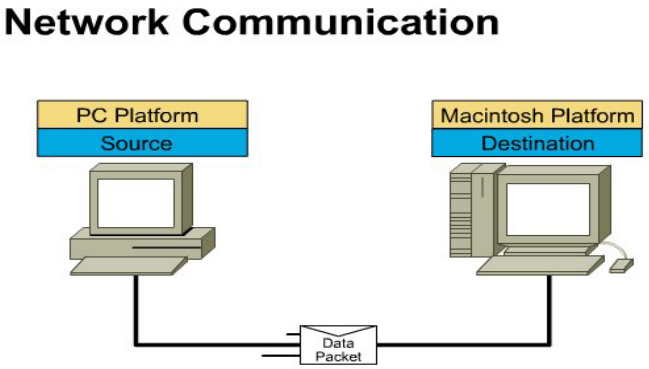
# BAB 2

# LANDASAN TEORI

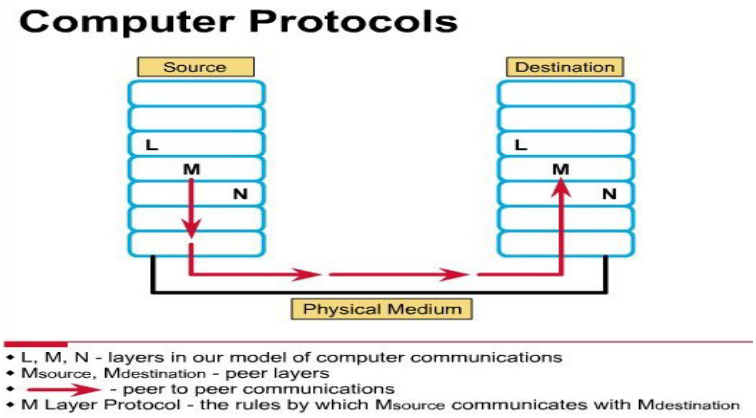
## 2.1 Network Communication




**Gambar 2.1. Ilustrasi Pengiriman Paket Data (Krzenski,2013).**

Seperti yang diilustrasikan pada gambar 1.2, informasi yang dibawa pada suatujaringan disebut sebagai *data* , *packet*, atau *data packet*. Sebuah *data packet* secara logika merupakan sebuah unit grup dari informasi yang berpindah diantara sistem-sistem komputer. Di dalam sebuah paket data terdapat informasi mengenai tempat asal pengirim (*source*) dengan tujuan untuk membuat hubungan komunikasi dengan tujuan penerima (*destination*) lebih *reliable*. Alamat pengirim (*source address*) merupakan identitas si pengirim yang mengirimkan paket. Alamat tujuan (*destination address*) merupakan identitas dari sebuah komputer yang berhasil menerima paket yang dikirimkan. Jika satu komputer (host A) ingin mengirimkan data ke komputer yang lainnya (host B), maka data tersebut pertama kali harus mengalami proses yang disebut dengan enkapsulasi data (Krzenski, 2013).

Untuk memungkinkan paket-paket data dapat dikirimkan dari *source address* menuju *destination address* pada suatu jaringan komputer, sangatlah penting apabila semua perangkat-perangkat yang ada pada suatu jaringan berbicara atau berkomunikasi dengan menggunakan bahasa yang sama yaitu *protocol*. Protokol merupakan seperangkat aturan (*rules*) yang membuat komunikasi pada suatu jaringan menjadi lebih efisien (Krzenski, 2013).




**Gambar 2.2. Ilustrasi Protokol Komunikasi Data (Krzenski,2013)**

Secara teknis definisi dari *data communications protocol* adalah sekumpulan aturan (*rules*), atau kesepakatan (*agreement*), yang menentukan format dan transmisi dari data. Dalam gambar 1.3 dijelaskan, bahwa *layer* M dari suatu komputer berkomunikasi dengan *layer* M yang berada pada komputer lainnya. Aturan-aturan dan kesepakatan yang digunakan pada komunikasi tersebut dikenal dengan skema *layer* dan *protocol* .

*OSI Preference Model* merupakan model utama (dasar) dari komunikasi jaringan. Walaupun ada bentuk permodelan lain yang telah dikembangkan , kebanyakan *vendor* jaringan, pada hari ini mengadopsi model ini pada produk-produk yang dikembangkan, terutama ketika memberikan pelatihan mengenai tata cara penggunaan produk mereka. Dikarenakan model ini cocok dalam memberikan pengetahuan kepada *user* mengenai pengiriman dan penerimaan data pada jaringan.

Model OSI memungkinkan kita untuk dapat melihat fungsi-fungsi yang ada pada jaringan di setiap lapisan (*layer*). Lebih penting lagi, model OSI merupakan *framework* yang dapat digunakan untuk memahami bagaimana informasi di transmisikan melalui jaringan (Gurlen, K, 2014). Di lain hal, kita dapat menggunakan OSI *reference model* untuk melakukan visualisasi bagaimana informasi, atau paket-paket data, berjalan dari program aplikasi (seperti *spreadsheets*, *documents*, dan lain sebagainya), melalui media transmisi jaringan (contoh : kabel), ke program aplikasi lainnya yang berlokasi di komputer (*host*) yang lain di satu jaringan atau bahkan jika pengirim dan penerima memiliki perbedaan jenis media jaringan.

### 2.1.1 Enkapsulasi Data

Proses enkapsulasi menyesuaikan bentuk data yang sesuai dengan protokol informasi tertentu sebelum ditransmisikan melalui jaringan. Jadi, paket data yang bergerak melalui lapisan-lapisan yang ada pada OSI model, akan memperoleh informasi tambahan seperti *headers*, *tailers*, maupun informasi lainnya. *Header* data yang dimaksud disini adalah informasi pengalamatan yang telah ditambahkan.

Lapisan *application, presentation,* maupun *session* dikenal sebagai *upper layer* atau lapisan atau *layer* pada tingkatan atas dan diimplementasikan pada sebuah perangkat lunak (*software*). Lapisan *transport* dan *network* lebih berfokus pada pengiriman dan *routing* paket-paket data menuju ke tempat tujuan (*destination*). Lapisan *data link* diimplementasikan di perangkat lunak maupun perangkat keras. Dan lapisan fisik (*physical layer*) diimplementasikan hanya pada perangkat keras saja. Dua *layer* yang terakhir dapat menentukan spesifikasi jaringan LAN maupun WAN (Gurlen, K, 2014).

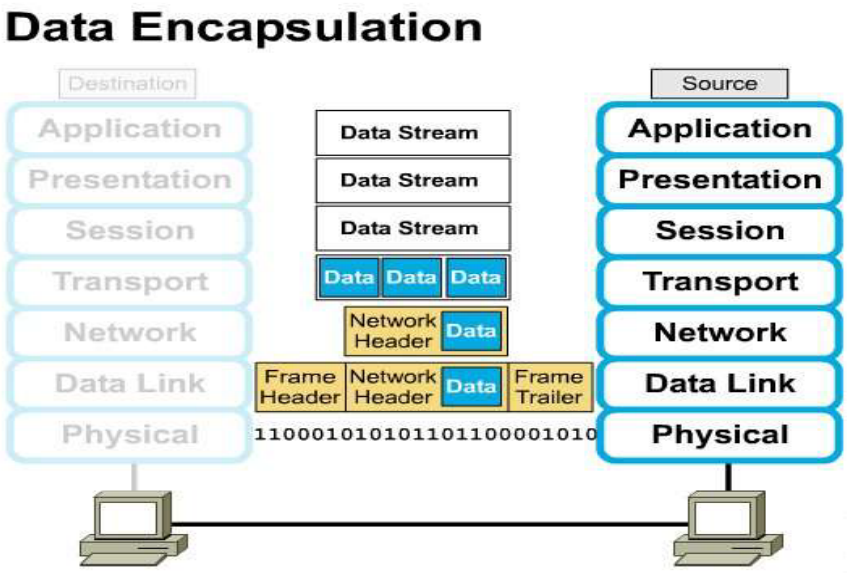
Berikut merupakan sebuah detail deskripsi singkat mengenai transmisi data dari satu komputer *host* ke komputer *host* yang lainnya.

- Lapisan (*layer*) *application, presentation,* dan *session* menerima input dari *user* dan dikonversikan ke dalam data.
- *Transport layer* menambahkan sebuah *segment headers* untuk mengkonversikan data menjadi *segment*.
- *Network layer* menambahkan sebuah *network header* dan mengkonversikan *segment* menjadi *packet* atupun *datagram*.

- *Data link layer* menambahkan sebuah *frame header* dan *trailer* pada data dan mengkonversikan *packet* menjadi *frame*.
- MAC *sublayer layer* mengkonversikan *frame* menjadi sekumpulan bit-bit yang digunakan oleh *physical layer* pada sebuah media transmisi kabel.

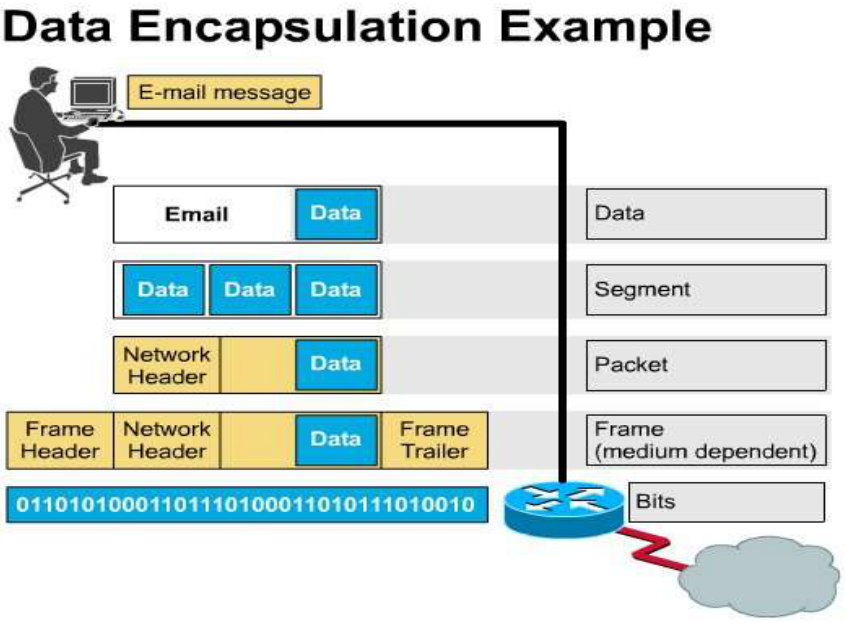
Kelima proses yang dilalui data dari level aplikasi hingga ke level fisik tersebut itulah dinamakan sebagai proses enkapsulasi data. Dan pada saat *bits stream* tiba di tempat tujuan (*destination*), *physical layer* mengambilnya dari media transmisi kabel dan dikonversikan kembali menjadi *frame*, dan proses tersebut trus berlanjut hingga kembali ke bentuk data semula dan ditampilkan ke dalam antarmuka aplikasi, proses inilah yang disebut dengan dekapsulasi.

Untuk melihat bagaimana semua proses enkapsulasi itu berjalan, dapat dilihat dalam ilustrasi yang ada pada gambar 2.3.




**Gambar 2.3 : Proses enkapsulasi data** (Krzenski, 2013)

Ketika suatu data dikirimkan dari *source host*, seperti yang dijelaskan pada gambar 2.3, data tersebut akan bergerakdimulai dari lapisan aplikasi menuju lapisan lain dibawahnya. Seperti yang dapat dilihat, proses *packaging* dan aliran (*flow*) data yang dipertukarkan seiring dengan perubahan dari jaringan dalam memberikan kinerja *services* nya kepada *end user*. Contoh proses enkapsulasi ditunjukkan pada gambar 2.4 berikut ini, dalam kasus *user* mengirimkan email dari aplikasi email (*application layer*).




**Gambar 2.4 : Ilustrasi enkapsulasi data pada proses pengiriman *email***(Krzenski,2013).

Seperti yang diilustrasikan pada gambar 2.4, jaringan harus melakukan lima tahapan konversi dalam proses enkapsulasi, yaitu :

- Membangun data.
  Ketika *user* mengirimkan emailnya, karakter-karakter *alphanumeric* dikonversikan ke dalam bentuk data yang dapat ditransmisikan antar jaringan.
- Membuat paket data untuk transportasi *end-to-end*.
  Data dipaketkan untuk keperluan transportasi antar jaringan. Dengan membentuk segmen-segmen, fungsi transportasi meyakinkan bahwa *host* pesan yang ada pada di kedua sisi (*sender* maupun *receiver*) sistem *email* dapat berkomunikasi dengan baik secara *reliable*.
- Menambahkan *network address* pada *header* data
  Data dibentuk ke dalam paket atau datagram yang mengandung informasi tentang *network header* dengan alamat logikal sumber (*source*) dan tujuan (*destination*). *Address* yang ada pada data ini membantu perangkat jaringan untuk mengirimkan paket-paket melewati jaringan berdasarkan jalan yang sudah dipilih.

- Menambahkan *local address* pada header *data link.*

  Setiap perangkat jaringan harus mengubah paket data menjadi *frame*. *Frame* dapat membuat koneksi secara langsung ke perangkat jaringan lainnya pada tautan (*link*) yang sama. Setiap perangkat yang berada pada jalur (*path*) yang sama memerlukan proses *framing* untuk dapat terkoneksi dengan perangkat lainnya.

- Konversi *frame* menjadi bit-bit data.

  *Frame* haruslah dikonversi ke dalam bentuk biner (0 dan 1) untuk dapat ditransmisikan pada sebuah medium jaringan (contoh: kabel). Fungsi *clocking* memungkinkan bagi perangkat jaringan untuk membedakan bit-bit data yang ditransmisikannya melalui sebuah medium.

## 2.2 Java Packet Capture (JPCAP)

JPCAP merupakan suatu *java library* yang berbasis libpcap/Winpcap, diimplementasikan dengan menggunakan bahasa pemrograman C dan Java yang digunakan untuk melakukan *capturing* dan pengiriman paket data pada suatu jaringan (Ansari M.U,2012).Dengan menggunakan *library* JPCAP kita dapat mengembangkan sebuah aplikasi yang dapat menangkap dan menganalisis paket data dari antarmuka jaringan (*network interface*) kita.

Data yang di *capture* dapat di visualisasikan untuk keperluan analisis buat para sistem administrator. JPCAP dapat melakukan *capturing* pada beberapa paket-paket data yaitu, Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, dan ICMPv4 (Ansari, 2012).

JPCAP berisi sekumpulan kelas-kelas yang dapat memberikan beberapa antarmuka dan fungsionalitas untuk melakukan capturing paket data jaringan, serta ketersedian *tools* yang dapat digunakan oleh para pengembang untuk memvisualisasikan *traffic network* secara *real time* (Krzesinki, 2013).

JPCAP menyembunyikan beberapa detail *low-level* paket jaringan yang di *capture* dengan melakukan abstraksi berbagai macam jenis protokol jaringan dan tipe-tipe paket data menjadi kelas-kelas pada java. Dalam penerapannya, secara internal JPCAP melakukan *bindings* ke dalam sistem *library* libpcap melalui perantara JNI (*Java Native Interface*) (Krzesinki, 2013).

Menurut (Dhillon K, 2012), JPCAP merupakan *library open source* yang dapat digunakan untuk menangkap (*capturing*) dan mengirimkan data paket-paket jaringan

melalui aplikasi berbasis java. Beberapa fasilitas yang tersedia pada sistem *library* JPCAP, yaitu :

- Dapat secara *real time* menangkap (*capturing*) paket data langsung dari kabel atau media transmisi jaringan.
- Dapat menyimpan data mentah jaringan ke dalam file secara *offline*, dan dapat juga membaca paket data mentah jaringan yang tersimpan pada suatu file secara *offline.*
- Secara otomatis dapat mengidentifikasi jenis paket data dan dapat di *generate* ke dalam suatu *Java Object*. Sebagai contoh, Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, dan ICMPv4 paket data.
- Paket-paket data yang di *capture* dapat di filter sesuai dengan spesifikasi yang ditentukan oleh *user* sebelum menampilkannya pada antarmuka aplikasi.
- Selain dapat di *filter*, user dapat menggunakan fitur pada JPCAP untuk mengirimkan *raw packet data*.

JPCAP memungkinkan *use r*menggunakan sebuah model *event* untuk memproses paket data. Dalam melakukan proses *capturing* paket data jaringan, pengguna dapat mengatur JPCAP untuk melakukan proses *listening* terhadap jaringan mana yang akan di pantau. Setelah memilih perangkat jaringan yang ada, pengguna dapat membuka perangkat tersebut untuk melakukan *listening*. Paket-paket data yang lewat akan di *capture* dan disimpan pada lokasi atau format file tertentu. JPCAP juga menyediakan fungsi untuk melakukan proses *filtering* terhadap suatu protokol jaringan tertentu (Das Vipin, *et al*, 2010).

### 2.2.1 Windows Packet Capture (WinPcap)

WinPcap merupakan suatu *tools* standar industri yang digunakan untuk melakukan akses pada *link-layer network* di lingkungan sistem operasi Windows (Dhillon *et al.* 2012). Dengan menggunakan WinPcap, dapat membuat aplikasi *capturing* dan mentransmisikan paket data jaringan dengan melakukan *bypass* pada tumpukan (*stack*) struktur protokol.

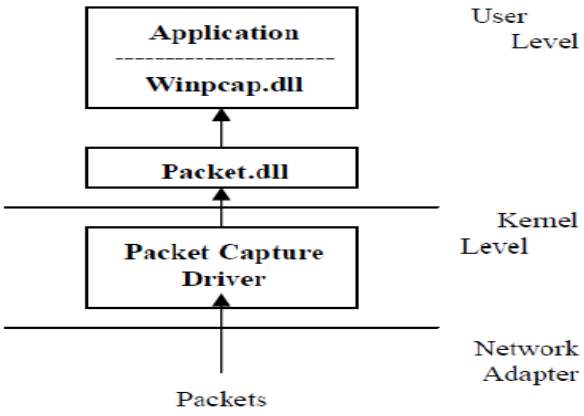
WinPcap dilengkapi beberapa fitur-fitur tambahan seperti, *filtering* paket data pada level kernel, mendukung fitur statistik jaringan untuk keperluan *remote* jaringan. WinPcap terdiri dari suatu perangkat lunak *driver* yang dapat membuat sistem operasi

memberikan fasilitas atau layanan untuk mengakses jaringan level rendah, dan adanya *library* yang tersedia untuk memudahkan akses pada lapisan jaringan tingkat rendah. *Library* ini juga mengandung beberapa versi Windows yang dapat dikenali oleh sistem *library* libpcap Unix API (Dhillon, *et.al*, 2012).

WinPcap merupakan *engine* yang digunakan untuk melakukan *capturing* dan *filtering* paket data dari berbagai macam *tools* jaringan yang bersifat *open source* maupun yang bersifat komersial. Termasuk dintaranya seperti *Protocol Analyzer, network monitors,* dan *network intrusion detection system* (Dhillon *et al.* 2012).

Beberapa perangkat lunak jaringan yang juga menggunakan WinPcap sebagai *engine* yaitu, Wireshark, Nmap, Snort dan Nop, dimana aplikasi tersebut digunakan secara luas oleh berbagai komunitas yang berkecimpung di dunia *networking*.

Arsitektur perangkat lunak (*software*) untuk *packet capturing* dan analisa jaringan pada platform Win32, telah memungkinkan sistem operasi untuk memiliki kemampuan untuk secara efisien melakukan *capturing traffic* jaringan dari berbagai jenis jaringan dengan memanfaatkan *network adapter* dari mesin.




**Gambar 2.5 : *Logical structure of WinPcap***

Dalam melakukan proses *capturing* data dari sebuah jaringan, aplikasi *capturing* harus secara langsung berkomunikasi dengan perangkat yang disebut *network adapter*. Sehingga sistem operasi harus memberikan sekumpulan *primitives* untuk melakukan *capturing* dengan tampilan *view* yang terhubung dengan *adapter*. Maksud dan tujuan dari penggunaan *primitives* ini adalah untuk melakukan *capturing* jaringan secara transparan dari sudut pandang pengguna dan mentransfer paket-paket tersebut ke aplikasi yang membutuhkan (Huysamen *et al* , 2013).

Bagian dari kernel tersebut harus secara cepat dan efisien dalam melakukan *capturing* semua paket-paket data secara *real time* tanpa harus kehilangan informasi yangpenting.Arsitektur secara logikal dari winpcap merupakan arsitektur hierarkial dengan tingkatan 3 level (mulai dari *network adapter* hingga menuju aplikasi), yang ditunjukkan pada gambar 2.5. *Packet Capture Driver* merupakan level terendah dari aplikasi yang menjalankan proses *capturing*. Berfungsi dalam penyedian daftar kumpulan fungsi-fungsi yang digunakan untuk melakukan proses *read* / *write* data dari jaringan pada level lapisan *data-link* (Dhillon N.K, 2013).

Jpcap.dll merupakan *dynamic link library* yang memisahkan aplikasi dengan *driver* untuk membentuk sebuah *interface* sistem *capturing* yang *independent* (berdiri sendiri). Memungkinkan untuk para pengguna untuk menjalankan aplikasi pada sistem operasi Windows yang berbeda tanpa harus di *compile* ulang. Direpresentasikan dalam bentuk sekumpulan fungsi-fungsi API yang berfungsi untuk mengakses *driver* secara langsung.

WinPCap.dll (*third level party*) merupakan *library* statis yang digunakan pada *capturing packet* pada aplikasi. Menggunakan *service* yang diekspor oleh Packet.dll, dan memberikan aplikasi tampilan antarmuka tingkat tertinggi dan *powerful*.

Tampilan antarmuka aplikasi merupakan bagian terpenting pada aplikasi *capturing*. Karena dapat mengatur interaksi dengan pengguna dan menampilkan hasil dari proses *capturing*.

Menurut Das Vipin (2010), Winpcap juga tersusun atas beberapa fungsionalitas, yaitu :

- Implementasi beberapa *low level library* dari beberapa sistem operasi yang terdaftar, yang dapat berkomunikasi langsung dengan *driver* nya.
- Port dari libpcap menggunakan API yang disediakan pada implementasi *low level library*.
- Memiliki *drivers* untuk windows 95/98/ME dan untuk windows keluarga NT yang mana menggunakan NIDS untuk membaca paket data jaringan melalui *network adapter*.

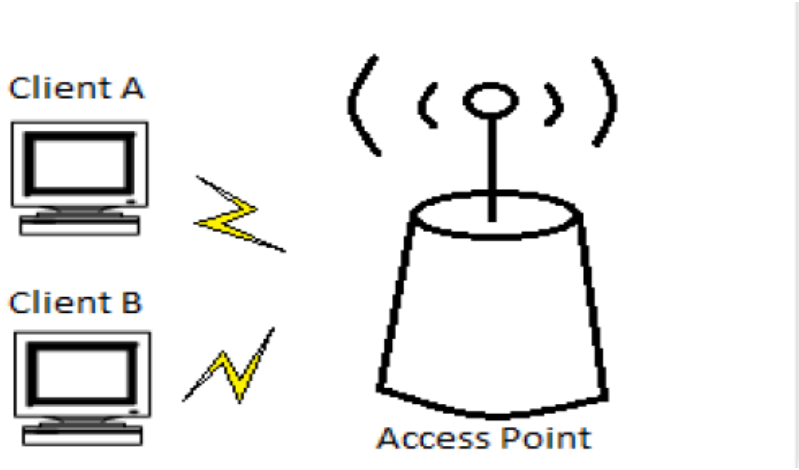
## 2.3 *Campus Wireless* LAN

Untuk dapat menjaga para staff akademik maupun mahasiswa kampus agar tetap *up-to-date* terhadap perkembangan teknologi yang ada, *wireless local area network* (WLAN) ataupun kabel *Ethernet* yang membentuk struktur penggunaan dari sebuah *access point* jaringan nirkabel, merupakan model jaringan yang umum dikembangkan di lingkungan kampus (Tompkins J, 2011).

Perangkat-perangkat dan teknologi *wireless* mengalami fase pekembangan yang begitu cepat, yang otomatis mendapat tempat untuk diadopsi pada teknologi LAN (Ramesh, G, 2010). Disamping menjadi faktor yang signifikan pada jaringan kampus, jaringan *wireless* LAN terus meningkatkan pengembangan layanan-layanan IT di lingkungan kampus (Tompkins, J, 2011).

Permintaan akan jaringan *wireless* yang semakin meningkat, dan juga disebabkan oleh beberapa keuntungan yang diperoleh dengan tidak diperlukannya lagi kabel-kabel untuk terkoneksi ke perangkat-perangkat yang ada. Kelebihan lainnya selain memberikan kontribusi terhadap kemudahan proses instalasi, dan ini juga secara luas meningkatkan baik tingkat fleksibiltas dan portabilitas dari perangkat-perangkat bergerak (*mobile devices*). Jika ada pemindahan lokasi dari *workstation* tidak perlu lagi untuk melakukan hal-hal seperti menambah ataupun mengurangi jumlah kabel. Keuntungan lain yang diperoleh universitas dengan menggunakan jaringan *wireless* dibandingkan dengan jaringan berkabel adalah dapat mengurangi beban *financial cost*.

Untuk dapat terkoneksi dengan jaringan kampus, pengguna jaringan harus menggunakan sebuah *wireless* NIC (network interface card) atau dengan sebuah *wireless adapter*. Dengan itu, pengguna dapat berkomunikasi dengan *station* lain yang terkoneksi ke jaringan (Tompkins, 2010). *Access point* (seperti yang ditunjukkan pada gambar 2.6) juga diperlukan dalam komunikasi secara nirkabel, sejak transmisi frekuensi radio digunakan untuk komunikasi antar perangkat-perangkat jaringan yang ada dan juga pada perangkat komputer.




**Gambar 2.6 : *Access Point* pada arsitektur *wireless*.(Tompkins, 2010)**

Dalam bentuk yang sederhana, *access point* (AP) melakukan *encompass* pada *transceiver* yang pertama, yang beroperasi pada *channel wireless* yang pertama, dan *transceiver* kedua, yang akan beroperasi pada *channel wireless* yang kedua (Tompkins, J, 2010). Sejumlah perangkat yang sedang *roaming* memiliki kemampuan untuk dapat berkomunikasi dengan *channel* yang lainnya.

Ada beberapa versi *wireless* LAN (WLAN) yang digunakan pada umumnya , yaitu versi wireless LAN: 802.11b mempunyai kecepatan transfer data sampai 11Mbps pada frekuensi 2,4 GHz. 802.11a mempunyai kecepatan transfer data sampai 54 Mbps pada frekuensi 5 GHz. 802.11g mempunyai kecepatan transfer data sampai 54 Mbps pada frekuensi 2,4 GHz. Wireless LAN merupakan teknologi yang berhasil dan populer, yang menyebar luar dan diintegrasikan ke dalam laptop sebagai perangkat standar.

### *2.3.1 Campus Network*

Menurut *searchsdn.techtarget.com*, jaringan kampus (*campus network*) adalah jaringan lokal (LAN) proprietary (yang sudah mempunyai hak milik) ataupun sekumpulan jaringan lokal (LAN) yang saling melayani antar korporasi, lembaga pemerintah, universitas atau organisasi yang serupa.

Dalam konteks ini, kampus merupakan sekumpulan bangunan yang letaknya saling berdekatan atau berada di dalam satu lingkup yang sama. Komputer-komputer yang berada di lingkungan kampus sebuah universitas mengelola banyak data yang sifatnya sensitif seperti data tentang mahasiswa, termasuk di dalamnya data akademik, status kesehatan, dan informasi keuangan (Tompkins, 2010). Ini sangatlah penting,

baik secara legal maupun moral, untuk dapat menjamin bahwa data-data penting tersebut tersimpan dengan aman.

Jaringan kampus yang digunakan didesain secara hierarkial yang mana merupakan struktur umum yang diterapkan pada jaringan kampus ataupun jaringan *enterprise*. Jaringan kampus menyediakan topologi modular dari blok-blok bangunan yang berguna untuk memudahkan proses *evolving* jaringan.

### *2.3.2 IEEE 802.3 Ethernet*

Ethernet merupakan salah satu rumpun atau keluarga dari teknologi-teknologi jaringan komputer untuk jaringan *local area networks* (LAN), yang secara komersial diperkenalkan pada tahun 1980. Terstandarisasi pada IEEE 802.3, Ethernet secara besar-besaran menggantikan teknologi-teknologi jaringan berkabel (Dhillon. K, 2012). Sistem komunikasi pada ethernet membagi sebuah *stream* data menjadi individual paket-paket yang disebut dengan *frames*. Setiap *frame* mengandung informasi alamat asal (*source address*) dan alamat tujuan (*destination address*) dan adanya *error-checking* data sehingga data-data yang mengalami kerusakan dapat dideteksi dan ditransmisikan ulang (*re-transmitted*).

Standar-standar tersebut mendefinisikan beberapa varian perkabelan dan *signaling*. Rating data (*data rates*) secara periodik meningkat dari yang aslinya 10 megabits per *second* menjadi 100 gigabits per *second*.

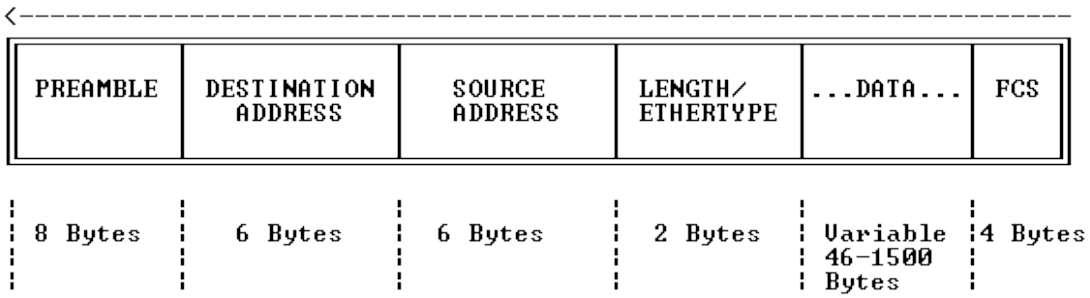
## 2.4 Ethernet Frame

Paket-paket data pada *traffic* ethernet yang dikirimkan pada media transmisi suatu jaringan disebut dengan *frame*. Sebuah *frame* terdiri dari beberapa struktur yang dimulai dari bagian yang disebut dengan *preamble* dan titik pembatas (*delimiter*) bagian awal dari *frame*. Setelah itu diikuti oleh bagian *ethernet header* yang memiliki informasi mengenai MAC *address* dari suatu host sunber (*source*) ke host tujuan (*destination*) (Dhillon *et al*. 2012).

Ukuran maksimum dari sebuah *frame* tersebut dinamakan dengan *Maximum Transmission Unit* (MTU). Ketika sebuah perangkat jaringan menerima sebuah *frame* yang ukuran kapasitasnya lebih besar dari MTU, paket data tersebut di fragmentasikan menjadi bentuk *frame* yang kecil-kecil atau dibuang. Biasanya, *Ethernet* memiliki

ukuran maksimun *frame* sebesar 1500 *bytes*. Paket *Ethernet* yang memiliki besaran lebih dari 1500 *bytes* disebut dengan jumbo *frame*.

Pada bagian lainnya di dalam struktur paket data, ada bagian yang disebut dengan *middle section* yang didalamnya terdapat *data payload* , yang berisi informasi *headers* dari suatu tipe protokol (Internet Protocol) paket data yang dibawa (Huysamen *et al*. 2013). Pada bagian akhir dari suatu *frame* terdapat 32-bit *cyclic redundancy check*, yang berfungsi untuk mendeteksi tanda-tanda terjadinya *corruption* pada data pada saat ditransmisikan.




**Gambar 2.7. Struktur Ethernet *Frame*.**

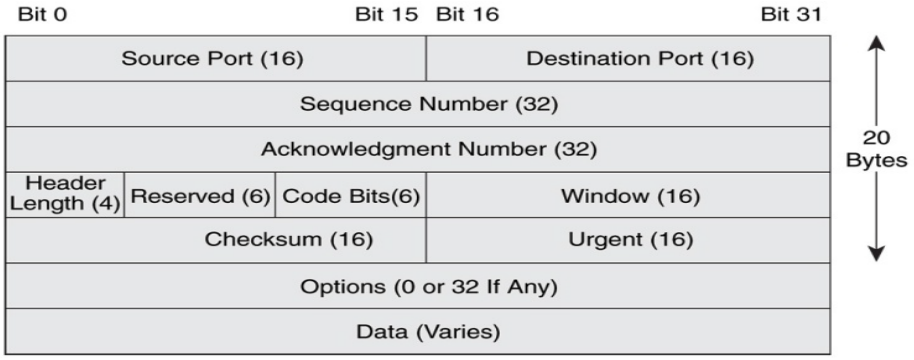
Masalah kinerja (*performance*) pada suatu jaringan akan timbul ketika jumlah MTU dari satu perangkat berbeda dengan jumlah MTU dari perangkat yang lainnya. Menggantibesaran jumlah MTU secara bebas untuk mengoptimalkan kinerja secara tidak langsung akan memberikan efek yang berlawanan yang malah dapat menurunkan kinerja dari suatu jaringan (Huysamen *et al*, 2013).

Semua perangkat yang berada pada lapisan jaringan dua (2), yang mana berarti berada pada LAN atau VLAN yang sama, harus mendukung jumlah ukuran *frame* yang sama. Secara *default*, ukurannnya adalah 1500 *bytes*.

Jika kamu menaikkan jumlah MTU dari satu perangkat jaringan, *switch* harus segera melewatkan ukuran *frame* yang lebih besar (jumlah besaran *frame* yang sama atau yang lebih besar dari ukuran *default*), dan si penerima akhir harus dapat melakukan *handling* terhadap paket yang besar ini. Jika perubahan ini tidak terkoordinasi dengan baik, jaringan anda akan menjadi lambat atau bahkan *loss connection* (Kumar *et al*, 2014).

## 2.5 TCP Segment

TCP merupakan protokol transport yang berorientasi koneksi (*connection oriented*), *reliable*, pencegahan duplikasi pada data, *congestion control*, dan *flow control*. Fungsi tambahan yang ditetapkan oleh TCP adalah pengiriman urutan yang sama, pengiriman yang handal, dan *flow control* (Nugroho, M *et.al* , 2014).




**Gambar 2.8. TCP *segment* (Nugroho, M, et.al, 2014)**

Setiap segmen TCP memiliki 20 *Bytes overhead* di *header*, sedangkan setiap segmen UDP hanya memiliki 8 *bytes overhead*. Pada saat transmisi data menggunakan protokol TCP, data akan dibagi-bagi menjadi bagian-bagian atau disebut juga dengan segmen. Pada setiap transmisi, *maximum transfer unit* (MTU) adalah sebesar 1500 *bytes* untuk *network access* yang menggunakan *ethernet* (Nugroho M, et.al, 2014).

Untuk mendapatkan jumlah data pada segmen TCP, MTU tersebut dikurangi dengan ukuran TCP *header* dan ukuran IP *header*. Perhitungan tersebut dapat diformulasikan sebagai berikut :

$$MSS = MTU - (Fixed\_IPhdrSize + Fixed\_TCPhdrSize) \quad (1)$$

TCP *Maximum Segment Size* (MSS) didefinisikan sebagai jumlah data riil yang terdapat pada segmen TCP. Nilai tetap dari ukuran TCP *header* adalah 20 *bytes* dan untuk IP *header* adalah 20 *bytes* pada IPv4 dan 40 *bytes* pada IPv6. Jadi, misalnya digunakan Ethernet dengan IPv4, MSS atau jumlah data maksimal yang terdapat pada segmen TCP adalah sebesar 1460 *bytes* (Nugroho, M *et.al*, 2014).

## 2.6 Penelitian Terdahulu

Penelitian tentang *monitoring* dan data paket jaringan (*frame*) telah banyak dilakukan sebelumnya, dengan menganalisa paket-paket jaringan yang berguna untuk keperluan

*monitoring* serta untuk mengetahui status baik atau tidaknya suatu aliran data pada jaringan tertentu.

Pada tahun 2012, Dhillon , K.D dan Ansari, M.Z, melakukan penelitian tentang pendiagnosaan status suatu jaringan. Dengan memanfaatkan JPCAP API, mereka terfokus untuk pengolahan beberapa informasi yaitu, *source address* dari suatu *frame, destination address* yang dituju oleh *frame*, serta jenis *protocol* yang digunakan.

Penelitian selanjutnya juga dilakukan oleh Huysamen N.F, dan Krzesinski pada tahun 2013 dari universitas Stellenbosch, Afrika Selatan. Mereka melakukan *monitoring* jaringan serta *cloud system* untuk menganalisa pemakaian *bandwith* yang dilakukan oleh *user* dan mengamati pemakaian *bandwith* yang tidak normal yang dilakukan oleh *user*, sehingga dapat diambil tindakan dengan menormalkan *bandwith user* tersebut.

Pada tahun 2010, Das Vipin, et.al, melakukan perancangan *network intrusion detection system* dengan memanfaatkan algoritma *machine learning* untuk proses identifikasi atau pengenalan beberapa jenis serangan pada jaringan dengan memanfaatkan data paket (*frame*) jaringan. Metode pembelajaran yang mereka gunakan adalah *Rough Set Theory* (RST) dan *Support Vector Machine* (SVM).

Penelitian tentang analisa paket data jaringan pernah dilakukan oleh Kumar, D.G, et.al, dengan merancang sebuah aplikasi *Intrusion Detection System* (IDS) dengan memanfaatkan JPCAP API. Cara kerja IDS yang mereka kembangkan adalah dengan menganalisa paket jaringan untuk mengetahui dan mencegah jenis serangan *denial of service* (Dos).

Pada tahun 2012, Rahmat R.F, et.al, melakukan perancangan aplikasi *Distributed Network Monitoring* dengan SNMP-RMON (DNM SNMP-RMON). Aplikasi tersebut merupakan aplikasi yang memantau jaringan secara terdistribusi dan dibangun berdasarkan protokol *Simple Monitoring Network Protocol* (SNMP) dan menerapkan teknologi *Remote Monitoring* (RMON) versi 1 dan 2. Penerapan aplikasi ini menggunakan Java Eclipse RCP yang akan diintegrasikan dengan iNetmon *Monitoring Suite*, dan dapat beroperasi di berbagai *platform* sistem operasi.

**Tabel 2.1. Tabel Daftar Peneliti Terdahulu**

| No | Nama Peneliti | Tahun | Judul Penelitian |
|---|---|---|---|
| 1 | Dhillon, K.D dan Ansari, M.Z. | 2012 | *Network Traffic Monitoring, Analysis, **and** Reporting Using WINPCAP Tool **With** JPCAP API.* |
| 2 | Huysamen, N.F. dan Krzesinski | 2013 | *A Scalable Network Monitoring and Bandwidth Throttling System for Cloud Computing.* |
| 3 | Das Vipin, et al. | 2010 | *Network Intrusion Detection System Based On Machine LearningAlgorithms.* |
| 4 | Kumar, D.G et.al | 2014 | *Network-based IDS for Distributed Denial of Service Attacks.* |
| 5 | Rahmat, R.F et.al | 2012 | *Penerapan Aplikasi Distributed Network Monitoring with SNMP-RMON.* |
| 6 | Gurlen,K dan Kaur ,M. | 2014 | *Use of Wireless Network to Extract Password Using Packet Analyzer* |