

MODUL PRAKTIKUM

JARINGAN KOMPUTER

S1 INFORMATIKA



Published by school of computing



Lembar Pengesahan

Saya yang bertanda tangan di bawah ini:

Nama : Siti Amatullah Karimah, S.T.,M.T.
NIK : 17900086-1
Koordinator Mata Kuliah : Jaringan Komputer
Prodi : S1 Informatika
KK : Cyber Physical Systems

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2020/2021 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.



Bandung, 11 Februari 2021

Mengesahkan,

Koordinator Mata Kuliah Jaringan Komputer

Siti Amatullah Karimah, S.T.,M.T.



Mengetahui,

Kaprodi S1 Informatika

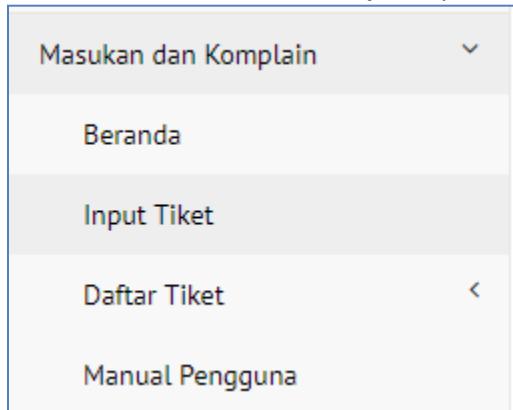
Niken Dwi Wahyu Cahyani, Ph.D.

Peraturan Praktikum Laboratorium Informatika 2020/2021

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan di Gedung Kultubai Selatan (IFLAB 1 s/d IFLAB 5) & Kultubai Utara (IFLAB 6 s/d IFLAB 7) sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa modul praktikum, kartu tanda mahasiswa, dan alat tulis.
4. Praktikan wajib melakukan tapping menggunakan kartu tanda mahasiswa diawal praktikum.
5. Praktikan wajib mengisi daftar hadir *rooster* praktikum dengan bolpoin bertinta hitam.
6. Durasi kegiatan praktikum S1 = 2 jam (100 menit).
7. Jumlah pertemuan praktikum 14 kali di lab (dimulai dari minggu pertama perkuliahan).
8. Presensi praktikum termasuk presensi matakuliah.
9. Praktikan yang datang terlambat tidak mendapat tambahan waktu.
10. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - a. Wajib menggunakan seragam sesuai aturan institusi.
 - b. Wajib mematikan atau mengondisikan semua alat komunikasi.
 - c. Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - d. Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
 - e. Dilarang membawa makanan maupun minuman di ruang praktikum.
 - f. Dilarang memberikan jawaban ke praktikan lain.
 - g. Dilarang menyebarkan soal praktikum.
 - h. Dilarang membuang sampah di ruangan praktikum.
 - i. Dilarang mencoret, mengotori, atau merusak fasilitas laboratorium.
 - j. Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
11. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum.
 - a. Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau kedukaan (menunjukkan surat keterangan dari orangtua atau wali mahasiswa.)
 - b. Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
 - c. Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Bengkel Informatika dan tidak dapat diganggu gugat.
12. Pelanggaran terhadap peraturan praktikum akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Fakultas, hingga Universitas.
13. Website IFLAB : <https://informatics.labs.telkomuniversity.ac.id/>
Asman IFLAB (Pak Aulia) : 085755219604, aulwardana@telkomuniversity.ac.id
Laboran IFLAB (Oku Dewi) : 085294057905, laboratorium.informatika@gmail.com

Tata Cara Komplain Praktikum IFLAB Melalui IGRACIAS

1. Login Igracias.
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**.



3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**.
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**.
5. Pilih Layanan: **Praktikum**.
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

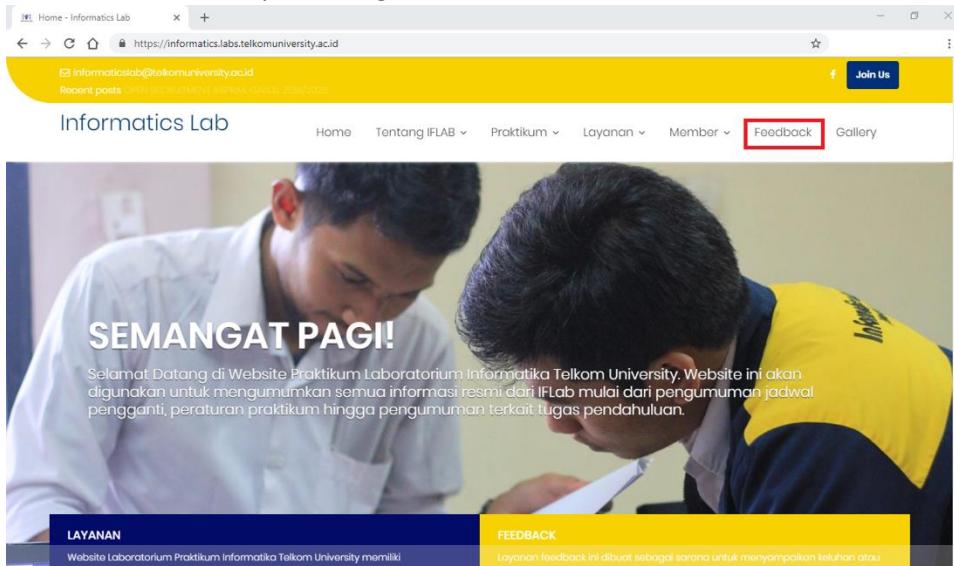
A screenshot of a web-based form titled 'Input Keluhan' (Report Complaint). The form is part of the 'Fakultas Informatika School of Computing Telkom University' website, as indicated by the background watermark. The form fields include:

- Fakultas / Bagian : BIDANG AKADEMIK (FIF)
- Program Studi / Urusan : URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)
- Pelapor : RIZQILLAH ZAHRA LESTARI
- Layanan : PRAKTIKUM
- Kategori : Pelaksanaan Praktikum
- Sub Kategori : Please Select...
- Tipe Masukan : Komplain Masukan
- Deskripsi : (A rich text editor toolbar is positioned above this field, containing various formatting icons like bold, italic, underline, etc.)

8. Lampirkan *file* jika perlu. Lalu klik Kirim.

Tata Cara Komplain Praktikum IFLAB Melalui Website

1. Buka website <https://informatics.labs.telkomuniversity.ac.id/> melalui browser.
2. Pilih menu **Feedback** pada navigation bar website.



3. Pilih tombol **Link Form Feedback**.



4. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
5. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

Daftar Isi

Lembar Pengesahan.....	i
Peraturan Praktikum Laboratorium Informatika 2020/2021	ii
Tata Cara Komplain Praktikum IFLAB Melalui IGRACIAS	iii
Tata Cara Komplain Praktikum IFLAB Melalui <i>Website</i>	iv
Daftar Isi.....	v
Daftar Gambar	viii
Modul 1 <i>Running Modul</i>	1
1.1 Pendahuluan	1
1.2 Wireshark Packet Sniffer.....	1
1.3 Cisco Packet Tracer	2
1.4 Ubuntu	2
1.4.1 Instalasi Ubuntu	3
Modul 2 Pengenalan Linux	10
2.1 Pendahuluan	10
2.2 Keunggulan Linux	10
2.3 Distribusi Linux.....	12
2.4 Filesystem Hierarchy Standard	13
2.5 Jenis File pada Linux.....	14
2.6 Perintah Dasar pada Linux	14
2.7 Cara penulisan perintah dasar	14
2.8 Menggunakan Vi <i>Text Editor</i>	15
2.8.1 Menggerakkan Kursor.....	15
2.8.2 Menghapus Teks	15
2.8.3 Pencarian String	15
2.8.4 Mengubah String.....	16
2.8.5 Count.....	16
2.8.6 Range / Jangkauan	16
2.8.7 File	16
2.8.8 Lainnya	16
2.8.9 Keluar	16
Modul 3 Network Analysis.....	17
3.1 Pendahuluan	17
3.1.1 Cara Kerja Wireshark	17
3.1.2 Pengenalan Wireshark	18
3.2 Skenario	19
3.2.1 Penggunaan Wireshark	19

3.2.2	Analisa Skenario.....	21
Modul 4	IPv4 Header	22
4.1	Struktur Header IPv4.....	22
4.2	Skenario	23
4.2.1	Inspeksi Header IPv4 Pada Wireshark.....	23
Modul 5	Static Routing	26
5.1	Pendahuluan	26
5.2	Langkah-langkah Simulasi Static Routing pada Cisco Packet Tracer.....	26
Modul 6	Dynamic Routing	36
6.1	Pendahuluan	36
6.1.1	Autonomous System (AS)	37
6.1.2	Perbedaan Intradomain Routing dan Interdomain Routing	38
6.1.3	Klasifikasi Dynamic Routing	38
6.1.4	Distance vector vs. link state routing algorithms.....	38
6.2	Langkah Simulasi Dynamic Routing pada Cisco Packet Tracer	41
Modul 7	UDP Header Analysis dan ICMP Analysis	47
7.1	Pendahuluan	47
7.2	Skenario	47
7.2.1	Topologi	47
7.2.2	Resource yang Dibutuhkan	47
7.2.3	Merekam info konfigurasi IP pada sebuah PC (Part 1)	47
7.2.4	Menggunakan Wireshark untuk mengambil DNS Queries dan Responses I (Part 2)	48
7.2.5	Menganalisa DNS yang diambil atau UDP packets (Part 3)	48
7.3	Kesimpulan.....	52
Modul 8	Three-Way Handshaking.....	53
8.1	Pendahuluan	53
8.2	Skenario	54
Modul 9	Congestion Control	56
9.1	Mininet.....	56
9.1.1	iPerf.....	57
9.2	Congestion Control Cubic, Reno dan DCTCP.....	58
9.3	Skenario Penggerjaan	58
Modul 10	Queue.....	62
10.1	Pendahuluan	62
10.2	Membuat <i>Custom Topology</i> di Mininet.....	62
10.2.1	Membuat Source Code	63
10.2.2	Menjalankan Custom Topology	64

10.3	Jenis-jenis Queue	64
10.3.1	CBQ (Class Based Queue).....	64
10.3.2	HTB (<i>Hierachial Token Bucket</i>)	65
10.4	Contoh Perintah CBQ.....	66
10.5	Contoh Skenario CBQ dan HTB pada Mininet.....	66
10.5.1	Skenario CBQ.....	66
10.5.2	Skenario HTB	69
Modul 11	Advanced Packet Capture	72
11.1	Pendahuluan	72
11.2	Instalasi TCPDump di Linux	72
11.3	<i>Capture</i> Menggunakan TCPDump.....	72
Modul 12	Spesifikasi Tugas Besar.....	74
12.1	Simulasi MPTCP Pada Mininet	74
12.2	Ketentuan Pengumpulan Tugas Besar	75
Modul 13	Kemajuan Pengerjaan Tugas Besar	76
13.1	Kegiatan Praktikum	76
Modul 14	Presentasi Tugas Besar.....	77
14.1	Kegiatan Praktikum	77
Daftar Pustaka	78



Daftar Gambar

Gambar 1. 1 GUI Wireshark	1
Gambar 1. 2 GUI Cisco Packet Tracer.	2
Gambar 1. 3 Distro Linux.....	3
Gambar 1. 4 Pilihan Bahasa.	4
Gambar 1. 5 Pilihan Instalasi.....	4
Gambar 1. 6 Tipe Instalasi.....	5
Gambar 1. 7 Pilihan Partisi.....	5
Gambar 1. 8 Pengisian Data dan Pilihan Kota.....	5
Gambar 1. 9 Proses Instalasi.	6
Gambar 1. 10 Selesai Instalasi.	6
Gambar 1. 11 Virtual Box.	7
Gambar 1. 12 Isi Nama VM.	7
Gambar 1. 13 Alokasi RAM.	7
Gambar 1. 14 Tipe Penyimpanan.....	8
Gambar 1. 15 Ukuran Penyimpanan.....	8
Gambar 1. 16 Konfigurasi Instalasi.	8
Gambar 1. 17 Mulai Instalasi.	9
Gambar 1. 18 Command Linux Umum.....	9
Gambar 2. 1 <i>Malware</i>	11
Gambar 2. 2 Distribusi Linux.	13
Gambar 2. 3 Terminal	14
Gambar 2. 3 Terminal	14
Gambar 2. 1 <i>Malware</i>	14
Gambar 3. 1 Layer-layer TCP/IP.....	17
Gambar 3. 2 Konsep Enkapsulasi.	17
Gambar 3. 3 Struktur <i>Packet Sniffer</i>	18
Gambar 3. 4 Wireshark <i>Graphical User Interface (GUI)</i>	18
Gambar 3. 5 <i>Capture Options Window</i>	19
Gambar 3. 6 <i>Capturing Process</i>	20
Gambar 3. 7 <i>Packets Summary</i>	20
Gambar 3. 8 <i>Filtering the Captured Protocol</i>	21
Gambar 4. 1 Struktur <i>Header IPv4</i>	22
Gambar 4. 2 Tampilan awal Wireshark.....	23
Gambar 4. 3 <i>interface</i> yang tersedia.....	23
Gambar 4. 4 Definisikan <i>ip address</i>	24
Gambar 4. 5 Tekan tombol <i>start</i>	24
Gambar 4. 6 Tampilan <i>command prompt</i> saat melakukan <i>ping</i>	24
Gambar 4. 7 Tampilan wireshark saat melakukan <i>capture packet</i>	25
Gambar 5. 1 Cisco <i>Packet Tracer</i>	26
Gambar 5. 2 Tampilan <i>router</i> pada Cisco <i>Packet Tracer</i>	27
Gambar 5. 3 Contoh topologi.	28
Gambar 5. 4 Tampilan <i>setting PC</i>	31
Gambar 5. 5 Tampilan <i>IP Configuration</i>	31
Gambar 5. 6 Tampilan <i>setting PC</i>	34
Gambar 5. 7 Tampilan <i>command prompt</i> PC1.	34
Gambar 5. 8 Tampilan <i>command prompt</i> PC2.	35
Gambar 5. 9 Tampilan <i>command prompt</i>	35
Gambar 6. 1 Ilustrasi <i>routing</i>	36

Gambar 6. 2 Ilustrasi <i>Autonomous System</i>	37
Gambar 6. 3 Hirarki algoritma <i>routing</i>	38
Gambar 6. 4 Ilustrasi <i>routing</i> protokol OSPF	39
Gambar 6. 5 <i>Cisco Packet Tracer</i>	41
Gambar 6. 6 Contoh topologi.	41
Gambar 6. 7 <i>Initial Configuration Dialog</i>	42
Gambar 6. 8 <i>Initial Configuration Dialog</i>	42
Gambar 6. 9 Masuk mode <i>admin</i>	42
Gambar 6. 10 Terminal konfigurasi.....	43
Gambar 6. 11 Konfigurasi <i>router</i> RIP.	43
Gambar 6. 12 Konfigurasi RIPv2.....	44
Gambar 6. 13 Pengujian konektivitas.	44
Gambar 6. 14 Menu awal.....	45
Gambar 6. 15 Menyimpan konfigurasi <i>router</i>	45
Gambar 6. 16 Konfigurasi OSPF.	46
Gambar 7. 1 Topologi yang digunakan.	47
Gambar 7. 2 Tampilan Wireshark saat mem-filter protocol DNS.	49
Gambar 7. 3 Rincian <i>packet</i> pada Wireshark.....	49
Gambar 7. 4 UDP Header.	50
Gambar 7. 5 Rincian User Datagram Protocol.	50
Gambar 7. 6 <i>Packet</i> data permintaan DNS (gambar diblok).	50
Gambar 7. 7 Rincian <i>packets</i> pada Wireshark.	51
Gambar 7. 8 Rincian <i>packets</i> DNS.....	52
Gambar 8. 1 Ilustrasi Three-Way Handshake.	53
Gambar 8. 2 Capture <i>packets</i> pada Wireshark.	54
Gambar 8. 3 Request SYN.	54
Gambar 8. 4 Balasan SYN-ACK.	55
Gambar 8. 5 Balasan ACK.....	55
Gambar 9. 1 Topologi Skenario Mininet.	59
Gambar 9. 2 Congestion Control Reno dan Vegas.	61
Gambar 10. 1 Topologi.....	62
Gambar 10. 2 CBQ Link Sharing Allocation.	64
Gambar 10. 3 CBQ Hierarchical Link-Sharing Structure.....	65
Gambar 10. 4 Struktur HTB.	66
Gambar 10. 5 Topologi Skenario CBQ.	66
Gambar 10. 6 Hasil Skenario CBQ.	69
Gambar 10. 7 Topologi Skenario HTB.	69
Gambar 10. 8 Hasil Skenario HTB.	71
Gambar 12. 1 Topologi MPTCP.	74

Modul 1 Running Modul

Tujuan Praktikum

1. Memahami instalasi tools yang digunakan.

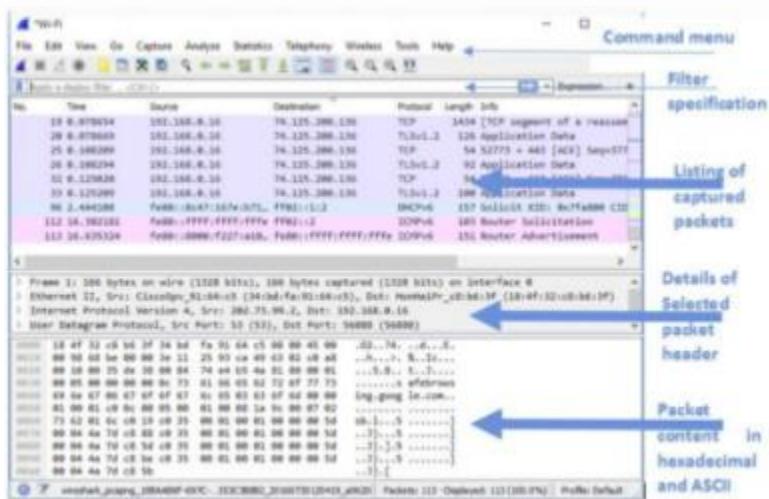
1.1 Pendahuluan

Pada praktikum kali ini praktikan akan diminta untuk mencoba dan mengeksplorasi dasar dasar jaringan komputer, seperti menganalisis paket menggunakan Wireshark, mensimulasikan jaringan menggunakan Packet Tracer dan lainnya. Dalam praktikum kali ini guna mempermudah proses dan pemahaman, praktikan akan dikenalkan pada beberapa tools / software yang perlu diinstall dan jalankan semasa praktikum. Beberapa software dan tools tersebut adalah Wireshark Packet Sniffer, Cisco Packet Tracer dan Mininet. OS yang digunakan adalah Linux distro Ubuntu. Khusus Ubuntu praktikan dipersilakan untuk menggunakan VMware atau VBox jika ingin menginstall secara virtual atau menginstall secara dual boot.

1.2 Wireshark Packet Sniffer

Wireshark adalah penganalisis protokol jaringan yang terkemuka dan banyak digunakan di dunia. Ini memungkinkan anda melihat apa yang terjadi di jaringan anda pada tingkat mikroskopis dan merupakan standar de facto (dan sering kali secara de jure) di banyak perusahaan komersial dan nirlaba, lembaga pemerintah, dan lembaga pendidikan. Pengembangan Wireshark berkembang pesat berkat kontribusi relawan dari pakar jaringan di seluruh dunia dan merupakan kelanjutan dari proyek yang dimulai oleh Gerald Combs pada tahun 1998.

Sebelum memulai kalian bisa mendownload versi terbaru dari wireshark di <https://www.wireshark.org/download.html>. Setelah selesai mendownload silakan di install secara biasa.



Gambar 1. 1 GUI Wireshark.

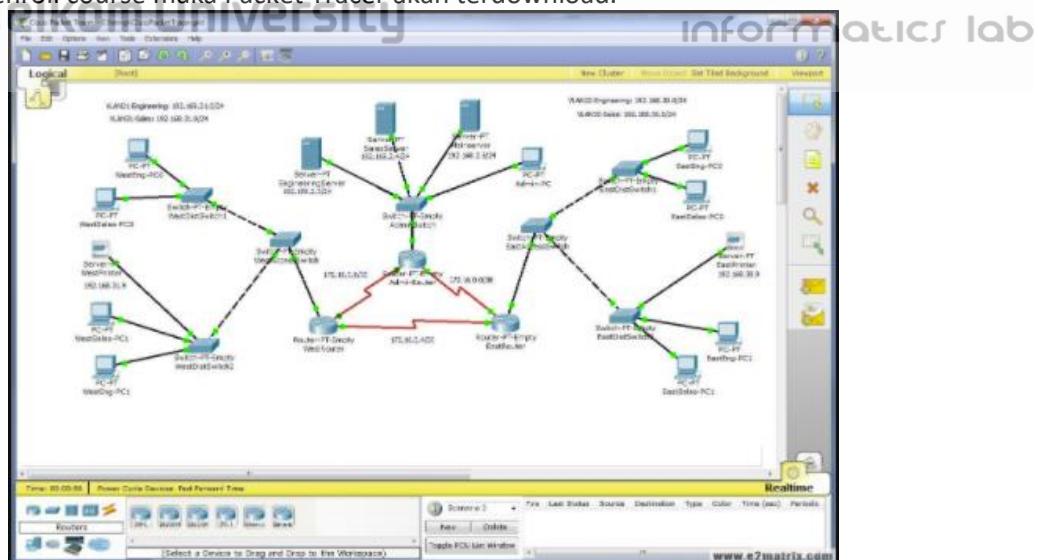
Interface Wireshark punya komponen utama, diantaranya:

- 1) **Command Menus**, merupakan menu pulldown standar yang terdapat pada bagian atas window. Yang akan menjadi perhatian utama sekarang adalah menu File dan Capture. Pada menu File terdapat berbagai menu standar seperti Save (untuk menyimpan packet data yang telah di-capture), Open (untuk membuka packet data yang pernah disimpan) dan Exit (untuk keluar dari program). Menu Capture digunakan untuk memulai meng-capture data.

- 2) **Packet Listing Window**, berisi baris-baris summary dari tiap packet yang di-capture. Termasuk di dalam summary tersebut adalah packet number (di-assign oleh Wireshark, bukan oleh protokol manapun), waktu kapan packet tersebut di-capture, alamat dari sumber dan tujuan packet itu, tipe dari protokol yang digunakan dan informasi lainnya (spesifik, tergantung dari protokolnya).
- 3) **Packet-Header Details Window**, menunjukkan detil dari packet yang dipilih pada packet listing window. Detil yang dimaksud, termasuk informasi mengenai Ethernet frame dan IP datagram yang terdapat pada packet tersebut. Jumlah dari detil informasi yang ditampilkan bisa di-expand atau di-minimize sesuai kebutuhan. Jika packet tersebut ditransportasikan melewati TCP atau UDP, detil dari TCP/UDP tersebut juga akan ditampilkan, begitu pula untuk protokol-protokol yang bekerja pada layer atasnya.
- 4) **Packet-Content Window**, menampilkan keseluruhan isi dari frame yang di-capture dalam format ASCII dan Hexadecimal.

1.3 Cisco Packet Tracer

Packet Tracer adalah alat simulasi visual lintas platform yang dirancang oleh Cisco Systems yang memungkinkan pengguna membuat topologi jaringan dan meniru jaringan komputer modern. Perangkat lunak ini memungkinkan pengguna untuk mensimulasikan konfigurasi router dan switch Cisco menggunakan antarmuka command line yang disimulasikan. **CATATAN : Sebelum bisa mendownload dan menggunakan Packet Tracer, diharuskan membuat akun Cisco terlebih dulu.** Jika tidak memiliki akun Cisco maka kita tetap bisa menggunakan Packet Tracer namun tidak bisa save project yang telah selesai dibuat. <https://www.netacad.com/courses/packet-tracer/introduction-packet-tracer> bis masuk ke link tersebut dan mendaftar course gratis disitu (sekalian membuat akun Cisco) setelah enroll course maka Packet Tracer akan terdownload.



Distro Linux seperti Ubuntu bisa didapatkan dengan berbagai cara, bisa dengan mengunduh langsung di website distro Linux tersebut, membeli melalui toko distro Linux Online, membeli di toko CD, komunitas pengguna Linux, majalah Linux seperti InfoLinux dan masih banyak cara lainnya. Beberapa distro Linux yang menyediakan link untuk mengunduh di website-nya antara lain:



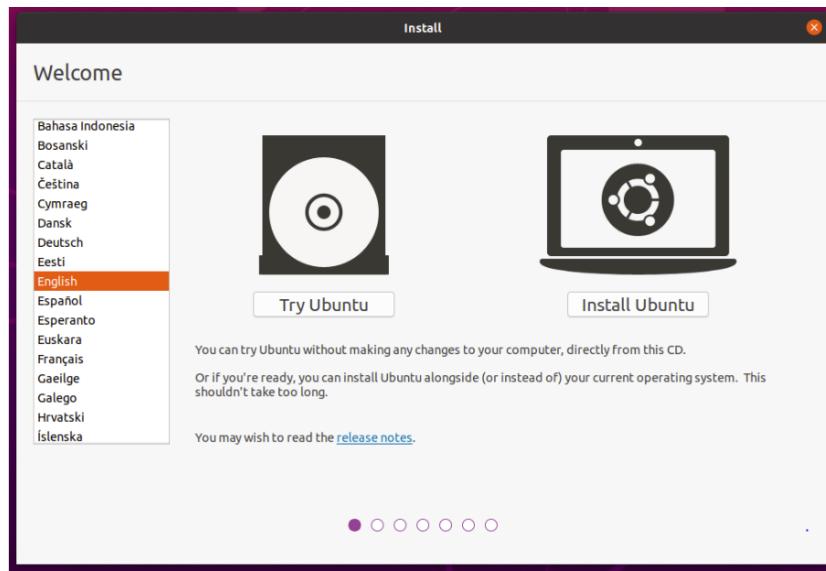
Gambar 1. 3 Distro Linux.

Pada praktikum kali ini kami merekomendasikan praktikan untuk menggunakan Ubuntu versi terbaru karena Ubuntu dinilai memiliki GUI yang paling baik dan merupakan distro Linux yang paling popular. Kalian bisa mendownload Ubuntu versi terbaru disini <https://ubuntu.com/download/desktop>.

1.4.1 Instalasi Ubuntu

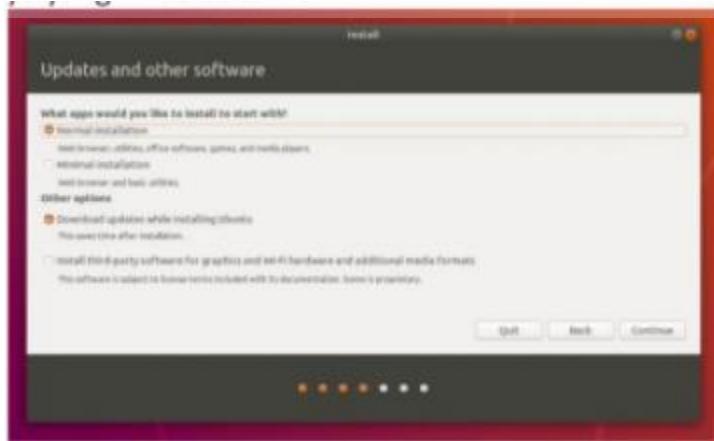
Berikut adalah langkah-langkah instalasi Ubuntu:

- 1) Masukkan DVD Ubuntu ke dalam DVD Drive Anda kemudian restart komputer Anda. Dalam kasus ini anda bisa mengganti DVD dengan Flashdisk yang dikosongkan dan hanya diisi file .iso Ubuntu, lalu ubah menjadi bootable drive dengan software Rufus.
- 2) Setelah restart, akan muncul layar seperti dibawah ini. Pada layar ini, Anda dapat memilih Bahasa yang ingin digunakan kemudian Anda dapat memilih jenis instalasi yang diperlukan. Pada tutorial ini, pilih 'Install Ubuntu'.



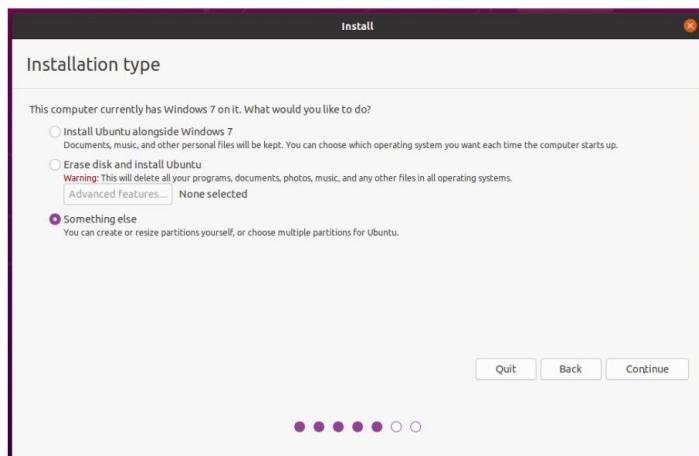
Gambar 1. 4 Pilihan Bahasa.

- 3) Jika Anda melakukan instalasi Ubuntu menggunakan USB Drive, lakukan hal yang sama seperti poin (2).
- 4) Pada layar selanjutnya, Anda akan diminta untuk memilih keyboard layout. Kemudian pilih 'Continue'.
- 5) Kemudian pada layar 'What apps would you like to install to start with', dua opsi tersebut adalah 'Instalasi normal' dan 'Instalasi minimal'. Yang pertama adalah setara dengan bundel bawaan lama dari utilitas, aplikasi, game, dan pemutar media - sebuah launchpad untuk setiap instalasi Linux. Yang kedua membutuhkan ruang penyimpanan yang jauh lebih sedikit dan memungkinkan Anda untuk menginstal hanya yang Anda butuhkan.

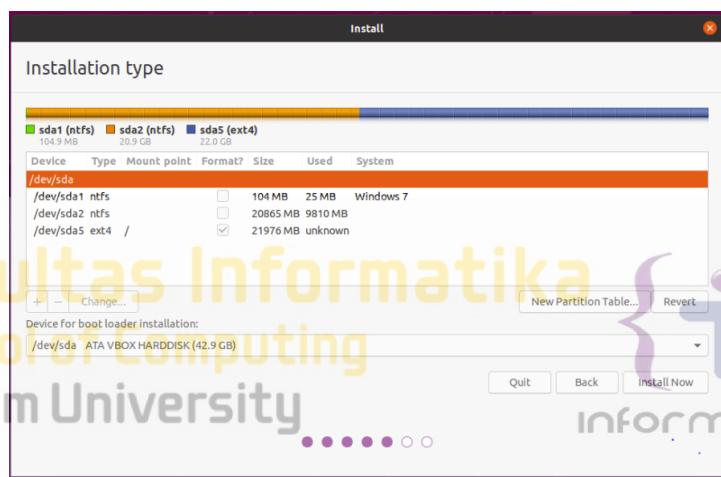


Gambar 1. 5 Pilihan Instalasi.

Di bawah pertanyaan tipe instalasi adalah dua kotak centang, satu untuk mengaktifkan pembaruan ketika menginstal dan yang lain untuk mengaktifkan perangkat lunak pihak ketiga. Alokasikan ruang drive. Gunakan kotak centang untuk memilih apakah Anda ingin menginstal Ubuntu di samping sistem operasi lain, hapus sistem operasi yang ada dan gantilah dengan Ubuntu, atau, jika anda cukup mahir, pilih opsi 'Something else'.



Gambar 1. 6 Tipe Instalasi.



Gambar 1. 7 Pilihan Partisi.

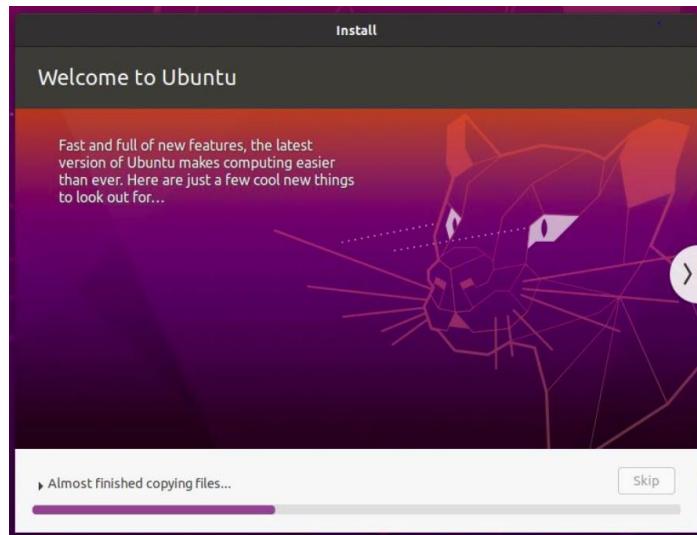
- 6) Setelah mengonfigurasi penyimpanan, klik tombol ‘Install Now’. Sebuah panel kecil akan muncul dengan gambaran tentang opsi penyimpanan yang Anda pilih, dengan kesempatan untuk Kembali jika rincianya salah. Klik ‘Continue’ untuk memperbaiki perubahan tersebut di tempat dan memulai proses instalasi.
- 7) Pada layar selanjutnya Anda akan diminta untuk memilih lokasi Anda. Jika Anda terhubung ke internet, lokasi Anda akan terdeteksi secara otomatis. Periksa lokasi Anda benar dan klik ‘Forward’ untuk melanjutkan. Jika Anda tidak yakin dengan zona waktu Anda, ketik nama kota atau kota setempat atau gunakan peta untuk memilih lokasi Anda.



Gambar 1. 8 Pengisian Data dan Pilihan Kota.

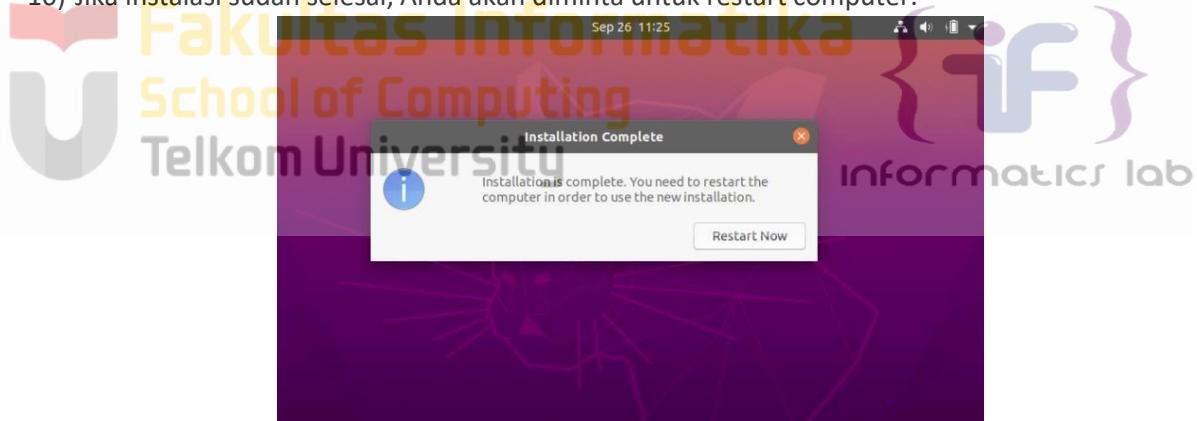
- 8) Kemudian isikan nama Anda, kata sandi, dan juga username Anda.

- 9) Penginstal akan selesai di latar belakang sementara jendela penginstalan mengajarkan Anda sedikit tentang betapa mengagumkannya Ubuntu. Tergantung pada kecepatan mesin dan koneksi jaringan Anda, penginstalan hanya perlu waktu beberapa menit.



Gambar 1. 9 Proses Instalasi.

- 10) Jika instalasi sudah selesai, Anda akan diminta untuk restart computer.

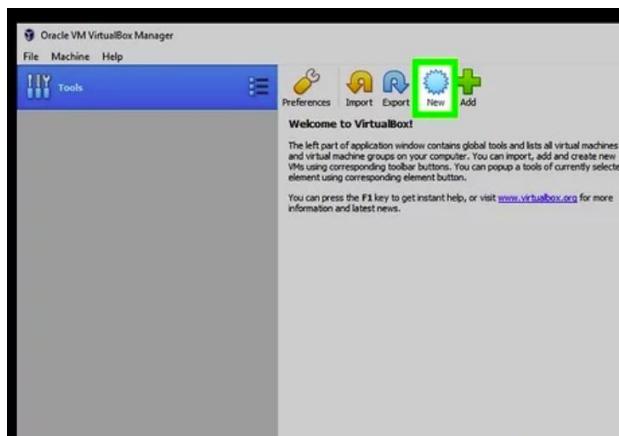


Gambar 1. 10 Selesai Instalasi.

Jika kalian menggunakan Virtual Box, maka harus mendownload dan menginstall Virtual Box terlebih dahulu

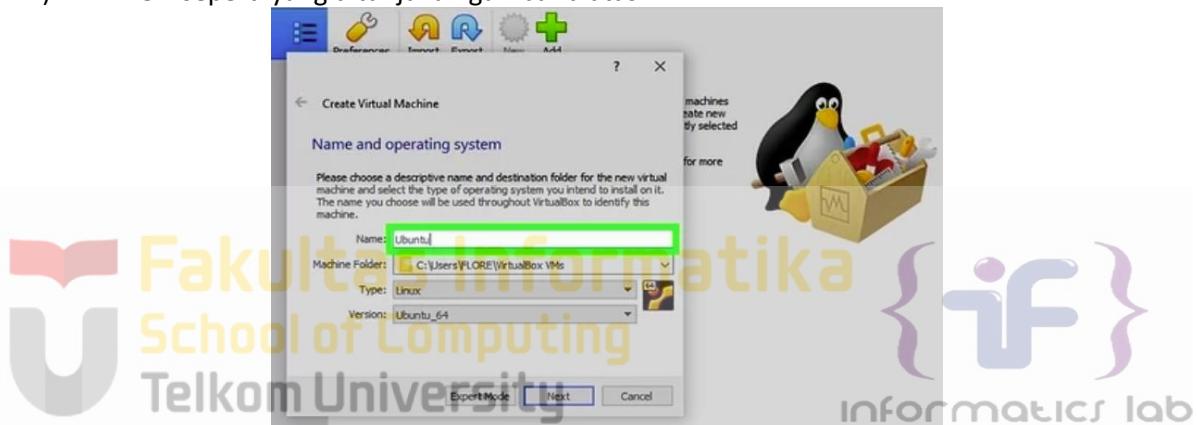
<https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html>.

Langkah-langkah instalasi menggunakan Virtual Box :



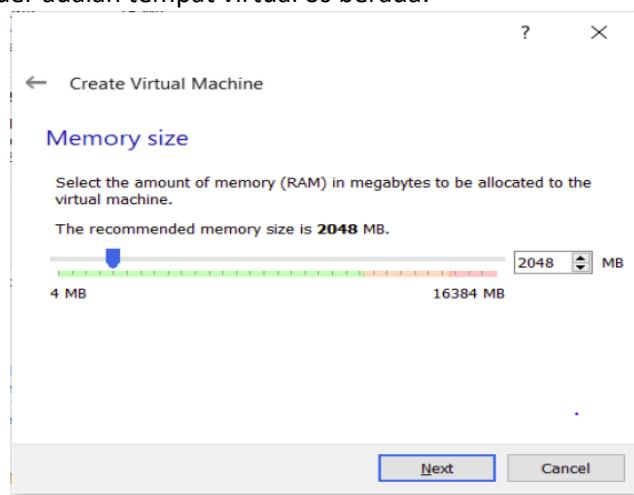
Gambar 1. 11 Virtual Box.

- 1) Klik new seperti yang ditunjukan gambar diatas.



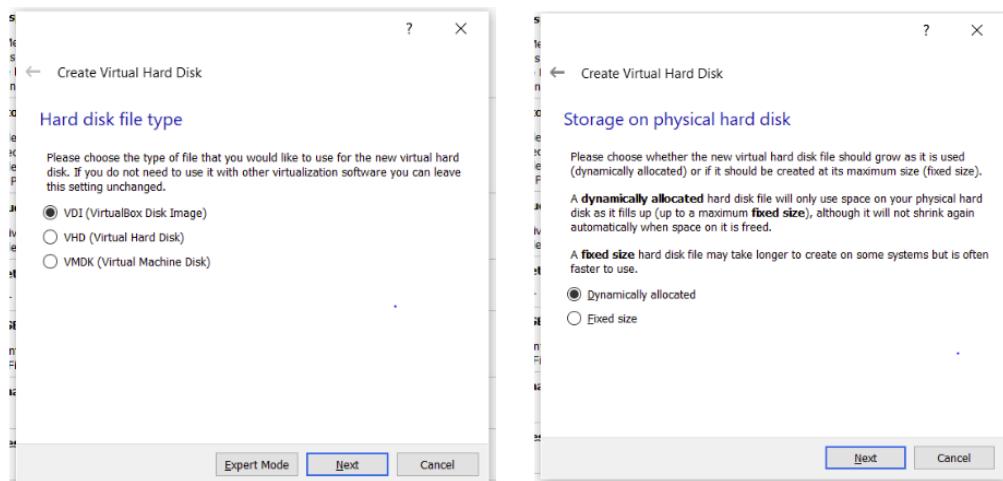
Gambar 1. 12 Isi Nama VM.

- 2) Silakan beri nama virtual machine, dan jangan lupa pilih type nya linux dengan version Ubuntu 64 bit. Machine folder adalah tempat virtual os berada.



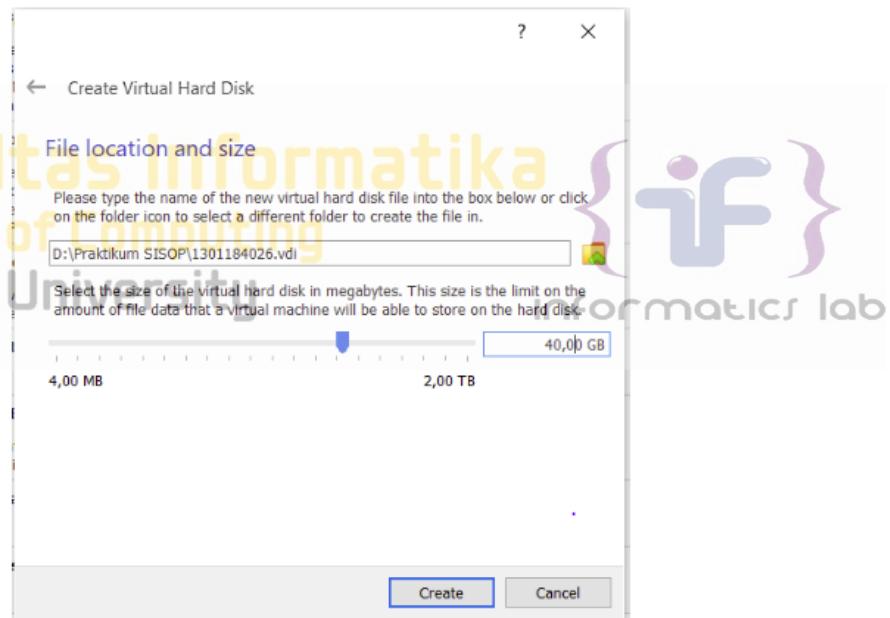
Gambar 1. 13 Alokasi RAM.

- 3) Alokasikan Sebagian RAM yang kita punya khusus untuk virtual machine kita.



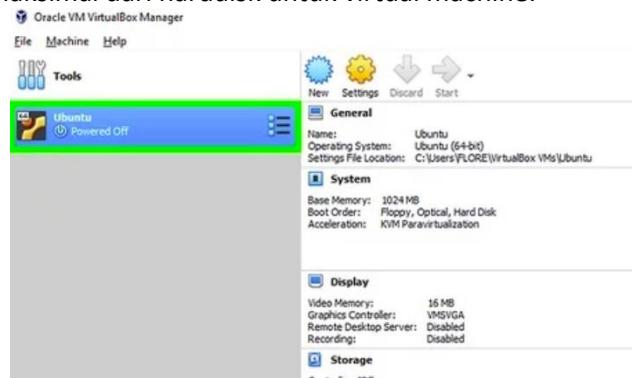
Gambar 1. 14 Tipe Penyimpanan.

- 4) Pilih tipe harddisk yang kita inginkan dan pilih apakah dynamic allocated yang artinya ukuran harddisk akan membesar sesuai kebutuhan atau memiliki maksimal ukuran.



Gambar 1. 15 Ukuran Penyimpanan.

- 5) Tetapkan ukuran maksimal dari harddisk untuk virtual machine.



Gambar 1. 16 Konfigurasi Instalasi.

- 6) Sejauh ini semua konfigurasi sudah selesai. Saatnya untuk mulai menginstall. Dengan cara masuk ke setting dan pilih pada menu DVD / Optical DVD.



Gambar 1. 17 Mulai Instalasi.

- 7) Setelah itu klik ikon folder dan isi dengan file .iso ubuntu yang sudah di download. Setelah itu tinggal klik start dan semua proses Instalasi akan sama persis dengan cara diatas sebelumnya.

Extra :

sudo	Menjalankan perintah dalam mode <i>super user/administrator</i> .
sudo su	Mengubah mode menjadi mode <i>super user/administrator</i> .
Perintah berikut adalah perintah padanan yang biasa kita gunakan di Windows. Semua perintah DOS atau <i>command</i> di Windows memiliki padanannya di Linux. Sebaliknya, banyak perintah di Linux, misalnya untuk mengkonfigurasi server, tidak dapat dilakukan dengan Windows.	
ls atau dir	Menampilkan file dan direktori.
cd	Memindahkan direktori yang sedang aktif.
cp	Men-copy file/direktori.
mv	Mengubah nama file atau memindahkan file ketempat yang baru.
mkdir	Membuat direktori baru.

Gambar 1. 18 Command Linux Umum.

Modul 2 Pengenalan Linux

Tujuan Praktikum

1. Memahami Linux.
2. Mengetahui keunggulan Linux.
3. Mengetahui distribusi Linux.
4. Mengetahui *File System Hierarchy Standard*.
5. Mengetahui perintah-perintah pada Linux.
6. Menguasai penggunaan *Vi Text Editor*.

2.1 Pendahuluan

Linux adalah sistem operasi berbasis *open source* dikembangkan dengan menggunakan model lisensi GNU GPL (*GNU General Public License*), dimana kode sumber sistem operasi ini dapat dimodifikasi, digunakan dan didistribusikan kembali secara bebas tanpa harus mengeluarkan biaya untuk pembelian lisensi. Linux di kembangkan pertama kali oleh Linus Torvalds, seorang mahasiswa Universitas Helsinki, Finlandia pada bulan April 1991, dan pertama kali dipublikasikan pada tanggal 26 Agustus 1991. Linux telah lama dikenal untuk penggunaannya di *server*, dan didukung oleh perusahaan-perusahaan komputer ternama seperti Intel, Dell, Hewlett-Packard, IBM, Novell, Oracle Corporation, Red Hat, dan Sun Microsystems. Linux digunakan sebagai sistem operasi di berbagai macam jenis perangkat keras seperti superkomputer, *server*, dan *embedded system* seperti *E-Book Reader*, konsol *game* (PlayStation 2, PlayStation 3 dan XBox), *handphone* (Android) dan *router*. Para pengamat teknologi informatika beranggapan kesuksesan Linux dikarenakan Linux tidak bergantung kepada *vendor* (*vendor independence*), biaya operasional yang rendah, dan memiliki kompatibilitas *hardware* yang tinggi, serta faktor keamanan dan kestabilannya yang tinggi dibandingkan dengan sistem operasi lainnya seperti Microsoft Windows, BeOS, Macintosh dan lainnya. Ciri-ciri ini juga menjadi bukti atas keunggulan model pengembangan perangkat lunak sumber terbuka (*opensource software*).

Karena keunggulan-keunggulan yang dimilikinya, saat ini Linux mulai di gunakan untuk penggunaan komputer *desktop*, baik untuk penggunaan pribadi maupun penggunaan perkantoran. Sistem operasi Linux sendiri terdiri dari Linux Kernel dan perangkat lunak pendukung seperti *desktop environment* (KDE, Gnome, XFCE), aplikasi perkantoran (OpenOffice, GNCash).

2.2 Keunggulan Linux

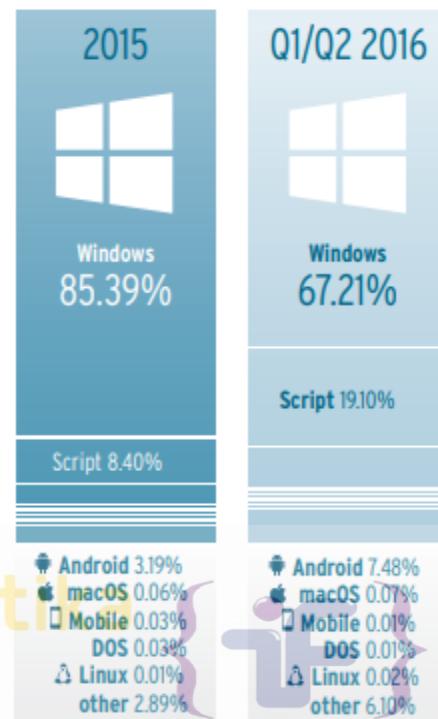
Meningkat pesatnya pengguna Linux saat ini bukanlah tanpa sebab, banyak keunggulan yang menjadi alasan kenapa Linux mulai digunakan baik secara individu maupun oleh perusahaan, berikut adalah keunggulan yang di miliki oleh sistem operasi Linux:

- 1) **Terbuka**, sistem operasi Linux di kembangkan dengan model lisensi terbuka dimana pengguna dapat melihat kode sumber, menambahkan fitur, memodifikasi sesuai kebutuhan dan mendistribusikan kembali. Selama tetap menyertakan nama pengembang awal dan mengikuti lisensi GNU GPL hal ini dapat dilakukan tanpa harus melanggar undang-undang Hak Cipta. Bisa di ilustrasikan sebagai berikut, misalnya kita membutuhkan suatu aplikasi tertentu, karena tidak memiliki sumber daya kita harus membayar perusahaan lain untuk mengerjakan aplikasi tersebut. Setelah sekian lama perusahaan tersebut bangkrut sedangkan kita menemukan ada kekurangan yang harus secepatnya di perbaiki dari aplikasi yang kita beli tersebut. Tentunya kita harus membuat ulang aplikasi tersebut oleh perusahaan lain, dan ini akan sangat memakan biaya. Hal ini tidak akan terjadi apabila kita menggunakan Sistem operasi/aplikasi yang di kembangkan dengan model lisensi terbuka sebagaimana yang di miliki Linux ini. Karena kode sumbernya terbuka, kita bisa mempekerjakan perusahaan lain untuk mengerjakan

aplikasi tersebut tanpa harus memulainya dari awal dan cukup memperbaiki kekurangannya saja.

- 2) **Murah**, keunggulan yang di miliki *open source*, selain sifatnya yang terbuka, pengguna tidak perlu membayar biaya lisensi. Bandingkan dengan sistem operasi lain seperti Microsoft Windows misalnya, kita harus mengeluarkan uang yang tidak kecil, belum lagi apabila kita membeli aplikasi tambahan seperti Microsoft Office, Photoshop, Coreldraw dan lain sebagainya. Kita hanya perlu mengeluarkan biaya *support* layaknya saat kita membeli produk *proprietary* (kode sumber tertutup). seperti membeli CD, pelatihan dan membayar teknisi.
- 3) **Bebas digunakan**, selama ini banyak *software* yang bisa kita dapatkan dengan gratis (*Freeware*), namun apabila kita teliti lebih jauh didalam perjanjian penggunaannya (*EULA*) biasanya kita akan menemukan batasan-batasan Seperti hanya boleh digunakan untuk *personal* (tidak boleh untuk usaha), hanya boleh digunakan di satu komputer saja, satu pengguna saja, tidak boleh di perbanyak, dan lain sebagainya. Tidak hanya *freeware*, bahkan perangkat Lunak *proprietary* lainnya seperti Microsoft Windows pun mempunyai banyak batasan yang harus kita ikuti agar dapat menggunakan tanpa harus melanggar hak penggunaan perangkat lunak tersebut. Linux dapat digunakan dan digandakan secara bebas tanpa batasan yang membatasinya. Kita dapat menggunakan Linux yang kita miliki untuk digunakan di satu kantor, di perbanyak, digunakan bersama-sama dan lain sebagainya.
- 4) **Stabil dan reliable**, Linux dapat beroperasi tanpa henti tanpa mengalami gangguan seperti BSOD, *Blue Screen of Dead* yang seringkali kita temui pada sistem operasi seperti Microsoft Windows. Ini menjadi alasan kenapa banyak *internet service provider* (ISP), penerbangan, bank, portal seperti Google, Facebook dan penggunaan lainnya yang membutuhkan komputer untuk bekerja maksimal menggunakan sistem operasi Linux sebagai *server*. Bayangkan misalnya pesawat yang sedang terbang tiba-tiba komputer pengendaliannya mengalami BSOD atau bayangkan pula misalnya kita sedang melakukan transfer uang di ATM tiba-tiba layar ATM tersebut berwarna biru BSOD.
- 5) **Aman**, dengan sifatnya yang terbuka, kita tidak perlu khawatir ada yang menyisipkan kode berbahaya, karena kita bisa dengan mudah melihat baris kode sumber perangkat lunak tersebut walaupun tentu saja kita membutuhkan keahlian pemrograman khusus untuk dapat membaca kode sumber aplikasi tersebut. Selain Linux memiliki resistansi yang sangat tinggi terhadap serangan *virus* dan *malware* lainnya. Hal ini bukan tanpa alasan, salah satu keunggulan struktur sistem operasi ini membuatnya mampu membuat Linux tidak mengalami kerusakan berarti apabila mengalami serangan tersebut. Dengan sistem administrasi per-layer, dimana pengguna dibatasi oleh *rule* yang dimilikinya saja, membuat pengguna (aplikasi yang digunakan pengguna) hanya dapat bekerja yang di lingkungan yang dimilikinya saja.
 - a. Apabila misalnya pengguna terserang *virus*, *virus* tersebut tidak dapat menyebar ke pengguna lain sehingga administrator dapat mengalokasi dengan cepat dan menghapus *virus* tersebut dengan mudah. Ini juga yang menjadi alasan kenapa sampai saat ini tidak banyak *virus* yang menginfeksi Linux, bandingkan dengan sistem operasi

Malware detection sorted by operating systems



Gambar 2. 1 Malware.

lain yang dalam 1 hari saja ada ribuan *virus* baru yang siap menyerang sistem operasi tersebut.

- 6) **Purnajual**, karena sifat yang dimilikinya Linux memiliki dukungan purnajual yang sangat luas, tidak terpaku satu perusahaan tertentu saja. Dukungan Linux biasanya didapatkan melalui *peer* (dalam konteks ini maksudnya kelompok pengguna linux/KPLI) – pengguna Linux lain di dalam forum internet, IRC, *newsgroup* dan *mailing list*. Tiap kota di Indonesia memiliki komunitas pengguna Linux ini, contoh untuk daerah bekasi adalah BELL (KPLI daerah Bekasi). Selain dukungan komunitas/*peer* saat ini banyak perusahaan yang juga menyediakan jasa dukungan profesional untuk membantu implementasi sistem operasi ini.
- 7) **Modular**, disisi pengembang, Linux memiliki kemudahan pengembangan karena di kembangkan dengan sistem modular. Linux menggunakan sebuah kernel monolitik, kernel Linux yang menangani kontrol proses, jaringan, periferal dan pengaksesan sistem berkas. *Device driver* telah terintegrasi ke dalam *kernel*. Fungsi-fungsi tingkat tinggi di Linux ditangani oleh proyek-proyek terpisah yang berintegrasi dengan *kernel*. Demikian juga aplikasi, aplikasi dikembangkan terpisah oleh masing-masing pengembang dan dapat digabungkan dengan mudah kedalam sistem operasi Linux sehingga ini sangat memudahkan para pengembang untuk turut serta mengembangkan sistem operasi ini.

2.3 Distribusi Linux

Distro Linux (singkatan dari distribusi Linux) adalah sebutan untuk sistem operasi komputer dan aplikasinya yang dikemas menjadi satu dengan menggunakan *kernel* Linux Kernel. Setiap individu ataupun perusahaan bebas mengembangkan suatu distribusi Linux (distro Linux) tanpa adanya monopoli. Distro Linux bisa berupa kumpulan perangkat lunak bebas seperti Debian, Zencafe, dan Ubuntu. Bisa juga kumpulan aplikasi komersial seperti Red Hat *Enterprise*, SuSE, dan lain-lain. Kadangkala distro Linux juga dikemas untuk kebutuhan khusus, seperti misalnya Zencafe yang didesain untuk penggunaan Linux pada Kios atau di Indonesia lebih dikenal sebagai warung internet atau warnet. Saat ini ada ratusan bahkan ribuan distro Linux yang dikembangkan, distrowatch.com membuat list dan memberikan rating berdasarkan pengguna terbanyak dari distro yang saat ini beredar di dunia. Walaupun berbeda namun pada dasarnya semua distro Linux tersebut menggunakan perintah dasar yang sama, perbedaan biasanya terletak pada tampilan, konfigurasi dan aplikasi-aplikasi yang ditambahkan didalamnya. Banyaknya pilihan distro Linux yang dikembangkan kadangkala membuat pengguna pemula bingung untuk memilih distro yang akan digunakan. Berikut adalah beberapa hal yang dapat menjadi pertimbangan pada saat memilih distro Linux.

- 1) Spesifikasi komputer.
- 2) Aplikasi yang dibutuhkan.
- 3) Komunitas pengguna.
- 4) *Support & Purnajual*.

Distro Linux bisa didapatkan dengan berbagai cara, bisa dengan mengunduh langsung di *website* distro Linux tersebut, membeli melalui toko distro Linux *Online*, membeli di toko CD, komunitas pengguna Linux, majalah Linux seperti InfoLinux dan masih banyak cara lainnya. Beberapa distro Linux yang menyediakan *link* untuk mengunduh di *website*-nya antara lain:



Gambar 2. 2 Distribusi Linux.

2.4 Filesystem Hierarchy Standard

Walaupun Linux dikembangkan menjadi berbagai distro Linux, namun apabila kita perhatikan struktur penempatan direktori dan file sistem memiliki karakteristik yang sama. Hal ini bisa terjadi karena adanya *Filesystem Hierarchy Standard*, sebuah standar penyusunan hirarki file sistem yang dibuat untuk mempermudahkan para pengguna Linux untuk mengoperasikan sistem operasi ini. FHS awalnya diberinama FSSTND (*Filesystem Standard*) mulai di kembangkan pada bulan Agustus 1993, karena pada saat itu pengembang mengikuti beberapa struktur *filesystem* yang sudah ada seperti struktur yang ada di BSD ataupun Unix. Variasi struktur ini tentunya menyulitkan para pengembang dan pengguna Linux. Untuk memudahkan hal ini Linux Foundation memprakarsai terbentuknya standar yang berfungsi menyeragamkan struktur *filesystem* yang ada di Linux yang kini diberi nama ***Filesystem Hierarchy Standard***.

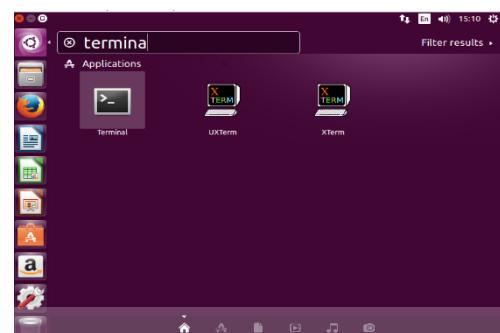
/	Akar/dasar dari hirarki file sistem Linux.
Boot	File boot loader seperti <i>kernel</i> , <i>initrd</i> , dan file lain untuk kebutuhan <i>booting</i> .
sbin	Direktori yang berisi aplikasi sistem untuk pengoperasian Linux.
bin	Direktori yang berisi aplikasi perintah dasar untuk administrasi Linux.
dev	Direktori setiap hardware di Linux memiliki file yang berkomunikasi dengan OS.
etc	<i>etcetera directory</i> , berisi file konfigurasi.
lib	Berisi kumpulan file library.
media / mnt	<i>Mounting removable device</i> seperti <i>flashdisk</i> , <i>harddisk</i> eksternal.
proc / sys	Berisi virtualisasi proses yang ada didalam Linux.
var	Berisi variabel.
opt	<i>Optional</i> .
tmp	Berisi file sementara/temporary.
usr	Sub struktur, struktur yang digunakan oleh user-user pengguna.
home	Berisi data pengguna seperti konfigurasi dan file dokumen.

2.5 Jenis File pada Linux

Pada dasarnya hampir tidak ada perbedaan jenis *file* yang digunakan oleh Linux dengan sistem operasi lain. *File-file* umum seperti direktori, *image/gambar* (gif, png), *file kompresi* (zip, tar.gz), ASCII (txt), dokumen (doc, xl, odt, odp), HTML (html), PHP dan lain sebagainya. Perbedaan jenis *file* pada Linux biasanya muncul pada jenis *file binary/file eksekusi*, dimana *file binary* pada sistem operasi lain tidak bisa digunakan pada sistem operasi Linux begitupula sebaliknya. Untuk dapat menggunakan *file binary* sistem operasi lain pada Linux biasanya dibutuhkan *Emulator* seperti Wine, CrossOver dan Cedega. Contoh *file binary* (exe, bin, com).

2.6 Perintah Dasar pada Linux

Dengan semakin majunya teknologi *desktop*, kita sudah jarang menggunakan perintah-perintah dasar ini, karena sudah mulai digantikan dengan cukup melakukan klik pada *mouse*. Pengguna Windows biasa menggunakan perintah dasar ini pada saat menggunakan *Command Prompt*. Linux memiliki dua cara untuk menggunakan perintah dasar ini, baik menggunakan *Console Mode* atau membuka aplikasi *Terminal*. *Console Mode* bisa dilakukan dengan menekan tombol Ctrl + Alt + F2, sedangkan aplikasi Terminal bisa dibuka dengan mengklik menu ***Applications -> Accessories -> Terminal***.



Gambar 2. 3 Terminal

2.7 Cara penulisan perintah dasar

Tata cara penulisan perintah di Linux adalah sebagai berikut:

perintah [option] [argument]

Untuk melihat cara penggunaan masing-masing perintah, bisa dilakukan dengan mengetik:

perintah -help

Untuk melihat cara penggunaan lebih detail bisa dilakukan dengan mengetik:

man perintah

Perintah yang sering digunakan:

sudo	Menjalankan perintah dalam mode <i>super user/administrator</i> .
sudo su	Mengubah mode menjadi mode <i>super user/administrator</i> .

Perintah berikut adalah perintah padanan yang biasa kita gunakan di *Windows*. Semua perintah DOS atau *command* di Windows memiliki padanannya di Linux. Sebaliknya, banyak perintah di Linux, misalnya untuk mengonfigurasi *server*, tidak dapat dilakukan dengan *Windows*.

ls atau dir	Menampilkan <i>file</i> dan direktori.
cd	Memindahkan direktori yang sedang aktif.
cp	Men-copy <i>file/direktori</i> .
mv	Mengubah nama <i>file</i> atau memindahkan <i>file</i> ketempat yang baru.
mkdir	Membuat direktori baru.

Perintah dasar sangat bermanfaat bagi *administrator*, termasuk apabila kita ingin mengakses komputer melalui jaringan. *Administrator server*, bisa dikatakan wajib mengenal dan memahami perintah-perintah dasar ini untuk mempermudah pengoperasian komputer *server* tersebut.

2.8 Menggunakan Vi Text Editor

vi adalah salah satu *text editor* yang biasanya tersedia di hampir semua distro Linux. Keunggulan text editor ini selain memiliki *footprint* (penggunaan *resource* seperti *memory*, *processor*, dan lain-lain) yang sangat kecil, *text editor* ini juga bisa dibilang sangat *powerful*. vi memiliki dua mode penggunaan, yaitu mode *command* dan mode *insertion*. Mode *command* adalah mode yang digunakan untuk memasukkan perintah seperti melakukan pencarian teks, menghapus sekitar baris kedepan, dan lain-lain. Mode *insertion* adalah mode yang digunakan untuk melakukan fungsi *edit* dokumen. Untuk berpindah dari mode *command* ke mode *insertion*, kita bisa menekan tombol “i” atau “**Insert**” dan untuk kembali dari mode *insertion* ke mode *command*, bisa dilakukan dengan menekan tombol “**ESC**”. Berikut adalah perintah dasar penggunaan vi.

2.8.1 Menggerakkan Kursor

h	Menggerakkan kursor ke kiri.
j	Menggerakkan kursor ke bawah.
k	Menggerakkan kursor ke atas.
l	Menggerakkan kursor ke kanan.
w	Menggerakkan kursor ke kata berikutnya.
W	Menggerakkan kursor ke pembatas kosong kata berikutnya.
b	Menggerakkan kursor ke awal kata.
B	Menggerakkan kursor ke awal pembatas kosong kata.
e	Menggerakkan kursor ke akhir kata.
E	Menggerakkan kursor ke akhir pembatas kosong kata.
(Menggerakkan kursor ke kalimat sebelumnya.
)	Menggerakkan kursor ke kalimat berikutnya.
{	Menggerakkan kursor ke paragraf sebelumnya.
}	Menggerakkan kursor ke paragraf berikutnya.
0	Menggerakkan kursor ke awal baris.
\$	Menggerakkan kursor ke akhir baris.
1G	Menggerakkan kursor ke baris pertama dari isi <i>file</i> .
G	Menggerakkan kursor ke baris terakhir dari isi <i>file</i> .
nG	Menggerakkan kursor ke baris ke-n dari isi <i>file</i> .
:n	Menggerakkan kursor ke baris ke-n dari isi <i>file</i> .
fc	Menggerakkan kursor ke huruf c terdekat selanjutnya (huruf c dapat diganti oleh huruf apa saja yang dicari).
Fc	Menggerakkan kursor ke huruf c terdekat berikutnya (huruf c dapat diganti oleh huruf apa saja yang pengin dicari).
H	Menggerakkan kursor ke atas tampilan layar.
M	Menggerakkan kursor ke tengah tampilan layar.
L	Menggerakkan kursor ke bawah tampilan layar.

2.8.2 Menghapus Teks

x	Hapus sebuah karakter pada posisi kursor.
X	Hapus sebuah karakter di sebelah kiri kursor.
D	Hapus seluruh karakter dari posisi kursor sampai akhir baris.
dd	Hapus satu baris pada kursor.
:d	Hapus satu baris pada kursor.

2.8.3 Pencarian String

/string	Melakukan pencarian <i>string</i> ke depan.
?string	Melakukan pencarian <i>string</i> mundur.
n	Menuju hasil pencarian <i>string</i> selanjutnya.
N	Menuju hasil pencarian <i>string</i> sebelumnya.

2.8.4 Mengubah String

Kita juga dapat mengubah sebuah *string/kalimat* sesuai yang kita inginkan dengan menggunakan perintah :s. Biasanya perintah ini dikombinasikan dengan perintah *range* atau perintah :g.

:s/pattern/string/flags	Lakukan pencarian terhadap pola <i>pattern</i> dan ganti dengan <i>string</i> sesuai dengan <i>flags</i> -nya.
g Flag	Ganti semua pola yang ditemukan.
c Flag	Meminta konfirmasi bila terjadi penggantian & Ulangi perintah :s terakhir.

2.8.5 Count

Perintah *count* adalah perintah untuk mengeksekusi suatu perintah dalam beberapa kali. Contoh: 10dw akan menghapus kata sebanyak 10 kata.

2.8.6 Range / Jangkauan

Untuk menentukan jarak suatu perintah (*range*) kita dapat menggunakan perintah *colon* (titik dua). Umumnya *range* dikombinasikan dengan perintah :s untuk melakukan penggantian beberapa baris. Contoh, perintah: **1,10d** akan mengeksekusi penghapusan baris dari baris 1 sampai ke baris 10.

:n,m Range	Baris n sampai m.
.. Range	Baris sekarang.
:\$ Range	Baris terakhir.
:'c Range	Penanda c.
:% Range	Semua baris dalam file.
:g/pattern/ Range	Semua baris yang sesuai dengan <i>pattern</i> .

2.8.7 File

vi dapat digunakan untuk membuka beberapa *file* sekaligus dan juga dapat menyimpan teks yang telah kita edit, bahkan kita dapat mengeksekusi perintah Linux pada saat kita masih berada didalam vi. Berikut adalah beberapa perintah tersebut.

:w file	Menulis ke <i>file</i> .
:r file	Membaca/membuka <i>file</i> .
:n	Menuju <i>file</i> berikutnya.
:p	Menuju <i>file</i> sebelumnya.
:e	Mengedit <i>file</i> .
!!program	Ganti baris dengan output dari program.

2.8.8 Lainnya

~	<i>Toggle</i> untuk huruf besar dan huruf kecil.
J	Menggabungkan baris.
.	Mengulangi perintah perubahan-teks terakhir.
u	Membatalkan perubahan terakhir (<i>Undo</i>).
U	Membatalkan semua perubahan.

2.8.9 Keluar

x	Keluar dan simpan perubahan yang telah dilakukan, ini sama dengan perintah :wq.
:q	Keluar, selama tidak melakukan perubahan terhadap <i>file</i> .
ZZ	Keluar, dan menyimpan perubahan ketika terjadi perubahan.
:q!	Keluar dengan mengabaikan semua perubahan yang telah diakukan terhadap <i>file</i> .
:sh	Keluar sementara dan menjalankan perintah linux dari dalam vi, untuk kembali kedalam vi, ketik perintah <i>exit</i> pada <i>shell</i> .

Modul 3 Network Analysis

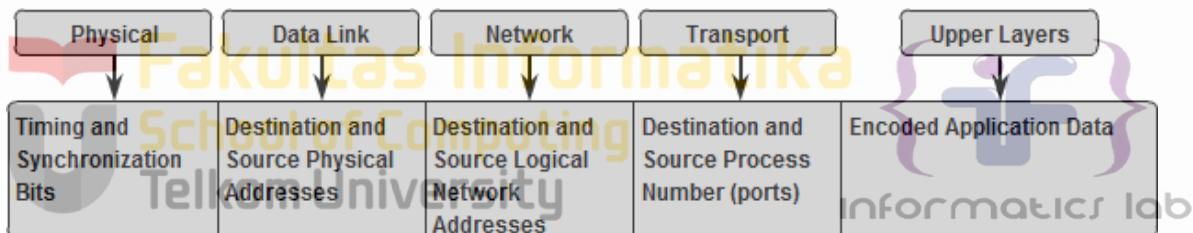
Tujuan Praktikum

1. Memahami konsep enkapsulasi.
2. Memahami cara kerja Wireshark.

3.1 Pendahuluan

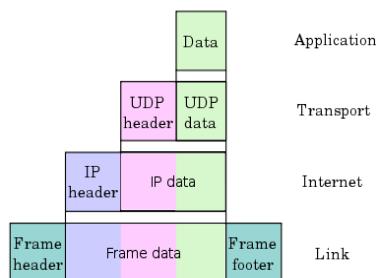
Wireshark merupakan *tool* untuk meng-*capture* dan menganalisa *packet* yang keluar ataupun masuk ke komputer tempat wireshark dijalankan. *Software* ini mengerti struktur dari berbagai macam protokol jaringan. Oleh karena itu, dia mampu menampilkan enkapsulasi dan *field-field* dari berbagai macam *packet* yang dispesifikasikan oleh berbagai macam protokol jaringan sekaligus menerjemahkan maknanya.

Apa itu enkapsulasi? Enkapsulasi adalah metode yang digunakan untuk menyembunyikan dan mengabstraksi informasi antara suatu objek dengan objek lainnya yang berbeda *level (layer)*. Dengan demikian, setiap *layer* hanya perlu menambahkan spesifikasi yang menjadi perhatian utama pada *layer* tersebut tanpa perlu mengkhawatirkan perhatian utama dari *layer* yang lain (spesifikasi yang dimaksud dapat berupa bagaimana data harus diformat, dialamatkan, dikirimkan dan kemudian dirutekan untuk sampai ke tempat tujuan yang dimaksud, untuk lebih jelasnya, lihat gambar dibawah)



Gambar 3. 1 Layer-layer TCP/IP.

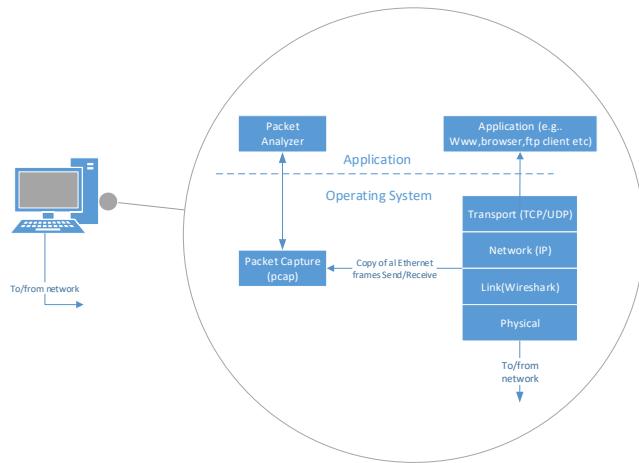
dienkapsulasi menjadi **UDP Segment** pada *layer transport*, yang kemudian dienkapsulasi menjadi **packet IP** pada *layer Internet* yang kemudian dienkapsulasi lagi menjadi **Frame** pada *layer Data Link*. Dengan memanfaatkan kondisi ini, wireshark bisa mengekstrak IP *datagram* dari *frame* yang di-*capture*, kemudian dari IP *datagram* tersebut, diekstrak TCP *segment*, lalu dari TCP *segment* diekstrak data yang dihasilkan *layer aplikasi*.



Gambar 3. 2 Konsep Enkapsulasi.

3.1.1 Cara Kerja Wireshark

Wireshark bekerja dengan memanfaatkan 2 komponen utamanya, yaitu **packet capture library** dan **packet analyzer**.



Gambar 3. 3 Struktur *Packet Sniffer*.

Packet capture library menerima *copy* dari setiap *frame* yang keluar/masuk ke dalam komputer.

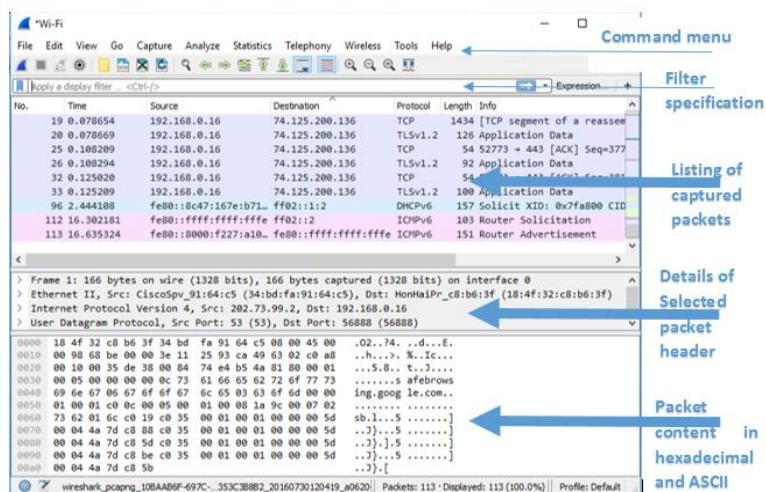
Seperti yang sudah dijelaskan tadi, semua *message* yang dipertukarkan pada *layer* aplikasi (HTTP, FTP, TCP, UDP, dll) dienkapsulasi ke dalam *frame*. Jadi, dengan mendapatkan *frame*, bisa membaca pesan yang sedang dipertukarkan antar *computer* (*message* antar aplikasi)

Packet analyzer bertugas menampilkan isi dari setiap *field* yang terdapat pada *packet* data.

Dengan memanfaatkan *database* mengenai protokol-protokol jaringan yang cukup besar (dapat mengenali lebih dari 500 protokol), wireshark dapat membaca **data/udp/packet/frame** yang dihasilkan oleh berbagai macam protokol yang bekerja pada tiap *layer* untuk kemudian ditampilkan isinya (*content*-nya).

3.1.2 Pengenalan Wireshark

Versi terbaru dari wireshark dapat diunduh di <https://www.wireshark.org/download.html>.



Gambar 3. 4 Wireshark Graphical User Interface (GUI).

Interface Wireshark punya 5 komponen utama:

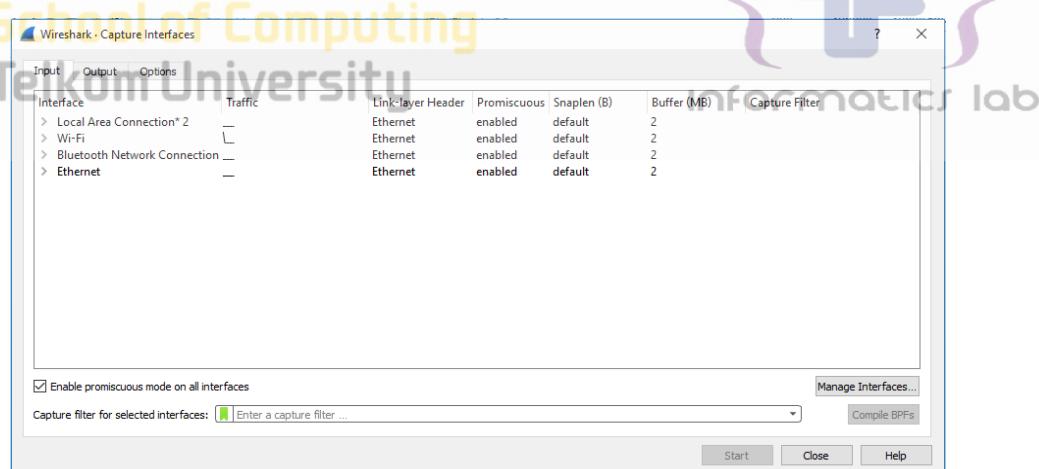
- 1) **Command Menus**, merupakan menu pulldown standar yang terdapat pada bagian atas window. Yang akan menjadi perhatian utama sekarang adalah menu *File* dan *Capture*. Pada menu *File* terdapat berbagai menu standar seperti *Save* (untuk menyimpan *packet* data yang telah di-*capture*), *Open* (untuk membuka *packet* data yang pernah disimpan) dan *Exit* (untuk keluar dari program). Menu *Capture* digunakan untuk memulai meng-*capture* data.

- 2) **Packet Listing Window**, berisi baris-baris *summary* dari tiap *packet* yang di-*capture*. Termasuk di dalam *summary* tersebut adalah *packet number* (di-*assign* oleh Wireshark, bukan oleh protokol manapun), waktu kapan *packet* tersebut di-*capture*, alamat dari sumber dan tujuan *packet* itu, tipe dari protokol yang digunakan dan informasi lainnya (spesifik, tergantung dari protokolnya).
- 3) **Packet-Header Details Window**, menunjukkan detil dari *packet* yang dipilih pada *packet listing window*. Detil yang dimaksud, termasuk informasi mengenai Ethernet *frame* dan IP *datagram* yang terdapat pada *packet* tersebut. Jumlah dari detil informasi yang ditampilkan bisa *expand* atau *minimize* sesuai kebutuhan. Jika *packet* tersebut ditransportasikan melewati TCP atau UDP, detil dari TCP/UDP tersebut juga akan ditampilkan, begitu pula untuk protokol-protokol yang bekerja pada *layer* atasnya.
- 4) **Packet-Content Window**, menampilkan keseluruhan isi dari *frame* yang di-*capture* dalam format ASCII dan Hexadecimal.

3.2 Skenario

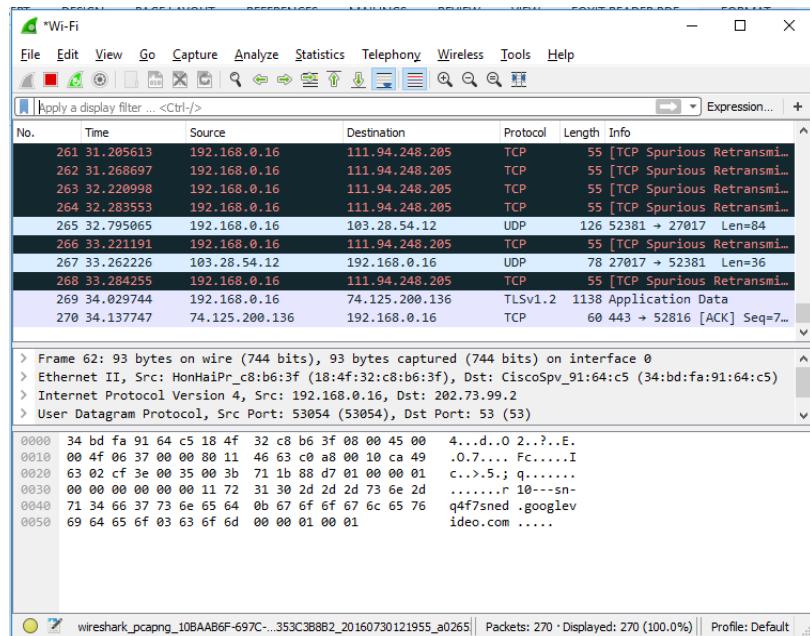
3.2.1 Penggunaan Wireshark

- 1) Buka *web browser* yang akan menampilkan *homepage* yang telah dipilih.
- 2) Buka Wireshark *Software*, lalu akan tampil *window* yang mirip dengan Gambar 3.1. Perbedaannya terletak pada tidak adanya *packet data* yang tampil pada *window packetlisting*, *packet-header*, atau *packet-contents*, karena Wireshark belum mulai meng-*capture* *packet*.
- 3) Pilih menu *Capture* lalu pilih *Option*. Lalu akan tampil *window* "Wireshark: Capture Options" seperti pada gambar 2.5.



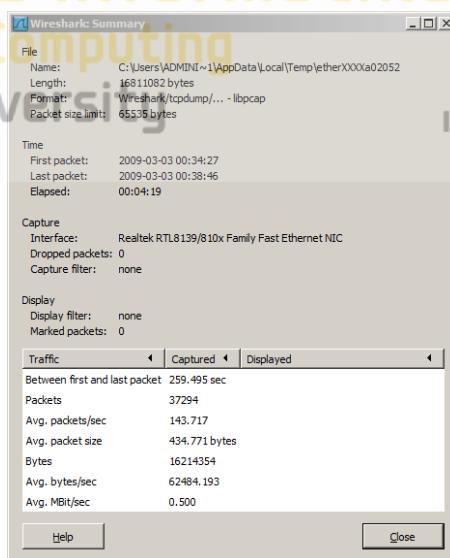
Gambar 3. 5 Capture Options Window.

Gambar 3.5 Semua *default values* pada *window* ini dapat dipakai. *Interface* jaringan (seperti koneksi fisik) yang dimiliki komputer yang dapat dipakai akan tampil pada menu *interface* di bagian atas *window Capture Options*. Jika komputer kamu memiliki lebih dari satu *interface* jaringan yang aktif (contohnya, memiliki koneksi *wireless* dan *Ethernet*), kamu harus memilih *interface* mana yang akan dipakai untuk mengirim dan menerima *packet* (biasanya *interface* *Ethernet*). Setelah memilih *interface* jaringan (atau menggunakan *interface default* dari Wireshark), pilih **OK**. *Packet capture* akan dimulai – semua *packet* yang dikirim/diterima dari/oleh komputer kamu akan di-*capture* oleh Wireshark.



Gambar 3. 6 Capturing Process.

- 4) Kamu dapat melihat *statistics* dari *packet-packet* yang ter-capture dengan memilih menu *Statistics*, kemudian *summary*. Maka akan muncul statistik mengenai *packet-packet* yang telah di-capture (hingga saat itu) seperti yang ada di bawah ini:



Gambar 3. 7 Packets Summary.

- 5) Selama Wireshark berjalan, buka *browser* dan masukkan alamat *website* yang kamu inginkan, dan tampilkan pada *browser*. Agar dapat menampilkan halaman tersebut, *browser* akan menghubungi server HTTP pada alamat web site yang tadi kamu pilih dan akan bertukar HTTP messages untuk men-download halaman index.html. Frame-frame Ethernet yang berisi HTTP messages ini akan di-capture oleh Wireshark.
 6) Setelah *browser* menampilkan halaman index html dan halaman yang dimaksud, stop peng-capture-an *packet* dengan cara menekan tombol *stop* pada Wireshark *capture window*. Lalu, Wireshark *capture window* akan hilang dan *main Wireshark window* akan menampilkan semua *packet* yang telah di-capture. *Main Wireshark window* akan terlihat seperti pada Gambar 3.1. Sekarang, kamu telah memiliki *packet data* yang berisi semua *protocol messages* yang

dipertukarkan antara komputer kamu dengan entitas jaringan yang lainnya. HTTP *messages* yang dipertukarkan dengan *web server* tujuan akan muncul pada *listing of packet* yang telah di-*capture*. Namun, akan banyak tipe *packet* yang tampil pada *listing of packet* yang telah di-*capture* (lihat, banyak tipe protokol yang tampil pada kolom Protokol pada Gambar 2). Walaupun yang kamu lakukan hanyalah men-*download* halaman *web*, tapi sebenarnya banyak protokol lain yang berjalan pada komputer kamu yang tidak terlihat oleh *user*.

- 7) Ketik “http” (tanpa tanda kutip dan dengan huruf kecil) pada tampilan spesifikasi *filter* di atas *main window* Wireshark. Lalu, pilih tombol *Apply* (yang berada di sebelah kanan *field filter*). Dengan begitu, hanya HTTP *message* yang akan tampil pada *packet-listing window*.

No.	Time	Source	Destination	Protocol	Length	Info
31	1.202233	192.168.0.16	103.233.100.119	HTTP	410	GET / HTTP/1.1
33	1.219659	103.233.100.119	192.168.0.16	HTTP	402	HTTP/1.1 200 OK (text/html)
49	1.884492	192.168.0.16	203.190.242.69	HTTP	351	GET / HTTP/1.1
51	1.896353	203.190.242.69	192.168.0.16	HTTP	535	HTTP/1.1 301 Moved Permanently (text/html)
68	1.971338	192.168.0.16	203.190.242.218	HTTP	873	GET / HTTP/1.1
102	2.089067	203.190.242.218	192.168.0.16	HTTP	483	HTTP/1.1 200 OK (text/html)
107	2.209766	192.168.0.16	203.190.242.218	HTTP	863	GET /js/lodash.min.js HTTP/1.1
134	2.232126	192.168.0.16	203.190.242.218	HTTP	872	GET /js/jquery.autocomplete.js HTTP/1.1
136	2.232394	192.168.0.16	203.190.242.218	HTTP	870	GET /js/handlebars-v3.0.1.js HTTP/1.1
144	2.236412	192.168.0.16	203.190.242.218	HTTP	892	GET /cssjs/detikframe.style.css?123 HTTP/1.1
145	2.236696	192.168.0.16	203.190.242.218	HTTP	885	GET /css/detik.style.css?123 HTTP/1.1
146	2.236894	192.168.0.16	203.190.242.218	HTTP	891	GET /css/detiksport.style.css?1234 HTTP/1.1
150	2.244868	203.190.242.218	192.168.0.16	HTTP	688	HTTP/1.1 200 OK (application/x-javascript)
151	2.245180	192.168.0.16	203.190.242.218	HTTP	899	GET /css/detiksport.detail.style.css?v=101 HTTP/1.1
157	2.248124	203.190.242.218	192.168.0.16	HTTP	723	HTTP/1.1 200 OK (application/x-javascript)

Gambar 3. 8 Filtering the Captured Protocol.

- 8) Pilih http *message* yang pertama tampil pada *packet-listing window*, yaitu HTTP GET *message* yang dikirim dari komputer kamu ke *server* alamat website (yang telah dipilih pada langkah ke-6). Saat kamu memilih HTTP GET *message*, informasi *frame* Ethernet, IP *datagram*, segmen TCP, dan HTTP *message header* akan ditampilkan pada *packet-header window*.
- 9) Selanjutnya, kamu bisa mencoba fasilitas-fasilitas lain yang disediakan oleh wireshark untuk menganalisa *packet-packet* data yang telah di-*capture* tersebut. Contohnya dengan memilih menu *Statistics*, kemudian *Protocol Hierarchy* untuk menampilkan statistik dari protokol-protokol yang bekerja sesuai *hierarki*-nya pada OSI *layer*.
- 10) Selamat bereksperimen!

3.2.2 Analisa Skenario

- 1) Dengan memanfaatkan fasilitas *Analyze* carilah nilai dari parameter *delay*, *packet loss* dan *throughput* dari ketika anda membuka salah satu *video* di *Youtube*.
- 2) Meneruskan soal nomor 1; Berdasarkan analisa *packet-packet* yang di-*capture* oleh wireshark, tuliskan MAC *address* dan IP *address* dari komputermu dan *web server* yang dituju!

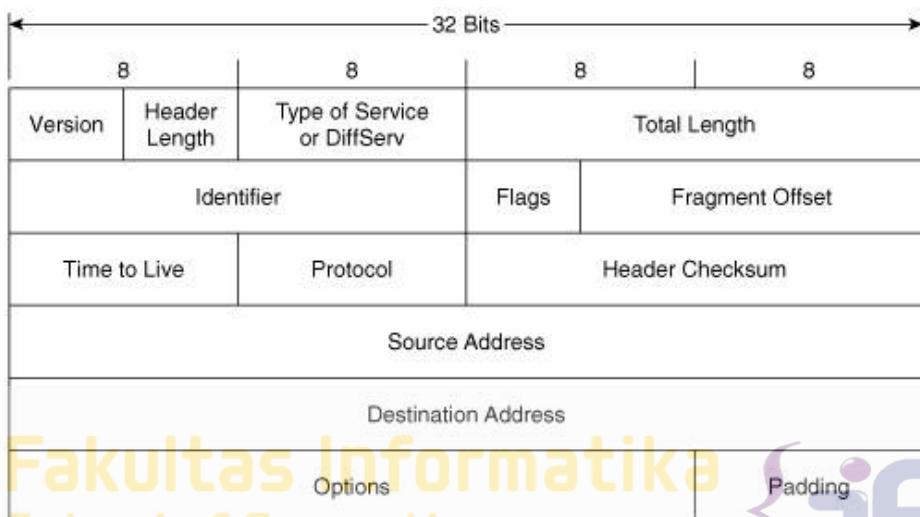
Dengan memanfaatkan menu *Statistics*, terlihat TCP *packet* mengambil porsi lebih banyak pada saat proses *capturing*. Jelaskan mengapa hal ini bisa terjadi? (petunjuk: *packet-packet* yang ter-*capture* sebagian besar dari protokol HTTP).

Modul 4 IPv4 Header

Tujuan Praktikum

1. Memahami manfaat penggunaan IPv4.
2. Mampu melakukan inspeksi pada *header* IPv4.

4.1 Struktur Header IPv4



Gambar 4. 1 Struktur Header IPv4.

Berikut merupakan struktur *header* IPv4 dengan beberapa *field* antara lain:

- 1) **Version:** menunjukkan versi IP pada paket. *Field* dapat berisi bit **0100** yang merupakan versi 4 (IPv4) atau **0110** yang berarti merupakan versi 6 (IPv6).
- 2) **Header Length:** *field* yang menunjukkan panjang *header* suatu paket IP dalam bentuk 32-bit *words*.
- 3) **Type of Service (TOS):** *field* 32-bit yang menerangkan tipe servis dari sebuah paket. Terdapat dua sub-field yaitu *Predence* yang menetapkan prioritas paket dan TOS yang memilih servis pengiriman dalam hal *throughput* dan *reliability*, namun *field* ini juga jarang digunakan dalam penerapannya.
- 4) **Total Length:** *field* 16-bit yang menerangkan panjang total dari sebuah paket.
- 5) **Identifier:** mengidentifikasi paket yang akan difragmentasikan sehingga memiliki identitas yang jelas dan unik.
- 6) **Flags:** menerangkan apakah paket yang akan dikirim tersebut dapat difragmentasikan atau tidak.
- 7) **Fragment Offset:** merupakan bit awal dari paket yang dikirim aslinya yang ditetapkan oleh *router* yang melakukan fragmentasi digunakan untuk mengidentifikasi di mana posisi *fragment* tersebut di paket aslinya.
- 8) **Time to Live:** setiap paket dikirimkan akan diberikan nilai TTL tertentu yang menerangkan berapa banyak *router* yang dapat dilalui oleh paket ini. Setiap melewati satu *router* maka nilainya akan dikurangi satu, jika mencapai nol maka paket akan diabaikan. Ini berguna untuk menghindari pengulangan yang terus menerus dalam pengiriman paket.
- 9) **Protocol:** menjelaskan pada *layer Network* pada *host tujuan* bahwa paket ini termasuk pada protokol mana.
- 10) **Header Checksum:** *field* ini digunakan untuk melakukan pengecekan *error* pada IP *Header*. Pada setiap *router* yang dilewati akan dilakukan pengecekan ulang pada *header checksum*,

jika terjadi perbedaan saat perbandingan maka paket tidak dapat diterima dan dianggap terjadi kerusakan saat pengiriman.

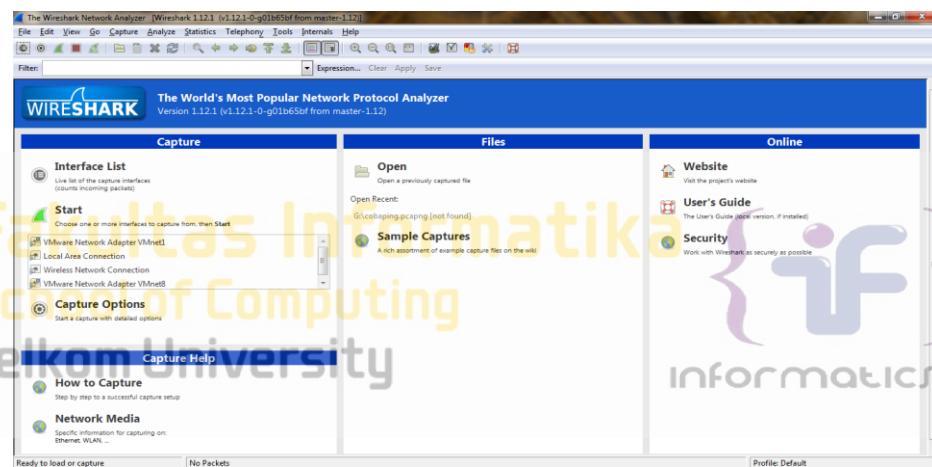
- 11) **Source Address:** merupakan *field* yang menerangkan alamat IP pengirim paket tersebut.
- 12) **Destination Address:** merupakan *field* yang menerangkan alamat IP tujuan untuk paket yang dikirimkan.
- 13) **Option Field:** merupakan *field* tambahan yang biasanya digunakan oleh *Type of Service* yang memiliki nilai lebih dari 32-bit.

4.2 Skenario

4.2.1 Inspeksi Header IPv4 Pada Wireshark

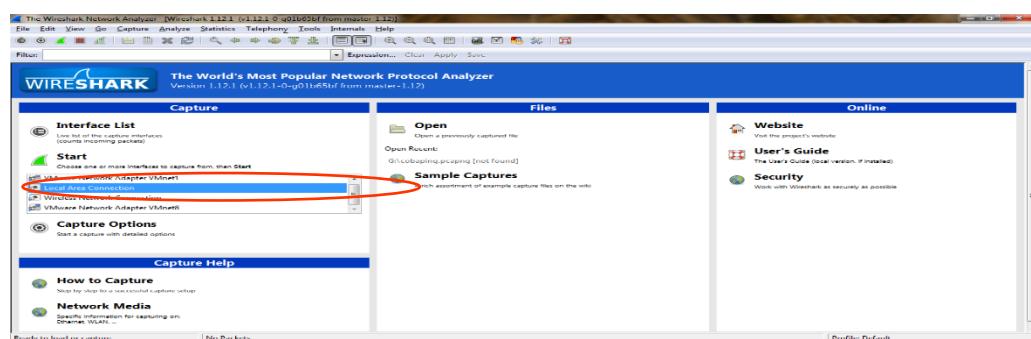
Beberapa langkah yang diperlukan untuk melakukan inspeksi *header* IPv4 dapat dilakukan sebagai berikut :

- 1) Buka aplikasi Wireshark Network Analyzer



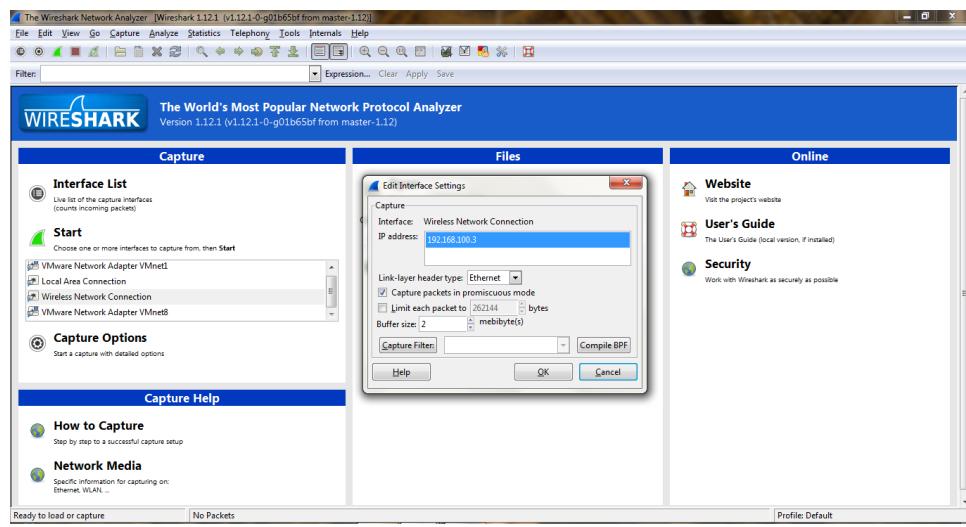
Gambar 4. 2 Tampilan awal Wireshark.

- 2) Pilihlah *Interface* yang akan *capture*



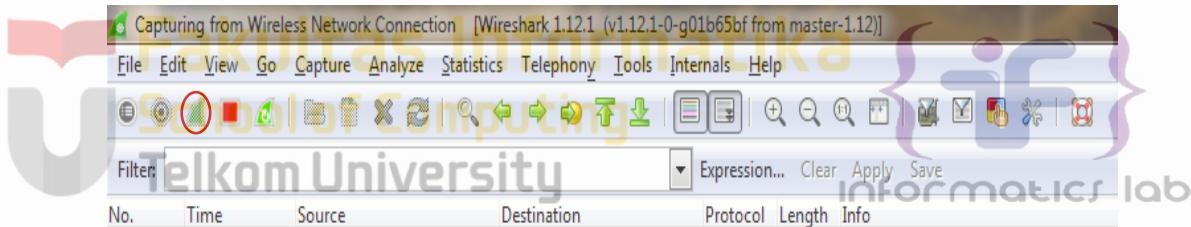
Gambar 4. 3 *interface* yang tersedia.

3) Definisikan alamat IP yang akan di-capture



Gambar 4. 4 Definisikan ip address.

4) Klik tombol *start* untuk mulai melakukan *capture* dari *interface* dan alamat IP yang diinginkan.



Gambar 4. 5 Tekan tombol start.

5) Lakukan *ping* terhadap alamat yang diinginkan.

User dengan alamat IP 192.168.100.3 melakukan *ping* (menggunakan *protocol ICMP*) terhadap alamat IP 74.125.130.100 dimana terlihat *reply* dari alamat IP 74.125.130.100 yang menandakan terbangunnya koneksi antara User dan alamat yang dituju.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Galuh>ping google.com

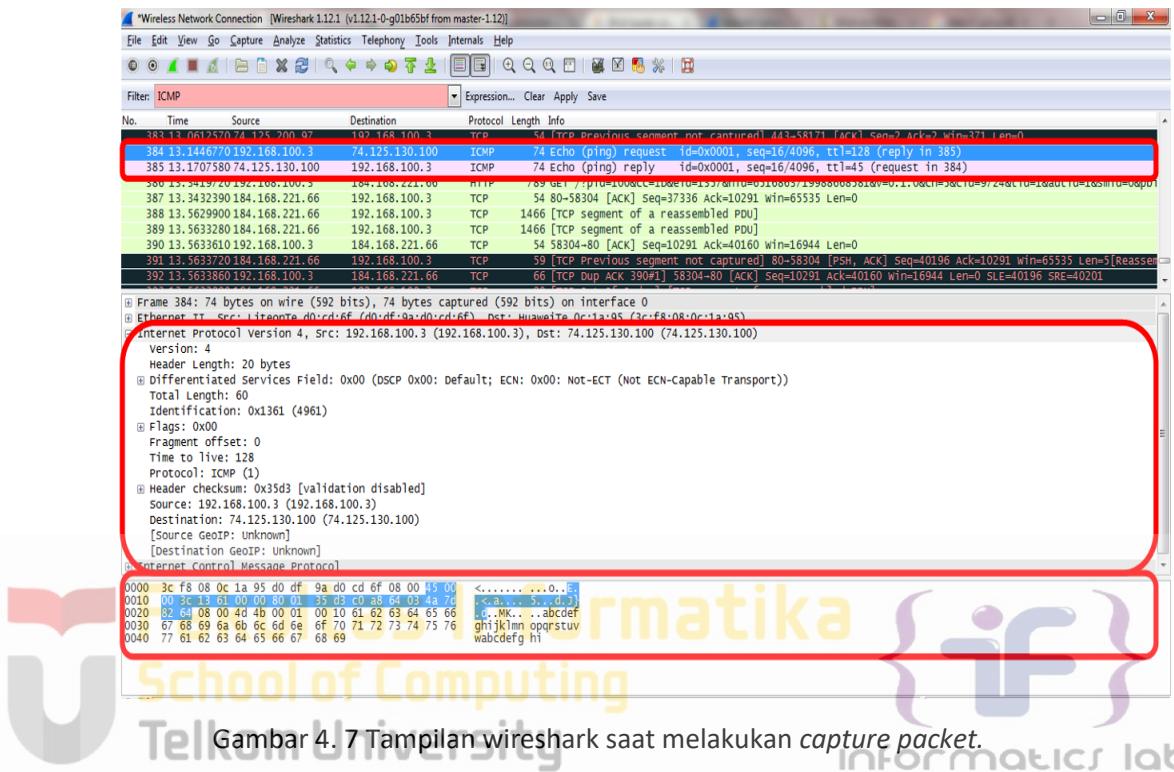
Pinging google.com [74.125.68.100] with 32 bytes of data:
Reply from 74.125.68.100: bytes=32 time=26ms TTL=45
Reply from 74.125.68.100: bytes=32 time=26ms TTL=45
Reply from 74.125.68.100: bytes=32 time=27ms TTL=45
Reply from 74.125.68.100: bytes=32 time=27ms TTL=45

Ping statistics for 74.125.68.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 26ms, Maximum = 27ms, Average = 26ms

C:\Users\Galuh>
```

Gambar 4. 6 Tampilan command prompt saat melakukan *ping*.

- 6) Selanjutnya lakukan proses inspeksi IPv4 *header* menggunakan Wireshark dapat dilihat melalui *middle pane Internet Protocol version 4. Detail* dari isi 20 byte *header* IPv4 tersebut dapat dilihat pada kolom paling bawah, dimana 20 byte kombinasi angka dan huruf tersebut merepresentasikan isi *header* IPv4.



Gambar 4.7 Tampilan wireshark saat melakukan *capture packet*.

Penjelasan *header* ipv4 paket ICMP yang ter-capture:

- 1) **Version** : pada paket tersebut jelas terlihat bahwa versi IP yang digunakan adalah IP versi 4.
- 2) **Header Length** : *header length* pada paket tersebut sebesar 20 bytes.
- 3) **Total Length** : total *length* yang dimiliki paket ini adalah sebanyak 60 bytes.
- 4) **Identification** : paket ini memiliki nomor *id* 0x1361 dalam bentuk heksadesimal.
- 5) **Flags** : *flags* dengan nilai 0x00.
- 6) **Fragment Offset** : *fragment offset*-nya bernilai 0.
- 7) **Time to Live** : paket ini masih memiliki nilai TTL sebesar 128 bit.
- 8) **Protocol** : pada paket ini *protocol* yang digunakan bernilai 1 yang berarti merupakan *protocol* ICMP.
- 9) **Header Checksum** : *header checksum* yang dimiliki bernilai 0x35d3 dalam bentuk heksadesimal.
- 10) **Source Address** : IP address pengirim adalah 192.168.100.3.
- 11) **Destination Address** : IP address tujuan paket ini adalah 74.125.130.100.

Modul 5 Static Routing

Tujuan Praktikum

1. Memahami penerapan *static routing*.
2. Mampu melakukan konfigurasi static routing pada *Cisco Packet Tracer*.

5.1 Pendahuluan

Routing merupakan metode yang digunakan untuk meneruskan paket-paket dari satu jaringan ke jaringan yang berbeda melalui sebuah *internetwork*. Untuk melakukan hal ini, digunakan sebuah perangkat jaringan bernama *router* dimana *router-router* yang berada pada jaringan akan menghubungkan jaringan satu dan yang lainnya agar dapat saling bertukar paket. Keputusan pembentukan *routing* pada *router* dilakukan berdasarkan *IP Address destination*. Agar *routing* yang dilakukan dapat secara benar mengantarkan paket yang dimaksud, *router* diharuskan mengenal topologi jaringan yang ada.

Terdapat 2 fungsi dasar dari *routing* yakni sebagai penentu jalur yang akan dilalui paket hingga ke tujuan dan melakukan fungsi *switching* karena sifatnya yang dapat meneruskan paket.

Untuk bisa melakukan *routing* paket, ada hal-hal yang harus diketahui :

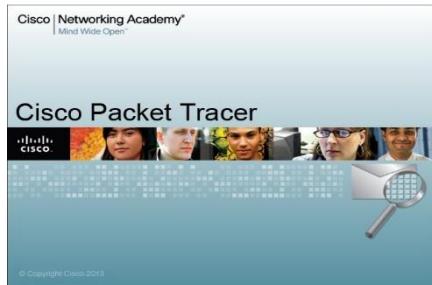
- 1) Alamat tujuan.
- 2) *Router-router* tetangga (untuk mempelajari topologi jaringan).
- 3) *Router* yang mungkin ke semua *network*.
- 4) *Router* terbaik untuk setiap *network*.

Terdapat beberapa jenis metode *routing* yang ada. Salah satunya adalah *static route*. *Static route* adalah metode *routing* jaringan dimana tabel jaringan dibangun secara manual oleh *administrator* jaringan. *Static routing* mengharuskan admin untuk merubah *route* atau memasukkan *command* secara manual di *router* setiap kali terjadi perubahan jalur. *Router* meneruskan paket dari sebuah jaringan ke jaringan lainnya berdasarkan *route* yang ditentukan oleh *administrator*. *Route* pada *static routing* tidak berubah, kecuali jika diubah secara *manual* oleh *administrator*.

5.2 Langkah-langkah Simulasi Static Routing pada Cisco Packet Tracer

Berikut urutan langkah-langkah simulasi *Static Routing* pada *Cisco Packet Tracer* :

- 1) *Install* dan buka aplikasi *Cisco Packet Tracer*.



Gambar 5. 1 Cisco Packet Tracer.

- 2) Siapkan *device* yang akan digunakan pada simulasi dengan ketentuan sebagai berikut.
 - a. Siapkan *router* dan pilihlah *Generic Router*. Pastikan didalamnya terdapat minimal 2 *serial port* dan 1 *fast ethernet port* yang akan digunakan pada simulasi.
Router ke router : Serial

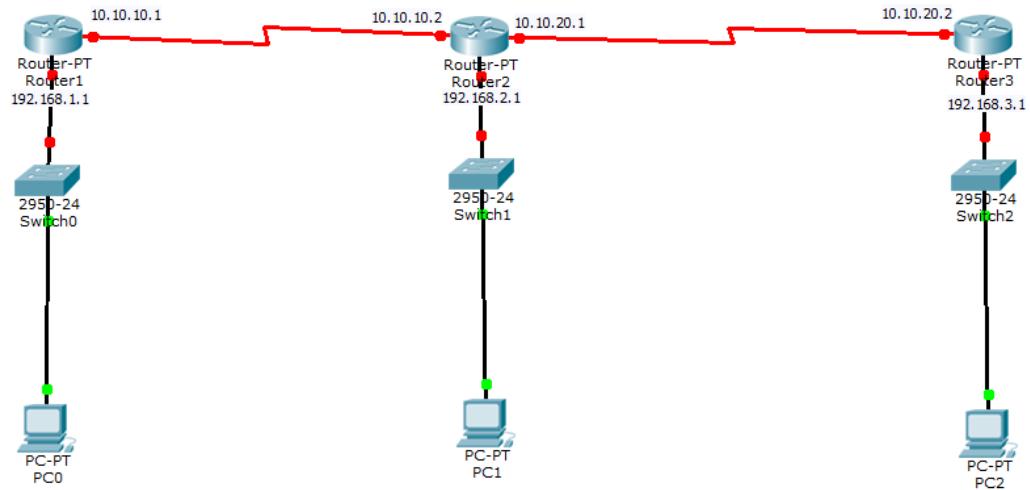
Router ke switch : FastEthernet
Switch ke PC : FastEthernet



Gambar 5. 2 Tampilan *router* pada Cisco *Packet Tracer*.

b. Pemilihan kabel:

- i. Gunakan kabel *Straight-through* untuk menghubungkan *device* jaringan yang berbeda jenisnya:
Router – Switch
Router – Hub
PC – Switch
PC – Hub
 - ii. Gunakan kabel *Cross-over* untuk menghubungkan *device* jaringan yang sama jenisnya:
Router - Router (pada simulasi ini konektor menggunakan DCE dan DTE)
Router – PC
Switch - Switch
Switch – Hub
 - iii. Gunakan konektor DCE dan DTE untuk menghubungkan antar *Router*:
DCE – Membutuhkan pendefinisian clock rate
DTE – Tidak membutuhkan pendefinisian clock rate
Pada *link* simulasi antar *router* dibutuhkan minimal 1 konektor DCE untuk menghubungkan antar *router*.
- 3) Buatlah topologi seperti pada gambar dibawah dengan menggunakan 3 buah *router*, 3 buah *switch* dan 3 buah *PC*.



Gambar 5. 3 Contoh topologi.

- 4) Lakukan konfigurasi pada *network device*:
- Mengubah *hostname* pada *router*.

Gunakan CLI (*Command Line Interface*) untuk melakukan konfigurasi pada *router*.

Router_A

```
Router>enable
Router#configure terminal
Router(config)#hostname Router_A
Router_A(config)#+
```

Router_B

```
Router>enable
Router#configure terminal
Router(config)#hostname Router_B
Router_B(config)#+
```

Router_C

```
Router>enable
Router#configure terminal
Router(config)#hostname Router_C
Router_C(config)#+
```

- b. Definisikan IP Address pada tiap *port router*

Setting alamat IP dan *subnet mask* pada tiap *port router* berdasarkan tabel berikut :

Table 4-1 Tabel Daftar IP Address.

Router	Interface	IP Address	Subnet Mask
Router_A	Fa0/0	192.168.1.1	255.25.255.0
	Se2/0	10.10.10.1	255.25.255.0
Router_B	Se2/0	10.10.10.2	255.25.255.0
	Fa0/0	192.168.2.1	255.25.255.0
	Se3/0	10.10.20.1	255.25.255.0
Router_C	Se3/0	10.10.20.2	255.25.255.0
	Fa0/0	192.168.3.1	255.25.255.0

Router_A

- 1) Interface Fast Ethernet 0/0 (sesuaikan dengan port yang digunakan)

```
Router_A(config)#int fa0/0
Router_A(config-if)#ip address 192.168.1.1 255.255.255.0
Router_A(config-if)#no shutdown
```

- 2) Interface Serial 2/0 (sesuaikan dengan port yang digunakan)

```
Router_A(config)#int se2/0
Router_A(config-if)#ip address 10.10.10.1 255.255.255.0
Router_A(config-if)# clock rate 64000
```

```
Router_A(config-if)#no shutdown
```

Router_B

1) Interface Fast Ethernet 0/0 (sesuaikan dengan port yang digunakan)

```
Router_B(config)#int fa0/0
```

```
Router_B(config-if)#ip address 192.168.2.1 255.255.255.0
```

```
Router_B(config-if)#no shutdown
```

2) Interface Serial 2/0 (sesuaikan dengan port yang digunakan)

```
Router_B(config)#int se2/0
```

```
Router_B(config-if)#ip address 10.10.10.2 255.255.255.0
```

```
Router_B(config-if)#no shutdown
```

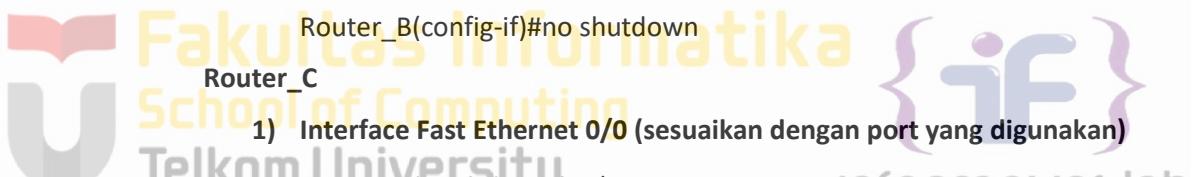
3) Interface Serial 3/0 (sesuaikan dengan port yang digunakan)

```
Router_B(config)#int se3/0
```

```
Router_B(config-if)#ip address 10.10.20.1 255.255.255.0
```

```
Router_B(config-if)# clock rate 64000
```

```
Router_B(config-if)#no shutdown
```



1) Interface Fast Ethernet 0/0 (sesuaikan dengan port yang digunakan)

```
Router_C(config)#int fa0/0
```

```
Router_C(config-if)#ip address 192.168.2.1 255.255.255.0
```

```
Router_C(config-if)#no shutdown
```

2) Interface Serial 2/0 (sesuaikan dengan port yang digunakan)

```
Router_C(config)#int se2/0
```

```
Router_C(config-if)#ip address 10.10.10.2 255.255.255.0
```

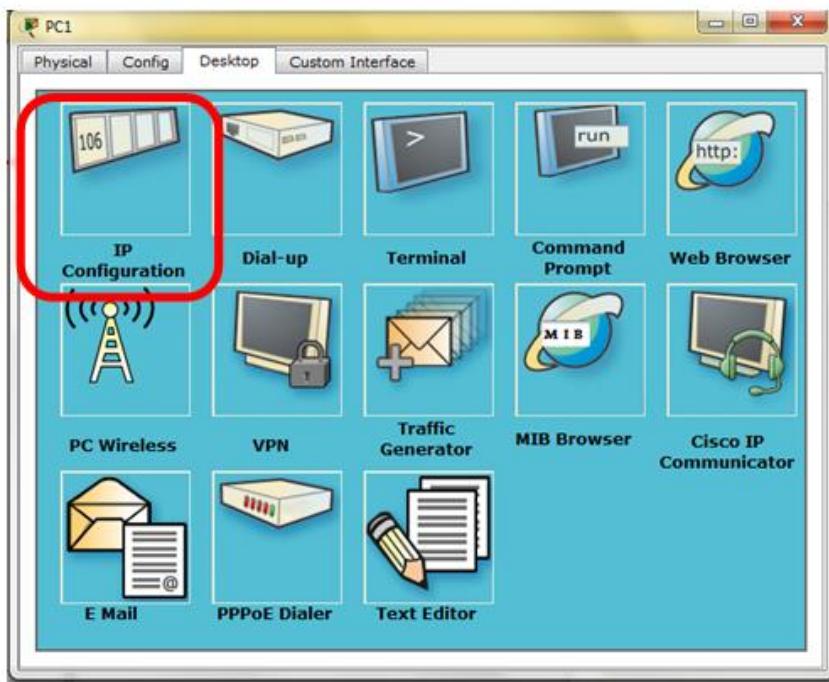
```
Router_C(config-if)#no shutdown
```

c. Definisikan IP Address pada tiap PC

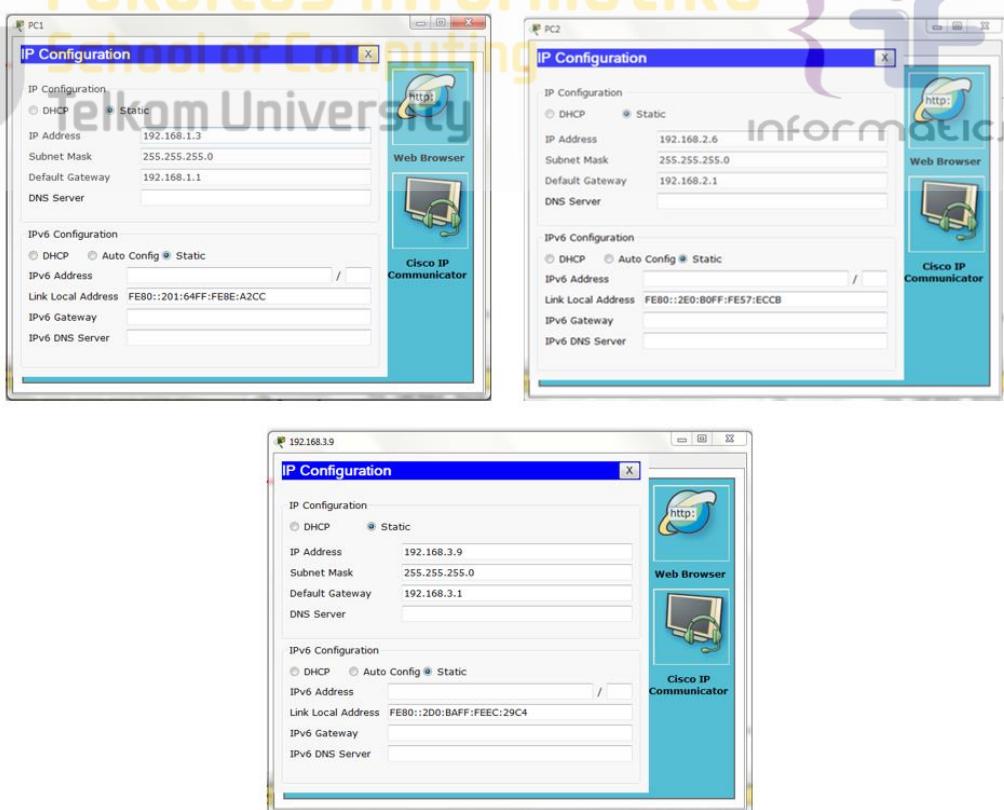
Setting alamat IP dan *subnetmask static* pada tiap PC menggunakan ‘IP Configuration’ berdasarkan tabel berikut :

Table 4-2 Tabel IP Configuration.

PC	IP Address	Subnet Mask	Default Gateway
1	192.168.1.3	255.255.255.0	192.168.1.1
2	192.168.2.6	255.255.255.0	192.168.2.1
3	192.168.3.9	255.255.255.0	192.168.3.1



Gambar 5. 4 Tampilan setting PC.



Gambar 5. 5 Tampilan IP Configuration.

Masukan IP Address, Subnet Mask dan Default Gateway yang disesuaikan dengan di jaringan mana PC berada.

d. Konfigurasi *Static Route*

Selanjutnya, agar antar PC yang berada pada jaringan berbeda dapat saling berkomunikasi, definisikanlah *static route* pada tiap-tiap *router* yang ada. Setelah itu pastikanlah bahwa *routing* telah benar-benar didefinisikan menggunakan perintah ‘*show ip route*’ pada *global configuration*.

Berikut ini adalah detail untuk masing-masing opsi :

- 1) *ip route*: perintah untuk membuat static *routing* itu sendiri.
- 2) *Destination: network* tujuan yang hendak ditambahkan ke *routing table*.
- 3) *Mask: subnetmask* yang digunakan dalam *network*.
- 4) *Next_hop_address: address* dari *hop router* selanjutnya, yakni yang akan menerima paket dan mem-forward-nya lagi ke *network* tujuan. Tidak lain berupa *interface router* dari *router* dari *network* yang terkoneksi secara langsung.

Formula yang digunakan pada *static route* pada Cisco *Packet Tracer* adalah sebagai berikut

#*ip route <Destination IP Address><Destination Mask><Next Hop Address>*

Router_A

```
Router_A(config)#ip route 192.168.2.0 255.255.255.0 10.10.10.2
Router_A(config)#ip route 10.10.20.0 255.255.255.0 10.10.10.2
Router_A(config)#ip route 192.168.3.0 255.255.255.0 10.10.10.2
```

#Melihat konfigurasi *routing*#

```
Router_A#show ip route
```

Codes: C - *connected*, S - *static*, I - *IGRP*, R - *RIP*, M - *mobile*, B - *BGP*
D - *EIGRP*, EX - *EIGRP external*, O - *OSPF*, IA - *OSPF inter area*
N1 - *OSPF NSSA external type 1*, N2 - *OSPF NSSA external type 2*
E1 - *OSPF external type 1*, E2 - *OSPF external type 2*, E - *EGP*
i - *IS-IS*, L1 - *IS-IS level-1*, L2 - *IS-IS level-2*, ia - *IS-IS inter area*
candidate default, U - *per-user static route*, o - *ODR*
P - *periodic downloaded static route*

Gateway of last resort is not set

```
10.0.0.0/24 is subnetted, 2 subnets
C 10.10.10.0 is directly connected, Serial2/0
S 10.10.20.0 [1/0] via 10.10.10.2
C 192.168.1.0/24 is directly connected, FastEthernet0/0
S 192.168.2.0/24 [1/0] via 10.10.10.2
S 192.168.3.0/24 [1/0] via 10.10.10.2
```

Router_B

```
Router_B(config)#ip route 192.168.1.0 255.255.255.0 10.10.10.1
Router_B(config)#ip route 192.168.3.0 255.255.255.0 10.10.20.2
```

#Melihat konfigurasi routing#

Router_B#sh ip route

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/24 is subnetted, 2 subnets

C 10.10.10.0 is directly connected, Serial2/0
C 10.10.20.0 is directly connected, Serial3/0
S 192.168.1.0/24 [1/0] via 10.10.10.1
C 192.168.2.0/24 is directly connected, FastEthernet0/0
S 192.168.3.0/24 [1/0] via 10.10.20.2

Router_C

Router_C(config)#ip route 192.168.2.0 255.255.255.0 10.10.20.1

Router_C(config)#ip route 10.10.10.0 255.255.255.0 10.10.20.1

Router_C(config)#ip route 192.168.1.0 255.255.255.0 10.10.20.1

#Melihat konfigurasi routing#

Router_C#sh ip route

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/24 is subnetted, 2 subnets

S 10.10.10.0 [1/0] via 10.10.20.1
C 10.10.20.0 is directly connected, Serial3/0
S 192.168.1.0/24 [1/0] via 10.10.20.1
S 192.168.2.0/24 [1/0] via 10.10.20.1
C 192.168.3.0/24 is directly connected, FastEthernet0/0

e. Tes Konektivitas

Setelah dilakukan pendefinisian *static route* pada tiap *router*, ujilah konektivitas antar PC pada jaringan yang berbeda menggunakan perintah ‘*ping*’ pada ‘*Command Prompt*’. Pastikan terdapat *reply message* yang menandakan terjalinnya konektivitas antar jaringan yang berbeda.



Gambar 5. 6 Tampilan setting PC.

PC1

Ping PC1 → PC2 & Ping PC1 → PC3

```

Packet Tracer PC Command Line 1.0
PCping 192.168.2.6

Pinging 192.168.2.6 with 32 bytes of data:
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=2ms TTL=126

Ping statistics for 192.168.2.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
PCping 192.168.3.9

Pinging 192.168.3.9 with 32 bytes of data:

Reply from 192.168.3.9: bytes=32 time=2ms TTL=125
Reply from 192.168.3.9: bytes=32 time=0ms TTL=125
Reply from 192.168.3.9: bytes=32 time=11ms TTL=125
Reply from 192.168.3.9: bytes=32 time=2ms TTL=125

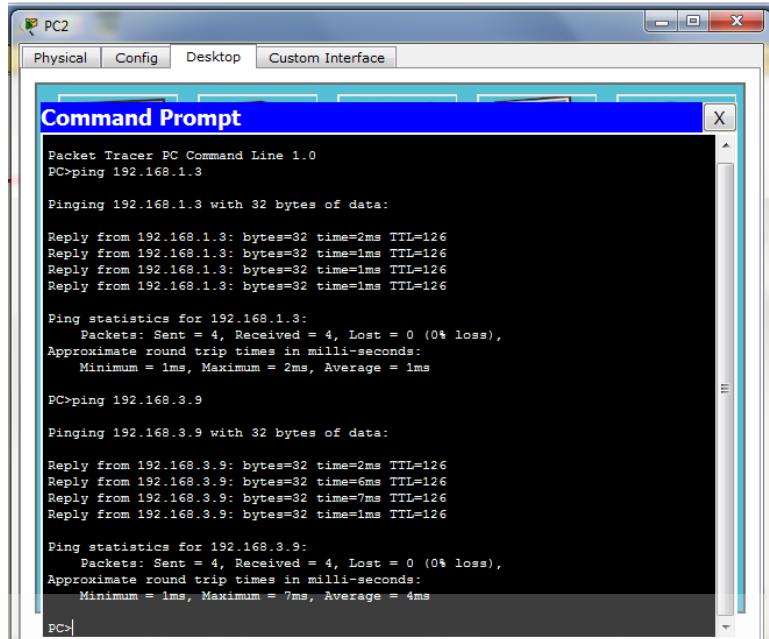
Ping statistics for 192.168.3.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 11ms, Average = 5ms
PC>

```

Gambar 5. 7 Tampilan command prompt PC1.

PC2

Ping PC2 → PC1 & Ping PC2 → PC3



```
PC2
Physical Config Desktop Custom Interface

Command Prompt
X

Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:
Reply from 192.168.1.3: bytes=32 time=2ms TTL=126
Reply from 192.168.1.3: bytes=32 time=1ms TTL=126
Reply from 192.168.1.3: bytes=32 time=1ms TTL=126
Reply from 192.168.1.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

PC>ping 192.168.3.9

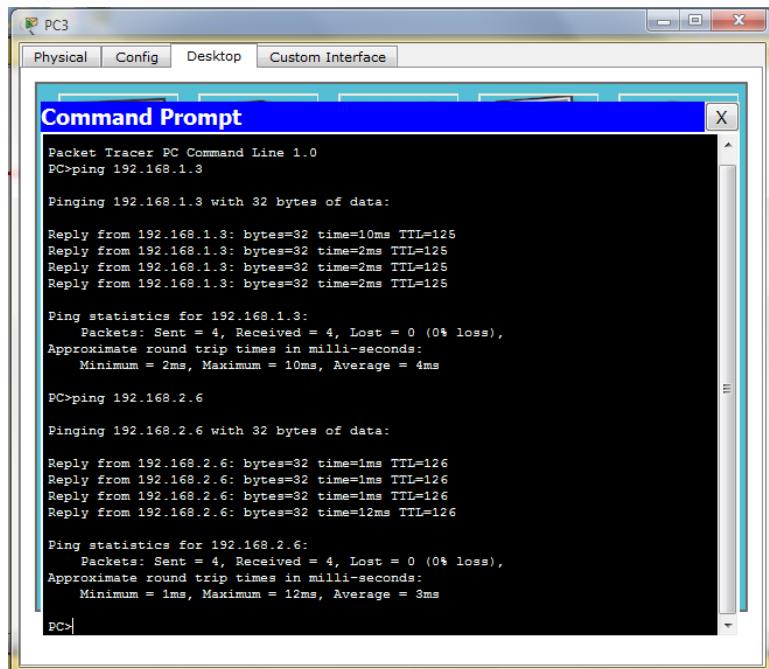
Pinging 192.168.3.9 with 32 bytes of data:
Reply from 192.168.3.9: bytes=32 time=2ms TTL=126
Reply from 192.168.3.9: bytes=32 time=6ms TTL=126
Reply from 192.168.3.9: bytes=32 time=7ms TTL=126
Reply from 192.168.3.9: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.3.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 7ms, Average = 4ms

PC>
```



Gambar 5. 8 Tampilan command prompt PC2.



```
PC3
Physical Config Desktop Custom Interface

Command Prompt
X

Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:
Reply from 192.168.1.3: bytes=32 time=10ms TTL=125
Reply from 192.168.1.3: bytes=32 time=2ms TTL=125
Reply from 192.168.1.3: bytes=32 time=2ms TTL=125
Reply from 192.168.1.3: bytes=32 time=2ms TTL=125

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 10ms, Average = 4ms

PC>ping 192.168.2.6

Pinging 192.168.2.6 with 32 bytes of data:
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=1ms TTL=126
Reply from 192.168.2.6: bytes=32 time=12ms TTL=126

Ping statistics for 192.168.2.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 12ms, Average = 3ms

PC>
```

Gambar 5. 9 Tampilan *command prompt*.

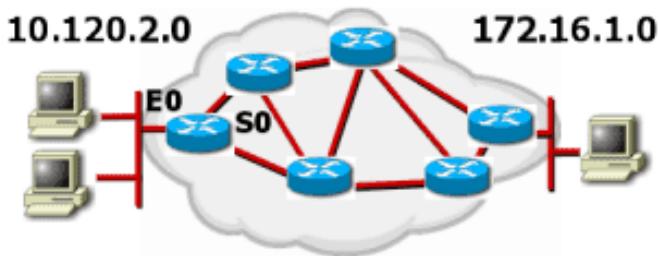
Modul 6 Dynamic Routing

Tujuan Praktikum

1. Memahami konsep *routing*, khususnya OSPF *Routing*.

6.1 Pendahuluan

Routing merupakan proses pencarian *path* atau alur guna memindahkan informasi dari *host sumber (source address)* ke *host tujuan (destinations address)* melalui koneksi *internetwork*.



Gambar 6. 1 Ilustrasi *routing*.

Router menyaring (*filter*) lalu lintas data. Penyaringan dilakukan bukan dengan melihat alamat paket data, tetapi dengan menggunakan protokol tertentu. Router muncul untuk menangani perlunya membagi jaringan secara logikal bukan fisikal. Sebuah IP *router* bisa membagi jaringan menjadi beberapa *subnet* sehingga hanya lalu lintas yang ditujukan untuk IP *address* tertentu yang bisa mengalir dari satu segmen ke segmen lain. akan menggunakan *router* ketika akan menghubungkan jaringan komputer ke jaringan lain, baik jaringan pribadi (LAN/WAN) atau jaringan publik (*Internet*).

Diperlukan adanya *router* untuk melakukan *routing* di dalam jaringan, dimana *router* membutuhkan informasi-informasi sebagai berikut:

- 1) **Alamat Tujuan/Destination Address** - Tujuan atau alamat *item* yang akan di-*routing*
- 2) **Mengenal sumber informasi** - Dari mana sumber (*router* lain) yang dapat dipelajari oleh *router* dan memberikan jalur sampai ke tujuan.
- 3) **Menemukan rute** - Rute atau jalur mana yang mungkin diambil sampai ke tujuan.
- 4) **Pemilihan rute** - Rute yang terbaik yang diambil untuk sampai ke tujuan.
- 5) **Menjaga informasi routing** - Suatu cara untuk menjaga jalur sampai ke tujuan yang sudah diketahui dan paling sering dilalui.

Analogi:

Misalkan berada pada persimpangan jalan, mungkin akan merasa bingung jika tidak ada petunjuk jalan, di setiap persimpangan jalan (*router*) seharusnya ada petunjuk jalan supaya orang tidak bingung dan tersesat. Untuk jalan yang rumit dan berputar-putar tidaklah cukup jika menggunakan *static routing*. Tentunya akan merasa bingung jika disetiap persimpangan harus bertanya pada orang apalagi kepada orang yang tidak tahu. Oleh karena itu disini diperlukan *dynamic routing*, analoginya seperti ada polisi yang membawa HT dan memberikan jalur mana saja yang bisa dilewati. Polisi akan selalu koordinasi beberapa kali sehari, agar jika ada jalan yang macet, ada tabrakan, ada pohon rubuh, polisi akan segera meng-update petunjuk jalan yang lain.

Biasanya polisi yang bertingkat rendah akan memakai HT yang sebut sebagai *RIP*, yang memiliki jarak paling jauh 30 *hop* (simpangan). Polisi yang berada pada tempat yang ramai bisa menggunakan *ISIS* atau *OSPF*, biasanya sudah membawa HP maupun PDA jadi akan lebih pintar dan cepat untuk

melakukan *update*. Polisi tingkat dunia biasanya memiliki kantor pada persimpangan dan sudah mempunyai peralatan pengacak jaringan seluruh dunia, ini disebut BGP.

Ada dua bagian *routing* paket IP:

- 1) Bagaimana meneruskan paket dari *interface input* ke *interface output* pada suatu *router* (“*IP forwarding*”)?
 - a. Paket biasanya diteruskan (*forwarding*) kesejumlah *router* sebelum mencapai *host* tujuan
 - b. IP *forwarding* dilaksanakan atas dasar *hop-by-hop* yaitu tidak ada yang tau *rute* yang lengkap. Tujuan *forwarding* adalah membawa paket IP lebih dekat ke tujuan
- 2) Bagaimana mencari dan men-setup rute (“*Routing algorithm*”)?

Protokol *routing* membentuk suatu tabel *routing* yang digunakan untuk menyeleksi jalur yang akan digunakan. Didalam tabel *routing* terdapat suatu alamat tujuan paket data dan *hop* yaitu suatu *router* yang akan dituju setelah *router* tersebut.

Konsep berikut sangatlah penting untuk memahami *routing* pada jaringan IP:

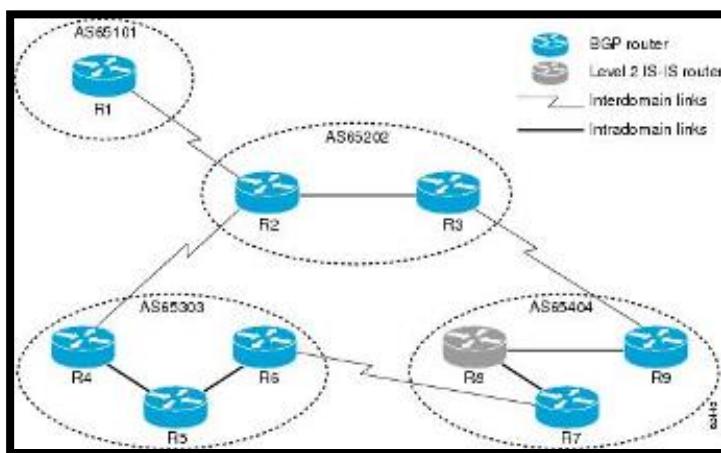
- a. *Autonomous system*
- b. Interdomain vs. intradomain *routing*
- c. *Distance vector* vs. *link state routing algorithms*

6.1.1 Autonomous System (AS)

Suatu *autonomous system* adalah bagian *logical* dari jaringan IP yang besar, biasanya dimiliki oleh sebuah organisasi jaringan dan diadministrasikan oleh sebuah manajemen resmi. Setiap *router* dapat berkomunikasi dengan *router* yang lain dalam satu *autonomous system*.

Contoh dari *autonomous region* adalah:

- 1) Internet Service Provider Regional
- 2) Jaringan kampus ITB



Gambar 6. 2 Ilustrasi Autonomous System.

Di dalam *autonomous system*, *routing* dilaksanakan secara:

- 1) **Intradomain Routing** yaitu dalam *autonomous system*
- 2) **Interdomain Routing** yaitu antara *autonomous system*

6.1.2 Perbedaan Intradomain Routing dan Interdomain Routing

Intradomain Routing	Interdomain Routing
Routing di dalam suatu AS	Routing antara AS
Protokol untuk Intradomain routing juga disebut <i>Interior Gateway Protocol / IGP</i> . Protokol yang populer: 1) RIP (sederhana, lama) 2) OSPF (lebih baik)	Protokol untuk <i>interdomain routing</i> disebut <i>Exterior Gateway Protocol/ EGP</i> Protokol <i>routing</i> : 1) EGP 2) BGP (lebih baru)
Mengabaikan Internet di luar AS	Mengasumsikan Internet terdiri dari sekumpulan <i>interkoneksi AS</i> .

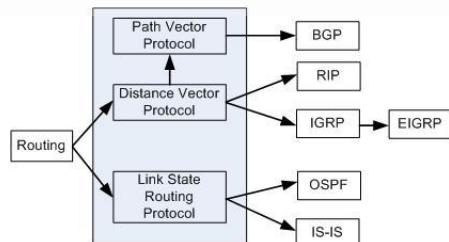
6.1.3 Klasifikasi Dynamic Routing

Algoritma	Protokol Routing
Distance Vector	RIPv1
	RIPv2
	IGRP
	EGRP
Link State	OSPF
	IS-IS

6.1.4 Distance vector vs. link state routing algorithms

Perbedaan mendasar antara *distance vector* dan *link state* adalah:

- 1) *Distance Vector* hanya memiliki informasi *routing* dari *router* tetangganya, sedangkan *Link State* memiliki informasi *routing* dari setiap *node* yang ada.
- 2) Untuk mendapatkan lintasan/rute yang terbaik, *Distance Vector* menggunakan Algoritma *Bellman-Ford*, sedangkan *Link State* menggunakan Algoritma *Dijkstra*.



Gambar 6. 3 Hirarki algoritma *routing*.

6.1.4.1 Distance Vector

Pembentukan tabel *routing* pada *Distance Vector* dilakukan dengan cara tiap-tiap *router* atau *PC router* akan saling bertukar informasi *routing* dengan *router* atau *PC router* yang terhubung langsung. Proses pertukaran informasi *routing* dilakukan secara periodik, misal tiap 30 detik.

Proses pembentukan tabel pada protokol *routing* yang menggunakan konsep *distance vector* adalah sebagai berikut:

- 1) Mula-mula tabel *routing* yang dimiliki oleh masing-masing *router* atau *PC router* akan berisi informasi alamat jaringan yang terhubung langsung dengan *router* atau *PC router* tersebut.
- 2) Secara periodik masing-masing *router* atau *PC router* akan saling bertukar informasi sehingga isi tabel *routing* dari semua *router* terisi lengkap (*converged*).

Routing Information Protocol (RIP)

Routing protokol yang menggunakan algoritma *distance vector*, yaitu algoritma *Bellman-Ford*. Pertama kali dikenalkan pada tahun 1969 dan merupakan algoritma *routing* yang pertama pada ARPANET. Versi awal dari *routing* protokol ini dibuat oleh *Xerox Parc's PARC Universal Packet Internetworking* dengan nama *Gateway Internet Protocol*. Kemudian diganti nama menjadi *Router Information Protocol (RIP)* yang merupakan bagian *Xerox network Services*.

RIP yang merupakan *routing protocol* dengan algoritma *distance vector*, yang menghitung jumlah *hop* (*count hop*) sebagai *routing metric*. Jumlah maksimum dari *hop* yang diperbolehkan adalah 15 *hop*. Tiap RIP *router* saling tukar informasi *routing* tiap 30 detik, melalui UDP port 520. Untuk menghindari *loop routing*, digunakan teknik *split horizon with poison reverse*. RIP merupakan *routing protocol* yang paling mudah untuk di konfigurasi.

RIP memiliki 3 versi yaitu RIPv1, RIPv2, dan RIPng.

6.1.4.2 Link State

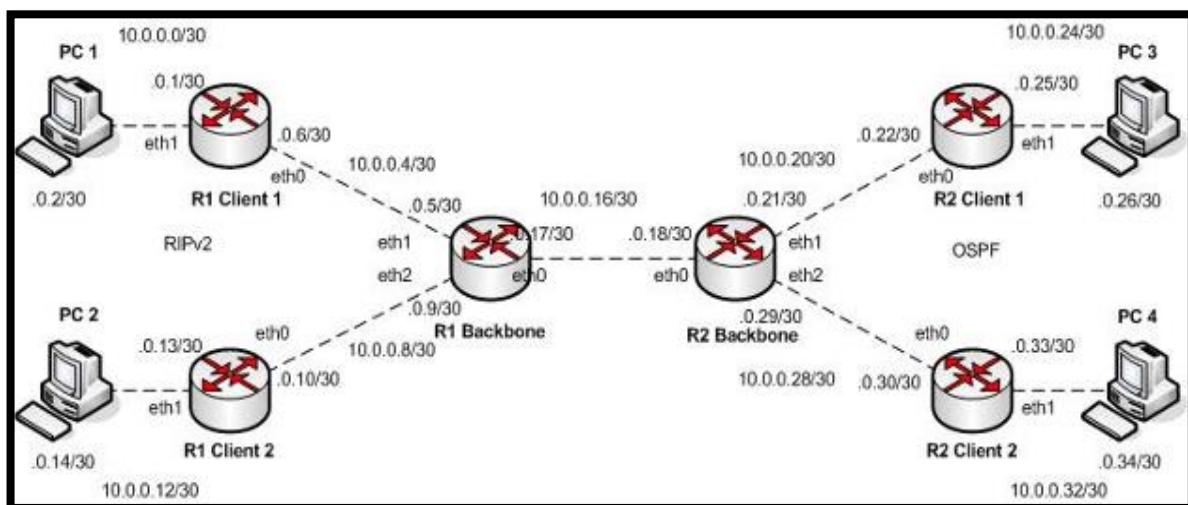
Protokol *routing* yang menggunakan konsep *link state* akan membentuk tabel *routing* menurut pandangan atau perhitungan *router* atau PC *router* masing-masing, tidak bergantung pada pendapat *router* atau PC *router* tetangga.

Tabel *routing* yang dibentuk dengan menggunakan konsep *link state* dilakukan melalui beberapa tahapan sebagai berikut:

- 1) Pada awalnya setiap *router* atau PC *router* akan saling mengirimkan dan melewaskan paket *link state*.
- 2) Paket *link state* yang diterima dari *router* atau PC *router* lain dikumpulkan dalam sebuah *database topologi*.
- 3) Berdasarkan informasi yang terkumpul di dalam *database*, *router* atau PC *router* melakukan perhitungan dengan menggunakan algoritma *short path first* (SPF).
- 4) Algoritma SPF menghasilkan *short path first tree*.
- 5) Akhirnya SPF Tree membentuk daftar isi tabel *routing*.

Kelima proses di atas dilakukan oleh masing-masing *router* atau PC *router*. Jika terjadi perubahan topologi jaringan, pemberitahuannya akan dikirimkan segera ke tiap-tiap *router* atau PC *router* sehingga proses *update* informasi *routing* dapat segera dilakukan.

Open Shortest Path First (OSPF)



Gambar 6. 4 Ilustrasi *routing* protokol OSPF.

OSPF merupakan *routing protocol* berbasis *link state*, termasuk dalam *Interior Gateway Protocol* (IGP). Menggunakan algoritma *Dijkstra* untuk menghitung *Shortest Path First* (SPF). Menggunakan *cost* sebagai *routing metric*. Setelah antar *router* bertukar informasi maka akan terbentuk *database link state* pada masing-masing *router*. OSPF mungkin merupakan IGP yang paling banyak digunakan. Menggunakan metode MD5 untuk autentikasi antar *router* sebelum menerima *Link-state Advertisement* (LSA). Dari awal OSPF sudah mendukung CIDR dan VLSM, berbeda dengan RIP. Bahkan untuk OSPFv3 sudah mendukung untuk IPv6. OSPF tidak menggunakan TCP atau UDP melainkan IP protokol 89.

OSPF memiliki 3 table di dalam router:

1) Routing table

Routing table biasa juga disebut sebagai *Forwarding database*. *Database* ini berisi the *lowest cost* untuk mencapai *router-router/network-network* lainnya. Setiap *router* mempunyai *Routing table* yang berbeda-beda.

2) Adjacency database

Database ini berisi semua *router* tetangganya. Setiap *router* mempunyai *Adjacency database* yang berbeda-beda.

3) Topological database

Database ini berisi seluruh informasi tentang *router* yang berada dalam satu *network-nya/areanya*.

Keuntungan dari OSPF adalah:

- 1) OSPF menggunakan pembagian jaringan berdasarkan konsep *area*.
- 2) Konsep jaringannya yang hirarki, sehingga membuat proses *update* informasinya lebih termanajemen dengan baik.
- 3) Adanya *Convergence*, dimana *router* akan menerima informasi dari *router* lain yang bertindak sebagai tetangganya, sehingga pada akhirnya seluruh informasi yang ada dalam sebuah jaringan dapat diketahui oleh semua *router* yang ada dalam jaringan.
- 4) Sistem *update* informasi *routing* yang cukup teratur.
- 5) OSPF menghemat penggunaan *bandwidth* jaringan.
- 6) OSPF menggunakan *cost* sebagai *metric*.

a. Metriks *Routing*

Metriks *Routing* terdiri dari:

- i. *Hop count*, berdasarkan pada banyaknya *router* atau PC *router* yang dilewati.
- ii. *Ticks*, berdasarkan waktu yang diperlukan dengan satuan waktu *ticks*.
- iii. *Cost*, berdasarkan perbandingan sebuah nilai patokan *standard* dengan *bandwidth* yang tersedia.
- iv. *Composite Metric*, berdasarkan hasil perhitungan dari parameter-parameter berikut:
 1. *Bandwidth*
 2. *Delay*
 3. *Load*
 4. *Reliability*
 5. *MTU (Maximum Transmit Unit)*

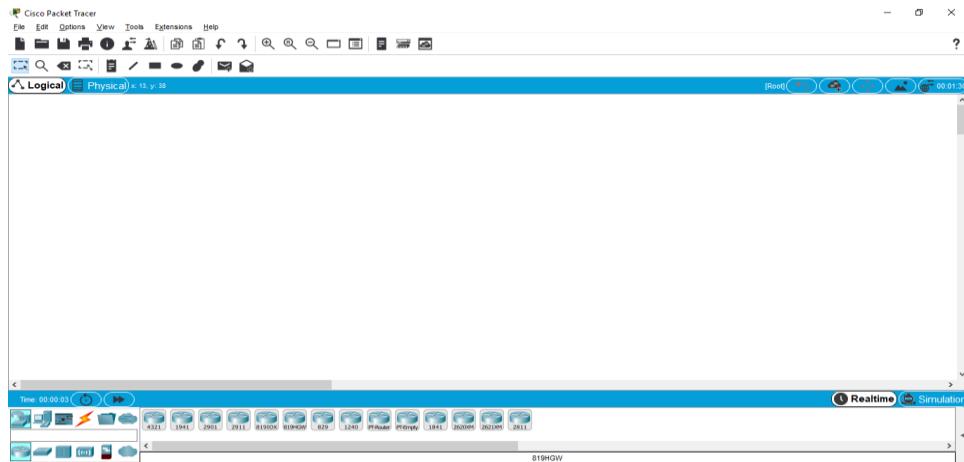
Metriks *Routing* pada OSPF adalah menggunakan *Cost*.

- b. Metriks *Routing* pada *RIP* adalah menggunakan banyaknya lompatan (*hop count*).

6.2 Langkah Simulasi Dynamic Routing pada Cisco Packet Tracer

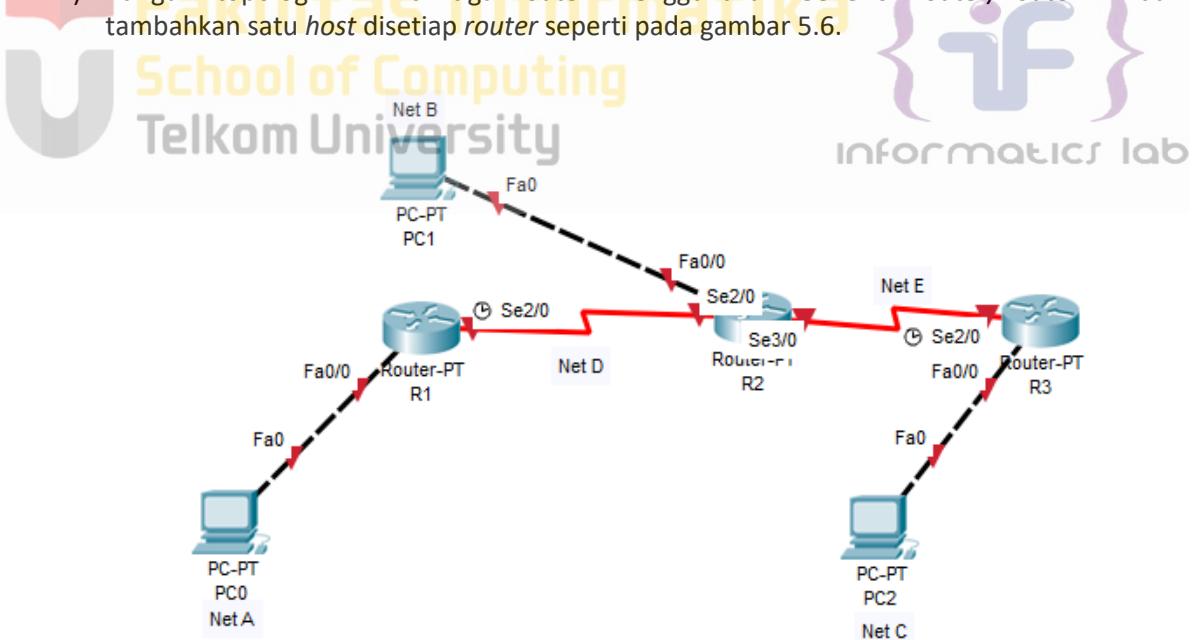
Berikut ini adalah langkah-langkah simulasi *dynamic routing* menggunakan protokol RIPv2 pada *Cisco Packet Tracer*:

- 1) Jalankan *Cisco Packet Tracer*.



Gambar 6. 5 Cisco Packet Tracer.

- 2) Bangun topologi minimal tiga router menggunakan *Generic Router/Router-PT* dan tambahkan satu host disetiap router seperti pada gambar 5.6.



Gambar 6. 6 Contoh topologi.

- 3) Konfigurasi alamat IP seperti pada gambar 5.7.

Subnet Name	Needed Size	Allocated Size	Address	Mask	Dec Mask	Assignable Range	Broadcast
Net C	95	126	10.0.0.0	/25	255.255.255.128	10.0.0.1 - 10.0.0.126	10.0.0.127
Net A	17	30	10.0.0.128	/27	255.255.255.224	10.0.0.129 - 10.0.0.158	10.0.0.159
Net B	12	14	10.0.0.160	/28	255.255.255.240	10.0.0.161 - 10.0.0.174	10.0.0.175
Net D	2	2	10.0.0.176	/30	255.255.255.252	10.0.0.177 - 10.0.0.178	10.0.0.179
Net E	2	2	10.0.0.180	/30	255.255.255.252	10.0.0.181 - 10.0.0.182	10.0.0.183

Gambar 6. 7 Initial Configuration Dialog.

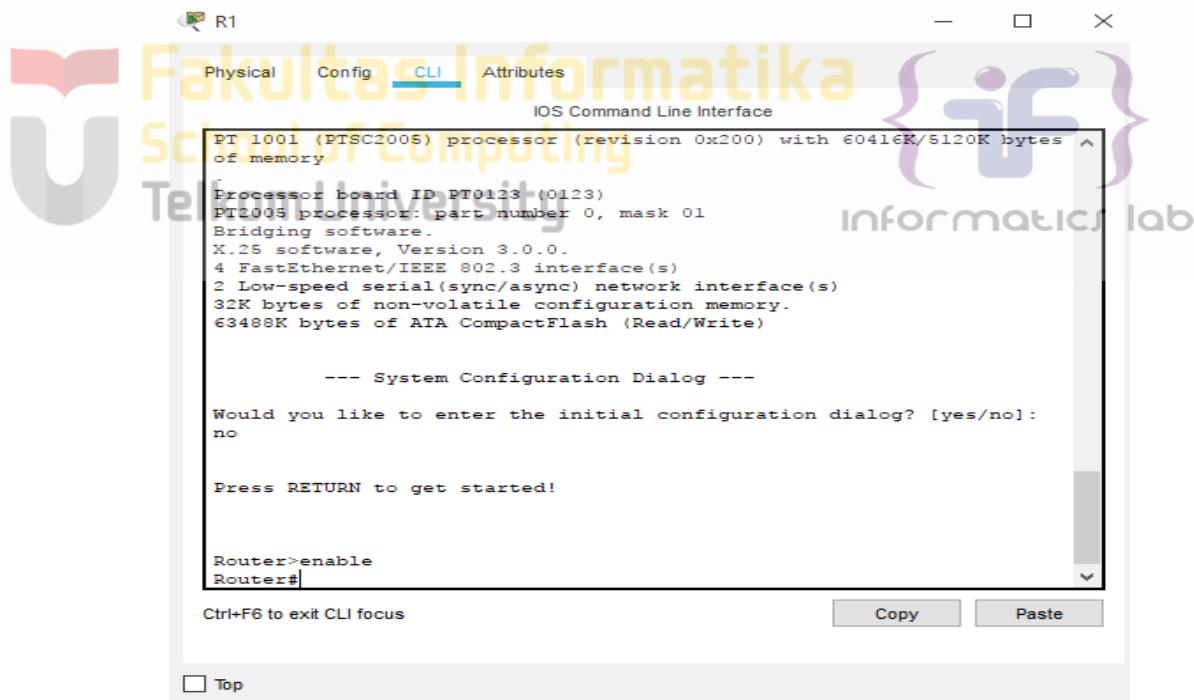
- 4) Setelah selesai mengkonfigurasi alamat IP, buka terminal pada *router* yang akan dikonfigurasi dengan cara meng-click *router* yang akan dikonfigurasi dan pilih tab CLI.
 5) Apabila muncul tampilan seperti gambar 5-8 Tulis "no" dan tekan *Enter*.

```
--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: |
```

Gambar 6. 8 Initial Configuration Dialog.

- 6) Masuk dengan mode *admin* dengan perintah 'enable'.



```
R1
Physical Config CLI Attributes
IOS Command Line Interface

PT 1001 (PTSC2005) processor (revision 0x200) with 60416K/6120K bytes ^ of memory
Processor board ID PT0123 (0123)
PTQ009 processor part number 0, mask 01
Bridging software.
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)

--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: no

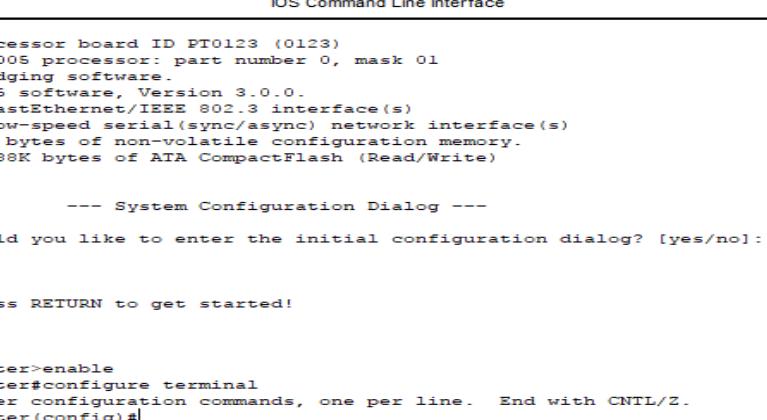
Press RETURN to get started!

Router>enable
Router#|
```

The screenshot shows the Cisco IOS CLI interface for a router named 'R1'. It displays system hardware details, including a PT 1001 processor and four FastEthernet interfaces. A configuration dialog box is open, asking if the user wants to enter the initial configuration dialog, with the response 'no' entered. Below the dialog, it says 'Press RETURN to get started!'. At the bottom, there are 'enable' and '#' prompts, indicating the user has entered privileged mode ('enable') and is now at the router's command line ('#').

Gambar 6. 9 Masuk mode *admin*.

- 7) Masuk ke mode terminal konfigurasi dengan perintah '*configure terminal*'.



R1

Physical Config **CLI** Attributes

IOS Command Line Interface

```
Processor board ID PT0123 (0123)
PT2005 processor: part number 0, mask 01
Bridging software.
X.25 software, Version 3.0.0.
4 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
32K bytes of non-volatile configuration memory.
63488K bytes of ATA CompactFlash (Read/Write)

--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: no

Press RETURN to get started!

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
```

Ctrl+F6 to exit CLI focus

Gambar 6. 10 Terminal konfigurasi.

- 8) Masuk ke konfigurasi *routing* RIP dengan perintah 'router rip'



R1

Physical Config **CLI** Attributes

IOS Command Line Interface

end	Exit from configure mode
exit	Exit from configure mode
hostname	Set system's network name
interface	Select an interface to configure
ip	Global IP configuration subcommands
key	Key management
line	Configure a terminal line
lldp	Global LLDP configuration subcommands
logging	Modify message logging facilities
no	Negate a command or set its defaults
ntp	Configure NTP
policy-map	Configure QoS Policy Map
priority-list	Build a priority list
privilege	Command privilege parameters
queue-list	Build a custom queue list
radius-server	Modify Radius query parameters
router	Enable a routing process
service	Modify use of network based services
snmp-server	Modify SNMP engine parameters
tacacs-server	Modify TACACS query parameters
username	Establish User Name Authentication

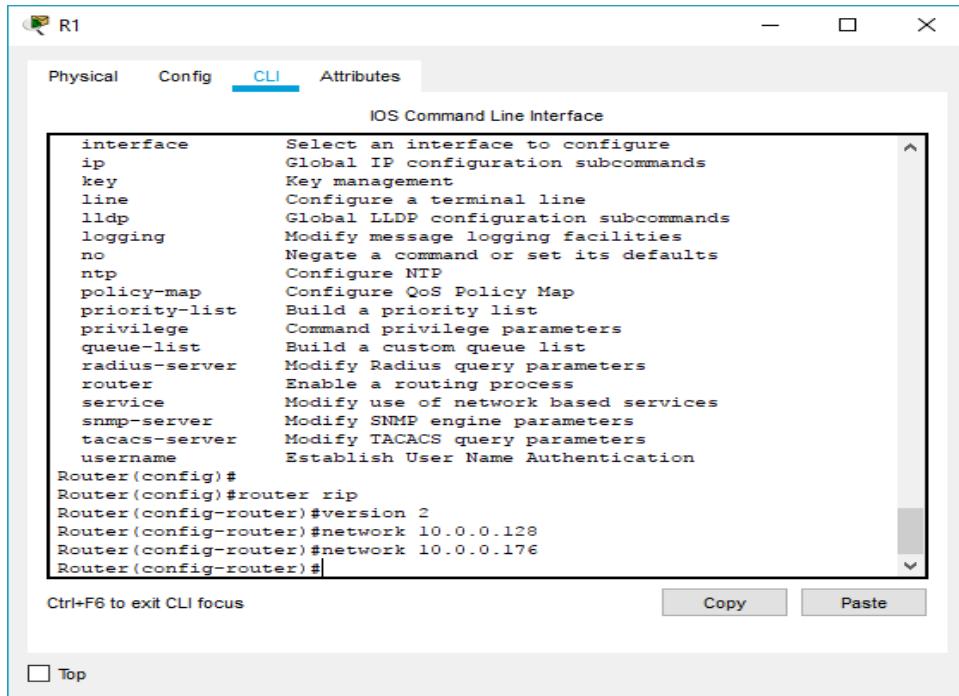
```
Router(config)#  
Router(config)#router rip  
Router(config-router) #
```

Ctrl+F6 to exit CLI focus

Top

Gambar 6. 11 Konfigurasi router RIP.

- 9) Pilih RIP versi 2 dengan perintah ‘version 2’.
 - 10) Masukan *network ID* yang terhubung langsung dengan *router* yang sedang dikonfigurasi. Contoh, pada R1 ada dua *network ID* yang langsung terhubung yaitu Net A dan Net D. Masukan kedua *network ID* tersebut dengan perintah ‘*network [network ID]*’.



Gambar 6. 12 Konfigurasi RIPv2.

- 11) Lakukan hal serupa ke semua *router* yang ada.
- 12) Setelah selesai lakukan pengujian koneksi dengan perintah ping dari *host* ke *host* lainnya.

```

C:\>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=3ms TTL=128
Reply from 10.0.0.2: bytes=32 time=3ms TTL=128
Reply from 10.0.0.2: bytes=32 time=4ms TTL=128
Reply from 10.0.0.2: bytes=32 time=2ms TTL=128

Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 4ms, Average = 3ms

C:\>ping 10.0.0.162

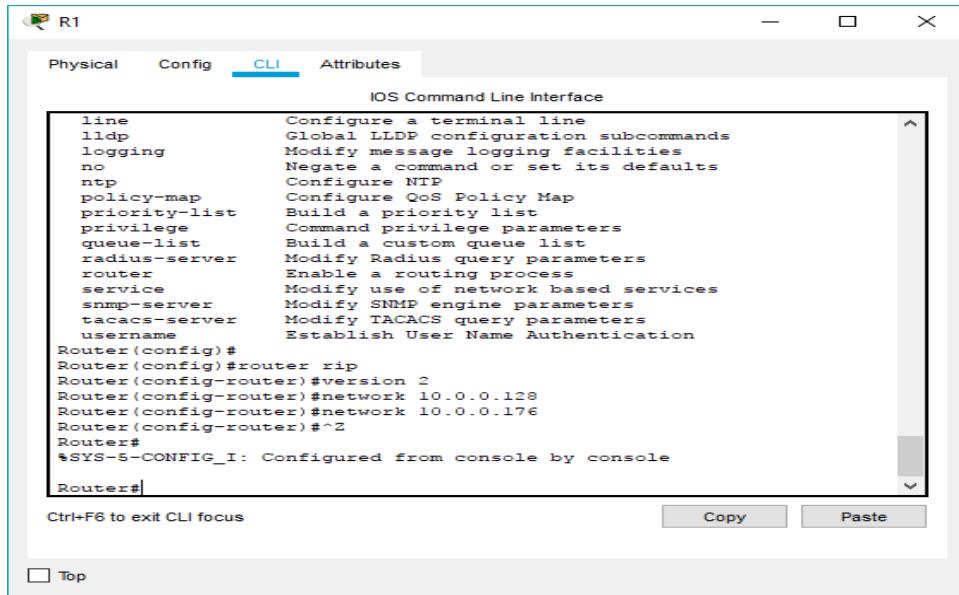
Pinging 10.0.0.162 with 32 bytes of data:
Reply from 10.0.0.162: bytes=32 time=8ms TTL=126
Reply from 10.0.0.162: bytes=32 time=5ms TTL=126
Reply from 10.0.0.162: bytes=32 time=1ms TTL=126
Reply from 10.0.0.162: bytes=32 time=5ms TTL=126

Ping statistics for 10.0.0.162:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 8ms, Average = 4ms

```

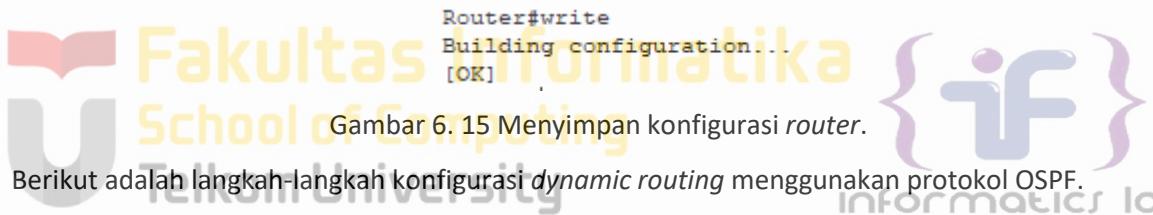
Gambar 6. 13 Pengujian koneksi.

- 13) Setelah semua terhubung, kembali pada terminal *router* dan masuk ke menu awal masuk sebagai *admin* dengan menekan tombol CTRL + Z.



Gambar 6. 14 Menu awal.

- 14) Simpan konfigurasi *router* dengan perintah ‘*write*’. Dan lakukan disemua *router*.



Gambar 6. 15 Menyimpan konfigurasi *router*.

Berikut adalah langkah-langkah konfigurasi *dynamic routing* menggunakan protokol OSPF.

- 1) Pada konfigurasi terminal ketikan perintah ‘*router OSPF [process ID]*’. Jangka nilai *process ID* mulai dari 1 sampai 65535.
- 2) Asumsikan semua perangkat jaringan terletak pada *area 0*. Pada konfigurasi *router OSPF* masukan *network* yang terhubung langsung dengan *router* yang sedang dikonfigurasi dengan perintah ‘*network [network ID] [wildcard mask] area [id area]*’. Nilai *wildcard mask* dengan mudah didapatkan dengan cara mengurangi nilai *masking prefix* /32 dengan nilai *masking* yang digunakan pada *network* tersebut. Contoh *masking* yang digunakan pada 10.0.0.0 adalah 255.255.255.128, maka nilai *wildcard mask*-nya adalah

255.255.255.255

255.255.255.128

0.0.0.127

The screenshot shows a Cisco IOS CLI window titled 'R1'. The tab bar at the top has 'Physical', 'Config', 'CLI' (which is selected), and 'Attributes'. Below the tabs is the text 'IOS Command Line Interface'. A message 'Press RETURN to get started.' is displayed. The main area contains configuration commands:

```
Router>en
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#network 10.0.0.128 0.0.0.31 area 0
Router(config-router)#network 10.0.0.176 0.0.0.3 area 0
Router(config-router)#{
```

At the bottom of the window, there are 'Copy' and 'Paste' buttons, and a checkbox labeled 'Top'.

Gambar 6. 16 Konfigurasi OSPF.

- 3) Lakukan konfigurasi tersebut ke semua router.
- 4) Setelah selesai lakukan pengujian koneksi dengan *ping*.
- 5) Setelah semua benar dan terhubung, simpan konfigurasi router.

Modul 7 UDP Header Analysis dan ICMP Analysis

Tujuan Praktikum

1. Mampu merekam info konfigurasi IP pada sebuah PC.
2. Mampu menggunakan Wireshark untuk mengambil DNS *Queries* dan *Responses*.
3. Mampu menganalisa DNS yang diambil atau UDP *packets*.

7.1 Pendahuluan

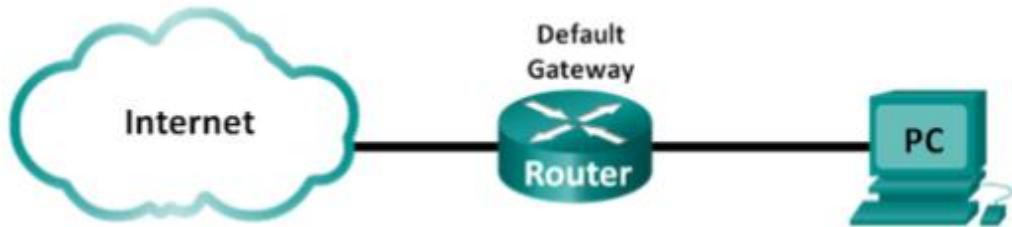
Jika kalian pernah menggunakan internet, kalian pasti telah menggunakan *Domain Name System* (DNS). DNS merupakan jaringan *network server* terdistribusi yang menerjemahkan nama *user-friendly domain* seperti www.google.com menjadi sebuah alamat IP. Ketika kalian mengetikan sebuah URL website kedalam *browser*, PV akan menjalankan DNS *Query* ke alamat IP *DNS server*. *DNS server query* PC kalian dan respon *DNS server* menggunakan *User Datagram Protocol* (UDP) pada *layer protocol transport*. UDP berbasis *connectionless* dan tidak membutuhkan pengaturan *session* seperti TCP. DNS *queries* dan *responses* cukup kecil dan tidak membutuhkan *overhead* dari TCP.

Di praktikum ini, kalian akan berhubungan dengan *DNS server* dengan mengirim DNS *queries* menggunakan UDP *transport protocol*. Kalian akan menggunakan Wireshark untuk memantau pertukaran DNS *query* dan *response* dengan server.

Note: Pada praktikum ini tidak dapat diselesaikan dengan Netlab dan berasumsi memiliki akses Internet.

7.2 Skenario

7.2.1 Topologi



Gambar 7. 1 Topologi yang digunakan.

7.2.2 Resource yang Dibutuhkan

1 PC (*Windows XP, Vista, atau XP* dengan *access command prompt*, *internet* dan *Wireshark*).

7.2.3 Merekam info konfigurasi IP pada sebuah PC (Part 1)

Pada *Part 1*, kalian akan menggunakan *command ipconfig/all* pada PC untuk mencari dan menentukan alamat MAC dan IP dari *network interface card* (NIC) PC kalian, *default gateway*, dan *DNS server IP address* PC. Simpan informasi ini pada *table* berikut. Info tersebut akan digunakan pada *part* berikutnya.

IP address	
MAC address	
Default gateway IP address	
DNS server IP address	

7.2.4 Menggunakan Wireshark untuk mengambil DNS Queries dan Responses I (Part 2)

Pada part 2, Kalian akan mengatur Wireshark untuk mengambil DNS *Queries* dan *Responses packets* untuk mendemonstrasikan penggunaan USP *transport protocol* sambai berhubungan dengan DNS server.

- 1) Klik Windows **Start button** dan cari Wireshark
- 2) **Note:** Jika Wireshark belum terpasang, dapat diunduh di <http://www.wireshark.org/download.html>.
- 3) Pilih sebuah *interface* untuk Wireshark untuk pengambilan *packets*. Gunakan **interface list** untuk memilih *interface* yang berhubungan dengan perekaman IP dan *Media Access Control (MAC)* address PC pada part 1.
- 4) Setelah memilih *interface* yang diinginkan, klik **Start** untuk melaukan pengambilan *packets*.
- 5) Buka *web browser* dan ketik **www.google.com**. Tekan *Enter* untuk lanjut.

Klik **Stop** untuk menghentikan penangkapan Wireshark ketika *homepage google* muncul.

7.2.5 Menganalisa DNS yang diambil atau UDP packets (Part 3)

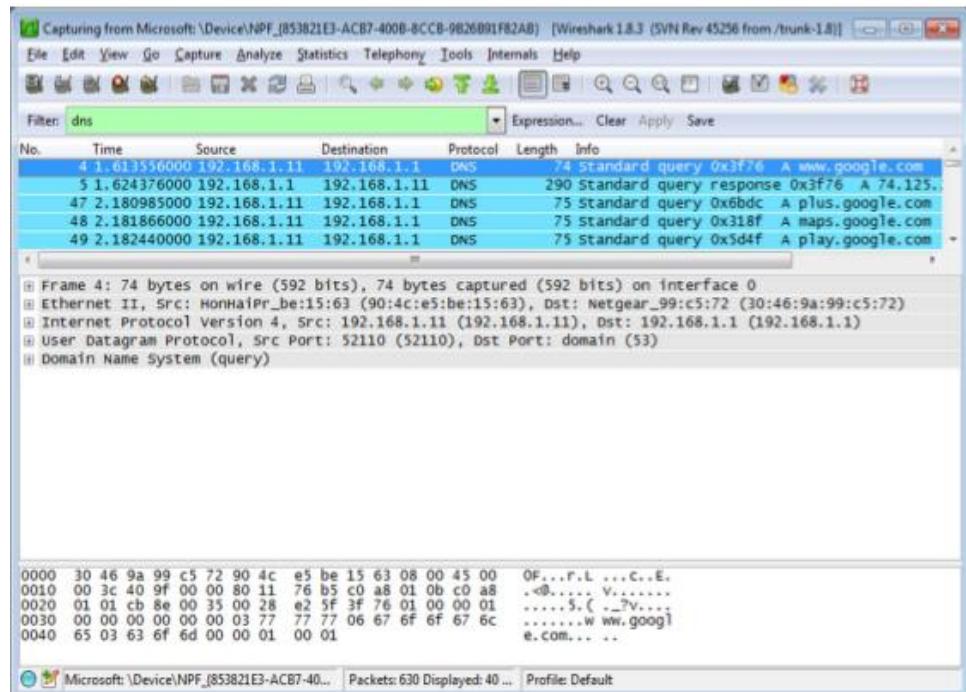
Pada part 3, Kalian akan memeriksa paket-paket UDP yang dihasilkan ketika berkomunikasi dengan server DNS untuk alamat IP untuk www.google.com.

Step 1: Filter DNS packets



- 1) Di jendela utama Wireshark, ketik dns di daerah masuknya **Filter toolbar**. Klik **Apply** atau tekan *Enter*.

Note: Jika Kalian tidak melihat hasil apapun setelah *filter* DNS diterapkan, tutup *browser web* dan di jendela *command prompt*, ketik ipconfig/flushdns untuk menghapus semua hasil DNS sebelumnya. *Restart capture* Wireshark dan ulangi petunjuk di Part 2b-2e. Jika ini tidak menyelesaikan masalah, di jendela *command prompt*, Kalian dapat mengetik nslookup www.google.com sebagai alternatif untuk *web browser*.

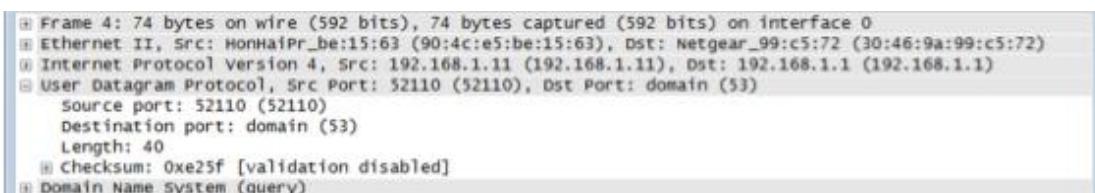


Gambar 7. 2 Tampilan Wireshark saat mem-filter protocol DNS.

- 2) Di panel daftar paket (bagian atas) dari jendela utama, cari paket yang mencakup "stktalianrd query" dan "A www.google.com". Lihat framei 4 sebagai contoh.

Step 2: Periksa segmen UDP menggunakan kueri DNS.

Periksa UDP dengan menggunakan *query DNS* untuk *www.google.com* seperti yang ditangkap oleh Wireshark. Dalam contoh ini, Wireshark menangkap *frame 4* di panel daftar paket yang dipilih untuk analisis. Protokol dalam *query* ini akan ditampilkan di panel rincian paket (bagian tengah) dari jendela utama. Entri protokol yang disorot dalam abu-abu.



Gambar 7. 3 Rincian packet pada Wireshark.

- 1) Di panel rincian paket, *frame 4* memiliki 74byte data pada kabel seperti yang ditampilkan pada baris pertama. Ini adalah jumlah byte untuk mengirim *query DNS* ke *server* nama yang meminta alamat IP dari *www.google.com*.
- 2) *Line Ethernet II* menampilkan sumber dan tujuan alamat MAC. Sumber alamat MAC adalah dari PC lokal kalian karena PC lokal kalian berasal *query DNS*. Tujuan alamat MAC adalah dari *default gateway*, karena ini adalah perhentian terakhir sebelum *query* ini keluar dari jaringan lokal.

Apakah sumber alamat MAC yang sama seperti direkam dari Part 1 untuk PC lokal?

- 3) Dalam *Internet Protocol Version 4* baris, paket IP Wireshark *capture* menunjukkan bahwa sumber alamat IP dari permintaan DNS ini 192.168.1.11 dan alamat IP tujuan adalah

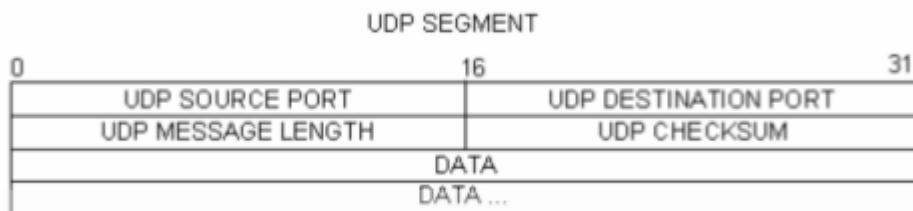
192.168.1.1. Dalam contoh ini, alamat tujuan adalah *default gateway*. *router* adalah *default gateway* dalam jaringan ini.

Dapatkan kalian memasangkan alamat IP dan MAC untuk sumber dan tujuan perangkat?

Device	IP Address	MAC address
PC local		
Default Gateway		

IP paket dan *header* menggabungkan segmen UDP. segmen UDP berisi permintaan DNS sebagai data.

- Sebuah *header* UDP hanya memiliki empat bidang: *port sumber*, *port tujuan*, panjang, dan *checksum*. Setiap bidang dalam *header* UDP hanya 16bit seperti yang digambarkan di bawah ini.



 Gambar 7. 4 UDP Header.

Perluas *User Datagram Protocol* di *panel* rincian paket dengan mengklik tombol *plus* (+). Perhatikan bahwa hanya ada empat bidang. Nomor *port* sumber dalam contoh ini adalah 52110. *Port* sumber dihasilkan secara acak oleh PC menggunakan nomor *port* lokal yang tidak dilindungi. *Port* tujuan adalah 53. *Port* 53 adalah *port* terkenal dicadangkan untuk digunakan dengan DNS. *server DNS* mendengarkan pada *port* 53 untuk permintaan DNS dari klien.

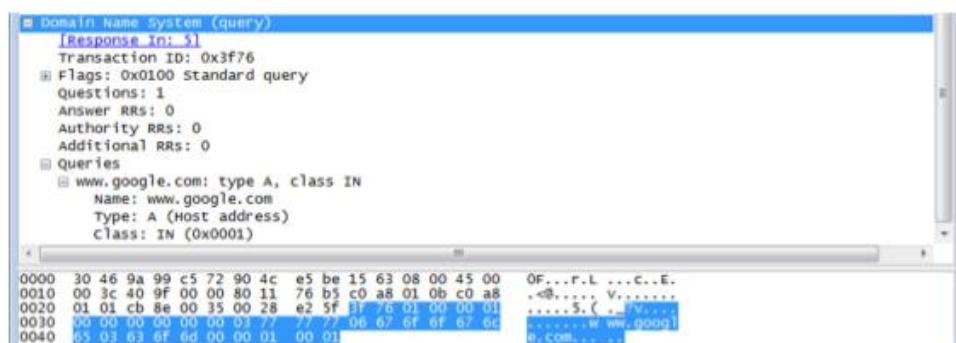
```

User Datagram Protocol, Src Port: 52110 (52110), Dst Port: domain (53)
  Source port: 52110 (52110)
  Destination port: domain (53)
  Length: 40
Checksum: 0xe25f [validation disabled]
  [Good Checksum: False]
  [Bad Checksum: False]

```

Gambar 7. 5 Rincian User Datagram Protocol.

Dalam contoh ini, panjang segmen UDP ini adalah 40 byte. Dari 40 byte, 8 byte digunakan sebagai *header*. 32 byte lainnya yang digunakan oleh *query* data DNS. 32 byte data permintaan DNS disorot pada ilustrasi berikut di *byte* paket *panel* (bagian bawah) dari jendela utama Wireshark.



Gambar 7. 6 Packet data permintaan DNS (gambar diblok).

checksum digunakan untuk menentukan integritas paket setelah itu dilalui *Internet*.

Header UDP memiliki *overhead* yang rendah karena UDP tidak memiliki bidang yang berkaitan dengan *three-way handshake* di TCP. Setiap *transfer* data masalah yang terjadi harus ditangani oleh lapisan aplikasi.

Frame Size	
Source MAC address	
Destination MAC address	
Source IP address	
Destination IP address	
Source Port	
Destination Port	

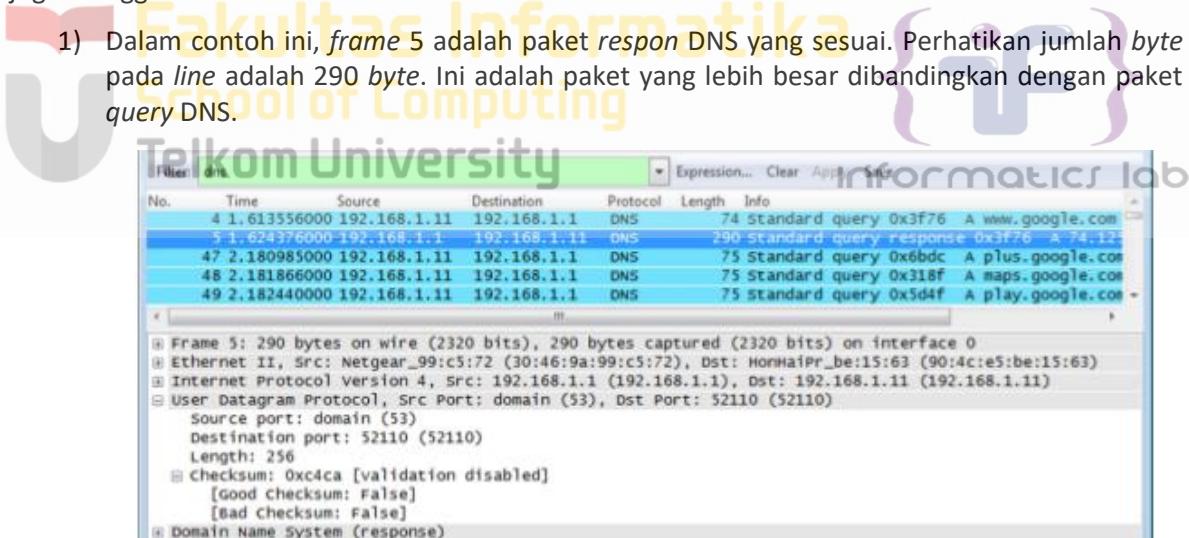
Apakah IP sumber alamat sama dengan alamat IP PC lokal yang tercatat di *Part 1*?

Apakah IP tujuan alamat yang sama sebagai *default gateway* dicatat dalam *Part 1*?

Step 3: Memeriksa UDP menggunakan *respon DNS*.

Pada step ini, Kalian akan memeriksa paket *respon DNS* dan memverifikasi bahwa paket *respon DNS* juga menggunakan UDP.

- 1) Dalam contoh ini, *frame 5* adalah paket *respon DNS* yang sesuai. Perhatikan jumlah byte pada *line* adalah 290 byte. Ini adalah paket yang lebih besar dibandingkan dengan paket *query DNS*.



Gambar 7. 7 Rincian *packets* pada Wireshark.

- 2) Dalam *frame Ethernet II* untuk *respon DNS*, dari perangkat apa sumber alamat MAC dan apa perangkat tujuan alamat MAC?
- 3) Perhatikan sumber dan alamat IP tujuan dalam paket IP. Apa tujuan alamat IP? Apa sumber alamat IP?

Tujuan alamat IP: _____

Source alamat IP: _____

Apa yang terjadi dengan peran sumber dan tujuan untuk *localhost* dan *default gateway*?

- 4) Di segmen UDP, peran nomor *port* juga terbalik. Nomor *port* tujuan adalah 52110. Nomor *port* 52110 adalah *port* yang sama yang dihasilkan oleh PC lokal ketika *query DNS* dikirim ke *server DNS*. PC lokal Anda mendengarkan *DNS reponse* pada *port* ini.

Nomor port sumber adalah 53. DNS server mendengarkan DNS query pada port 53 dan kemudian mengirimkan DNS response dengan nomor port sumber 53 kembali ke asal query DNS.

Ketika respon DNS diperluas, perhatikan alamat IP yang diselesaikan untuk www.google.com di bagian **Answers**.

█ User Datagram Protocol, Src Port: domain (53), Dst Port: 52110 (52110)
 Source port: domain (53)
 Destination port: 52110 (52110)
 Length: 256
█ checksum: 0xc4ca [validation disabled]
 [Good Checksum: False]
 [Bad Checksum: False]
█ Domain Name System (Response)
 [Request In: 4]
 [Time: 0.010820000 seconds]
 Transaction ID: 0x3f76
█ Flags: 0x8180 Standard query response, No error
 Questions: 1
 Answer RRs: 5
 Authority RRs: 4
 Additional RRs: 4
█ Queries
█ Answers
 ① www.google.com: type A, class IN, addr 74.125.227.84
 ② www.google.com: type A, class IN, addr 74.125.227.80
 ③ www.google.com: type A, class IN, addr 74.125.227.81
 ④ www.google.com: type A, class IN, addr 74.125.227.82
 ⑤ www.google.com: type A, class IN, addr 74.125.227.83
█ Authoritative nameservers
 ⑥ google.com: type NS, class IN, ns ns1.google.com
 ⑦ google.com: type NS, class IN, ns ns2.google.com
 ⑧ google.com: type NS, class IN, ns ns3.google.com
 ⑨ google.com: type NS, class IN, ns ns4.google.com
█ Additional records
 ⑩ ns1.google.com: type A, class IN, addr 216.239.32.10
 ⑪ ns2.google.com: type A, class IN, addr 216.239.34.10
 ⑫ ns3.google.com: type A, class IN, addr 216.239.36.10
 ⑬ ns4.google.com: type A, class IN, addr 216.239.38.10

Gambar 7. 8 Rincian *packets* DNS.

7.3 Kesimpulan

Apa Keuntungan menggunakan UDP *disbanding* TCP sebagai *transport protocol* untuk DNS?

Modul 8 Three-Way Handshaking

Tujuan Praktikum

1. Memahami teori *Three-Way Handshaking*.
2. Mampu melihat terjadinya *Three-Way Handshake* menggunakan *Packet Tracer*.

8.1 Pendahuluan

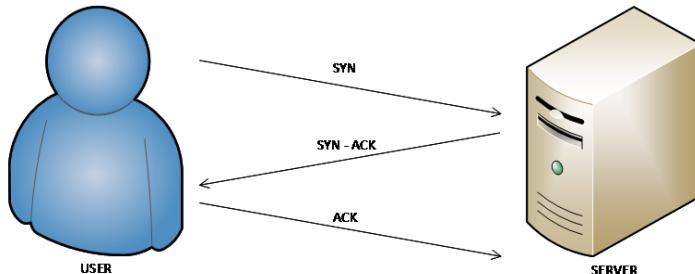
TCP *Three-way handshake* pada *Transmission Control Protocol* merupakan metode yang digunakan oleh TCP untuk membangun TCP/IP *connection* diatas jaringan berbasis IP. TCP *Three-way handshake* ini juga dikenal sebagai "SYN, SYN-ACK, ACK" karena terdapat tiga *message* yang ditransmisikan oleh TCP untuk menegosiasikan dan memulai sebuah TCP *session* antara dua buah mesin (komputer). Mekanisme TCP *handshaking* ini di desain agar dua buah komputer yang melakukan percobaan pembangunan koneksi dapat menegosiasikan parameter koneksi *network TCP socket* sebelum mentransmisikan data seperti HTTP web *browser request* dan lain-lain.

Pada modul ini diperlihatkan contoh *capture* sebuah HTTP *request* pada *web server*, dimana *client web browser* me-*request* sebuah halaman web dan dibalas oleh *server* dengan halaman yang di-*request* oleh *user*. Analisis *capture* dari *request* yang dilakukan *user* hingga terbangunnya koneksi antara *device user* dan *server tujuan* dilakukan menggunakan Wireshark *packet analyzer*.

TCP memanfaatkan beberapa jenis *flag* atau 1 *field bit Boolean* pada *header*-nya untuk mengontrol status koneksi. Tiga jenis *flag* yang digunakan antara lain:

- 1) **SYN** - (*Synchronize*) untuk menginisiasi koneksi
- 2) **FIN** - (*Final*) untuk terminasi koneksi
- 3) **ACK** - (*Acknowledges*) sebagai tanda diterimanya data

Dalam proses pembangunan koneksi melalui *Threeway handshake*, hanya 2 jenis *flag* saja yang digunakan yakni SYN dan ACK. Dimana prosesnya diperlihatkan pada diagram di bawah ini:



Gambar 8. 1 Ilustrasi *Three-Way Handshake*.

Pertama-tama *user* melakukan *request* dengan mengirimkan *message* SYN kepada *server*.

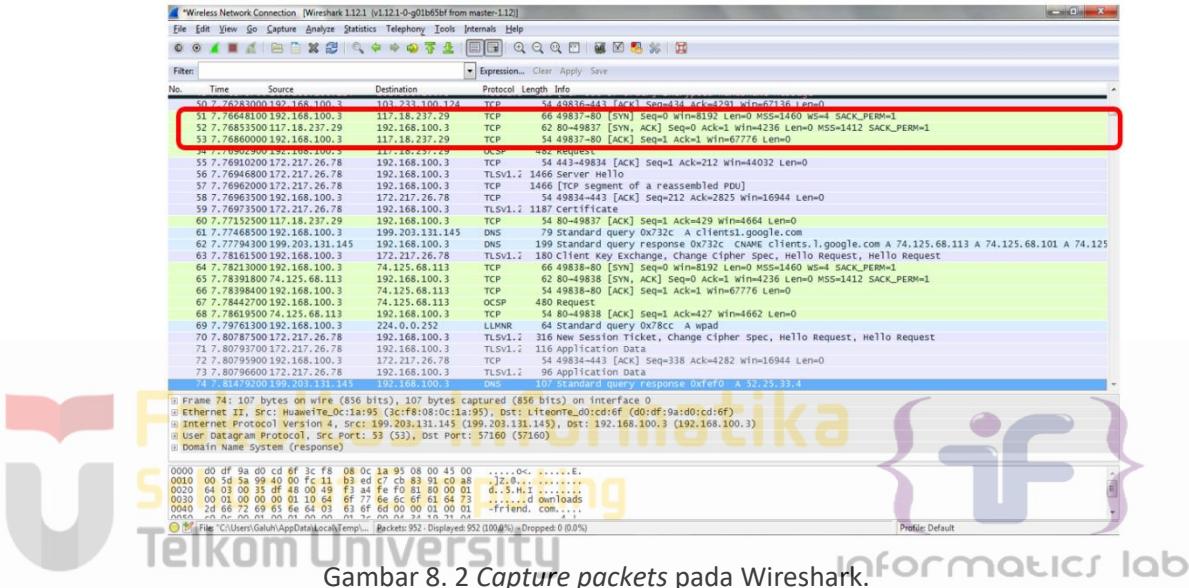
Setelah itu, *server* merespon *message* SYN tersebut dengan *message* SYN-ACK yang menandakan dibangunnya koneksi antara *user* dan *server*.

Terakhir, setelah terbangun koneksi maka *user* akan mengirimkan *message* ACK yang menandakan telah terbangunnya koneksi antara *user* dan *server* dan siap melakukan pertukaran data.

8.2 Skenario

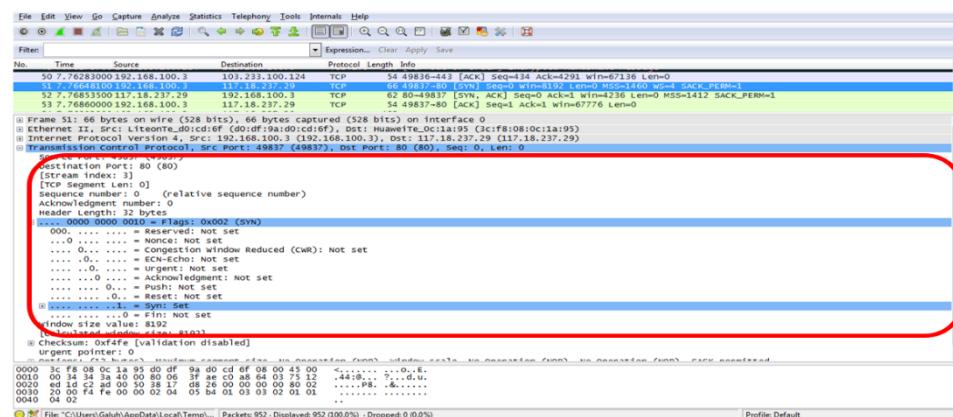
Sampai sini beberapa langkah yang harus dilakukan diantaranya :

- 1) Buka internet browser (IE, Mozilla, Safari, Google Chrome dll).
- 2) Akses halaman web yang diinginkan.
- 3) Perhatikan hasil capture *Three-way Handshake* pada Wireshark ketika melakukan akses halaman web. Untuk melihat status *flag* pada Wireshark, expand *TCP layer analysis* pada *middle pane*, kemudian expand kembali field ‘Flags’ pada *TCP Header*. Disana dapat dilihat seluruh detail dari *TCP Flag* yang digunakan.



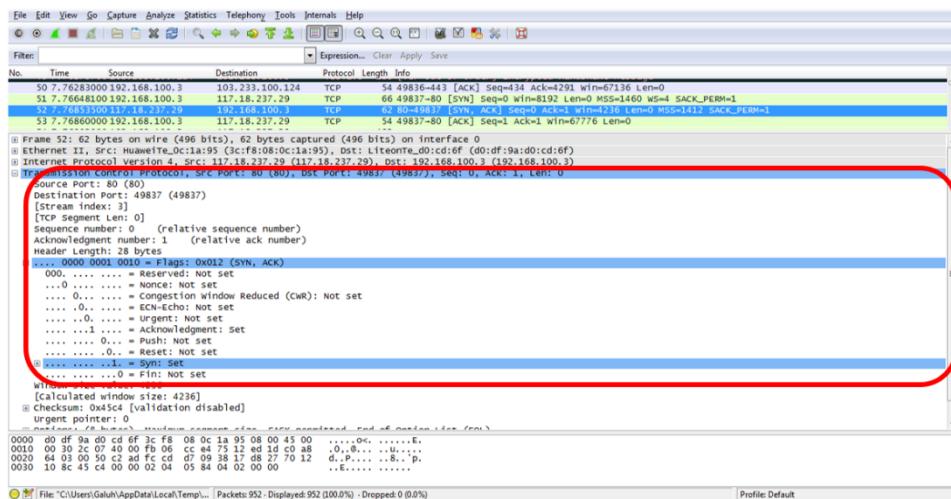
Gambar 8. 2 Capture packets pada Wireshark.

Pada percobaan yang dilakukan, *user* dengan alamat IP 192.168.100.3 mengakses halaman *web* pada IP 117.18.237.29.



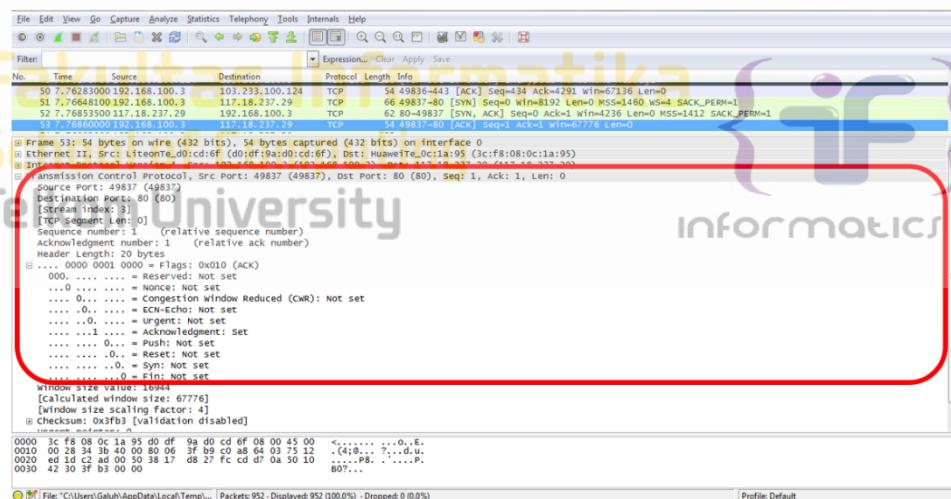
Gambar 8. 3 Request SYN.

Gambar diatas menunjukkan *request user* terhadap halaman *web* yang diinginkan. *Request* ini ditandakan dengan *message SYN* yang dikirim *user* ke *server* dengan alamat 117.16237.29. Perhatikan bahwa *flag SYN* pada *header* di set dengan angka 1.



Gambar 8. 4 Balasan SYN-ACK.

Selanjutnya, setelah *server* menerima *message SYN* dari *user*, *server* akan mengirimkan *message SYN – ACK* yang menandakan dibangunnya koneksi antara *user* dan *server*. Perhatikan bahwa *flag SYN* dan *ACK* pada *header* di *set* dengan angka 1.



Gambar 8. 5 Balasan ACK.

Terakhir, *user* akan mengirimkan kembali *message ACK* kepada *server* yang menandakan telah terbangunnya koneksi antara *user* dan *server* untuk selanjutnya *user* dan *server* dapat memulai proses pertukaran data. Perhatikan bahwa *flag ACK* pada *header* di *set* dengan angka 1.

Modul 9 Congestion Control

Tujuan Praktikum

1. Mempelajari mininet sebagai *emulator* jaringan SDN.
2. Memahami penggunaan *iPerf*.
3. Mempelajari *congestion control cubic, reno* dan *DCTCP*.

9.1 Mininet

Mininet adalah sebuah *emulator* jaringan SDN berbasis CLI dengan cara menjalankan kumpulan *virtual host, switch, router*, dan *link* pada satu kernel Linux. Tiap perangkat *virtual* yang dijalankan oleh Mininet berperilaku seperti mesin sungguhan, yang berarti pengguna dapat mengkonfigurasi tiap perangkat *virtual* tersebut layaknya *physical device*. Pengguna juga dapat menentukan kecepatan, *bandwidth*, serta *delay*. Beberapa alasan mengapa Mininet digunakan sebagai *emulator*, yaitu:

- 1) Memungkinkan pengujian topologi kompleks, tanpa perlu memasang jaringan fisik.
- 2) Ketersediaan CLI yang digunakan untuk *debugging* atau menjalankan uji coba jaringan secara luas.
- 3) Mendukung kustomisasi topologi dan pengaturan parameter-parameter jaringan.
- 4) Dapat digunakan dengan minim pemrograman.
- 5) Ketersediaan API dalam bahasa Python untuk membuat eksperimen pada jaringan.

Adapun langkah-langkah Instalasi Mininet yaitu:

Buka Terminal (*shortcut: ctrl+alt+T*) lalu unduh Mininet dari Github.

```
$ git clone git://github.com/mininet/mininet
```

Masuk ke *root* dengan perintah:

```
$ sudo su
```

Lakukan instalasi Mininet yang telah diunduh.

```
# mininet/util/install.sh -a
```

Setelah proses instalasi selesai, ketikkan perintah berikut untuk menjalankan Mininet.

```
# Mn
```

Secara *default*, Mininet akan membuat topologi sederhana dengan dua buah *host* dan sebuah *switch* serta secara otomatis menjalankan *controller POX*.

Sedangkan berikut ini merupakan *command* dasar Mininet:

- 1) Mengetahui koneksi antar *device*:

```
mininet> Net
```

```
mininet> dump
```

- 2) Mengetahui informasi *interface* dari *host h1*:

```
mininet> h1 ifconfig
```

- 3) Melakukan *ping* dari *host h1* ke *host h2*:

```
mininet> h1 ping h2
```

- 4) Melakukan *ping* dari dan ke seluruh *host*:

```
mininet> pingall
```

- 5) Keluar dari Mininet *environment*:

```
mininet> Exit
```

- 6) Membersihkan sisa konfigurasi Mininet:

```
# mn -c
```

- 7) Membuat *custom topology* dengan 3 *host* dan 1 *switch*:

```
# mn --topo single,3
```

- 8) Membuat *custom topology* linear (terdapat 4 *switch* yang masing-masing terhubung ke sebuah *host*, dan seluruh *switch* tersambung dalam satu baris):

```
# mn --topo linear,4
```

- 9) Mengatur parameter QoS untuk *bandwidth* dan *delay*:

```
# mn --link tc,bw=10,delay=10ms
```

- 10) Mengetahui alamat MAC *host h1*:

```
mininet> py h1.IP()
```

- 11) Menonaktifkan sambungan antara *switch s1* dan *host h1*:

```
mininet> link s1 h1 down
```

- 12) Menjalankan *custom topology* (hanya jika sudah terdapat topologi sebelumnya):

```
# mn --custom mininet/custom/topologoe.py --topo mytopo
```

- 13) Menjalankan Mininet dengan *controller* berbeda:

```
# mn --controller remote
```

9.1.1 iPerf

iPerf merupakan sebuah *tool* untuk mengukur performansi jaringan. *iPerf* sendiri bisa digunakan untuk mengukur performansi *link* dari sisi TCP maupun UDP. Di ubuntu bisa menggunakan perintah apt-get install iperf, di FreeBSD bisa menggunakan perintah pkg_add iperf. Untuk ujicoba pastikan bahwa komputer tujuan ter-install dengan baik *iPerf* dan *client test*-nya.

Penggunaan *iPerf* sebagai *server*

```
# iperf -s
```

Penggunaan *iPerf* sebagai *Client*

```
# iperf -c [ip_iperf_server]
```

Penggunaan *iPerf* sebagai *server UDP*

```
# iperf -s -u
```

Penggunaan *iPerf* sebagai *client* UDP

```
# iperf -c [ip_iperf_server] -u
```

9.2 Congestion Control Cubic, Reno dan DCTCP

Congestion Control di Linux ada beberapa jenis beberapa diantaranya adalah *Cubic*, *Reno*, *DCTCP* dan lain-lain.

Berikut perintah – perintah yang digunakan saat mengatur *Congestion Control*:

Mengetahui modul yang ter-install

```
# ls -la /lib/modules/$(uname -r)/kernel/net/ipv4
```

Mengetahui *Congestion Control* yang tersedia

```
# sysctl net.ipv4.tcp_available_congestion_control
```

Mengatur *Congestion Control*

```
# echo "nama_modul" > /proc/sys/net/ipv4/tcp_congestion
```

atau

```
# sysctl net.ipv4.tcp_congestion_control=nama_modul
```

Berikut tahapan untuk mengatur *Congestion Control*:

Menggunakan *Congestion Control Reno*

```
# echo reno > /proc/sys/net/ipv4/tcp_congestion_control
```

Menggunakan *Congestion Control Cubic*

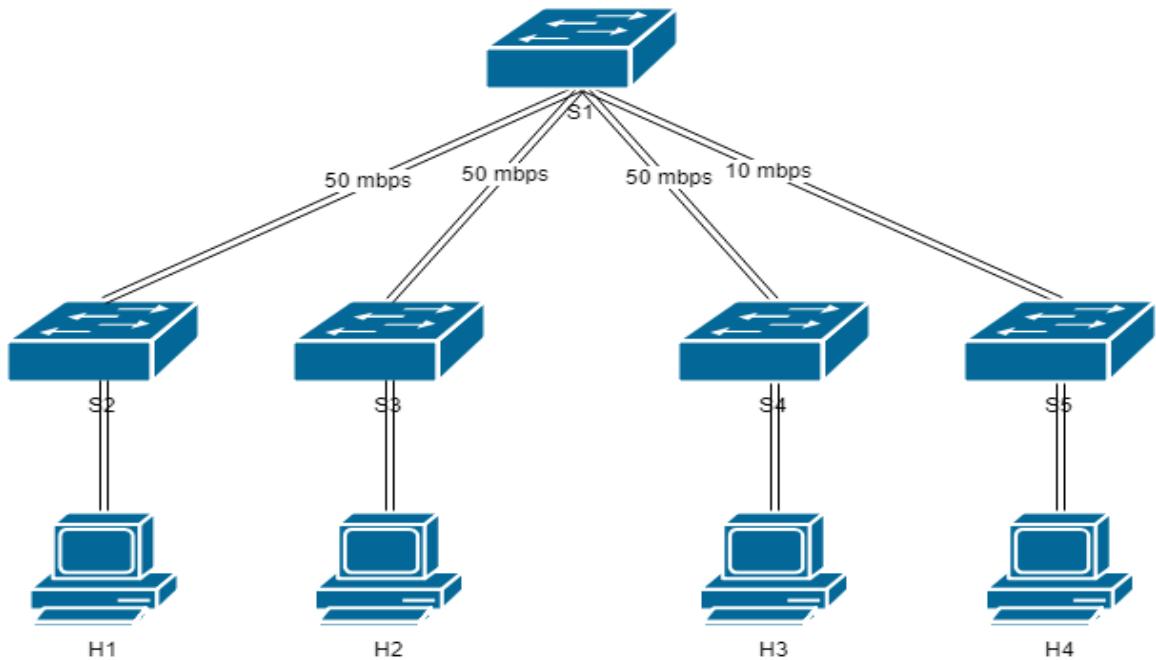
```
# echo cubic > /proc/sys/net/ipv4/tcp_congestion_control
```

Menggunakan *Congestion Control DCTCP*

```
# echo dctcp > /proc/sys/net/ipv4/tcp_congestion_control
```

9.3 Skenario Pengerjaan

Dalam skenario pengerjaan ini akan dilakukan menggunakan sistem operasi Ubuntu 18.04 64-bit, *congestion control* Reno dan Vegas, dan pengujian jaringan menggunakan iPerf. Berikut ini adalah topologi yang akan digunakan dalam skenario ini.



Gambar 9. 1 Topologi Skenario Mininet.

Dari gambar 8.1, spesifikasi jaringan sebagai berikut:

- 1) Bandwidth link dari S2, S3, dan S4 menuju S1 adalah 50 mbps.
 - 2) Bandwidth link dari S5 menuju S1 adalah 10 mbps.
 - 3) Kondisi pengujian dari H1 menuju H4 dengan H2 dan H3 aktif berkomunikasi dengan H4.

Hal ini membuat lalu lintas dari H1, H2, dan H3 menuju H4 terhambat atau kondisi ini lebih dikenal dengan istilah *bottleneck*. Gunakan *source code* berikut untuk membangun topologi seperti pada gambar 9.1.

Source Code:

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node
from mininet.log import setLogLevel, info
from mininet.cli import CLI
from mininet.link import TCLink
from mininet.node import CPULimitedHost
import time
import os

class MyTopo ( Topo ):
    def __init__( self, **opts ):
        Topo.__init__( self, **opts )

        # Membuat objek host
        H1 = self.addHost( 'H1', ip='10.0.0.1/24' )
        H2 = self.addHost( 'H2', ip='10.0.0.2/24' )
        H3 = self.addHost( 'H3', ip='10.0.0.3/24' )
        H4 = self.addHost( 'H4', ip='10.0.0.4/24' )
```

```

# Membuat objek switch
S1 = self.addSwitch( 'S1' )
S2 = self.addSwitch( 'S2' )
S3 = self.addSwitch( 'S3' )
S4 = self.addSwitch( 'S4' )
S5 = self.addSwitch( 'S5' )

# Membuat link antar objek
self.addLink( H1, S2 )
self.addLink( H2, S3 )
self.addLink( H3, S4 )
self.addLink( H4, S5 )
self.addLink( S1, S2, bw=50 )
self.addLink( S1, S3, bw=50 )
self.addLink( S1, S4, bw=50 )
self.addLink( S1, S5, bw=10 )

def runTopo():
    # Memastikan mininet bersih dari cache sebelumnya
    os.system('mn -c')

    # Membangun Topologi
    topo = MyTopo()
    net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink)
    net.start()

    # Konfigurasi congestion control
    print net['H1'].cmd('sysctl -w net.ipv4.tcp_congestion_control=[reno/vegas]')
    print net['H2'].cmd('sysctl -w net.ipv4.tcp_congestion_control=[reno/vegas]')
    print net['H3'].cmd('sysctl -w net.ipv4.tcp_congestion_control=[reno/vegas]')
    print net['H4'].cmd('sysctl -w net.ipv4.tcp_congestion_control=[reno/vegas]')

    # Memasukan objek host pada variabel
    h1, h2, h3, h4 = net.get ('H1', 'H2', 'H3', 'H4')

    # Menjalankan iPerf dibackground process
    h4.cmd('iperf -s&')
    h2.cmd('iperf -c 10.0.0.4&')
    h3.cmd('iperf -c 10.0.0.4&')
    h1.cmd('iperf -c 10.0.0.4 -i 1&')
    time.sleep(15)

    # Menampilkan hasil iPerf dari H1 ke H4
    h1.cmdPrint('fg')
    CLI(net)
    net.stop()

if __name__== '__main__':
    setLogLevel('info')
    runTopo()

```



Hasil dari *source code* diatas dapat dilihat pada gambar 9.2, hasil setiap kali dijalankan bisa berbeda-beda.

```
===== H1 Congestion Control =====
net.ipv4.tcp_congestion_control = reno

===== H2 Congestion Control =====
net.ipv4.tcp_congestion_control = reno

===== H3 Congestion Control =====
net.ipv4.tcp_congestion_control = reno

===== H4 Congestion Control =====
net.ipv4.tcp_congestion_control = reno

* * * * *
*** H1 : ('fg',)

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
[ 3] local 10.0.0.1 port 50190 connected with 10.0.0.4 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0- 1.0 sec 1.22 MBytes 10.2 Mbits/sec
[ 3] 1.0- 2.0 sec 191 KBytes 1.56 Mbits/sec
[ 3] 2.0- 3.0 sec 1.37 MBytes 11.5 Mbits/sec
[ 3] 3.0- 4.0 sec 0.00 Bytes 0.00 bits/sec
[ 3] 4.0- 5.0 sec 2.86 MBytes 24.0 Mbits/sec
[ 3] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[ 3] 6.0- 7.0 sec 0.00 Bytes 0.00 bits/sec
[ 3] 7.0- 8.0 sec 0.00 Bytes 0.00 bits/sec
[ 3] 8.0- 9.0 sec 382 KBytes 3.13 Mbits/sec
[ 3] 9.0-10.0 sec 382 KBytes 3.13 Mbits/sec
[ 3] 0.0-10.2 sec 6.38 MBytes 5.23 Mbits/sec
bash: fg: no job control
*** Starting CLI:
mininet> █

===== H1 Congestion Control =====
net.ipv4.tcp_congestion_control = vegas

===== H2 Congestion Control =====
net.ipv4.tcp_congestion_control = vegas

===== H3 Congestion Control =====
net.ipv4.tcp_congestion_control = vegas

===== H4 Congestion Control =====
net.ipv4.tcp_congestion_control = vegas

* * * * *
*** H1 : ('fg',)

Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
[ 3] local 10.0.0.1 port 50340 connected with 10.0.0.4 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0- 1.0 sec 448 KBytes 3.67 Mbytes/sec
[ 3] 1.0- 2.0 sec 191 KBytes 1.56 Mbytes/sec
[ 3] 2.0- 3.0 sec 63.6 KBytes 521 Kbytes/sec
[ 3] 3.0- 4.0 sec 127 KBytes 1.04 Mbytes/sec
[ 3] 4.0- 5.0 sec 384 KBytes 3.15 Mbytes/sec
[ 3] 5.0- 6.0 sec 384 KBytes 3.15 Mbytes/sec
[ 3] 6.0- 7.0 sec 512 KBytes 4.19 Mbytes/sec
[ 3] 7.0- 8.0 sec 384 KBytes 3.15 Mbytes/sec
[ 3] 8.0- 9.0 sec 512 KBytes 4.19 Mbytes/sec
[ 3] 9.0-10.0 sec 384 KBytes 3.15 Mbytes/sec
[ 3] 0.0-10.2 sec 3.31 MBytes 2.74 Mbytes/sec
bash: fg: no job control
*** Starting CLI:
mininet> █
```

Gambar 9.2 Congestion Control Reno dan Vegas.



Modul 10 Queue

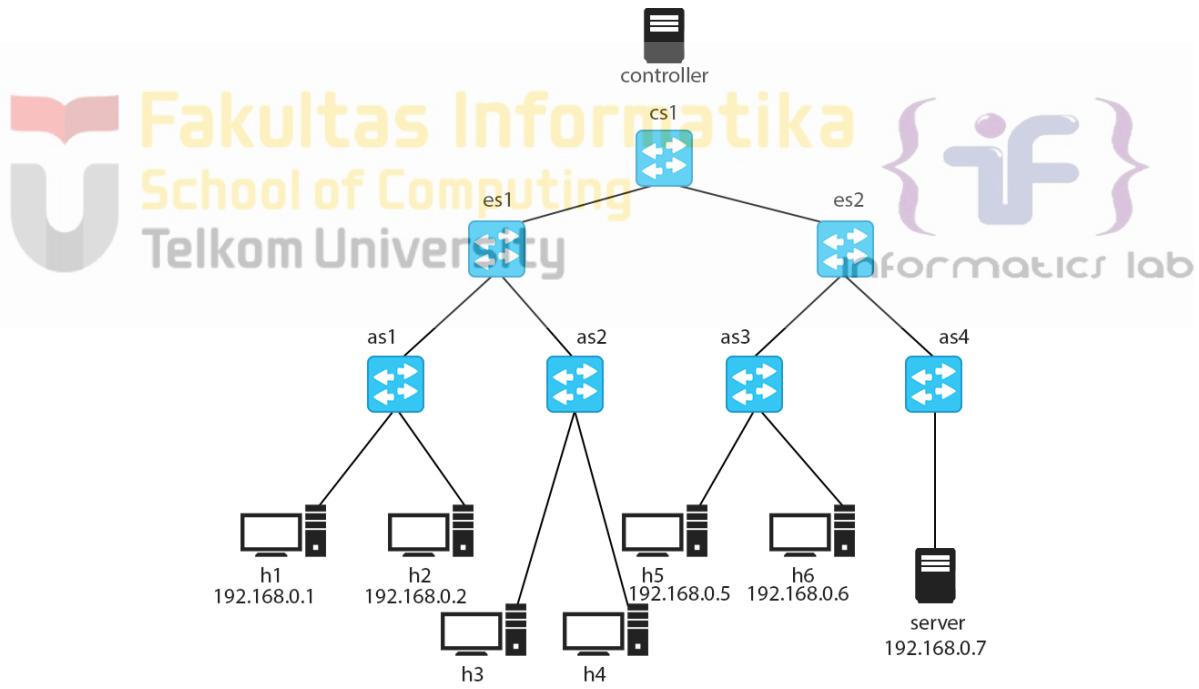
Tujuan Praktikum

1. Memahami *Network Scheduler*.
2. Memahami jenis-jenis *Queue*.
3. Mengerti perintah-perintah CBQ.

10.1 Pendahuluan

Network Scheduler atau juga disebut *packet scheduler*, disiplin antrian, *qdisc* atau algoritma *queue* adalah sebuah pengatur pada *node* dalam jaringan komunikasi *packet switching*. *Network Scheduler* mengatur *sequence* dari paket-paket jaringan dalam antrian dikirim dan diterima dari *interface* jaringan. Logika dari *Network Scheduler* memutuskan untuk memilih paket yang mana untuk diteruskan terlebih dahulu. Sistem mungkin memiliki satu atau lebih antrian yang mungkin menyimpan paket dalam satu alur, klasifikasi, atau prioritas.

10.2 Membuat *Custom Topology* di Mininet



Gambar 10. 1 Topologi.

Topologi tersebut pada dasarnya merupakan topologi *tree* dengan sebuah *core switch* (cs), dua *edge switch* (es), dan empat *aggregator switch* (as). Tiga *aggregator switch* masing-masing memiliki dua *host* serta satu lainnya terhubung ke *server HTTP*. Keseluruhan jaringan tersebut berada di satu *subnet* yang sama dengan alamat IP yaitu 192.168.0.0/24.

Seluruh *node* tersebut berjalan secara *virtual* di dalam Mininet. Dengan menggunakan bahasa pemrograman Python, pengguna dapat membuat topologinya sendiri dengan cara membuat sebuah *file* dan disimpan di direktori *mininet/custom*.

10.2.1 Membuat Source Code

Berikut ini dijabarkan langkah demi langkah cara membuat *custom topology* di Mininet.

- 1) Buka Terminal dan pindah ke direktori *custom*.

```
$ cd mininet/custom
```

- 2) Masuk ke *root* dengan perintah:

```
$ sudo su
```

- 3) Buat sebuah *file* dengan *text editor* Gedit.

```
# gedit topologiku.py
```

- 4) Lalu isi dengan *source code* berikut ini.

```
from mininet.topo import Topo
class MyTopo(Topo):
def __init__(self):
    Topo.__init__(self)
        # Add nodes (hosts and switches)
    h1 = self.addHost( 'h1' , ip='192.168.0.1' )
    h2 = self.addHost( 'h2' , ip='192.168.0.2' )
    h3 = self.addHost( 'h3' , ip='192.168.0.3' )
    h4 = self.addHost( 'h4' , ip='192.168.0.4' )
    h5 = self.addHost( 'h5' , ip='192.168.0.5' )
    h6 = self.addHost( 'h6' , ip='192.168.0.6' )
    server = self.addHost( 'server' , ip='192.168.0.7' )
    cs1 = self.addSwitch('cs1')
    es1 = self.addSwitch( 'es1' )
    es2 = self.addSwitch( 'es2' )
    as1 = self.addSwitch( 'as1' )
    as2 = self.addSwitch( 'as2' )
    as3 = self.addSwitch( 'as3' )
    as4 = self.addSwitch( 'as4' )
    # Add links
    self.addLink(cs1, es1)
    self.addLink(cs1, es2)
    self.addLink(es1, as1)
    self.addLink(es1, as2)
    self.addLink(es2, as3)
    self.addLink(es2, as4)
    self.addLink(as1, h1)
    self.addLink(as1, h2)
    self.addLink(as2, h3)
    self.addLink(as2, h4)
    self.addLink(as3, h5)
    self.addLink(as3, h6)
    self.addLink( as4, server )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

- 5) Simpan dan tutup Gedit

10.2.2 Menjalankan Custom Topology

Berikut ini merupakan langkah-langkah bagaimana cara menjalankan *custom topology*.

- 1) Buka Terminal dan pindah ke direktori *custom*.

```
$ cd mininet/custom
```

- 2) Masuk ke *root* dengan perintah:

```
$ sudo su
```

- 3) Jalankan *controller*. Jika Ryu yang dipilih sebagai *controller*, maka *command*-nya adalah:

```
# ryu-manager ryu/ryu/app/rest_firewall.py
```

- 4) Sedangkan untuk *Floodlight* adalah:

```
# java -jar target/floodlight.jar
```

- 5) Jalankan Mininet dengan beberapa parameter tambahan:

```
# mn --custom topologiku.py --topo mytopo --controller remote
```

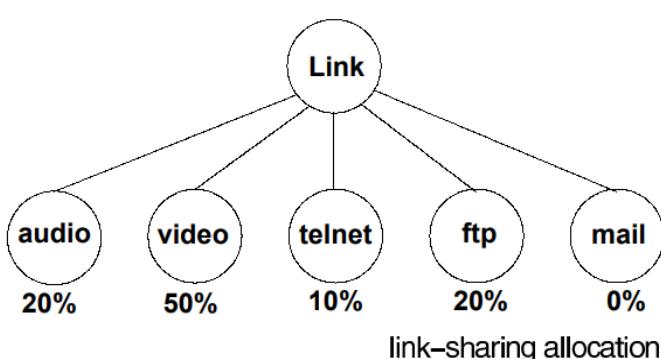
10.3 Jenis-jenis Queue

Terdapat banyak jenis *queue* yang dapat digunakan pada jaringan, namun dalam praktikum ini akan membahas hanya dua jenis *queue* yaitu CBQ (*Class Based Queue*) dan HTB (*Hierachial Token Bucket*).

10.3.1 CBQ (Class Based Queue)

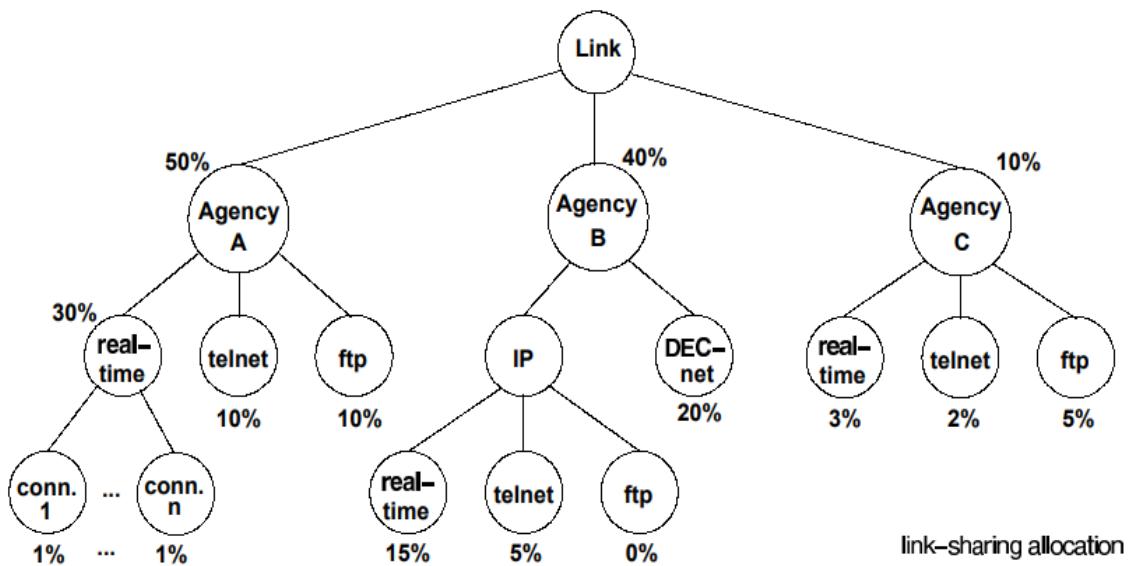


CBQ atau *Class Based Queue* merupakan disiplin antrian untuk *network scheduler* yang memungkinkan *traffic* untuk membagi *bandwidth* secara merata, setelah dibagi ke beberapa kelas. Kelas bisa berdasarkan prioritas, *interfaces*, atau aplikasi. Karena CBQ beroperasi pada *network Layer*, CBQ menyediakan keuntungan yang sama melalui teknologi *layer 2* apapun dan efektif dengan semua IP Protokol. Untuk ilustrasi pembagian berdasarkan aplikasi, perhatikan gambar 9.2.



Gambar 10. 2 CBQ *Link Sharing Allocation*.

CBQ juga memungkinkan membuat kelas secara hirarkis multilevel. Perhatikan gambar 10.3.



Gambar 10. 3 CBQ *Hierarchical Link-Sharing Structure*.

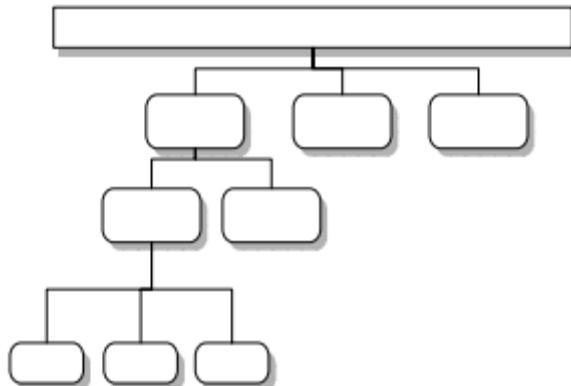
Tujuan dari *link-sharing* adalah:

- 1) Setiap interior atau kelas daun harus menerima *bandwidth link-sharing* yang telah dialokasikan selama *interval* waktu berdasarkan permintaan.
- 2) Jika semua interior atau kelas daun telah menerima permintaan *bandwidth link-sharing*, pembagian *bandwidth* berlebih tidak bisa dibagikan sewenang-wenang, tapi harus mengikuti pedoman logika.

Oleh karena itu dalam konfigurasi CBQ sendiri terdapat dua *mode* pengaturan pembagian *bandwidth* untuk kelas-kelasnya, yaitu *isolated* dan *borrow*. Yang dimaksud dengan *isolated* adalah setiap *bandwidth* yang telah dialokasikan ke masing-masing kelas tidak akan bisa berubah. Oleh karena itu, kelas yang diberikan prioritas yang paling kecil masih bisa menggunakan jaringan dengan optimal. Sedangkan *borrow*, *bandwidth* yang telah dialokasikan ke masing-masing kelas dapat 'dipinjam' oleh kelas lainnya apabila pada kelas tersebut tidak ada aktivitas atau prioritas kelas terlalu rendah. Hal ini menyebabkan *bandwidth* yang didapatkan oleh kelas prioritas tertinggi sangat besar, sedangkan kelas prioritas kecil tidak dapat beraktivitas dengan optimal.

10.3.2 HTB (*Hierachial Token Bucket*)

HTB atau *Hierarchical Token Bucket* merupakan pengganti antrian yang lebih bagus dari CBQ di Linux ini berguna untuk membatasi kecepatan unduh atau unggah klien sehingga klien terbatas tidak dapat memenuhi total *bandwidth*. HTB mengimplementasikan mekanisme *queuing classful* untuk sistem kontrol lalu lintas linux, dan menyediakan *rate* dan *ceil* untuk memungkinkan pengguna untuk mengontrol *bandwidth absolut* ke kelas tertentu dari lalu lintas serta menunjukkan rasio distribusi *bandwidth* ketika *bandwidth* ekstra menjadi tersedia. Struktur hirarki dari HTB bisa dilihat pada gambar 10.4.



Gambar 10. 4 Struktur HTB.

Secara singkat struktur HTB menyerupai struktur CBQ multilevel hirarki. Karena HTB ini menjadi *alternative* pilihan yang lebih baik daripada CBQ. Namun keduanya memiliki kelebihan dan kekurangan masing-masing. Penggunaan HTB dan CBQ ini disesuaikan dengan kebutuhan jaringan.

10.4 Contoh Perintah CBQ

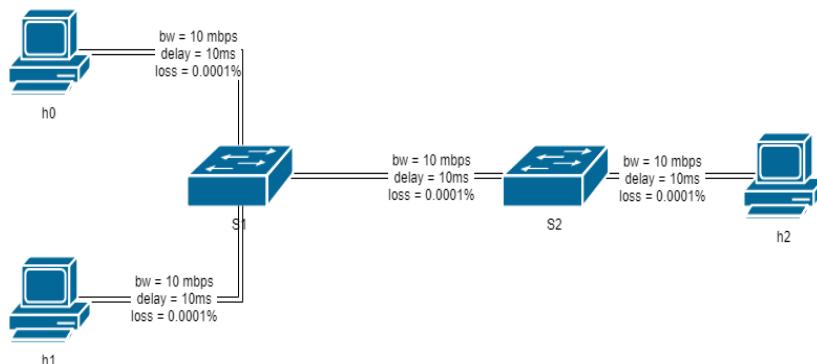
```
# tc qdisc add dev r1-eth1 root handle 1: cbq avpkt 1000 bandwidth 60mbit
# tc class add dev r1-eth1 parent 1: classid 1:1 cbq rate 2mbit allot 1500
# prio 5 bounded isolated
# tc filter add dev r1-eth1 protocol ip parent 1: prio 16 u32 match ip src
# 0/0 flowid 1:
```

10.5 Contoh Skenario CBQ dan HTB pada Mininet

Dalam skenario ini akan dilakukan pada sistem operasi Ubuntu 18.04 64-bit.

10.5.1 Skenario CBQ

Topologi yang akan digunakan pada skenario ini dapat dilihat pada gambar .



Gambar 10. 5 Topologi Skenario CBQ.

Berikut adalah *source code python* untuk skenario CBQ.

```
#!/usr/bin/python
from mininet.net import Mininet
from mininet.node import Node
from mininet.link import TCLink
from mininet.log import setLogLevel, info
from threading import Timer
from mininet.util import quietRun
from time import sleep
from mininet.cli import CLI

def myNet(cname='controller', cargs='-v ptcp:'):
    "Create network from scratch using Open vSwitch."
    info( "*** Creating nodes\n" )
    controller = Node( 'c0', inNamespace=False )
    s0 = Node( 's0', inNamespace=False )
    s1 = Node( 's1', inNamespace=False )
    h0 = Node( 'h0' )
    h1 = Node( 'h1' )
    h2 = Node( 'h2' )

    info( "*** Creating links\n" )
    linkopts0=dict(bw=10, delay='1ms', loss=0.0001)
    TCLink( h0, s0, **linkopts0)
    TCLink( h1, s0, **linkopts0)
    TCLink( s0, s1, **linkopts0)
    TCLink( s1, h2, **linkopts0)

    info( "*** Configuring hosts\n" )
    h0.setIP( '192.168.123.1/24' )
    h1.setIP( '192.168.123.2/24' )
    h2.setIP( '192.168.123.3/24' )

    info( "*** Starting network using Open vSwitch\n" )
    s0.cmd( 'ovs-vsctl del-br dp0' )
    s0.cmd( 'ovs-vsctl add-br dp0' )
    s1.cmd( 'ovs-vsctl del-br dp1' )
    s1.cmd( 'ovs-vsctl add-br dp1' )
    controller.cmd( cname + ' ' + cargs + '&' )
    for intf in s0.intfs.values():
        print intf
        print s0.cmd( 'ovs-vsctl add-port dp0 %s' % intf )

    for intf in s1.intfs.values():
        print intf
        print s1.cmd( 'ovs-vsctl add-port dp1 %s' % intf )
    # Note: controller and switch are in root namespace, and we
    # can connect via loopback interface
    s0.cmd( 'ovs-vsctl set-controller dp0 tcp:127.0.0.1:6633' )
    s1.cmd( 'ovs-vsctl set-controller dp0 tcp:127.0.0.1:6633' )
```

```

info( '*** Waiting for switch to connect to controller' )
while 'is_connected' not in quietRun( 'ovs-vsctl show' ):
    sleep( 1 )
    info( '.' )
info( '\n' )

#print s0.cmd('ovs-ofctl show dp0')
#info( "*** Running test\n" )
h0.cmdPrint( 'ping -c 3 ' + h2.IP() )
h1.cmdPrint( 'ping -c 3 ' + h2.IP() )
h2.cmd('iperf -s &')
print "iperf: h0--s0--s1--h2"
h0.cmdPrint('iperf -c 192.168.123.3 -t 10')
print "iperf: h1--s0--s1--h2"
h1.cmdPrint('iperf -c 192.168.123.3 -t 10')
print "limit the bandwidth for flow h0-h2"
s0.cmdPrint('ethtool -K s0-eth2 gro off')
s0.cmdPrint('tc qdisc del dev s0-eth2 root')
s0.cmdPrint('tc qdisc add dev s0-eth2 root handle 1: cbq avpkt 1000 bandwidth 10Mbit')
s0.cmdPrint('tc class add dev s0-eth2 parent 1: classid 1:1 cbq rate 512kbit allot 1500 prio 5
bounded isolated')
s0.cmdPrint('tc filter add dev s0-eth2 parent 1: protocol ip prio 16 u32 match ip src 192.168.123.1
flowid 1:1')
s0.cmdPrint('tc qdisc add dev s0-eth2 parent 1:1 sfq perturb 10')
h0.cmdPrint('iperf -c 192.168.123.3 -t 10')
print "iperf: h1--s0--s1--h2"
h1.cmdPrint('iperf -c 192.168.123.3 -t 10')

info( "**** Stopping network\n" )
controller.cmd( 'kill %' + cname )
s0.cmd( 'ovs-vsctl del-br dp0' )
s0.deleteIntfs()
s1.cmd( 'ovs-vsctl del-br dp1' )
s1.deleteIntfs()
info( '\n' )

if __name__ == '__main__':
    global net
    setLogLevel( 'info' )
    info( '*** Scratch network demo (kernel datapath)\n' )
    Mininet.init()
    myNet()

```



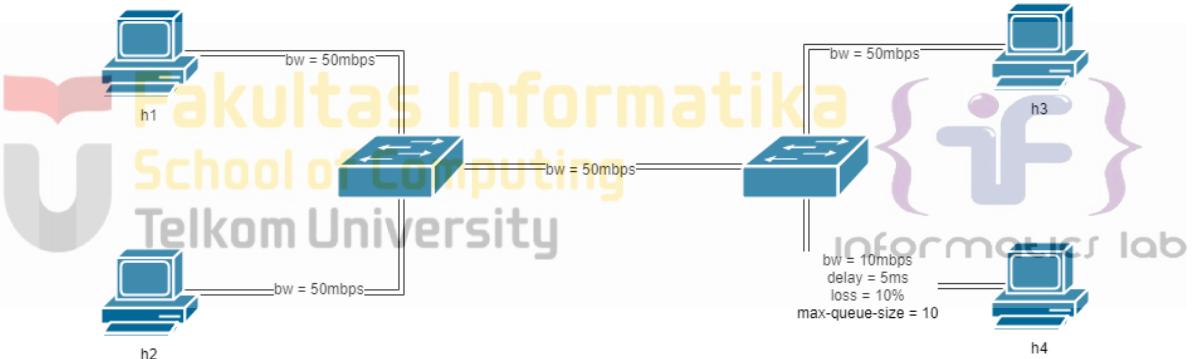
Hasil dari *source code* tersebut dapat dilihat pada gambar 10.6.

```
limit the bandwidth for flow h0-h2
*** s0 : ('ethtool -K s0-eth2 gro off',)
*** s0 : ('tc qdisc del dev s0-eth2 root',)
*** s0 : ('tc qdisc add dev s0-eth2 root handle 1: cbq avpkt 1000 bandwidth 10Mbit',)
*** s0 : ('tc class add dev s0-eth2 parent 1: classid 1:1 cbq rate 512kbit allot 1500 prio 5 bounded isolated',)
*** s0 : ('tc filter add dev s0-eth2 parent 1: protocol ip prio 16 u32 match ip src 192.168.123.1 flowid 1:1',)
*** s0 : ('tc qdisc add dev s0-eth2 parent 1:1 sfq perturb 10',)
*** h0 : ('iperf -c 192.168.123.3 -t 10',)
-----
Client connecting to 192.168.123.3, TCP port 5001
TCP window size: 85.3 KByte (default)
[ 3] local 192.168.123.1 port 41942 connected with 192.168.123.3 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3]  0.0-10.9 sec   768 Kbytes   575 Kbytes/sec
iperf: h1-s0--s1-h2
*** h1 : ('iperf -c 192.168.123.3 -t 10',)
-----
Client connecting to 192.168.123.3, TCP port 5001
TCP window size: 85.3 KByte (default)
[ 3] local 192.168.123.2 port 33468 connected with 192.168.123.3 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3]  0.0-10.1 sec   11.5 MBytes   9.58 Mbites/sec
*** Stopping network
-----
root@ROG:/home/harude/mininet/custom#
```

Gambar 10. 6 Hasil Skenario CBQ.

10.5.2 Skenario HTB

Topologi yang digunakan pada skenario ini dapat dilihat pada gambar 10.7.



Gambar 10. 7 Topologi Skenario HTB.

Berikut adalah *source code* untuk skenario HTB pada gambar 10.7.

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.link import TCLink
from mininet.node import CPULimitedHost
from mininet.log import setLogLevel, info
from mininet.cli import CLI
import time
import os

class MyTopo( Topo ):

    def __init__( self, **opts ):

        # Initialize topology
        Topo.__init__( self, **opts )
```

```

# Add hosts and switches
h1 = self.addHost( 'h1' )
h2 = self.addHost( 'h2' )
h3 = self.addHost( 'h3' )
h4 = self.addHost( 'h4' )
s1 = self.addSwitch( 's1' )
s2 = self.addSwitch( 's2' )

# Add links
self.addLink( h1, s1, bw=50 )
self.addLink( h2, s2, bw=50 )
self.addLink( h3, s1, bw=50 )
self.addLink( h4, s2, bw=10, delay='5ms', loss=10, max_queue_size=10, use_htb=True )
self.addLink( s1, s2, bw=50)

def run():
    os.system('mn -c')
    topo = MyTopo()
    link = TCLink
    host = CPULimitedHost
    net = Mininet(topo=topo, link=link, host=host)
    net.start()
    h1, h2, h3, h4 = net.get('h1', 'h2', 'h3', 'h4')
    h4.cmd('iperf -s&')
    h1.cmdPrint('iperf -c 10.0.0.4 -i 1&')
    h2.cmdPrint('iperf -c 10.0.0.4 -i 1&')
    h3.cmdPrint('iperf -c 10.0.0.4 -i 1&')
    for i in range (11):
        info(i, ' ')
        time.sleep(1)

    info('\n')
    h1.cmdPrint('fg')
    h2.cmdPrint('fg')
    h3.cmdPrint('fg')
    net.pingAll()
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    run()

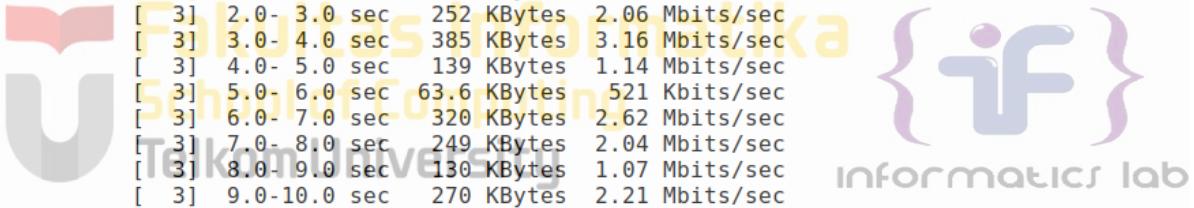
```



Hasil dari skenario dapat dilihat pada gambar 10.8.

```
-----  
Client connecting to 10.0.0.4, TCP port 5001  
TCP window size: 85.3 KByte (default)  
-----  
[ 3] local 10.0.0.2 port 41772 connected with 10.0.0.4 port 5001  
[ ID] Interval Transfer Bandwidth  
[ 3] 0.0- 1.0 sec 462 KBytes 3.79 Mbits/sec  
[ 3] 1.0- 2.0 sec 192 KBytes 1.57 Mbits/sec  
[ 3] 2.0- 3.0 sec 322 KBytes 2.64 Mbits/sec  
[ 3] 3.0- 4.0 sec 127 KBytes 1.04 Mbits/sec  
[ 3] 4.0- 5.0 sec 255 KBytes 2.09 Mbits/sec  
[ 3] 5.0- 6.0 sec 255 KBytes 2.09 Mbits/sec  
[ 3] 6.0- 7.0 sec 254 KBytes 2.08 Mbits/sec  
[ 3] 7.0- 8.0 sec 127 KBytes 1.04 Mbits/sec  
[ 3] 8.0- 9.0 sec 192 KBytes 1.57 Mbits/sec  
[ 3] 9.0-10.0 sec 254 KBytes 2.08 Mbits/sec  
[ 3] 0.0-10.1 sec 2.38 MBytes 1.97 Mbits/sec  
bash: fg: no job control  
*** h3 : ('fg',)  
-----  
Client connecting to 10.0.0.4, TCP port 5001  
TCP window size: 85.3 KByte (default)  
-----  
[ 3] local 10.0.0.3 port 40446 connected with 10.0.0.4 port 5001  
[ ID] Interval Transfer Bandwidth  
[ 3] 0.0- 1.0 sec 326 KBytes 2.67 Mbits/sec  
[ 3] 1.0- 2.0 sec 279 KBytes 2.29 Mbits/sec  
[ 3] 2.0- 3.0 sec 252 KBytes 2.06 Mbits/sec  
[ 3] 3.0- 4.0 sec 385 KBytes 3.16 Mbits/sec  
[ 3] 4.0- 5.0 sec 139 KBytes 1.14 Mbits/sec  
[ 3] 5.0- 6.0 sec 63.6 KBytes 521 Kbits/sec  
[ 3] 6.0- 7.0 sec 320 KBytes 2.62 Mbits/sec  
[ 3] 7.0- 8.0 sec 249 KBytes 2.04 Mbits/sec  
[ 3] 8.0- 9.0 sec 130 KBytes 1.07 Mbits/sec  
[ 3] 9.0-10.0 sec 270 KBytes 2.21 Mbits/sec  
[ 3] 0.0-10.4 sec 2.36 MBytes 1.91 Mbits/sec
```

Gambar 10.8 Hasil Skenario HTB.



Modul 11 Advanced Packet Capture

Tujuan Praktikum

1. Memahami penggunaan TCPDump.
2. Mampu menggunakan TCPDump dalam *Advanced Packet Capture*.

11.1 Pendahuluan

TCPDump adalah sebuah *tools command line* atau terminal paket *sniffer* atau paket analisis yang biasa dipakai untuk menyaring paket tcp/ip yang diterima atau yang dikirimkan melalui jaringan pada *interface* tertentu. Tcpdump banyak digunakan untuk mencari masalah-masalah di jaringan atau untuk memonitor aktivitas jaringan, Tcpdump menggunakan API pcap (*packet capture*) yaitu LibPcap (Unix) atau winPcap (windows) untuk menangkap paket. Data hasil *capture* tcpdump sering disebut sebagai *dumpfile*. Tcpdump banyak digunakan untuk mencari masalah-masalah di jaringan atau untuk memonitor aktivitas jaringan, Tcpdump menggunakan API pcap (*packet capture*) yaitu LibPcap (Unix) atau winPcap (windows) untuk menangkap paket. Data hasil *capture* tcpdump sering disebut sebagai *dumpfile*.

Perbedaan utama Tcpdump dengan Wireshark adalah tcpdump tidak melakukan analisa terhadap data, namun hanya melakukan *copy* paket secara keseluruhan (*dump raw packet data*). Karena itu beban analisis terletak sepenuhnya pada *user*, namun demikian, kesalahan analisa yang disebabkan oleh *software* (semisal wireshark) dapat dihindari. Tcpdump dapat bekerja dengan baik bila dipasang pada jaringan yang menggunakan *hub*. Bila tcpdump diletakkan pada jaringan berbasis *switch*, maka tcpdump tersebut hanya dapat melihat lalu lintas antara *user* yang menggunakan tcpdump dan *switch* tersebut.

11.2 Instalasi TCPDump di Linux

Untuk sistem operasi berbasis Debian gunakan perintah

```
# apt-get install tcpdump
```

Untuk sistem operasi berbasis red hat gunakan perintah

```
# yum install tcpdump
```

11.3 Capture Menggunakan TCPDump

Buka terminal lalu masuk sebagai *root* dengan perintah

```
$ sudo su
```

Setelah masuk gunakan perintah berikut

```
# tcpdump -i [nama_interface]
```

Capture jumlah paket tertentu gunakan perintah

```
# tcpdump -c [jumlah_paket] -i [nama_interface]
```

Capture paket dalam bentus ASCII

```
# tcpdump -A -i [nama_interface]
```

Menampilkan *interface* tersedia gunakan perintah

```
# tcpdump -D
```

Menampilkan Paket dalam bentuk HEX dan ASCII

```
# tcpdump -XX -i [nama_interface]
```

Capture dan menyimpan dalam bentuk *file*

```
# tcpdump -w [nama_file].pcap -i [nama_interface]
```

Membaca *file* hasil Capture

```
# Tcpdump -r [nama-file]
```

Capture IP Address

```
# tcpdump -n -i [nama_interface]
```

Capture hanya Paket TCP

```
# tcpdump -i [nama_interface] tcp
```

Capture paket Dari Port tertentu

```
# tcpdump -i [nama_interface] port [nomor_port]
```

Capture paket dari IP sumber

```
# tcpdump -i [nama_interface] src [ip_address]
```

Capture paket ke IP tujuan

```
# tcpdump -i [nama_interface] dst [ip_address]
```



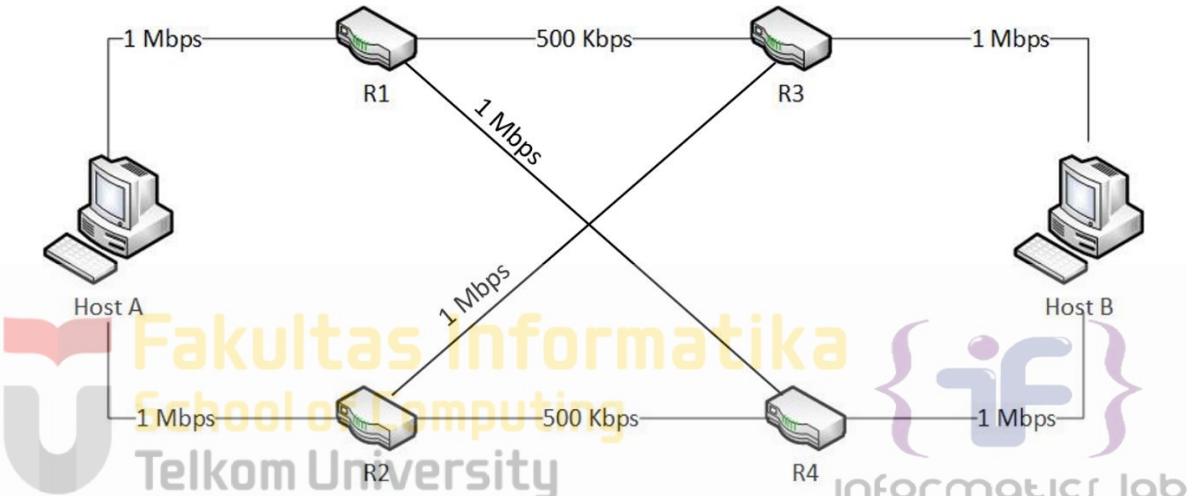
Modul 12 Spesifikasi Tugas Besar

Tujuan Praktikum

1. Mahasiswa mampu memahami persoalan yang diberikan.
2. Mahasiswa mampu mengimplementasikan topologi, routing, MPTCP dan mekanisme queueing dan fairness pada jaringan berbasis Mininet.

12.1 Simulasi MPTCP Pada Mininet

Topologi yang digunakan pada skenario ini dapat dilihat pada gambar 12.1.



Gambar 12. 1 Topologi MPTCP.

1) CLO 1.

Pada CLO ini terdapat spesifikasi penggerjaan dan kriteria penilaian yang akan dilakukan.

- Goal : Membandingkan QoS antara TCP dan MPTCP dari host A ke host B.
 - Generate *traffic* dari Host A ke Host B menggunakan iPerf.
 - Implementasi MPTCP level kernel pada Host A dan Host B.
(<https://multipath-tcp.org/pmwiki.php/Users/AptRepository>)
 - Metric QoS yang digunakan berupa *Throughput*, *Packet Loss* dan *Delay*.
 - Analisis hasil perbandingan metric QoS antara TCP dan MPTCP.
- Penilaian yang akan dilakukan adalah :
 - Kesesuaian topologi yang dibangun dengan soal yang diberikan, (30).
 - Uji konektivitas Host A → Host B (30).
 - Ketepatan analisis perbandingan performansi TCP dan MPTCP (40).
 - NILAI TOTAL = 100.

2) CLO 2.

Pada CLO ini terdapat spesifikasi penggerjaan dan kriteria penilaian yang akan dilakukan.

- Goal : Mengimplementasikan mekanisme Routing pada topologi yang ada.
 - Generate *traffic* and *background traffic* menggunakan iPerf.
 - Implementasi MPTCP level kernel pada Host A dan Host B.
(<https://multipath-tcp.org/pmwiki.php/Users/AptRepository>)
 - Melakukan pemutusan link R1 ke R4 untuk mensimulasikan *link failure* .
 - Implementasikan Routing pada topologi (RIP, OSPF).

- Metric yang digunakan berupa *Convergence Time*, *Delay*.
 - Analisis durasi *Convergence Time* dan *Delay* yang terjadi.
- Penilaian yang akan dilakukan adalah :
 - Ketepatan konfigurasi static routing pada MPTCP (25).
 - Ketepatan mekanisme Subnetting (25).
 - Uji konektivitas Host A → Host B (10).
 - Ketepatan analisis proses routing? (40).
 - NILAI TOTAL = 100.
- 3) CLO 3.
- Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.
- Goal : Membuktikan bahwa MPTCP telah di-implementasikan dengan benar pada topologi.
 - Generate *traffic* dan *background traffic* menggunakan iPerf.
 - Implementasi MPTCP level kernel pada Host A dan Host B.
(<https://multipath-tcp.org/pmwiki.php/Users/AptRepository>)
 - Capture trafik yang mengindikasikan MPTCP menggunakan custom script atau wireshark.
 - Analisis hasil *capture traffic*, buktikan bahwa MPTCP terimplementasi.
 - Penilaian yang akan dilakukan adalah :
 - Ketepatan konfigurasi, instalasi dan verifikasi MPTCP (30).
 - Uji konektivitas Host A → Host B (30).
 - Ketepatan analisis implementasi MPTCP (40).
 - NILAI TOTAL = 100.
- 4) CLO 4.
- Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.
- Goal : Menginspeksi penggunaan queue pada router jaringan.
 - Generate *traffic* dan *background traffic* menggunakan iPerf.
 - Implementasi MPTCP level kernel pada Host A dan Host B.
(<https://multipath-tcp.org/pmwiki.php/Users/AptRepository>)
 - Set ukuran buffer pada router : 20, 40, 60 dan 100.
 - Capture pengaruh ukuran buffer terhadap *delay*.
 - Analisis eksperimen hasil variasi ukuran buffer.
 - Penilaian yang akan dilakukan adalah :
 - Kemampuan memanipulasi parameter konfigurasi (Link./Queue/Delay) (30).
 - Kemampuan meng-capture queue length (30).
 - Ketepatan analisis pengaruh manipulasi pada parameter? (40).
 - NILAI TOTAL = 100.

12.2 Ketentuan Pengumpulan Tugas Besar

Pengumpulan pengerjaan tugas besar ini memiliki ketentuan penting, yaitu :

- 1) Membuat Laporan Tugas Besar yang di upload di LMS.
- 2) Merekam demo hasil pengerjaan tugas besar dengan durasi maksimal 15 menit, upload rekaman pada kanal youtube masing-masing. (Wajah wajib diperlihatkan ketika merekam dan menjelaskan demo).
- 3) Sertakan link rekaman youtube pada LMS.

Modul 13 Kemajuan Pengerjaan Tugas Besar

Tujuan Praktikum
1. Mahasiswa mampu memberikan kemajuan proses pengerjaan tugas besar yang diberikan.

13.1 Kegiatan Praktikum

Pada praktikum kali ini terdapat empat kegiatan utama, yaitu:

- 1) Praktikan memberikan kemajuan pengerjaan tugas besar kepada asisten.
- 2) Asisten membantu mengarahkan penyelesaian tugas besar.
- 3) Asisten memverifikasi pengerjaan tugas besar praktikan, berdasarkan poin-poin penilaian tugas besar yang telah diberikan.
- 4) Asisten mencatat kemajuan penyelesaian tugas besar mahasiswa.



Modul 14 Presentasi Tugas Besar

Tujuan Praktikum

- Presentasi tugas besar yang telah dikerjakan praktikan.

14.1 Kegiatan Praktikum

Pada praktikum kali ini asisten menilai penggerjaan mahasiswa, sesuai ketentuan penilaian berikut :

- 1) CLO 1.
 - Kesesuaian topologi yang dibangun dengan soal yang diberikan, (30).
 - Uji konektivitas Host A → Host B (30).
 - Ketepatan analisis perbandingan performansi TCP dan MPTCP (40).
 - NILAI TOTAL = 100.
- 2) CLO 2.
 - Ketepatan konfigurasi static routing pada MPTCP (25).
 - Ketepatan mekanisme Subnetting (25).
 - Uji konektivitas Host A → Host B (10).
 - Ketepatan analisis proses routing? (40).
 - NILAI TOTAL = 100.
- 3) CLO 3.
 - Ketepatan konfigurasi, instalasi dan verifikasi MPTCP (30).
 - Uji konektivitas Host A → Host B (30).
 - Ketepatan analisis implementasi MPTCP (40).
 - NILAI TOTAL = 100.
- 4) CLO 4.
 - Kemampuan memanipulasi parameter konfigurasi (Link,/Queue/Delay) (30).
 - Kemampuan meng-capture queue length (30).
 - Ketepatan analisis pengaruh manipulasi pada parameter? (40).
 - NILAI TOTAL = 100.

Daftar Pustaka

- [1] sharif, 31 July 2016. [Online]. Available: <http://ce.sharif.edu/courses/91-92/1/ce443-1/resources/root/Projects/OPNET%20Exercise%203.pdf>.
- [2] T. Svensson dan A. Popescu, 31 July 2016. [Online]. Available: http://www.pitt.edu/~dtipper/2120/MM1_Tutorial1.pdf.
- [3] Rahul, "SecureNet," 24 01 2017. [Online]. Available: <https://securenetweb.wordpress.com/2017/01/24/add-tcp-congestion-control-variant-to-linux-ubuntu-comparing/>. [Diakses 29 07 2018].
- [4] gigihfordanama, "Gigih Forda Nama," 14 02 2011. [Online]. Available: <http://staff.unila.ac.id/gigih/2011/02/14/iperf-tool-untuk-mengecek-performance-jaringan/>. [Accessed 30 07 2018].
- [5] I. Cho, "Introduction to Mininet," Mininet, 26 September 2018. [Online]. Available: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>. [Diakses 20 December 2018].





Kontak Kami :

 @fiflab

 Praktikum IF LAB

 informaticslab@telkomuniversity.ac.id

 informatics.labs.telkomuniversity.ac.id