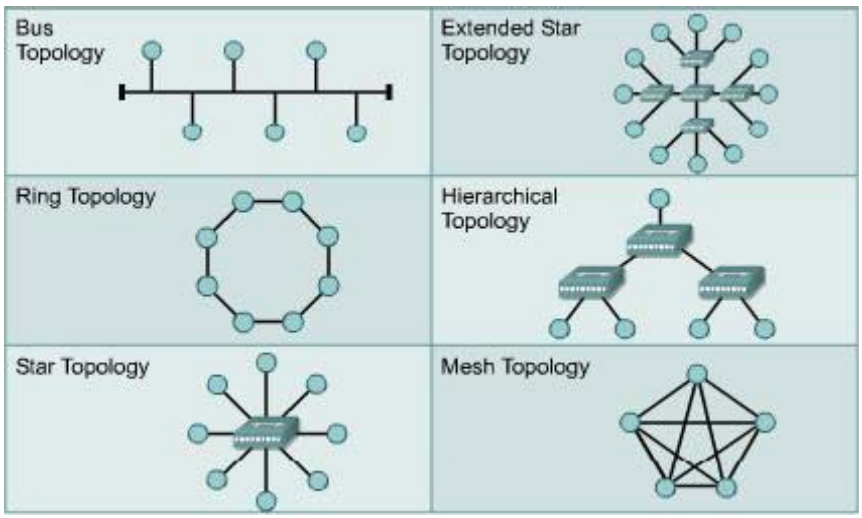
• tidak dapat digunakan kabel yang “lower grade” karena hanya menghandel satu traffic node, karena untuk berkomunikasi antara satu node ke node lainnya membutuhkan beberapa kali hops.

Topologi hierarchical

Topologi ini biasa disebut sebagai topolodi tree. Dibangun oleh seperti halnya topologi extended star yang dihubungkan melalui sub node dalam satu central node. Topologi ini dapat mensupport baik baseband maupun broadband signaling dan juga mensupport baik contention maupun token bus access.

6. Topologi Mesh

MESH topologi dibangun dengan memasang link diantara atation-station. Sebuah ‘fully-connected mesh’ adalah sebauh jaringan dimana setiap terminal terhubung secara langsung ke semua terminal-terminal yang lain. Biasanya digunakan pada jaringan komputer kecil. Topologi ini secara teori memungkinkan akan tetapi tidak praktis dan biayanya cukup tinggi untuk di-implementasikan. Mesh topologi memiliki tingkat redundancy yang tinggi. Sehingga jika terdapat satu link yang rusak maka suatu station dapat mencari link yang lainnya.



Gambar 34 Jenis-jenis topologi

Sedangkan Logical Topology adalah FDDI, Token Ring, dan Ethernet.

Tipe Jaringan

Dalam jaringan terdapat tiga buah peran yang dijalankan. Yang pertama adalah client. Peran ini hanya sebatas pengguna tetapi tidak menyediakan sumber daya (sharing), informasi, dan lain-lain. Peran kedua adalah sebagai peer, yaitu client yang menyediakan sumber daya untuk dibagi kepada client lain sekaligus memakai sumber daya yang tersedia pada client yang lain (peer to peer). Sedangkan peran yang terakhir adalah sebagai server, yaitu menyediakan sumber daya secara maksimal untuk digunakan oleh client tetapi tidak memakai sumber daya yang disediakan oleh client. Dibawah ini akan dijelaskan jenis-jenis jaringan yang ada.

Jaringan Berbasis Server

Jaringan berbasis server atau client-server diartikan dengan adanya server didalam sebuah jaringan yang menyediakan mekanisme pengamanan dan pengelolaan jaringan tersebut. Jaringan ini terdiri dari banyak client dari satu atau lebih server. Client juga biasa disebut front-end meminta layanan seperti penyimpanan dan pencetakan data ke printer jaringan, sedangkan server yang sering disebut back-end menyampaikan permintaan tersebut ke tujuan yang tepat.

Pada Windows NT, Windows 2000, dan Windows Server 2003, jaringan berbasis server diorganisasikan di dalam domain-domain. Domain adalah koleksi jaringan dan client yang saling berbagi informasi. Keamanan domain dan perizinan log on dikendalikan oleh server khusus yang disebut domain controlle. Terdapat satu pengendali domain utama atau Primary Domain Controller (PDC) dan beberapa domain controller pendukung atau backup Domain Controller (BDC) yang membantu PDC pada waktu-waktu sibuk atau pada saat PDC tidak berfungsi karena alasan tertentu.

Primasry Domain Controller juga diterapkan di dalam jaringan yang menggunakan server Linux. Software yang cukup andal menangani masalah ini adalah samba yang sekaligus dapat digunakan sebagai penyedia layanan file dan print yang membuat computer Windows dapat mengakses file-file di mesin Linux dan begitu pula sebaliknya.

Jaringan berbasis server memiliki beberapa keuntungan diantaranya adalah :

1. Media penyimpanan data yang terpusat memungkinkan semua user menyimpan dan menggunakan data di server dan memberikan kemudahan melakukan back-up data di saat kritis. Pemeliharaan data juga menjadi lebih mudah karena data tidak tersebar di beberapa computer.
2. Kemampuan server untuk menyatukan media penyimpanan di satu tempat akan menekan biaya pembangunan jaringan. Server yang telah dioptimalkan membuat jaringan berjalan lebih cepat daripada jaringan peer-to-peer. Membebaskan user dari pekerjaan mengelola jaringan.

3. Kemudahan mengatur jumlah pengguna yang banyak. Kemampuan untuk sharing peralatan mahal seperti printer laser. Mengurangi masalah keamanan karena pengguna harus memasukkan password untuk setiap peralatan jaringan yang akan digunakan.

2. Jaringan Peer-to-peer

Setiap computer di dalam jaringan peer mempunyai fungsi yang sama dan dapat berkomunikasi dengan computer lain yang telah memberi izin. Jadi, secara sederhana setiap komputer pada jaringan peer berfungsi sebagai client dan server sekaligus. Jaringan peer digunakan di sebuah kantor kecil dengan jumlah computer sedikit, dibawah sepuluh workstation.

Keuntungan menggunakan jaringan peer adalah :

1. Tidak memerlukan investasi tambahan untuk pembelian hardware dan software server.
2. Tidak diperlukan seorang network administrator dan setupnya mudah serta meminta biaya yang murah.

Kerugian menggunakan jaringan peer adalah :

1. Sharing sumberdaya pada suatu komputer didalam jaringan akan sangat membebani computer tersebut.
2. Masalah lain adalah kesulitan dalam mengatur file-file. User harus menangani komputernya sendiri jika ditemui masalah keamanan sangat lemah.

3. Jaringan Hybrid

Jaringan hybrid memiliki semua yang terdapat pada tiga tipe jaringan di atas. Ini berarti pengguna dalam jaringan dapat mengakses sumber daya yang dishare oleh jaringan peer, sedangkan di waktu bersamaan juga dapat memanfaatkan seumber daya yang disediakan oleh server.

Keuntungan jaringan hybrid adalah sama dengan keuntungan menggunakan jaringan berbasis server dan berbasis peer. Jaringan hybrid memiliki kekurangan seperti pada jaringan berbasis server.

- Peralatan Jaringan

Ada beberapa peralatan yang digunakan dalam jaringan, peralatan ini sering digunakan di dalam perkantoran dan perusahan besar. Peralatan ini adalah :

1. Network Interface Card

Dalam memilih network interface card, ada beberapa pertimbangan yang harus diperhatikan. Pertimbangan-pertimbangan ini sangat penting untuk diperhatikan, yaitu :

- Tipe jaringan seperti Ethernet LANs, Token Ring, atau Fiber Distributed Data Interface (FDDI).
- Tipe Media seperti Twisted Pair, Coaxial, Fiber-Optic, dan Wireless.

- Tipe Bus seperti ISA dan PCI



Gambar 35 Network Interface Card

2. PCMCIA Network Interface Card

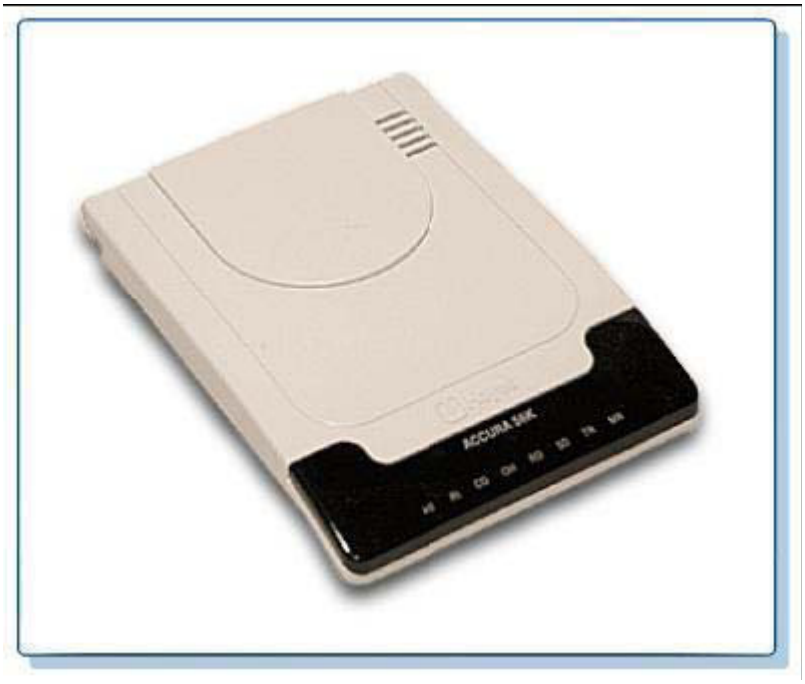
PCMCIA card adalah card jaringan yang digunakan untuk terhubung kedalam sebuah jaringan tanpa menggunakan kabel.



Gambar 36 PCMCIA Network Interface Card

3. Modem

Modem atau Modul the Modulator adalah peralatan jaringan yang digunakan untuk terhubung ke jaringan internet menggunakan kabel telepon.



Gambar 37 PCMCIA Network Interface Card

4. HUB/Switch

HUB atau Switch digunakan untuk menghubungkan setiap node dalam jaringan LAN. Peralatan ini sering digunakan pada topologi star dan extended star. Perbedaan antara HUB dan Switch adalah kecepatan transfer datanya. Yaitu 10:100 Mbps.



Gambar 38 HUB 8 Port dan Switch 24 Port

5. Bridge

Bridge adalah peralatan jaringan yang digunakan untuk memperluas ata memecah jaringan. Bridge berfungsi untuk menghubungkan dan menggabungkan media jaringan yang tidak sama seperti kabel *unshielded twisted pair* (UTP) dan kabel fiber-optic, dan untuk menggabungkan arsitektur jaringan yang berbeda seperti Token Ring dan Ethernet. Bridge me*regenerate* sinyal tetapi tidak melakukan konversi protocol, jadi protocol jaringan yang sama (seperti TCP/IP) harus berjalan

kepada kedua segemen jaringan yang terkoneksi ke bridge. Bridge dapat juga mendukung *Simple Network Management Protocol* (SNMP), serta memiliki kemampuan diagnosa jaringan.

Bridge hadir dalam tiga tipe dasar yaitu *Local, Remote,* dan *Wireless*. Bridge local secara langsung menghubungkan Local Area Network (LAN). Bridge remote yang dapat digunakan untuk membuat sebuah Wide Area Network (WAN) menghubungkan dua atau lebih LAN. Sedangkan wireless bridge dapat digunakan untuk menggabungkan LAN atau menghubungkan mesin-mesin yang jauh ke suatu LAN.

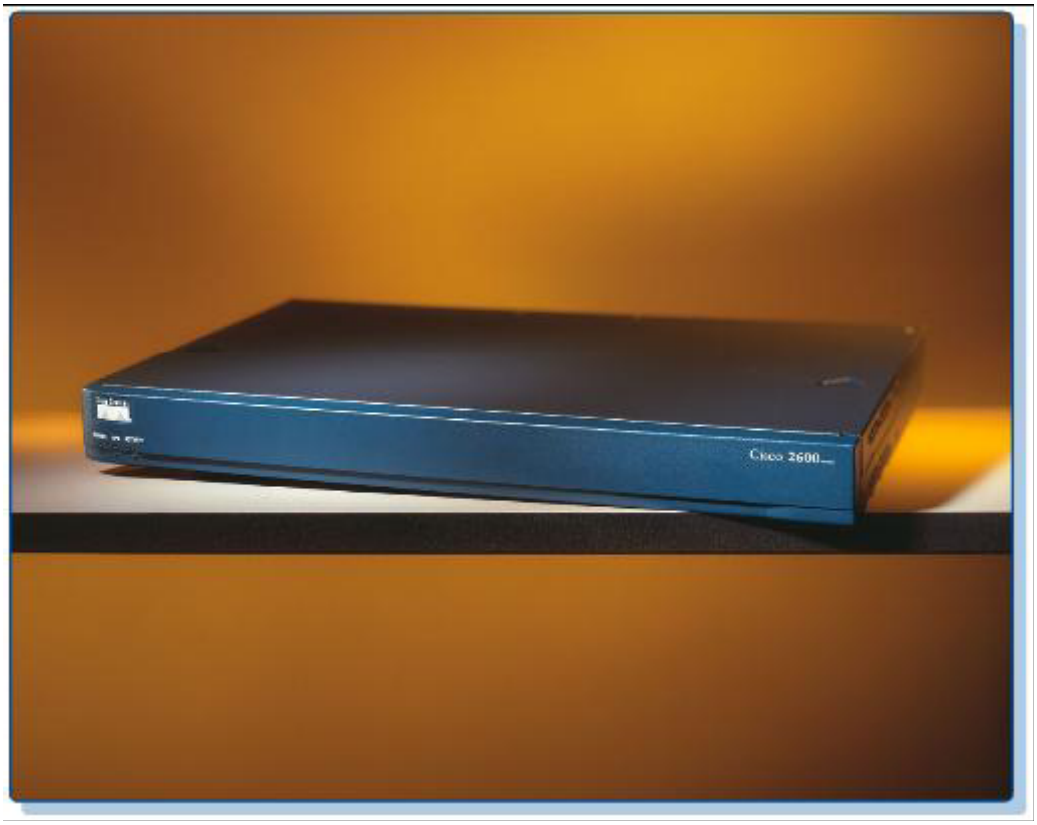
Bridge beroperasi mengenali alamat MAC address node asal yang mentransmisi data ke jaringan dan secara automatis membangun sebuah table routing internal. Table ini digunakan untuk menentukan ke segmen mana paket akan di route dan menyediakan kemampuan penyaringan (filtering). Setelah mengetahui ke segmen mana suatu paket hendak disampaikan, bridge akan melanjutkan pengiriman paket secara langsung ke segmen tersebut. Jika bride tidak mengenali alamat tujuan paket, maka paket akan di forward ke semua segmen yang terkoneksi kecuali segmen alamat asalanya. Dan jika alamat tujuan berada dalam segmen yang sama dengan alamat asal, bridge akan menolak paket. Bridge juga melanjutkan paket-paket broadcast ke semua segmen kecuali segmen asalnya.



Gambar 39 Wireless Bridge

6. Router

Router adalah peralatan jaringan yang digunakan untuk memperluas atau memecah jaringan dengan melanjutkan paket-paket dari satu jaringan logika ke jaringan yang lain. Router banyak digunakan di dalam internetwork yang besar menggunakan keluarga protocol TCP/IP dan untuk menghubungkan semua host TCP/IP dan Local Area Network (LAN) ke internet menggunakan *dedicated leased line*. Saat ini, masih banyak perusahaan menggunakan router Cisco 2500 series untuk mengkoneksikan dua buah LAN (WAN dengan anggota dua LAN), LAN ke ISP (Internet Service Provider). Koneksi seperti ini menyebabkan semua workstation dapat terkoneksi ke internet selama 24 jam. Router berisi table-tabel informasi internal yang disebut label routering yang melakukan pencatatan terhadap semua alamat jaringan yang diketahui dan lintasan yang mungkin dilalui. Router membuat jalur paket-paket berdasarkan lintasan yang tersedia dan waktu tempuhnya. Karena menggunakan alamat paket jaringan tujuan, router bekerja hanya jika protocol yang dikonfigurasi adalah protocol yang routetable seperti TCP/IP atau atau IPX/SPX. Ini berbeda dengan bridge yang bersifat protocol independent.



Gambar 40 Cisco Router 2600 series

7. Crimping Tools

Crimping tools berguna untuk memotong, merapikan dan mengunci kabel UTP dalam melakukan instalasi Networking.



Gambar 41 Crimping Tools

Pengkabelan

1. Kupas lapisan luar kabel UTP sepanjang ± 1 Cm dari ujung, sehingga 8 urat kabel terlihat dari luar.
2. Susun urutan warna kabel sesuai dengan standard internasional

Digunakan untuk memotong dan Digunakan untuk mengupas

Gambar	**Nomor kaki (pin)**	Nama Warna
1	Putih orange	
2	Orange	
3	Putih hijau	
4	Biru	
5	Putih biru	
6	Hijau	
7	Putih coklat	
8	Coklat	



Gambar 35 Susunan kabel straight

Nomor kaki (pin)	**Nama Warna**
1	Putih hijau
2	Hijau
3	Putih orange
4	Biru
5	Putih biru
6	Orange
7	Putih coklat

8	Coklat



Gambar 36 Susunan kabel cross

3. Masukkan Ujung kabel UTP yang telah disusun menurut urutan internasional, kemudian jepit dengan menggunakan *crimping tool*.



Gambar 37 Memasukkan Kabel UTP ke dalam RJ-45



Gambar 38 Menjepit kabel menggunakan Crimping



Gambar 39 Hasil Crimping kabel yang baik

4. Pasang satu sisi RJ-45 ke dalam Network Card, dan sisi lainnya ke HUB/Switch

5. Jaringan siap dioperasikan

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

IP Address

Abstract

Modul ini akan menjelaskan tentang IP Address.

Kompetensi

Mahasiswa dapat memahami IP Address.

IP Address

IP Address adalah sebuah alamat logical yang digunakan untuk menghubungkan antara satu komputer dengan komputer lainnya. IP Address saat ini terdiri dari dua jenis (versi), yang pertama adalah IP V4, dan IP V6. Untuk modul ini, akan dibahas IP V4 saja, untuk IP V6 akan dibahas pada modul berikutnya.

IP V4 terdiri dari 32 bit, contohnya sebagai berikut:

IP address dengan alamat 192.168.1.1 jika setiap decimal diubah menjadi bilangan binary maka akan menjadi 11000000.10101000.00000001.00000001

Kelas di dalam IP V4

IP V4 terdiri dari beberapa kelas, di antaranya adalah:

Kelas A

Kelas A dimulai dari 0 – 127

Subnet mask yang digunakan: 255.0.0.0

Contoh 10.1.2.3

Subnet mask menentukan jumlah host pada IP Address.

Jumlah host pada kelas A adalah jumlah bit 0 pada subnet mask kelas A

255.0.0.0 = 11111111.00000000.00000000.00000000

Jadi, jumlah host pada kelas A = 2^{24} = 16 juta

Kelas B

Kelas B dimulai dari 128-191

Subnet mask yang digunakan adalah 255.255.0.0

Contoh: 128.1.2.3

Jumlah host pada kelas B, adalah jumlah bit 0 pada subnet mask kelas B.

255.255.0.0 = 11111111.11111111.00000000.00000000

Jadi, jumlah host pada kelas B = 2^{16} = 16 ribu.

Kelas C

Kelas C dimulai dari 192 – 223

Subnet mask 255.255.255.0

Contoh: 192.168.1.1

Jumlah host pada kelas C, adalah jumlah bit 0 pada subnet mask kelas C.

255.255.255.0 = 11111111.11111111.11111111.00000000

Jumlah host = 2^8 = 256

Kelas D dan E

Digunakan sebagai percobaan.

Kelas D dimulai dari 224 -239.

Kelas E dimulai dari 240 – 255.

Namun dalam praktiknya semua IP dapat digunakan dengan permainan subnetting.

IP yang tidak boleh digunakan untuk jaringan

Alamat Loopback

Alamat loopback untuk tiap komputer umumnya memiliki IP yang sama yaitu 127.x.x.x artinya dari 127.0.0.0 sampai 127.255.255.255.

Alamat Global Address

Adalah alamat yang digunakan sementara untuk merequest IP Address dari sebuah server. Yaitu 0.x.x.x.

Alamat broadcast

Alamat yang digunakan untuk mengirimkan data kesemua jaringan secara bersamaan. Alamat tersebut 255.x.x.x.

IP Private dan IP Public

IP Private

IP yang boleh digunakan untuk konfigurasi jaringan LAN (Local Area Network) tapi tidak boleh digunakan untuk jaringan internet.

Contoh IP Private:

192.168.0.0 – 192.168.255.255

10.0.0.0 – 10.255.255.255.255

172.16.0.0 – 172.31.255.255

IP Public

IP address yang dapat diakses melalui internet. IP public biasanya hanya disediaakan oleh Internet Service Provider (ISP).

Aplikasi mendeteksi IP

Aplikasi ini bisa digunakan untuk mendeteksi IP dari sebuah host name yang diberikan. Host name yang diinput bisa seperti detik.com, google.co.id, atau local host. Aplikasi akan memproses pencarian IP berdasarkan host yang diinput, kemudian outputnya adalah IP dari host yang diinput di awal.

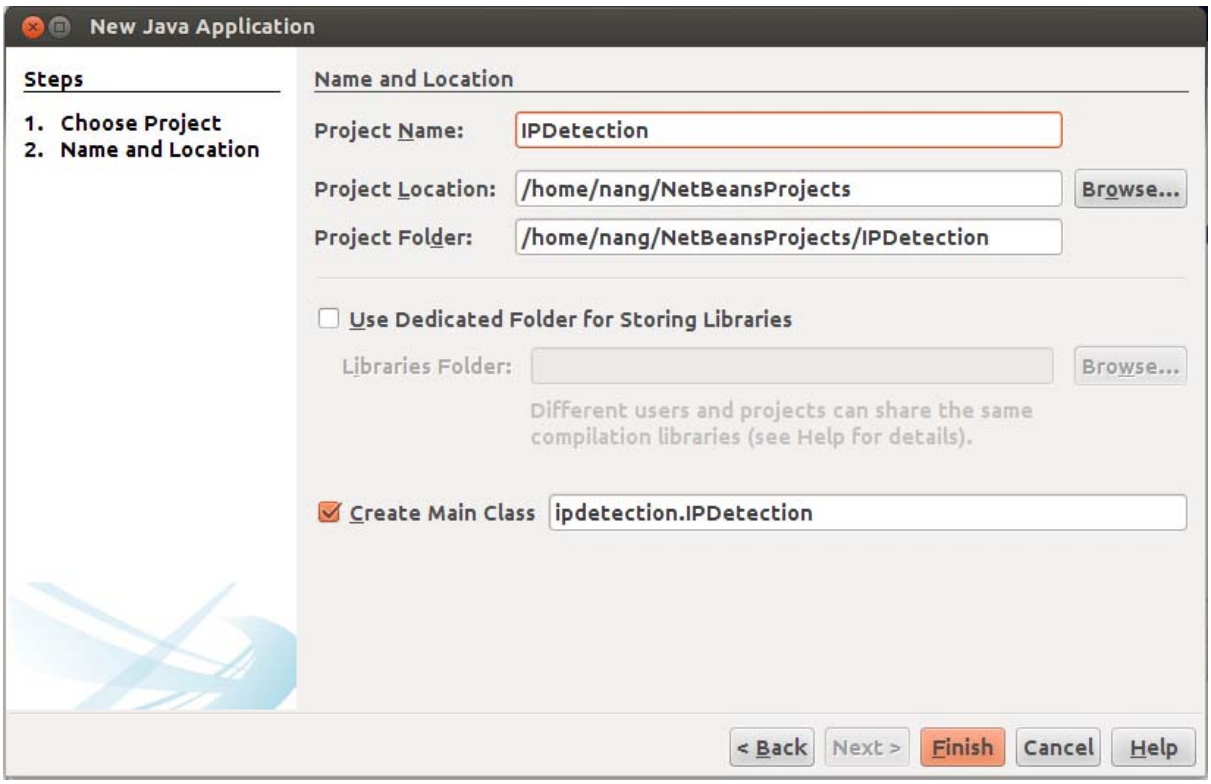
Tujuan pembuatan aplikasi

Mampu memahami cara mendapatkan IP Address sebuah host menggunakan pemrograman java.

Tahapan-tahapan pembuatan aplikasi adalah sebagai berikut:

Buka netbeans

Buat sebuah projek baru dengan nama IPDetection.

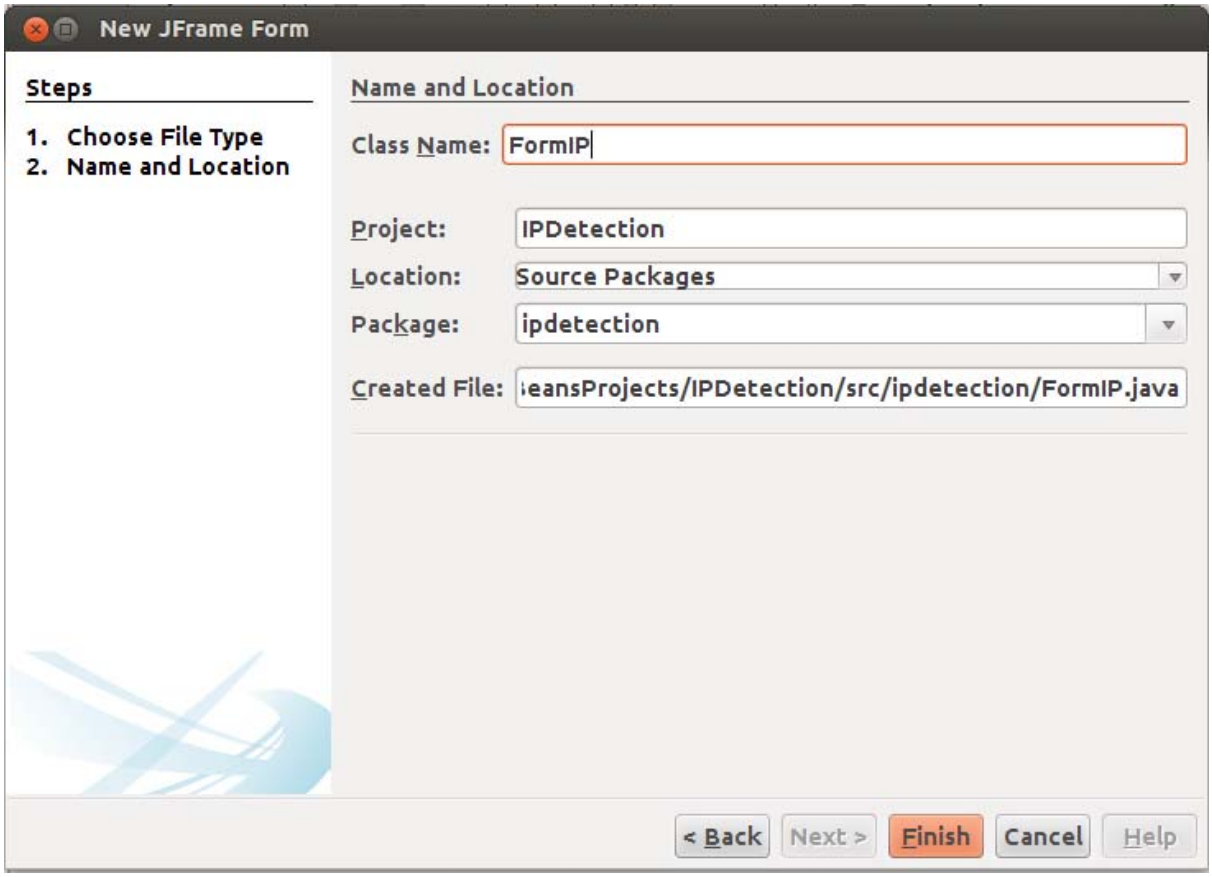


Kemudian buat sebuah JFrame form.

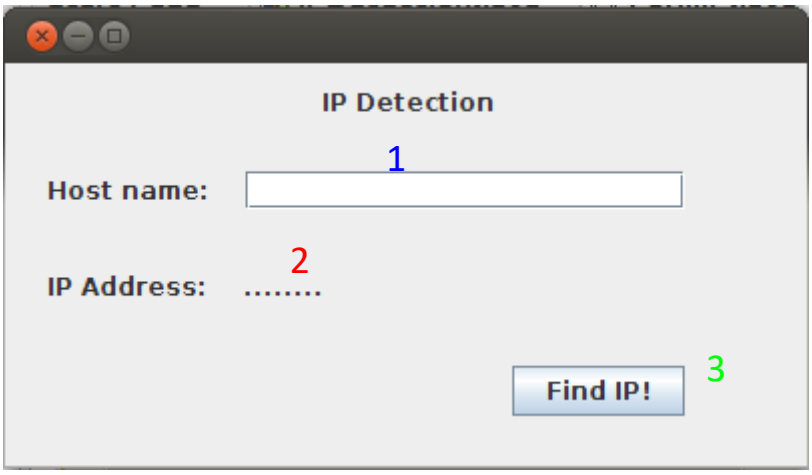
Klik kanan project

Pilih “new” > pilih “JFrame From ...”

Jika tidak ada pilih “Other” > pilih “Swing Gui Forms” > pilih “JFrame Form”.



Tambahkan



Komponen bernomor perlu diperhatikan sebagai berikut:

Variable name: jTextFieldIP

Text: kosong

Variable name: jLabelIP

Text: …....

Variable name: jButtonFind

Text: Find IP!

Buat sebuah event pada jButtonFind, dengan cara sebagai berikut:

Klik kanan jButtonFind

Pilih “Events” > pilih “Action” > pilih “actionPerformed”.

Akan muncul potongan program seperti berikut:

```
private void jButtonFindActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    InetAddress address;
    String host = jTextFieldIP.getText();
    try {
        address = InetAddress.getByName(host);
        jLabelIP.setText(address.toString());
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

Tambahkan kode program seperti berikut pada method yang baru saja dibuat.

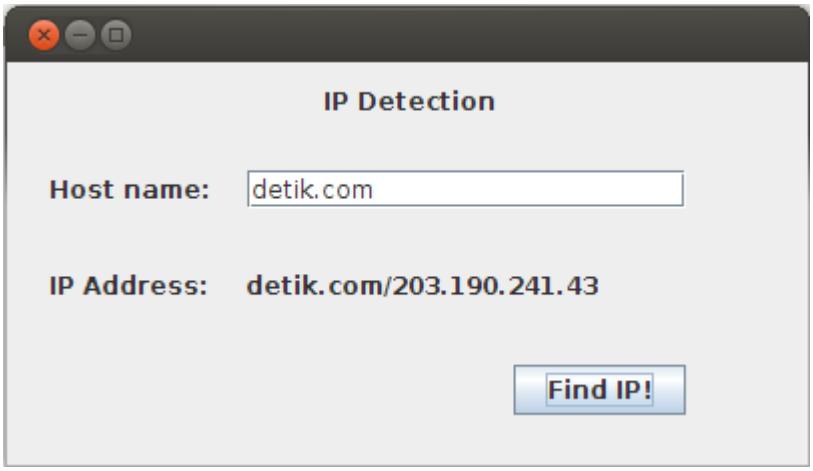
```
private void jButtonFindActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    InetAddress address;
    String host = jTextFieldIP.getText();
    try {
        address = InetAddress.getByName(host);
        jLabelIP.setText(address.toString());
    } catch(Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null,
                "host tidak di temukan / tidak ada jaringan internet",
                "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Kemudian panggil FormIP dari class utama, IPDetection, dengan cara membuat objek dari class FormIP.

Untuk melakukannya tambahkan kode program berikut pada IPDetection.java.

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new FormIP().setVisible(true);
}
```

Output dari program adalah sebagai berikut:



Pengalokasian IP address

IP Address terdiri atas dua bagian yaitu *network ID* dan *host ID*. Network ID menunjukkan nomor network, sedangkan host ID mengidentifkasikan host dalam satu network. Pengalokasian IP address pada dasarnya ialah proses memilih network ID dan host ID yang tepat untuk suatu jaringan. Tepat atau tidaknya konfigurasi ini tergantung dari tujuan yang hendak dicapai, yaitu mengalokasikan IP address se-efisien mungkin.

Terdapat beberapa aturan dasar dalam menentukan network ID dan host ID yang hendak digunakan. Aturan tersebut adalah :

Network ID 127.0.0.1 tidak dapat digunakan karena ia secara default digunakan dalam keperluan 'loop-back'. ('Loop-Back' adalah IP address yang digunakan komputer untuk menunjukan dirinya sendiri).

Host ID tidak boleh semua bitnya diset 1 (contoh klas A: 126.255.255.255), karena akan diartikan sebagai alamat broadcast. ID broadcast merupakan alamat yang mewakili seluruh anggota jaringan. Pengiriman paket ke alamat ini akan menyebabkan paket ini didengarkan oleh seluruh anggota network tersebut.

Network ID dan host ID tidak boleh sama dengan 0 (seluruh bit diset 0 seperti 0.0.0.0), Karena IP address dengan host ID 0 diartikan sebagai alamat network. Alamat network adalah alamat yang digunakan untuk menunjuk suatu jaringan, dan tidak menunjukan suatu host.

Host ID harus unik dalam suatu network (dalam satu network, tidak boleh ada dua host dengan host ID yang sama).

Aturan lain yang menjadi panduan network engineering dalam menetapkan IP Address yang dipergunakan dalam jaringan lokal adalah sebagai berikut:

0.0.0.0/8 → 0.0.0.1 **s.d.** 0.255.255.254 Hosts/Net: 16.777.214

10.0.0.0/8 → 10.0.0.1 **s.d.** 10.255.255.254 Hosts/Net: 16.777.214

127.0.0.0/8 → 127.0.0.1 **s.d.** 127.255.255.254 Hosts/Net: 16.777.214

172.16.0.0/12 → 172.16.0.1 **s.d.**172.31.255.254 Hosts/Net: 1.048.574 (Private Internet)

192.0.2.0/24 → 192.0.2.1 **s.d.** 192.0.2.254 Hosts/Net: 254

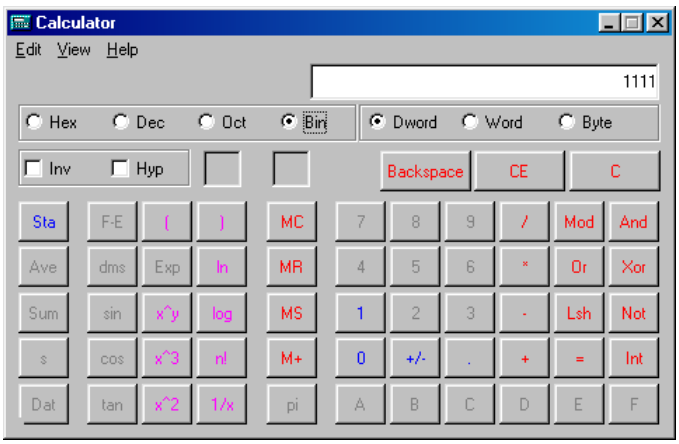
192.168.0.0/16→ 192.168.0.1 **s.d.**192.168.255.254 Hosts/Net: 65.534 (Private Internet)
169.254.0.0/16→ 169.254.0.1 **s.d.**169.254.255.254 Hosts/Net: 65.534
dan semua space dari klas D dan E dapat digunakan untuk IP Address local area network, karena IP ini tidak digunakan (di publish) di internet.

IP address, subnet mask, broadcast address merupakan dasar dari teknik routing di Internet. Untuk memahami ini semua kemampuan matematika khususnya matematika boolean, atau matematika binary akan sangat membantu memahami konsep routing Internet.

Mungkin pertanyaan seperti berikut pernah akan terlontar oleh anda:
Mengapa kita memilih IP address 192.168.1.5?
Mengapa subnet mask yang digunakan 255.255.255.0?
Mengapa bukan angka lain?
Mengapa network address 172.16.0.0?
Mengapa broadcast address-nya 202.159.32.15?
Bagaimana menentukan semua alamat-alamat tersebut? dan sebagainya.
Hal tersebut yang akan coba dijelaskan secara sederhana dalam uraian berikut, anda bisa juga mencobanya dengan komputer dirumah atau di rental. Alat bantu yang dibutuhkan cuma **(calculator scientific)**.

Untuk memudahkan kehidupan anda, ada baiknya memanfaatkan teknologi secara maksimal (jangan sampai gaptec ☺), contohnya menggunakan fasilitas kalkulator yang ada di Windows98 atau Win2000 juga WinXP, dapat diakses melalui **Start → Programs → Accessories → Calculator.**

Kalkulator yang standar memang sulit digunakan untuk membantu kalkulasi biner, oleh karena itu pilih **View → Scientific** untuk memperoleh tampilan kalkulator *scientific* yang dapat digunakan untuk perhitungan biner, seperti gambar berikut.



Gambar 4.2. Calculator Scientific (Win98)

Dengan cara memindahkan mode operasi ke bin, maka nilai yang ada akan berubah menjadi *binary*. Pada gambar contoh diperlihatkan nilai awal 15 desimal, dipindahkan menjadi 1111 binary.

Mengenal Aljabar Boolean

Aljabar Boolean adalah teknik menghitung dalam bilangan binary seperti 101010111. Proses konversi dari desimal ke binary sudah tidak perlu kita pikirkan lagi karena sudah dibantu menggunakan kalkulator yang ada di SO Windows.

Dari sekian banyak fungsi yang ada di aljabar boolean, seperti *and, or, xor, not* dan lain-lain, untuk keperluan teknik routing di Internet, kita hanya memerlukan fungsi "dan" atau "and." Contoh:

1 and 1 = 1

1 and 0 = 0

0 and 1 = 0

0 and 0 = 0

atau yang lebih kompleks:

11001010.10011111.00010111.00101101

di AND dengan

11111111.11111111.11111111.00000000

menjadi

11001010.10011111.00010111.00000000

Tidak percaya?

Coba saja masukkan angka-angka di atas ke kalkulator Windows, atau mungkin juga di SO lain, anda akan memperoleh hasil persis seperti tertera di atas.

Pusing?

Mari kita konversikan bilangan binary di atas menjadi bilangan desimal supaya anda tidak terlalu pusing melihat angka 10101 dan sebagainya.

Dalam notasi desimal, kalimat di atas menjadi, 202.159.23.45 di AND dengan 255.255.255.0 menjadi 202.159.23.0 **Cukup familiar, khan?**

Coba perhatikan nilai-nilai alamat IP yang bisa kita masukan di **Start** → **Settings** → **Control Panel** → **Network** → **TCP/IP Properties** (Win98), atau dengan klik kanan network neighborhood → properties → di menu Configuration pilih TCP/IP (Win98), My Network Place di Win2000 atau WinXP, trus pilih Propertis → Local Area Connection (Oh..ya icon Network ini hanya ada di desktop Window apabila komputer anda telah memiliki LAN Card atau Network Adapter).

Kalau kita perhatikan baik-baik maka panjang sebuah alamat IP adalah 32 bit, yang dibagi dalam empat segmen yang di beri tanda titik "." antar segmennya. Artinya setiap segmen terdapat 8 bit.

Alokasi IP Address di Jaringan

Teknik subnet merupakan cara yang biasa digunakan untuk mengalokasikan sejumlah alamat IP di sebuah jaringan (LAN atau WAN). Teknik subnet menjadi penting bila kita mempunyai alokasi IP yang terbatas misalnya hanya ada 200 IP yang akan di distribusikan ke beberapa LAN.

Untuk memberikan gambaran, misalkan kita mempunyai alokasi alamat IP dari 192.168.1.0 s/d 192.168.1.255 untuk 254 host, maka parameter yang digunakan untuk alokasi tersebut adalah:

192.168.1.255 - broadcast address LAN

255.255.255.0 - subnet mask LAN

192.168.1.0 - netwok address LAN.

192.168.1.25 - contoh IP salah satu workstation di LAN.

Perhatikan bahwa,

Alamat IP pertama 192.168.1.0 tidak digunakan untuk *workstation*, tapi untuk menginformasikan bahwa LAN tersebut menggunakan alamat 192.168.1.0. Istilah keren-nya alamat IP 192.168.1.0 di sebut *network address*.

Alamat IP terakhir 192.168.1.255 juga tidak digunakan untuk workstation, karena digunakan untuk alamat *broadcast*. Alamat broadcast digunakan untuk memberikan informasi ke seluruh workstation yang berada di network 192.168.1.0 tersebut. Contoh informasi broadcast adalah informasi routing menggunakan Routing Information Protocol (RIP).

Subnet mask LAN 255.255.255.0, dalam bahasa yang sederhana dapat diterjemahkan bahwa setiap bit "1" menunjukan posisi network address, sedang setiap bit "0" menunjukkan posisi *host address*.

Konsep network address dan host address menjadi penting sekali berkaitan erat dengan subnet mask. Perhatikan dari contoh di atas maka alamat yang digunakan adalah :

192.168.1.0 network address

192.168.1.1 host ke 1

192.168.1.2 host ke 2

192.168.1.3 host ke 3

……

192.168.1.254 host ke 254

192.168.1.255 broacast address

Perhatikan bahwa angka 192.168.1 tidak pernah berubah sama sekali. Hal ini menyebabkan network address yang digunakan 192.168.1.0. Jika diperhatikan maka 192.168.1 terdiri dari 24 bit yang konstan tidak berubah, hanya 8 bit terakhir yang berubah memberikan identifikasi mesin yang mana. Tidak heran kalau *netmask* yang digunakan adalah (binary) 11111111.11111111.11111111.00000000 (desimal) 255.255.255.0.

Walaupun alamat IP workstation tetap, tetapi netmask yang digunakan di masing-masing router akan berubah-ubah bergantung pada posisi router dalam jaringan.

Masih bingung?

Mari kita lihat analogi di jaringan telepon yang biasa kita gunakan sehari-hari, misalnya kita mempunyai nomor telepon yang dapat di telepon dari luar negeri dengan nomor, +62 21 420 1234. Lokasi nomor telepon tersebut di Jakarta, dengan sentral di sekitar Ps.Senen dan Cempaka Putih. Kita perhatikan perilaku sentral telepon di tiga lokasi

1. Sentral di Amerika Serikat
2. Sentral di Indosat Jakarta
3. Sentral telepon di Telkom Jakarta Gatot Subroto dan
4. Sentral telepon di Senen, Cempaka Putih.

Pada saat seseorang di Amerika Serikat akan menghubungi rekannya di Jakarta dengan nomor +62 21 420 1234, maka pada sentral di Amerika Serikat, hanya memperhatikan dua digit pertama (+62). Setelah membaca angka +62 tanpa mempedulikan angka selanjutnya maka sentral di Amerika Serikat akan menghubungi gerbang SLI di Indosat Jakarta untuk memperoleh sambungan.

Perhatikan di sini netmask di sentral Amerika Serikat untuk jaringan di Indonesia hanya cukup dua digit pertama, selebihnya dianggap *host (handset)* di jaringan telepon Indonesia yang tidak perlu di perdulikan oleh sentral di Amerika Serikat. Sentral Indosat Jakarta, berbeda dengan sentral di Amerika Serikat, Indosat akan memperhatikan dua digit selanjutnya (jadi total +62 21). Dari informasi tersebut sentral Indosat mengetahui bahwa trafik tersebut untuk Jakarta dan akan meneruskan trafik ke sentral Telkom di Jl. Gatot Subroto di Jakarta. (sekarang netmask menjadi 4 digit).

Sentral Telkom di Gatot Subroto Jakarta akan melihat 3 digit selanjutnya, yakni 420 (+62 21 420), dari informasi tersebut maka sentral Telkom Gatot Subroto akan meneruskan trafik ke sentral yang lebih rendah, kemungkinan di Gambir atau sekitar Senen.

Perhatikan sekarang netmask menjadi 7 digit. Pada sentral terakhir di Gambir atau Senen, akan dilihat pelanggan mana yang dituju yang terdapat dalam empat digit terakhir (1234). Maka sampailah trafik ke tujuan. Nomor pelanggan kira-kira ekuivalen dengan host address di jaringan Internet.

Secara sederhana netmask digunakan untuk memisahkan antara network address dan host address untuk memudahkan proses routing di jaringan Internet. Dengan adanya netmask

kita tidak perlu memperhatikan seluruh alamat IP yang ada, tetapi cukup memperhatikan segelintir network address-nya saja.

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Pemrograman Client Server

Abstract

Modul ini akan menjelaskan tentang pemrograman cleint server pada Java.

Kompetensi

Mahasiswa dapat memahami pemrograman client server yang dibuat dengan bahasa Java.

Pemrograman Client Server

Arsitektur Client Server

Beragam komputer dari vendor yang bermacam-macam bisa saling berinteraksi. Istilah interoperatibilitas sering dipakai untuk menyatakan keadaan ini. Perkembangan ini akhirnya juga disusul oleh kemudahan perangkat lunak untuk saling berinteraksi . Sebuah basis data pada prinsipnya dapat diakses oleh perangkat lunak apa saja. Kebebasan ini merupakan suatu ciri khas pada arsitektur yang dinamakan client/server. Pada arsitektur ini, ada bagian yang disebut client dan ada yang disebut server.

Client adalah sembarang sistem atau proses yang melakukan suatu permintaan data atau layanan ke server.

Server adalah sistem atau proses yang menyediakan data atau layanan yang diminta oleh client . Secara fisik, sebuah serevr dapat berupa komputer mainframe, mini-komputer, workstation, ataupun PC atau peranti lain seperti printer, server tidak harus berupa sistem fisik, tetapi juga suatu proses.

Client mempunyai kemampuan untuk melakukan pemrosesan sendiri. Ketika sebuah client meminta suatu data ke server, server akan segera menanggapinya dengan memberikan data yang diminta ke client bersangkutan. Setelah diterima. Client segera melakukan pemprosesan.

Model komputasi yang berbasis client server mulai banyak diterapkan pada sistem informasi. Dengan menggunakan arsitektur ini , sistem informasi dapat digunakan dan dibangun dengan perangkat lunak client server yang bermacam-macam dan berbeda-beda.

Poin-poin Client Server

- Interoperabilitas
- Client, sembarang sistem atau proses yang melakukan suatu permintaan data atau layanan ke server
- Server, sistem atau proses yang menyediakan data atau layanan yang diminta oleh client
- Sistem informasi dapat dibangun dengan menggunakan perangkat lunak *gado-gado*

Implementasi Client Server

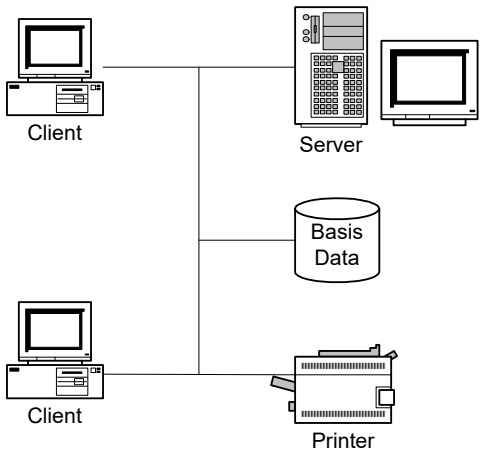
- Aplikasi pesan, misalnya surat elektronis (email)
- Penyebaran basis data pada beberapa jaringan komputer
- Memungkinkan berbagi berkas atau periferal atau pengaksesan komputer melalui jarak jauh

- Pemrosesan aplikasi yang intensif dengan suatu pekerjaan (*job*) dibagi menjadi tugas-tugas (*task*) yang masing-masing dilaksanakan pada komputer yang berbeda

Keuntungan Client Server

Fitur	**Keuntungan**
Jaringan mesin-mesin yang kecil tetapi berdaya guna	Jika sebuah mesin macet, bisnis tetap berjalan
Kumpulan komputer dengan ribuan MIPS (million instruction per second)	Sistem memberikan kekuatan dalam melaksanakan suatu tugas tanpa memonopoli sumber-sumber daya. Pemakai akhir diberi hak untuk bekerja secara local
Beberapa workstation sangat handal seperti mainframe, tetapi dengan biaya 90% lebih rendah	Menawarkan keluwesan untuk melakukan pembelian pada hal-hal lain atau untuk meningkatkan keuntungan
Sistem terbuka	Bebas memilih perangkat keras, perangkat lunak, dan layanan dari berbagai vendor
Sistem tumbuh dengan mudah dan dapat diperluas secara tak terbatas	Mudah untuk memperbaharui system
Lingkungan operasi client yang bersifat individual	Dapat mencampur dan mencocokkan platform komputer yang gsesuai dengan kebutuhan masing-masing departemen dan pemakai

Contoh Implementasi Jaringan Client Server



Aplikasi Client Server

Aplikasi client-server adalah sebuah aplikasi yang dihubungkan dengan sebuah metode untuk mengirimkan data atau mengambil data dari salah peer. Untuk aplikasi client-server yang akan dibuat pada modul ini adalah sebuah aplikasi client-server sederhana, di mana client dapat mengirimkan pesan kepada server, dan server dapat menerima pesan yang dikirimkan oleh client.

Pada aplikasi ini akan menggunakan port sebagai media penghubung, dan socket programming. Di Java, mendukung pemrograman menggunakan TCP ataupun UDP. Untuk aplikasi kali ini akan menggunakan TCP.

Tujuan pembuatan aplikasi

Tujuan dari pembuatan aplikasi ini adalah memahami konsep dasar untuk menghubungkan antara satu server dengan client agar terkoneksi melalui jaringan.

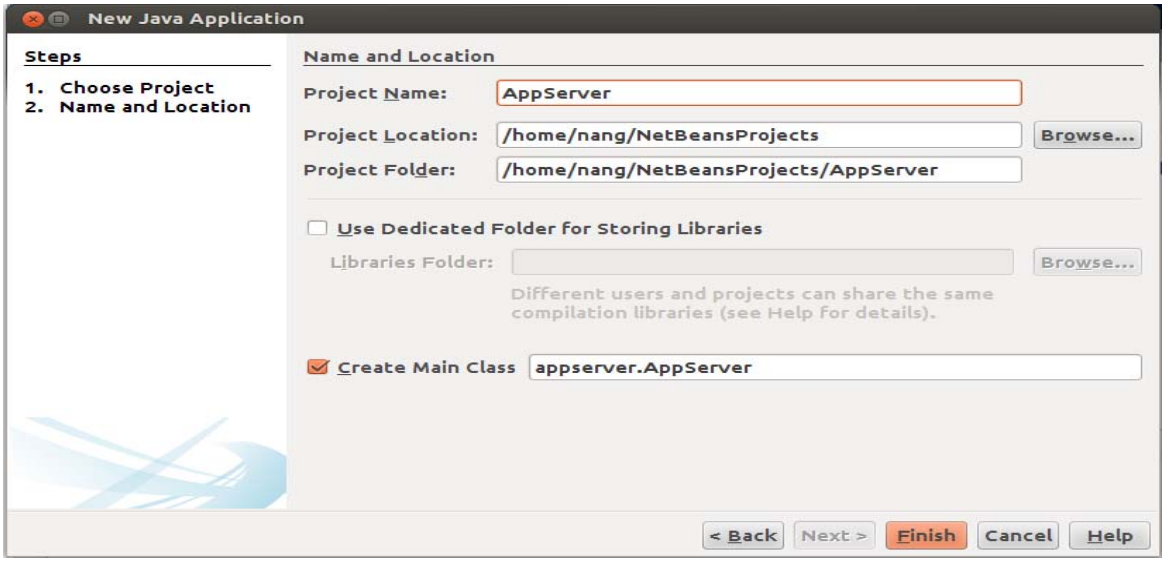
Langkah-langkah pembuatan aplikasi

Untuk membuat aplikasi ini, terdapat dua buah aplikasi, yaitu aplikasi server dan aplikasi client. Untuk tahap pertama akan dibuat terlebih dahulu aplikasi server.

Aplikasi Server

Tahapan pembuatan:

1. Buat sebuah projek baru pada Netbeans dengan nama AppServe



2. Tambahkan kode program berikut pada fungsi utama dari aplikasi server:

```
public static void main(String[] args) {
    // TODO code application logic here
    System.out.println("Opening port...\n");
    try {
        serverSocket = new ServerSocket(PORT);
    } catch(IOException ioEx) {
        ioEx.printStackTrace();
        System.out.println("Unable to attach to port!");
        System.exit(1);
    }
    do {
        handleClient();
    } while(true);
}
```

3. Kemudian membuat sebuah method dengan nama handleClient di bawah dari fungsi utama. Kode programnya adalah sebagai berikut:

```
private static void handleClient() {
    Socket link = null;
    try {
        link = serverSocket.accept();
        Scanner input = new Scanner(link.getInputStream());
        PrintWriter output = new PrintWriter(link.getOutputStream(),true);
        int numMessages = 0;
        String message = input.nextLine();
        while (!message.equals("***CLOSE***")) {
            System.out.println("Message received.");
            numMessages++;
            output.println("Message "+numMessages+": "+message);
            message = input.nextLine();
        }
        output.println(numMessages+" messages received");
    } catch(IOException ioEx) {
        ioEx.printStackTrace();
    } finally {
        try {
            System.out.println("\n Closing connection...");
            link.close();
        } catch(IOException ioE) {
            System.out.println("Unable to disconnect!");
            System.exit(1);
        }
    }
}
```

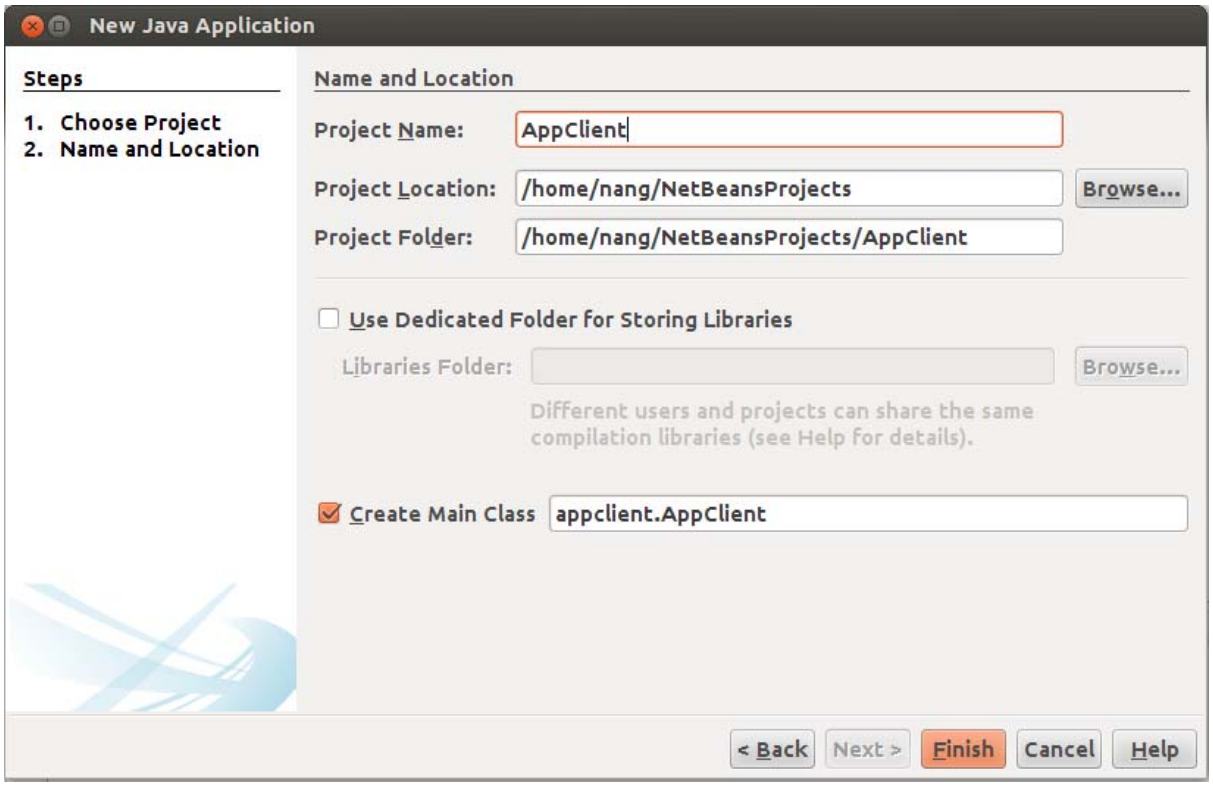
Aplikasi ini menggunakan class ServerSocket. Class ServerSocket digunakan untuk mendengarkan port yang telah disetting / diinisialisasi di dalam aplikasi.

Method accept() digunakan untuk membuat server berada di posisi stand by / ready untuk menerima koneksi dari client.

Aplikasi Client

Tahapan pembuatannya:

1. Buat project baru dengan nama AppClient



2. Buat inisiatlisasi untuk class ServerSocket dan juga PORT yang akan digunakan.
3. Tambahkan kode program berikut pada fungsi utama dari aplikasi client:

```
public static void main(String[] args) {
    // TODO code application logic here
    try {
        host = InetAddress.getLocalHost();
    } catch(UnknownHostException uhEx) {
        System.out.println("Host ID not found!");
        System.exit(1);
    }
    accessServer();
}
```

4. Kemudian buat sebuah method accessServer di bawah dari fungsi utama. Kode programnya adalah sebagai berikut:

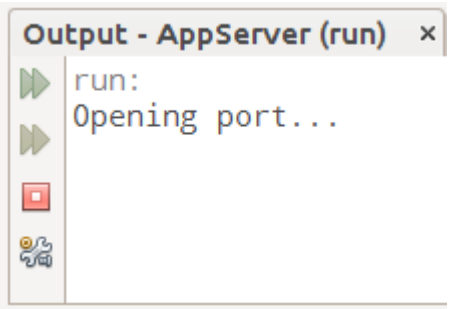
```
private static void accessServer() {
    Socket link = null;
    try {
        link = new Socket(host,PORT);
        Scanner input = new Scanner(link.getInputStream());
        PrintWriter output = new PrintWriter(link.getOutputStream(), true);
        Scanner userEntry = new Scanner(System.in);
        String message, response;
        do {
            System.out.print("Enter message: ");
            message = userEntry.nextLine();
            output.println(message);
            response = input.nextLine();
            System.out.println("\nSERVER> "+response);
        } while (!message.equals("***CLOSE***"));
    } catch(IOException ioEx) {
        ioEx.printStackTrace();
    } finally {
        try {
            System.out.println("\n Closing connection...");
            link.close();
        } catch(IOException ioEx) {
            System.out.println("Unable to disconnect!");
            System.exit(1);
        }
    }
}
```

Aplikasi client menggunakan class Socket. Class Socket digunakan untuk melakukan koneksi dengan server. Parameter yang digunakan adalah port dan host dari server yang dituju.

Untuk mengirimkan data ke server, aplikasi menggunakan bantuan class PrintWriter. Dan untuk menerima data dari server menggunakan bantuan dari class Scanner.

Tampilan dari aplikasi

Tampilan Server



Server membuka port, dan server berada dalam state siap menerima koneksi dari client.

Tampilan Client

```
run:
Enter message: kirim ke server

SERVER> Message 1: kirim ke server
Enter message:
```

Client mengirimkan data (sebuah kalimat dalam bentuk teks), kemudian server menerima dan mengembalikan respon ke client. Respon yang diberikan adalah data yang dikirimkan ke server.

Daftar Pustaka

1. http://developer.android.com
2. Purbo, O.W. VoIP Cock Book.
3. *http://opensource.telkomspeedy.com/wiki/index.php/VoIP*

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Abstract

Modul ini akan menjelaskan tentang pemrograman simplex pada Java.

Kompetensi

Mahasiswa dapat memahami pemrograman simplex dengan bahasa Java.

Pemrograman Simplex

Jaringan Simplex

- Sinyal ditransimisi dalam satu arah.
- Stasiun yang satu bertindak sebagai transmitter, dan yang lainnya adalah receiver. Tugasnya adalah tetap.

Jarang digunakan untuk sistem komunikasi data.



Contoh program yang menggunakan Simplex.

Di bawah ini adalah sebuah program yang digunakan untuk menerima data.

```
import java.net.*;
import java.io.*;
class datagramReceiver{
public static void main(String[ ] args){
try{
int MAX_LEN = 40;
int localPortNum = Integer.parseInt(args[0]);
DatagramSocket mySocket = new DatagramSocket(localPortNum);
byte[] buffer = new byte[MAX_LEN];
DatagramPacket packet = new DatagramPacket(buffer, MAX_LEN);
mySocket.receive(packet);
String message = new String(buffer);:
System.out.println(message);
mySocket.close( );
}
catch(Exception e){e.printStackTrace( );}
}
}
```

Sedangkan untuk kode program untuk mengirimkan data adalah sebagai berikut:

```
import java.net.*;
import java.io.*;
class datagramSender{
public static void main(String[ ] args){
```

```
try{
InetAddress receiverHost = InetAddress.getByName(args[0]);
int receiverPort = Integer.parseInt(args[1]);
String message = args[2];
DatagramSocket mySocket = new DatagramSocket( );
byte[] buffer = message.getBytes( );
DatagramPacket packet = new DatagramPacket(buffer, buffer.length, receiverHost,
receiverPort);
mySocket.send(packet);
mySocket.close( );
}
catch(Exception e){ e.printStackTrace( ); }
}
}
```

Teknik Komunikasi : Bagaimana menyampaikan informasi ke tempat tujuan dengan cepat dan tepat.

Cara untuk melakukan komunikasi anatara lain :

- **Lama** : Suara, gerak-gerik/lambang dalam bentuk gambar.
- **Modern** : Sinyal listrik, elektro optik.

Proses komunikasi umumnya sinyal informasi tidak langsung disalurkan ke penerima, tapi mengalami perubahan, seperti gambar berikut ini :

Bentuk komunikasi :

1. Komunkasi suara.
2. Komunikasi gerak.
3. Komunikasi data.

Berdasarkan cara penyampaian informasi, bentuk komunikasi dapat kita bedakan,antara lain :

1. Komunikasi dari titik ke titik/point to point communications, dikenal juga dengan sistem telekomunikasi.
2. Komunikasi dengan cara Broadcassting, adalah dari satu titik ke banyak titik.

Hal-hal penting pada sistem komunikasi :

1. Berita/informasi haruslah dapat dimengerti oleh penerima.

2. Karakteristik sistem komunikasi.
3. Derau/gangguan.

Berdasarkan cara penyampaian informasi, bentuk komunikasi dapat kita bedakan,antara lain :

Teknik penyambungan :

1. Circuit switching.
2. Message switching.

Pengolahan data : Adalah pengolahan data dengan menggunakan mesin/komputer..
Komunikasi data : Merupakan komunikasi dari mesin ke mesin/komputer ke komputer, dengan menggunakan proses pengiriman data menggunakan saluran telekomunikasi.
Jadi, pengertian dari Komunikasi data adalah
proses pengiriman informasi(data) yang telah diolah dalam bentuk kode tertentu yang disepakati melalui media listrik / elektronik dari titik ke titik yang lainnya.

Komponen dasar komunikasi data :

DTE : Data Terminal Equipment.
DCE : Data Communication Equipment.

Komuikasi data dapat melakukan sesuatu dalam waktu bersamaan, antara lain :

1. Pengumpulan dan persiapan data.
2. Pengolahan data.
3. Distribusi data.

Tujuan komunikasi data :

1. Pengiriman data dalam jumlah besar dan efisien.
2. Penggunaan sistem komputer dan alat pendukung secara bersamaan.
3. Menggunakan sistem komputer secara terpusat/tersebar.
4. Mempermudah pengolahan dan pengaturan data.
5. Mendapatkan data secara langsung dari sumbernya.
6. Mempercepat penyebaran informasi.

Operasi komunikasi data :

1. Pengumpulan data/data collection.
2. Pertukaran informasi/information exchange.
3. Menyimpan dan mengambil data/data storage and access.
4. Time sharing.
5. Program to program communications.
6. Remote computing.

DTE : Data Terminal Equipment.
DCE : Data Communication Equipment.

Komuikasi data dapat melakukan sesuatu dalam waktu bersamaan, antara lain :
Pengumpulan dan persiapan data.
Pengolahan data.
Distribusi data.

Tujuan komunikasi data :
Pengiriman data dalam jumlah besar dan efisien.
Penggunaan sistem komputer dan alat pendukung secara bersamaan.
Menggunakan sistem komputer secara terpusat/tersebar.
Mempermudah pengolahan dan pengaturan data.
Mendapatkan data secara langsung dari sumbernya.
Mempercepat penyebaran informasi.

Operasi komunikasi data :
Pengumpulan data/data collection.
Pertukaran informasi/information exchange.
Menyimpan dan mengambil data/data storage and access.
Time sharing.
Program to program communications.
Remote computing.

Pendekatan Dasar Jaringan pada Pemrosesan

Terdapat dua pendekatan pada jaringan komunikasi data yaitu ***Timesharing***, **Pemrosesan Terdistribusi**, dan ***Client/Server computing***.

Jaringan *Timesharing* terdiri dari satu komputer yang dipakai bersama oleh beberapa pemakai yang memperoleh akses melalui terminal-terminal.
Pemrosesan terdistribusi adalah suatu pendekatan dengan cara mendistribusikan komputer mini dan komputer mikro ke seluruh jaringan organisasi, pendekatan ini menggantikan *Timesharing* sebagai strategi jaringan.
Sedangkan *Client/Server Computing* adalah suatu pendekatan bagi penggunaan jaringan yang didasarkan pada konsep bahwa sebagian fungsi paling baik ditangani secara lokal dan sebagian paling baik ditangani secara terpusat. Oleh karenanya merupakan gabungan pendekatan *Timesharing* dangan Pemrosesan Terdistribusi.

Jaringan Komputer khusus PC – LAN

Penggunaan komputer PC terus meningkat diberbagai bidang. Hal ini terus memacu perusahaan pembuat komputer dan perangkat lunak untuk berbuat lebih baik / maju lagi. Sejalan dengan perkembangan sistem operasi yang dikembangkan oleh suatu organisasi ataupun sistem komputerisasi yang terjadi di dalam organisai tersebut, maka sangatlah dibutuhkan akan adanya sistem jaringan kerja komputer PC.
Sistem jaringan komputer ini bisa terbatas pada suatu lokasi tertentu saja yang dikenal dengan PC LAN (*Personal Computer Local Area Network*), atau sistem jaringan komputer jarak jauh (antar daerah/ negara dengan menggunakan bantuan satelit atau dengan sistem bantuan *Modem*).
Pada kesempatan ini kita tinjau sekilas tentang sistem jaringan komputer PC dengan sistem PC LAN.

Metodologi Pengendalian LAN

Cara pesan melintasi LAN ditentukan oleh jenis pengendlian LAN. Ada dua metodologi yang popular yaitu :
Pengendalian Contention Based;
Pertama-tama memperhatikan sinyal sibuk. Bila sinyal tersebut tidak terdeteksi maka workstation tersebut mengirimkan pesannya. Karena tidak ada pengendalian terpusat, memungkinkan dua workstation atau lebih dapat mencoba transmisi pada waktu bersamaan. Hal tersebut dinamakan tabrakan (collision). Sehingga diperlukan skema pengendalian, pengendalian contention based merupakan bentuk pengendalian sederhana, namun kinerja jaringan dapat menurun cepat jika transmisi sibuk.
Pengendalian Token Passing;

Sentralisasi pengendalian dilakukan dengan menggunakan token. Token adalah seperangkat bit data yang diteruskan dari satu workstation ke workstation lain. Sehingga pesan hanya bias terkirim bila memiliki token pada jaringan tersebut. Metode ini sama dengan perlombaan lari estafet yang pelarinya hanya dapat berlari jika memegang tongkat.

Daftar Pustaka

1. http://developer.android.com
2. Purbo, O.W. VoIP Cock Book.
3. *http://opensource.telkomspeedy.com/wiki/index.php/VoIP*

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Aplikasi Full Duplex

Abstract

Modul ini akan menjelaskan tentang pemrograman full duplex.

Kompetensi

Mahasiswa dapat memahami pemrograman full duplex dengan bahasa Java.

Pemrograman Duplex

Jaringan Duplex

Duplex adalah sebuah istilah dalam bidang telekomunikasi yang merujuk kepada komunikasi dua arah.

Terdapat dua metode duplexing, yakni

- Full-Duplex
- Half Duplex

Full Duplex

Dalam komunikasi full-duplex, dua pihak yang saling berkomunikasi akan mengirimkan informasi dan menerima informasi dalam waktu yang sama, dan umumnya membutuhkan dua jalur komunikasi.

Komunikasi full-duplex juga dapat diraih dengan menggunakan teknik multiplexing, di mana sinyal yang berjalan dengan arah yang berbeda akan diletakkan pada slot waktu (time slot) yang berbeda. Kelemahan teknik ini adalah bahwa teknik ini memotong kecepatan transmisi yang mungkin menjadi setengahnya.

Contoh implementasi dari full duplex adalah telepon.

Half-Duplex

Half-duplex merupakan sebuah mode komunikasi di mana data dapat ditransmisikan atau diterima secara dua arah tapi tidak dapat secara bersama-sama. Contoh paling sederhana adalah walkie-talkie, di mana dua penggunanya harus menekan sebuah tombol untuk berbicara dan melepaskan tombol tersebut untuk mendengar. Ketika dua orang menggunakan walkie-talkie untuk berkomunikasi pada satu waktu tertentu, hanya salah satu di antara mereka yang dapat berbicara sementara pihak lainnya mendengar. Jika kedua-duanya mencoba untuk berbicara secara serentak, kondisi "collision" (tabrakan) pun terjadi dan kedua pengguna walkie-talkie tersebut tidak dapat saling mendengarkan apa yang keduanya kirimkan.

Contoh Aplikasi Full-Duplex

Contoh aplikasi full-duplex yang akan diberikan pada modul ini adalah sebuah contoh aplikasi VoIP client. Aplikasi VoIP client memiliki konsep yang sama dengan telepon, tugasnya untuk menerima dan mengirim suara melalui perangkat yang disediakan.

Contoh aplikasi VoIP Client

Saat ini, banyak sekali VoIP Client yang dibuat oleh para pengembang aplikasi. Mulai dari versi desktop sampai dengan versi mobile. Akan tetapi, semua VoIP client tersebut rata-rata menggunakan protocol yang sama yaitu SIP (Session Initiation Protocol).
SIP ini digunakan untuk membangun komunikasi dari dua buah VoIP client melalui jaringan VoIP. SIP tidak bekerja sendiri, akan tetapi dibantu oleh protocol-protocol lainnya dalam membangun komunikasi di jaringan VoIP, salah satu contoh protocol pembantunnya adalah RTP (Real Time Protocol), yang berfungsi membawa paket-paket data secara real-time.
Beberapa aplikasi VoIP client yang ada di pasaran saat ini adalah sebagai berikut:
Versi Desktop:

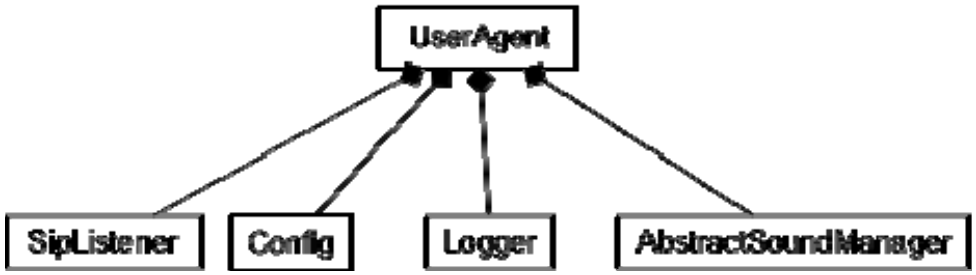
1. Xlite. Softphone yang sudah memiliki fitur yang sangat lengkap. Saat ini xlite merupakan aplikasi berbayar.
2. Ekiga. Softphone berbasis open source. Menjadi salah satu andalan dalam sistem operasi Linux.
3. Linphone. Softphone berbasis open source.
4. Peers. Softphone berbasis open source dengan bahasa pemrograman java. Masih banyak yang harus dikembangkan pada aplikasi ini.

Versi Mobile:

1. Sipdroid
2. CSipSimple
3. Line
4. Dll

Contoh Aplikasi VoIP Client

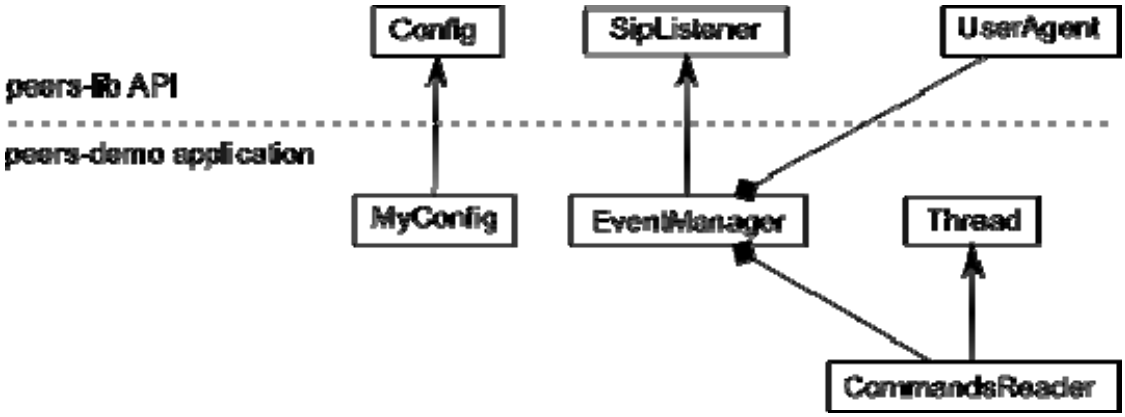
Berikut ini akan ditampilkan sebuah aplikasi VoIP client berbasis Java dan berlisensi open source. Peers, adalah sebuah sofphone yang sederhana dibangun dengan bahasa pemrograman Java.
Arsitektur API dari Peers adalah sebagai berikut:



Peers menyediakan satu class untuk menggunakan protocol SIP dan media stack (UserAgent), dan sebuah interface untuk menerima notifikasi dari SIP stack (SipListener).
UserAgent konstruktor terdiri dari beberapa parameter, yaitu:

- SipListener
- Config
- Logger
- AbstractSoundManager

Selanjutnya, akan ditampilkan class diagram dari Peers



Dari gambar di atas, dapat dilihat bahwa konsep dari Peers memang dibuat sederhana, sehingga tidak banyak class yang dibuat pada aplikasi ini.
Selanjutnya akan dibahas salah satu class yang digunakan pada aplikasi Peers ini.

Class CommandsReader

Class yang digunakan untuk membaca perintah dari console.

```
package net.sourceforge.peers.demo;
[...]
public class CommandsReader extends Thread {

   [...]

   @Override
   public void run() {
      InputStreamReader inputStreamReader = new InputStreamReader(System.in);
```

```
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    setRunning(true);
    while (isRunning()) {
        String command;
        try {
            command = bufferedReader.readLine();
        } catch (IOException e) {
            e.printStackTrace();
            break;
        }
        command = command.trim();
        if (command.startsWith(CALL)) {
            String callee = command.substring(
                    command.lastIndexOf(' ') + 1);
            eventManager.call(callee);
        } else if (command.startsWith(HANGUP)) {
            eventManager.hangup();
        } else {
            System.out.println("unknown command " + command);
        }
    }
}
```

EventManager Interfaces

EventManager class mengimplementasikan method yang dipanggil dari CommandsReader, dan SipListener Interface dari Pees API.

```
public class EventManager implements SipListener {

    // commands methods
    public void call(String callee) { }
    public void hangup() { }


    // SipListener methods

    @Override
    public void registering(SipRequest sipRequest) { }
```

```
    @Override
    public void registerSuccessful(SipResponse sipResponse) { }

    @Override
    public void registerFailed(SipResponse sipResponse) { }

    @Override
    public void incomingCall(SipRequest sipRequest, SipResponse provResponse) { }

    @Override
    public void remoteHangup(SipRequest sipRequest) { }

    @Override
    public void ringing(SipResponse sipResponse) { }

    @Override
    public void calleePickup(SipResponse sipResponse) { }

    @Override
    public void error(SipResponse sipResponse) { }

}
```

Membuat UserAgent di EventManager

```
private UserAgent userAgent;
    private CommandsReader commandsReader;

    public EventManager() throws SocketException {
        Config config = new CustomConfig();
        Logger logger = new FileLogger(null);
        JavaxSoundManager javaxSoundManager = new JavaxSoundManager(false,
            logger, null);
        userAgent = new UserAgent(this, config, logger, javaxSoundManager);
        new Thread() {
            public void run() {
```

```
            try {
                userAgent.register();
            } catch (SipUriSyntaxException e) {
                e.printStackTrace();
            }
        }
    }.start();
    commandsReader = new CommandsReader(this);
    commandsReader.start();
  }
```

Disini, digunakan custom implementation dari Config Interface (CustomConfig) untuk menyediakan hardcoded account parameter.

CustomConfig

```
package net.sourceforge.peers.demo;

[...]
public class CustomConfig implements Config {

  private InetAddress publicIpAddress;

  @Override
  public InetAddress getLocalInetAddress() {
    InetAddress inetAddress;
    try {
      // if you have only one active network interface, getLocalHost()
      // should be enough
      //inetAddress = InetAddress.getLocalHost();
      // if you have several network interfaces like I do,
      // select the right one after running ipconfig or ifconfig
      inetAddress = InetAddress.getByName("192.168.1.11");
    } catch (UnknownHostException e) {
      e.printStackTrace();
      return null;
    }
    return inetAddress;
  }
```

```
    @Override
    public InetAddress getPublicInetAddress() { return publicIpAddress; }
    @Override public String getUserPart() { return "alice"; }
    @Override public String getDomain() { return "atlanta.com"; }
    @Override public String getPassword() { return "secret1234"; }
    @Override  // use microphone and speakers to capture and playback sound
    public MediaMode getMediaMode() { return MediaMode.captureAndPlayback; }

    @Override
    public void setPublicInetAddress(InetAddress inetAddress) {
        publicIpAddress = inetAddress;
    }
@Override public SipURI getOutboundProxy() { return null; }
    @Override public int getSipPort() { return 0; } // use default sip port 5060
    @Override public boolean isMediaDebug() { return false; }
    @Override public String getMediaFile() { return null; }
    @Override public int getRtpPort() { return 0; } // use random rtp port

    // in this simple example, we don't need those modifiers, but they are
    // required by the interface
    @Override public void setLocalInetAddress(InetAddress inetAddress) { }
    @Override public void setUserPart(String userPart) { }
    @Override public void setDomain(String domain) { }
    @Override public void setPassword(String password) { }
    @Override public void setOutboundProxy(SipURI outboundProxy) { }
    @Override public void setSipPort(int sipPort) { }
    @Override public void setMediaMode(MediaMode mediaMode) { }
    @Override public void setMediaDebug(boolean mediaDebug) { }
    @Override public void setMediaFile(String mediaFile) { }
    @Override public void setRtpPort(int rtpPort) { }
    @Override public void save() { }

}
```

Daftar Pustaka

1. http://peers.sourceforge.net
2. *http://opensource.telkomspeedy.com/wiki/index.php/VoIP*
3. *http://id.wikipedia.org/wiki/Duplex*

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Paket Data TCP

Abstract

Modul ini akan menjelaskan tentang paket data TCP.

Kompetensi

Mahasiswa dapat memahami pemrograman untuk mengirim paket data di protocol TCP.

TCP (Transmission Control Protocol)

Pengertian TCP

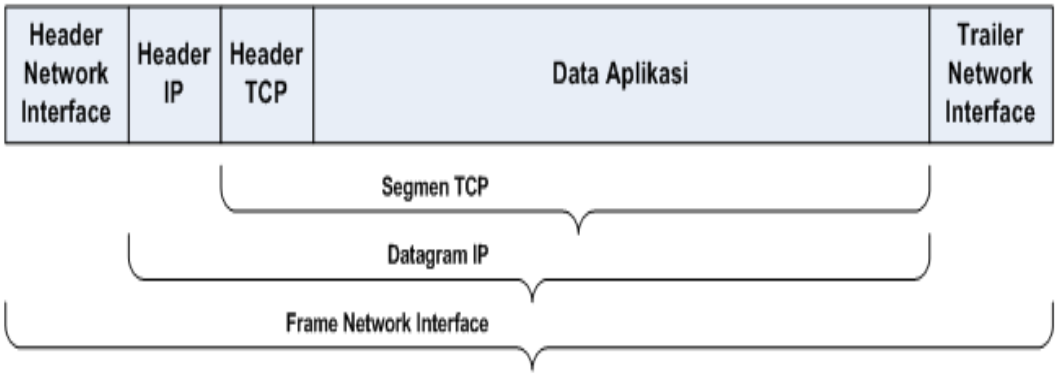
Transmission Control Protocol (TCP) adalah suatu protocol yang berada di lapisan transport (baik itu dalam tujuh lapis model referensi OSI atau model DARPA) yang berorientasi sambungan (connection-oriented) dan dapat diandalkan (reliable).
TCP dispesifikasikan dalam RFC 793.
Karakteristik TCP

- Berorientasi sambungan (connection-oriented): sebelum data dapat ditransmisikan antara dua host, dua proses yang berjalan pada lapisan aplikasi harus melakukan negosiasi untuk membuat sesi koneksi terlebih dahulu. Koneksi TCP ditutup dengan menggunakan proses terminasi koneksi TCP (TCP connection termination).
- Full duplex: Untuk setiap host TCP, koneksi yang terjadi antara dua host terdiri atas dua buah jalur, yakni jalur keluar dan jalur masuk. Dengan menggunakan teknologi lapisan yang lebih rendah yang mendukung full-duplex, maka data pun dapat secara simultan diterima dan dikirim. Header TCP berisi nomor urut (TCP sequence number) dari data yang ditransmisikan dan sebuah acknowledgment dari data yang masuk.
- Dapat diandalkan (reliable): Data yang dikirimkan ke sebuah koneksi TCP akan diurutkan dengan sebuah nomor urut paket dan akan mengharapkan paket *positive acknowledgment* dari penerima. Jika tidak ada paket Acknowledgment dari penerima, maka segmen TCP (protocol data unit dalam protokol TCP) akan ditransmisikan ulang. Pada pihak penerima, segmen-segmen duplikat akan diabaikan dan segmen-segmen yang datang tidak sesuai dengan urutannya akan diletakkan di belakang untuk mengurutkan segmen-segmen TCP. Untuk menjamin integritas setiap segmen TCP, TCP mengimplementasikan penghitungan TCP Checksum.
- Byte Stream: TCP melihat data yang dikirimkan dan diterima melalui dua jalur masuk dan jalur keluar TCP sebagai sebuah *byte stream* yang berdekatan (kontigu). Nomor urut TCP dan nomor acknowlegment dalam setiap header TCP didefinisikan juga dalam bentuk byte. Meski demikian, TCP tidak mengetahui batasan pesan-pesan di dalam byte stream TCP tersebut. Untuk melakukannya, hal ini diserahkan kepada protokol lapisan aplikasi (dalam DARPA Reference Model), yang harus menerjemahkan byte stream TCP ke dalam "bahasa" yang ia pahami.

- Memiliki layanan flow control: Untuk mencegah data terlalu banyak dikirimkan pada satu waktu, yang akhirnya membuat "macet" jaringan internetwork IP, TCP mengimplementasikan layanan *flow control* yang dimiliki oleh pihak pengirim yang secara terus menerus memantau dan membatasi jumlah data yang dikirimkan pada satu waktu. Untuk mencegah pihak penerima untuk memperoleh data yang tidak dapat disangganya (*buffer*), TCP juga mengimplementasikan *flow control* dalam pihak penerima, yang mengindikasikan jumlah *buffer* yang masih tersedia dalam pihak penerima.
- Melakukan segmentasi terhadap data yang datang dari lapisan aplikasi (dalam *DARPA Reference Model*)
- Mengirimkan paket secara "*one-to-one*": hal ini karena memang TCP harus membuat sebuah sirkuit logis antara dua buah protokol lapisan aplikasi agar saling dapat berkomunikasi. TCP tidak menyediakan layanan pengiriman data secara *one-to-many*.

Segmen TCP

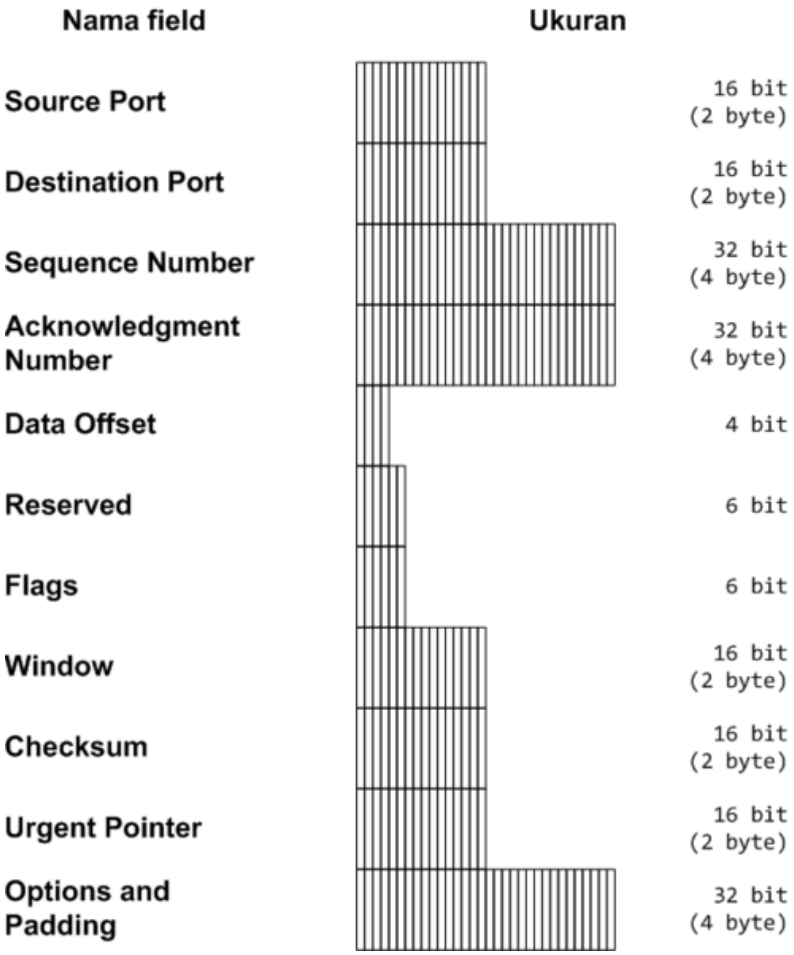
Segmen-segmen TCP akan dikirimkan sebagai datagram-datagram IP (datagram merupakan satuan protocol data unit pada lapisan *internetwork*). Sebuah segmen TCP terdiri atas sebuah *header* dan segmen data (*payload*), yang dienkapsulasi dengan menggunakan *header* IP dari protokol IP.



Proses enkapsulasi data protokol TCP/IP: Data aplikasi + header TCP + header IP + header network interface (Ethernet, Token Ring, dll) + trailer network interface.
Sebuah segmen dapat berukuran hingga 65495 byte: 216-(ukuran header IP terkecil (20 byte)+ukuran header TCP terkecil (20 byte)). *Datagram* IP tersebut akan dienkapsulasi lagi dengan menggunakan header protokol network interface (lapisan pertama dalam DARPA Reference Model) menjadi frame lapisan Network Interface. Gambar berikut mengilustrasikan data yang dikirimkan ke sebuah host.

Di dalam header IP dari sebuah segmen TCP, *field* **Source IP Address** diatur menjadi alamat *unicast* dari sebuah antarmuka host yang mengirimkan segmen TCP yang bersangkutan. Sementara itu, field **Destination IP Address** juga akan diatur menjadi alamat *unicast* dari sebuah antarmuka host tertentu yang dituju. Hal ini dikarenakan, protokol TCP hanya mendukung transmisi *one-to-one*.

TCP Header



Source Port = 2 byte (16 bit)

Mengindikasikan sumber protokol lapisan aplikasi yang mengirimkan segmen TCP yang bersangkutan. Gabungan antara *field* **Source IP Address** dalam *header IP* dan *field* **Source Port** dalam *field header TCP* disebut juga sebagai ***socket* sumber**, yang berarti sebuah alamat global dari mana segmen dikirimkan. Lihat juga Port TCP.

Destination Port:2 byte (16 bit)

Mengindikasikan tujuan protokol lapisan aplikasi yang menerima segmen TCP yang bersangkutan. Gabungan antara field Destination IP Address dalam header IP dan field

Destination Port dalam field header TCP disebut juga sebagai ***socket* tujuan**, yang berarti sebuah alamat global ke mana segmen akan dikirimkan.

Sequence Number:4 byte (32 bit)

Mengindikasikan nomor urut dari oktet pertama dari data di dalam sebuah segmen TCP yang hendak dikirimkan. Field ini harus selalu diset, meskipun tidak ada data (payload) dalam segmen.

Ketika memulai sebuah sesi koneksi TCP, segmen dengan flag SYN (Synchronization) diset ke nilai 1, field ini akan berisi nilai Initial Sequence Number (ISN). Hal ini berarti, oktet pertama dalam aliran byte (byte stream) dalam koneksi adalah ISN+1.

Acknowledgment Number:4 byte (32 bit)

Mengindikasikan nomor urut dari oktet selanjutnya dalam aliran byte yang diharapkan oleh untuk diterima oleh pengirim dari si penerima pada pengiriman selanjutnya.

Acknowledgment number sangat dipentingkan bagi segmen-segmen TCP dengan flag ACK diset ke nilai 1.

Data Offset:4 bit

Mengindikasikan di mana data dalam segmen TCP dimulai. Field ini juga dapat berarti ukuran dari header TCP. Seperti halnya field **Header Length** dalam header IP, field ini merupakan angka dari word 32-bit dalam header TCP. Untuk sebuah segmen TCP terkecil (di mana tidak ada opsi TCP tambahan), field ini diatur ke nilai 0x5, yang berarti data dalam segmen TCP dimulai dari oktet ke 20 dilihat dari permulaan segmen TCP. Jika field Data Offset diset ke nilai maksimumnya (24=16) yakni 15, header TCP dengan ukuran terbesar dapat memiliki panjang hingga 60 byte.

Reserved:6 bit

Direservasikan untuk digunakan pada masa depan. Pengirim segmen TCP akan mengeset bit-bit ini ke dalam nilai 0.

Flags:6 bit

Mengindikasikan flag-flag TCP yang memang ada enam jumlahnya, yang terdiri atas: URG (Urgent), ACK (Acknowledgment), PSH (Push), RST (Reset), SYN (Synchronize), dan FIN (Finish).

Window:2 byte (16 bit)

Mengindikasikan jumlah byte yang tersedia yang dimiliki oleh buffer host penerima segmen yang bersangkutan. Buffer ini disebut sebagai Receive Buffer, digunakan untuk menyimpan

byte stream yang datang. Dengan mengimbuhkan ukuran window ke setiap segmen, penerima segmen TCP memberitahukan kepada pengirim segmen berapa banyak data yang dapat dikirimkan dan disangga dengan sukses. Hal ini dilakukan agar si pengirim segmen tidak mengirimkan data lebih banyak dibandingkan ukuran Receive Buffer. Jika tidak ada tempat lagi di dalam Receive buffer, nilai dari field ini adalah 0. Dengan nilai 0, maka si pengirim tidak akan dapat mengirimkan segmen lagi ke penerima hingga nilai field ini berubah (bukan 0). Tujuan hal ini adalah untuk mengatur lalu lintas data atau *flow control*.

Checksum:2 byte (16 bit)

Mampu melakukan pengecekan integritas segmen TCP (*header*-nya dan *payload*-nya). Nilai field Checksum akan diatur ke nilai 0 selama proses kalkulasi checksum.

Urgent Pointer:2 byte (16 bit)

Menandakan lokasi data yang dianggap "urgent" dalam segmen.

Options:4 byte (32 bit)

Berfungsi sebagai penampung beberapa opsi tambahan TCP. Setiap opsi TCP akan memakan ruangan 32 bit, sehingga ukuran header TCP dapat diindikasikan dengan menggunakan field Data offset

TCP Flag



URG

Mengindikasikan bahwa beberapa bagian dari segmen TCP mengandun data yang sangat penting, dan field Urgent Pointer dalam header CP harus digunakan untuk menentukan lokasi di mana data penting tersebut berada dala segmen

ACK

Mengindikasikan field Acknowledgment mengandung oktet selanjutnya yang diharapkan dalam koneksi. Flag ini selalu diset, kecuali pada segmen pertama pada pembuatan sesi koneksi TCP

PSH

Mengindikasikan bahwa isi dari TCP Receive buffer harus diserahkan kepada protokol lapisan aplikasi. Data dalam receive buffer harus berisi sebuah blok data yang berurutan (kontigu), dilihat dari ujung paling kiri dari buffer. Dengan kata lain, sebuah segmen yang memiliki flag PSH diset ke nilai 1, tidak bolah ada satu byte pun data yang hilang dari aliran byte segmen tersebut; data tidak dapat diberikan kepada protokol lapisan aplikasi hingga segmen yang hilang tersebut datang. Normalnya, TCP Receive buffer akan dikosongkan (dengan kata lain, isi dari buffer akan diteruskan kepada protokol lapisan aplikasi) ketika buffer tersebut berisi data yang kontigu atau ketika dalam "proses perawatan".
Flag PSH ini dapat mengubah hal seperti itu, dan membuat akan TCP segera mengosongkan TCP Receive buffer. Flag PSH umumnya digunakan dalam protokol lapisan aplikasi yang bersifat interaktif, seperti halnya Telnet, karena setiap penekanan tombol dalam sesi terminal virtual akan dikirimkan dengan sebuah flag PSH diset ke nilai 1. Contoh dari penggunaan lainnya dari flag ini adalah pada segmen terakhir dari berkas yang ditransfer dengan menggunakan protokol FTP. Segmen yang dikirimkan dengan flag PSH aktif tidak harus segera di-acknowledge oleh penerima.

RST

Mengindikasikan bahwa koneksi yang dibuat akan digagalkan. Untuk sebuah koneksi TCP yang sedang berjalan (aktif), sebuah segmen dengan flag RST diset ke nilai 1 akan dikirimkan sebagai respons terhadap sebuah segmen TCP yang diterima yang ternyata segmen tersebut bukan yang diminta, sehingga koneksi pun menjadi gagal. Pengiriman segmen dengan flag RST diset ke nilai 1 untuk sebuah koneksi aktif akan menutup koneksi secara paksa, sehingga data yang disimpan dalam buffer akan dibuang (dihilangkan). Untuk sebuah koneksi TCP yang sedang dibuat, segmen dengan flag RST aktif akan dikirimkan sebagai respons terhadap request pembuatan koneksi untuk mencegah percobaan pembuatan koneksi.

SYN

Mengindikasikan bahwa segmen TCP yang bersangkutan mengandung Initial Sequence Number (ISN). Selama proses pembuatan sesi koneksi TCP, TCP akan mengirimkan sebuah segmen dengan flag SYN diset ke nilai 1. Setiap host TCP lainnya akan memberikan jawaban (acknowledgment) dari segmen dengan flag SYN tersebut dengan menganggap bahwa segmen tersebut merupakan sekumpulan byte dari data. Field Acknowledgment Number dari sebuah segmen SYN diatur ke nilai ISN + 1.

FIN

Menandakan bahwa pengirim segmen TCP telah selesai dalam mengirimkan data dalam sebuah koneksi TCP. Ketika sebuah koneksi TCP akhirnya dihentikan (akibat sudah tidak ada data yang dikirimkan lagi), setiap host TCP akan mengirimkan sebuah segmen TCP dengan flag FIN diset ke nilai 1. Sebuah host TCP tidak akan mengirimkan segmen dengan flag FIN hingga semua data yang dikirimkannya telah diterima dengan baik (menerima paket acknowledgment) oleh penerima. Setiap host akan menganggap sebuah segmen TCP dengan flag FIN sebagai sekumpulan byte dari data. Ketika dua host TCP telah mengirimkan segmen TCP dengan flag FIN dan menerima acknowledgment dari segmen tersebut, maka koneksi TCP pun akan dihentikan.

Three Way Handshake

Proses pembuatan koneksi TCP disebut juga dengan "**Three-way Handshake**". Tujuan metode ini adalah agar dapat melakukan sinkronisasi terhadap nomor urut dan nomor acknowledgement yang dikirimkan oleh kedua pihak dan saling bertukar ukuran TCP Window. Prosesnya dapat digambarkan sebagai berikut:

Host pertama (yang ingin membuat koneksi) akan mengirimkan sebuah segmen TCP dengan flag SYN diaktifkan kepada *host* kedua (yang hendak diajak untuk berkomunikasi).

Host kedua akan meresponsnya dengan mengirimkan segmen dengan acknowledgment dan juga SYN kepada host pertama.

Host pertama selanjutnya akan mulai saling bertukar data dengan host kedua.

TCP menggunakan proses jabat tangan yang sama untuk mengakhiri koneksi yang dibuat. Hal ini menjamin dua *host* yang sedang terkoneksi tersebut telah menyelesaikan proses transmisi data dan semua data yang ditransmisikan telah diterima dengan baik. Itulah sebabnya, mengapa TCP disebut dengan koneksi yang *reliable*.

Daftar Pustaka

1. Tanembaum, A. Computer Network.

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Aplikasi Broadcast Chat

Abstract

Modul ini akan menjelaskan tentang aplikasi broadcast chat.

Kompetensi

Mahasiswa dapat memahami pemrograman untuk membuat aplikasi broadcast chat dengan Java.

Aplikasi Broadcast Chat

Pada modul ini akan dibuat sebuah aplikasi untuk broadcast chat. Aplikasi ini digunakan untuk melakukan chatting menggunakan IP Broadcast. Salah satu IP yang dapat digunakan untuk melakukan pengiriman data kebanyak alamat sekaligus, yang terhubung ke IP broadcast tersebut.

Tahapan-tahapan pembuatan aplikasinya adalah sebagai berikut:

1. Buat sebuah projek baru di Netbeans
2. Beri nama projek tersebut dengan MulticastChat
3. Kemudian Buka class utamanya.

Tambahkan kode program berikut:

```
public class MulticastChat implements Runnable, WindowListener, ActionListener {
  protected InetAddress group;
  protected int port;
  public MulticastChat (InetAddress group, int port) {
    this.group = group;
    this.port = port;
    initAWT ();
  }
}
```

Tambahkan kode program dari di bawah dari constructor

```
public MulticastChat() {
….
}
protected Frame frame;
  protected TextArea output;
  protected TextField input;
  protected void initAWT () {
    frame = new Frame
      ("MulticastChat [" + group.getHostAddress () + ":" + port + "]");
    frame.addWindowListener (this);
    output = new TextArea ();
    output.setEditable (false);
    input = new TextField ();
```

```
    input.addActionListener (this);
    frame.setLayout (new BorderLayout ());
    frame.add (output, "Center");
    frame.add (input, "South");
    frame.pack ();
  }
```

Tambahkan di bawah method initAWT()

```
protected Thread listener;
  public synchronized void start () throws IOException {
    if (listener == null) {
      initNet ();
      listener = new Thread (this);
      listener.start ();
      frame.setVisible (true);
    }
  }
```

Tambahkan di bawah method start()

```
protected MulticastSocket socket;
  protected DatagramPacket outgoing, incoming;
  protected void initNet () throws IOException {
    socket = new MulticastSocket (port);
    socket.setTimeToLive (5);
    socket.joinGroup (group);
    outgoing = new DatagramPacket (new byte[1], 1, group, port);
    incoming = new DatagramPacket (new byte[65508], 65508);
  }
```

Tambahkan di bawah method initNet()

```
public synchronized void stop () throws IOException {
    frame.setVisible (false);
    if (listener != null) {
      listener.interrupt ();
      listener = null;
      try {
        socket.leaveGroup (group);
```

```
    } finally {
      socket.close ();
    }
  }
}
```

Tambahkan dibawah method stop()

```
public void windowOpened (WindowEvent event) {
  input.requestFocus ();
}
 public void windowClosing (WindowEvent event) {
  try {
    stop ();
  } catch (IOException ex) {
    ex.printStackTrace ();
  }
}
```

Tambahkan di bawah method windowClosing()

```
public void actionPerformed (ActionEvent event) {
  try {
    byte[] utf = event.getActionCommand ().getBytes ("UTF8");
    outgoing.setData (utf);
    outgoing.setLength (utf.length);
    socket.send (outgoing);
    input.setText ("");
  } catch (IOException ex) {
    handleIOException (ex);
  }
}
```

Tambahkan di bawah method actionPerformed()

```
protected synchronized void handleIOException (IOException ex) {
  if (listener != null) {
    output.append (ex + "\n");
    input.setVisible (false);
    frame.validate ();
```

```
    if (listener != Thread.currentThread ())
      listener.interrupt ();
    listener = null;
    try {
      socket.leaveGroup (group);
    } catch (IOException ignored) {
    }
    socket.close ();
  }
}
```

Tambahkan di bawah method handleIOException()

```
public void run () {
  try {
    while (!Thread.interrupted ()) {
      incoming.setLength (incoming.getData ().length);
      socket.receive (incoming);
      String message = new String
        (incoming.getData (), 0, incoming.getLength (), "UTF8");
      output.append (message + "\n");
    }
  } catch (IOException ex) {
    handleIOException (ex);
  }
}
```

Tambahkan di bawah method run()

```
public static void main (String[] args) throws IOException {
  InetAddress group = InetAddress.getByName(“239.1.2.3”);
  int port = 2521;
  MulticastChat chat = new MulticastChat (group, port);
  chat.start ();
}
```

Tampilan penuh dari aplikasi Multicast chat adalah sebagai berikut:

```
import java.io.*;
import java.net.*;
```

```
import java.awt.*;
import java.awt.event.*;

public class MulticastChat implements Runnable, WindowListener, ActionListener {
  protected InetAddress group;
  protected int port;

  public MulticastChat (InetAddress group, int port) {
    this.group = group;
    this.port = port;
    initAWT ();
  }

  protected Frame frame;
  protected TextArea output;
  protected TextField input;

  protected void initAWT () {
    frame = new Frame
      ("MulticastChat [" + group.getHostAddress () + ":" + port + "]");
    frame.addWindowListener (this);
    output = new TextArea ();
    output.setEditable (false);
    input = new TextField ();
    input.addActionListener (this);
    frame.setLayout (new BorderLayout ());
    frame.add (output, "Center");
    frame.add (input, "South");
    frame.pack ();
  }

  protected Thread listener;

  public synchronized void start () throws IOException {
    if (listener == null) {
      initNet ();
      listener = new Thread (this);
```

```
    listener.start ();
    frame.setVisible (true);
  }
}

protected MulticastSocket socket;
protected DatagramPacket outgoing, incoming;

protected void initNet () throws IOException {
  socket = new MulticastSocket (port);
  socket.setTimeToLive (5);
  socket.joinGroup (group);
  outgoing = new DatagramPacket (new byte[1], 1, group, port);
  incoming = new DatagramPacket (new byte[65508], 65508);
}

public synchronized void stop () throws IOException {
  frame.setVisible (false);
  if (listener != null) {
    listener.interrupt ();
    listener = null;
    try {
      socket.leaveGroup (group);
    } finally {
      socket.close ();
    }
  }
}

public void windowOpened (WindowEvent event) {
  input.requestFocus ();
}

public void windowClosing (WindowEvent event) {
  try {
    stop ();
  } catch (IOException ex) {
```

```
    ex.printStackTrace ();
  }
}

public void windowClosed (WindowEvent event) {}
public void windowIconified (WindowEvent event) {}
public void windowDeiconified (WindowEvent event) {}
public void windowActivated (WindowEvent event) {}
public void windowDeactivated (WindowEvent event) {}

public void actionPerformed (ActionEvent event) {
  try {
    byte[] utf = event.getActionCommand ().getBytes ("UTF8");
    outgoing.setData (utf);
    outgoing.setLength (utf.length);
    socket.send (outgoing);
    input.setText ("");
  } catch (IOException ex) {
    handleIOException (ex);
  }
}

protected synchronized void handleIOException (IOException ex) {
  if (listener != null) {
    output.append (ex + "\n");
    input.setVisible (false);
    frame.validate ();
    if (listener != Thread.currentThread ())
      listener.interrupt ();
    listener = null;
    try {
      socket.leaveGroup (group);
    } catch (IOException ignored) {
    }
    socket.close ();
  }
}
```

```
  public void run () {
    try {
      while (!Thread.interrupted ()) {
        incoming.setLength (incoming.getData ().length);
        socket.receive (incoming);
        String message = new String
          (incoming.getData (), 0, incoming.getLength (), "UTF8");
        output.append (message + "\n");
      }
    } catch (IOException ex) {
      handleIOException (ex);
    }
  }

  public static void main (String[] args) throws IOException {
    if ((args.length != 1) || (args[0].indexOf (":") < 0))
      throw new IllegalArgumentException
        ("Syntax: MulticastChat <group>:<port>");

    int idx = args[0].indexOf (":");
      InetAddress group = InetAddress.getByName (args[0].substring (0, idx));
    int port = Integer.parseInt (args[0].substring (idx + 1));

    MulticastChat chat = new MulticastChat (group, port);
    chat.start ();
  }
}
```

Daftar Pustaka

1. Hughes, M. et al. Java Network Programming, Second Edition.

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Pengiriman data melalui Protocol UDP

Abstract

Modul ini akan menjelaskan tentang aplikasi pengiriman data melalui protocol UDP.

Kompetensi

Mahasiswa dapat memahami penggunaan dari protocol UDP untuk pengiriman data dan implementasinya pada bahasa pemrograman Java.

Membuat Aplikasi Pengiriman data Menggunakan Protocol UDP

Pada pertemuan kali ini, akan dibahas sebuah aplikasi untuk mengirimkan data stream pada protocol UDP.
Aplikasi pertama digunakan hanya untuk mengirimkan data yang dimasukkan dari keyboard, sedangkan aplikasi ke-dua akan membaca data dari sebuah teks file dan dikirimkan melalui jaringan.

Aplikasi Client 1

```
import java.io.*;
import java.net.*;

class UDPClient {
   public static void main(String args[]) throws Exception
   {
    try {
      String serverHostname = new String ("127.0.0.1");

      if (args.length > 0)
        serverHostname = args[0];

     BufferedReader inFromUser =
      new BufferedReader(new InputStreamReader(System.in));

     DatagramSocket clientSocket = new DatagramSocket();

     InetAddress IPAddress = InetAddress.getByName(serverHostname);
     System.out.println ("Attemping to connect to " + IPAddress +
                         ") via UDP port 9876");

     byte[] sendData = new byte[1024];
```

```
byte[] receiveData = new byte[1024];

System.out.print("Enter Message: ");
String sentence = inFromUser.readLine();
sendData = sentence.getBytes();

System.out.println ("Sending data to " + sendData.length +
                    " bytes to server.");
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacket receivePacket =
  new DatagramPacket(receiveData, receiveData.length);

System.out.println ("Waiting for return packet");
clientSocket.setSoTimeout(10000);

try {
    clientSocket.receive(receivePacket);
    String modifiedSentence =
       new String(receivePacket.getData());

    InetAddress returnIPAddress = receivePacket.getAddress();

    int port = receivePacket.getPort();

    System.out.println ("From server at: " + returnIPAddress +
                        ":" + port);
    System.out.println("Message: " + modifiedSentence);

   }
catch (SocketTimeoutException ste)
   {
    System.out.println ("Timeout Occurred: Packet assumed lost");
}
```

```
      clientSocket.close();
     }
    catch (UnknownHostException ex) {
     System.err.println(ex);
    }
    catch (IOException ex) {
     System.err.println(ex);
    }
  }
}
```

Aplikasi Client ke-2

```
import java.io.*;
import java.net.*;
import java.util.*;

class UDPClient2
{
 private InetAddress IPAddress;
 boolean done;
 boolean keepGoing;

 public UDPClient2(String sHostName)
  {
   String s1;
   ArrayList lines = new ArrayList();
   int size;
   BufferedReader br;

   try {
       IPAddress = InetAddress.getByName(sHostName);
       System.out.println ("Attemping to connect to " + IPAddress +
                        ") via UDP port 9876");
      }
```

```
 catch (UnknownHostException ex)
    {
     System.err.println(ex);
     System.exit(1);
    }


// set up the buffered reader to read from the keyboard
try {
    br = new BufferedReader (new FileReader ("UDPInputFile.txt"));
    s1 = br.readLine();
    while (s1 != null)
     {
      lines.add(s1);
      s1 = br.readLine ();
     }
    size = lines.size();
    System.out.println ("ArrayList lines has size of: " + size);

    done = false;

    DatagramSocket clientSocket = new DatagramSocket();
    for (int i = 0; i < size ; i++)
      {

       byte[] sendData = new byte[1024];

       s1 = (String) lines.get(i);
       sendData = s1.getBytes();

       System.out.println ("Sending data to " + sendData.length +
                           " bytes to server from line " + (i + 1));
       DatagramPacket sendPacket =
         new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

       clientSocket.send(sendPacket);
      }
```

```
done = true;

byte[] receiveData = new byte[1024];

keepGoing = true;

DatagramPacket receivePacket =
    new DatagramPacket(receiveData, receiveData.length);

System.out.println ("Waiting for return packet");
clientSocket.setSoTimeout(10000);

while (keepGoing)
  {
   try {
      clientSocket.receive(receivePacket);
      String modifiedSentence =
          new String(receivePacket.getData());

      //InetAddress returnIPAddress = receivePacket.getAddress();

      //int port = receivePacket.getPort();

      //System.out.println ("From server at: " + returnIPAddress +
      //                    ":" + port);
      System.out.println("Message: " + modifiedSentence);

     }
  catch (SocketTimeoutException ste)
     {
      System.out.println ("Timeout Occurred: Packet assumed lost");
      if (done)
        keepGoing = false;
     }

  }
clientSocket.close();
```

```
    }
  catch (IOException ex)
    {
     System.err.println(ex);
    }
  }


 public static void main(String args[]) throws Exception
  {
   String serverHostname = new String ("127.0.0.1");

   if (args.length > 0)
     serverHostname = args[0];

   new UDPClient2 (serverHostname);

  }
}
```

Aplikasi Server

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
   {
    try
    {
     DatagramSocket serverSocket = new DatagramSocket(9876);

     byte[] receiveData = new byte[1024];
     byte[] sendData  = new byte[1024];

     while(true)
```

```
  {

    receiveData = new byte[1024];

    DatagramPacket receivePacket =
      new DatagramPacket(receiveData, receiveData.length);

    System.out.println ("Waiting for datagram packet");

    serverSocket.receive(receivePacket);

    String sentence = new String(receivePacket.getData());

    InetAddress IPAddress = receivePacket.getAddress();

    int port = receivePacket.getPort();

    System.out.println ("From: " + IPAddress + ":" + port);
    System.out.println ("Message: " + sentence);

    String capitalizedSentence = sentence.toUpperCase();

    sendData = capitalizedSentence.getBytes();

    DatagramPacket sendPacket =
      new DatagramPacket(sendData, sendData.length, IPAddress,
                  port);

    serverSocket.send(sendPacket);

  }

}
 catch (SocketException ex) {
  System.out.println("UDP Port 9876 is occupied.");
  System.exit(1);
 }
```

```
    }
}
```

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Membuat aplikasi FTP

Abstract

Modul ini akan menjelaskan tentang pembuatan aplikasi FTP

Kompetensi

Mahasiswa dapat memahami penggunaan dari protocol FTP untuk pengiriman data dan implementasinya pada bahasa pemrograman Java.

Membuat Aplikasi Pengiriman data Menggunakan FTP

Pada pertemuan kali ini, akan dibahas sebuah aplikasi untuk mengirimkan data stream pada protocol FTP.

Konsep aplikasi FTP ini adalah aplikasi sederhana.untuk mengirimkan data dari client ke server, dan juga untuk mengunduh data dari server ke client.

Berikut ini adalah kode program untuk membuat aplikasi FTP sederhana menggunakan bahasa pemrograman Java.

```
//FTP Client
import java.net.*;
import java.io.*;
import java.util.*;

class FTPClient
{
    public static void main(String args[]) throws Exception
    {
        Socket soc=new Socket("127.0.0.1",5217);
        transferfileClient t=new transferfileClient(soc);
        t.displayMenu();

    }
}
class transferfileClient
{
    Socket ClientSoc;

    DataInputStream din;
    DataOutputStream dout;
    BufferedReader br;
    transferfileClient(Socket soc)
    {
        try
```

```
    {
        ClientSoc=soc;
        din=new DataInputStream(ClientSoc.getInputStream());
        dout=new DataOutputStream(ClientSoc.getOutputStream());
        br=new BufferedReader(new InputStreamReader(System.in));
    }
    catch(Exception ex)
    {
    }
}
void SendFile() throws Exception
{

    String filename;
    System.out.print("Enter File Name :");
    filename=br.readLine();

    File f=new File(filename);
    if(!f.exists())
    {
        System.out.println("File not Exists...");
        dout.writeUTF("File not found");
        return;
    }

    dout.writeUTF(filename);

    String msgFromServer=din.readUTF();
    if(msgFromServer.compareTo("File Already Exists")==0)
    {
        String Option;
        System.out.println("File Already Exists. Want to OverWrite (Y/N) ?");
        Option=br.readLine();
        if(Option=="Y")
        {
            dout.writeUTF("Y");
        }
```

```
        else
        {
            dout.writeUTF("N");
            return;
        }
    }

    System.out.println("Sending File ...");
    FileInputStream fin=new FileInputStream(f);
    int ch;
    do
    {
        ch=fin.read();
        dout.writeUTF(String.valueOf(ch));
    }
    while(ch!=-1);
    fin.close();
    System.out.println(din.readUTF());

}

void ReceiveFile() throws Exception
{
    String fileName;
    System.out.print("Enter File Name :");
    fileName=br.readLine();
    dout.writeUTF(fileName);
    String msgFromServer=din.readUTF();

    if(msgFromServer.compareTo("File Not Found")==0)
    {
        System.out.println("File not found on Server ...");
        return;
    }
    else if(msgFromServer.compareTo("READY")==0)
    {
        System.out.println("Receiving File ...");
```

```
        File f=new File(fileName);
        if(f.exists())
        {
           String Option;
           System.out.println("File Already Exists. Want to OverWrite (Y/N) ?");
           Option=br.readLine();
           if(Option=="N")
           {
              dout.flush();
              return;
           }
        }
        FileOutputStream fout=new FileOutputStream(f);
        int ch;
        String temp;
        do
        {
           temp=din.readUTF();
           ch=Integer.parseInt(temp);
           if(ch!=-1)
           {
              fout.write(ch);
           }
        }while(ch!=-1);
        fout.close();
        System.out.println(din.readUTF());

    }


}

public void displayMenu() throws Exception
{
    while(true)
    {
        System.out.println("[ MENU ]");
```

```
            System.out.println("1. Send File");
            System.out.println("2. Receive File");
            System.out.println("3. Exit");
            System.out.print("\nEnter Choice :");
            int choice;
            choice=Integer.parseInt(br.readLine());
            if(choice==1)
            {
               dout.writeUTF("SEND");
               SendFile();
            }
            else if(choice==2)
            {
               dout.writeUTF("GET");
               ReceiveFile();
            }
            else
            {
               dout.writeUTF("DISCONNECT");
               System.exit(1);
            }
         }
      }
}

// FTP Server

import java.net.*;
import java.io.*;
import java.util.*;

public class FTPServer
{
   public static void main(String args[]) throws Exception
   {
      ServerSocket soc=new ServerSocket(5217);
      System.out.println("FTP Server Started on Port Number 5217");
```

```
        while(true)
        {
            System.out.println("Waiting for Connection ...");
            transferfile t=new transferfile(soc.accept());

        }
    }
}

class transferfile extends Thread
{
    Socket ClientSoc;

    DataInputStream din;
    DataOutputStream dout;

    transferfile(Socket soc)
    {
        try
        {
            ClientSoc=soc;
            din=new DataInputStream(ClientSoc.getInputStream());
            dout=new DataOutputStream(ClientSoc.getOutputStream());
            System.out.println("FTP Client Connected ...");
            start();

        }
        catch(Exception ex)
        {
        }
    }
    void SendFile() throws Exception
    {
        String filename=din.readUTF();
        File f=new File(filename);
        if(!f.exists())
        {
```

```
            dout.writeUTF("File Not Found");
            return;
        }
        else
        {
            dout.writeUTF("READY");
            FileInputStream fin=new FileInputStream(f);
            int ch;
            do
            {
                ch=fin.read();
                dout.writeUTF(String.valueOf(ch));
            }
            while(ch!=-1);
            fin.close();
            dout.writeUTF("File Receive Successfully");
        }
    }

    void ReceiveFile() throws Exception
    {
        String filename=din.readUTF();
        if(filename.compareTo("File not found")==0)
        {
            return;
        }
        File f=new File(filename);
        String option;

        if(f.exists())
        {
            dout.writeUTF("File Already Exists");
            option=din.readUTF();
        }
        else
        {
            dout.writeUTF("SendFile");
```

```
            option="Y";
        }

            if(option.compareTo("Y")==0)
            {
                FileOutputStream fout=new FileOutputStream(f);
                int ch;
                String temp;
                do
                {
                    temp=din.readUTF();
                    ch=Integer.parseInt(temp);
                    if(ch!=-1)
                    {
                        fout.write(ch);
                    }
                }while(ch!=-1);
                fout.close();
                dout.writeUTF("File Send Successfully");
            }
            else
            {
                return;
            }

    }


    public void run()
    {
        while(true)
        {
            try
            {
            System.out.println("Waiting for Command ...");
            String Command=din.readUTF();
            if(Command.compareTo("GET")==0)
```

```
            {
               System.out.println("\tGET Command Received ...");
               SendFile();
               continue;
            }
            else if(Command.compareTo("SEND")==0)
            {
               System.out.println("\tSEND Command Receiced ...");
               ReceiveFile();
               continue;
            }
            else if(Command.compareTo("DISCONNECT")==0)
            {
               System.out.println("\tDisconnect Command Received ...");
               System.exit(1);
            }
            }
            catch(Exception ex)
            {
            }
         }
      }
}
```

Cara menjalankan:

1. Compile dengan 'javac nama_class.java'
2. run program dengan 'java nama_class'

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Pemrograman SIP dengan Java

Abstract

Modul ini akan menjelaskan tentang pemrograman SIP dengan Java.

Kompetensi

Mahasiswa dapat memahami penggunaan dari pemrograman SIP dengan Java, dan dapat mengimplementasikannya.

Pemrograman SIP dengan Java

Sebelum masuk ke dalam pemrograman dari SIP, terlebih dahulu akan kita bahas tentang pengenalan SIP dan penggunaannya pada VoIP.

Pengertian SIP

SIP adalah sebuah signaling protocol yang digunakan untuk pengaturan sesi multimedia. SIP dibangun oleh IETF (Internet Engineering Task Force) dan versi terakhir dari spesifikasinya adalah RFC 3261.

SIP adalah protokol layer aplikasi berbasis teks dan menggunakan Uniform Resource Identifiers (URIs), sebagai contoh “sip:client@voiprakyat.org”, untuk mengalamatkan pemanggil (caller) dan penerima (callee).

Cara kerja SIP menyerupai dengan HTTP, yaitu menggunakan model transaksi request-response. Sebagai contoh seorang client meminta pemanggilan sebuah method di server, dan server mengirim kembali minimal sebuah respon. SIP tidak tergantung dari layer transport yang ada dan mekanisme transaksi membolehkan untuk menggunakan protokol transport yang tidak handal seperti UDP, atau protokol transport yang handal seperti TCP, T/TCP, TCP over TLS/SSL, dan lain-lain.

Sebagaimana yang dijelaskan dalam RFC 3261, SIP mendukung lima fitur untuk komunikasi multimedia, yaitu:

Lokasi pengguna.

Mengetahui di mana untuk mengontak penerima (callee).

Keberadaan pengguna

Mengetahui jika penerima (callee) tersedia dan bersedia untuk berkomunikasi.

Kapabilitas pengguna

Mengetahui format media apa yang digunakan.

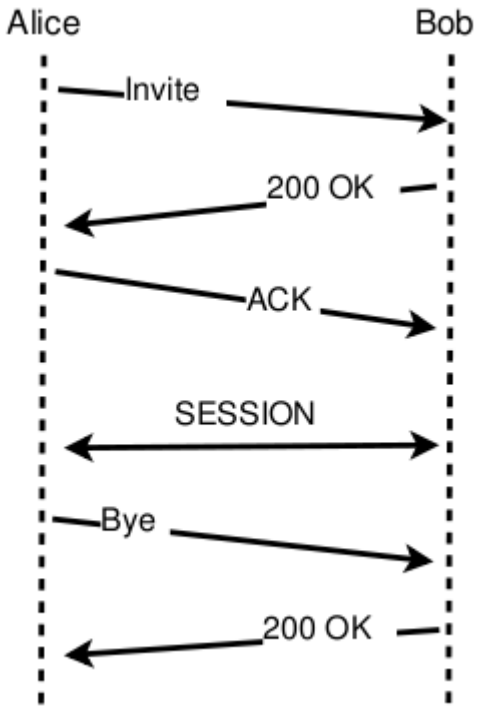
Pembangunan sesi

Membangun sesi untuk komunikasi di antara pemanggil (caller) dan penerima (callee).

Pengaturan Sesi

Memodifikasi atau memutuskan sesi yang sedang berlangsung.

Berikut ini akan digambarkan sebuah skenario sederhana panggilan menggunakan SIP antara Alice dan Bob.



Contoh Aplikasi SIP

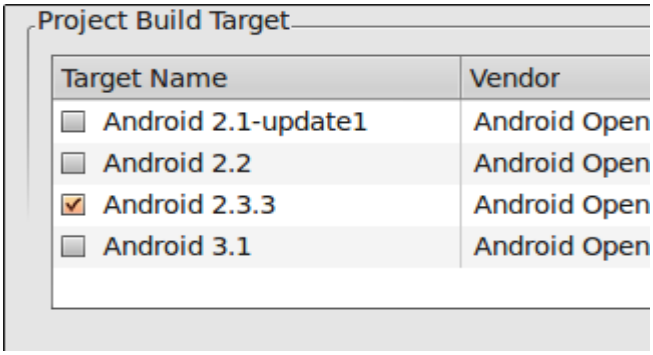
Pada contoh di bawah ini akan dijelaskan sebuah cara untuk melakukan sebuah panggilan menggunakan internet dengan API SIP (Session Initiation Protocol).

Pada program ini akan di buat sebuah aplikasi SIP sederhana yang dapat dijalankan pada Android yang dengan versi 2.3. Karena SIP hanya ada di dalam Android 2.3.

Contoh program ini digunakan untuk menampilkan bagaimana cara memenuhi tiga buah kunci pada SIP yaitu membuat panggilan, menerima panggilan, dan memberikan tanda ke platform android bahwa aplikasi ini siap untuk menerima panggilan masuk dengan SIP.

Cara-caranya adalah sebagai berikut:

1. Buat sebuah aplikasi android dengan nama SipDemo
2. Beri nama kelas dengan WalkieTalkieActivity. Dan package namenya adalah 'com.example.android.sip'.
3. Setting platform Android dengan pilihan Android 2.3.3.



4. Buat sebuah direktori dengan nama drawable di dalam /res. Jika sudah terdapat direktori tersebut, maka tidak perlu dibuat lagi.
5. Buat sebuah file dengan nama btn_record.xml. Dan isikan seperti berikut:

```
<?xml version="1.0"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_pressed="true" android:drawable="@drawable/btn_speak_pressed"
/>
   <item android:state_selected="true" android:drawable="@drawable/btn_speak_selected"
/>
   <item android:drawable="@drawable/btn_speak_normal" />
</selector>
```

6. Selanjutnya salin gambar-gambar berikut ke dalam direktori drawable:

btn_speak_normal.png
btn_speak_pressed.png
btn_speak_selected.png

7. Pindah ke dalam /res/layout dan buat sebuah file dengan nama call_address_dialog.xml.
8. Edit call_address_dialog.xml menjadi seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">

   <TextView
      android:id="@+id/calladdress_view"
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      android:layout_marginLeft="20dip"
      android:layout_marginRight="20dip"
      android:text="@+string/contactAddress"
      android:gravity="left"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <EditText
      android:id="@+id/calladdress_edit"
```

```
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:scrollHorizontally="true"
        android:autoText="false"
        android:capitalize="none"
        android:gravity="fill_horizontal"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

9. Setelah itu buat lagi sebuah file dengan nama pref_layout.xml.
10. Edit pref_layout.xml seperti di bawah ini:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
        android:id="@+id/sip_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SIP Username:"/>
  <EditText
        android:id="@+id/input_sip_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

11. Selanjutnya buat sebuah file lagi dengan nama walkietalkie.xml
12. Edit walkietalkie.xml seperti di bawah ini:

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
```

```
  android:layout_height="fill_parent">
   <RelativeLayout android:padding="12dp"
      android:id="@+id/mainlayout"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

      <TextView
         android:id="@+id/sipLabel"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         android:layout_centerHorizontal="true"
         android:textAppearance="?android:attr/textAppearanceLarge"
      />

      <ToggleButton
         android:layout_height="400dp"
         android:layout_width="400dp"
         android:text="@+string/talk"
         android:id="@+id/pushToTalk"

         android:layout_centerHorizontal="true"
         android:layout_centerVertical="true"
         android:background="@drawable/btn_record"
         android:textOff=""
         android:textOn=""
         android:layout_marginTop="-20dp" />

   </RelativeLayout>
</FrameLayout>
```

13. Kemudian buka /res/values/strings.xml, edit file tersebut seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
   <string name="app_name">SIP Demo</string>
   <string name="contactAddress">SIP Address to contact</string>
```

```
    <string name="talk">Talk</string>
</resources>
```

14. Selanjutnya buatlah sebuah direktori di dalam /res dengan nama xml
15. Setelah itu buatlah sebuah file di dalam /res/xml dengan nama preferences.xml.
16. Edit preferences.xml seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>

<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference
        android:name="SIP Username"
        android:summary="Username for your SIP Account"
        android:defaultValue=""
        android:title="Enter Username"
        android:key="namePref" />
    <EditTextPreference
        android:name="SIP Domain"
        android:summary="Domain for your SIP Account"
        android:defaultValue=""
        android:title="Enter Domain"
        android:key="domainPref" />
    <EditTextPreference
        android:name="SIP Password"
        android:summary="Password for your SIP Account"
        android:defaultValue=""
        android:title="Enter Password"
        android:key="passPref"
        android:password="true" />
</PreferenceScreen>
```

17. Selanjutnya pindah ke dalam /src, buat sebuah class baru di dalam com.example.android.sip dengan nama IncomingCallReceiver.
18. Edit IncomingCallReceiver seperti berikut:

```
package com.example.android.sip;

import android.content.BroadcastReceiver;
import android.content.Context;
```

```
import android.content.Intent;
import android.net.sip.*;
import android.util.Log;

/**
 * Listens for incoming SIP calls, intercepts and hands them off to WalkieTalkieActivity.
 */
public class IncomingCallReceiver extends BroadcastReceiver {
    /**
     * Processes the incoming call, answers it, and hands it over to the
     * WalkieTalkieActivity.
     * @param context The context under which the receiver is running.
     * @param intent The intent being received.
     */
    @Override
    public void onReceive(Context context, Intent intent) {
        SipAudioCall incomingCall = null;
        try {

            SipAudioCall.Listener listener = new SipAudioCall.Listener() {
                @Override
                public void onRinging(SipAudioCall call, SipProfile caller) {
                    try {
                        call.answerCall(30);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            };

            WalkieTalkieActivity wtActivity = (WalkieTalkieActivity) context;

            incomingCall = wtActivity.manager.takeAudioCall(intent, listener);
            incomingCall.answerCall(30);
            incomingCall.startAudio();
            incomingCall.setSpeakerMode(true);
            if(incomingCall.isMuted()) {
```

```
                incomingCall.toggleMute();
            }

            wtActivity.call = incomingCall;

            wtActivity.updateStatus(incomingCall);

        } catch (Exception e) {
            if (incomingCall != null) {
                incomingCall.close();
            }
        }
    }

}
```

Daftar Pustaka

1. http://developer.android.com
2. Purbo, O.W. VoIP Cock Book.
3. *http://opensource.telkomspeedy.com/wiki/index.php/VoIP*

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Pemrograman SIP dengan Java II

Abstract

Modul ini akan menjelaskan tentang pemrograman SIP dengan Java.

Kompetensi

Mahasiswa dapat memahami penggunaan dari pemrograman SIP dengan Java, dan dapat mengimplementasikannya.

Pemrograman SIP dengan Java II

Lanjutan cara-cara membuat program SIP sebagai berikut :

1. Selanjutnya buat sebuah java file lagi dengan nama SipSettings.
2. Edit SipSettings seperti berikut:

```
package com.example.android.sip;

import android.os.Bundle;
import android.preference.PreferenceActivity;

/**
 * Handles SIP authentication settings for the Walkie Talkie app.
 */
public class SipSettings extends PreferenceActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // Note that none of the preferences are actually defined here.
        // They're all in the XML file res/xml/preferences.xml.
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.pref_layout);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

3. Selanjutnya buka file WalkieTalkieActivity dan edit file tersebut menjadi seperti berikut:

```
package com.example.android.sip;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.PendingIntent;
import android.content.DialogInterface;
import android.content.Intent;
```

```
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.*;
import android.net.sip.*;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.ToggleButton;

import java.text.ParseException;

/**
 * Handles all calling, receiving calls, and UI interaction in the WalkieTalkie app.
 */
public class WalkieTalkieActivity extends Activity implements View.OnTouchListener {

    public String sipAddress = null;

    public SipManager manager = null;
    public SipProfile me = null;
    public SipAudioCall call = null;
    public IncomingCallReceiver callReceiver;

    private static final int CALL_ADDRESS = 1;
    private static final int SET_AUTH_INFO = 2;
    private static final int UPDATE_SETTINGS_DIALOG = 3;
    private static final int HANG_UP = 4;


    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.walkietalkie);
```

```
    ToggleButton pushToTalkButton = (ToggleButton) findViewById(R.id.pushToTalk);
    pushToTalkButton.setOnTouchListener(this);

    // Set up the intent filter.  This will be used to fire an
    // IncomingCallReceiver when someone calls the SIP address used by this
    // application.
    IntentFilter filter = new IntentFilter();
    filter.addAction("android.SipDemo.INCOMING_CALL");
    callReceiver = new IncomingCallReceiver();
    this.registerReceiver(callReceiver, filter);

    // "Push to talk" can be a serious pain when the screen keeps turning off.
    // Let's prevent that.
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    initializeManager();
}

@Override
public void onStart() {
    super.onStart();
    // When we get back from the preference setting Activity, assume
    // settings have changed, and re-login with new auth info.
    initializeManager();
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (call != null) {
        call.close();
    }

    closeLocalProfile();

    if (callReceiver != null) {
        this.unregisterReceiver(callReceiver);
```

```
    }
  }

  public void initializeManager() {
    if(manager == null) {
      manager = SipManager.newInstance(this);
    }

    initializeLocalProfile();
  }

  /**
   * Logs you into your SIP provider, registering this device as the location to
   * send SIP calls to for your SIP address.
   */
  public void initializeLocalProfile() {
    if (manager == null) {
      return;
    }

    if (me != null) {
      closeLocalProfile();
    }

    SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
    String username = prefs.getString("namePref", "");
    String domain = prefs.getString("domainPref", "");
    String password = prefs.getString("passPref", "");

    if (username.length() == 0 || domain.length() == 0 || password.length() == 0) {
      showDialog(UPDATE_SETTINGS_DIALOG);
      return;
    }

    try {
      SipProfile.Builder builder = new SipProfile.Builder(username, domain);
```

```
        builder.setPassword(password);
        me = builder.build();

        Intent i = new Intent();
        i.setAction("android.SipDemo.INCOMING_CALL");
        PendingIntent pi = PendingIntent.getBroadcast(this, 0, i, Intent.FILL_IN_DATA);
        manager.open(me, pi, null);


        // This listener must be added AFTER manager.open is called,
        // Otherwise the methods aren't guaranteed to fire.

        manager.setRegistrationListener(me.getUriString(), new SipRegistrationListener() {
            public void onRegistering(String localProfileUri) {
                updateStatus("Registering with SIP Server...");
            }

            public void onRegistrationDone(String localProfileUri, long expiryTime) {
                updateStatus("Ready");
            }

            public void onRegistrationFailed(String localProfileUri, int errorCode,
                    String errorMessage) {
                updateStatus("Registration failed.  Please check settings.");
            }
        });
    } catch (ParseException pe) {
        updateStatus("Connection Error.");
    } catch (SipException se) {
        updateStatus("Connection error.");
    }
}

/**
 * Closes out your local profile, freeing associated objects into memory
 * and unregistering your device from the server.
 */
```

```
public void closeLocalProfile() {
    if (manager == null) {
        return;
    }
    try {
        if (me != null) {
            manager.close(me.getUriString());
        }
    } catch (Exception ee) {
        Log.d("WalkieTalkieActivity/onDestroy", "Failed to close local profile.", ee);
    }
}

/**
 * Make an outgoing call.
 */
public void initiateCall() {

    updateStatus(sipAddress);

    try {
        SipAudioCall.Listener listener = new SipAudioCall.Listener() {
            // Much of the client's interaction with the SIP Stack will
            // happen via listeners.  Even making an outgoing call, don't
            // forget to set up a listener to set things up once the call is established.
            @Override
            public void onCallEstablished(SipAudioCall call) {
                call.startAudio();
                call.setSpeakerMode(true);
                call.toggleMute();
                updateStatus(call);
            }

            @Override
            public void onCallEnded(SipAudioCall call) {
                updateStatus("Ready.");
            }
```

```
        };

        call = manager.makeAudioCall(me.getUriString(), sipAddress, listener, 30);

    }
    catch (Exception e) {
        Log.i("WalkieTalkieActivity/InitiateCall", "Error when trying to close manager.", e);
        if (me != null) {
            try {
                manager.close(me.getUriString());
            } catch (Exception ee) {
                Log.i("WalkieTalkieActivity/InitiateCall",
                        "Error when trying to close manager.", ee);
                ee.printStackTrace();
            }
        }
        if (call != null) {
            call.close();
        }
    }
}

/**
 * Updates the status box at the top of the UI with a messege of your choice.
 * @param status The String to display in the status box.
 */
public void updateStatus(final String status) {
    // Be a good citizen.  Make sure UI changes fire on the UI thread.
    this.runOnUiThread(new Runnable() {
        public void run() {
            TextView labelView = (TextView) findViewById(R.id.sipLabel);
            labelView.setText(status);
        }
    });
}

/**
```

```
     * Updates the status box with the SIP address of the current call.
     * @param call The current, active call.
     */
    public void updateStatus(SipAudioCall call) {
        String useName = call.getPeerProfile().getDisplayName();
        if(useName == null) {
          useName = call.getPeerProfile().getUserName();
        }
        updateStatus(useName + "@" + call.getPeerProfile().getSipDomain());
    }

    /**
     * Updates whether or not the user's voice is muted, depending on whether the button is
pressed.
     * @param v The View where the touch event is being fired.
     * @param event The motion to act on.
     * @return boolean Returns false to indicate that the parent view should handle the touch
event
     * as it normally would.
     */
    public boolean onTouch(View v, MotionEvent event) {
        if (call == null) {
            return false;
        } else if (event.getAction() == MotionEvent.ACTION_DOWN && call != null &&
call.isMuted()) {
            call.toggleMute();
        } else if (event.getAction() == MotionEvent.ACTION_UP && !call.isMuted()) {
            call.toggleMute();
        }
        return false;
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, CALL_ADDRESS, 0, "Call someone");
        menu.add(0, SET_AUTH_INFO, 0, "Edit your SIP Info.");
        menu.add(0, HANG_UP, 0, "End Current Call.");
```

```
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case CALL_ADDRESS:
            showDialog(CALL_ADDRESS);
            break;
        case SET_AUTH_INFO:
            updatePreferences();
            break;
        case HANG_UP:
            if(call != null) {
                try {
                  call.endCall();
                } catch (SipException se) {
                    Log.d("WalkieTalkieActivity/onOptionsItemSelected",
                            "Error ending call.", se);
                }
                call.close();
            }
            break;
    }
    return true;
}

@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case CALL_ADDRESS:

            LayoutInflater factory = LayoutInflater.from(this);
            final View textBoxView = factory.inflate(R.layout.call_address_dialog, null);
            return new AlertDialog.Builder(this)
                    .setTitle("Call Someone.")
                    .setView(textBoxView)
                    .setPositiveButton(
```

```
                    android.R.string.ok, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            EditText textField = (EditText)
                                    (textBoxView.findViewById(R.id.calladdress_edit));
                            sipAddress = textField.getText().toString();
                            initiateCall();

                        }
                    })
                    .setNegativeButton(
                            android.R.string.cancel, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            // Noop.
                        }
                    })
                    .create();

            case UPDATE_SETTINGS_DIALOG:
                return new AlertDialog.Builder(this)
                    .setMessage("Please update your SIP Account Settings.")
                    .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener()
{
                        public void onClick(DialogInterface dialog, int whichButton) {
                            updatePreferences();
                        }
                    })
                    .setNegativeButton(
                            android.R.string.cancel, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            // Noop.
                        }
                    })
                    .create();
        }
        return null;
    }
```

```
    public void updatePreferences() {
        Intent settingsActivity = new Intent(getBaseContext(),
                SipSettings.class);
        startActivity(settingsActivity);
    }
}
```

4. Setelah selesai, cek kembali apakah ada kesalahan dalam penulisan kode program. Jika masih terdapat kesalahan perbaiki kesalahan yang terjadi terlebih dahulu.
5. Jika sudah tidak ada kesalahan, jalankan aplikasi dengan mengklik button run atau dengan menekan ctrl+F11.



6. Jika muncul jendela pilihan, pilih Run As 'Android Application'.

Daftar Pustaka

1. http://developer.android.com
2. Purbo, O.W. VoIP Cock Book.
3. *http://opensource.telkomspeedy.com/wiki/index.php/VoIP*

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Voice Over Internet Protocol (VoIP)

Abstract

Modul ini akan menjelaskan tentang Voice Over Internet Protocol (VoIP).

Kompetensi

Mahasiswa dapat memahami penggunaan dari VoIP.

Pengenalan VoIP

Konsep Dasar VoIP

Teknik dasar *Voice over Internet Protocol* atau yang biasa dikenal dengan sebutan VoIP adalah teknologi yang memungkinkan kemampuan melakukan percakapan telepon dengan menggunakan jalur komunikasi data pada suatu jaringan (*networking*). Sehingga teknologi ini memungkinkan komunikasi suara menggunakan jaringan berbasis IP (*internet protocol*) untuk dijalankan diatas infrastruktur jaringan *packet network.* Jaringan yang digunakan bisa berupa internet atau intranet. Teknologi ini bekerja dengan jalan merubah suara menjadi format digital tertentu yang dapat dikirimkan melalui jaringan IP.

Tujuan pengimplementasian VoIP adalah untuk menekan biaya operasional perusahaan maupun individu dalam melakukan komunikasi jarak jauh (interlokal/SLI). Penekanan biaya itu dapat dilakukan dengan cara memanfaatkan jaringan data yang sudah ada. Sehingga apabila kita ingin membuat jaringan telekomunikasi VoIP tidak perlu membangun infrastruktur baru yang mengeluarkan biaya yang sangat besar. Dengan menggunakan jaringan data yang ada, maka kita melakukan percakapan interlokal maupun internasional hanya dikenakan biaya lokal melalui PSTN.

Internet telephony lebih mengacu pada layanan komunikasi suara (*voice*), *faksimili,* dan *voice messaging applications.* Teknologi ini pada dasarnya mengkonversi sinyal *analog* (suara) ke format *digital* dan kemudian dikompres atau ditranslasikan ke dalam paket-paket IP yang kemudian ditransmisikan melalui jaringan internet.



Gambar 1. Gambaran VoIP

VoIP dalam penerapannya menggunakan sistem jaringan LAN dan *protocol-protocol* VoIP. Standarisasi *protocol* komunikasi pada teknologi VoIP seperti SIP (*session initiation protocol*) dan H.323. (Mochammad Taufiq,2007)

Latar Belakang Teknologi VoIP

Latar belakang dibangunnya teknologi *Voice over Internet Protocol* (VoIP), antara lain:

1. Latar belakang perkembangan teknologi

a. Perkembangan teknologi komunikasi data.

Semakin handalnya kualitas media transmisi–transmisi sinyal elektrik membutuhkan sebuah media transmisi yang normalnya membentuk jalur transmisi. Tipe media transmisi sangat penting untuk ditentukan pada awal pembentukan jalur komunikasi karena mempengaruhi jumlah maksimum bit *(binary digit)* yang dapat ditransmisikan.

b. Perkembangan teknologi sistem kompresi
c. Perkembangan teknologi pemrosesan data

2. Latar belakang perkembangan bisnis.
    a. Persaingan di bidang bisnis telekomunikasi
    b. Tuntutan konsumen akan biaya komunikasi yang murah
3. Efisiensi penggunaan media trasmisi.

Kelebihan VoIP

Adapun kelebihan *Voice over Internet Protocol* (VoIP), adalah sebagai berikut:

1. Biaya lebih rendah untuk sambungan langsung jarak jauh. Penekanan utama dari VoIP adalah biaya. Dengan dua lokasi yang terhubung dengan internet maka biaya percakapan menjadi sangat rendah.
2. Memanfaatkan infrastruktur jaringan data yang sudah ada untuk suara. Berguna jika perusahaan sudah mempunyai jaringan. Jika memungkinkan jaringan yang ada bisa dibangun jaringan VoIP dengan mudah. Tidak diperlukan tambahan biaya bulanan untuk penambahan komunikasi suara.
3. Penggunaan bandwidth yang lebih kecil daripada telepon biasa. Dengan majunya teknologi, penggunaan bandwidth untuk voice sekarang ini menjadi sangat kecil. Tehnik pemampatan data memungkinkan suara hanya membutuhkan sekitar 8kbps bandwidth.
4. Memungkinkan digabung dengan jaringan telepon lokal yang sudah ada. Dengan adanya gateway bentuk jaringan VoIP bisa disambungkan dengan PABX *(Privat Automated Branch exchange)* yang ada dikantor. Komunikasi antar kantor bisa menggunakan pesawat telepon biasa.
5. Berbagai bentuk jaringan VoIP bisa digabungkan menjadi jaringan yang besar. Contoh di Indonesia adalah VoIP Rakyat.

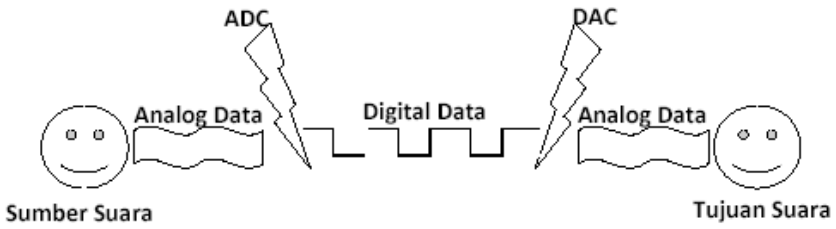
Kekurangan VoIP

Adapun kekurangan *Voice over Internet Protocol* (VoIP), adalah sebagai berikut:

1. Kualitas suara tidak sejernih Telkom. Merupakan efek dari kompresi suara dengan bandwidth kecil maka akan ada penurunan kualitas suara dibandingkan jaringan PSTN konvensional.
2. Ada jeda dalam berkomunikasi. Proses perubahan data menjadi suara, jeda jaringan, membuat adanya jeda dalam komunikasi dengan menggunakan VoIP.
3. Peralatan relatif mahal. Peralatan VoIP yang menghubungkan antara VoIP dengan PABX *(IP telephony gateway)* relatif berharga mahal. Diharapkan dengan makin populernya VoIP ini maka harga peralatan tersebut juga mulai turun harganya.
4. Jika pemakaian VoIP semakin banyak, maka jaringan data yang ada menjadi penuh jika tidak diatur dengan baik.

Cara Kerja VoIP

Pengiriman sebuah sinyal ke *remote destination* dapat dilakukan secara digital, yaitu sebelum dikirim data yang berupa sinyal analog, diubah dulu ke bentuk data digital dengan ADC *(analog to digital converter)*, kemudian ditransmisikan, dan dipenerima dipulihkan kembali menjadi data analog dengan DAC *(digital to analog converter)*. Begitu juga dengan VoIP, digitalisasi voice dalam bentuk packets data, dikirimkan dan dipulihkan kembali dalam bentuk voice dipenerima. Voice diubah dulu kedalam format digital karena lebih mudah dikendalikan dalam hal ini dapat dikompresi, dan dapat diubah ke format yang lebih baik.dan data digital lebih tahan terhadap noise dari pada analog.



Gambar 2. Cara kerja VoIP

Protokol Penunjang VoIP

Protokol-protokol yang menunjang terjadinya komunikasi VoIP adalah :

1. TCP (Transmision Control Protocol)

Dalam mentransmisikan data pada layer transport ada dua protokoll yang berperan yaitu TCP dan UDP. TCP merupakan protokoll yang connection-oriented yang artinya menjaga relaibilitas hubungan komunikasi end-to-end. Konsep dasar cara kerja TCP adalah mengirim dan menerima segment -segment informasi dengan panjang data bervariasi pada suatu

datagram internet. TCP menjamin reliabilitas hubungan komunikasi karena melakukan perbaikan terhadap data yang rusak, hilang atau kesalahan kirim. Hal ini dilakukuan dengan memberikan nomor urut pada setiap oktet yang dikirimkan kembali. Pada sisi penerima, nomor urut tadi berguna untuk mencegah kesalahan urutan data dan duplikasi data. TCP juga memiliki mekanisme flow control dengan cara mencatumkan informasi dalam sinyal ACK mengenai batas jumlah oktet data yang masih boleh ditransmisikan pada setiap segment yang diterima dengan sukses.

Dalam hubungan VoIP, TCP digunakan pada saat signaling, TCP digunakan untuk menjamin setup suatu *call* pada sesi *signaling*. TCP tidak digunakan dalam pengiriman data suara pada VoIP karena pada suatu komunikasi data VoIP penanganan data yang mengalami keterlambatan lebih penting dari pada penanganan paket yang hilang.

2. UDP (User Datagram Protocol)

UDP yang merupakan salah satu protokol utama diatas IP merupakan transport protokol yang lebih sederhana dibandingkan dengan TCP. UDP digunakan untuk situasi yang tidak mementingkan mekanisme reliabilitas. *Header* UDP hanya berisi empat *field* yaitu *source port, destination port, lengt*h dan UDP *checksum* dimana fungsinya hampir sama dengan TCP, namun fasilitas *checksum* pada UDP bersifat pilihan.

UDP pada VoIP digunakan untuk mengirimkan *audio stream* yang dikirimkan secara terus menerus. UDP digunakan pada VoIP karena pada pengiriman *audio streaming* yang berlangsung terus menerus lebih mementingkan kecepatan pengiriman data agar tiba di tujuan tanpa memperhatikan adanya paket yang hilang walaupun mencapai 50% dari jumlah paket yang dikirimkan.

Karena UDP mampu mengirimkan data streaming dengan cepat, maka dalam teknologi VoIP UDP merupakan salah satu protokol penting yang digunakan sebagai header pada pengiriman data selain RTP dan IP. Untuk mengurangi jumlah paket yang hilang saat pengiriman data (karena tidak terdapat mekanisme pengiriman ulang) maka pada teknologi VoIP pengiriman data banyak dilakukan pada *private network*.

3. IP (Internet Protocol)

Internet Protocol didesain untuk interkoneksi sistem komunikasi komputer pada jaringan *packet-switched*. Pada jaringan TCP/IP, sebuah komputer diidentifikasi dengan alamat IP. Tiap-tiap komputer memiliki alamat IP yang unik, masing-masing berbeda satu sama lainnya. Hal ini dilakukan untuk mencegah kesalahan pada transfer data. Terakhir, protocol data akses berhubungan langsung dengan media fisik. Secara umum protokol ini bertugas untuk menangani pendeteksian kesalahan pada saat transfer data. Untuk komunikasi

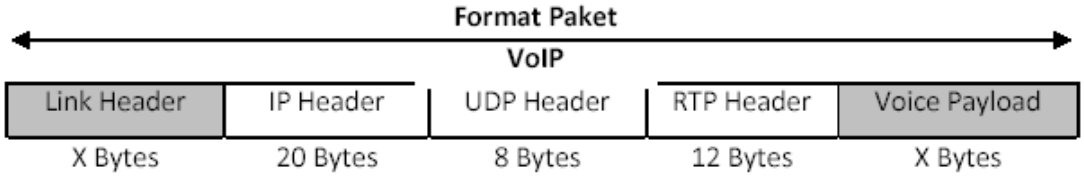
datanya, Internet Protokol mengimplementasikan dua fungsi dasar yaitu addressing dan fragmentasi.

Salah satu hal penting dalam IP dalam pengiriman informasi adalah metode pengalamatan pengirim dan penerima. Saat ini terdapat standar pengalamatan yang sudah digunakan yaitu IPv4 dengan alamat terdiri dari 32 bit. Jumlah alamat yang diciptakan dengan IPv4 diperkirakan tidak dapat mencukupi kebutuhan pengalamatan IP sehingga dalam beberapa tahun mendatang akan diimplementasikan sistim pengalamatan yang baru yaitu IPv6 yang menggunakan sistim pengalamatan 128 bit.

Format Paket VoIP

Tiap paket VoIP terdiri atas dua bagian, yakni header dan *payload* (beban). Header terdiri atas IP header*, Real-time Transport Protocol* (RTP) header*, User Datagram Protocol* header*.*

IP header bertugas menyimpan informasi routing untuk mengirimkan paket-paket ke tujuan. Pada tiap header IP disertakan tipe layanan atau type of service (ToS) yang memungkinkan paket tertentu seperti paket suara diperlakukan berbeda dengan paket yang non real time. UDP header memiliki ciri tertentu yaitu tidak menjamin paket akan mencapai tujuan sehingga UDP cocok digunakan pada aplikasi *real time* yang sangat peka terhadap delay. RTP header adalah header yang dapat dimanfaatkan untuk melakukan *framing* dan segmentasi data *real time.* Seperti UDP, RTP juga mendukung realibilitas paket untuk sampai di tujuan. RTP menggunakan protocol kendali yang mengendalikan RTCP (*real-time transport control protocol*) yang mengendalikan QoS dan sinkronisasi media stream yang berbeda.



Gambar 3. Format Paket VoIP

Untuk link header, besarnya sangat bergantung pada media yang digunakan. Tabel berikut menunjukan ukuran header untuk media yang berbeda dengan metode kompersi G.729. (Mochammad Taufiq,2007)

Tabel 1. Ukuran header pada kompresi G.729

Media	Link Layer Header Size	Bit Rate
Ethernet	14 Byte	29.6 kbps
PPP	6 Byte	26.4 kbps
Frame Relay	4 Byte	25.6 kbps
ATM	5 Byte	42.4 kbps

Komponen VoIP

Ada empat unsur pembentuk jaringan VoIP, yaitu: *User Agent, Proxy, Protocol,* dan CODEC (*Coder-Decoder*). Dari unsur-unsur tersebut jaringan VoIP terbentuk.

1. User Agent

User agent ada yang berupa *software*, ada pula yang berupa *hardware*. *User agent* seperti layaknya telepon yang kita kenal yang berfungsi untuk melakukan pemanggilan atau penerima telepon, baik berasal dari sambungan komputer dengan komputer, komputer dengan IP-*Phone*, komputer dengan PSTN (perlu ditambah alat ATA).

User agent berupa *software (softphone)* yang poluler:

1. Jenis softphone SIP, misalnya Sjphone dan X-lite.
2. Jenis softphone IAX, misalkan Idefisk dan iaxLite.
3. Jenis softphone H.323, misalnya NetMeeting. Umumnya sudah terinstall di sistem operasi Microsoft Windows. Untuk instalasi pertama kali jalankan melalui Run → C:\Program Files\NetMeeting\conf.exe

User Agent berupa perangkat keras antara lain:

1. IP-Phone dengan bentuk seperti telepon biasa pada umumnya. Cara penyambungannya dengan mengkoneksikan ke jaringan komputer berbasis TCP/IP, dapat melalui switch pada jaringan komputer yang telah memiliki fasilitas VoIP.
2. USB Phone, bentuknya seperti telepon seluler. Koneksi yang digunakan melalui USB port dari komputer.
3. Internet Telephony Gateway (ITG), adalah user agent VoIP yang memiliki dua jenis port, yaitu port FXS (terhubung ke telepon biasa) dan FXO (terhubung ke PSTN langsung atau bisa juga melalui PABX).

4. Analog Telephone Adapter (ATA), ini merupakan alat yang digunakan agar telepon rumah kita dapat digunakan menjadi user agent VoIP. Alat ini sama dengan ITG tapi hanya memiliki satu port, yaitu FXS.

Proxy

Karena VoIP akan dijalankan di Internet maka perlu dibentuk jembatan penghubung, jembatan tersebut biasa disebut proxy. Proxy merupakan komponen penengah antar *user agent*, bertindak sebagai server yang menerima *request message* dari *user agent* dan menyampaikan pada *user agent* lainnya.
Contoh-contoh aplikasi proxy VoIP server untuk versi softwitch *Open Source* dan cukup terkenal dan telah teruji keandalannya adalah Asterisk (http://www.asterisk.org), selain itu ada juga OpenSER (http://www.openser.org), SER (http://www.iptel.org/ser/), Yate (http://yate.null.ro). Untuk softwitch yang proprietary (Non Open Source) dapat diperoleh di website Axon (http://www.nch.com.au/pbx/) atau ada pula OnDO SIP Server (http://www.brekeke.com/).

Protocol VoIP

Protokol merupakan sebuah aturan atau *rule* yang harus dipenuhi agar akses komunikasi dalam hal ini komunikasi VoIP dapat melewati jaringan, dalam hal ini *internet*. Di dalam komunikasi VoIP mengenal tiga macam protokol tambahan selain protokol standar *internet*, yaitu TCP/IP, yaitu:

- H.323 yang merupakan protokol yang dikembangkan oleh ITU-T (International Telecomunications Union Telecommunication). Salah satu standar komunikasi pada VoIP menurut rekomendasi ITU-T adalah H.323 (1995-1996). Standar H.323 terdiri dari komponen, protokol dan prosedur yang menyediakan komunikasi multimedia melalui jaringan web based.
- SIP *(session initiation protocol)*
- *IAX (The Inter-Asterisk Exchange) yang merupakan protokol dari asterisk. Sekarang ini, dengan dimotori di Indonesia melalui VoIP Rakyat dimasyarakatkan protokol IAX2, untuk user agent, software Idefisk menggunakan IAX2.*

Codec

Codec adalah kependekan dari compression/ decompression, mengubah signal audio dan dimampatkan ke bentuk data digital untuk ditransmisikan kemudian dikembalikan lagi kebentuk signal audio seperti data yang dikirim. Codec berfungsi untuk penghematan bandwidth di jaringan. Codec melakukan pengubahan dengan cara Sampling signal audio sebanyak 1000 kali per detik. Sebagai gambaran G.711 codec men-sample signal audio 64.000 kali per detik. Kemudian merubahnya ke bentuk data digital dan di mapatkan kemudian ditransmisikan.

Beberapa jenis rata-rata waktu men-sampling VoIP untuk codec yang sering digunakan:

- 64.000 times per second
- 32.000 times per second
- 8.000 times per second

Contoh-contoh codec:

a) Open source dan Free
    a. GSM (codec bit rate 13,2 Kbps)
    b. iLBC (codec bit rate 15,2 Kbps)
    c. G711 (codec bit rate 64 Kbps)
b) Licensed
    a. G729 (codec bit rate 8 Kbps)
    b. G723 (codec bit rate 5,3 Kbps)

SIP (Session Initiation Protocol)

Session Initiation Protocol atau disingkat SIP adalah suatu protokol yang dikeluarkan oleh group yang tergabung dalam Multiparty Multimedia Session Control (MMUSIC) yang berada dalam organisasi Internet Engineering Task Force (IETF) yang di dokumentasikan ke dalam dokumen request for command (RFC) 2543 pada bulan Maret 1999. SIP merupakan protocol yang berada pada layer aplikasi yang mendefinisikan proses awal, pengubahan, dan pengakhiran (pemutusan) suatu sesi komunikasi multimedia. Sesi komunikasi ini termasuk hubungan multimedia, distance learning, dan aplikasi lainnya.

SIP berkarakteristik *client-server*, karena *request* diberikan oleh *client* dan *request* ini dikirimkan ke server. Kemudian server mengolah *request* dan memberikan tanggapan (*response*) terhadap *request* tersebut ke *client*. *Request* dan tanggapan (*response*) terhadap *request* disebut transaksi SIP.

SIP saat ini dipandang sebagai protokol signaling yang baik untuk *Voice over IP* (VoIP). SIP juga dipandang sebagai pengganti dari protokol H.323 yang sebelumnya digunakan juga dalam teknologi VoIP.

Fungsi SIP

Adapun fungsi SIP terdiri dari tiga fungsi, yaitu:

1. Call Initiation
    a. Membangun sebuah sesi komunikasi
    b. Negosiasi media transfer protocol
    c. Mengundang user agent lain untuk bergabung di dalam sesi komunikasi.
2. Call Modification
    a. Bila perlu, SIP dapat memodifikasi sesi komunikasi
3. Call Termination
    a. Menutup sesi komunikasi

Susunan Protocol SIP

SIP adalah *signalling protocol*, bukan media *transfer protocol*, sehingga SIP tidak membawa paket data *voice* atau *video*. Sehingga protokol SIP didukung oleh beberapa protokol lain, diantaranya RSVP (*Resource Reservation Protocol*) yang bertugas untuk melakukan pemesanan pada jaringan, RTP (*real-time transport protocol*) dan RTCP (*real-time transport control protocol*) yang bertugas mentransmisikan media dan mengetahui kualitas layanan, serta SDP (*session description protocol*) bertugas untuk mendeskripsikan sesi media. Secara *default* SIP menggunakan protokol UDP tetapi pada beberapa kasus dapat juga menggunakan TCP sebagai *protocol transport*.

RSVP (Resource Reservation Protocol)

RSVP bekerja pada *layer transport*. Digunakan untuk menyediakan *bandwidth* agar data suara yang dikirimkan tidak mengalami *delay* ataupun kerusakan saat mencapai alamat tujuan *unicast* maupun *multicast*.

RTP (Real-time Transport Protocol)

Protokol RTP menyediakan *transfer* media secara *real-time* pada jaringan paket. Protokol RTP menggunakan protokol UDP dan *header* RTP mengandung informasi kode bit yang

spesifik pada tiap paket yang dikirimkan, hal ini membantu si penerima untuk melakukan antisipasi jika terjadi paket yang hilang.

RTCP (Real-time Transport Control Protocol)

Protokol RTCP merupakan protokol yang megendalikan *transfer* media. Protokol ini bekerja sama dengan protokol RTP. Dalam satu sesi komunikasi, protokol RTP mengirimkan paket RTCP secara periodik untuk memperoleh informasi *transfer* media dalam memperbaiki kualitas layanan.

SDP (Session Description Protocol)

Protokol SDP merupakan protokol yang mendeskripsikan media dalam suatu komunikasi. Tujuan protokol SDP adalah untuk memberikan informasi aliran media dalam satu sesi komunikasi agar penerima yang menerima informasi tersebut dapat berkomunikasi. Hal-hal yang dicakup dalam protokol SDP, antara lain:

- Nama sesi komunikasi dan tujuannya
- Waktu sesi (jika aktif)
- Media dalam sesi komunikasi
- Informasi bagaimana cara menerima media (misal: port, format, dan sebagainya)
- Bandwidth yang di gunakan dalam komunikasi

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London
2. Berbagai Sumber

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Voice Over Internet Protocol (VoIP) Bagian II

Abstract

Modul ini akan menjelaskan tentang Voice Over Internet Protocol (VoIP).

Kompetensi

Mahasiswa dapat memahami penggunaan dari VoIP.

Pengenalan VoIP

Komponen SIP

Dalam hubungannya dengan IP *Telephony*, ada dua komponen yang ada dalam sistem SIP, yaitu:

1. User Agent

User agent terdiri dari dua komponen utama yaitu:

2. User Agent Client (UAC)

User agent ini bertugas memulai sesi komunikasi.

- User Agent Server (UAS)

User agent ini bertugas menerima atau menanggapi sesi komunikasi.

Network Server

Agar user pada jaringan SIP dapat memulai suatu panggilan dan dapat pula di panggil, maka user terlebih dahulu harus melakukan registrasi agar lokasinya dapat diketahui. Registrasi dapat dilakukan dengan mengirimkan pesan REGISTER ke server SIP. Lokasi user dapat berbeda-beda sehingga untuk mendapatkan lokasi user yang aktual diperlukan location server. Pada jaringan SIP, ada tiga tipe network server, yaitu:

Proxy Server

Proxy server adalah komponen penengah antar user agent, bertindak sebagai server dan client yang menerima request message dari user agent dan menyampaikan pada user agent lain. Proxy server bertugas menerjemahkan dan menulis ulang request message sebelum menyampaikan pada user agent tujuan atau proxy lain. Selain itu, proxy server bertugas menyimpan seluruh state sesi komunikasi antara UAC dan UAS.

Redirect server

Komponen ini adalah komponen yang menerima request message dari user agent, memetakan alamat SIP user agent atau proxy server tujuan kemudian menyampaikan hasil pemetaan kembali pada user agent pengirim (UAC). Redirect Server tidak menyimpan state sesi komunikasi antara UAC dan UAS setelah pemetaan disampaikan pada UAC. Redirect server berbeda dengan proxy server, yaitu tidak dapat memulai inisiasi request message. Selain itu, redirect server tidak dapat menerima dan menutup sesi komunikasi.

Registar Server

Registar server adalah komponen yang menerima request message REGISTER. Registar server menyimpan database user untuk otentikasi dan lokasi sebenarnya (berupa IP dan port) agar user yang terdapat dapat duhubungi oleh komponen SIP lainnya (berfungsi sebagai Location Server juga). Registar server dapat dipasangkan dengan Proxy Server.

SIP Request

Telah di bahas diatas, bahwa pesan request pada SIP dikirimkan dari client ke server. Ada enam tipe pesan request, antara lain:

1. INVITE

Pesan ini digunakan untuk memulai suatu komunikasi atau mengundang user agent lain untuk bergabung dalam sesi komunikasi. Message body pesan INVITE berisikan deskripsi media yang dapat digunakan dalam komunikasi.

2. ACK

Pesan ini berfungsi memberitahukan bahwa client telah menerima tanggapan terakhir terhadap INVITE. *Message body* pada pesan ACK dapat membaca deskripsi media yang akan digunakan oleh user yang dipanggil.

3. BYE

Pesan ini dikirim oleh *client* untuk mengakhiri komunikasi.

4. CANCLE

Pesan CANCLE dikirim untuk membatalkan pesan *request* (INVITE).

5. OPTIONS

Pesan ini dikirimkan oleh *client* ke *server* untuk mengetahui tentang informasi kemampuan *server*.

6. REGISTER

Client dapat melakukan registrasi lokasinya dengan mengirimkan pesan REGISTER ke *server* SIP dimana *server* yang menerima pesan REGISTER disebut SIP *register.*

SIP Response

Pesan respon dikirimkan setelah menerima pesan request yang menunjukan status keberhasilan server. Pesan respon didefinisikan dengan tiga angka, angka pertama merupakan kelas respon (seluruhnya ada 6 kelas respon). Angka kedua dan ketiga menunjukan arti dari respon tersebut. Table berikut menunjukan arti kelas respon SIP.

Tabel 2. Response SIP

Kelas Respon	Jenis Respon	Kategori Respon
1xx	Informational Message	Provisional
2xx	Successful Response	Final
3xx	Redirection Response	Final
4xx	Request Response	Final
5xx	Server Failure Response	Final
6xx	Global Failures Response	Final

Pesan respon terbagi atas dua kategori, yakni:

1. Provisional response

Respon ini merupakan respon yang dikirim oleh server untuk menunjukkan proses sedang berlangsung, tapi tidak mengakhiri transaksi SIP.

2. Final response

Respon ini merupakan respon yang mengakhiri transaksi SIP.

SIP Header

Header SIP ditulis dalam format teks, mirip dengan HTTP dan SMTP. Berikut adalah contoh SIP *Header* :

INVITE sip:102@192.168.0.1 SIP/2.0

Via: SIP/2.0/UDP 192.168.1.78:5060;branch=z9hG4bK1385dfa1
From: "line1" <sip:101@192.168.1.78>;tag=as1da29517

To: <sip:102@192.168.0.1>

Contact: <sip:101@192.168.1.78;transport=UDP>

Call-ID: 362711e7794362431acdcd3a36f02774@192.168.1.78

CSeq: 102 INVITE

User-Agent: Asterisk PBX

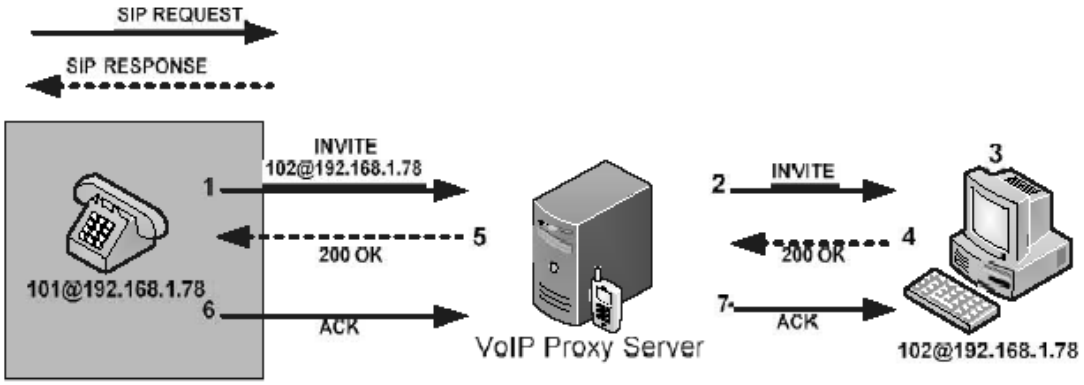
Max-Forwards: 70

Date: Mon, 05 Feb 2007 11:39:27 GMT

Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY

Content-Type: application/sdp…

Cara kerja Protocol SIP



Gambar 4. Cara kerja Protokol SIP

Secara umum langkah demi langkah standar/ prosedur interaksi *internet telephony* yang normal, adalah:

1. Pemanggil akan mengirimkan sinyal INVITe ke proxy server.
2. Proxy server akan meneruskan message INVITE ke tujuan.
3. Bell akan berbunyi di komputer tujuan.
4. Jika tujuan ternyata bersedia menerima, maka tujuan akan mengirimkan message OK ke proxy server.
5. Proxy server akan meneruskan message OK ke pemanggil.
6. Telepon pemanggil akan memberikan message acknowledge (ACK) ke proxy server.
7. Proxy server akan meneruskannya ke mesin tujuan yang benar.
8. Setelah proses pembentukan sambungan ini terbentuk maka hubungan komunikasi suara akan terjadi.

Perbandingan H.323 dan SIP

SIP diciptakan setelah VoIP, dan VoIP diciptakan setelah H.323. Tentu saja, secara umum SIP akan memiliki *feature* yang lebih tepat bagi VoIP dan VoIP secara umum, daripada

H.323. Tabel berikut (Schulzrinne-Rosenberg 1998) memaparkan feature-feature yang ditambahkan pada SIP.

Tabel 3. Kriteria pada H.323 dan SIP

Kriteria	H.323	SIP
Blind Transfer	Bisa	Bisa
Operator-assisted transfer	Tidak	Bisa
Hold	Tidak	Bisa, dengan SDP
Multicast conferences	Bisa	Bisa
Multiunicast conferences	Bisa	Bisa
Bridged conferences	Bisa	Bisa
Forward	Bisa	Bisa
Call park	Tidak	Bisa
Directed call pickup	Tidak	Bisa

Tentu perbedaan antara H.323 dan SIP bukan hanya pada penambahan kriteria. H.323 mengumpulkan protokol yang dipilih untuk dapat mudah beroperasi dengan sistem telefoni yang telah ada; jadi cukup alami bahwa sebagian besar persinyalannya menggunakan model biner yang relatif rumit Sementara, SIP merupakan protokol persinyalan yang ramping dan berbasis teks, yang dioptimasikan agar mudah dikembangkan bersama aplikasi-aplikasi Internet. Tabel berikut [Schulzrinne-Rosenberg1998] memaparkan perbedaan sifat H.323 dan SIP.

Tabel 4. Perbedaan sifat H.323 dan SIP

Kriteria	*H.323*	*SIP*
Kompeksitas	Sangat kompleks	Sederhana
Jumlah transfer pesan sinyal	Banyak	Sedikit
Debugging	Harus menambah tools jika protokol diperluas	Tools sederhana
Perluasan	Bisa diperluas	Mudah diperluas
Perluasan oleh user	ASN.1 – Rumit	Teks – Mudah

Elemen yang harus diperhatikan statenya	Client, GK, MCU, GW, UA, Proxy	-
Pemakaian prosesor	Overhead besar	Overhead kecil
Feature telefoni	Kuat	Kuat
Aplikasi di host	Rumit	**Sederhana**
Ukuran kode	Besar	**Kecil**
Pemakaian memori dinamis	**Besar**	**Kecil hingga sedang**

Praktek membuat Voip

http://192.168.58.44/voip/

- astrerisk
- ippbx ->

instalasi asterisk

1. extract asterisk
   - tar xvfz asterisk-1.8.4.tar.gz
2. buka terminal -> Application > Accessories > Terminal
3. masuk ke dalam direktori di mana asterisk diinstal
   - cd asterisk-1.8.4
4. Cek dependency asterisk
   - ./configure
5. Kompilasi paket
   - make
6. Install paket
   - sudo make install
   - sudo make samples
7. Menjalankan asterisk
   - asterisk -rv

Install libPRI, dahdi-linux, bison

1. Install libpri
   - sudo apt-get install libpri1.4
   - sudo apt-get install libpri-dev

2. Install dahdi-linux

- sudo apt-get install dahdi-linux

instalasi FreePBX

1. Extract FreePBX

- tar xvfz freepbx-2.9.0.tar.gz

2. copy file extract ke dalam direktori /usr/src/

- sudo cp -r freepbx-2.9.0 /usr/src

install beberapa paket pendukung server:

- sudo apt-get install apache2
- sudo apt-get install php5-cli
- sudo apt-get install mysql-server
- sudo apt-get install php5-mysql
- sudo apt-get install php-pear
- sudo apt-get install php-db
- sudo apt-get install php5-gd
- sudo apt-get install curl
- sudo apt-get install sox
- sudo apt-get install libssl-dev
- sudo apt-get install libmysqlclient16-dev
- sudo apt-get install subversion

restart apache

- sudo service apache2 restart

membuat database menggunakan mysql

- masuk ke dalam direktori freepbx
  - cd /usr/src/freepbx-2.9.0/
- membuat database dengan nama asterisk (dengan asumsi asterisk sudah di jalankan dengan perintah './start_asterisk start' di dalam direktori asterisk)
  - mysql -u root -p
  - create database asterisk;
  - create database asterisk_user;
- import sql
  - use asterisk;
  - SQL/newinstall.sql;
  - SQL/cdr_mysql_table.sql;

- ./install_amp

paket yang perlu diinstal sebelum menjalankan './install_amp'
- sudo apt-get install asterisk-prompt-es asterisk-sounds-extra

jalankan install_amp
- sudo ./install_amp --username=root --password=root

rubah settingan amportal.conf
- sudo chmod 777 /etc/amportal.conf

install voip client menggunakan ekiga
-sudo apt-get install ekiga

camera
- sudo apt-get install cheese
import sql
/usr/src/freepbx.../SQL
mysql -u root -p asterisk < newinstall.sql
mysql -u root -p asterisk_user < cdr_mysql_table.sql

Menjalankan asterisk
- sudo service asterisk start
Untuk melihat status asterisk apakah sudah berjalan
- sudo service asterisk status

Jika belum bisa run asterisk beberapa file harus dirubah owner
- sudo chown asterisk. /var/run/asterisk
- sudo chown -R asterisk. /etc/asterisk
- sudo chown -R asterisk. /var/lib/asterisk
- sudo chown -R asterisk. /var/log/asterisk
- sudo chown -R asterisk. /var/spool/asterisk
- sudo chown -R asterisk. /var/www

Pada MySQL terdapat perubahan pada database
- DROP/delete database asterisk_user
- Create database baru dengan nama asteriskcdrdb

- import cdr_mysql_table.sql ke dalam database asteriskcdrdb

membuat account asterisk

adduser asterisk --disabled-password --no-create-home --gecos "asterisk PBX user"

sudo apt-get install asterisk

sudo chmod -R 777 /var/www/html/

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London
2. Berbagai Sumber

MODUL PERKULIAHAN

Pemrograman Jaringan Komputer

Pembuatan OpenSSL

Abstract

Modul ini akan menjelaskan tentang open SSL.

Kompetensi

Mahasiswa dapat memahami penggunaan dari open SSL

OpenSSL

OpenSSL adalah sebuah toolkit kriptograpi yang mengimplementasikan protokol jaringan Secure Socket Layer (SSL v2/v3) dan Transport Layer Security (TLS v1) dan standar kriptograpi yang berhubungan dengan SSL dan TLS.

Program openssl adalah sebuah tool berbasis command line untuk menggunakan berbagai fungsi kriptograpi dari library kripto OpenSSLdi shell.

OpenSSL dapat digunakan sebagai berikut:

1. Membuat dan mengatur private key (kunci privat), public key (kunci umum), dan parameter-parameter
2. Operasi-operasi kriptograpi public key
3. Pembuatan sertifikat X.509, CSR dan CRL
4. Perhitungan dari pencernaan pesan (message digests)
5. Enkripsi dan dekripsi dengan sandi
6. Penanganan S/MIME signed atau encrypted mail
7. permintaan time stamp, pembangkitan dan verifikasi

Hampir pada setiap distribusi Linux sudah terinstall program openssl.

Berikut akan dijelaskan sintaks dasar penggunaan dari openssl pada linux (perintah openssl semuanya dijalankan pada terminal/console)

1. openssl command [command_opts] [command_args]
2. openssl [list-standard-commands | list-message-digest-commands | list-cipher-commands]
3. openssl no-XXX [arbitrary options]

Openssl program menyediakan variasi perintah yang banyak ('command' pada sintaks di atas), masing-masing perintah sering memiliki perbendaharaan option dan argumen ('command_opts' dan 'command_args' di sintaks di atas)

Untuk perintah 'list-standard-commands', 'list-message-digest-commands' dan 'list-cipher-commands' mengeluarkan sebuah daftar (satu entry per baris) nama-nama seluruh standar command, message digest command, atau cipher command, masing-masing, yang tersedia pada utilitas openssl saat ini.

Untuk perintah no-XXX menguji apakah sebuah command nama tertentu tersedia. Jika command bernama XXX tidak ada, akan mengembalikan nilai 0 (success) dan mencetak no-XXX; sebaliknya program akan mengembalikan nilai 1 dan mencetak XXX. Dalam kedua kasus, semua dicetak dalam standar stdout.

Standar-standar command yang ada di openssl

Nama command	Deskripsi
ca	Certificate Authority (CA) Management
ciphers	Cipher suite description determination
crl	Certificate Revocation List (CRL) Management
crl2pkcs7	CRL to PKCS#7 Conversion
dgst	Message Digest Calcluation
dh	Diffie-Hellman Parameter Management.
dsa	DSA data management
dsaparam	DSA Parameter Generation
enc	Encoding with Ciphers
errstr	Error Number to Error String Conversion
dhparam	Generation and Management of Diffie-Hellman Parameters
gendh	Generation of Diffie-Hellman Parameters
gendsa	Generation of DSA Parameters
genrsa	Generation of RSA Parameters
ocsp	Online Certificate Status Protokol Utility
passwd	Generation of hashed passwords
pkcs12	PKCS#12 Data Management
pkcs7	PKCS#7 Data Management
rand	Generate pseudo-random bytes
req	X.509 Certificate Signing Request (CSR) Management
rsa	RSA Data Management
rsautl	RSA utility for signing, verification, encryption, dan decryption
s_client	This implements a generic SSL/TLS client which can establish a transparent connection to a remote server speaking SSL/TLS. It's intended for testing purpose only and provides only rudimentary interface functionality of the OpenSSL ssl library.
s_server	This implements a generic SSL/TLS server which accepts connections from remote clients speaking SSL/TLS. It's intended for testing purposes only and provides only rudimentary interface functionality but internally uses mostly all functionality of the OpenSSL ssl library. It providess both an own command line oriented protocol for testing SSL functions and a simple HTTP response facility to emulate an SSL/TLS-aware webserver
s_time	SSL Connection Timer
sess_id	SSL Session Data Management

smime	S/MIME mail processing
speed	Algorithm Speed Measurement
verify	X.509 Certificate Verification
version	OpenSSL Version Information
x509	X.509 Certificate Data Management

Message Digest Command

Nama Command	Deskripsi
md2	MD2 Digest
md5	MD5 Digest
mdc2	MDC2 Digest
rmd160	RMD-160
sha	SHA Digest
sha1	SHA-1 Digest
sha224	SHA-224 Digest
sha256	SHA-256 Digest
sha384	SHA-384 Digest
sha512	SHA-512 Digest

Encoding dan Cipher Command

base64	Base64 Encoding
bf bf-cbc bf-cfb bf-ecb bf-ofb	Blowfish Cipher
cast cast-cbc	CAST Cipher
cast5-cbc cast5-cfb cast5-ecb cast5-ofb	CAST5 Cipher
des des-cbc des-cfb des-ecb des-ede-cbc des-ede-cfb des-ede-ofb des-ofb	DES Cipher
des3 desx des-ede3 des-ede3-cbc des-ede3-cfb des-ede3-ofb	Triple-DES Cipher
idea idea-cbc idea-cfb idea-ecb idea-ofb	IDEA Cipher

rc2 rc2-cbc rc2-cfb rc2-ecb rc2-ofb	RC2 Cipher
rc4	RC4 Cipher
rc5 rc5-cbc rc5-cfb rc5-ecb rc5-ofb	RC5 Cipher

Beberapa command dapat menggunakan argument, umumnya menggunakan -passin dan -passout untuk password masuk dan keluar masing-masing. Ini membolehkan password untuk diperoleh dari berbagai sumber. Dari kedua pilihan ini mengambil argument tunggal yang formatnya akan dijelaskan di bawah. Jika tidak ada argument password yang diberikan dan sebuah password dibutuhkan kemudian user akan dihadapi dengan tampilan untuk memasukkan password. Beberapa argument adalah sebagai berikut:

Sintak argument	Deskripsi
pass:password	Password yang aktual adalah password. Sejak password dapat dilihat untuk utilitas (seperti 'ps' di Unix) bentuk ini hanya digunakan di mana keamanan tidak penting
env:var	Mendapatkan password dari variabel lingkungan var. Sejak lingkungan proses lain dapat dilihat di platform tertentu pilihan ini harus digunakan dengan perhatian
file:pathname	Baris pertama pathname adalah password. Jika argument pathname yang sama diberikan ke argument -passin dan -passout maka baris pertama akan digunakan untuk output password.
fd:number	Membaca password dari file descriptor number. Ini dapat digunakan untuk mengirim data melalui sebuah pipa sebagai contoh.
stdin	Membaca password dari standard input

Untuk melakukan konfigurasi pada SSL akan dijelaskan secara ringkas dan praktis sebagai berikut:

Asumsi

Server web Anda (Apache 2.0) berada pada mesin dengan nama (hostname/FQDN) sotnec.nurulfikri.com, perlu diperhatikan bahwa hostname mesin Anda harus dapat dilookup (valid terdaftar di DNS). Sebelumnya web server Anda hanya menjalankan service http(port 80), sekarang Anda berkeinginan mengaktifkan service https(port 443) pada web server Anda. Pada web server Anda terdapat 2 virtualhost yang jalan dalam mode HTTPS yang masing masing virtual host memiliki hostname (FQDN) yaitu "mars.nurulfikri.com" dan "bumi.nurulfikri.com".

Langkah-langkah konfigurasi SSL di Apache 2.0 adalah sebagai berikut:

Tahap 1: Setup CA (Certificate Authority)

Agar apache web server Anda dapat menjalankan secure (SSL/TLS encrypted) web server, Anda harus memmiliki sebuah "private key" dan sebuah "certificate" untuk web server Anda. Untuk website commercial, Mungkin Anda dapat membeli certificate yang telah ditanda tangani oleh root CA yang terkenal. Untuk Intranet atau special-purpose , Anda dapat membuat CA sendiri. Ini dapat dilakukan dengan menggunakan tools OpenSSL .

Disini, kita akan membuat sebuah private CA key dan sebuah certificate private CA X.509. Kita juga akan membuat direktori untuk menyimpan certs dan keys:

```
# mkdir /root/CA
# chmod 0770 /root/CA
# cd /root/CA
# openssl genrsa -des3 -out my-ca.key 512
# openssl req -new -x509 -days 3650 -key my-ca.key -out my-ca.crt
# openssl x509 -in my-ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ID, ST=Jawa Barat, L=Depok, O=PT NCI, OU=IT,
CN=sotnec.nurulfikri.com/emailAddress=henry@nurulfikri.com
        Validity
            Not Before: Apr  3 02:10:25 2007 GMT
            Not After : Mar 31 02:10:25 2017 GMT
        Subject: C=ID, ST=Jawa Barat, L=Depok, O=PT NCI, OU=IT,
CN=sotnec.nurulfikri.com/emailAddress=henry@nurulfikri.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:d2:55:bb:2c:54:17:11:8e:15:9d:5f:58:c5:a2:
                    ae:f2:a6:c2:a7:c3:9d:bd:7c:f7:2a:b0:ac:1a:25:
                    3e:4c:4c:ee:c7:27:ed:68:79:85:22:77:7f:46:9a:
                    e3:48:2a:b4:c7:87:f9:03:6f:47:54:c7:31:4f:35:
                    b7:57:b0:02:d0:0e:9c:5a:87:52:58:09:3c:c6:cd:
                    1b:a5:53:b7:4f:97:9f:52:e7:c8:22:3b:fa:0d:3a:
```

```
            6c:98:1b:ae:87:9e:7b:78:b3:c1:d1:87:97:b8:8f:
            88:29:a7:2d:18:60:30:4a:fb:84:3f:c8:e8:8c:bd:
            86:1f:9c:b9:45:a0:1f:be:04:66:37:60:e8:c4:0a:
            e1:fd:04:84:f8:cd:4a:4a:95:5f:c4:6e:20:d7:e0:
            c8:c4:a0:1f:3b:e7:01:7c:16:06:11:b8:b3:1a:65:
            ed:f1:da:7d:76:80:5a:3e:7c:05:4a:4c:da:cf:8a:
            6f:8f:e4:6c:65:ed:ec:4c:61:4f:8e:0b:3c:28:9b:
            fd:47:7c:40:68:c0:7b:74:cc:03:87:7d:ed:29:e0:
            18:b9:01:64:e7:4b:f6:cb:a0:bc:3c:85:e7:4a:4d:
            14:80:16:ea:54:80:a7:00:40:f9:fc:21:4a:c9:45:
            e5:32:c3:f1:3f:d9:bd:ce:e6:86:f5:c9:c4:4a:ea:
            9a:bb
        Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Subject Key Identifier:
        43:F7:93:F4:F5:7A:D1:24:F5:A8:F1:4E:CF:F9:9C:3C:83:73:25:34
        X509v3 Authority Key Identifier:
        keyid:43:F7:93:F4:F5:7A:D1:24:F5:A8:F1:4E:CF:F9:9C:3C:83:73:25:34
        DirName:/C=ID/ST=Jawa Barat/L=Depok/O=PT
NCI/OU=IT/CN=sotnec.nurulfikri.com/emailAddress=henry@nurulfikri.com
        serial:00

        X509v3 Basic Constraints:
        CA:TRUE
  Signature Algorithm: md5WithRSAEncryption
    8f:75:79:48:6b:d0:7e:02:9f:1c:f8:9d:39:e5:5b:6e:c7:1e:
    e9:6e:a3:e4:d1:d3:9b:db:33:62:f6:67:aa:54:90:38:78:61:
    4a:fb:b8:fd:0c:74:d5:ac:08:ff:00:a2:fb:98:b3:56:44:6e:
    7f:8d:77:d9:5d:0f:ee:40:06:62:46:aa:bc:8b:ff:c7:a3:e6:
    83:b9:63:f8:c7:ef:1a:ed:6e:eb:57:e9:d1:2c:f0:12:50:4b:
    7d:5b:c5:22:b0:a0:12:65:93:81:e3:54:f8:85:10:8d:5b:d3:
    9b:56:59:b8:3d:01:07:6e:33:d6:52:c5:8e:b3:c9:48:95:7b:
    26:c2:74:ab:d3:b0:3a:ad:52:bb:69:86:dc:83:40:b6:9a:f0:
    b3:f7:46:5d:ca:99:61:c6:7e:ac:92:c5:a9:3b:80:c1:05:e7:
    5f:7b:24:8d:1e:eb:dc:85:fe:77:c5:99:4b:10:d3:d5:3c:fa:
    24:f2:f5:a4:e6:7d:dd:c7:e2:25:8a:c1:18:59:92:f7:42:77:
    27:1e:3d:36:c3:6a:65:ab:1d:c5:34:a8:ab:66:9e:1f:d6:9c:
```

50:46:76:94:bc:67:27:29:60:55:b3:88:65:58:63:85:c5:f2:
d9:dc:f9:06:d4:27:a2:18:22:65:36:72:80:44:cf:b6:d2:d5:
28:e4:25:35

Catatan

Perintah openssl yang pertama akan membuat CA key. Perintah yang kedua akan membuat X.509 certificate dengan waktu hidup 10 tahun (10-year lifetime).

Perintah ketiga akan menampilkan certificate kita secara lengkap.

Tahap 2: Membuat key dan certificate untuk web server:

Sekarang, kita membuat X.509 certificate dan private key nya untuk web server. Karena kita membuat certificate secara langsung, kita akan membuat key dan certificate, kemudian certificate yang diminta akan di"tandatangani" dengan CA key yang telah kita buat pada tahap 1. Kita juga dapat membuat multiple keys untuk multiple web servers (virtual host) dengan cara yang sama. Satu hal yang perlu dicatat bahwa SSL/TLS private keys untuk web server memerlukan 512 atau 1024 bits. Ukuran key yang lain bisa jadi tidak kompatibel dengan browser-browser tertentu.

Dalam contoh kasus disini kita akan membuat key dan certificate untuk virtualhost mars.nurulfikri.com dan bumi.nurulfikri.com.

```
# openssl genrsa -des3 -out server.key 512
# openssl req -new -key server.key -out server.csr
# openssl x509 -req -in server.csr -out server.crt -sha1 -CA my-ca.crt -CAkey my-ca.key -CAcreateserial -days 3650
# openssl x509 -in server.crt -text -noout
# openssl genrsa -des3 -out bumi-server.key 1024
# openssl req -new -key bumi-server.key -out bumi-server.csr
# openssl x509 -req -in bumi-server.csr -out bumi-server.crt -sha1 -CA my-ca.crt -CAkey my-ca.key -CAcreateserial -days 3650
# openssl x509 -in bumi-server.crt -text -noout
```

Sebaiknya semua file key yang ada di /root/CA dirubah ijin aksesnya menjadi 0400 sbb:

```
# chmod 0400 *.key
# ls /root/CA/
```

Tahap 3: Mengexport/Menyalin Keys dan Certificate, ke dalam direktori konfigurasi apache.

```
# mkdir /etc/httpd/conf/ssl-crt
# mkdir /etc/httpd/conf/ssl-key

# cp my-ca.crt /etc/httpd/conf/ssl.crt
# cp mars-server.crt /etc/httpd/conf/ssl.crt
# cp mars-server.key /etc/httpd/conf/ssl.key
```

cp bumi-server.key /etc/httpd/conf/ssl.key

cp bumi-server.crt /etc/httpd/conf/ssl.crt

Tahap 4: Membuat direktori document root & index.html untuk masing-masing virtualhost

mkdir /var/www/mars

chmod 0775 /var/www/mars

cd /var/www/mars

echo "Hello mars" > index.html

mkdir /var/www/bumi

cd /var/www/bumi

chmod 0775 /var/www/bumi

echo "Hello bumi" > index.html

Tahap 5: Mengkonfigurasi Apache, agar mensuport SSL/TLS

vi /etc/httpd/conf.d/ssl.conf

Isikan dengan file di bawah ini

```
# This is the Apache server configuration file providing SSL support.
# It contains the configuration directives to instruct the server how to
# serve pages over an https connection. For detailing information about these
# directives see <URL:http://httpd.apache.org/docs-2.0/mod/mod_ssl.html>
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#

LoadModule ssl_module modules/mod_ssl.so

#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the HTTPS port
#
Listen 443
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

SSLPassPhraseDialog  builtin

SSLSessionCache        shmcb:/var/cache/mod_ssl/scache(512000)
```

```
SSLSessionCacheTimeout  300

#   Semaphore:
#   Configure the path to the mutual exclusion semaphore the
#   SSL engine uses internally for inter-process synchronization.
SSLMutex default

SSLRandomSeed startup file:/dev/urandom  256
SSLRandomSeed connect builtin

SSLCryptoDevice builtin

##
## SSL Virtual Host Context
##

NameVirtualHost 192.168.1.33:443
<VirtualHost 192.168.1.33:443>

# General setup for the virtual host, inherited from global configuration
DocumentRoot "/var/www/mars"
ServerName mars.nurulfikri.com:443
ErrorLog logs/mars-ssl_error_log
TransferLog logs/mars-ssl_access_log
LogLevel warn
SSLEngine on
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
SSLCertificateFile /etc/httpd/conf/ssl.crt/mars-server.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/mars-server.key
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
SetEnvIf User-Agent ".*MSIE.*" \
        nokeepalive ssl-unclean-shutdown \
```

```
        downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log \
        "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>

<VirtualHost 192.168.1.33:443>

# General setup for the virtual host, inherited from global configuration
DocumentRoot "/var/www/bumi"
ServerName bumi.nurulfikri.com:443
ErrorLog logs/bumi-ssl_error_log
TransferLog logs/bumi-ssl_access_log
LogLevel warn
SSLEngine on
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
SSLCertificateFile /etc/httpd/conf/ssl.crt/bumi-server.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/bumi-server.key
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
SetEnvIf User-Agent ".*MSIE.*" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log \
        "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

Setelah dikonfigurasi apache, restart service apache:

```
# service httpd restart
```

Catatan:

Pada saat Anda restart service apache maka Anda akan ditanyakan pass phrase untuk keys yang telah Anda buat.

Lalu coba akses web server yang telah Anda buat.

Setelah Anda melakukan tahap-tahap di atas maka web server Anda sudah mensupport SSL, namun perlu diingat setiap kali Anda restart atau start service apache Anda maka Anda harus memasukkan pass pharse (password) dari key masing-masing virtual host, menurut sebagian orang hal ini kurang fleksibel namun sebenarnya lebih aman.
Jika Anda tidak ingin memasukkan password setiap merestart ada beberapa tahap yang harus Anda lakukan:

```
 # cd /etc/httpd/conf/ssl.key
# cp bumi-server.key bumi-server.key.org
# less bumi-server.key
# service httpd restart
# cp mars-server.key mars-server.key.org
# openssl rsa -in mars-server.key.org -out
# service httpd restart
```

Instalasi Open SSL pada Linux
instalasi openssl untuk ubuntu
- sudo apt-get install openssl
instalasi openssl untuk centos
- yum install openssl
rubah user menjadi root khusus ubuntu
- sudo -s
membuat direktori CA di dalam root
- mkdir /root/CA
- chmod 770 /root/CA
pindah ke dalam direktori CA
- cd /root/CA
konfigurasi openssl
- openssl genrsa -des3 -out my-ca.key 512
- openssl req -new -x509 -days 3650 -key my-ca.key -out my-ca.crt
- openssl x509 -in my-ca.crt -text -noout
- Catatan
Perintah openssl yang pertama akan membuat CA key. Perintah yang kedua akan membuat X.509 certificate dengan waktu hidup 10 tahun (10-year lifetime).
Perintah ketiga akan menampilkan certificate kita secara lengkap.
Membuat key dan certificate untuk server
- openssl genrsa -des3 -out server.key 512
- openssl req -new -key server.key -out server.csr

- openssl x509 -req -in server.csr -out server.crt -sha1 -CA my-ca.crt -CAkey my-ca.key -CAcreateserial -days 3650

Merubah file akses untuk semua file key

chmod 400 *.key

Menyalin Keys dan Certificate ke dalam konfigurasi apache

- mkdir /etc/httpd/conf/ssl-crt
- mkdir /etc/httpd/conf/ssl-key
- cp my-ca.crt /etc/httpd/conf/ssl-crt
- cp server.crt /etc/httpd/conf/ssl-crt
- cp server.key /etc/httpd/conf/ssl-key

Membuat direktori untuk voip di apache

- mkdir /var/www/voip
- chmod 775 /var/www/voip
- cd /var/www/voip
- echo "Hello Voip" > index.html

Install mod-ssl

- yum install mod_ssl

Konfigurasi file ssl.conf

- vim /etc/httpd/conf.d/ssl.conf

copy dari ubuntu ke virtual box

- cd /home/user
- scp ssl.conf root@IP_ASTERISK:/etc/httpd/conf.d/

restart httpd

- service httpd restart

Daftar Pustaka

1. Graba, J (2013). An Introduction To Network Programming with Java. Springer: London
2. Berbagai sumber