2. TEORI PENUNJANG

2.1. Hypertext Markup Language (HTML)

Hypertext Markup Language(HTML) merupakan *plain-text* (atau lebih dikenal sebagai ASCII) yang digunakan untuk membangun suatu halaman *web* dan dapat dibuat menggunakan beberapa *text editor*, seperti Emacs atau Vi pada UNIX, Simple Text pada Macintosh atau Notepad pada Windows. Dengan HTML, teks, gambar, suara serta *link* dapat digabungkan menjadi satu. HTML adalah bahaa yang sangat tepat dipakai untuk menampilkan informasi pada halaman *web* karena HTML menampilkan informasi dalam bentuk *hypertext.* Di samping itu, HTML mendukung sekumpulan perintah yang dapat digunakan untuk mengatur bagaimana informasi tersebut akan ditampilkan. HTML bukan merupakan bahasa pemrograman, karena seperti tercermin dari namanya, HTML adalah suatu *mark-up language* yang digunakan untuk melakukan *mark-up* atau penandaan terhadap sebuah dokumen teks. Tanda tersebut digunakan untuk menentukan format atau *style* dari teks yang ditandai.

Ciri utama dari file HTML adalah ekstensi .htm, .html atau .shtml. Halaman *web* yang dibuat dengan skrip HTML murni, tanpa ditambah skrip lain atau bahasa lain seperti VBScript, akan bersifat statis. Halaman *web* tersebut hanya dapat dibaca, tidak dapat ditulisi maupun dieksekusi oleh *user* lain. Selain itu, HTML juga memiliki sifat fleksibel karena dapat dikombinasikan dengan skrip atau bahasa pemrograman lainnya. *File* HTML merupakan *file* teks biasa yang mengandung *tag* dengan ekstensi .htm atau .html. *Tag* adalah penandaan yang digunakan dalam HTML. *Tag* HTML ini terdiri dari tanda lebih kecil (<), nama *tag* dan tanda lebih besar (>). Dalam sebuah *file* HTML harus terkandung struktur sebagai berikut:

```
<HTML>
…
</HTML>
```

Tag <HTML> tersebut harus diletakkan pada bagian paling awal *tag* </HTML> harus diletakkan pada bagian paling akhir dan penulisannya tidak bersifat *case sensitive*. Penulisan *tag* dengan huruf kapital hanya untuk mempermudah perbedaan antara teks biasa dengan *tag*. Secara lengkap, *file* HTML biasanya mempunyai bagian *head* dan bagian *body*, jadi struktur lengkapnya adalah sebagai berikut:

```
<HTML>
<HEAD>
…
</HEAD>
<BODY>
…
</BODY>
</HTML>
```

Bagian *head* umumnya berisi informasi mengenai dokumen tersebut, misalnya judul dokumen, versi HTML yang digunakan, dan lain sebagainya. Sedangkan bagian *body* berisi *layout* atau desain halaman *web*.

2.1.1. Bagian Head

Bagian *head* sebenarnya tidak harus ada pada dokumen HTML, tetapi pemakaian *head* yang benar akan meningkatkan kegunaan suatu dokumen HTML. Isi bagian *head*, kecuali judul dokumen, tidak akan terlihat oleh pembaca dokumen tersebut. Elemen-elemen pada bagian *head* akan mengerjakan tugas-tugas sebagai berikut:

- Menyediakan judul dokumen.
- Menyediakan metode untuk menfirim pesan ke tipe *browser* tertentu.
- Membantu *search engine* untuk melakukan proses pencarian dokumen dengan deskripsi halaman *web*, pencarian *domain*, dan data lain yang tertulis pada bagian *head* ini.
- Memisahkan halaman ke dalam dua bagian yang berbeda. Pemisahan ini bertujuan agar apabila halaman *web* tidak dapat di-*load*, maka informasi yang berada dalam *tag* <HEAD> akan tetap dapat di-*load* oleh *browser*.

- Memberikan perintah kepada *browser* untuk melakukan proses *loading* pada semua informasi yang terdapat pada *tag* <HEAD>.
- Memberikan tanda adanya keterikatan antar dokumen

2.1.2. Bagian *Body*

Bagian *body* merupakan isi dari dokumen HTML. Semua informasi yang akan ditampilkan, mulai dari teks, gambar, suara dan lain-lain akan ditempatkan di bagian ini. Bagian *body* ini diawali oleh *tag* <BODY> dan ditutup dengan *tag* </BODY>. *Tag* <BODY> mempunyai beberapa atribut yang digunakan untuk mengatur penampilan dokumen HTML, antara lain alink, background, bgcolor, bgproperties, leftmargin, link, text, topmargin, vlink, dan sebagainya.

2.2. Active Server Pages (ASP)

ASP atau *Active Server Pages* pertama kali dikenalkan kepada para pengembang *web* pada bulan Oktober 1996 dengan dikeluarkannya versi beta IIS 3 (*Internet Information Server*) untuk umum oleh Microsoft. Pada waktu itu, ASP dikenal juga dengan nama proyek pengembangannya, yaitu "Denali". Microsoft kemudian meluncurkan ASP 2 pada bulan Agustus 1997 sebagai bagian dari IIS 4, IIS 5, serta *Windows* 2000.

Active Server Pages (ASP) adalah suatu skrip yang bersifat *server-side* yang memungkinkan *developer* untuk mengerjakan proses dalam *server*. ASP bersifat *browser independent*, yang berarti bahwa aplikasi *web* dapat dijalankan oleh *browser* manapun serta memiliki kemampuan untuk dikombinasikan dengan teks, HTML dan komponen-komponen lain untuk membuat halaman *web* yang lebih menarik, dinamis dan interaktif. Komponen adalah objek yang sudah terkompilasi dengan *native code* di masing-masing *platform*, baik *platform* Windows, maupun *platform* lainnya.

Dengan ASP, *developer* akan mendapatkan kemudahan dalam membuat aplikasi *web*. Pilihan bahasa skrip yang digunakan adalah VBScript sebagai *default* dan Jscript. Namun *developer* juga dapat menggunakan bahasa skrip lainnya dengan menambahkan *add-in* untuk ASP, baik yang disediakan oleh Microsoft atau pihak lainnya.

Bukan hanya sisi kemudahannya saja yang membuat ASP dipakai oleh banyak *web developer* di dunia, akan tetapi jug karena ASP merupakan bagian dari *active platform* yang berbasiskan teknologi *Component Object Model* (COM). Dengan teknologi ini, ASP menjadi sangat efisien dalam segi konektivitas maupun penanganan aplikasi untuk transaksi yang jumlahnya sangat banyak. Hal ini dimungkinkan dengan pemakaian *Microsoft Transaction Server* (MTS).

2.2.1. Kelebihan ASP

ASP dimaksudkan untuk menggantikan teknologi yang lama yang bersifat *server-side,* seperti CGI (*Common Gateway Interface*) yang memiliki beberapa kelemahan dan berjalan di lingkungan UNIX. ASP memiliki beberapa kelebihan dalam pembuatan aplikasi dinamis, di antaranya:

- Dapat dijalankan pada sebuah PC berbasis Windows tanpa terhubung ke Internet dengan *Personal Web Server (PWS)* atau *Microsoft Internet Information Server* (IIS).
- Tidak ada proses *compiling* dan *linking*
- Merupakan skrip yang berorientasi pada obyek dan dapat dikembangkan lebih jauh dengan menggunakan komponen-komponen ActiveX *server* atau ADO.
- Sintak-sintaksnya mudah dipelajari karena tidak mengenal pendeklarasian variabel dan akses tingkat rendah lainnya
- Kode program atau skrip terintegrasi dengan *file* HTML sehingga memudahkan pendesainan tampilan.
- Dapat berinteraksi dengan aplikasi-aplikasi *web* yang dibuat dengan bahasa CGI, ISAP, serta skrip-skrip lainnya

2.2.2. Cara Kerja ASP

Cara kerja dari *Active Server Pages* tersebut dapat dijabarkan sebagai berikut:

1. *Browser* atau *client* mengakses suatu halaman *web* di mana halaman tersebut memiliki ekstensi .asp serta mengandung sintaks-sintaks dalam bahasa *scripting* ASP.

2. Permintaan atau *request* dari *client* atau *browser* berupa *file .asp* akan dikirim ke *server.*
3. Setelah permintaan dikirim ke *server,* maka tugas *server* adalah memeriksa isi *file* dan menentukan apakah ada kode dalam *file* tersebut yang harus dieksekusi.
4. Skrip ASP memproses di dalam *server* dan membuat hasil proses dalam bentuk halaman HTML serta mengirimkannya ke *browser* sehingga apabila seseorang ingin melakukan *View source* (melihat kode), maka skrip ASP yang memproses aplikasi tersebut tidak akan terlihat karena hanya hasil aplikasinya saja yang akan dikirim ke *browser* dalam bentuk halaman HTML

2.2.3. Objek-objek ASP

Ada enam jenis objek ASP yang dapat digunakan, yaitu Application, Session, Response, Request, Server, dan ObjectContext. Berikut adalah penjelasan singkat dari masing-masing objek dari ASP:

1. Application Object

   Kita dapat menggunakan application object untuk berbagi informasi pada suatu aplikasi di antara pemakainya. Aplikasi ASP didefinisikan sebagai semua file .asp pada virtual directory dan sub directory.
2. Session Object

   Session dimulai ketika user membuka web kita dan berakhir ketika user menutup window browser atau ketika session tersebut mencapai waktu timeout.
3. Response Object

   Response Object akan mengirimkan data yang kita inginkan ke client. Beberapa properti dan metode yang ada pada Response Object ini antara lain:

   - Response.write

     menuliskan sebuah variabe ke output HTTP sebagai sebuah string. Contoh penggunaan:

     ```
     <% Response.Write "Hello World" %>
     ```
   - Response.Redirect

mengirimkan pesan yang dialihkan ke browser, yang menyebabkannya mencoba untuk terhubung ke URL yang berbeda. Contoh penggunaanya:

```
<% Response. Redirect ("index.asp?id=10")%>
```

4. Request Object

   Request Object melakukan pengambilan data dari client untuk diproses lebih lanjut. Beberapa properti dan metode Request Object yang ada antara lain:

   - Form

     Request.Form digunakan untuk mengambil data yang dikirim melalui form dengan metode POST. Contoh penggunaannya:

     ```
     <% Request. Form ("nama") %>
     ```

   - QueryString

     Request.QueryString digunakan untuk mengambil data yang dikirim melalui query (URL yang tertulis di atas address bar pada browser) atau yang dikirim lewat form dengan metode GET. Contoh penggunaannya:

     ```
     <% Request.QueryString ("nama") %>
     ```

5. Server Object

   Server Object dapat melakukan akses ke metode-metode atau properti-properti yang ada di server.

   - CreateObject

     Server.CreateObject berfungsi untuk membuat object yang berasal dari komponen yang terpasang di server. Contoh penggunaannya:

     ```
     <% Set Conn = Server.CreateObject ("komponen.pemanggil") %>
     ```

   - MapPath

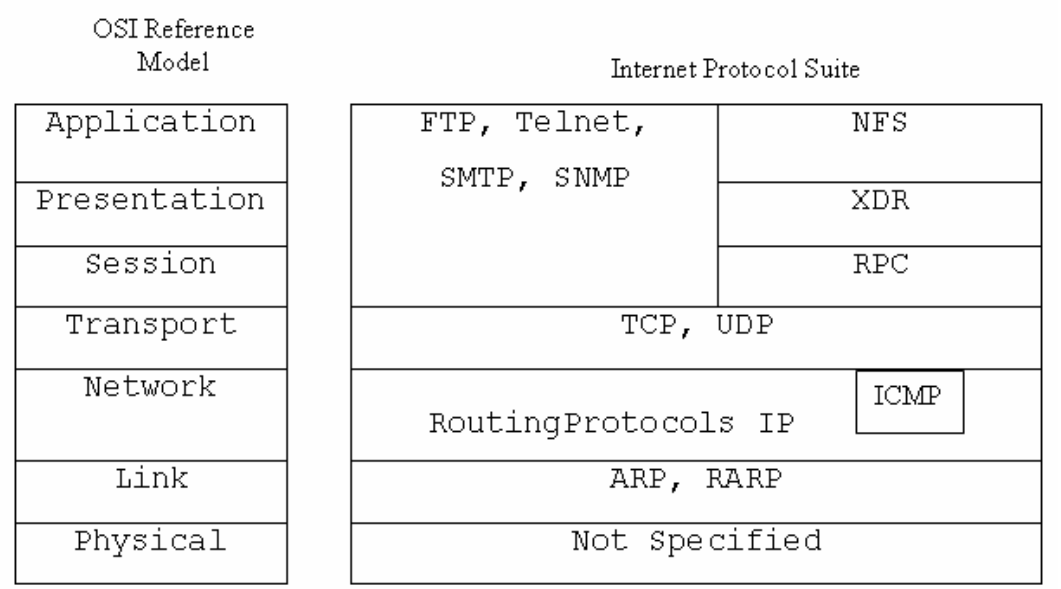
     mendapatkan letak direktori fisik di server. Contoh penggunaannya:

     ```
     <%= Server.MapPath ("default.asp")%>
     ```

2.3. Internet Protocols

Internet protocols adalah sistem protokol terbuka yang paling populer didunia. Hal ini disebabkan karena *Internet protocols* dapat digunakan untuk berkomunikasi antar rangkaian jaringan, dan tepat sekali digunakan untuk komunikasi *Local Area Network* (LAN) dan *Wide Area Network* (WAN). *Internet protocols* terbentuk dari beberapa *suite* protokol-protokol komunikasi, yang lebih dikenal dengan sebutan *Transmission Control Protocol* (TCP) dan *Internet*

Protocol (IP). Macam *suite* protokol yang ada tidak hanya terbatas pada *low-layer protocols* (misalnya TCP dan IP), tetapi juga dapat dipakai untuk menjalankan aplikasi umum, seperti misalnya *electronic mail, terminal emulation*, dan *file transfer*.

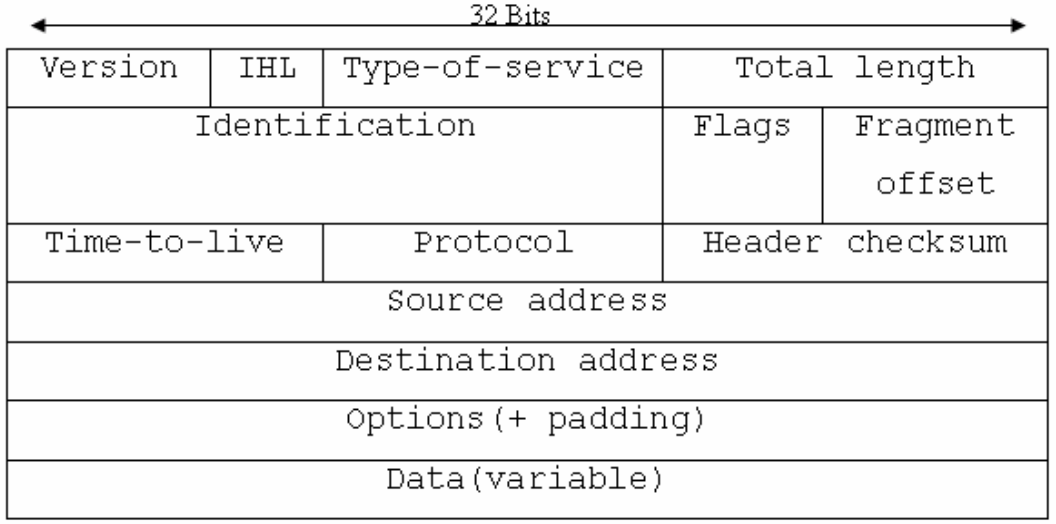


Gambar 2.1. Diagram Internet Protocols

2.3.1. Internet Protocol (IP)

Internet Protocol (IP) adalah merupakan *network-layer protocol* (*Layer* 3) yang berisi informasi pengalamatan dan pengendalian untuk mengedalikan paket-paket yang dikirim melalui jaringan. IP didokumentasikan pada RFC791 dan merupakan *network-layer protocol* utama didalam *Internet protocols suite.* Bersamaan dengan adanya *Transmission Control Protocol* (TCP), IP dapat dianalogikan seperti layaknya jantung dari semua protokol internet. IP memiliki dua tanggung jawab utama, yaitu:

- Menyediakan *connectionless best-effort delivery* dari *datagrams* melalui jaringan.
- Menyediakan fragmentasi dan perancangan kembali dari *datagrams* untuk mendukung hubungan-hubungan antar data yang memiliki ukuran *maximum-transmission unit* (MTU) berbeda.



Gambar 2.2. Format Paket Internet Protocol

Penjelasan mengenai bagian paket IP diatas adalah sebagai berikut:

- *Version* : menunjukkan versi IP yang digunakan.
- *IP Header Length* (IHL) : menunjukkan panjang *header* dari *datagrams* dalam 32-*bit words*.
- *Type-of-Service* : menjelaskan bagaimana *datagrams* harus diproses dan informasi tentang *importance level*.
- *Total Length* : menjelaskan panjang paket IP secara keseluruhan.
- *Identification* : berisi nilai *integer* yang mengidentifikasi *current datagram*.
- *Flags* : berisi 3-*bit field* untuk mengatur fragmentasi dari paket.
- *Fragment Offset* : berisi posisi dari fragmen data, sehingga IP tujuan dapat menggabungkan *datagram* tersebut ke bentuk semula.
- *Time-to-Live* : angka yang menunjukkan sampai kapan *datagram* tersebut harus dibuang.
- *Protocol* : untuk menandai *upper-layer* mana yang berhak menerima paket, setelah IP *processing* selesai.
- *Header Checksum* : menjaga integritas IP.
- *Source Address* : menunjukkan alamat pengirim.
- *Destination Address* : menunjukkan alamat penerima.
- *Options* : berisi pilihan tambahan, disesuaikan dengan kebutuhan.
- *Data* : berisi *upper-layer information*.

2.3.1.1. IP Addressing

Seperti dalam kehidupan sehari-hari, jaringan pun perlu mendapatkan alamat-alamat khusus yang dapat digunakan untuk menandai pengirim dan penerima. Alamat dalam jaringan tersebut lebih dikenal dengan istilah *IP Address*. Penulisan *IP Address* memiliki komponen khusus dan format penulisan tertentu.

Tiap *host* dalam jaringan TCP/IP memiliki 32-*bit logical address* yang unik, dan dibagi menjadi dua bagian utama, yaitu:

- Network number

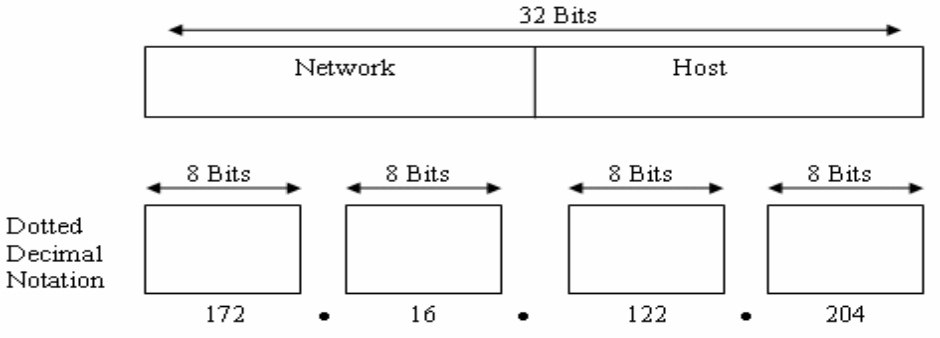
  Mengidentifikasikan sebuah jaringan, dimana nilai yang tercantum disini diisi oleh *Internet Network Information Center* (InterNIC), jika jaringan tersebut adalah bagian dari Internet. *Internet Service Provider* (ISP) dapat memperoleh *blocks* berupa *network address* dari InterNIC dan dapat menambah alamat lagi bila diperlukan.

- Host number

  Mengidentifikasikan *host* dalam sebuah jaringan, dimana nilai yang tercantum disini diisi oleh administrator jaringan.

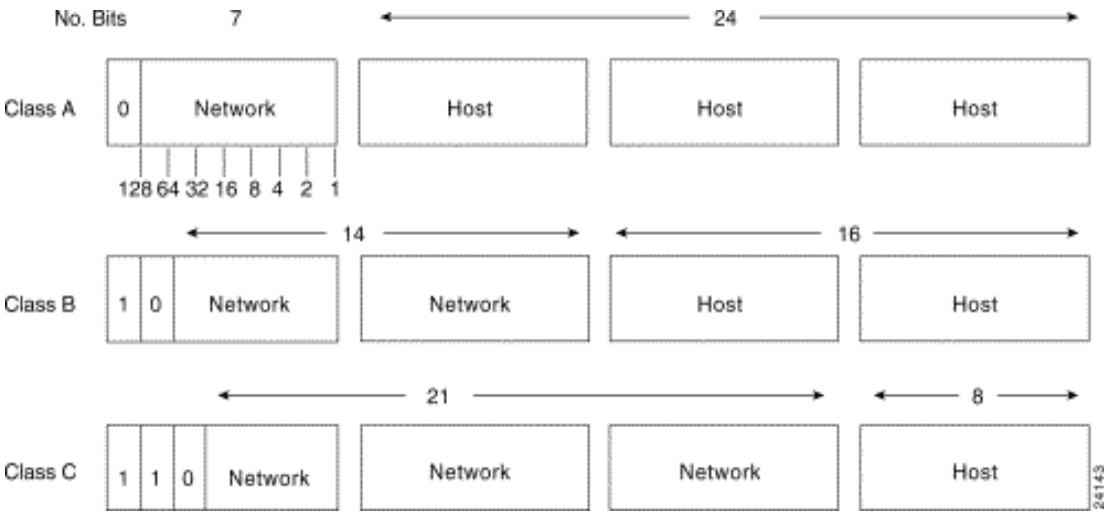
2.3.1.2. IP Address Format

Format 32-*bit IP Address* dibagi menjadi delapan *bits* pada saat yang sama, dipisahkan oleh *dots* (titik), dan ditampilkan dalam format *decimal*. Penamaan ini disebut juga *dotted decimal notation*. Tiap *bit* dalam *octet* memiliki *binary weight* tertentu (128, 64, 32, 16, 8, 4, 2, 1). Nilai minimumnya adalah 0, dan nilai maksimumnya adalah 255.



Gambar 2.3. Format Dasar IP Address

IP addressing mampu mendukung lima macam kelas *addressing* yang berbeda, yaitu: A, B, C, D, dan E. Hanya kelas A, B, dan C saja yang dimungkinkan untuk tujuan komersil.



Gambar 2.4. Format Kelas A, B, dan C

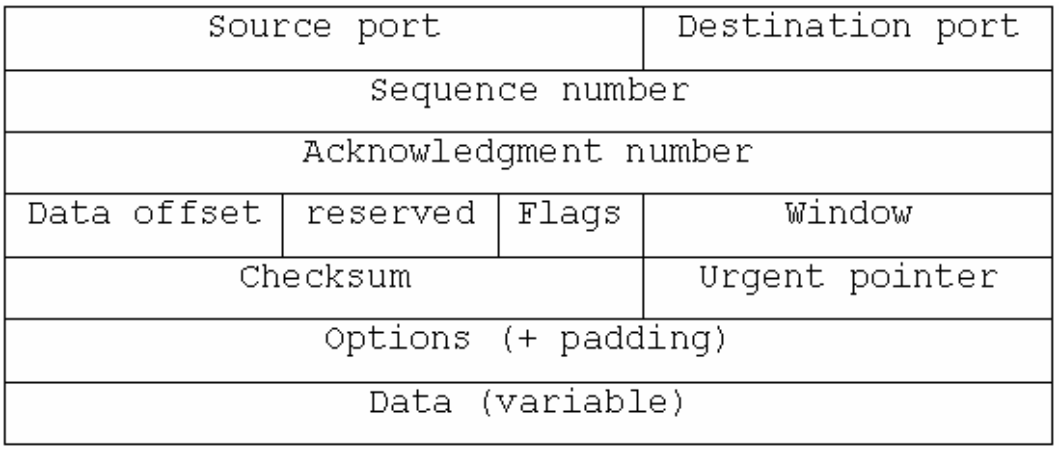
IP Address Client	Format	Purpose	High-Order Bit(s)	Address Range	No. Bits Network/Host	Max. Hosts
A	N.H.H.H	Few large organizations	0	1.0.0.0 to 126.0.0.0	7/24	16.777.214 (2^{24}-2)
B	N.N.H.H	Medium-size organizations	1,0	128.1.0.0 to 191.254.0.0	14/16	65.543 (2^{16}-2)
C	N.N.N.H	Small organizations	1,1,0	192.0.1.0 to 223.255.254.0	22/8	245 (2^{8}-2)
D	N/A	Multicast groups(RFC 1112)	1,1,1,0	224.0.0.0 to 239.255.255.255	N/A (not for commercial use)	N/A
E	N/A	Experimental	1,1,1,1	240.0.0.0 to 254.255.255.255	N/A	N/A
N = Network, H = Host number						

Gambar 2.5. Kelima Kelas IP Addressing

2.3.2. Transmission Control Protocol (TCP)

TCP menyediakan transmisi data yang memiliki reliabilitas tinggi didalam lingkungan jaringan IP. TCP memiliki koresponden terhadap *transport layer* (*Layer* 4) dari OSI *reference model*. TCP mendukung *stream data transfer*, *reliability*, *efficient flow control*, *full-duplex operation*, dan *multiplexing*.

Reliabilitas yang ditawarkan oleh TCP adalah dengan menggunakan sistem *connection-oriented*. Dimana paket yang dikirimkan melalui jaringan akan diberi nomor *acknowledgment* yang mengindikasikan nomor paket berikutnya. Paket yang tidak mendapatkan *acknowledgment* dalam durasi waktu tertentu akan dikirimkan ulang. Reliabilitas inilah yang memungkinkan TCP untuk mengatasi hilangnya data, duplikasi data, ataupun paket yang salah kirim. Pada saat pengiriman *acknowledgment* kembali ke sumber, TCP akan memproses nomor paket selanjutnya yang dapat diterima, tanpa mengakibatkan terjadinya *overflow* pada *internal buffers*. Konsep ini disebut juga sebagai *Full-duplex operation*, dimana TCP dapat menangani kedua macam proses kirim dan terima pada saat yang bersamaan.



Gambar 2.6. Format Paket TCP

Penjelasan mengenai paket TCP diatas adalah sebagai berikut:

- *Source Port* dan *Destination Port* : mengidentifikasikan *upper-layer source* mana dan proses penerimaan pada alamat tujuan.
- *Sequence Number* : nomor yang ditujukan pada *byte* pertama data, dan juga untuk menandai nomor paket berikutnya.

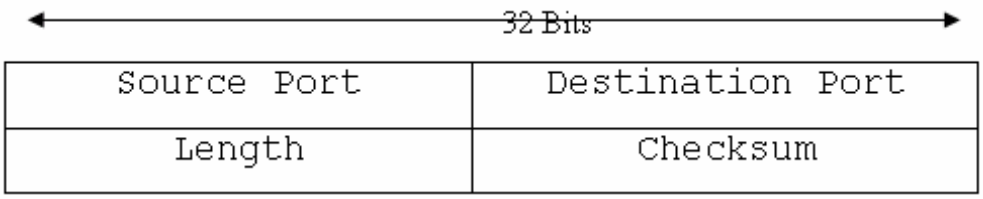
- *Acknowledgment Number* : berisi nomor urut dari *byte* berikutnya yang harus diterima.
- *Data Offset* : menunjukkan nomor 32-*bit words* pada TCP *header*.
- *Reserved* : belum diimplementasikan.
- *Flags* : berisi *control information*, misalnya SYN, ACK, dan FIN.
- *Window* : ukuran buffer yang perlu disediakan.
- *Checksum* : memeriksa apakah *header* rusak atau tidak selama *transit*.
- *Urgent Pointer* : menunjuk pada *urgent data byte* pertama pada paket.
- *Options* : berisi pilihan-pilihan TCP.
- *Data* : berisi informasi tentang *upper-layer*.

2.3.3. User Datagram Protocol (UDP)

UDP adalah sebuah *connectionless transport-layer protocol* (*Layer* 4) yang juga merupakan bagian dari *Internet protocols*. Pada dasarnya, UDP adalah *interface* antara IP dan *upper-layer processes*. Port protokol UDP membedakan beberapa aplikasi yang berjalan di sebuah *device*.

Tidak seperti hal nya TCP, UDP tidak memberikan reliabilitas, *flow-control*, ataupun *error-recovery* pada IP. Karena UDP ini begitu sederhana, maka *header* yang dimiliki mengandung lebih sedikit *bytes*. Sehingga tidak membutuhkan *network overhead* dibandingkan TCP. Walaupun demikian, UDP cukup berguna dalam situasi dimana mekanisme reliabilitas dari TCP tidak terlalu penting, misalnya pada saat *higher-layer protocol* memungkinkan untuk terjadi *error* dan *flow control*.

UDP juga merupakan *transport protocol* untuk beberapa aplikasi umum pada *application-layer protocols*, termasuk didalamnya adalah *Network File System* (NFS), *Simple Network Management Protocol* (SNMP), *Domain Name System* (DNS), dan *Trivial File Transfer Protocol* (TFTP).



Gambar 2.7. Format Paket UDP

Source dan *destination ports* berisi 16-*bit* UDP *protocol port number* yang digunakan untuk *demultiplex datagrams* di dalam penerimaan pada *application-layer processes*. *Length* menunjukkan berapa panjang *header* dan data UDP. *Checksum* menyediakan pemeriksaan integritas pada UDP *header* dan data, dan sifatnya adalah *optional*.

2.4. Windows Media Services (WMS)

WMS merupakan komponen *built-in* dari *Microsoft Windows* 2000 *Server*. Komponen ini memungkinkan *server* untuk melakukan *streaming* file-file multimedia melalui jaringan(*networks*), mulai dari *low-bandwidth, dial-up internet connections, Local Area Networks*(LAN).

WMS ini hampir sama cara kerjanya dengan sebuah web server. Dengan menggunakan komponen ini kita dapat menambah kemampuan web site kita untuk memainkan *online* radio dan program televisi, slide-slide presentasi, file transfer, *movie* dan *multimedia shows.*

Dalam menggunakan WMS, ada beberapa hal yang perlu diperhatikan. Hal-hal yang perlu diperhatikan dalam aplikasinya di internet adalah:

- Tidak semua jaringan internet *multicast-enabled*, jadi harus menyediakan alternatif lain berupa *unicast streaming*.
- Bandwith dari internet memiliki *reliabilitas* yang kurang, dimana *internet traffic* dapat meningkat dengan cepat dan type *connection* dari tiap user dapat berbeda-beda.
- Jaringan yang menggunakan *proxy server* harus melakukan beberapa konfigurasi dahulu agar *client* dapat mengakses WMS.

Hal-hal yang perlu diperhatikan dalam aplikasinya di intranet adalah:

- Beberapa sistem dari komponen yang ada mungkin terpisah oleh *firewall*.

- Untuk melakukan *multicasting*, jaringan harus memiliki *routers* yang *multicast-enabled.*
- Perhitungkan berapa jumlah administrator yang akan mengolah server, apakah diperlukan remote admin atau tidak.
- Perhitungkan berapa jumlah client yang akan mengakses server.

Untuk mengoperasikan WMS ini dapat dilakukan dengan cara mengakses *Windows Media Administrator*. Dimana *interface* dari *Windows Media Administrator* ini sendiri merupakan bentuk luar dari WMS yang akan memudahkan kita dalam mengoperasikannya secara manual. Dari *Windows Media Administrator* ini, kita dapat mengoperasikan WMS secara fungsional dan menyeluruh. Adapun yang dapat kita lakukan diantaranya adalah membangun *Unicast Publishing Point*, *Multicast Station*, *Multicast File Transfer*, dan *Server Properties*. Berikut ini adalah tampilan yang akan kita dapatkan bila kita membuka *Windows Media Administrator*.

Gambar 2.8. Windows Media Administrator

2.4.1. Konsep

Proses *streaming* yang dapat dilakukan oleh WMS meliputi file-file bertype .ASF, .WMA, .MP3, dan .WAV.

Tiap *client* dapat memainkan isi file pada saat file itu dikirimkan melalui jaringan tanpa harus mendownloadnya terlebih dahulu. Hal ini dikarenakan data tersebut dikirim secara *streaming*. *Streaming* tersebut dapat mengurangi waktu tunggu dan tempat penyimpanan pada *client* secara signifikan. Juga memungkinkan untuk melakukan presentasi tanpa terbatas waktu, seperti halnya *live broadcasts.*

File-file yang didukung untuk di*stream* oleh WMS antara lain:

- .ASF

  Advanced Streaming Format (ASF), adalah sebuah format dari file multimedia *streaming* yang dikembangkan oleh Microsoft dan digunakan oleh isi Windows Media Audio dan/atau Video yang dikompresi dengan menggunakan *codec-codec* yang beraneka ragam dan dapat disimpan sebagai sebuah *file* ASF, dimainkan dengan menggunakan Windows Media Player, serta di-*stream* dengan menggunakan Windows Media Services. ASF telah diajukan kepada ISO dan IETF untuk mendapatkan standarisasi. File ASF ini diharapkan dapat mengambil alih peran dari format AVI sebagai pendahulunya. ASF adalah merupakan format file yang sangat baik di dalam menyimpan data multimedia tersinkronisasi. ASF memberikan dukungan terhadap pengiriman data melalui jaringan-jaringan dan protokol-protokol yang beraneka ragam.

- .WMA

  Windows Media Audio adalah bentuk lain daripada .ASF. File in berisi data *audio* terkompresi dengan menggunakan metode kompresi tingkat tinggi (*codec*) yang disebut *Windows Media Audio*.

- .MP3

  File MP3 dapat di*stream* oleh secara langsung dengan menggunakan WMS seperti halnya file .ASF dan .WMA. Sehingga file tersebut dapat dimainkan secara langsung tanpa harus di*download*.

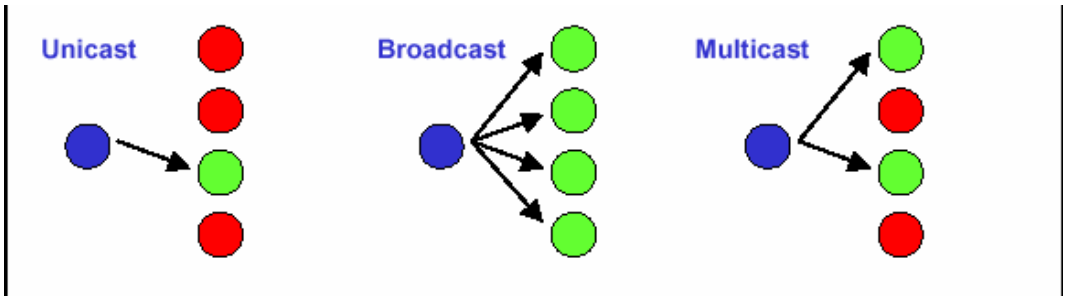
2.5. Streaming Technologies

Proses pengiriman *file* dari satu tempat ke tempat lain dapat dilakukan melalui sebuah jaringan, dimana konsepnya adalah dengan membagi *file* tersebut menjadi paket-paket kecil, dan pada paket tersebut diberi sistem penomoran, yang fungsinya sebagai pemberi informasi tambahan tentang alamat pengirim dan penerima, ukuran paket, dan lain sebagainya. Sehingga pada saat paket tersebut dikirimkan, pihak penerima akan membaca sistem penomorannya. Apabila ada kesalahan dalam pengiriman, misalnya ada data yang hilang/rusak, paket dapat dikirim ulang. Untuk konsep mengenai pengiriman paket dan sistem penomorannya, dapat dilihat pada sub bab 2.3 mengenai *Internet Protocols*.

Seiring dengan perkembangan teknologi dan aplikasinya, ditemukan bahwa sistem pengiriman *file* seperti itu cukup memberatkan jaringan, terutama bila *file* yang dikirimkan itu adalah *multimedia file*. Hal ini dikarenakan ukuran *multimedia file* yang cukup besar, sehingga bila dikirimkan secara utuh melalui jaringan dapat menghabiskan *bandwith* yang cukup banyak. Pada umumnya *multimedia file* tersebut tidak diperbolehkan untuk diduplikasi secara sembarangan, karena itu mungkin saja melanggar hak cipta dari pemilik *multimedia file* yang sesungguhnya, contoh kasus adalah maraknya *file* berformat MP3 di Internet yang kebanyakan adalah *illegal copy* dari album lagu yang orisinal.

Bersumber dari masalah yang dihadapi sebelumnya, muncullah sebuah teknologi pengiriman *multimedia contents* yang dianggap lebih baik dan *reliable*. Teknologi ini adalah dengan menggunakan cara *streaming*. *Streaming* ini adalah sebuah teknologi baru, dimana *client* dapat menerima isi dari *file* yang dikirim oleh *server*, dan langsung memainkannya tanpa harus menunggu semua isi *file* terkirim secara lengkap. Teknologi *streaming* ini dibagi menjadi dua tipe, yaitu *unicast* dan *multicast*. Kedua teknologi ini memiliki konsep yang hampir sama, yaitu mengirimkan *multimedia content* melalui jaringan secara *streaming*, dimana ketika *server* mengirimkan isi *file*, *client* akan melakukan *buffering* terlebih dahulu untuk mengumpulkan isi *file* tersebut didalam *buffer*. Setelah *buffer* penuh, barulah isi paket tersebut dimainkan dengan menggunakan program *media player*,

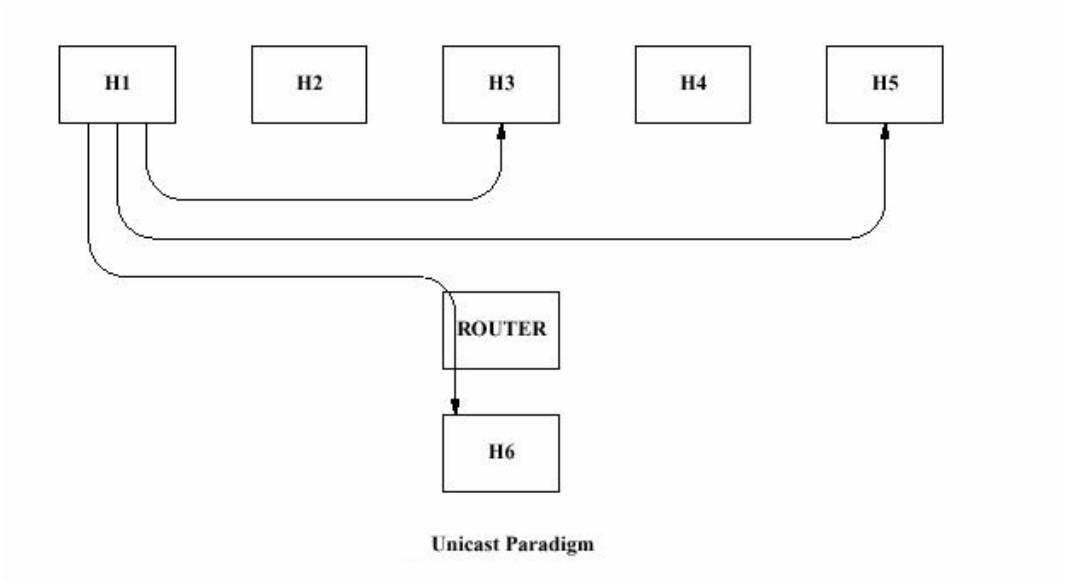
dimana untuk kasus ini adalah *Windows Media Player*. Perbedaan umum antara *unicast* dengan *multicast* akan dibahas secara terpisah melalui sub berikut ini.



Gambar 2.9. Type Teknologi Streaming

2.5.1. Unicast

Unicast connection adalah koneksi secara langsung antara *server* dan *client*, dimana *client* akan menerima *stream* yang dikhususkan untuk *client* tersebut. *Stream* yang dikirim oleh *server* adalah hasil permintaan dari *request client*.



Gambar 2.10. Diagram Konsep Unicast

Unicast stream dapat dikirimkan melalui 2 macam cara, yaitu:

- *On-demand*

  Menggambarkan isi dari *media content* yang memungkinkan untuk dikirim secara *streaming* melalui *Windows Media Services*. *Windows Media Services*

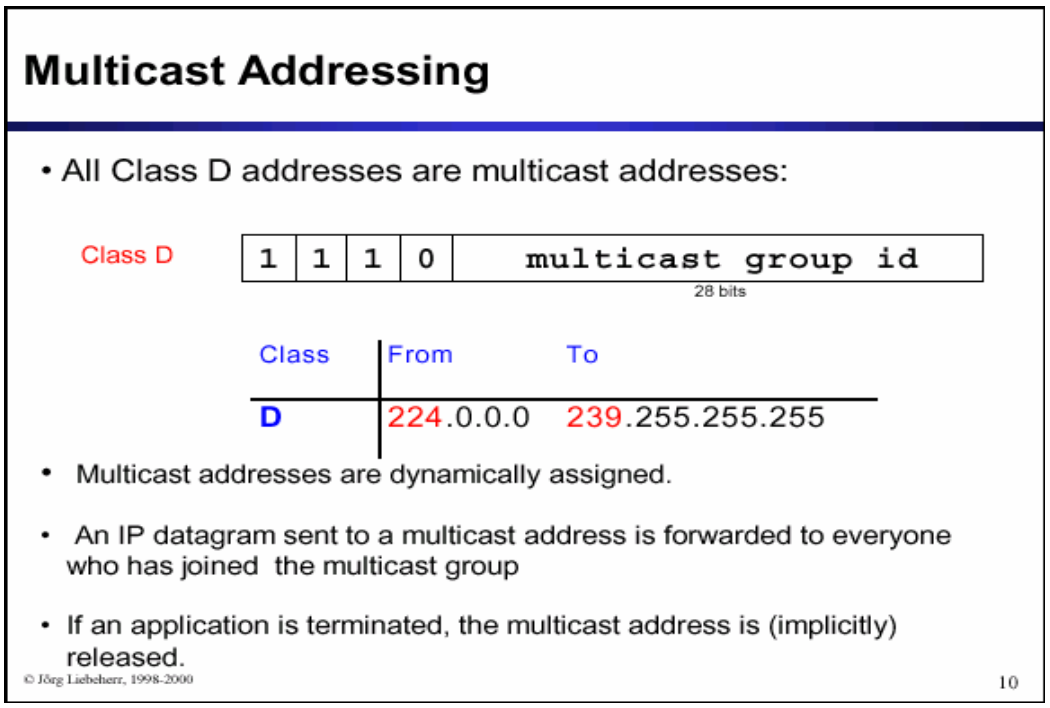
mengirimkan *media content* tersebut secara *streaming* melalui *publishing point* ataupun menggunakan *Windows Media Encoder*.

- *Broadcast*

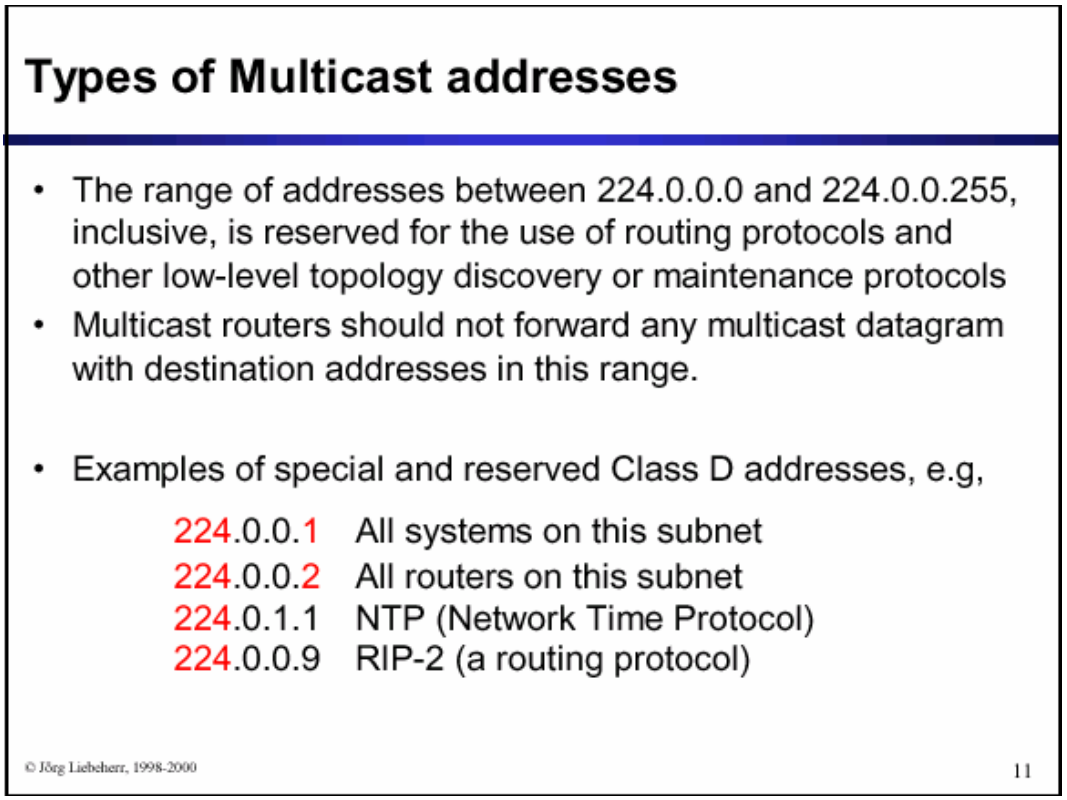
  Memiliki konsep yang hampir sama dengan *On-demand Unicast*, hanya saja yang membedakan adalah *client* disini bersifat pasif. Sehingga *client* tidak memiliki peran apa-apa pada terhadap *media file* yang dikirimkan, selain menerima *file contents.*

2.5.2. Multicast

Adalah sebuah teknologi komunikasi didalam jaringan, dimana konsep yang digunakan adalah mengirimkan paket data melalui sebuah sumber ke banyak penerima/tujuan didalam jaringan. Konsep ini hampir sama dengan *broadcast*, namun yang membedakan adalah, pada *broadcast*, *file* yang dikirimkan akan diterima oleh semua *client* yang ada didalam jaringan, tanpa peduli apakah *client* tersebut menginginkan *file* itu atau tidak. Namun *file* tersebut tidak dapat dikirimkan keluar melalui *router*. Sehingga bila ada *client* yang ingin tahu tentang *file* tersebut tidak dapat melihatnya, karena berada diluar jaringan lokal. Sedangkan pada *multicast*, *file* publikasikan melalui IP tertentu, dimana IP yang dapat dipakai untuk *multicast* ini adalah IP kelas D. Ini berarti bahwa IP yang dapat digunakan untuk *multicast* memiliki jangkauan antara 224.0.0.1 sampai 239.225.225.225. Melalui IP inilah data tersebut dikirim. Sehingga bila ada *client* yang tertarik untuk melihat *file* yang dikirimkan dapat membukanya pada *multicast* IP yang telah ditentukan oleh *host*.



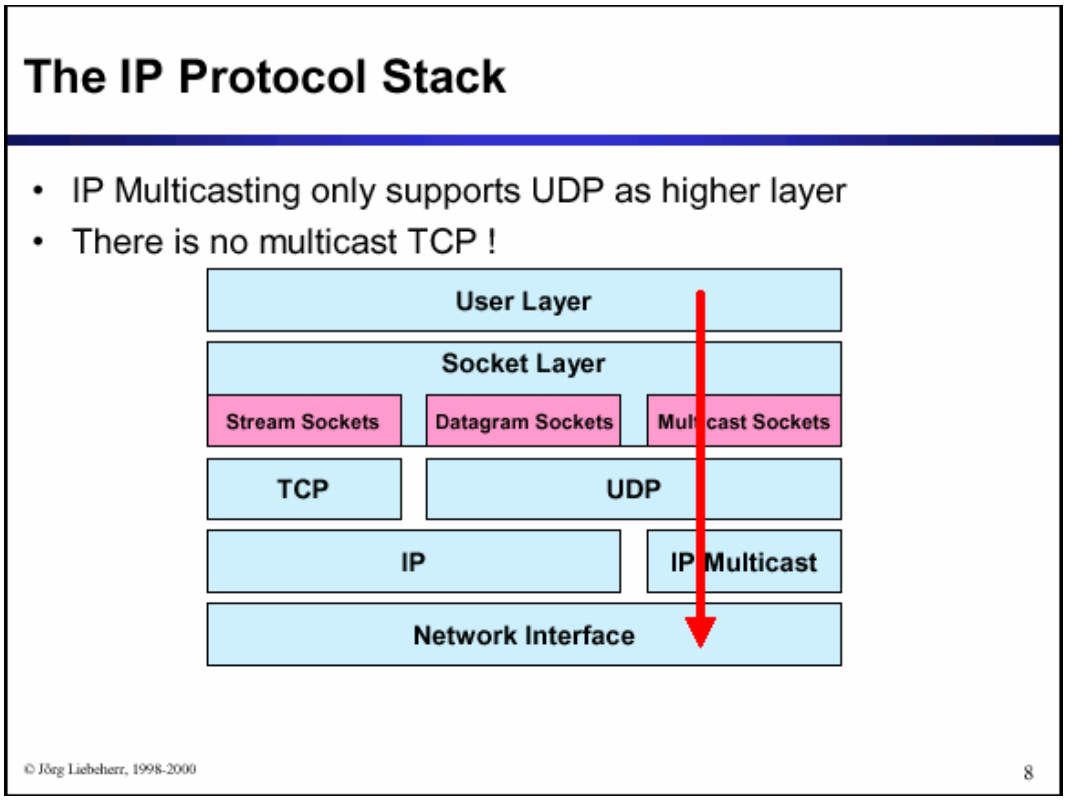
Gambar 2.11. Multicast Addressing



Gambar 2.12. Type Multicast Addressing

Konsep *Multicast* ini adalah pengiriman data secara *streaming* dengan menggunakan protokol dasar UDP, oleh karena itu proses transmisi yang terjadi

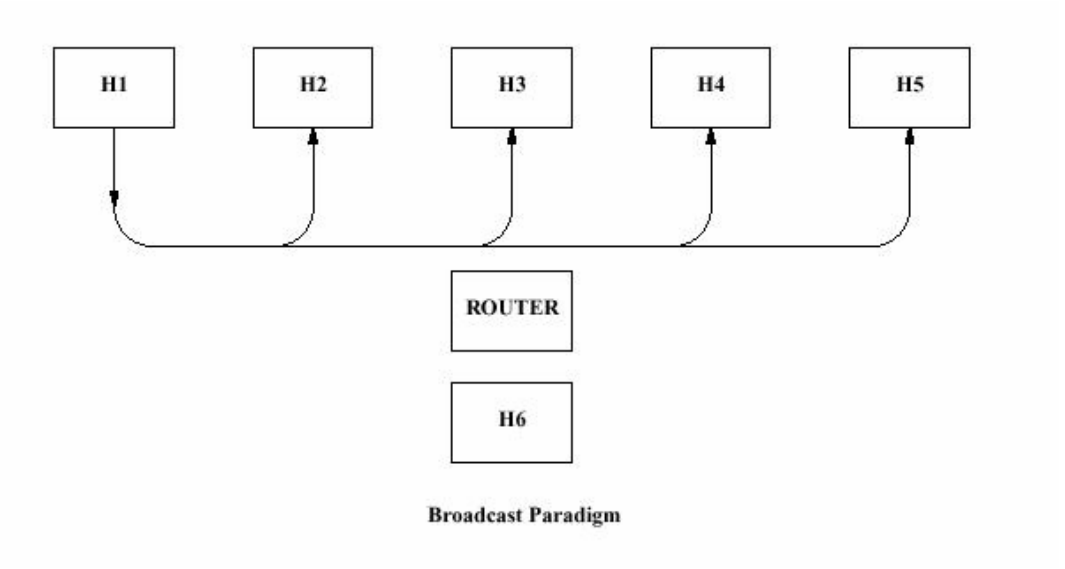
jelas tidak *reliable*. Keuntungan dari penggunaan *multicast* ini adalah satu pesan/paket dari *host* hanya akan dikirimkan sekali saja kebanyak *client*. Dengan demikian tentu saja proses ini lebih menghemat *bandwith.* Keterbatasan dari protokol yang dipakai ini, yaitu UDP, adalah tidak adanya *flow control* dan reliabilitas seperti halnya TCP. Namun keuntungan dari penggunaan UDP ini dikarenakan kecilnya format paket yang dikirim, sehingga tidak memerlukan *bandwith* yang terlalu besar. Penghematan *bandwith* ini juga terjadi karena UDP tidak menggunakan *acknowledgment* terhadap paket yang dikirim. Hal ini cukup menghemat banyak waktu.



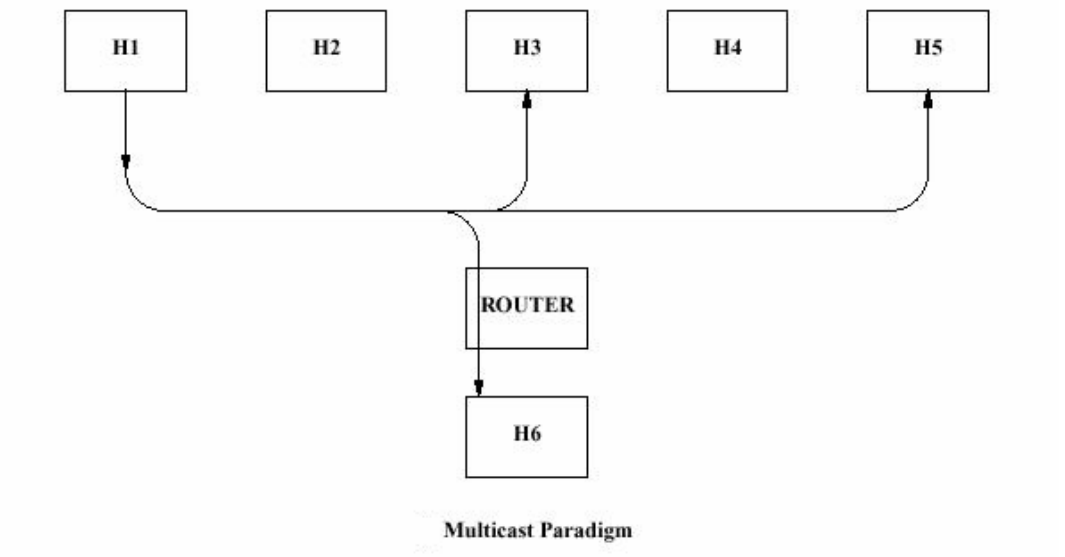
Gambar 2.13. IP Protocol Stack

Yang merupakan ciri khas dari *multicast* ini adalah, tidak adanya koneksi secara langsung antara *Windows Media Services* dengan *clients. Windows Media Services* akan membuat sebuah *file* .nsc (NetShow Channel) pada saat pertama kali *station* dibuat. Pada umumnya, .nsc ini dikirimkan kepada *client* melalui sebuah *web server*. Isi dari .nsc ini adalah informasi yang diperlukan oleh *Windows Media Player* untuk menerima paket hasil *multicast*. Hal ini dapat dianalogikan seperti bila kita mengatur radio kita ke frekuensi tertentu untuk

mendengarkan acara dari *radio station* yang memiliki frekuensi tersebut. Penambahan *client* dalam menerima hasil *multicast* ini tidak memberikan penambahan *overhead* pada *server*. Sebab pada kenyataannya, *server* hanya mengirimkan sebuah *stream* per *multicast station*, dan tidak dipengaruhi oleh jumlah *client*.



Gambar 2.14. Diagram Konsep Broadcast



Gambar 2.15. Diagram Konsep Multicast