DIKTAT KULIAH

IT507 - ROUTING & SWITCHING 1

Theophilus Wellem, M.S.

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN SATYA WACANA
SALATIGA
2011

DIKTAT KULIAH

IT507 - ROUTING & SWITCHING 1

Theophilus Wellem, M.S.

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS KRISTEN SATYA WACANA

SALATIGA

2011

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan berkat dan rahmatNya sehingga penulis dapat menyelesaikan diktat ini dengan baik. Penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang secara langsung atau tidak langsung membantu penulis dalam penyusunan diktat ini.

Diktat ini merupakan acuan tambahan dari buku-buku teks dan materi yang diperoleh dari Cisco Networking Academy untuk matakuliah Routing dan Switching 1 yang diampu oleh penulis. Penulis berharap diktat ini dapat menambah wawasan mahasiswa dalam mempelajari dasar dari protocol TCP/IP dan OSI Layer.

Salatiga, September 2011

Theophilus Wellem, M.S.

DAFTAR ISI

BAB I
PENGENALAN JARINGAN KOMPUTER

1.1 Pengertian

Jaringan Komputer merupakan kumpulan dari beberapa komputer yang saling terhubung oleh sebuah media dan dapat membagi data, printer, modem, dan *hardware resource* yang lain (*sharing resources*). Dalam area yang terbatas, jaringan ini disebut *Local Area Network* (LAN). Pada dasarnya ada 2 tipe dari *network* yaitu:

1. *Peer-to-peer Network*.
2. *Server-based Network*.

1.2 Komponen pada Sebuah Jaringan

Komponen-komponen yang berada pada sebuah jaringan terdiri dari:

1. *Server*.
2. *Client*.
3. *Media*.
4. *Shared data*.
5. *Shared printer and other peripherals*.

Server

Server merupakan komputer yang menyediakan *resources* yang dapat dibagi kepada pengguna jaringan. *Server* ini dapat

terdiri lebih dari satu komputer dangan tugas yang berbeda-beda, ada yang berfungsi sebagai *file server*, *mail server*, dan lain sebagainya.

Client

Client merupakan komputer-komputer yang mengakses *resources* dalam jaringan yang disediakan oleh *server*.

Media

Media menggambarkan bagaimana sebuah komputer dihubungkan dengan komputer yang lain dalam sebuah jaringan.

Shared Data

Shared data merupakan *file-file* yang disediakan oleh *server* dalam jaringan.

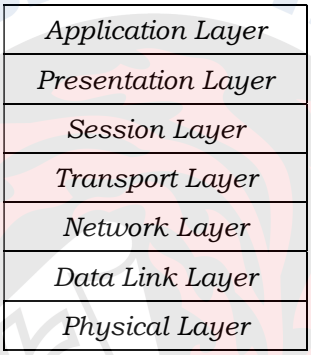
Shared Printers and Other Peripherals

Shared printers and other peripherals merupakan *resources* lain yang disediakan *server* untuk digunakan oleh *client*.

1.3 Model Komunikasi Data

Agar komunikasi data antar komputer dapat dilakukan maka dibutuhkan suatu aturan yang akan mengatur jalannya komukasi tersebut, aturan-aturan ini disebut dengan protokol. Model data komunikasi yang dikembangkan oleh *International Standards Organization* (ISO) sering digunakan untuk menggambarkan struktur dan fungsi dari protokol

komunikasi data. Arsitektur ini disebut *Open System Interconnection* (OSI) *Reference Model*. Model OSI ini terdiri dari 7 *layer* yang mendefinisikan fungsi dari protokol komunikasi data. Sebuah *layer* tidak mendefinisikan sebuah protokol. Sebuah *layer* dapat memiliki lebih dari satu protokol untuk menjalankan fungsi komunikasi data pada *layer* tersebut. Model 7 *layer* OSI ditunjukkan pada Gambar 1.1.



Gambar 1.1 7 Layer OSI

7 layer dari OSI dengan fungsinya secara singkat adalah sebagai berikut :

1. *Physical Layer*, mendefinisikan karakteristik fisik dari *network media*, juga mengirim dan menerima bit-bit.
2. *Data Link Layer*, menyediakan layanan pengantaran data melalui hubungan fisik (*physical link*).
3. *Network Layer*, mengatur hubungan antar *network* untuk *layer* di atasnya.
4. *Transport Layer*, menyediakan layanan *reliable*, mencakup pendeteksian *error* dan pengiriman ulang data bila terjadi *error*.

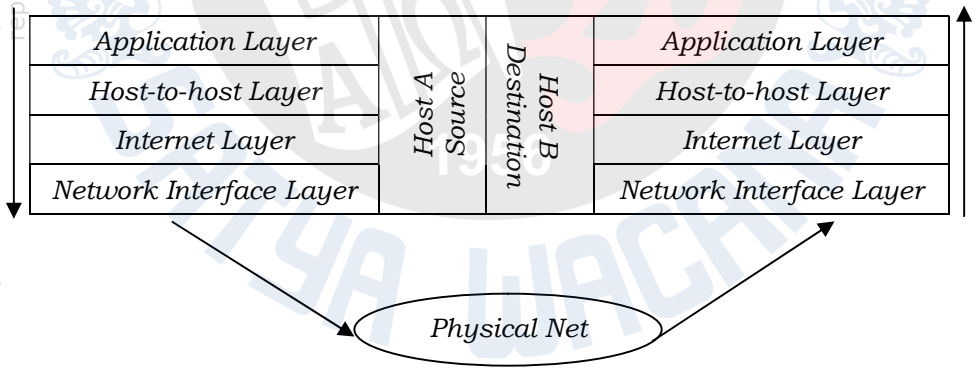
5. *Session Layer*, mengatur sesi antar aplikasi.
6. *Presentation Layer*, mengatur penyajian data kepada *application layer* (menerjemahkan data dari satu format kepada format yang lain).
7. *Application Layer*, menyediakan pelayanan aplikasi yang digunakan untuk berkomunikasi melalui *network*.

Model Internet tidak sepenuhnya menggunakan model OSI.

Arsitektur protokol TCP/IP dibagi menjadi 4 layer, yaitu:

1. *Network Access Layer*.
2. *Internet Layer*.
3. *Host-to-host Layer*.
4. *Application Layer*.

Proses komunikasi data dalam jaringan TCP/IP dapat dilihat pada Gambar 1.2.

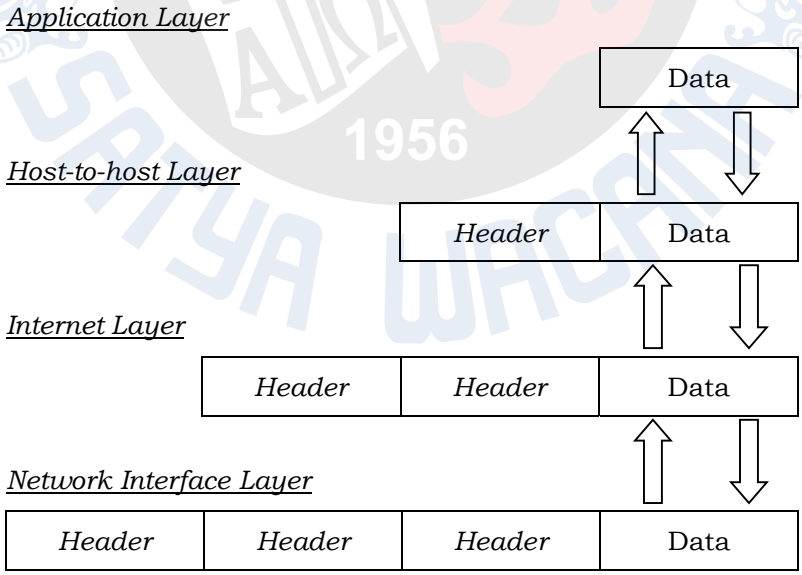


Gambar 1.2 Proses Komunikasi Data pada TCP/IP

Setiap protokol berkomunikasi dengan *peer*-nya. *Peer* adalah implementasi dari protokol yang sama pada *remote host*. Saat data akan dikirim dari *host* A (*source host*) ke *host* B

(*destination host*), data dari *Application layer* diberikan ke *Host-to host layer*, dari *Host-to-host layer* ke *Internet layer*, dari *Internet layer* ke *Network Interface layer*, baru kemudian dikirimkan melalui media fisik menuju *host* B (*destination host*). Pada *destination host* terjadi proses yang sebaliknya, dimana data dikirimkan dari *layer* terbawah hingga *layer* teratas. Saat pengiriman data dari lapisan teratas hingga lapisan terbawah terjadi proses yang disebut sebagai pengkapsulan data (*data encapsulation*), dimana setiap *layer* menambahkan informasi kontrol pada data, informasi ini disebut *header*. Saat penerimaan dan pengiriman data dari *layer* terbawah hingga *layer* teratas pada *host* B, setiap *layer* akan mengambil *header*-nya masing-masing.

Proses pengkapsulan data (*Data encapsulation*) dapat dilihat pada Gambar 1.3.



Gambar 1.3 Proses Pengkapsulan Data

Arah panah ke bawah menunjukkan proses yang terjadi saat pengiriman data dan arah sebaliknya menunjukkan proses yang terjadi saat penerimaan data.

Setiap *layer* menggunakan istilah yang berbeda untuk menunjukkan data yang sedang dikirim. Struktur data pada tiap *layer* adalah sebagai berikut.

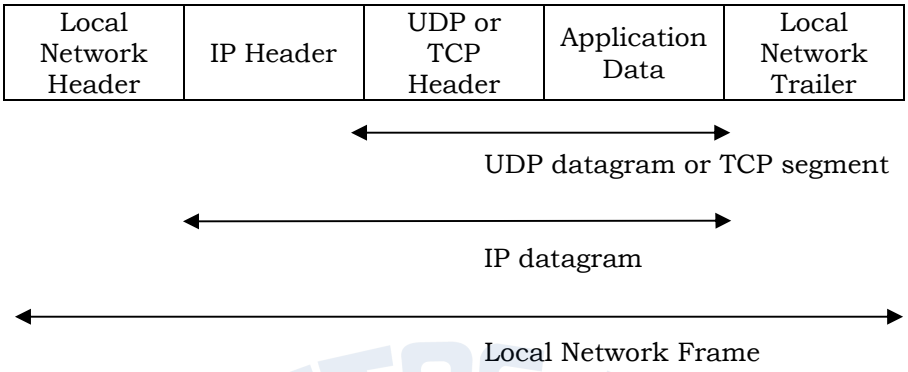
Application Layer
TCP : *Stream*
UDP : *Message*

Host-to-host Layer
TCP : *Segment*
UDP : *Packet*

Internet Layer
TCP : *Datagram*
UDP : *Datagram*

Network Interface Layer
TCP : *Frame*
UDP : *Frame*

Internet Transmission Frame dapat ditunjukkan pada Gambar 1.4.



Gambar 1.4 *Internet Transmission Frame*

Arsitektur dari TCP/IP akan dijelaskan lebih detail pada Bab II.

1.4 Jenis *Local Area Network*

Jenis dari *Local Area Network* dapat dilihat dari beberapa hal, yaitu:

1. Media Transmisi.
2. Metode Transmisi.
3. Topologi *Network*.
4. Metode Akses.

Media Transmisi

Jenis Media yang banyak digunakan dalam LAN adalah :

1. *Twisted Pair*.

   Media ini paling banyak digunakan. Ada 2 jenis kabel *twisted pair* yaitu *Shielded Twisted Pair* (STP) dan *Unshielded Twisted Pair* (UTP).
2. *Coaxial Cable*.

Kabel koaksial ini dihubungkan ke komputer pada jaringan melalui sebuah *transceiver* yang berfungsi untuk mengambil data yang ada pada kabel koaksial tersebut untuk diberikan kepada komputer.

3. *Fiber Optic*.

   Media ini menggunakan cahaya dalam mengantar data ke tujuan. Kelebihan dari media ini terletak pada kecepatan transfer datanya dan keamanan datanya (tidak mudah terganggu oleh *noise*).

Metode Transmisi

1. *Baseband*

   Pada metode ini data yang berupa sinyal digital langsung dikirim tanpa melalui modulasi.

2. *Broadband*

   Pada metode ini data digital harus diubah dulu menjadi sinyal analog (dimodulasi). Dengan memodulasi data digital menjadi sinyal analog, maka data dapat dikirimkan hingga tempat yang jauh.

Topologi Jaringan

Ada 3 jenis standar topologi jaringan, yaitu :

1. Star.
2. Bus.
3. Ring

Topologi Star

Pada topologi star, komputer-komputer dihubungkan dengan kabel ke sebuah sentral yang disebut *hub*. Sinyal

ditransmisikan dari sebuah komputer melalui *hub* menuju komputer yang lain pada *network*.

Topologi Bus

Data pada jaringan (dalam bentuk sinyal elektrik) dikirim ke seluruh komputer yang berada pada jaringan, tetapi data hanya diterima oleh komputer yang alamatnya cocok dengan alamat yang di-*encoding* pada data. Hanya satu komputer pada suatu waktu yang dapat mengirimkan data. Karena data atau sinyal elektrik dikirimkan ke seluruh jaringan, data akan berjalan dari ujung yang satu ke ujung yang lain. Jika sinyal diijinkan untuk terus berjalan tanpa dihentikan, maka sinyal akan terpantul kembali sepanjang kabel dan menghalangi komputer lain untuk mengirimkan data. Jadi sinyal ini harus dihentikan setelah sinyal tersebut mencapai tujuannya. Untuk menghentikan sinyal ini digunakan terminator yang dipasang pada setiap ujung kabel. Terminator ini akan menyerap sinyal yang bebas.

Topologi Ring

Pada topologi ini, komputer dalam jaringan dihubungkan secara melingkar sehingga tidak mempunyai ujung. Kerugian topologi ini adalah bila sebuah komputer tidak bekerja maka jaringan juga tidak akan bekerja.

Metode Akses

1. *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD).

   Pada metode ini setiap *node* sebelum mengirimkan data akan "mendengarkan" (*sensing*) apakah jaringan sedang

dipakai oleh *node* lain atau tidak. Bila bebas, maka *node* ini akan mengirimkan datanya. Bila secara bersamaan ada *node* lain yang mengirimkan data, maka akan terjadi tabrakan dan hal ini dapat dideteksi oleh *node* pengirim, *node* pengirim kemudian akan menunggu dalam waktu yang acak untuk mengirimkan ulang datanya.

2. *Token Passing*

   Untuk pengiriman digunakan suatu token yang akan dikirimkan secara "estafet" dari *node* yang satu ke *node* yang lain. *Node* yang menerima token yang bebas yang dapat mengirimkan data.

1.5 *Ethernet*

Ethernet merupakan teknologi LAN yang paling umum digunakan. *Ethernet* menggunakan metode akses CSMA/CD. Pada *ethernet*, data dikirimkan kepada semua *node* yang ada pada jaringan, tetapi hanya *node* tujuan yang mengambil data tersebut. Standar untuk *ethernet* ditetapkan oleh IEEE dan hal ini akan dibahas lebih lengkap pada Bab III. Untuk membuat suatu jaringan lokal (LAN) digunakan sebuah *Ethernet card* pada masing-masing komputer. Media yang digunakan yaitu kabel UTP dan kabel koaksial. Standar yang umum dipakai adalah 10BASE2 *Interface* untuk kabel koaksial (BNC) dan 10BASET untuk kabel UTP.

1.6 Peralatan Lain Pendukung Jaringan

Repeater

Repeater pada dasarnya berguna untuk "memperpanjang" pembatasan kabel. *Repeater* akan memperkuat semua sinyal yang diterimanya dan diteruskan ke kabel berikutnya. *Repeater* digunakan pada topologi bus, karena pada topologi ring setiap *node* sudah merupakan *repeater*. Bila dibandingkan dengan model OSI, maka *repeater* hanya menyangkut *Physical layer* saja.

Bridge

Bridge digunakan untuk menghubungkan 2 buah jaringan dengan tipe yang sama. *Bridge* membangun dan menjaga *database* yang memuat alamat peralatan yang dikenal dan bagaimana untuk mencapai peralatan itu. Ketika menerima *frame*, *bridge* akan memeriksa *database*-nya untuk menentukan yang mana dari hubungannya yang harus digunakan untuk meneruskan frame. *Bridge* ini hanya mengimplementasikan *Physical layer* dan *Data Link layer* dari model OSI.

Router

Router merupakan alat yang digunakan untuk mengantar *datagram* hingga sampai di tujuan dengan tepat. *Router* juga dapat memilih jalan alternatif untuk mengantarkan data ke tujuan, bila terdapat beberapa jalan menuju tujuan dan salah satunya terputus. *Router* menyangkut *Physical layer*, *Data Link layer*, dan *Network layer* pada model OSI.

Hub

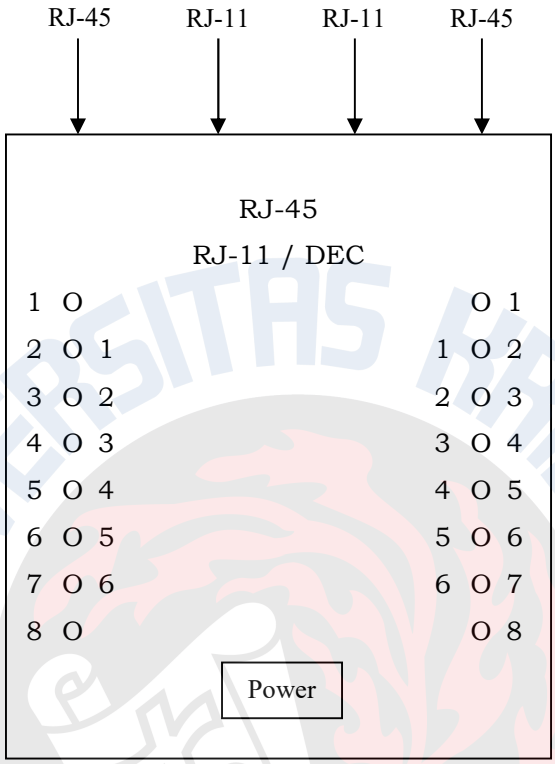
Hub merupakan alat untuk menghubungkan komputer-komputer pada sebuah jaringan. *Hub* juga disebut *Multiport Repeater*, karena kerjanya hampir sama dengan *repeater*. *Active hub* memperkuat sinyal yang diterimanya, kemudian meneruskannya ke komputer lain pada jaringan.

1.7 Alat Pendukung untuk Pemasangan Jaringan dengan Kabel UTP

Link Tester

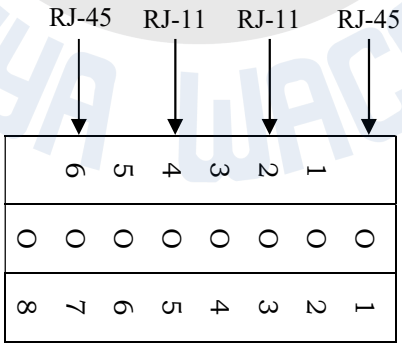
Link Tester merupakan alat yang digunakan untuk memeriksa kabel UTP apakah sudah terpasang dengan benar atau tidak. Jenis kabel yang dapat dicoba dengan alat ini adalah kabel dengan *connector* RJ-45 dan RJ-11/DEC. Alat ini terdiri dari 2 bagian, masing-masing bagian untuk masing-masing ujung kabel. Pada bagian atas dari alat ini terdapat 4 buah *connector*, 2 *connector* untuk RJ-45 dan 2 *connector* untuk RJ-11. Gambar alat ditunjukkan pada Gambar 1.5, Gambar 1.6, dan Gambar 1.7.

Bagian 1 (tampak depan)



Gambar 1.5 *Link Tester* (tampak depan)

Bagian 2 (tampak depan, horizontal)



Gambar 1.6 *Link Tester* (tampak belakang)

Keterangan :

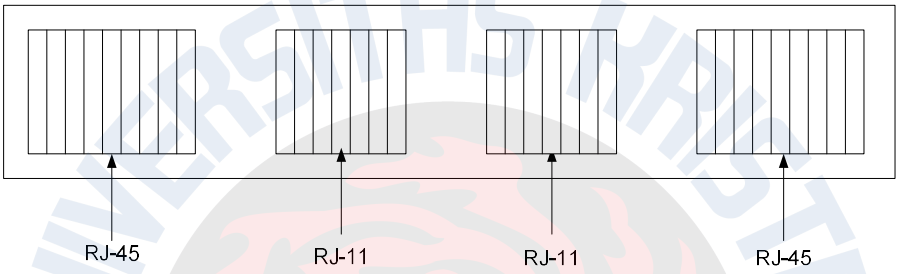
1 – 8 menunjukkan nomor pada connector RJ-45

1 – 6 menunjukkan nomor pada connector RJ-11

O menunjukkan LED

⟶ menunjukkan tempat ujung kabel dihubungkan dengan alat.

Bagian *Connector*



Gambar 1.7 Bagian *Connector*

Cara Pengkabelan UTP

Standar pengkabelan untuk UTP menggunakan standar sebagai berikut.

EIA 568B

1. *White with Orange stripe.*
2. *Orange with White stripe.*
3. *White with Green stripe.*
4. *Blue with White stripe.*
5. *White with Blue stripe.*
6. *Green with White stripe.*
7. *White with Brown stripe.*
8. *Brown with White stripe.*

EIA 568A

1. *White with Green stripe.*
2. *Green with White stripe.*
3. *White with Orange stripe.*
4. *Blue with White stripe.*
5. *White with Blue stripe.*
6. *Orange with White stripe.*
7. *White with Brown stripe.*
8. *Brown with White stripe.*

Untuk hubungan *hub* ke *hub* atau komputer ke komputer, salah satu ujung kabel menggunakan 568A dan ujung yang lain 568B atau *Cross Over Cable*, sedangkan untuk hubungan *hub* ke komputer, kedua ujung adalah 568B atau *Straight Cable*.

Arsitektur TCP/IP

Model protokol TCP/IP terdiri atas 4 layer, yaitu *Network Interface Layer*, *Internet Layer*, *Host-to-Host Layer*, dan *Application Layer*. Gambar 1.8 menunjukkan perbandingan antara model TCP/IP dan model OSI dan beberapa protokol yang terdapat pada layer dari model TCP/IP.

<table>
<tr><td rowspan="3">Application Layer</td><td>File Transfer</td><td>Email</td><td>Teminal Emulation</td><td>File Transfer</td><td>Client/ Server</td><td>Network Management</td><td>Application Layer</td></tr>
<tr><td>FTP

RFC 959</td><td>SMTP

RFC 821</td><td>TELNET

RFC 854</td><td>TFTP

RFC 783</td><td>NFS

RFC 1014</td><td rowspan="2">SNMP</td><td>Presentation Layer</td></tr>
<tr><td colspan="5"></td><td>Session Layer</td></tr>
<tr><td>Host-to-host Layer</td><td colspan="3">TCP
RFC 793</td><td colspan="3">UDP
RFC 768</td><td>Transport Layer</td></tr>
<tr><td>Internet Layer</td><td colspan="2">ARP
RFC 826</td><td colspan="2">IP
RFC 791</td><td colspan="2">ICMP
RFC 792</td><td>Network Layer</td></tr>
<tr><td rowspan="2">Network Interface Layer</td><td colspan="6">Ethernet, StarLAN, TokenRing, etc.</td><td>Data Link Layer</td></tr>
<tr><td colspan="6">Trans. Media : Coax, UTP, etc.</td><td>Physical Layer</td></tr>
</table>

Gambar 1.7 Perbandingan TCP/IP, OSI, dan Beberapa Protokol

BAB II

NETWORK INTERFACE LAYER

2.1 *Physical Layer & Data Link Layer*

Kedua lapisan (*layer*) ini bertanggung jawab untuk pertukaran data antara *host* dengan jaringan, serta untuk pengiriman data antara dua peralatan pada jaringan, atau dapat dikatakan bahwa lapisan ini bertugas untuk menaruh *frame* pada kabel untuk dikirimkan dan mengambil *frame* dari kabel jika menerima data. Protokol pada *layer* ini harus mengetahui jenis jaringan yang digunakan (misalnya X.25 atau ATM), juga bentuk paket dari jaringan. Karena TCP/IP dirancang untuk tidak bergantung pada *hardware* tertentu, maka TCP/IP ini dapat diterapkan pada berbagai jenis jaringan, misalnya *Ethernet*, *Token Ring*, ARCNet, FDDI, X.25, *Frame Relay*, dan ATM. Fungsi yang dilakukan oleh layer ini termasuk pengkapsulan (*encapsulation*) IP *datagram* dalam *frame* yang akan dikirimkan melalui jaringan, dan pemetaan IP *Address* ke *physical address*. Untuk pemetaan ini digunakan *Address Resolution Protocol* (ARP).

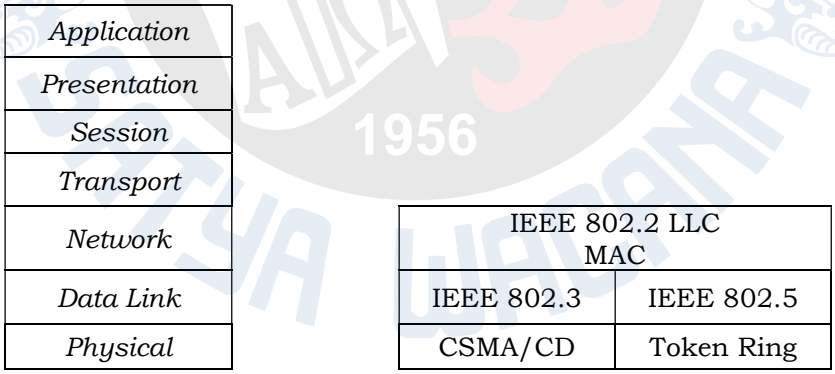
Institute of Electrical and Electronic Engineers (IEEE) mendefinisikan standar internasional bagi *Local Area Network* (LAN) yaitu IEEE 802. Standar IEEE 802 berhubungan dengan data *link layer* dan *physical layer* OSI. Arsitektur IEEE 802 mendefinisikan dua *sublayer* yang berhubungan dengan data *link layer* dari OSI, yaitu:

1. *Logical Link Control* (LLC)
2. *Medium Access Control* (MAC)

Standar 802.X adalah sebagai berikut :

- 802.1 menjelaskan hubungan standar IEEE 802 dengan model OSI.
- 802.2 mendefinisikan protokol *sublayer* LLC.
- 802.3 mendefinisikan MAC dan *physical layer* bagi CSMA/CD (Ethernet 802.3).
- 802.4 menjelaskan *Token Passing* dengan topologi bus.
- 802.5 mendefinisikan MAC dan *physical layer* bagi jaringan *Token Ring* berdasarkan pada teknologi *token ring* IBM.

Hubungan antara standar IEEE 802 dengan OSI ditunjukkan oleh Gambar 2.1.



Gambar 2.1 Hubungan antara Standar IEEE 802 dengan OSI

Dari Gambar 2.1 dapat disimpulkan bahwa standar IEEE 802 hanya mencakup 2 *layer* terbawah dari model OSI. Pada tulisan ini hanya dibahas standar **IEEE 802.3** yaitu

CSMA/CD yang paling sering digunakan sebagai metode akses pada LAN.

2.2 Jaringan IEEE 802.3 (*Ethernet* 802.3)

Jaringan-jaringan IEEE 802.3 menggunakan metode akses CSMA/CD. Media yang digunakan adalah kabel dengan standar yang telah ditetapkan sebagai berikut :

- 10BASE5, merupakan *ethernet* yang menggunakan kabel koaksial dan dapat diperpanjang sampai 500 meter tanpa repeater, sekarang jarang digunakan.
- 10BASE2, juga merupakan *ethernet* yang menggunakan kabel koaksial, dan lebih banyak digunakan.
- 10BASET, menggunakan kabel *twisted pair*.

Format *frame* pada IEEE 802.3 :

Preamble (7 octets)	Start Frame Delimiter (1 octet)	Destination Address (6 octets)	Source Address (6 octets)	Length (2 octets)	Data (46-1500 octets)	FCS (4 octets)

Format *frame* pada DIX *Ethernet* (*Ethernet* II) hampir sama dengan *frame* pada IEEE 802.3, sehingga tidak pernah ada konflik di antara keduanya, format *frame* yang sering digunakan adalah milik *Ethernet* II.

Preamble (8 octets)	Destination Address (6 octets)	Source Address (6 octets)	Length (2 octets)	Data (46-1500 octets)	FCS (4 octets)

Preamble merupakan *field* pembukaan yang menunjukkan dimulainya *frame*. *Preamble* tidak dihitung sebagai bagian dari panjang *frame*.

- *Destination Address* dan *Source Address* terdiri dari 48 bit. Alamat ini terdiri dari 1 bit yang menentukan alamat fisik

(='0') atau alamat *multicast* ('1') pada jaringan, 23 bit *Vendor Code* dan 24 bit *Globally Administered Address*.

- *Type* (juga disebut *Ethertype*) adalah suatu *field* 16 bit yang menunjukkan jenis data dari *field data*. Contohnya untuk nilai *Type* = 0800H, tipe datanya adalah IP, dan untuk nilai *Type* = 0805H, tipe datanya adalah X.25 Level 3.
- *Field Data* mengandung unit data protokol yang diterima dari protokol lapisan atas. Jika yang digunakan adalah TCP/IP, maka *field* data terdiri dari tiga komponen yaitu IP *Header*, TCP *Header*, dan data aplikasi.
- *Frame Check Sequence* (FCS) merupakan kode 32 bit yang memungkinkan *node* penerima untuk menentukan apakah ada kesalahan pada *frame* yang diterima.

Panjang minimal dari suatu *frame ethernet* adalah 64 oktet dan panjang maksimalnya adalah 1518 oktet. *Frame* yang lebih pendek dari 64 oktet tidak diperbolehkan, dan *field data* harus diisi untuk mencapai panjang minimum *frame* yang diperbolehkan, biasanya diisi dengan '0'.

BAB III

NETWORK LAYER

Lapisan ini bertanggung jawab untuk pengiriman data antar jaringan. Paket data pada Internet disebut *Internet Datagram* (IP Datagram). Protokol utama pada *layer* ini adalah *Internet Protocol* (IP). IP juga menggunakan beberapa protokol lain untuk menunjang kerjanya seperti *Address Resolution Protocol* (ARP), *Internet Control Message Protocol* (ICMP), dan *Internet Group Management Protocol* (IGMP).

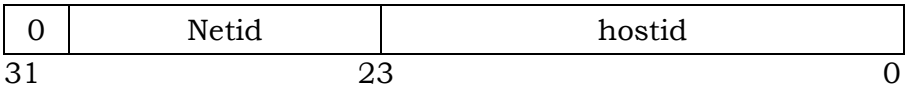
3.1 IP *Address*

Format IP *Address*

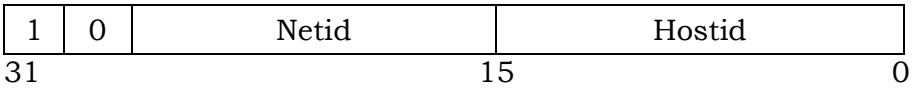
Panjang dari IP *Address* adalah 32 bit. Setiap IP *Address* mendefinisikan *network* ID (netid) dan *host* ID (hostid). *Field* netid menunjukkan jaringan (atau *segment* fisik) ke mana *host* dihubungkan, dan hostid memberikan suatu pengenal unik pada setiap host pada suatu jaringan.

Kelas-kelas Alamat:

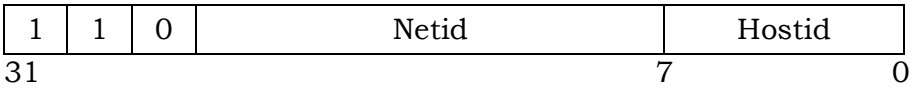
Class A



Class B

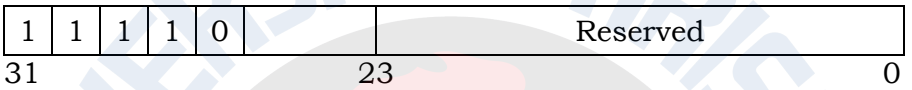


Class C



Class D

Class E



Agar memudahkan penggunaan IP *Address*, dibuat *dotted-decimal notation*, dimana setiap angka desimal mewakili 8 bit, contoh: 11000001 00001010 00011110 00000010 ditulis 193.10.30.2.

Alamat kelas A diberikan pada suatu jaringan yang memiliki jumlah *host* sangat banyak. *High-order bit* pada alamat kelas A selalu 0. Alamat kelas B diberikan pada jaringan berukuran medium hingga besar. *High order bit* pada alamat ini adalah 10. Alamat kelas C digunakan pada LAN yang kecil. Alamat kelas D digunakan untuk *multicast group*, dan alamat kelas E digunakan untuk eksperimen.

Pembatasan IP Address

Netid dan hostid 0 (biner 00000000) tidak diijinkan, karena 0 berarti "*this network*". Suatu alamat IP 155.123.0.0 mengenali jaringan dengan netid 155.123. Suatu alamat 0.0.0.35 mengenali *host* dengan hostid 35 pada jaringan lokal.

Netid 127 (biner 01111111) mempunyai penggunaan khusus. Ini merupakan alamat *loopback* yang digunakan untuk memeriksa konfigurasi jaringan *host*. Pesan-pesan yang dialamatkan ke netid 127 tidak dikirimkan ke jaringan, tetapi hanya dikembalikan lagi.

Hostid 255 dibatasi penggunaannya. Suatu pesan yang dikirim ke 255.255.255.255 dikirimkan ke setiap *host* pada jaringan ini. Suatu pesan yang dikirimkan ke 183.20.255.255 dikirimkan ke setiap *host* pada jaringan 183.20.

Alamat-alamat IP yang tersedia ditunjukkan pada tabel berikut.

Kelas	Jumlah *network*	Jumlah *host* per *network*	Jangkauan *network* ID
A	126	16.777.214	1-126
B	16.384	65.534	128-191
C	2.097.152	254	192-223

Untuk *host* ID yang valid dapat dilihat pada tabel berikut.

Address Class	*Beginning range*	*Ending range*
Class A	w.0.0.1	w.255.255.254
Class B	w.x.0.1	w.x.255.254
Claas C	w.x.y.1	w.x.y.254

Pengalamatan *Subnet*

Subnet adalah sebuah segmen fisik dalam lingkungan TCP/IP yang menggunakan IP *Address* yang diturunkan dari sebuah *network* ID tunggal. Sebuah *subnet* ID yang unik dibuat untuk setiap segmen dengan membagi bit pada *host* ID menjadi 2 bagian. Bagian yang pertama digunakan untuk

mengidentifikasi segmen sebagai sebuah jaringan yang unik, dan bagian yang lain digunakan untuk mengidentifikasi *host*. Dengan *subnetting*, IP *address* menjadi 3 bagian yaitu netid + subnetid + hostid. *Subnet* ID dibuat dengan mengambil bit dari *field host* ID menggunakan teknik yang disebut *subnet masking*. *Subnet mask* adalah sebuah bilangan 32 bit. Angka 1 pada *subnet mask* menunjukkan bahwa bit tersebut pada alamat IP adalah bagian dari netid. Angka 0 pada *subnet mask* menandakan bahwa bit tersebut adalah bagian dari hostid. *Subnet mask* ini digunakan untuk membedakan netid dari hostid dan menentukan apakah alamat tujuan berada pada jaringan lokal (*local network*) atau jaringan *remote* (*remote network*). Setiap *host* pada jaringan TCP/IP membutuhkan sebuah *subnet mask*, bila jaringan tidak dibagi dalam subnet, maka digunakan *subnet mask default*.

Kelas A : 11111111 00000000 00000000 00000000
= 255.0.0.0
Kelas B : 11111111 11111111 00000000 00000000
= 255.255.0.0
Kelas C : 11111111 11111111 11111111 00000000
= 255.255.255.0

Untuk menentukan apakah alamat tujuan berada pada jaringan lokal atau jaringan *remote*, IP *address* sumber dan IP *address* tujuan di-AND dengan *subnet mask*. Jika hasil keduanya sama, maka tujuan berada pada jaringan lokal, sehingga data langsung dikirim ke tujuan, tetapi jika tujuan berada pada jaringan *remote*, maka data akan dikirimkan ke *router* (*gateway*).

3.2 *Address Resolution Protocol* (ARP)

Address Resolution Protocol (ARP) bertugas untuk memetakan (*mapping*) IP *address* dari sebuah *host* ke *Hardware address*-nya. Pada setiap *host* terdapat ARP *cache*, ARP *Request* diinisialisasi saat sebuah *host* akan berkomunikasi dengan *host* lain. Jika *destination host* berada pada jaringan lokal, *source host* akan memeriksa ARP *cache*-nya apakah *hardware address* dari *destination host* ada atau tidak. Jika tidak ada, maka ARP akan mengirimkan *request* (pada *request* ini terdapat IP *Address*) pada setiap *host* pada jaringan. Untuk jaringan lokal ARP menggunakan *local broadcast*. *Host* yang memiliki IP *address* yang cocok dengan yang ada dalam *request* akan megirimkan ARP *Reply* beserta dengan *hardware address*-nya. Bila IP *address* tujuan adalah *remote address* (atau *host* yang berada pada jaringan lain) maka *source host* akan memeriksa *routing table*-nya dan akan mengirimkan ARP *Request* ke *router*. Untuk proses yang berkebalikan dengan yang dilakukan oleh ARP, dilakukan oleh RARP.

Format paket ARP/RARP ditunjukkan pada Gambar 3.1.

0 8 16 31

Hardware Type		*Protocol Type*
HLEN	PLEN	*Operation*
Sender HA (*octets* 0-3)		
Sender HA (*octets* 4-5)		*Sender* IP (*octets* 0-1)
Sender IP (*octets* 2-3)		*Target* HA (*octets* 0-1)
Target HA (*octets* 2-5)		
Target IP (*octets* 0-3)		

Gambar 3.1 Format Paket ARP/RARP

Hardware Type (2 octet). Mendefinisikan tipe *hardware* yang digunakan.

Contoh: 1 = *Ethernet*.

6 = IEEE 802.

Protocol Type (2 octet). Mendefinisikan *protocol address* yang digunakan.

HLEN (*Hardware Length*, 1 octet) dan PLEN (*Protocol Length*, 1 *octet*). Mendefiniskan panjang dari *address* yang digunakan. HLEN = 6 *octet* (48 bit). PLEN = 4 *octet* (32 bit).

Operation:

1 = ARP *Request*.

2 = ARP *Reply*.

3 = RARP *Request*.

4 = RARP *Reply*.

Pada saat terjadi ARP *Request*, *field* Target HA diisi dengan 0. ARP *Reply* dari *target host* akan mengisinya dengan *hardware address*-nya. Informasi ini kemudian disimpan oleh *source host* pada ARP *cache*-nya. Setiap *entry* dari ARP *cache* ini mempunyai waktu hidup (*life time*) untuk mencegah agar *cache* tidak terlalu besar.

3.3 *Internet Protocol* (IP)

Fungsi yang dijalankan oleh IP adalah :

1. Pengalamatan (*Addressing*).

2. Memindahkan data antara *Network Interface Layer* dan *Host-to-host Layer*.
3. *Routing datagram* ke *remote host*.
4. Fragmentasi dan penggabungan kembali *datagram*.

IP mempunyai karakteristik yang khusus. Pertama, IP adalah *connectionless protocol*. *Connectionless* berarti IP tidak melakukan *handshake* atau pertukaran informasi untuk membentuk suatu *end-to-end connection* sebelum mentransmisikan data, IP mengandalkan protokol pada *layer* di atasnya untuk membentuk *connection* bila suatu aplikasi membutuhkan layanan yang *connection-oriented*. Kedua, IP menyediakan layanan *unreliable* (*unreliable delivery*), artinya IP tidak menyediakan layanan *error detection* dan *error recovery*, yang bertanggung jawab terhadap *error detection* dan *error recovery* adalah protokol pada *layer* sebelah atas.

Setelah menerima data dari *Host-to-host Layer* untuk dikirimkan, pada *Internet Layer* ini, IP menambahkan *header*-nya pada data.

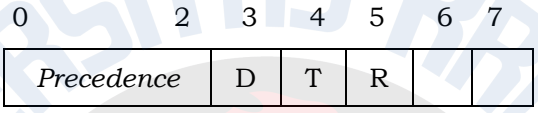
Gambar 3.2 menunjukkan format dari IP *Header*.

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragmentation offset</td></tr>
<tr><td>Time To Live</td><td colspan="2">Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
</table>

Gambar 3.2 Format IP *Header*

Header IP berisi *field-field* berikut :

- *Version* (4 bit). Menunjukkan versi dari IP. Versi saat ini adalah versi 4 (RFC 791).
- *Internet Header Length* (IHL, 4 bit). Menjelaskan panjang dari *header* menggunakan 32-bit *word*. Ukuran minimum *header* yang diijinkan adalah 5 *word*.
- *Type of Service* (8 bit). Menunjukkan kualitas layanan yang diinginkan.



Bit 0-2 : *Precedence*
111 – *Network Control*.
110 – *Internetwork Controlwork*.
101 – CRITIC/ECP.
100 – *Flash Override*.
011 – *Flash*.
010 – *Immediate*.
001 – *Priority*.
000 – *Routine*.

Bit 3 : *Delay* : 0 = *Normal Delay*.
1 = *Low Delay*.

Bit 4 : *Throughput* : 0 = *Normal Throughput*.
1 = *High Throughput*.

Bit 5 : *Realibility* : 0 = *Normal Reliability*.
1 = *High Reliability*.

Bit 6 – 7 : diset 00 (*reserved for future use*).

- *Total Length* (16 bit). Panjang keseluruhan *datagram* dalam oktet, termasuk *header* dan data. *Field* ini

memungkinkan *datagram* berisi sampai 65535 oktet. Panjang *datagram* minimum adalah 576 oktet.

- *Identification* (16 bit). *Field* ini digunakan untuk membantu proses penggabungan kembali *fragment* menjadi *datagram*.
- *Flags* (3 bit). *Field* ini berisi 3 *control flag*.

  Bit 0 : *Reserved* (Set to 0).

  Bit 1 : DF. 0 = *May fragment*.

  1 = *Don't fragment*.

  Bit 2 : MF. 0 = *Last fragment*.

  1 = *More fragment*.
- *Fragment Offset* (13 bit). Untuk *datagram* yang dipecah, *field* ini menunjukkan posisi *fragment* dalam *datagram*.
- *Time To Live* (TTL, 8 bit). Menunjukkan waktu maksimum bagi sebuah *datagram* untuk berada pada jaringan. Bila *field* ini bernilai 0 maka *datagram* akan dibuang. *Field* ini dimodifikasi selama tahap pemrosesan *header* IP dan umumnya dihitung dalam detik. Setiap kali *datagram* ini sampai pada ke sebuah *router* atau *host*, maka *field* TTL ini akan dikurangi 1.
- *Protocol* (8 bit). Menunjukkan protokol bagian atas yang berhubungan dengan *datagram* (didefinisikan pada RFC 1700: *Assigned Numbers*). Contoh nilai (desimal): 1 = ICMP, 6 = TCP, 17 = UDP.
- *Header Checksum* (16 bit). Nilai ini dihitung ulang setiap kali *header* dimodifikasi.
- *Source Address* (32 bit). Alamat IP dari *host* yang mengirimkan *datagram*.
- *Destination Address* (32 bit). Alamat IP dari host yang merupakan tujuan *datagram*.
- *Options*. Dapat berisi 0 atau lebih pilihan (RFC 791).

Routing Datagram

IP bertanggung jawab dalam pengiriman *datagram* ke tujuan melalui *internetwork*. Setiiap tipe jaringan mempunyai ukuran besarnya data yang dapat ditransfer, atau MTU (*maximum transmission unit*). Jika datagram yang diterima lebih besar dari MTU maka diperlukan pemecahan *datagram* menjadi *fragment-fragment* untuk kemudian ditransmisikan, IP pada *host* tujuan kemudian akan merakit kembali *fragment-fragment* ini menjadi *datagram*. Saat sebuah *datagram* dikirim, IP akan menetukan apakah *datagram* ini ditujukan ke jaringan lokal atau ke jaringan *remote*. Untuk pengiriman lokal, gambaran singkat prosesnya adalah sebagai berikut:

1. IP menerima *frame* dari protokol level di atasnya (dari *Host-to-host layer*).
2. IP membandingkan netid tujuan dangan netid dari jaringan lokal (termasuk memperhitungkan *subnet mask* bila digunakan *subnetting*). Bila netid-nya sama, maka *frame* dapat dikirimkan langsung ke alamat tujuan.
3. IP memperoleh alamat *hardware* dari tujuan dari ARP.
4. IP menyusun *datagram*.
5. IP meneruskan *datagram* ini ke lapisan *Network Interface*.
6. Lapisan *Network Interface* menyusun *frame* dimana terdapat alamat *hardware* pengirim dan penerima (tujuan) kemudian dikirimkan malalui media.

Untuk pengiriman data ke jaringan *remote*, paket dikirimkan ke *router*, biasanya pada jaringan lokal dikonfigurasikan sebuah *default gateway*, yaitu *host* yang harus dituju oleh paket bila paket tersebut akan dikirimkan ke *host* pada

jaringan *remote*. Sebuah *router* (*gateway*) pada prinsipnya adalah sebuah *host* yang mempunyai dua atau lebih koneksi jaringan (*multihomed host*). Sebuah *router* melaksanakan protokol dari 2 *layer* terbawah TCP/IP yaitu *Network Interface layer* dan *Internet layer*. Saat sebuah paket diterima pada sebuah *router*, paket tersebut akan diberikan ke *Internet layer*. IP pada *router* akan melakukan hal-hal sebagai berikut:

1. IP mengurangi TTL paling dengan 1 atau lebih bila paket berkurang pada *router* karena adanya kemacetan pada jaringan. Jika TTL = 0, maka paket dibuang (*discarded*).
2. IP dapat memecah paket menjadi paket-paket yang lebih kecil, jika paket terlalu besar untuk jaringan berikutnya.
3. Jika paket dipecah, IP membuat *header* baru untuk setiap paket baru, yang mengandung *flag*, *fragment* ID, dan *fragment offset*.
4. IP menghitung *checksum* yang baru.
5. IP menentukan alamat *hardware* tujuan dari *router* selanjutnya.
6. IP mengirimkan paket tersebut.

Hal di atas terus dilakukan hingga paket sampai pada tujuannya.

Protokol yang digunakan untuk melakukan *routing* pada *router* dibagi atas 2 kategori yaitu *Interior Gateway Protocol* (IGP) dan *Exterior Gateway Protocol* (EGP). IGP digunakan untuk melakukan dalam sebuah *autonomous system*, dan EGP digunakan untuk melakukan *routing* di antara *autonomous system*. *Autonomous system* adalah kumpulan *router* yang memiliki *routing protocol* yang sama. *Router* menggunakan ukuran *metric* atau pengukuran jarak

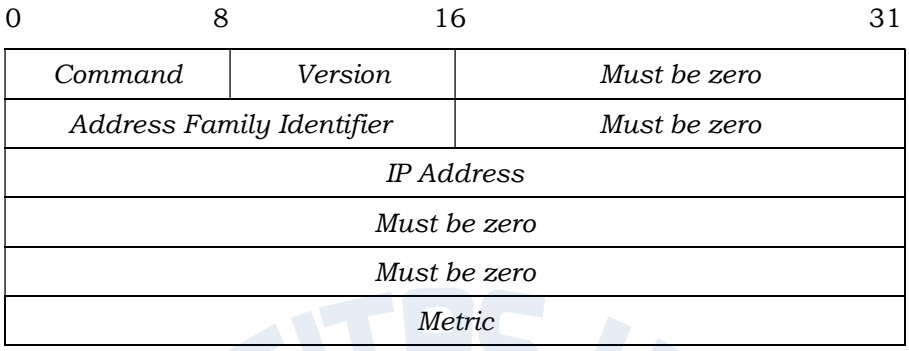
terpendek dari 2 buah titik untuk menentukan jalan yang optimum. *Router* dalam jaringan menggunakan *Distance Vector* (Bellman-Ford) *Algorithm* dan *Link-State Algorithm* untuk melakukan *routing*. Protokol-protokol yang menggunakan kedua algoritma ini adalah IGP. Yang termasuk dalam IGP adalah *Routing Information Protocol* (RIP) dan *Open Shortest Path First* (OSPF). Contoh dari EGP adalah *Border Gateway Protocol* (BGP).

***Routing Information Protocol* (RIP)**

RIP digunakan untuk komunikasi *intra-router* atau *intra-gateway*, RIP didasarkan pada Agoritma *Distance-Vector*. Dengan algoritma ini, *router-router* secara periodik bertukar informasi dari *routing table* mereka. *Routing decision* didasarkan pada jumlah hop terkecil untuk mencapai suatu tujuan. RIP mengijinkan panjangnya jalan adalah 15 hop. RIP menyimpan semua informasi mengenai rute ke jaringan tujuan dalam sebuah tabel *routing*, sehingga memungkinkan IP untuk memilih rute yang memiliki hop terkecil untuk mencapai tujuan. Suatu *entry* pada tabel *routing* berisi informasi-informasi sebagai berikut:

- IP *address* dari tujuan.
- Nilai *metric* untuk mengantarkan paket hingga tujuan.
- IP *address* dari router berikutnya pada rute menuju tujuan.
- *Flag* yang menandakan apakah informasi *routing* pada tabel *routing* telah berubah.
- *Timer*.

Format dari paket RIP ditunjukkan pada Gambar 3.3.



Gambar 3.3 Format Paket RIP

- Command: 1. *Request for routing table information*.
  2. *Response containing routing table information*.
  3. *Trace on* (*obsolete*).
- *Trace off* (*obsolete*).
- *Reserve for* SUN Microsystems.
- *Version* (*currently* = 1).
- *Address Family Identifier*, menunjukkan *address family* yagn ditransmisikan dalam paket RIP tersebut.
- IP *Address* = IP *Address destination*.
- *Metric*, menunjukkan *metric* untuk mencapai tujuan. Nilainya 1 sampai 15, bila *metric* = 16 maka tujuan yang diinginkan tak dapat dicapai atau *unreachable*.

32-bit word ke-2 dari paket dapat berulang hingga 25 kali.

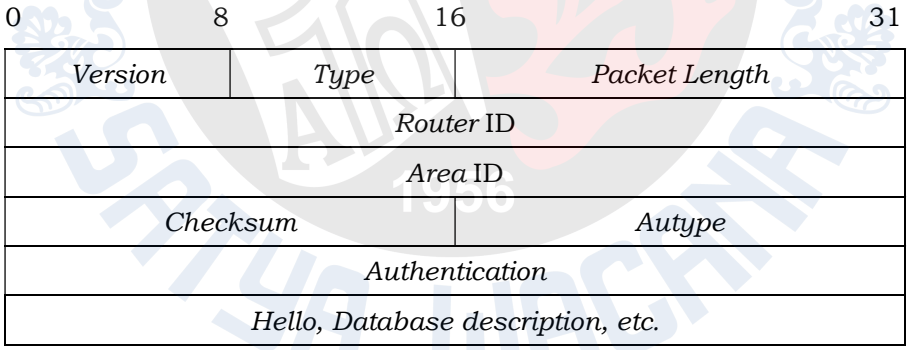
Open Shortest Path First* (OSPF) *Protocol

OSPF adalah protokol *routing* yang bersifat *link-state* (menggunakan algoritma *link-state*), yang berarti tiap *router* mengelola *database* yang berisi topologi dari *local autonomous system*. OSPF juga dapat menghitung minimum-cost routes

agar *traffic load* menjadi seimbang (*balance*). Ada 5 tipe paket OSPF yaitu:

1. *Hello Packet*, merupakan pengiriman yang dilakukan secara periodik yang membawa informasi mengenai *router* tetangga.
2. *Database Description Packet*, membawa informasi yang dibutuhkan untuk menginisialisasi *database* topologi dari *device* yang berdekatan.
3. *Link State Request Packet*, untuk memperoleh informasi *database* dari *router* tetangga.
4. *Link State Update Packet*, memberitahukan status dari berbagai *link* dalam jaringan.
5. *Link State Acknowledgement Packet*, memeriksa informasi *database* yang baru diterima.

Format paket OSPF dapat dilihat pada Gambar 3.4.



Gambar 3.4 Format Paket OSPF

- *Version*, menunjukkan versi dari OSPF (=1).
- *Type*, menunjukkan tipe paket OSPF (1-5)
- *Packet Length*, menunjukkan panjang dari paket OSPF beserta *header*-nya.
- *Router* ID, menunjukkan ID *router* yang memiliki paket.

- *Area* ID, menunjukkan area asal dari paket (area = *autonomuos system*).
- *Checksum*, *Authentication Type*, dan *Authentication field* digunakan untuk mengecek paket.

3.4 *Internet Control Message Protocol* (ICMP)

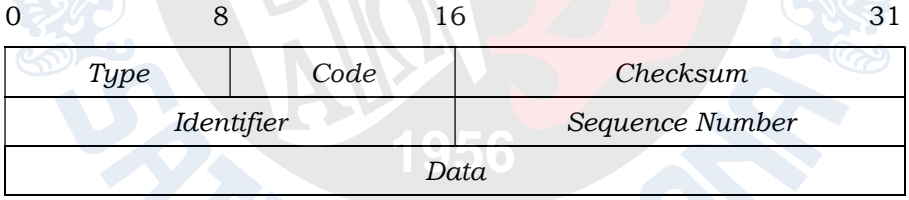
IP menyediakan sebuah *connectionless service* untuk menghubungkan *host-host*, karena itu digunakan ICMP untuk melaporkan *error-error* yang dapat terjadi selama memproses *datagram*, misalnya *undeliverable datagram* dan *incorrect routes*. ICMP ini merupakan bagian integral dari IP dan pesan ICMP dikirimkan dalam bentuk data pada IP *datagram*.

Bila *field Protocol* pada IP *Header* = 1 berarti terdapat pesan ICMP dalam *datagram*. Berikut ini adalah tipe dari pesan ICMP.

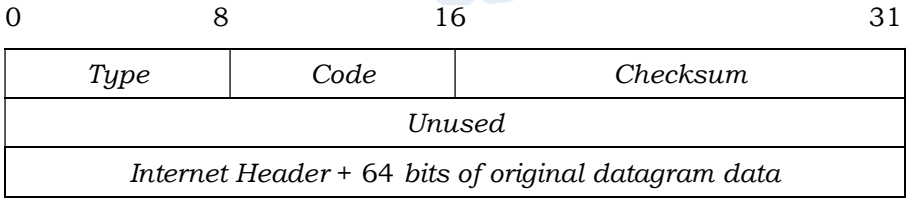
Format *header* dan pesan ICMP ditunjukkan pada Gambar 3.5.

Type	ICMP *Messages*
0	*Echo Reply*
3	*Destination Unreachable*
4	*Source Quench*
5	*Redirect*
8	*Echo*
11	*Time Exceeded*
12	*Parameter Problem*
13	*Timestamp*
14	*Timestamp Reply*
15	*Information Request (obsolete)*
16	*Information Reply (obsolete)*
17	*Address Mask Request*
18	*Address Mask Reply*

Echo dan Echo Reply message.



Destination Unreachable, *Source Quench*, dan *Time Exceeded messages.*



Redirect message.

0	8	16	31
Type	*Code*	*Checksum*	
Gateway Internet Address			
Internet Header + 64 *bits of original datagram data*			

Parameter problem message

0	8	16	31
Type	*Code*	*Checksum*	
Pointer	*Unused*		
Internet Header + 64 *bits of original datagram data*			

Timestamp dan Timestamp Reply message.

0	8	16	31
Type	*Code*	*Checksum*	
Identifier	*Sequence Number*		
Originate Timestamp			
Receive Timestamp			
Transmit Timestamp			

Address Mask Request dan Address Mask Reply maessages

0	8	16	31
Type	*Code*	*Checksum*	
Identifier	*Sequence Number*		
Address Mask			

Gambar 3.5 Format *header* dan Pesan ICMP

32-bit *word* yang pertama yang terdiri dari *Type* (1 *octet*), *Code* (1 *octet*), dan *Checksum* (2 *octet*) merupakan ICMP *header*. *Type* meidentifikasikan sebuah pesan ICMP yang

unik, dan *Code* menjelaskan tipe pesan yang lebih spesifik. Di bawah ini merupakan penjelasan singkat dari tiap pesan ICMP.

- *Echo message* (*type* = 8) memeriksa jalur komunikasi dari *source host* ke *destination host* (atau lebih dikenal dengan ping). *Source host* megirimkan sebuah *echo message* yang memuat sebuah *identifier* dan *sequence number*. Bila *destination host* menerima *echo message* ini, maka ia akan membalas dengan *Echo Reply* (*type* = 0).

- *Destination Unreachable* (*type* = 3), digunakan saat *router* atau *host* tidak dapat mengantarkan *datagram* ke *destination host*. Pesan ini dikirimkan ke *source host* beserta dengan masalah yang dialami saat pengantaran *datagram* ke tujuan, yang didefinisikan dalam *field Code*. *Code* yang digunakan oleh *router* adalah 0, 1, 4, atau 5. *Host* menggunakan 2 atau 3.

Code	*Meaning*
0	*Net Unreachable*
1	*Host Unreachable*
2	*Protocol Unreachable*
3	*Port Unreachable*
4	*Fragmentation needed and offset*
5	*Source route failed*

Saat sebuah *datagram* diterima oleh *router*, maka *router* akan melihat tabel *routing*-nya. Jika *destination* tidak dapat dicapai, pesan *Net Unreachable* akan dikirimkan ke

source host. Jika *host* tidak dapat memproses *datagram* karena protokol atau *port* yang diminta tidak aktif maka pesan *Protocol Unreachable* atau *Port Unreachable* yang akan dikembalikan.

- *Redirect Message* (type = 5), digunakan untuk mencegah kesalahan *routing*. Bila *routing table* pada *host* tidak lengkap, maka *host* dapat mengirim *datagram* ke *router* yang salah. Jika *router* menerima *datagram* yang salah, maka *router* akan mengembalikan sebuah *Redirect Message* ke *host* untuk memberikan rute yang lebih baik. *Redirect message* ini memuat alamat *router* yang tepat untuk mencapai *destination host*. *Field Code* pada *Redirect Message* berisi inforamasi sebagai berikut:
  0 = *Redirect datagrams for the network*.
  1 = *Redirect datagrams for the host*.
  2 = *Redirect datagrams for the type of service and network*.
  3 = *Redirect datagrams for the type of service and host*.

- *Source Quench* (*type* = 4), digunakan oleh *router* bila terjadi kemacetan. *Router* akan mengirim *Source Quench message* ke *source host* dan meminta agar *source host* mengurangi outputnya.

- *Time Exceeded Messages* (*type* = 11). *Datagram* dalam jaringan dapat hilang karena *datagram* tersebut terlalu lama berada dalam jaringan sehingga melampaui TTL-nya. Kemacetan (*congestion*) juga dapat menyebabkan semua *fragment* dari *datagram* gagal dirakit kembali. Kedua

keadaan ini dapat memicu ICMP *Time Exceeded message*. Ada 2 *Code* yang digunakan:

0 = *Time to Live Exceeded in transmit*.

1 = *Fragment Reassembly Time Exceeded*.

- *Parameter Problem Message* (*type* = 12). Pesan ini dikirim ke *source host* bila terjadi kesalahan parameter pada IP *Header*, seperti *Type of Service* yang salah. Pasan ini memuat sebuah pointer yang akan mengidentifikasikan *octet* yang mengandung *error*.

- *Timestamp Message* (*type* = 13) dan *Timestamp Reply* (*type* = 14), mengukur *round-trip transit time* antara 2 *host* dan mensinkronosasi *clock*-nya. *Host* yang meminta *Timestamp* mengisi *field Originate Timestamp* saat transmisi, dan penerimanya *field Receive Timestamp* saat menerima *request*. *Host* penerima akan mengisi *Transmit Timestamp* saat ia akan mengirimkan *Timestamp Reply*. Setelah menerima *Timestamp Reply*, *Host* yang meminta *Timestamp* melakukan estimasi *Remote Processing* dan *Round-trip Transit time*.
  - *Remote Process = Receive Timestamp – Transmit Timestamp*
  - *Round-trip transit time = Timestamp Reply message arrival time – Originate Timestamp*

- *Address Mask Request* (*type* = 17) dan *Address Mask Reply* (*type* = 18), dibutuhkan karena adanya *subnetting*. *Address Mask Request* ini dikirimkan ke tujuan. *Router* yang mengetahui *address mask* yang tepat akan

mengambalikan *Address Mask Reply*. *Filed Address Mask* dari pesan ICMP diisi dengan 0. Sebagai contoh, *respons* untuk *network* kelas B tanpa *subnetting* adalah 255.255.255.0.

Internet Group Management Protocol (IGMP) digunakan pada *multicast group*. IGMP menginformasikan pada *router* mengenai keberadaan sebuah *host* pada sebuah *multicast group*. Seperti ICMP, paket IGMP juga disertakan dalam IP *datagram*.

BAB 4

Host-to-host Layer (*Transport Layer*)

Host-to-host layer merupakan *layer* yang beroperasi pada *host*. Protokol-protokol pada *layer* ini menyediakan sesi komunikasi antara 2 buah *host*. Protokol pada *layer* ini adalah *Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP). Saat IP *datagram* diterima oleh *host*, maka *host* akan menjalankan proses (aplikasi) yang sesuai dengan yang dibutuhkan. Pada sebuah *host* dapat terjadi lebih dari satu proses, karena itu *datagram* membutuhkan tambahan alamat untuk mengidentifikasi pada proses mana *datagram* tersebut diberikan. Pengalamatan tambahan ini dibawa dalam UDP atau TCP *header* dan dikenal sebagai *Port Address*. *Port Address* ini menunjukkan sebuah proses atau aplikasi dalam sebuah *host*. Setiap *host* dapat memiliki beberapa aplikasi seperti *email*, *database*, dan lainnya. Sebuah *Port Number* mengidentifikasikan proses yang ingin diakses oleh *user*. Panjang dari *Port Number* ini adalah 16 bit. Jadi sebuah pesan atau data yang dikirimkan dari satu *host* ke *host* lainnya membutuhkan alamat ini untuk melengkapi jalur komunikasinya. *Port Address* (16 bit) mengidentifikasikan aplikasi dan dibawa dalam UDP atau TCP *header*. IP *Address* (32 bit) mengidentifikasikan alamat jaringan dan *host* dimana proses dijalankan, ditempatkan dalam IP *header*. Kombinasi dari IP *Address* dan *Port Address* disebut sebagai *Socket*.

4.1 *User Datagram Protocol* (UDP)

UDP adalah protokol yang mempunyai karakteristik *connectionless* dan *unreliable*, seperti IP, *connectionless* berarti tidak ada sesi yang dibentuk dalam pengiriman *datagram* (atau dapat dikatakan tidak melakukan *handshake*). *Unreliable delivery*, berarti pengiriman tidak dijamin berhasil, dan tidak ada *sequence number* dan *acknowledgement* yang memberitahu bahwa data telah sampai di tujuan dengan baik. UDP digunakan oleh aplikasi yang tidak membutuhkan *acknowledgement* dan yang mengirim data dalam jumlah kecil, contoh aplikasi yang menggunakan UDP adalah NetBIOS *Name Service*, NetBIOS *Datagram Service*, dan *Simple Network Management Protocol* (SNMP).

Format UDP *header*:

0 15 31

Source Port	*Destination Port*
Length	*Checksum*
Data	

- *Source Port* (2 *octet*) dan *Destination Port* (2 *octet*) adalah alamat *port* sumber dan *port* tujuan.
- *Length* (2 *octet*), menunjukkan panjang dari UDP *datagram*, nilai minimumnya adalah 8 *octet*.
- *Checksum* (2 *octet*), *field* ini adalah *optional* dan diisi dengan 0 bila proses pada protokol *layer* yang lebih atas tidak membutuhkan *checksum*. Bila *checksum* digunakan, *checksum* ini dihitung dari *pseudo header*. Pada *pseudo*

header ini terdapat *Source Address*, *Destination Address*, dan *Protocol field* dari IP *Header*.

Format UDP *Pseudo-header*:

0 15 31

Source Address		
Destination Address		
Zero	*Protocol*	*UDP Length*

Jadi IP *Destination Address* membawa *datagram* ke *host* yang tepat pada jaringan tertentu dan UDP *Port Address* membawa *datagram* ke proses yang tepat pada *host*.

Contoh alamat *port* UDP:

Port	*Keyword*	*Description*
15	NETSTAT	*What the Network Status is*
53	DNS	*Domain Name Service*
69	TFTP	*Trivial File Transfer Protokol*
137	NETBIOS-NS	NETBIOS *Name Service*
161	SNMP	SNMP *Network Monitor*

4.2 *Transmission Control Protocol* (TCP)

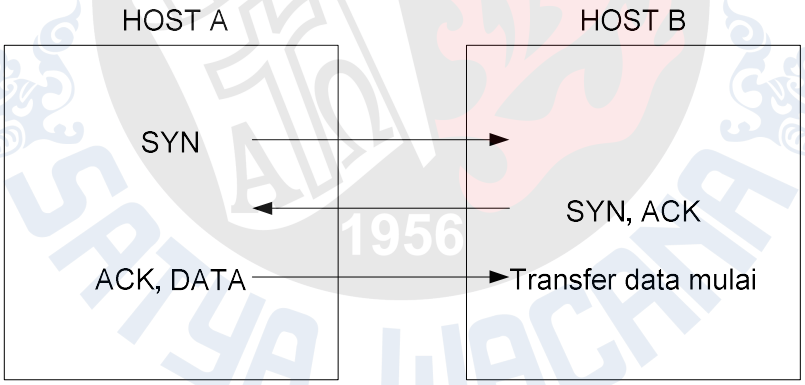
Transmission Control Protocol (TCP) merupakan protokol yang mempunyai sifat *connection-oriented* dan *reliable*. *Connection-oriented* berarti ada sebuah sesi komunikasi yang dibentuk sebelum *host* dapat saling bertukar data (disebut *handshake*). TCP menggunakan *byte-stream communication* dimana data yang dikirimkan merupakan urutan (*sequence*) *byte*. Jika

sebuah TCP *Segment* dipecah menjadi segmen-segmen yang lebih kecil, *host* penerima akan mengetahui apakah data telah diterima seluruhnya atau belum. *Reliability* dipenuhi dengan memberikan *sequence number* pada setiap segmen yang ditransmisikan, juga digunakan *acknowledgement*. Sebuah sesi TCP dibentuk melalui *three-way handshake*.

Tujuan dari *three-way handshake* ini adalah

- Mensinkronkan pengiriman dan penerimaan segmen.
- Menginformasikan *host* yang lain mengenai jumlah data yang dapat diterima dalam satu kali pengiriman (*Segment size* dan *Window size*).
- Membentuk sebuah *virtual connection*.

Proses *three-way handshake* dijelaskan pada Gambar 4.1.



Gambar 4.1 Proses *Three-Way Handshake*

Keterangan :

HOST A adalah *host* pengirim dan HOST B adalah *host* penerima.

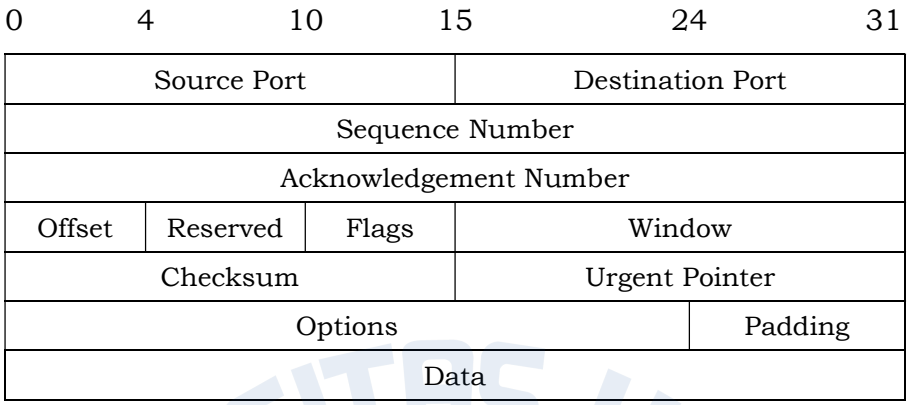
1. HOST A memulai sesi/hubungan dengan mengirimkan sebuah segmen dengan *synchronization* (SYN) *flag bit* yang

diset. Segmen ini memberitahukan pada *host* B bahwa ia ingin memulai hubungan.

2. HOST B membalas *request* dari HOST A dengan mengirimkan sebuah segmen dengan:
   - SYN *Flag set*.
   - Sebuah *sequence number* untuk menandakan *byte* awal untuk sebuah segmen yang dikirim.
   - Sebuah *acknowledgement* dengan *sequence number* dari segmen berikutnya yang ingin diterima.
3. HOST A mengirimkan segmen dengan *sequence number* dan *acknowledgement number* yang diterimanya dari HOST B.

Jika seluruh data sudah diterima dengan baik oleh HOST B, maka dilakukan pemutusan hubungan. TCP menggunakan proses *handshake* yang sama untuk mengakhiri sebuah hubungan. Sebuah segmen dengan FIN *bit* yang diset dikirimkan untuk menandakan bahwa proses sudah selesai. Sama seperti UDP, TCP juga menggunakan *Port Address* untuk mengidentifikasikan sebuah proses dalam *host*.

Format TCP *Segment* (6 *word* pertama merupakan TCP *header*) ditunjukkan pada Gambar 4.2.



Gambar 4.2 Format TCP *Segment*

TCP *header* mempunyai panjang minimum 20 octet. *Header* ini terdiri dari beberapa *field*, yang berhubungan dengan pengaturan hubungan, kontrol aliran data, dan *realibility*.

- *Source Port* (2 *octet*) dan *Destination Port* (2 *octet*), mengidentifikasikan *host process* pada kedua *host* yang saling berhubungan.
- *Sequence Number* (4 *octet*), adalah *sequence number* yang diberikan pada *octet* pertama dari data pada segmen. Saat SYN *flag* diset, *sequence number* menunjukkan *Initial Sequence Number* (ISN). *Octet* pertama dari *field* data dikirim dengan menggunakan *sequence number* berikutnya, yaitu ISN +1. *Sequence Number* menjamin urutan dari data.
- *Acknowledgement Number* (4 *octet*) melaporkan penerimaan data. Nilai pada *acknowledgement* adalah *octet* berikutnya (sama dengan *sequence number* berikutnya) yang diharapkan dari salah satu *host* dari hubungan. Saat *field Acknowledgement* sedang digunakan selama hubungan, *flag* ACK diset.

- *Offset* (4 bit), menandakan di mana akhir dari TCP *header* dan awal dari data untuk protokol lapisan yang lebih atas.
- *Flags* (6 bit). Ada 6 buah *flag* yang digunakan untuk mengontrol hubungan dan transfer data. *Flag-flag* ini adalah:
  1. URG = *Urgent Pointer field significant*.
  2. ACK = *Acknowledgement field siginificant*.
  3. PSH = *Push function*.
  4. RST = *Reset the connection*.
  5. SYN = *Synchronize Sequence Numbers*.
  6. FIN = *No more data from sender*.
- *Window* (2 *octet*), menyediakan *flow control* antara 2 *host* yang sedang melakukan hubungan. Nomor yang berada pada *window* menunjukkan kuantitas dari *octet*.
- *Checksum* (2 *octet*) digunakan untuk *control error*. Penghitungan *checksum* termasuk 12 *octet pseudo-header*, TCP *header*, dan data. TCP *pseudo-header* mirip dengan UDP *pseudo-header*.

Format TCP *Pseudo-header* terlihat pada Gambar 4.3:

0 15 31

Source Address		
Destination Address		
Zero	*Protocol*	*TCP Length*

Gambar 4.3 Format TCP *Pseudo-header*

Tujuan dari TCP *pseudo-header* adalah untuk menyediakan *control error* pada IP *header*, TCP *header* dan data. TCP *Length*

pada TCP *pseudo*-header tidak termasuk 12 octet *pseudo-header*.

- *Urgent Pointer* (2 *octet*) mengijinkan posisi data dalam TCP *segment* dapat diidentifikasi.
- *Options* dan *Padding* (panjangnya bervariasi), menunjukkan *options* yang dibutuhkan oleh TCP *process* dalam *host*, misalnya *option Maximum* TCP *Segment Size*, yang memberikan jumlah data yang dapat diterima oleh pengirim *option* ini. *Padding* biasanya berisi nol untuk memastikan bahwa TCP *header* menjadi lengkap (32-*bit word*).

TCP mempunyai 6 fungsi utama, yaitu *Basic Data Transfer*, *Reliability*, *Flow Control*, *Multiplexing*, *Connections*, dan *Precedence/Security*.

1. *Basic Data Transfer*

   Sebuah modul TCP mengirimkan beberapa *octet* yang disebut segmen dari *host* yang satu ke *host* yang lain. Hubungan antara TCP dan sebuah *local process* adalah *port*, yaitu suatu mekanisme yang memberikan kemampuan pada proses untuk memanggil TCP dan sebaliknya memungkinkan TCP untuk mengirimkan aliran data ke proses yang tepat. Untuk menghasilkan hubungan yang lengkap, IP *address host* ditambahkan ke nomor *port*. Kombinasi IP *address* dan nomor *port* disebut *socket*. Hubungan antara 2 *socket* memungkinkan jalur komunikasi 2 arah (*full-duplex*) antara 2 proses di ujungnya.

2. *Reliability*

Reliability dipenuhi dengan memberikan *sequence number* pada setiap segmen yang ditransmisikan, juga digunakan *acknowledgement*. *Sequence Number* adalah nomor urut yang diberikan pada *octet* pertama dari data pada segmen. Saat SYN *flag* diset, *sequence number* menunjukkan *Initial Sequence Number* (ISN). *Octet* pertama dari *field* data dikirim dengan menggunakan *sequence number* berikutnya, yaitu ISN +1. *Sequence Number* menjamin urutan dari data. *Acknowledgement Number* melaporkan penerimaan data. Nilai pada *acknowledgement* adalah *octet* berikutnya (sama dengan *sequence number* berikutnya) yang diharapkan dari salah satu *host* dari hubungan. Saat *field Acknowledgement* sedang digunakan selama hubungan, *flag* ACK diset. *Acknowledgement* digunakan untuk mengetahui apakah *host* penerima sudah menerima data atau belum. Untuk setiap segmen yang telah diterima, *host* penerima akan mengembalikan sebuah *acknowledgement* dalam waktu tertentu. Jika *acknowledgement* tidak diterima, maka oleh *host* pengirim, segmen tersebut akan dikirim ulang. Pada *host* penerima jika segmen yang diterima rusak, maka segmen tersebut akan dibuang (*discard*) dan karena tidak ada *acknowledgement* dari *host* penerima untuk segmen tersebut, *host* pengirim akan mengirim ulang segmen tersebut. Proses ini disebut *Positive Acknowledgement with Retransmission* (PAR).

3. *Flow Control*

   Mekanisme *flow control* dilakukan dengan adanya *sequence number*, *acknowledgement number*, dan *window* pada TCP *header*. *Window* ini dikirimkan dari *host* penerima ke *host* pengirim yang menunjukkan jumlah *octet* yang siap diterima untuk satu kali pengiriman. *Window* akan mengontrol aliran data dari pengirim ke penerima. Bila *Window* = 0 berarti jalur data diputuskan oleh penerima.

4. *Multiplexing*

   Pada sebuah *host* dapat terjadi lebih dari 1 proses secara simultan, karena setiap proses memiliki *port*-nya masing-masing, misalnya Telnet (*port* = 23) dan FTP (*port* =21).

5. *Connections*

   Karena TCP adalah *connection-oriented protocol*, maka sebuah hubungan logika, yang disebut sebagai *virtual circuit* harus dibentuk antara 2 *host* sebelum data dapat dikirimkan. Proses pembentukan hubungan ini disebut *three-way handshake* dan telah dijelaskan sebelumnya. Penanganan hubungan dapat dijelaskan secara singkat sebagai berikut:

   a. TCP menerima aliran data (*byte-stream*) dari proses lapisan atas.
   b. TCP dapat memecah *byte-stream* menjadi segmen-segmen yang memenuhi ukuran maksimum *datagram* pada IP.

c. IP dapat melakukan fragmentasi segmen sambil mempersiapkan *datagram* yang ukurannya disesuaikan untuk memenuhi MTU suatu jaringan.
d. Protokol *Network Interface layer* mengirimkan *datagram* dalam bentuk bit.
e. Protokol *Network Interface layer* pada *host* penerima menyusun ulang datagram dari bit-bit.
f. IP menerima *datagram* dan menyusun ulang segmen.
g. TCP menerima data dalam bentuk segmen dan memberikannya pada lapisan atas dalam bentuk aliran data.

Pada RFC 793 mengenai TCP dibahas mengenai TCP *Primitives*. *Primitive* adalah sebuah kejadian (*event*) yang meminta atau merespon suatu tindakan dari suatu *host*/TCP *interface*. TCP *primitive* yang utama adalah OPEN, SEND, RECEIVE, CLOSE, STATUS, dan ABORT. Perintah OPEN dari proses lapisan atas akan memicu terjadinya *three-way handshake*. Untuk mengatur hubungan, modul TCP membentuk suatu *record* yang disebut *Transmission Control Block* (TCB). TCB menyimpan sejumlah variabel termasuk nomor *socket* local dan *remote*, *security* dan *precedence* dari hubungan, dan pointer untuk mengidentifikasi keluar masuknya aliran data. Umumnya TCP membuat antrian untuk data yang akan dikirim, tetapi proses dapat meminta pengiriman seketika dengan men-set *flag* PUSH, yang memaksa TCP untuk mengirim data yang sedang diantrikan saat ini ke *host* penerima. Hal ini juga memaksa *host* penerima untuk membuang *buffer* yang ada saat ini dan mengirim data

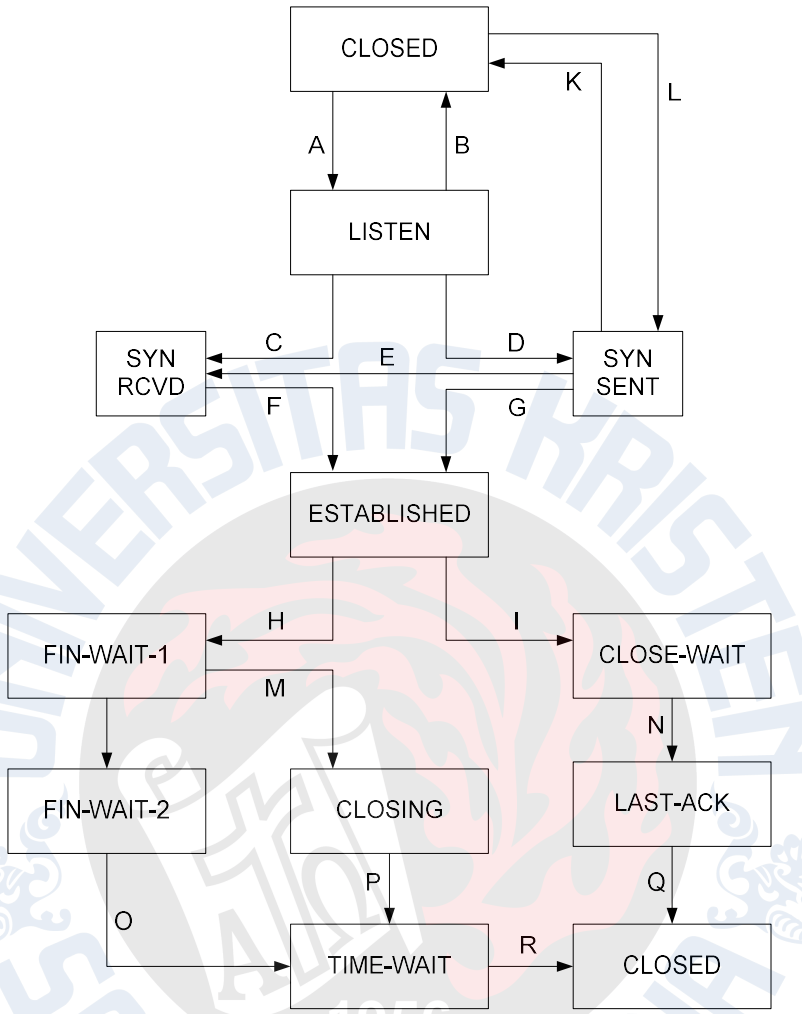
yang diterimanya saat ini ke proses lapisan atas. Penutupan hubungan TCP dapat dilakukan oleh *host* pengirim, *host* penerima, maupun keduanya secara simultan. Prosedur untuk memutuskan hubungan sama untuk ketiga kasus di atas. Bila sebuah *host* telah selesai melakukan tugasnya, maka ia akan mengirimkan sebuah TCP segmen dengan *flag* FIN diset, saat FIN ini di-*acknowledge* maka hubungan ditutup.

6. *Precedence*/*Security*
   IP *header* menyediakan *field Type of Service* dan *Security Option*, yang dapat digunakan oleh TCP untuk mengimplementasikan *precedence* dan *security*.

TCP *Connection State Diagram*

Operasi TCP berjalan dari satu *state* ke *state* yang lain sebagai respons terhadap kejadian (*events*) termasuk *user calls*, TCP segmen yang datang dan *timeouts*. *User calls* adalah perintah yang diberikan kepada TCP *interface*. Yang termasuk *user calls* adalah OPEN, RECEIVE, CLOSE, ABORT dan STATUS. Saat sebuah TCP segmen diterima, akan diperiksa apakah ada *flag* yang diset atau tidak. *Timeout* termasuk USER TIMEOUT, RETRANSMISSION TIMEOUT, dan TIME-WAIT TIMEOUT. *State* pertama yang dimasuki oleh proses TCP adalah CLOSED, yang mengindikasikan tidak ada hubungan (*connection*). *Passive* OPEN membuat sebuah *Transmission Control Block* (TCB), yang akan memindahkan proses ke *state* LISTEN. Proses kemudian akan menunggu adanya hubungan dari *remote* TCP dan *port*. Jika SYN diterima, sebuah (SYN,ACK) dikirimkan sebagai respons. Bila

sebuah ACK lagi diterima, berarti proses mencapai ESTABLISHED *state*. Saat pengiriman data hampir selesai, terjadi perintah CLOSE dan FIN dikirimkan, menyebabkan *state* berpindah ke FIN-WAIT-1 dan setelah itu ke FIN-WAIT-2 atau CLOSING. Setelah menunggu untuk memastikan bahwa *remote* TCP telah menerima ACK untuk FIN-nya, TCB dihapus dan hubungan mencapai *state* CLOSED. Dengan cara yang sama, sebuah FIN yang diterima dari ujung yang lain, membawa proses menuju *state* CLOSE-WAIT. Saat ACK dari FIN telah diterima, maka hubungan mencapai *state* CLOSED. Gambar dari TCP *Connection State diagram* ditunjukkan pada Gambar 4.4.



Gambar 4.4 TCP *Connection State Diagram*

Keterangan :

Causing Event / Resulting Action

MSL = *Maximum Segmant Lifetime*

A = Passive OPEN / create TCB

B = CLOSE / delete TCB

C = rcv SYN / snd SYN,ACK

D = SEND / snd SYN

E = rcv SYN / snd SYN,ACK

F = rcv ACK of SYN / x (no resulting action)

G = rcv SYN,ACK / snd ACK

H = CLOSE / snd FIN

I = rcv FIN / snd ACK

J = CLOSE / snd FIN

K = CLOSE / delete TCB

L = Active OPEN / Create TCB, snd SYN

M = rcv FIN / snd ACK

N = CLOSE / snd FIN

O = rcv FIN / snd ACK

P = rcv ACK of FIN / x (*no resulting action*)

Q = rcv ACK of FIN / x (*no resulting action*)

R = Timeout = 2 MSL / delete TCB

- LISTEN = menunggu sebuah *connection request* dari TCP *remote* dan *port*.
- SYN-SENT = menunggu sebuah permintaan hubungan yang cocok setelah mengirim sebuah *connection request*.
- SYN-RECEIVED = menunggu sebuah ACK setelah menerima sebuah *connection request* dan mengirim *connection request*.
- ESTABLISHED = menunjukkan sebuah *open connection*, data dapat dipertukarkan.
- FIN-WAIT-1 = menunggu *connection termination request* dari *remote* TCP, atau ACK dari *connection termination request* yang telah dikirim.
- FIN-WAIT-2 = menunggu *connection termination request* dari *remote* TCP.
- CLOSE-WAIT = menunggu *connection termination request* dari *local user*.

- CLOSING = menunggu *connection termination acknowledgement request* dari *remote* TCP.
- LAST-ACK = menunggu ACK dari *connection termination request* yang telah dikirim ke *remote* TCP.
- TIME-WAIT = menunggu agar *remote* TCP mempunyai waktu yang cukup untuk menerima ACK dari *connection termination request*-nya.
- CLOSED = hubungan tertutup.

BAB 5
APPLICATION LAYER

Application Layer merupakan *layer* teratas dari arsitektur TCP/IP. Pada *layer* inilah seluruh aplikasi yang menggunakan jaringan dijalankan. Aplikasi yang dijalankan pada *application layer* ini ada yang menggunakan TCP dan ada juga yang UDP. Yang menggunakan TCP misalnya: Telnet, FTP, dan SMTP, dan yang menggunakan UDP misalnya: TFTP, NFS, dan SNMP.

5.1 *File Transfer Protocol* (FTP)

FTP adalah protokol sekaligus program yang dapat digunakan untuk melakukan operasi file dasar pada *remote host* dan untuk mentransfer file antar *host*. Sebagai sebuah program, FTP dapat dioperasikan *user* untuk melakukan perintah *file* secara manual. FTP juga dapat digunakan sebagai protokol oleh aplikasi yang membutuhkan pelayanan *file*. FTP adalah aplikasi yang aman dan *reliable*. *User* yang mengakses host melalui FTP harus melewati *authentication login* yang dapat diamankan dengan *username* dan *password*. FTP terdiri dari komponen *client* dan *server*. Agar *file* sistem sebuah *host* dapat diakses oleh *user*, maka *host* harus menjalankan aplikasi yang berfungsi sebagai FTP *server*. *User* yang ingin mengakses FTP *server* harus menjalankan *software* FTP *client*. Saat FTP *client* membuka koneksi ke FTP *server*, diciptakan sebuah jalur *logical* (*virtual circuit* yang didefinisikan

oleh 2 *socket*) antara *host client* dan *server*. Jalur ini memungkinkan komponen FTP untuk saling berkomunikasi.

5.2 *Trivial File Transfer Protocol* (TFTP)

TFTP merupakan alternatif dari FTP yang digunakan bila pada proses transfer *file* tidak dibutuhkan *authentication* dan *login*. TFTP tidak seperti FTP, TFTP tidak membutuhkan *transport* yang *reliable*, sehingga TFTP ini berjalan di atas UDP.

5.3 *Network File System* (NFS)

NFS dikembangkan oleh Sun Microsystems Inc. NFS ini menyediakan *file transfer* yang transparan dan menyatu. *Server* NFS dapat mengekspor bagian dari direktori *tree*-nya untuk digunakan oleh NFS *client*. *Client* dapat melakukan *mount* pada direktori yang diekspor tersebut seolah-olah direktori itu adalah bagian dari *file* sistem *client* sendiri.

5.4 *Simple Network Management Protocol* (SNMP)

SNMP adalah keluarga protokol yang memenuhi strategi manajemen internet:

- SNMP adalah protokol yang memungkinkan *network management station* untuk berkomunikasi dengan peralatan yang akan di-*manage*.
- *Management Information Base* (MIB) adalah *database* yang menyimpan informasi tentang manajemen sistem.

- *Structure and Identification of Management Information* (SMI), menjelaskan bagaimana pandangan MIB terhadap obyek.

SNMP terdiri dari 2 peralatan, yaitu :

- *Network Management Station*, yang berfungsi sebagai penyimpanan pusat untuk pemgumpulan dan analisis dari data dan manajeman jaringan.
- Peralatan yang di-*manage* menjalankan SNMP *agent*, yaitu proses *background* yang memonitor peralatan tersebut dan mengkomunikasikannya ke *network managemant station*. Seringkali, peralatan yang tidak mendukung SNMP *agent* dapat dimonitor melalui peralatan *proxy*.

Salah satu tujuan dari manajemen jaringan adalah menyimpan sejarah unjuk kerja jaringan yang berisi *snapshot* (informasi sekilas) dari berbagai karakteristik pada jaringan. Tujuan lain adalah agar manajer dapat dengan cepat mengetahui adanya perubahan pada jaringan. Peralatan yang di-*manage* dapat dikonfigurasikan untuk mengirimkan peringatan (*alert* atau *trap*) bila terjadi kondisi tertentu. Manajer jaringan dapat menentukan ambang batas untuk karakteristik unjuk kerja jaringan tertentu. Saat nilai ambang ini terlampaui, *agent* yang terdapat pada peralatan yang di-*manage* langsung akan mengirim *trap* ke *management station*. Secara prinsip, *network management station* dan *agent* dapat melakukan tugas berikut:

- *Network management* dapat mem-*poll agent* untuk meminta informasi jaringan.

- *Network management station* dapat meng-*update*, menambah atau menghapus entri pada *database* milik *agent*. Tabel *routing* pada *router* contohnya, dapat di-*update* dari *console management*.
- *Network management station* dapat mengatur nilai ambang untuk *trap*.
- *Agent* pada peralatan yang di-*manage* dapat mengirimkan *trap* ke *network managemant station*.

MIB adalah kumpulan objek (jenis dari entiti jaringan) yang termasuk ke dalam *database management* jaringan. Spesifikasi MIB menjelaskan sifat dari objek, sementara SMI menjelaskan perwujudan dari objek.

Standar SNMP saat ini, versi 1 dapat melakukan 4 operasi:

- *Get*. Mengambil satu objek dari MIB.
- *Get next*. Menjelajah tabel pada MIB.
- *Set*. Memanipulasi objek MIB.
- *Trap*. Melaporkan alarm.

5.5 *Simple Mail Transfer Protocol* (SMTP)

SMTP merupakan protokol yang digunakan untuk mengirimkan *Electronic mail* (*e-mail*) antar *host*. *Host* yang mendukung *e-mail* menggunakan *mail transfer unit* (MTU) untuk menangani prosesnya. Contoh MTU yang populer di lingkungan UNIX adalah *sendmail*. Secaraa umum MTU mempunyai dua fungsi:

- Mengirimkan pesan dan menerima pesan dari *mail server* lain.

- Memberikan *interface* yang memungkinkan aplikasi *user* untuk mengakses sistem *mail*.

User berhadapan langsung dengan MTU menggunakan satu dari banyak *user agents* (UA) yang tersedia. UA menggunakan protokol mail untuk berkomunikasi dengan MTU. Salah satu protokol yang sering digunakan adalah *Post Office Protocol* versi 3 (POP3). Ada juga *Multipurpose Internet Mail Extensions* (MIME) yang mendukung transfer dari pesan non teks lewat SMTP.

5.6 *Telnet*

Telnet merupakan fasilitas untuk mengakses terminal secara *remote* melalui suatu jaringan. Telnet juga seperti FTP berbasis pada proses *client* dan *server*. Proses *telnet server* pada *host remote* mengelola terminal bayangan (*virtual terminal*), yaitu sebuah gambaran yang berjalan sepenuhnya pada *software* dari terminal yang dapat berinteraksi dengan *host remote*. *User* memulai *session telnet* dengan menjalankan *software telnet client* dan *logon* ke *telnet server*. *Telnet* berbasis pada emulasi dari beberapa model terminal bermode teks. Yang paling umum adalah Digital VT220, VT100/VT102, atau VT52.

DAFTAR PUSTAKA

Microsoft Official Curriculum, *Microsoft TCP/IP*. Microsoft Corp.

Microsoft Official Curriculum, *Networking Essentials*. Microsoft Corp.

Troubleshooting TCP/IP, Analyzing the Protocol of Internet

Douglas E. Comer, *Internetworking with TCP/IP Second Edition, Pronciples, Protocol, and Architecture*, Prentice-Hall Inc., 1991