

Praktikum Komunikasi Data & Jaringan Komputer

Auriza Akbar

2017

DAFTAR ISI

i	LAYER JARINGAN KOMPUTER	1
1	INSTALASI WEB SERVER VIRTUAL	3
	Membuat VM Ubuntu Server	3
	Setting <i>port-forwarding</i> VM	3
	Instalasi LAMP (Linux Apache MySQL PHP)	4
	Instalasi aplikasi web Wordpress	5
	Praktikum pekan depan: <i>cabling</i>	5
2	<i>cabling</i> JARINGAN LAN	7
	Standar LAN	7
	<i>Cabling</i>	7
	Alat dan Bahan	7
	Langkah	9
	Penilaian	9
	Praktikum pekan depan: <i>wireless infrastructure</i>	14
	Bahan Bacaan Lanjut	14
3	INFRASTRUKTUR <i>wireless</i>	15
	Frekuensi 2.4 GHz	15
	Keamanan Data	16
	Mode Kerja	17
	<i>Roaming</i> pada <i>Multiple</i> AP	18
	Pengaturan <i>Router</i> TL-WR1043ND	18
	Pengaturan <i>Access Point</i> TL-WA901ND	19
4	PEMROGRAMAN SOKET TCP	21
	Alur Penggunaan Soket TCP	21
	Program Server TCP	21
	Program Klien TCP	23
	Tugas	24
5	PEMROGRAMAN SOKET - PARALELISME	25
	Multi-Processing	25
	Multi-Threading	28
	Hybrid (Prefork - Prethread)	31
	Multithreading (Python)	33
	Tugas	34
6	PROTOKOL LAYER APLIKASI	35
	HTTP	35
	SMTP	37
	POP3	37
	IMAP	38
	Tugas	38
7	APLIKASI JARINGAN	39

Koneksi	39
ping	39
tracert	39
host	39
whois	40
nmap	40
Konfigurasi	41
ifconfig	41
arp	41
netstat	41
route	42
Monitoring	42
tcpdump	42
Wireshark	42
Web-based	43
Bonus Film	43
Tugas	45
ii SIMULASI PACKET TRACER	47
8 PENGENALAN PACKET TRACER	49
Operasi Dasar	49
Koneksi <i>Point-to-Point</i>	49
<i>Switch</i> dan <i>Hub</i>	50
<i>Broadcast</i>	51
Catatan	51
Tugas	51
9 APLIKASI SERVER DAN <i>wireless</i> PADA PACKET TRACER	53
DHCP	53
<i>Multiple Switch</i>	53
<i>Wireless AP</i>	54
Servis lainnya	54
Tugas	54
10 <i>router</i> JARINGAN LOKAL	57
Konfigurasi <i>Router</i> untuk Menghubungkan Dua Jaringan Lokal	57
Tugas	59
11 <i>routing</i> STATIS	61
Menghubungkan Jaringan yang Lokasinya Berjauhan	61
Tugas	63
12 <i>routing</i> DINAMIS: RIPv2	65
<i>Routing</i> Statis vs Dinamis	65
<i>Routing Information Protocol</i> (RIP)	65
<i>Routing</i> Dinamis dengan RIPv2	66
Konfigurasi <i>router</i> R1	66
Konfigurasi <i>router</i> R2	67

	Konfigurasi <i>router</i> R3	68
	Pengujian	69
	Tugas	69
	Referensi	69
13	<i>routing</i> DINAMIS: OSPF	71
	<i>Open Shortest Path First</i> (OSPF)	71
	<i>Routing</i> Dinamis dengan OSPF	72
	Konfigurasi <i>router</i> R1	72
	Konfigurasi <i>router</i> R2	73
	Konfigurasi <i>router</i> R3	74
	Pengujian	75
	Tugas	75
	Referensi	75

DAFTAR TABEL

Tabel 1	Aturan <i>port forwarding</i>	3
Tabel 2	Standar <i>wireless</i> IEEE 802.11	15

DAFTAR GAMBAR

Gambar 1	Setting <i>port forwarding</i> di VirtualBox	4
Gambar 2	Halaman utama Wordpress	5
Gambar 3	Standar T568B	7
Gambar 4	Alat dan bahan	8
Gambar 5	Kabel UTP kategori 5E	8
Gambar 6	Pengelupasan sarung kabel	9
Gambar 7	Kabel yang telah dikelupas ujungnya	10
Gambar 8	Susunan kabel T568B	10
Gambar 9	Kabel yang sudah diluruskan	11
Gambar 10	Sesuaikan dengan panjang konektor	11
Gambar 11	Pemotongan kabel dengan crimping tool	12
Gambar 12	Memasukkan kabel ke konektor RJ-45	12
Gambar 13	Memasukkan kabel ke konektor RJ-45	13
Gambar 14	<i>Crimp</i>	13
Gambar 15	Tes	14
Gambar 16	<i>Channel</i> 2.4 GHz (sumber: Wikipedia)	16
Gambar 17	Contoh pemilihan <i>channel</i> 2.4 GHz (sumber: MetaGeek)	16
Gambar 18	Contoh pengaturan <i>channel</i> yang baik dan buruk	17
Gambar 19	<i>Wireless access point</i>	17
Gambar 20	<i>Wireless router</i>	18
Gambar 21	<i>Wireless roaming</i>	18
Gambar 22	TCP socket call	22
Gambar 23	Prefork	26
Gambar 24	Prethread	28
Gambar 25	Layer jaringan TCP/IP (sumber: Wikipedia)	35
Gambar 26	Email telah terkirim	37
Gambar 27	nmap	40
Gambar 28	Wireshark	43
Gambar 29	Cacti	44
Gambar 30	Nagios	44
Gambar 31	<i>Point-to-point</i>	49
Gambar 32	<i>Switch Cisco 2960 48-port</i>	50
Gambar 33	<i>Switch</i>	50
Gambar 34	<i>Hub</i>	50
Gambar 35	Contoh <i>router</i> : Cisco 2801	57
Gambar 36	<i>Router</i> LAN	58
Gambar 37	<i>Router</i> untuk menghubungkan jaringan dengan lokasi yang berjauhan	61

Daftar Gambar

Gambar 38	Kabel <i>fiber optic single-mode</i>	62	
Gambar 39	<i>Routing</i> dinamis dengan RIPv2	66	
Gambar 40	Protokol <i>routing</i> dinamis (sumber: Cisco)		71
Gambar 41	Routing dinamis dengan OSPF	72	

Bagian I

LAYER JARINGAN KOMPUTER

INSTALASI WEB SERVER VIRTUAL

Tujuan praktikum ini adalah agar mahasiswa dapat menginstal aplikasi web pada *virtual private server* (VPS) berbasis Linux. VPS menyediakan fleksibilitas untuk menginstal aplikasi server apa saja, tidak terbatas hanya pada aplikasi web berbasis PHP. Layanan VPS banyak tersedia (misal: Niagahoster, DigitalOcean, dan Amazon) dengan harga yang bervariasi sesuai dengan spesifikasi server yang ditawarkan.

MEMBUAT VM UBUNTU SERVER

Telah tersedia *virtual disk image* (VDI) instalasi Ubuntu Server 16.04 di direktori `/opt/vm`. Salin file `ubuntu-server.vdi` tersebut ke direktori *home* anda. Kemudian, buat VM baru pada VirtualBox dengan tipe “Ubuntu 64-bit”. Gunakan *virtual disk* yang sudah disalin tadi.

PS: bagi yang ingin mencoba instalasi Ubuntu Server dari awal, silahkan unduh Ubuntu Server dan ikuti petunjuknya di sini.

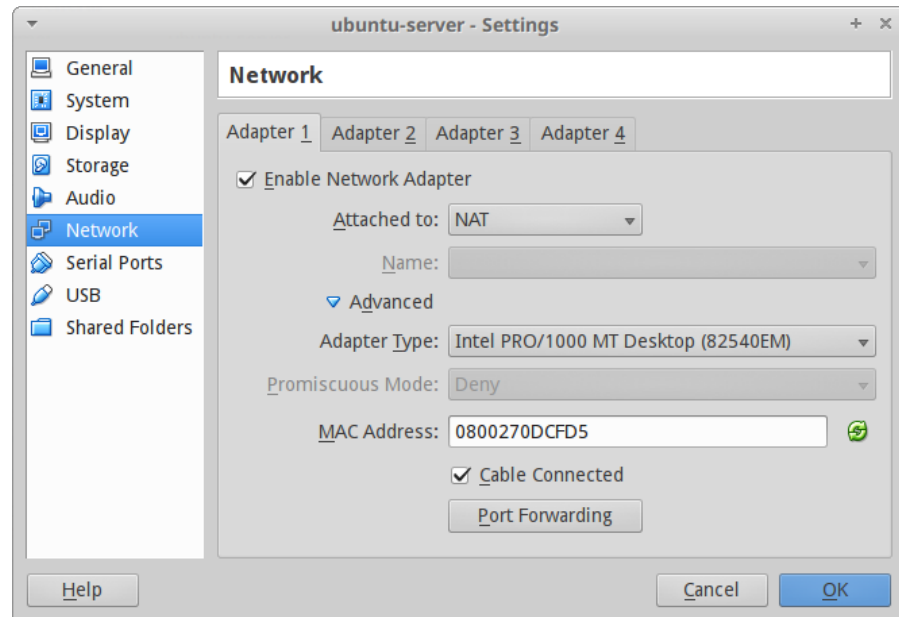
SETTING *port-forwarding* VM

Tujuannya adalah agar VM bisa diakses dari luar melalui alamat IP *host* (*localhost*). Masuk ke ‘*Settings -> Network -> Advanced -> Port Forwarding*’ dan tambahkan dua aturan berikut.

Tabel 1: Aturan *port forwarding*

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
http	TCP		8888		80
ssh	TCP		2222		22

Dengan demikian, jika kita mengakses `localhost:8888` di *host*, maka akan diteruskan ke `localhost:80` di *guest* (VM).



Gambar 1: Setting *port forwarding* di VirtualBox

Setelah semuanya beres, jalankan VM dengan login *username student* dan *password student*.

INSTALASI LAMP (LINUX APACHE MYSQL PHP)

```
# instal SSH
```

```
sudo apt update
```

```
sudo apt install ssh
```

Setelah terinstal SSH, kita bisa mengakses VM secara *remote*. Buka terminal di *host* untuk login *remote* ke *port 2222*.

```
# akses remote dari host
```

```
ssh student@localhost -p 2222
```

```
# instal Apache, MySQL, PHP
```

```
sudo apt install apache2
```

```
sudo apt install mysql-server
```

```
sudo apt install php
```

```
sudo apt install libapache2-mod-php
```

```
sudo apt install php-mysql
```

```
sudo apt install php-gd php-mcrypt php-mbstring php-xml php-ssh2
```

```
sudo service apache2 restart
```

Cek instalasi Apache dengan membuka laman <http://localhost:8888>.

INSTALASI APLIKASI WEB WORDPRESS

```
# buat database dan user untuk Wordpress
```

```
mysql -u root -p -v -e "
CREATE DATABASE wordpress;
CREATE USER wordpress IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON wordpress.* TO wordpress;"
```

```
# download Wordpress
```

```
wget "https://wordpress.org/latest.tar.gz"
```

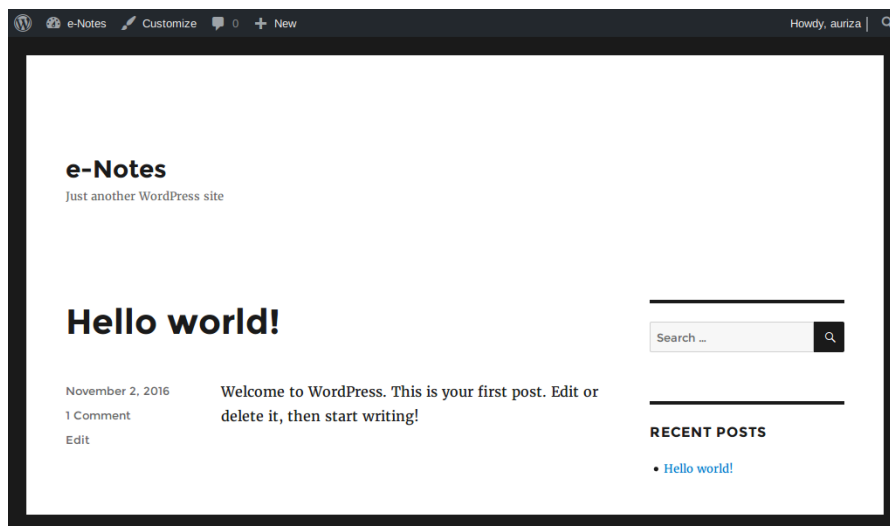
```
# ekstrak ke direktori web
```

```
sudo tar -xzf latest.tar.gz -C /var/www/html
```

```
# ubah kepemilikan ke user www-data (webserver)
```

```
sudo chown -R www-data:www-data /var/www/html/wordpress
```

Buka laman <http://localhost:8888/wordpress> untuk meneruskan instalasi.



Gambar 2: Halaman utama Wordpress

PRAKTIKUM PEKAN DEPAN: *cabling*

Setiap praktikan membawa:

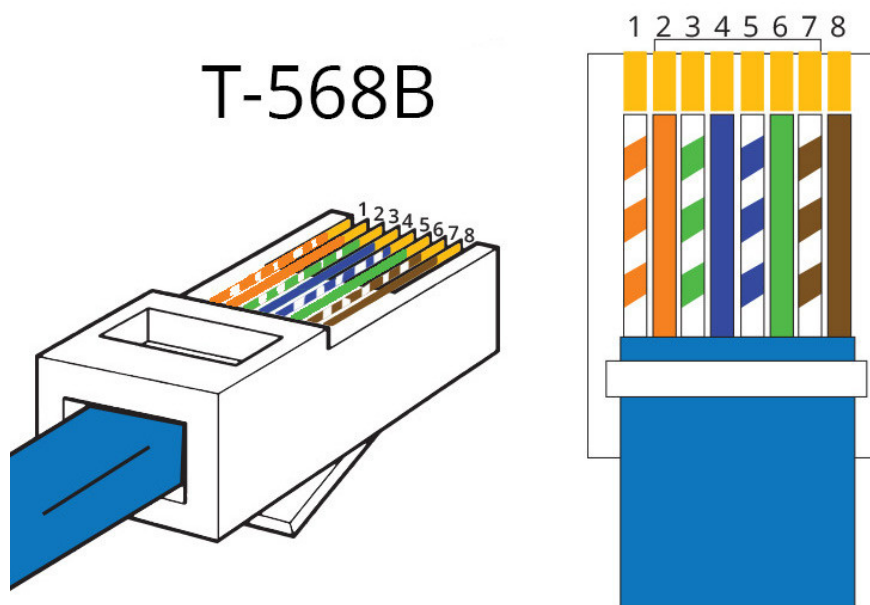
- kabel LAN Cat 5 (minimal 1 meter)
- konektor RJ-45 3 buah
- gunting
- *crimping tool* (jika ada)

Bagi yang mau kabel LAN bekas gratis, silahkan ke lab NCC.

CABLING JARINGAN LAN

Tujuan: mahasiswa dapat membuat infrastruktur jaringan kabel.

STANDAR LAN

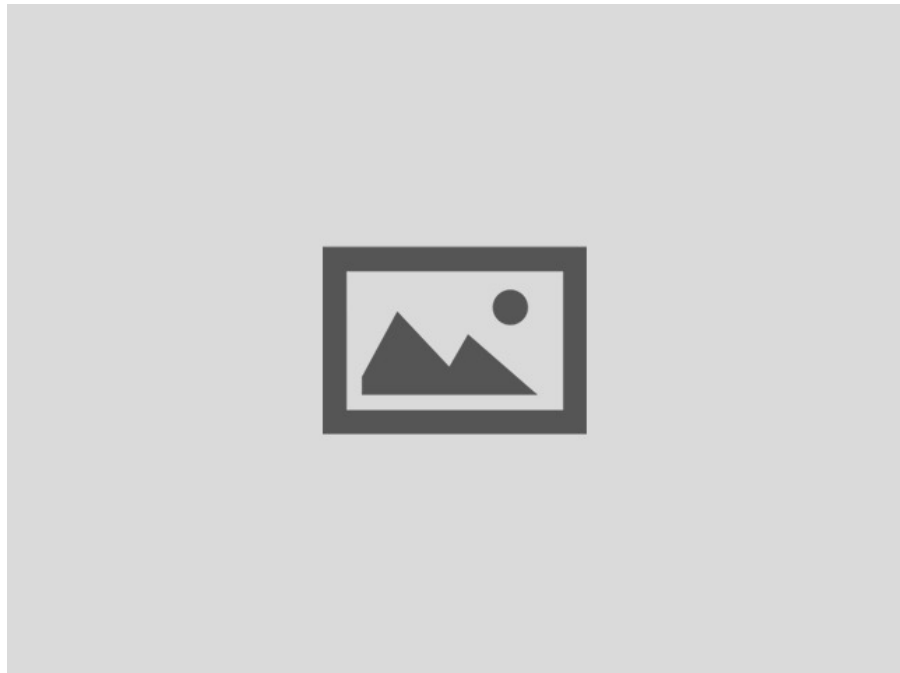


Gambar 3: Standar T568B

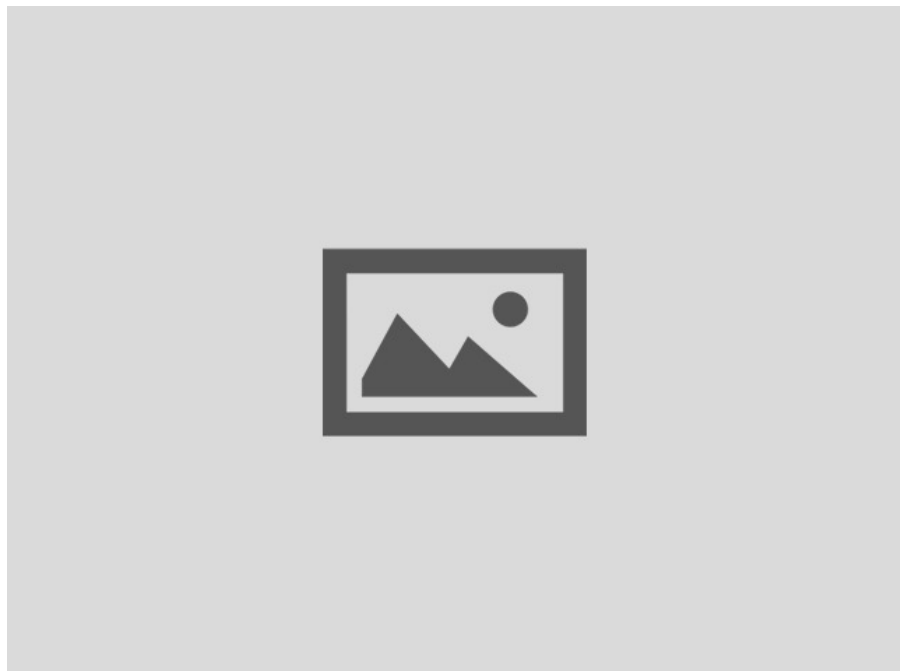
cabling

Alat dan Bahan

- Kabel UTP Cat5E
- Crimping tool
- Pengupas kabel atau gunting
- Konektor RJ-45 2 buah
- *Cable tester*



Gambar 4: Alat dan bahan



Gambar 5: Kabel UTP kategori 5E

Langkah

- Kelupas sarung kabel dengan *peeler* atau gunting

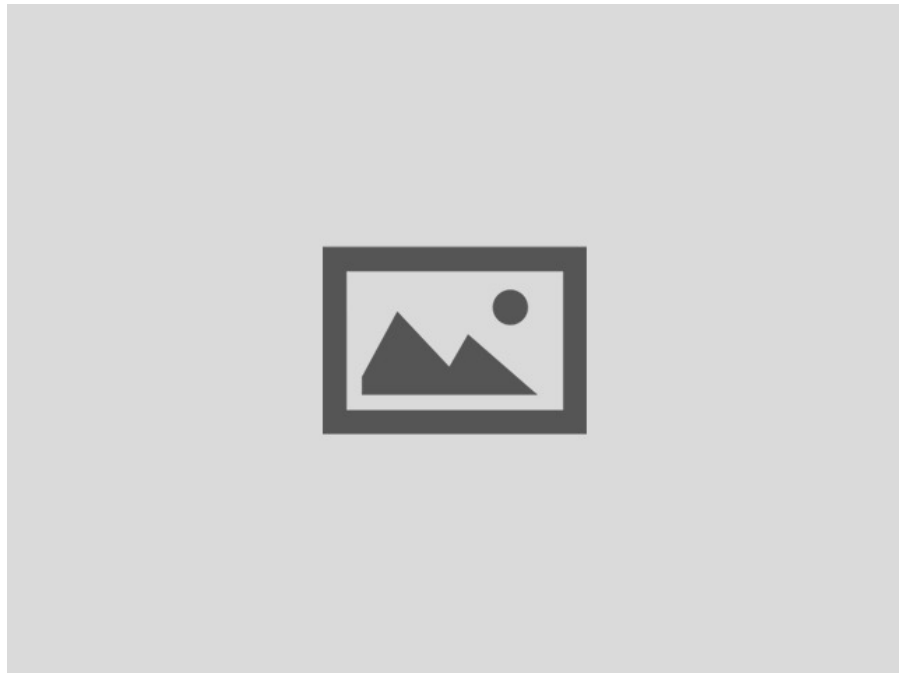


Gambar 6: Pengelupasan sarung kabel

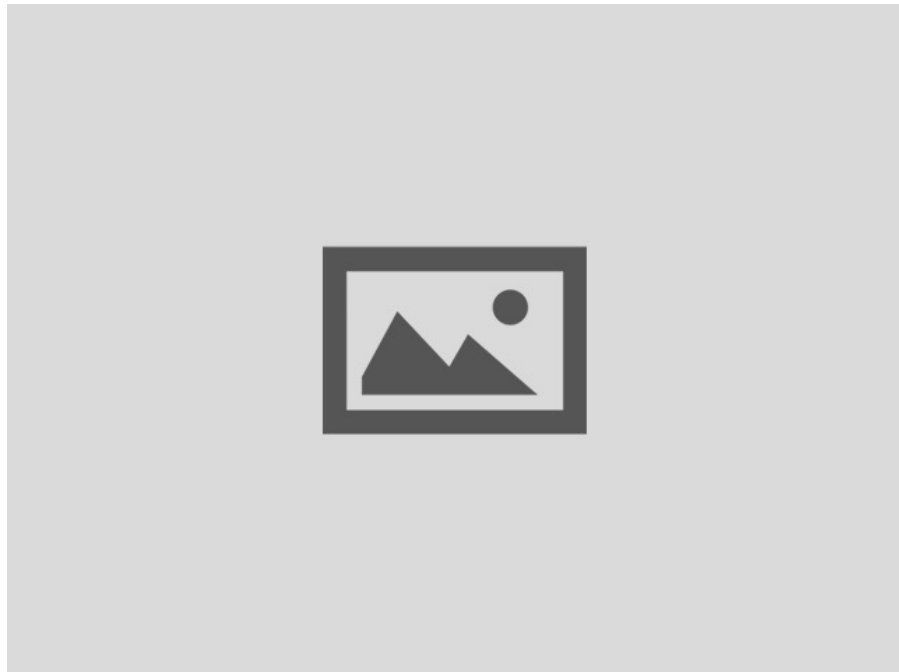
- Lepaskan pilinan dan susun kabel dengan standar T568B
- Luruskan semua kabel
- Potong ujungnya, sesuaikan dengan panjang konektor. Jaket kabel harus masuk dan terjepit oleh konektor.
- Masukkan semua kabel ke dalam konektor
 - pastikan ujung kabel masuk sampai ujung konektor
 - pastikan jaket kabel terjepit oleh konektor
- *Crimp* dengan *crimping tool*
- Ulangi lagi pada ujung satunya
- Tes dengan *cable tester*

Penilaian

- *Crimping* rapi dan semua kabel tersambung: **100**
- *Crimping* tidak rapi: **-10** per konektor
- Kabel nomor 1, 2, 3, atau 6 tidak tersambung: **mengulang**
- Kabel selain nomor di atas tidak tersambung: **-10** per kabel



Gambar 7: Kabel yang telah dikelupas ujungnya



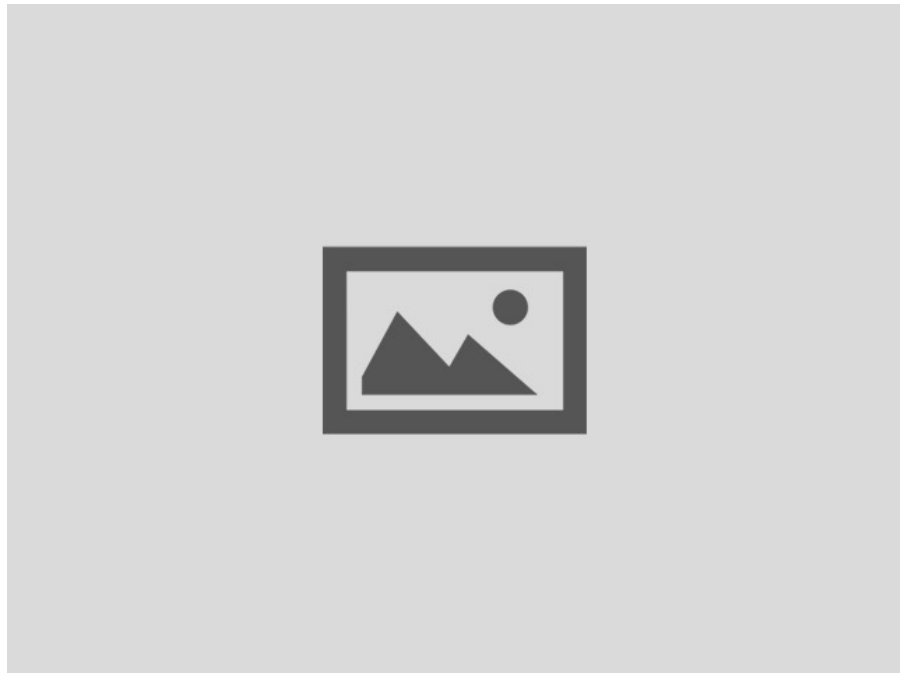
Gambar 8: Susunan kabel T568B



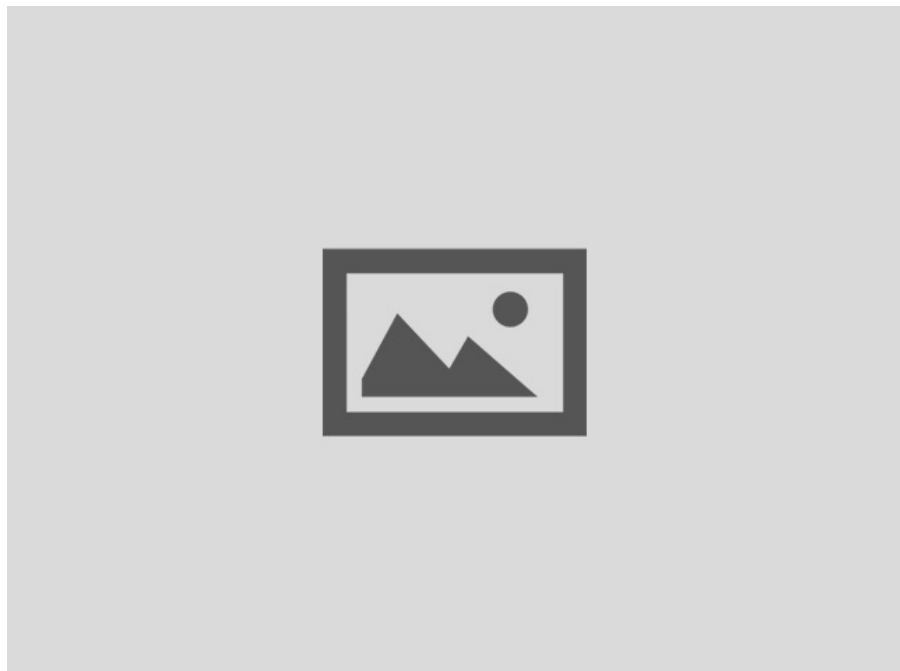
Gambar 9: Kabel yang sudah diluruskan



Gambar 10: Sesuaikan dengan panjang konektor



Gambar 11: Pemotongan kabel dengan crimping tool



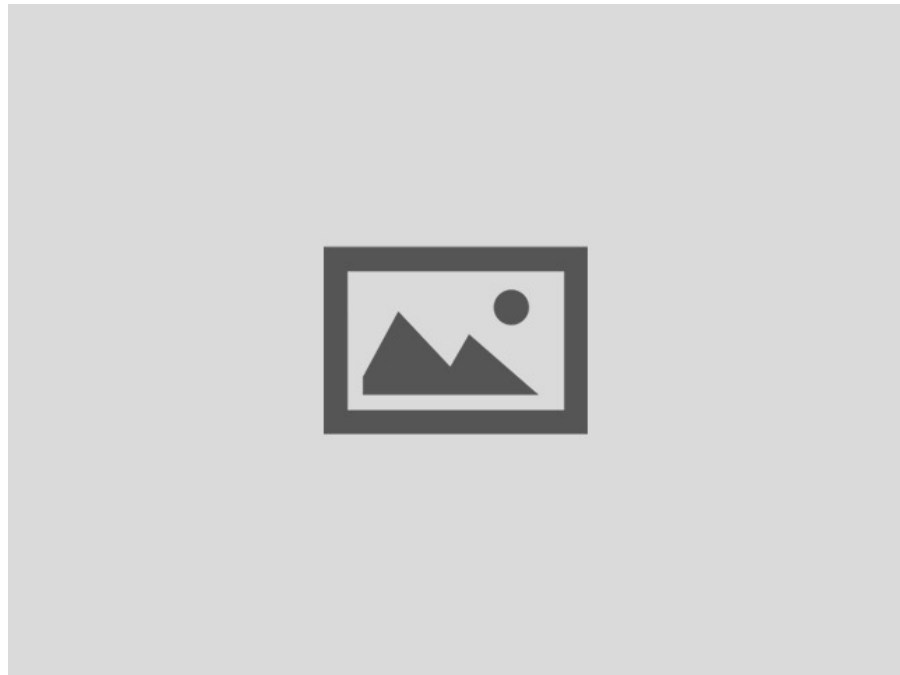
Gambar 12: Memasukkan kabel ke konektor RJ-45



Gambar 13: Memasukkan kabel ke konektor RJ-45



Gambar 14: *Crimp*



Gambar 15: Tes

PRAKTIKUM PEKAN DEPAN: *wireless infrastructure*

Setiap kelompok membawa:

- kabel LAN (*straight*)
- laptop

BAHAN BACAAN LANJUT

- Terrible Terminations: How even perfectly good Ethernet cable and connectors, put together badly, can result in lousy performance.

INFRASTRUKTUR *WIRELESS*

Tujuan: mahasiswa dapat membuat infrastruktur jaringan *wireless*.

Standar *wireless* LAN yang paling banyak dipakai adalah standar IEEE 802.11 (Wi-Fi). Wi-Fi beroperasi pada pita frekuensi 2.4 GHz dan 5 GHz. Standar Wi-Fi yang masih banyak dipakai adalah 802.11n yang mendukung *dual band* dan antena *multiple-input multiple-output* (MIMO) hingga 4 buah. Standar Wi-Fi terbaru di pasaran adalah 802.11ac yang mendukung MIMO hingga 8 buah.

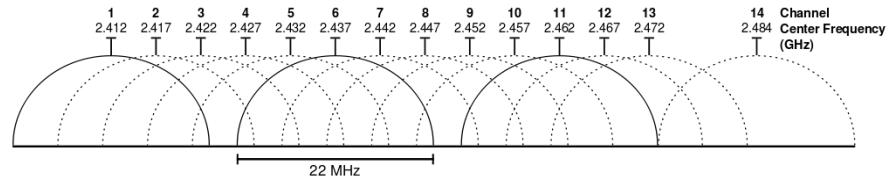
Tabel 2: Standar *wireless* IEEE 802.11

802.11	Frekuensi (GHz)	Bandwidth (MHz)	Stream rate (Mbps)	MIMO	Range (m)
–	2.4	22	1–2	–	20
a	5.0	20	6–54	–	35
b	2.4	22	1–11	–	35
g	2.4	20	6–54	–	38
n	2.4/5.0	20,40	7.2–150	4	70
ac	5.0	20,40,80,160	7.2–867	8	35

FREKUENSI 2.4 GHz

Standar 802.11b/g/n menggunakan frekuensi 2.4 GHz pada rentang spektrum 2400–2500 MHz. Rentang tersebut dibagi menjadi 14 *channel* yang lebarnya sekitar 20 MHz. Pusat tiap *channel* terpisah 5 MHz, dimulai dari *channel* 1 dengan pusat 2412 MHz. Untuk instalasi beberapa perangkat WiFi, perlu dipilih *channel* yang tidak *overlap* untuk meminimalkan interferensi. Contoh *non-overlap channel* yang banyak dipakai adalah *channel* 1, 6, dan 11 (<http://www.metageek.com/training/resources/why-channels-1-6-11.html>).

Lebar *channel* dapat diubah menjadi 40 MHz untuk meningkatkan *data rate* dua kali lipat. Namun penggunaannya tidak disarankan pada jaringan bersama, karena akan sulit menghindari *overlap* dengan *channel* lainnya.



Gambar 16: *Channel 2.4 GHz* (sumber: Wikipedia)

Berikut adalah contoh instalasi beberapa perangkat WiFi pada jaringan bersama. Pemilihan *channel* perlu diperhatikan untuk menghindari interferensi yang menyebabkan penurunan kinerja hingga 60%. Untuk memilih *channel*, kita harus melihat *channel* mana saja yang masih kosong dan tidak terlalu *crowded*. Gunakan aplikasi inSSIDer pada Windows atau Wifi Analyzer pada Android.

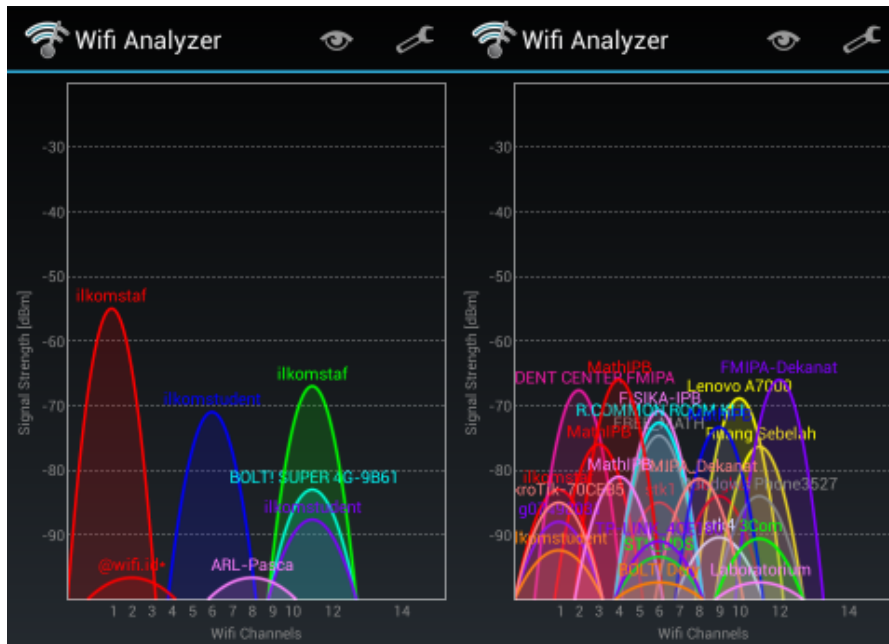


Gambar 17: Contoh pemilihan *channel 2.4 GHz* (sumber: MetaGeek)

KEAMANAN DATA

Berikut jenis enkripsi yang bisa dipakai untuk melindungi data yang dikirim via *wireless*:

- *Unsecured*
- WEP: ARC4
- WPA: TKIP
- WPA2: AES

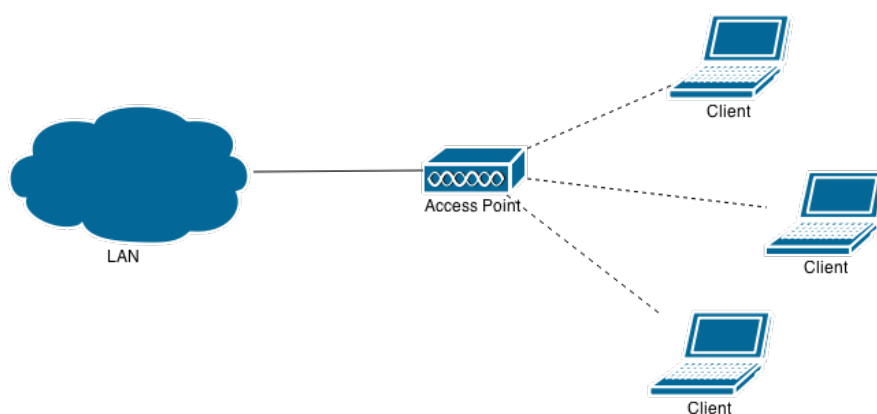


Gambar 18: Contoh pengaturan *channel* yang baik dan buruk

Keamanan terbaik adalah dengan WPA2 dan menonaktifkan fitur WPS (<http://www.metageek.com/training/resources/wireless-security-basics.html>).

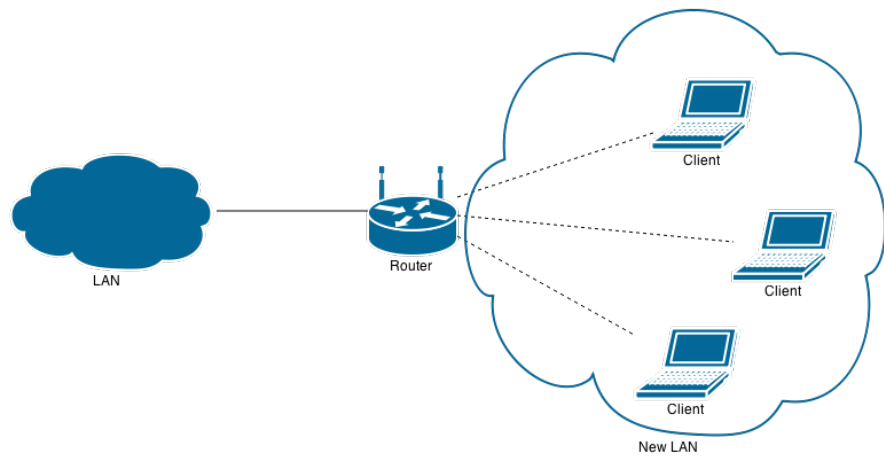
MODE KERJA

- *Access Point* (AP): untuk memperluas jaringan LAN yang sudah ada untuk klien *wireless*.



Gambar 19: *Wireless access point*

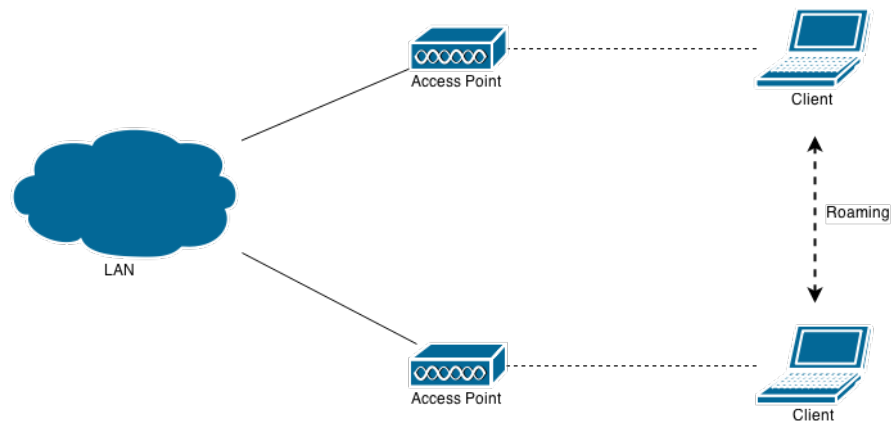
- *Router*: untuk membuat jaringan *wireless* baru



Gambar 20: *Wireless router*

roaming PADA *multiple* AP

Untuk memanfaatkan fitur *roaming*, gunakan SSID dan pengaturan keamanan yang sama pada setiap AP yang dipasang. Jika klien berpindah tempat dan sinyal AP lemah, klien dapat berpindah ke AP lain secara otomatis tanpa melakukan koneksi ulang.



Gambar 21: *Wireless roaming*

PENGATURAN *router* TL-WR1043ND

Simulator: http://static.tp-link.com/resources/simulator/TL-WR1043ND_UN_2.0/Index.htm atau <https://www.dd-wrt.com/demo/>.

- Nyalakan *device* lalu tekan tombol *reset* sampai semua lampu menyala (~ 10 detik)
- Colokkan kabel *straight* dari komputer ke *port* LAN (kuning)

- Colokkan kabel *straight* dari jaringan ke *port* WAN (biru)
- Akses ke <http://192.168.0.1> dengan *user:admin* dan *password:admin*
- “Quick Setup”
 - Network Name (SSID):
 - Region: **Indonesia**
 - Security: **WPA2-PSK**
 - Password:
 - More Advanced:
 - * Width: **20 MHz**
 - * Channel: **1, 6, atau 11**
- “System Tools”
 - Time setting
 - * Time zone: **GMT +7**
 - * Klik **Get GMT**
 - Password
 - * Ganti *username* dan *password*

PENGATURAN *access point* TL-WA901ND

Simulator: http://static.tp-link.com/resources/simulator/TL-WA901ND_V3/Index.htm

- Nyalakan *device* lalu tekan tombol *reset* sampai semua lampu menyala (~ 10 detik)
- Colokkan kabel *straight* dari komputer ke port LAN
- Akses ke <http://192.168.0.254> dengan *user:admin* dan *password:admin*
- “Quick Setup”
 - Country/Region: **Indonesia**
 - Change the login account: **Yes**
 - * Ganti *username* dan *password*
 - Mode: **Access Point**
 - Wireless
 - * SSID:
 - * Channel: **1, 6, atau 11**
 - * Security: **WPA2-PSK**
 - * Password:
 - Network type: **Smart IP (DHCP)**
 - Finish
- “Wireless”
 - Channel width: **20 MHz**
- Colokkan kabel *straight* dari jaringan ke port LAN

4

PEMROGRAMAN SOKET TCP

Tujuan: mahasiswa dapat membuat program server/klien TCP.

Soket adalah abstraksi untuk komunikasi jaringan. Pada sistem operasi UNIX, semua *resource*, termasuk komunikasi jaringan, diabstraksikan sebagai *file*. Jadi, anggap saja soket adalah sebuah *file* yang bisa dibuka, ditutup, dibaca, dan ditulis. Soket diidentifikasi dengan sebuah *integer* yang disebut *socket descriptor* (*pointer* ke struktur data yang berisi deskripsi soket). Struktur data tersebut berisi: jenis soket, alamat dan port lokal yang dipakai, dan alamat dan port *remote* yang akan menerima komunikasi dari soket.

Penggunaan soket terbagi menjadi dua:

- Soket pasif: server, menunggu koneksi masuk
- Soket aktif: klien, memulai koneksi ke server

ALUR PENGGUNAAN SOKET TCP

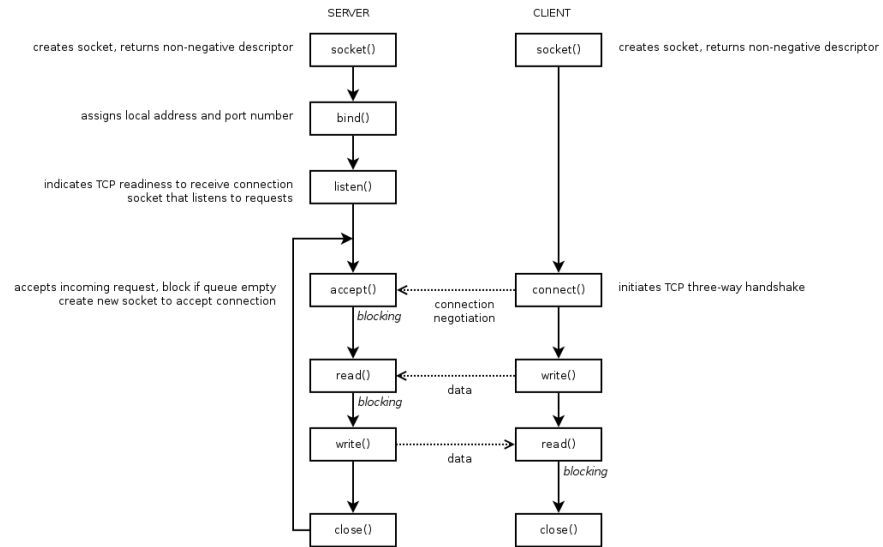
PROGRAM SERVER TCP

`server.c`

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT    2000
#define QUEUE   5

int main()
{
    int          server;
    int          client;
    struct sockaddr_in sv_addr = {AF_INET, htons(PORT), {INADDR_ANY}};
```



Gambar 22: TCP socket call

```

struct sockaddr_in  cl_addr;
char                welcome[] = "+OK Welcome, type your message:\n";
char                goodbye[] = "+OK Message accepted, goodbye!\n";
char                data[80]  = {0};

server = socket(AF_INET, SOCK_STREAM, 0);
setsockopt(server, SOL_SOCKET, SO_REUSEADDR, &(int){1}, sizeof (int));
bind(server, (struct sockaddr*)&sv_addr, sizeof sv_addr);
if (listen(server, QUEUE) == 0)
    puts("listening...");

while (1) {
    client = accept(server, (struct sockaddr*)&cl_addr, &(socklen_t){0});

    write(client, welcome, sizeof welcome);

    memset(data, 0, sizeof data);
    read(client, data, sizeof data);
    printf("[%s:%d]: %s", inet_ntoa(cl_addr.sin_addr), ntohs(cl_addr.sin_port), data);

    write(client, goodbye, sizeof goodbye);

    close(client);
}

close(server);
  
```



```

    return 0;
}

```

Jalankan program `server`, lalu gunakan `nc` sebagai klien untuk melakukan koneksi ke server.

```
nc localhost 2000
```

Coba buat dua sesi klien yang mengakses server secara bersamaan, apa yang terjadi? Mengapa demikian? Bagaimana agar server bisa melayani banyak klien sekaligus?

Dengan membuat program server menjadi *multithreaded*, server bisa melayani beberapa klien sekaligus. Tambahkan direktif OpenMP berikut di atas blok `while`. Kompilasi dengan menambahkan *flag* `-fopenmp`.

```
#pragma omp parallel private(client, cl_addr, data) num_threads(16)
```

PROGRAM KLIEN TCP

client.c

```

#include <stdio.h>
#include <unistd.h>
#include <arpa/inet.h>

#define HOST    "127.0.0.1"
#define PORT    2000

int main()
{
    int                server;
    struct sockaddr_in sv_addr = {AF_INET, htons(PORT), {inet_addr(HOST)}};
    char               mesg[80];
    char               data[80];

    server = socket(AF_INET, SOCK_STREAM, 0);
    connect(server, (struct sockaddr*)&sv_addr, sizeof sv_addr);

    read(server, mesg, sizeof mesg);
    printf("%s", mesg);

    fgets(data, sizeof data, stdin);
    write(server, data, sizeof data);

    read(server, mesg, sizeof mesg);
    printf("%s", mesg);
}

```

```
        close(server);  
        return 0;  
    }
```

Jalankan program `server`, lalu jalankan program `client` di atas.

TUGAS

Buat program klien untuk koneksi ke server web `http://xubuntu.org` (`162.213.33.66`) dan menampilkan keluarannya ke layar.

Petunjuk: kirimkan *request* HTTP berikut ke server dan tampilkan balasannya.

```
GET / HTTP/1.0  
Host: xubuntu.org
```

PEMROGRAMAN SOKET - PARALELISME

Tujuan: mahasiswa akan dapat membuat program soket multikoneksi pada Linux

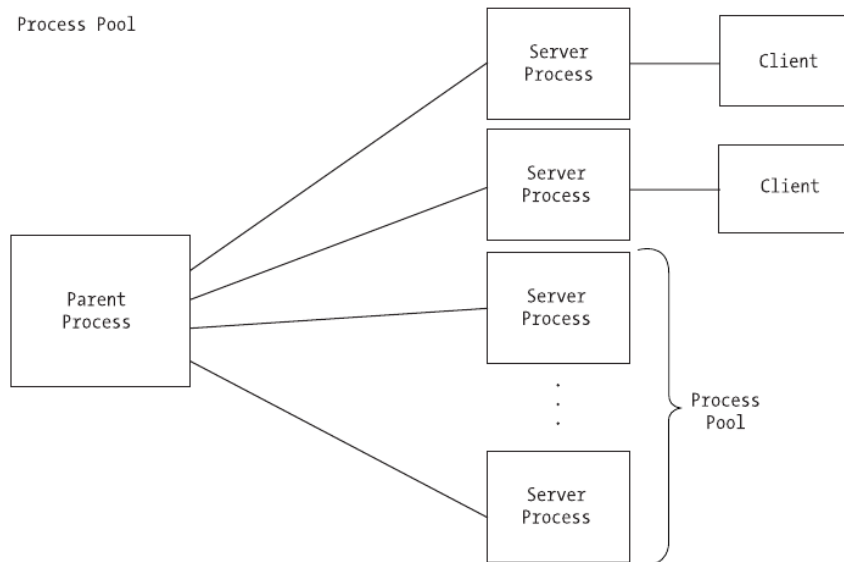
Pada praktikum sebelumnya, telah dibuat aplikasi server yang hanya bisa melayani satu klien tiap satu waktu. Aplikasi server ini kurang berguna pada dunia nyata. Klien lain harus menunggu lama untuk dapat dilayani oleh server. Untuk mengatasinya, aplikasi server biasanya menggunakan arsitektur paralel, baik menggunakan multiprocessing, multithreading, maupun kombinasi keduanya.

Contoh aplikasi server pada dunia nyata adalah Apache HTTP server, yang memiliki beberapa MPM (multi-processing module). Dua MPM yang paling banyak digunakan:

- **prefork**: menggunakan multiprocessing, memerlukan lebih banyak memori, memiliki banyak fitur, biasa dipakai bersama dengan modul PHP
- **worker**: hibrida, gabungan antara multiprocessing dan multithreading, lebih ringan tetapi fiturnya terbatas

MULTI-PROCESSING

- satu proses melayani satu klien
- menggunakan **fork()** untuk menduplikasi proses
- keuntungan:
 - implementasinya sederhana
 - jika salah satu proses child crash, proses lain tidak terpengaruh
- kerugian:
 - komunikasi antar-proses tidak efisien
 - memerlukan banyak resource untuk membuat proses baru
- proses parent harus memanggil **wait()** untuk menunggu sampai proses child selesai, supaya tidak terjadi proses zombie
- contoh implementasi: prefork (proses child dibuat terlebih dahulu sebanyak jumlah tertentu)



Gambar 23: Prefork

- kompilasi dan coba jalankan beberapa klien sekaligus yang mengakses server

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/wait.h>

#define NPROCESS 3

int main(int argc, char *argv[])
{
    int port = atoi(argv[1]);           /* port server */
    int i, server_id = 0, client_id = 0;

    /* 1. Membuat socket */
    int listen_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_socket == -1) {
        fprintf(stderr, "Error creating socket.\n");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in server_addr = {           /* alamat server */
        .sin_family = AF_INET,
        .sin_port = htons(port),
        .sin_addr.s_addr = INADDR_ANY
    };
    
```

```

/* 2. Memberikan alamat ke soket */
if (bind(listen_socket, (struct sockaddr*) &server_addr, sizeof (struct sockaddr_in))
    fprintf(stderr, "Error binding.\n");
    exit(EXIT_FAILURE);
}

/* 3. Listen --> soket pasif */
if (listen(listen_socket, 5) == -1) {
    fprintf(stderr, "Error listening.\n");
    exit(EXIT_FAILURE);
}
printf("[SERVER] listening...\n");

/* 4. Prefork sebanyak jumlah proses yang diinginkan */
for (i = 0; i < NPROCESS; i++) {
    printf("[SERVER] creating child-%d\n", i);
    if (fork() == 0) { /* jika proses anak, maka akan meneri
        server_id = i;
        while (1) {
            struct sockaddr_in client_addr; /* alamat klien */
            socklen_t client_addr_size = sizeof (struct sockaddr_in);

            /* 5. Membuat soket untuk menerima koneksi dari klien */
            int accept_socket = accept(listen_socket, (struct sockaddr*) &client_addr
            if (accept_socket == -1) {
                fprintf(stderr, "Error accepting accept_socket.\n");
                exit(EXIT_FAILURE);
            }
            client_id++;

            /* cetak alamat klien */
            printf("[SERVER] child-%d accepting client-%d.%d from %s:%d\n", server_id

            /* kirim pesan ke klien */
            char server_msg[] = "+OK Welcome, type your message.\n";
            write(accept_socket, server_msg, sizeof server_msg);

            /* baca pesan dari klien */
            char client_msg[100] = {0};
            read(accept_socket, client_msg, sizeof client_msg);
            printf("[CLIENT] client-%d.%d said: %s", server_id, client_id, client_msg

            /* balas pesan ke klien */
            char server_reply[] = "+OK Message accepted. Bye!\n";
            write(accept_socket, server_reply, sizeof server_reply);

```

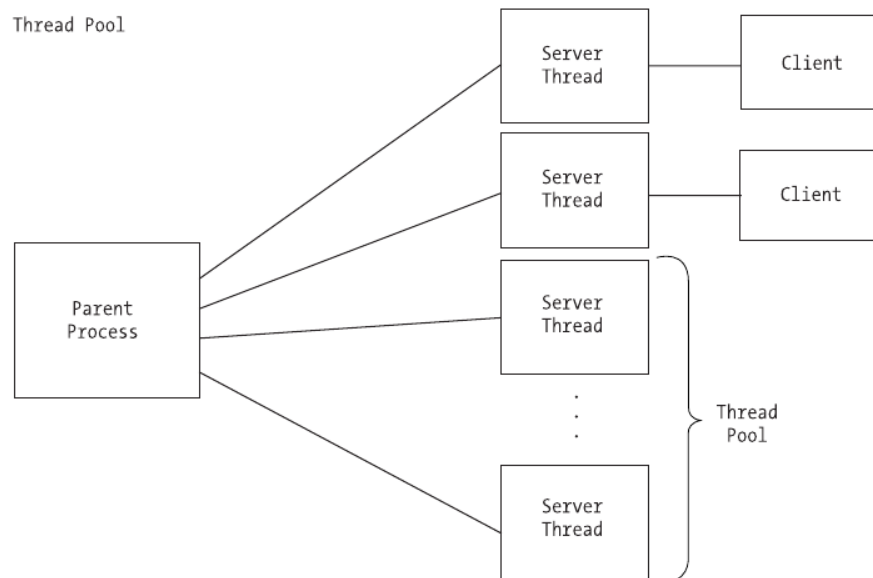
```

        /* 6. Tutup koneksi klien */
        close(accept_socket);
    }
}
wait(NULL); /* parent menunggu selesai */

/* tutup soket */
close(listen_socket);
exit(EXIT_SUCCESS);
}

```

MULTI-THREADING



Gambar 24: Prethread

- satu thread melayani satu klien
- menggunakan `pthread_create()` untuk membuat thread baru
- thread adalah *lighweight process* yang berbagi pakai memori utama dengan proses parent
- keuntungan:
 - thread menggunakan *resource* yang lebih sedikit
 - thread memiliki waktu *context-switch* yang lebih cepat
- kerugian
 - aplikasi multi-thread kurang stabil dibandingkan dengan aplikasi multi-proses
 - karena ruang memori dipakai bersama, satu thread yang crash akan mempengaruhi thread lain

- contoh implementasi: pthread (thread dibuat terlebih dahulu sebanyak jumlah tertentu)
- kompilasi dengan menambahkan opsi `-pthread` dan jalankan beberapa klien sekaligus untuk mengakses server

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <pthread.h>

#define NTHREAD 3

int i, client_id = 0;

void *accept_connection(void *arg);

int main(int argc, char *argv[])
{
    int port = atoi(argv[1]);                                /* port server */
    pthread_t thread[NTHREAD];

    /* 1. Membuat socket */
    int listen_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_socket == -1) {
        fprintf(stderr, "Error creating socket.\n");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in server_addr = {                       /* alamat server */
        .sin_family = AF_INET,
        .sin_port = htons(port),
        .sin_addr.s_addr = INADDR_ANY
    };

    /* 2. Memberikan alamat ke socket */
    if (bind(listen_socket, (struct sockaddr*) &server_addr, sizeof (struct sockaddr_in))
        != 0) {
        fprintf(stderr, "Error binding.\n");
        exit(EXIT_FAILURE);
    }

    /* 3. Listen --> socket pasif */
    if (listen(listen_socket, 5) == -1) {
        fprintf(stderr, "Error listening.\n");
        exit(EXIT_FAILURE);
    }

    printf("[SERVER] listening...\n");

```

```

    /* 4. Pre-threading */
    for (i = 0; i < NTHREAD; i++) {
        printf("[SERVER] creating thread-%d\n", i);
        pthread_create(&thread[i], NULL, accept_connection, (void *)&listen_socket);
        sleep(1);
    }
    for (i = 0; i < NTHREAD; i++) { /* parent menunggu sampai selesai */
        pthread_join(thread[i], NULL);
    }

    /* tutup soket */
    close(listen_socket);
    exit(EXIT_SUCCESS);
}

void *accept_connection(void *arg)
{
    int listen_socket = *(int *)arg;
    int tid = i;

    while (1) {
        struct sockaddr_in client_addr; /* alamat klien */
        socklen_t client_addr_size = sizeof (struct sockaddr_in);

        /* 5. Membuat soket untuk menerima koneksi dari klien */
        int accept_socket = accept(listen_socket, (struct sockaddr *)&client_addr, &client_addr_size);
        if (accept_socket == -1) {
            fprintf(stderr, "Error accepting accept_socket.\n");
            exit(EXIT_FAILURE);
        }
        int cid = client_id++;

        /* cetak alamat klien */
        printf("[SERVER] thread-%d accepting client-%d from %s:%d\n", tid, cid, inet_ntoa(client_addr.sin_addr), client_addr.sin_port);

        /* kirim pesan ke klien */
        char server_msg[] = "+OK Welcome, type your message.\n";
        write(accept_socket, server_msg, sizeof server_msg);

        /* baca pesan dari klien */
        char client_msg[100] = {0};
        read(accept_socket, client_msg, sizeof client_msg);
        printf("[CLIENT] client-%d said: %s", cid, client_msg);

        /* balas pesan ke klien */
    }
}

```



```

    char server_reply[] = "+OK Message accepted. Bye!\n";
    write(accept_socket, server_reply, sizeof server_reply);

    /* 6. Tutup koneksi klien */
    close(accept_socket);
}
}

```

HYBRID (PREFORK - PRETHREAD)

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <pthread.h>

#define NPROCESS 3
#define NTHREAD 3

int i, j, server_id, client_id = 0;

void *accept_connection(void *arg);

int main(int argc, char *argv[])
{
    int port = atoi(argv[1]);                /* port server */
    pthread_t thread[NTHREAD];

    /* 1. Membuat soket */
    int listen_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_socket == -1) {
        fprintf(stderr, "Error creating socket.\n");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in server_addr = {        /* alamat server */
        .sin_family = AF_INET,
        .sin_port = htons(port),
        .sin_addr.s_addr = INADDR_ANY
    };

    /* 2. Memberikan alamat ke soket */
    if (bind(listen_socket, (struct sockaddr*) &server_addr, sizeof (struct sockaddr_in))
        fprintf(stderr, "Error binding.\n");
}

```

```

        exit(EXIT_FAILURE);
    }

    /* 3. Listen --> soket pasif */
    if (listen(listen_socket, 5) == -1) {
        fprintf(stderr, "Error listening.\n");
        exit(EXIT_FAILURE);
    }
    printf("[SERVER] listening...\n");

    /* 4a. Preforking sebanyak jumlah proses yang diinginkan */
    for (i = 0; i < NPROCESS; i++) {
        printf("[SERVER] creating child-%d\n", i);
        if (fork() == 0) {
            /* jika proses anak, akan menjalankan proses */

            /* 4b. Prethreading, tiap thread menerima koneksi dari klien */
            for (j = 0; j < NTHREAD; j++) {
                printf("[SERVER] child-%d creating thread-%d\n", i, i, j);
                pthread_create(&thread[j], NULL, accept_connection, (void *)i);
                sleep(1);
            }
            for (j = 0; j < NTHREAD; j++) {
                pthread_join(thread[j], NULL);
            }
        }
    }
    wait(NULL);

    /* tutup soket */
    close(listen_socket);
    exit(EXIT_SUCCESS);
}

void *accept_connection(void *arg)
{
    int listen_socket = *(int *)arg;
    int tid = i;

    while (1) {
        struct sockaddr_in client_addr;
        socklen_t client_addr_size = sizeof (struct sockaddr_in);

        /* 5. Membuat soket untuk menerima koneksi dari klien */
        int accept_socket = accept(listen_socket, (struct sockaddr *) &client_addr, &client_addr_size);
        if (accept_socket == -1) {

```

```

        fprintf(stderr, "Error accepting accept_socket.\n");
        exit(EXIT_FAILURE);
    }
    int cid = client_id++;

    /* cetak alamat klien */
    printf("[SERVER] thread-%d accepting client-%d from %s:%d\n", tid, cid, inet_ntoa(
        /* kirim pesan ke klien */
        char server_msg[] = "+OK Welcome, type your message.\n";
        write(accept_socket, server_msg, sizeof server_msg);

        /* baca pesan dari klien */
        char client_msg[100] = {0};
        read(accept_socket, client_msg, sizeof client_msg);
        printf("[CLIENT] client-%d said: %s", cid, client_msg);

        /* balas pesan ke klien */
        char server_reply[] = "+OK Message accepted. Bye!\n";
        write(accept_socket, server_reply, sizeof server_reply);

        /* 6. Tutup koneksi klien */
        close(accept_socket);
    }
}

```

MULTITHREADING (PYTHON)

Thread on demand, setiap ada koneksi masuk, server akan membuat satu *thread* untuk melayaninya.

```

import socket
import sys
from thread import *

HOST = ''
PORT = 2001

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'

s.bind((HOST, PORT))
print 'Socket bind complete'

s.listen(5)
print 'Socket now listening'

```

```
def clientthread(conn):
    conn.send('+OK Welcome, type your message.\n');

    data = conn.recv(80)
    print 'Client said: ' + data

    reply = '+OK Message accepted. Bye!\n'
    conn.send(reply)

    conn.close()

while 1:
    conn, addr = s.accept()
    print 'Connected with ' + addr[0] + ':' + str(addr[1])

    start_new_thread(clientthread ,(conn,))

s.close()
```

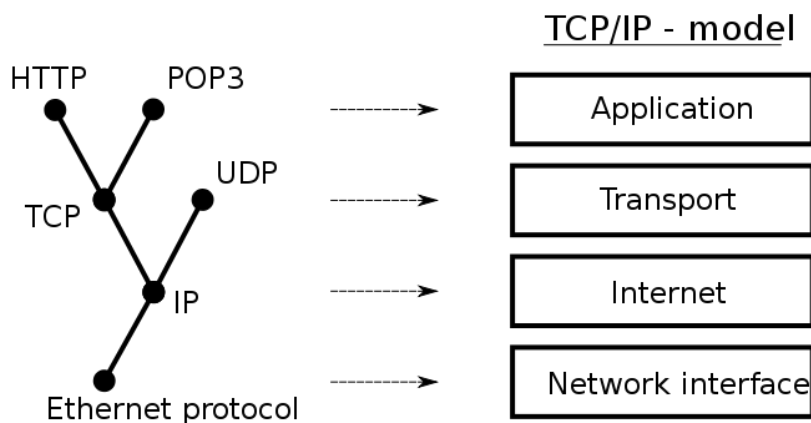
TUGAS

Sebutkan 10 aplikasi server dan cari tahu metode paralel apa yang dipakai:

- multiprocessing
- multithreading
- hibrida (multiprocessing + multithreading)

PROTOKOL LAYER APLIKASI

Protokol komunikasi adalah prosedur dan aturan standar dalam berkomunikasi. Klien yang ingin berkomunikasi dengan server harus mengikuti protokol tersebut. Misalnya klien untuk web seperti Firefox, harus menggunakan protokol HTTP untuk berkomunikasi dengan server. Namun, mekanisme protokol sangat jarang diperlihatkan pada aplikasi berbasis GUI. Untuk melihatnya, kita akan menggunakan program `netcat` dan `openssl s_client`. Umumnya protokol pada layer aplikasi ini berbasis teks, sehingga mudah dipahami.



Gambar 25: Layer jaringan TCP/IP (sumber: Wikipedia)

HTTP

Hypertext transfer protocol (HTTP) adalah dasar komunikasi pada *world wide web*. Server HTTP menggunakan *transport layer* TCP pada *port* 80. Spesifikasi HTTP versi 1.1 didefinisikan pada RFC 2616.

Jenis *request* dari klien:

- GET: mengambil data

- HEAD: mengambil *header*-nya saja
- POST: menambahkan data, misalnya *form submission*
- ...

Status respon dari server:

- 100 Continue
- 200 OK
- 206 Partial Content
- 301 Moved Permanently
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- ...

Contoh GET: halaman utama <http://ipb.ac.id>

Contoh POST: posting ke form <http://172.18.88.13/pesan.php> dengan empat variabel: **nama**, **email**, **pesan**, dan **tambah**.

Header HTTP dapat juga diamati menggunakan ‘Network Monitor’ (Ctrl+Shift+Q) pada Firefox.

FTP

File transfer protocol (FTP) adalah protokol standar untuk transfer *file* via jaringan. FTP menggunakan *transport layer* TCP. Server menerima perintah melalui *port* 21. Server mengirimkan data ke *port* 20 (mode aktif) atau *port* *ephemeral* (mode pasif). Mode pasif lebih banyak dipakai oleh klien FTP karena tidak terhalang oleh *firewall* (lihat <http://slacksite.com/other/ftp.html>). Spesifikasi FTP didefinisikan pada RFC 959.

Perintah FTP:

- USER: otentikasi nama pengguna
- PASS: otentikasi *password*
- STAT: status koneksi
- CWD: ganti direktori
- PWD: cetak nama direktori
- PASV: masuk ke mode pasif (dilakukan sebelum transfer data)
- LIST: list isi direktori
- RETR: mengunduh *file*
- STOR: mengunggah *file*
- QUIT: memutus koneksi

Contoh komunikasi dengan server FTP <ftp://ftp.debian.org>:

Setelah masuk mode PASV, buka satu klien lain ke alamat yang dikembalikan mode tersebut untuk menangkap transfer data dari server.

SMTP

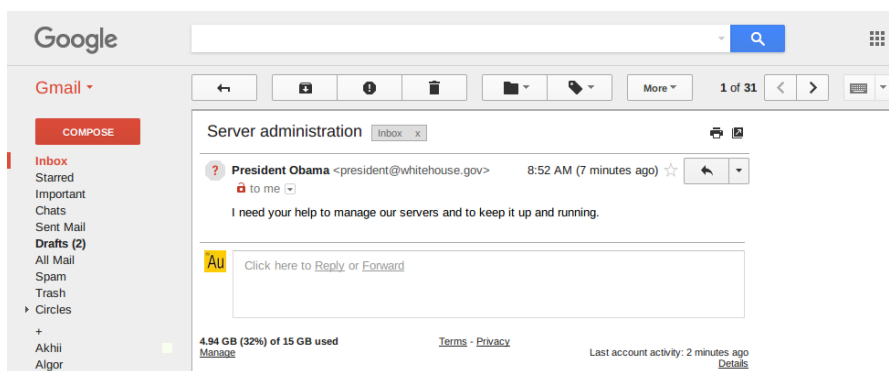
Simple mail transfer protocol (SMTP) adalah standar untuk pengiriman email melalui Internet. SMTP menggunakan *transport layer* TCP port 25, 465 (SSL), atau 587 (TLS). SSL atau TLS digunakan oleh SMTPS untuk mengenkripsi pesan. Spesifikasi SMTP didefinisikan pada RFC 5321.

Perintah SMTP:

- HELO: intro ke server
- AUTH: otentikasi
- MAIL: alamat pengirim
- RCPT: alamat penerima
- DATA: isi pesan, diakhiri dengan sebaris yang berisi satu titik
- QUIT: mengakhiri sesi

Enkode *username* dan *password* untuk otentikasi:

Contoh komunikasi dengan server SMTPS:



Gambar 26: Email telah terkirim

POP3

Post office protocol versi 3 (POP3) digunakan oleh klien untuk mengambil email dari server. POP3 menggunakan *transport layer* TCP port 110 atau 995 (POP3S). POP3S menggunakan SSL/TLS untuk mengenkripsi pesan. Spesifikasi POP3 didefinisikan pada RFC 1939.

Perintah POP3:

- USER: nama pengguna
- PASS: *password*
- STAT: status inbox
- LIST: list inbox
- RETR: membaca surat

- DELE: menghapus surat
- RSET: reset, batalkan semua modifikasi
- QUIT: mengakhiri sesi

Contoh komunikasi dengan server POP3S:

IMAP

Internet message access protocol (IMAP) digunakan oleh klien untuk mengambil email dari server. IMAP menggunakan *transport layer* TCP port 143 atau melalui SSL pada port 993 (IMAPS). Spesifikasi IMAP didefinisikan pada RFC 3501. IMAP memiliki fitur yang lebih canggih dan kompleks daripada POP3.

Perintah IMAP:

- LOGIN: nama dan *password* pengguna
- LIST: list mailbox
- SELECT: memilih mailbox
- FETCH: membaca surat
- STORE: mengubah atribut surat
- LOGOUT: mengakhiri sesi

Contoh komunikasi dengan server IMAPS:

TUGAS

Gunakan SMTP langsung untuk mengirim email dari akun email kalian masing-masing ke komdatjarkom2@gmail.com dengan isi sebagai berikut (sesuaikan dengan nama dan NIM kalian):

Subject: SMTP G6...
From: ...
To: komdatjarkom2@gmail.com

Hello, ...

.

APLIKASI JARINGAN

KONEKSI

ping

- untuk mengecek koneksi ke suatu *host*
- mengirimkan paket ICMP ECHO_REQUEST ke *host* tujuan dan menunggu balasannya
- digunakan untuk memberikan gambaran awal di mana letak masalah pada jaringan

`ping <dest>`

traceroute

- untuk menelusuri rute menuju *host* tujuan, serta waktu latensinya
- digunakan untuk mengetahui di mana letak masalah pada jaringan
- **traceroute** bekerja dengan mengatur nilai *time-to-live* (TTL) paket
 - setiap paket melewati *gateway*, TTL berkurang satu
 - jika TTL bernilai 0, paket tersebut dibuang dan *gateway* mengirimkan pesan *error* ICMP “time exceeded” ke *host* pengirim

`traceroute <dest>`

host

- untuk mendapatkan alamat IP dari nama domain yang diberikan
- memakai protokol DNS untuk menerjemahkan nama domain menjadi alamat IP

- konfigurasi server DNS terletak pada *file /etc/resolv.conf*

`host <domain>`

`host -a <domain>`

whois

- untuk melihat info registrasi pemilik suatu domain

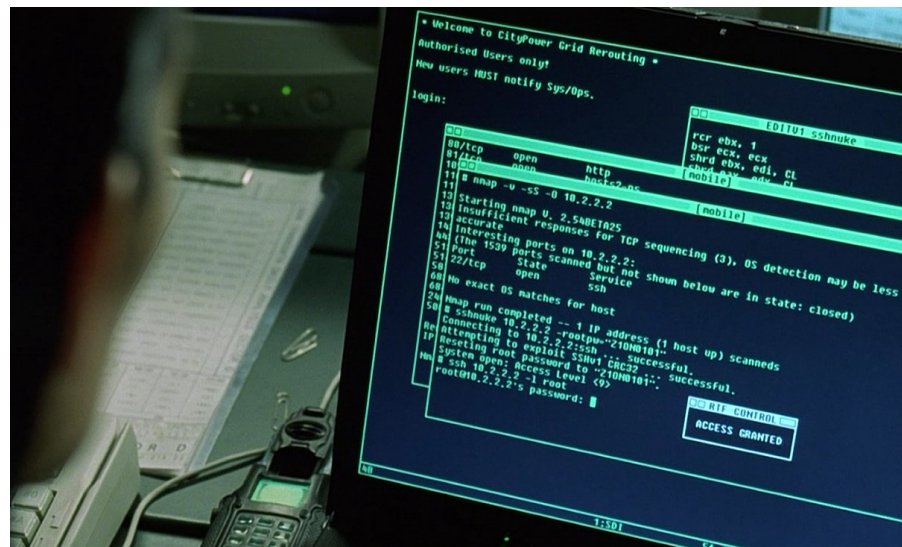
`whois <domain>`

nmap

- untuk mengetahui *port* yang terbuka pada suatu *host*
- juga informasi versi aplikasi dan sistem operasi yang digunakan

`nmap <host>`

`nmap -A <host>`



Gambar 27: nmap

Latihan:

- cari tahu alamat IP, nama admin, dan alamat admin domain `ipb.ac.id`
- cek *port* apa saja yang terbuka pada server `ipb.ac.id`
- cek jenis dan versi aplikasi server yang dipakai pada server `ipb.ac.id`

- dari data di atas, cari tahu apakah ada celah keamanan pada server tersebut

KONFIGURASI

ifconfig

- untuk mengetahui konfigurasi *interface* jaringan pada host
- satu host memiliki lebih dari satu *interface*: *loopback*, *ethernet*, *wireless*, *point-to-point*
- konfigurasi *interface* jaringan terletak pada *file /etc/network/interfaces*

ifconfig

arp

- untuk menampilkan tabel ARP
- tabel ARP berisi pasangan MAC address dan alamat IP
- MAC address dipakai untuk mengirim paket dalam satu jaringan (*layer 2: link*)

arp

netstat

- menampilkan koneksi jaringan, tabel *routing*, statistik *interface*, dan sebagainya.

menampilkan koneksi internet yang sedang aktif (kecuali server)

netstat

menampilkan koneksi internet yang sedang listening (server)

netstat -l

menampilkan statistik interface

netstat -i

menampilkan tabel routing

netstat -r

menampilkan statistik tiap protokol

netstat -s

APLIKASI JARINGAN

route

- untuk menampilkan, menambah, atau mengurangi aturan pada tabel *routing*
- penting jika sebuah *host* memiliki banyak *interface* dan *gateway* (misal: PC router)
- *flag*: U (*up*), G (*gateway*), H (*host*), D (*dynamic*), ! (*reject*)

menampilkan tabel routing

`route`

mengatur default gateway, misalnya 192.168.1.1

`route add default gw 192.168.1.1`

paket ke jaringan 192.168.3.0/24 akan di-forward ke interface 192.168.3.1

`route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.3.1`

memblok paket dari jaringan 192.168.3.0/24

`route add -net 192.168.3.0 netmask 255.255.255.0 reject`

memblok paket dari host 192.168.4.1

`route add -host 192.168.4.1 reject`

menghapus konfigurasi routing sebelumnya

`route del -host 192.168.4.1 reject`

MONITORING

tcpdump

- menampilkan semua *traffic* paket pada sebuah *interface* jaringan
- hasil keluarannya (.pcap) dapat dianalisis lebih lanjut

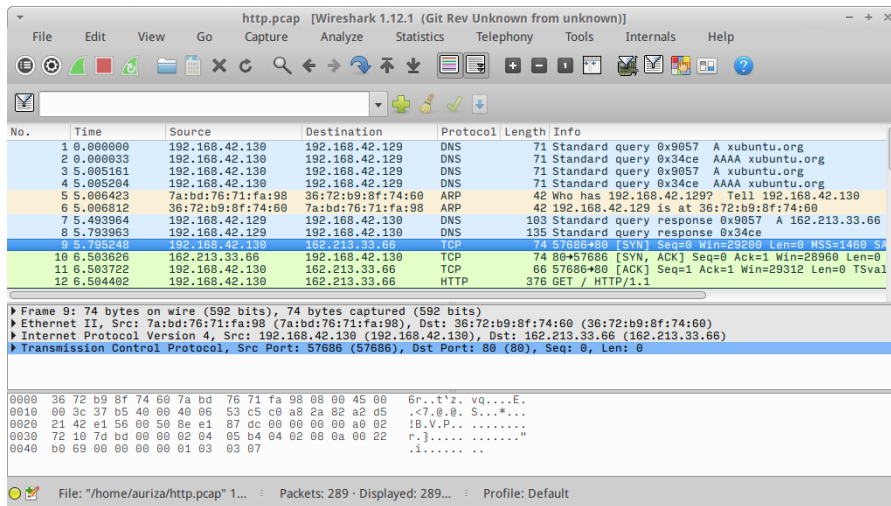
`tcpdump -i <interface>`

`tcpdump -i <interface> -w <file.pcap>`

Wireshark

- versi GUI dari `tcpdump`
- digunakan untuk analisis jaringan

Latihan:



Gambar 28: Wireshark

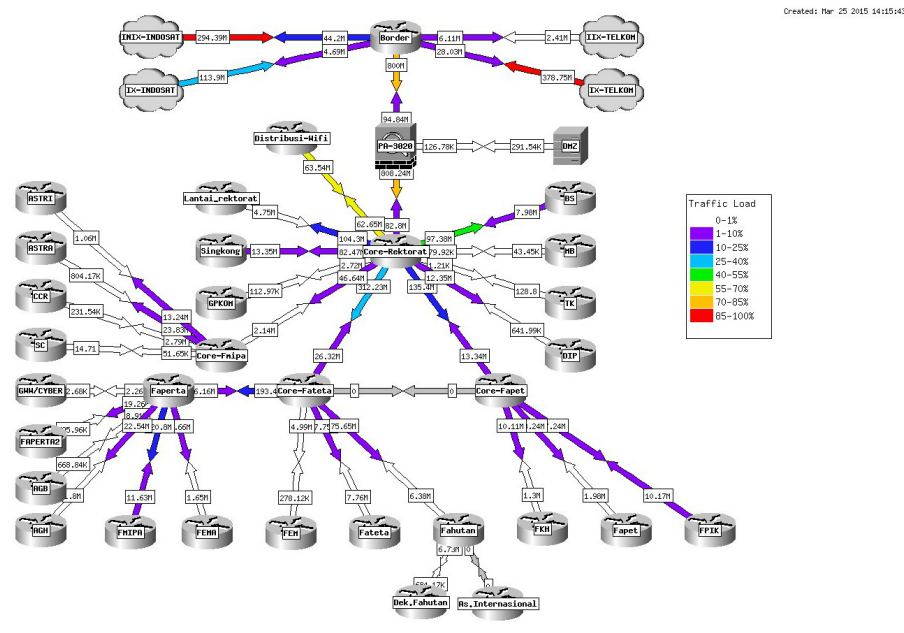
- *capture* semua paket HTTP saat membuka laman web `http://xubuntu.org`:
 - buka Wireshark dan mulai *capture* paket di *interface* Ethernet
 - buka *browser* dan akses ke laman `http://xubuntu.org`
 - tunggu sampai semua halaman termuat
 - stop *capture* paket
- filter semua paket dari/ke server web tersebut
- amati dan analisis
 - TCP *handshake*
 - HTTP *request* dan *response*
 - struktur *header* frame Ethernet, paket IP, segmen TCP, dan data HTTP
- simpan hasil *capture* dengan ekstensi `.pcap`

Web-based

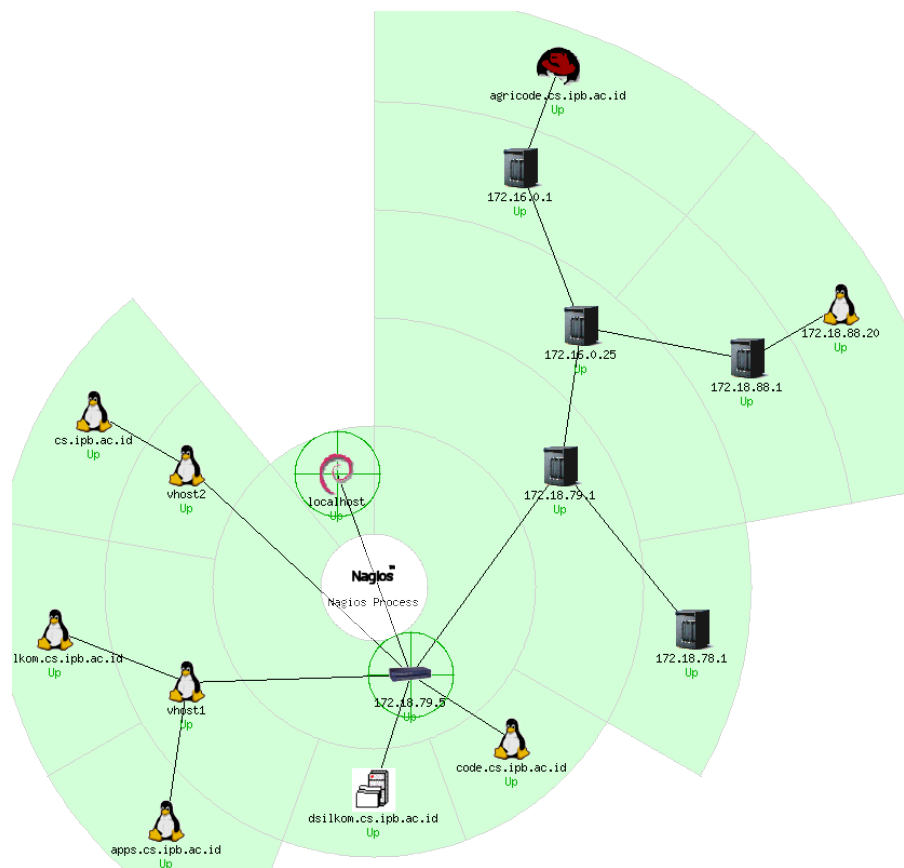
- Cacti `http://www.cacti.net/`
- MRTG `http://oss.oetiker.ch/mrtg/`
- SmokePing `http://oss.oetiker.ch/smokeping/`
- Nagios `http://www.nagios.org/`

BONUS FILM

telnet towel.blinkenlights.nl



Gambar 29: Cacti



Gambar 30: Nagios

TUGAS

Ulangi analisis paket dengan Wireshark untuk kasus aplikasi FTP!

Bagian II

SIMULASI PACKET TRACER

8

PENGENALAN PACKET TRACER

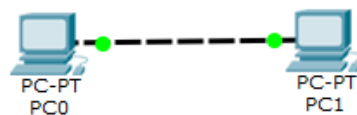
Packet Tracer adalah simulator protokol yang dikembangkan oleh Cisco. Silahkan unduh di <https://www.netacad.com/about-networking-academy/packet-tracer/>.

OPERASI DASAR

- Menambahkan *device* (PC, *switch*, *hub*, dll)
- Membuat koneksi antar *device*
 - jika *port* berwarna hijau, berarti *device* sudah terkoneksi
- Konfigurasi *device* bisa melalui *command prompt* atau GUI
- Verifikasi koneksi
 - mode *realtime*: dengan perintah `ping`
 - mode simulasi: dengan membuat *protocol data unit* (PDU) untuk mengamati jalannya paket secara visual

KONEKSI *point-to-point*

- Dua PC dengan IP statis dihubungkan dengan kabel LAN *crossover*
- Setting IP statis untuk tiap PC melalui *command prompt*, misal:
`ipconfig 192.168.0.1 255.255.255.0`
- Coba ganti dengan kabel *straight*, apa yang terjadi?
- Bagaimana kalau kita ingin menghubungkan 3 PC atau lebih?



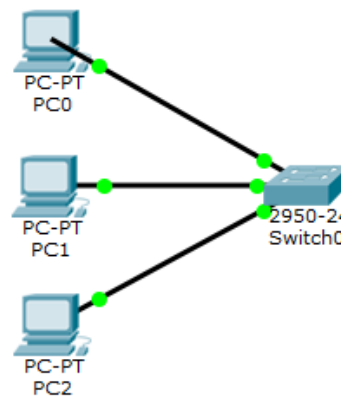
Gambar 31: *Point-to-point*

switch DAN *hub*

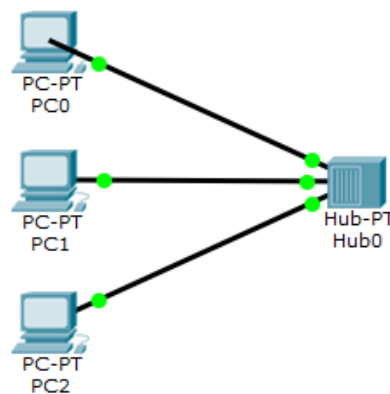
- Tiga PC dengan IP statis dihubungkan dengan *switch*
 - Cek tabel ARP pada tiap PC dan tabel MAC pada *switch* dengan tombol “Inspect”
- Kemudian coba juga dengan memakai *hub*
 - *hub* jarang dipakai karena cara kerjanya *broadcast*: membuat jaringan lebih sibuk
- Amati perbedaan cara kerja *hub* vs *switch* (pakai mode simulasi)



Gambar 32: *Switch Cisco 2960 48-port*



Gambar 33: *Switch*



Gambar 34: *Hub*

broadcast

- Coba ping *broadcast* untuk jaringan 192.168.0.0/24
 - ping 192.168.0.255
 - ping 255.255.255.255 (jika alamat jaringan tidak diketahui)
- Jalankan pada mode simulasi, amati jalannya paket ICMP

CATATAN

- Jaringan 192.168.0.0/24:
 - Alamat jaringan: 192.168.0.0
 - Alamat untuk *host*: 192.168.0.[1-254] → maksimal 254 *host* dalam jaringan ini
 - Alamat *broadcast*: 192.168.0.255
 - Prefiks jaringan: 24 → Subnet mask: 255.255.255.0
- Alamat jaringan: digunakan untuk *routing*
- Alamat *broadcast*: digunakan untuk mengetahui siapa saja *host* lain yang berada dalam satu jaringan

TUGAS

- Jenis protokol apa saja yang dipakai saat mengirim ping pertama kali?
- Jelaskan dengan singkat kegunaan protokol tersebut?

APLIKASI SERVER DAN *WIRELESS* PADA PACKET TRACER

- Hubungkan 3 PC dengan menggunakan *switch*
- alamat jaringan LAN yang akan dipakai: 192.168.0.0/24

DHCP

- DHCP digunakan untuk memberikan konfigurasi alamat IP secara dinamis kepada klien
- Tambahkan satu server, hubungkan ke *switch*
 - set alamat IP server statis: 192.168.0.2/24
 - aktifkan servis DHCP: **Services** > DHCP
 - * *range* alamat IP yang akan dialokasikan secara dinamis: 192.168.0.[101-250]/24
 - * *gateway* adalah alamat *router* yang akan digunakan untuk ke luar jaringan
 - * klik **Save**, lalu aktifkan servis DHCP
 - Default gateway: 192.168.0.1
 - DNS server : 192.168.0.2
 - Start IP addres: 192.168.0.101
 - Subnet mask : 255.255.255.0
 - Max num of user: 150
 - set konfigurasi IP semua PC menjadi dinamis: **Desktop** > **IP Configuration** > DHCP
 - pastikan PC telah mendapatkan alamat IP dari server DHCP
 - cek konektivitas dengan ping *broadcast*

multiple switch

- Tambahkan satu *switch* baru dan beberapa PC
 - contoh kasus: kita ingin menambahkan PC baru ke dalam jaringan, tetapi port pada *switch* pertama sudah terpakai

semua, maka perlu *switch* tambahan untuk memperluas jaringan LAN.

- hubungkan *switch* baru ke *switch* pertama dengan kabel *crossover*
- pastikan PC yang terhubung pada *switch* baru sudah mendapat alamat IP dari server DHCP
- cek konektivitas dengan ping *broadcast*, amati juga simulasi berjalannya paket DHCP, ARP, dan ICMP (gunakan filter paket)
- **penting:** jangan memasang *switch* membentuk *cycle*, karena akan membuat jaringan *looping*

wireless ap

- Tambahkan satu *wireless* AP dan beberapa *laptop* atau *smart-phone*
 - contoh kasus: kita ingin perangkat *mobile* juga dapat terhubung ke jaringan
 - set SSID, *channel*, dan *security* pada AP
 - matikan *laptop*, ganti *network interface* ethernet menjadi *wireless* (PT-LAPTOP-NM-1W), hidupkan *laptop* kembali
 - set koneksi wifi ke AP jaringan LAN pada *laptop*

SERVIS LAINNYA

- Cobakan servis HTTP pada server
 - modifikasi isi halaman web pada server
 - akses alamat IP server dari *browser* salah satu PC di jaringan
- Cobakan servis DNS pada server
 - DNS: menerjemahkan nama domain menjadi alamat IP
 - berikan nama domain untuk server ini, misal: **komdat.id**
 - akses alamat domain di atas dari *browser* salah satu PC di jaringan

TUGAS

Lengkapi tabel berikut ini! Kumpulkan pada selembar kertas!

Atribut	HTTP	DHCP	DNS
Kepanjangan	Hypertext Transfer Protocol
Standar	RFC 2616

Atribut	HTTP	DHCP	DNS
Layer	Aplikasi
Transport	TCP
Port	80
Fungsi	komunikasi data pada WWW
Jenis request	GET, POST, HEAD, PUT,
Aplikasi server	Apache, Nginx, IIS
Aplikasi klien	Firefox, Chrome, Opera

ROUTER JARINGAN LOKAL

Router: bekerja hingga *layer 3* (*network*), memiliki lebih dari satu alamat IP, dan bertugas mengarahkan paket ke jaringan yang lebih dekat ke tujuan.

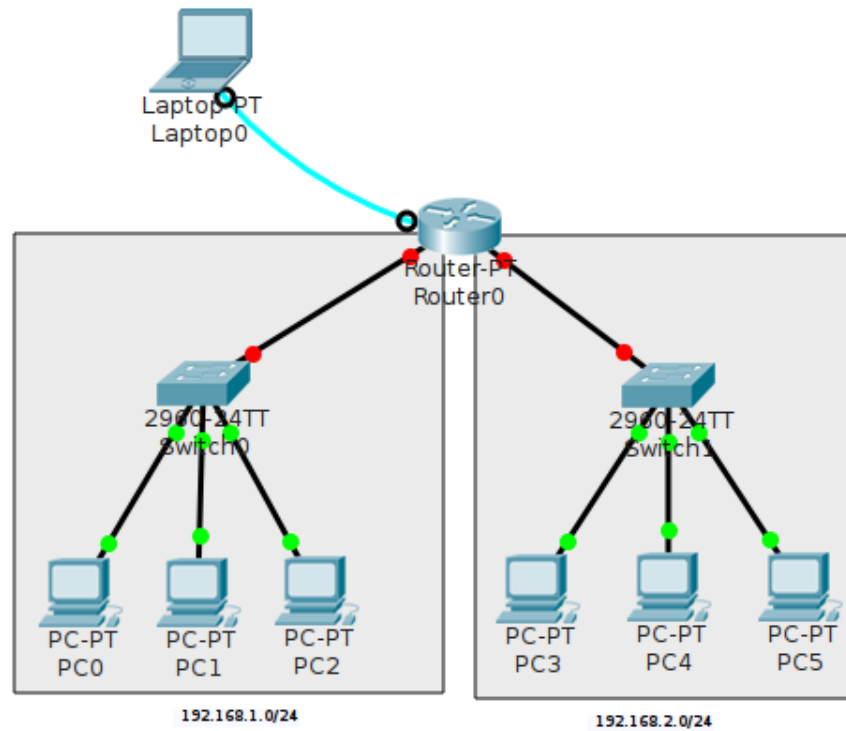


Gambar 35: Contoh *router*: Cisco 2801

KONFIGURASI *router* UNTUK MENGHUBUNGKAN DUA JARINGAN LOKAL

- Diberikan dua jaringan: 192.168.1.0/24 dan 192.168.2.0/24
 - untuk menghubungkan jaringan yang berbeda, dibutuhkan *router*
 - siapkan beberapa PC dan *switch* untuk dua jaringan lokal tersebut
- Tambahkan satu *router* untuk menghubungkan kedua jaringan tersebut
- Siapkan satu laptop untuk mengkonfigurasi *router*, hubungkan dengan kabel *console*
 - buka Terminal pada laptop untuk menampilkan CLI *router*
- Set alamat IP *router* dengan mengikuti perintah berikut
 - set *hostname* dan *password router*
 - set alamat IP *router* dan mengaktifkan *interface*-nya
 - biasanya *router* diberikan nomor *host* paling awal (.1)

```
enable
configure terminal
hostname R0
```



Gambar 36: Router LAN

```
enable secret *****
```

```
interface FastEthernet 0/0
  ip address 192.168.1.1 255.255.255.0
  no shutdown
  exit
```

```
interface FastEthernet 1/0
  ip address 192.168.2.1 255.255.255.0
  no shutdown
  exit
```

```
exit
show running-config
disable
```

- Setelah itu, atur layanan DHCP pada *router* dengan membuat *pool* untuk tiap jaringan

```
enable
configure terminal
```

```
ip dhcp pool NET1
  network 192.168.1.0 255.255.255.0
  default-router 192.168.1.1
```

```

    exit
ip dhcp excluded-address 192.168.1.1 192.168.1.100

ip dhcp pool NET2
    network 192.168.2.0 255.255.255.0
    default-router 192.168.2.1
    exit
ip dhcp excluded-address 192.168.2.1 192.168.2.100

    exit
disable

```

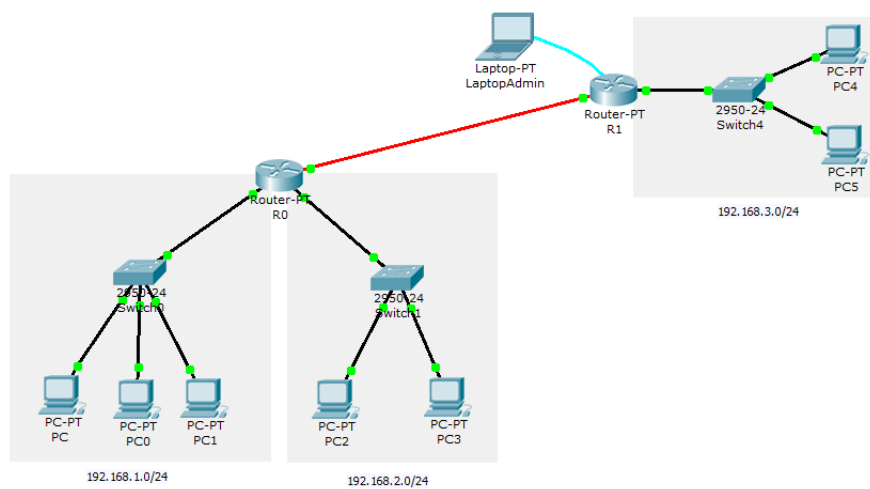
- Atur konfigurasi IP semua PC dengan DHCP
- Cek koneksi tiap PC antara dua jaringan
- Untuk mengecek daftar klien DHCP, gunakan perintah `show ip dhcp binding`
- **Penting:** simpan ke *file* `.pkt` untuk bahan praktikum pekan depan

TUGAS

Setting *router* untuk menghubungkan tiga jaringan lokal yang berbeda, yaitu jaringan untuk STAFF, STUDENT, dan NCC. Berikan alamat IP privat dengan *subnet* masing-masing 172.18.15.0/24, 172.18.16.0/24, dan 172.18.12.0/24.

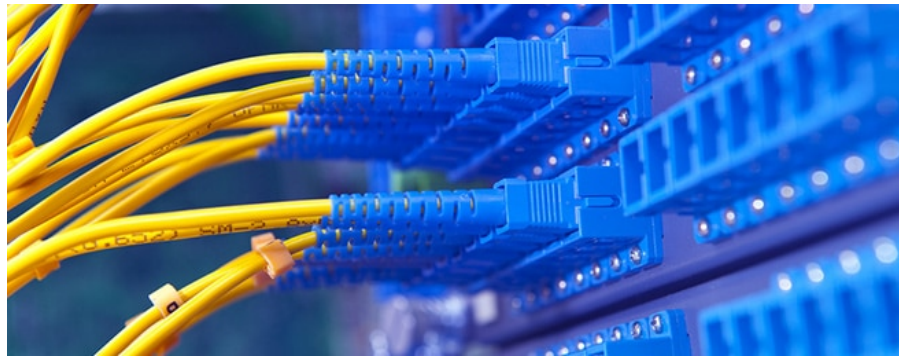
ROUTING STATIS

MENGHUBUNGKAN JARINGAN YANG LOKASINYA BERJAUHAN



Gambar 37: *Router* untuk menghubungkan jaringan dengan lokasi yang berjauhan

- Lanjutkan dari praktikum sebelumnya, tambahkan jaringan baru 192.168.3.0/24
 - jaringan baru ini jaraknya 5 km dari jaringan yang sudah ada
 - perlu memakai kabel *fiber optic* (FO)
- Tambahkan satu *router* baru R1
 - hubungkan *router* lama R0 dengan *router* R1 ini memakai kabel FO
 - hubungkan router R1 dengan jaringan baru tersebut
- Hubungkan antara *router* R0 dengan R1
 - antara *router* R0 dan R1 adalah jaringan baru, misalnya 192.168.0.0/24
 - konfigurasi alamat IP *interface* di *router* lama R0



Gambar 38: Kabel *fiber optic single-mode*

```
enable
configure terminal
  interface FastEthernet4/0
    ip address 192.168.0.1 255.255.255.0
    no shutdown
  exit
exit
disable
```

– konfigurasi alamat IP *interface* di *router* baru R1

```
enable
configure terminal
  hostname R1
  enable secret *****

  interface FastEthernet4/0
    ip address 192.168.0.2 255.255.255.0
    no shutdown
  exit

exit
disable
```

- Hubungkan *router* R1 dengan jaringan 192.168.3.0/24 dan set *pool* DHCP untuk jaringan tersebut

```
enable
configure terminal

  interface FastEthernet0/0
    ip address 192.168.3.1 255.255.255.0
    no shutdown
  exit

ip dhcp pool NET3
```



```

network 192.168.3.0 255.255.255.0
default-router 192.168.3.1
exit
ip dhcp excluded-address 192.168.3.1 192.168.3.100

```

```

exit
disable

```

- Konfigurasi *routing* statik di R0

- rute ke jaringan 192.168.3.0/24: *forward* ke 192.168.0.2 (R1)

```

enable
configure terminal

```

```

ip route 192.168.3.0 255.255.255.0 192.168.0.2

```

```

exit
show ip route
disable

```

- Konfigurasi *routing* statik di R1

- rute ke jaringan 192.168.1.0/24: *forward* ke 192.168.0.1 (R0)
- rute ke jaringan 192.168.2.0/24: *forward* ke 192.168.0.1 (R0)

```

enable
configure terminal

```

```

ip route 192.168.1.0 255.255.255.0 192.168.0.1
ip route 192.168.2.0 255.255.255.0 192.168.0.1

```

```

end
show ip route
disable

```

- Set konfigurasi IP semua PC yang baru dengan DHCP
- Cek koneksi antara jaringan baru dengan jaringan lama

TUGAS

Tambahkan *router* baru R2 dengan jarak 5 km dari R0 dan R1. *Router* R2 ini menghubungkan ke dua jaringan baru, yaitu 192.168.4.0/24 dan 192.168.5.0/24.

ROUTING DINAMIS: RIPV2

routing STATIS VS DINAMIS

Dua metode dasar untuk membangun tabel *routing*: statis dan dinamis (Cisco 2014).

Routing statis:

- tabel *routing* disusun secara manual oleh administrator jaringan
- rute statis untuk tiap jaringan harus dikonfigurasi pada setiap *router*
- menyediakan kontrol penuh pada konfigurasi *routing*, namun tidak praktis untuk jaringan yang besar
- jika ada *link* yang terputus, maka harus *update* tabel *routing* secara manual

Routing dinamis:

- tabel *routing* disusun oleh protokol *routing* yang berjalan pada *router*
- *router* berbagi informasi *routing* dengan *router* lainnya secara berkala
- mampu memilih jalur yang berbeda secara dinamis jika ada *link* yang terputus
- contoh: *routing information protocol* (RIP), *open shortest path first* (OSPF), dan *border gateway protocol* (BGP).

routing information protocol (RIP)

RIP didefinisikan dalam RFC 1058 pada tahun 1988. RIP adalah protokol vektor-jarak sederhana yang menggunakan jumlah *hop* sebagai ukuran jarak. RIP didesain untuk jaringan kecil dengan jumlah *hop* maksimum 15.

Terdapat tiga versi RIP (Nemeth *et al.* 2011):

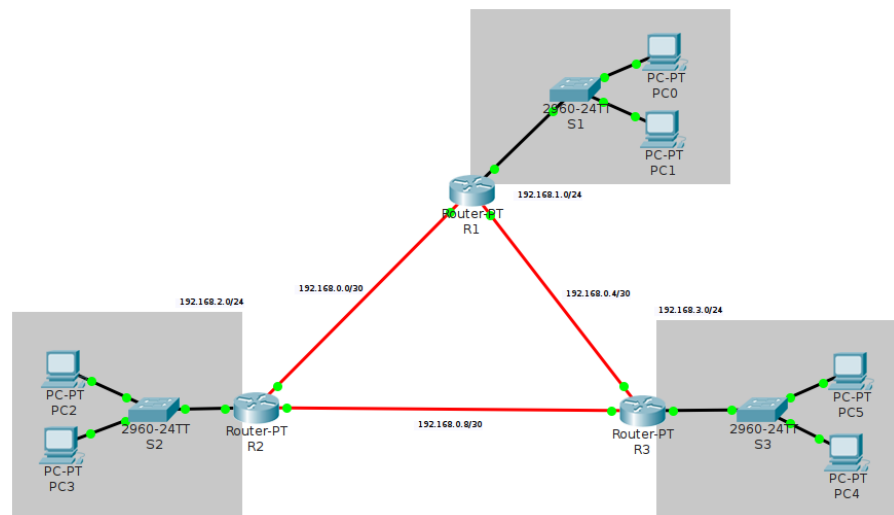
- RIPv1 hanya mendukung *classful routing*

routing DINAMIS: RIPv2

- RIPv2 menambahkan dukungan *subnet* dan *classless inter-domain routing* (CIDR)
- RIPvng adalah ekstensi dari RIPv2 untuk jaringan IPv6

Walaupun terkesan ketinggalan zaman, namun RIP masih digunakan karena sederhana, mudah dikonfigurasi, dan bekerja dengan baik pada jaringan berkompleksitas rendah.

routing DINAMIS DENGAN RIPv2



Gambar 39: *Routing* dinamis dengan RIPv2

- siapkan tiga *router*: R1, R2, dan R3, hubungkan dengan kabel fiber
- siapkan jaringan lokal untuk tiap *router*: 192.168.1.0/24, 192.168.2.0/24, dan 192.168.3.0/24

Konfigurasi router R1

- set IP *router* R1 yang terhubung ke LAN dan set servis DHCP

```
enable
configure terminal
hostname R1

interface FastEthernet 0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
```

```

ip dhcp pool NET1
  network 192.168.1.0 255.255.255.0
  default-router 192.168.1.1
  exit
ip dhcp excluded-address 192.168.1.1 192.168.1.20

```

- set IP router R1 yang terhubung dengan *router* lainnya

```

interface FastEthernet 4/0
  ip address 192.168.0.1 255.255.255.252
  no shutdown
  exit

```

```

interface FastEthernet 5/0
  ip address 192.168.0.5 255.255.255.252
  no shutdown
  exit

```

- konfigurasi RIP untuk *routing*, tambahkan **semua jaringan yang terhubung langsung** dengan *router* R1 dalam notasi *classful*

```

router rip
  version 2
  network 192.168.0.0
  network 192.168.1.0
  no auto-summary
  exit

```

- jangan kirim *update* RIP ke *interface* untuk LAN, kirimkan ke sesama *router* saja

```

router rip
  passive-interface FastEthernet 0/0
  exit

```

- lanjutkan dengan konfigurasi R2 dan R3

Konfigurasi router R2

```

enable
configure terminal
hostname R2

```

```

interface FastEthernet 0/0
  ip address 192.168.2.1 255.255.255.0
  no shutdown
  exit

```

routing DINAMIS: RIPV2

```
ip dhcp pool NET2
  network 192.168.2.0 255.255.255.0
  default-router 192.168.2.1
  exit
ip dhcp excluded-address 192.168.2.1 192.168.2.20

interface FastEthernet 5/0
  ip address 192.168.0.2 255.255.255.252
  no shutdown
  exit

interface FastEthernet 4/0
  ip address 192.168.0.9 255.255.255.252
  no shutdown
  exit

router rip
  version 2
  passive-interface FastEthernet 0/0
  network 192.168.0.0
  network 192.168.2.0
  no auto-summary
  exit

exit
disable
```

Konfigurasi router *R3*

```
enable
configure terminal
  hostname R3

  interface FastEthernet 0/0
    ip address 192.168.3.1 255.255.255.0
    no shutdown
    exit

  ip dhcp pool NET3
    network 192.168.3.0 255.255.255.0
    default-router 192.168.3.1
    exit
  ip dhcp excluded-address 192.168.3.1 192.168.3.20

  interface FastEthernet 4/0
```

```

no ip address
ip address 192.168.0.6 255.255.255.252
no shutdown
exit

interface FastEthernet 5/0
ip address 192.168.0.10 255.255.255.252
no shutdown
exit

router rip
version 2
passive-interface FastEthernet 0/0
network 192.168.0.0
network 192.168.3.0
no auto-summary
exit

exit
disable

```

Pengujian

- Cek koneksi antara ketiga jaringan tersebut (mode *realtime* dan simulasi)
- Cek isi tabel *routing* tiap *router* dengan perintah `show ip route`
- Cek detail protokol dengan perintah `show ip protocols`

TUGAS

Tambahkan satu *router* baru R4 yang tersambung ke R2, R3, dan jaringan baru NET4 192.168.4.0/24. Gunakan *routing* dinamis RIPv2 dan pastikan semua jaringan tersambung.

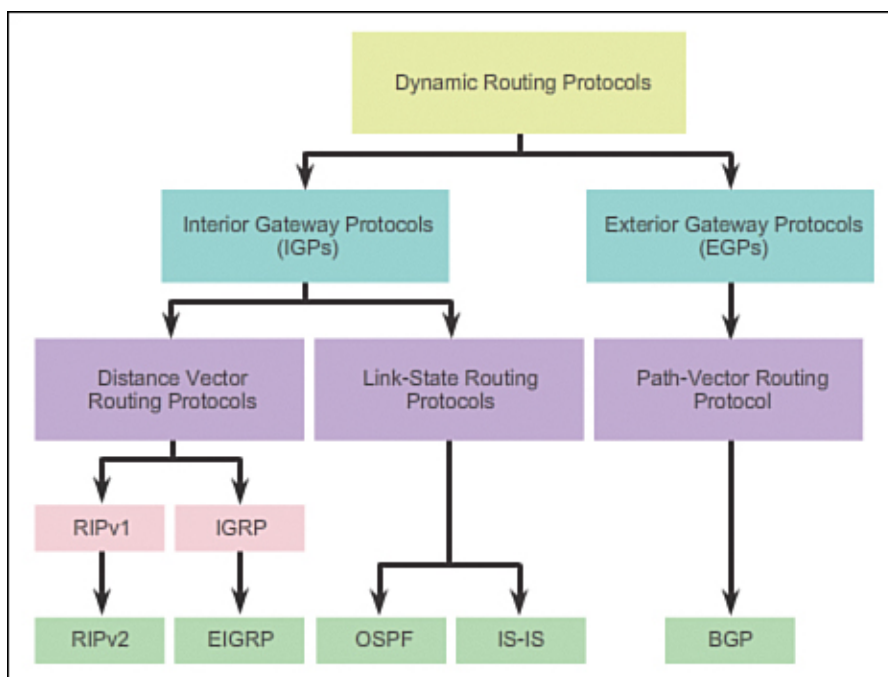
REFERENSI

Lihat dokumentasi lengkapnya di halaman berikut: *RIP and RIPv2 routing* dan *Configuring RIP*.

ROUTING DINAMIS: OSPF

Ada dua cara bagaimana algoritme *routing* dinamis bekerja, yaitu *distance-vector* (contoh: RIP, EIGRP) dan *link-state* (contoh: OSPF, IS-IS). Perbedaan antara keduanya dapat dibaca lebih lanjut pada halaman berikut:

- Types of routing protocols
- Distance vector and link state protocols



Gambar 40: Protokol *routing* dinamis (sumber: Cisco)

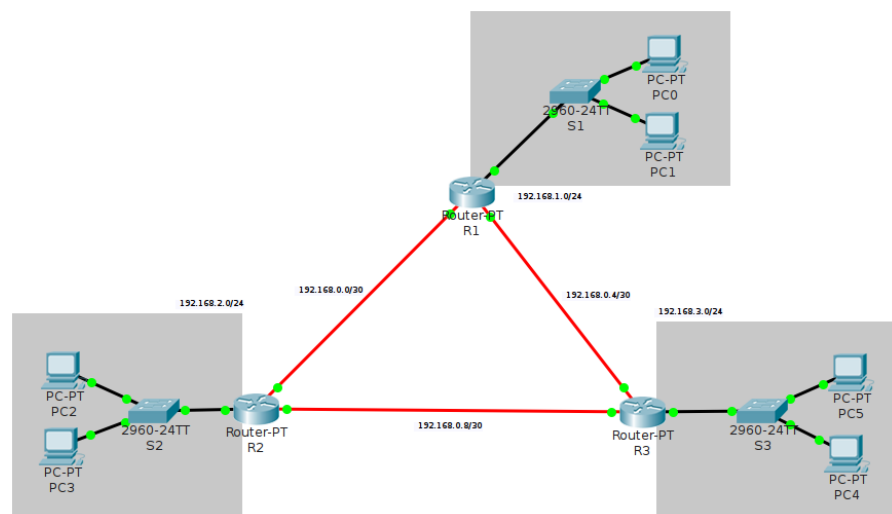
open shortest path first (OSPF)

OSPF adalah protokol berbasis *link-state* yang paling populer. “*Shortest path first*” mengacu pada nama algoritme yang dipakai dalam menghitung rute; sedangkan “*open*” menandakan bahwa protokol ini

bersifat terbuka. RFC2328 mendefinisikan protokol dasar (OSPFv2) dan RFC5340 menambahkan dukungan untuk IPv6 (OSPFv3). OSPF adalah protokol handal yang baik untuk topologi yang besar dan kompleks. Keunggulannya dibandingkan dengan RIP antara lain kemampuan mengatur beberapa jalur ke satu tujuan dan kemampuan mempartisi jaringan menjadi bagian (*area*) untuk mengurangi beban *router* dalam meng-*update* tabel *routing* (Nemeth *et al.* 2011).

routing DINAMIS DENGAN OSPF

Routing dengan OSPF dapat dibagi menjadi beberapa area. Pada contoh berikut, hanya digunakan satu area, yaitu **area 0**.



Gambar 41: Routing dinamis dengan OSPF

- siapkan tiga *router*: R1, R2, dan R3, hubungkan dengan kabel fiber
- siapkan jaringan lokal untuk tiap *router*: 192.168.1.0/24, 192.168.2.0/24, dan 192.168.3.0/24

Konfigurasi router R1

- set IP *router* R1 yang terhubung ke LAN dan set servis DHCP

```
enable
configure terminal
hostname R1

interface FastEthernet 0/0
ip address 192.168.1.1 255.255.255.0
```

```
no shutdown
exit
```

```
ip dhcp pool NET1
network 192.168.1.0 255.255.255.0
default-router 192.168.1.1
exit
ip dhcp excluded-address 192.168.1.1 192.168.1.20
```

- set IP router R1 yang terhubung dengan *router* lainnya

```
interface FastEthernet 4/0
ip address 192.168.0.1 255.255.255.252
no shutdown
exit
```

```
interface FastEthernet 5/0
ip address 192.168.0.5 255.255.255.252
no shutdown
exit
```

- konfigurasi OSPF pada tabel *routing*, tambahkan **semua jaringan dalam satu area routing** yang R1 terlibat di dalamnya, misalnya 192.168.0.0/16

```
router ospf 1
network 192.168.0.0 0.0.255.255 area 0
exit
```

- lanjutkan dengan konfigurasi R2 dan R3

Konfigurasi router R2

```
enable
configure terminal
hostname R2
```

```
interface FastEthernet 0/0
ip address 192.168.2.1 255.255.255.0
no shutdown
exit
```

```
ip dhcp pool NET2
network 192.168.2.0 255.255.255.0
default-router 192.168.2.1
exit
ip dhcp excluded-address 192.168.2.1 192.168.2.20
```

```
interface FastEthernet 5/0
  ip address 192.168.0.2 255.255.255.252
  no shutdown
  exit

interface FastEthernet 4/0
  ip address 192.168.0.9 255.255.255.252
  no shutdown
  exit

router ospf 1
  network 192.168.0.0 0.0.255.255 area 0
  exit

exit
disable
```

Konfigurasi router R3

```
enable
configure terminal
hostname R3

interface FastEthernet 0/0
  ip address 192.168.3.1 255.255.255.0
  no shutdown
  exit

ip dhcp pool NET3
  network 192.168.3.0 255.255.255.0
  default-router 192.168.3.1
  exit
ip dhcp excluded-address 192.168.3.1 192.168.3.20

interface FastEthernet 4/0
  no ip address
  ip address 192.168.0.6 255.255.255.252
  no shutdown
  exit

interface FastEthernet 5/0
  ip address 192.168.0.10 255.255.255.252
  no shutdown
  exit
```

```

router ospf 1
  network 192.168.0.0 0.0.255.255 area 0
  exit

exit
disable

```

Pengujian

- Cek koneksi antara ketiga jaringan tersebut (mode *realtime* dan simulasi)
- Cek isi tabel *routing* tiap *router* dengan perintah `show ip route`
- Cek detail protokol dengan perintah `show ip protocols`
- Cek tetangga *router* dengan perintah `show ip ospf neighbor`

TUGAS

Tambahkan satu *router* baru R4 yang tersambung ke R2, R3, dan jaringan baru NET4 192.168.4.0/24. Gunakan *routing* dinamis OSPF dan pastikan semua jaringan tersambung.

REFERENSI

Lihat dokumentasi lengkapnya di halaman berikut: Configuring OSPF.