BAB II
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian ini menggunakan beberapa referensi dari jurnal dan liberator hasil penelitian sejenis yang relevan sebelumnya, antara lain:

Pada penelitian yang dilakukan oleh Imas, Akuwan dan Idris (2021). Dengan judul "Analisa QoS Pada Jaringan MPLS Berbasis *Routing* OSPF" [7]. Penelitian ini menitikberatkan pada pembangunan aplikasi jaringan secara *test-bed* yang mampu merepresentasikan MPLS pada jaringan IPv6 berbasis OSPF beserta analisis terhadap parameter QoS dari hasil pengukuran. Dari hasil penelitian didapatkan bahwa jaringan MPLS (*Multi Protocol Label Switching*) akan bekerja secara optimal apabila terdapat banyak *switching* dalam jaringan, selain itu jaringan MPLS akan terasa dampaknya jika pengiriman data dilakukan dalam kapasitas ber-gigabyte. *Delay* pada jaringan OSPF mengalami stabilan begitupun juga dengan delay pada MPLS. *Throughput* MPLS dan jaringan OSPF mengalami kenaikan secara bertahap.

Pada penelitian yang dilakukan oleh Donny, Dadiek dan Nanda (2021). Dengan judul "Analisis Perbandingan Performansi Jaringan IPv4 dan IPv6 pada MPLS VPN Menggunakan Server IMS *Core*" [8]. Penelitian ini menggunakan 6VPE sebagai metode untuk translasi dari IPv4 ke IPv6 yang dapat berjalan pada jaringan *backbone* MPLS VPN. Dari hasil simulasi didapatkan *throughput* terbaik pada layanan *voice call* IPV4 sebesar 7488 kbps, *delay* dan *jitter* terbaik pada layanan *text messaging* IPv4 sebesar 0,0206 ms dan 0,0015 ms, *packet loss* terbaik pada layanan *voice call* dan *text messaging* dengan IPv4 dan IPv6 sebesar 0%. Sedangkan, pada layanan video *call* grafik *traffic loss* mengalami penurunan dengan nilai 0,2%, 0,1% dan 0,1%. Penelitian ini menyimpulkan bahwa *backbone* IPv4 lebih baik daripada *backbone* IPv6 digunakan pada jaringan MPLS VPN menggunakan server *Open IMS Core*.

Pada penelitian yang dilakukan oleh Muhaemin Alparisi, Indrarini dan Muhammad Iqbal (2020). Dengan judul "Implementasi Jaringan Menggunakan *Routing* Protokol OSPF (*Open Shortest Path First*) Dan MPLS (*Multi Protocol Label Switch*) Dengan Redudansi HSRP" [9]. Penelitian ini dilakukan dengan membuat jaringan dengan metode HSRP. Pengujian dilakukan dengan menggunakan perintah *ping* dan *traceroute,* untuk menguji *failover* pada HSRP. Dari hasil pengujian yang dilakukan menunjukkan bahwa dengan menggunakan metode HSRP dapat memberikan ketersediaan jaringan dengan *packet loss* sebesar 0%, *throughput* sebesar 564 kbps, *delay* sebesar 0.013s pada layanan data dengan *background traffic* 150 mbps. *Packet loss* sebesar 19,33%, *throughput* sebesar 106 kbps, *delay* sebesar 22 ms pada layanan *voip* dengan *background traffic* 150 mbps. *Packet loss* sebesar 0%, *throughput* sebesar 8 kbps, *delay* sebesar 474 ms pada layanan video *stream* dengan *background traffic* 150 mbps. Pada kondisi *failure, delay* yang terjadi lebih besar daripada saat kondisi normal. Hal ini terjadi karena perpindahan jalur dari *active router* ke *standby router*. Namun nilai *packet loss* dan *delay* masih memenuhi *standard* ITU-T G.1010.

Pada penelitian yang dilakukan oleh Nur, Herman dan Imam Riadi (2023). Dengan judul "Perbandingan Kinerja Routing Protokol OSPF dan OSPF-MPLS Dalam Pengiriman Paket TCP dan UDP" [10]. Hasil penelitian menunjukkan bahawa baik OSPF maupun OSPF MPLS menunjukkan kinerja yang baik dengan nilai indeks rata-rata sebesar 3.75, memenuhi standar kinerja jaringan menurut TIPHON. OSPF MPLS menunjukkan *throughput* yang sedikit lebih rendah dibandingkan dengan OSPF *non* MPLS, kedua skenario menunjukkan *packet loss* yang sangat rendah, *delay* yang rendah dan *jitter* yang stabil. Selain itu, penggunaan OSPF MPLS untuk layanan UDP mengurangi waktu *delay*, menunjukkan keunggulan OSPF MPLS dalam hal parameter tertentu, terutama untuk layanan UDP. Kedua skenario *routing* dapat memberikan performa yang memadai untuk berbagai jenis layanan jaringan, implementasi skenario OSPF dan OSPF MPLS menunjukkan kinerja yang baik untuk layanan TCP dan UDP.

Pada penelitian yang dilakukan oleh Rinto, Fati dan Asazidudu (2022). Dengan judul "Analisis QoS *Routing* OSPF IP Versi 4 dan OSPF IP Versi 6 Pada Mikrotik OS" [11]. Pada penelitian ini dilakukan untuk menganalisis kerja protokol *routing* OSPF dengan jaringan IPv4 dan IPv6 dengan melihat nilai dari rancangan serta konfigurasi jaringan menggunakan parameter QoS berupa *delay*, *packet loss*, *throughput* dan *jitter*. Dilakukaan perbandingan dengan simulasi GNS3 untuk menentukan protokol *routing* terbaik antara OSPF Ip versi 4 dan OSPF V3 IP versi 6. Berdasarkan hasil perbandingan OSPF pada multi area, didapati *routing* OSPF V3 pada IPV6 lebih unggul dari OSPF IPv4 dari semua parameter terutama pada waktu *delay* untuk IPv4 496ms, IPv6 438ms dan *packet loss* untuk IPv4 19,74% sedangkan IPv6 11,97%.

Tabel 2.1 menunjukkan *literatur review* untuk mengetahui data penelitian yang sudah dilakukan sebelumnya dan dijadikan referensi serta mengetahui pendekatan-pendekatan yang dilakukan oleh peneliti sebelumnya. Tabel ini juga menunjukkan persamaan dan perbedaan yang terdapat pada penelitian yang dilakukan oleh peneliti saat ini.

Tabel 2.1 *Literatur Review*

No	Judul	Hasil	*Review*
1	Analisa QoS Pada Jaringan MPLS Berbasis *Routing* OSPF	MPLS (*Multi Protocol Label Switching*) akan bekerja secara optimal apabila terdapat banyak *switching* dalam jaringan, *Delay* pada jaringan OSPF mengalami stabilan begitupun juga dengan delay pada MPLS.	1. Persamaan Menggunakan MPLS dan melakukan analisis parameter QoS. 2. Perbedaan Terdapat pada analisis layanan paket yang dikirimkan dari *server* menuju *client*.

No	Judul	Hasil	*Review*
2	Analisis Perbandingan Performansi Jaringan IPv4 dan IPv6 pada MPLS VPN Menggunakan Server IMS *Core*	Dari hasil simulasi didapatkan *throughput* terbaik pada layanan *voice call* IPV4 sebesar 7488 kbps, *delay* dan *jitter* terbaik pada layanan *text messaging* IPv4 sebesar 0,0206 ms dan 0,0015 ms, *packet loss* terbaik pada layanan *voice call* dan *text messaging* dengan IPv4 dan IPv6 sebesar 0%. Penelitian ini menyimpulkan bahwa *backbone* IPv4 lebih baik daripada *backbone* IPv6 digunakan pada jaringan MPLS VPN menggunakan server *Open IMS Core*.	1. Persamaan Membandingkan performa jaringan IPv4 dan Ipv6 dan QoS. 2. Perbedaan Penelitian dilakukan pada jaringan MPLS OSPF dan menggunakan *software* GNS3.
3	Implementasi Jaringan Menggunakan *Routing* Protokol OSPF (*Open Shortest Path First*) Dan MPLS (*Multi Protocol Label*	Dari hasil pengujian yang dilakukan menunjukkan bahwa dengan menggunakan metode HSRP dapat memberikan ketersediaan jaringan dengan *packet loss*	1. Persamaan Menggunakan routing protokol OSPF dan MPLS serta analisis parameter QoS. 2. Perbedaan Penelitian tanpa menggunakan metode

No	Judul	Hasil	*Review*
	Switch) Dengan Redudansi HSRP	sebesar 0%, *throughput* sebesar 564 kbps, *delay* sebesar 0.013s pada layanan data dengan *background traffic* 150 mbps. Pada kondisi *failure*, *delay* yang terjadi lebih besar daripada saat kondisi normal. Hal ini terjadi karena perpindahan jalur dari *active router* ke *standby router*.	HSRP dan melakukan pengujian pada layanan TCP.
4	Perbandingan Kinerja Routing Protokol OSPF dan OSPF-MPLS Dalam Pengiriman Paket TCP dan UDP	Hasil penelitian menunjukkan bahawa baik OSPF maupun OSPF MPLS menunjukkan kinerja yang baik dengan nilai indeks rata-rata sebesar 3.75, memenuhi standar kinerja jaringan menurut TIPHON. OSPF MPLS menunjukkan *throughput* yang sedikit lebih rendah dibandingkan dengan OSPF *non* MPLS, kedua skenario	1. Persamaan Menggunakan routing protokol OSPF serta analisis parameter QoS. 2. Perbedaan Terdapat pada perbandingan yang dilakukan, penelitian ini membandingkan kinerja IPv4 dan IPv6 hanya pada layanan TCP dengan simulasi GNS3.

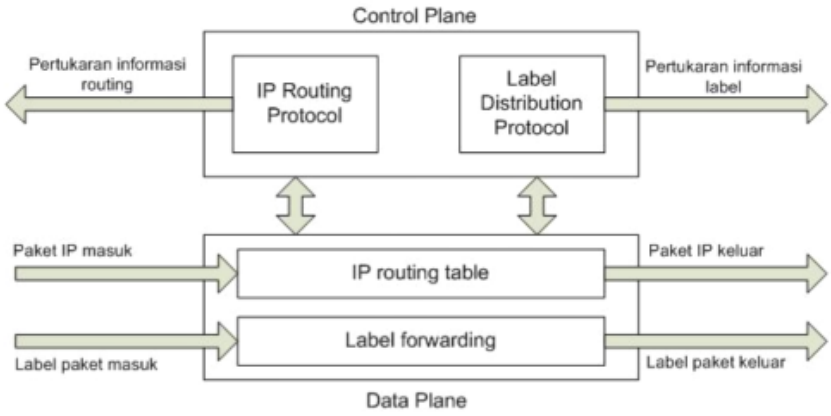
No	Judul	Hasil	*Review*
		menunjukkan *packet loss* yang sangat rendah, *delay* yang rendah dan *jitter* yang stabil	
5	Analisis QoS *Routing* OSPF IP Versi4 dan OSPF IP Versi 6 Pada Mikrotik OS	Berdasarkan hasil perbandingan OSPF pada multi area, didapati *routing* OSPF V3 pada IPV6 lebih unggul dari OSPF IPv4 dari semua parameter terutama pada waktu *delay* untuk IPv4 496ms, IPv6 438ms dan *packet loss* untuk IPv4 19,74% sedangkan IPv6 11,97%.	1. Persamaan Analisis QoS pada IPv4 dan IPv6 dengan OSPF *routing*. 2. Perbedaan Terdapat pada perbandingan yang dilakukan, penelitian ini membandingkan kinerja IPv4 dan IPv6 hanya pada layanan TCP dengan *software* simulasi GNS3.

2.2 DASAR TEORI

2.2.1 *Multiprotocol Label Switching* (MPLS)

Multiprotocol Label Switching (MPLS) adalah salah satu metode dengan performa tinggi yang dapat digunakan untuk tuning jaringan agar lebih meningkatkan performa jaringan. MPLS adalah teknologi penyampaian paket pada jaringan *backbone* berkecepatan tinggi yang menggabungkan beberapa kelebihan dari sistem komunikasi *circuit-switched* dan *packet switched* yang melahirkan teknologi yang lebih baik dari keduanya [12]. Arsitektur network MPLS yang didefinisikan oleh *Internet Engineering Task Force* (IETF) digunakan untuk memadukan

mekanisme *label swapping* di layer 2 dengan *routing* di layer 3 untuk mempercepat pengiriman paket. Teknologi berbasis *Multiprotocol Label Switching* menawarkan metode yang dikembangkan untuk memperbaiki kinerja dari teknologi IP dengan menerapkan konsep *routing* karena mekanisme pengiriman paket akan dilakukan secara sederhana dengan memberikan label pada tiap paket data yang masuk dan akan sampai pada alamat tujuan [13].

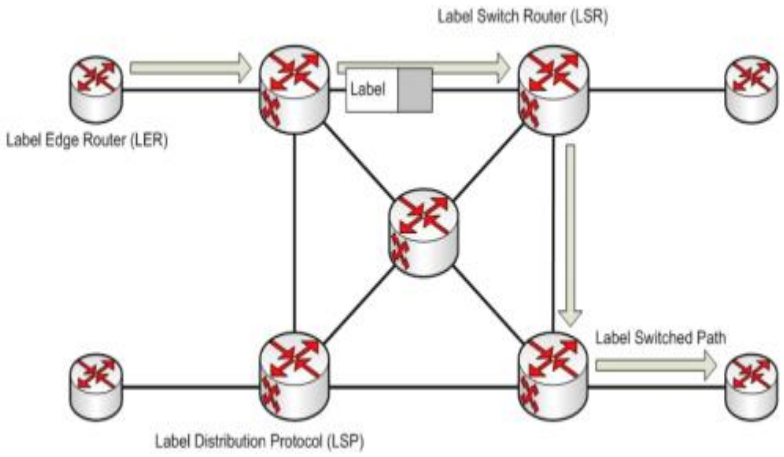


Gambar 2.1 Arsitektur node MPLS [14]

Pada gambar 2.1 menunjukkan bahwa MPLS memiliki dua komponen utama, yaitu:

1. *Control plane*, bertugas mempertukarkan informasi *routing* layer 3 dan label, berisi mekanisme kompleks untuk mempertukarkan informasi *routing* dengan menggunakan OSPF, EIGRP, IS-IS, dan BGP serta mempertukarkan label dengan menggunakan TDP, LDP, BGP dan RSVP.
2. *Data/Forwarding Plane*, bertugas meneruskan paket berdasarkan alamat tujuan maupun label. Memiliki mesin *forwarding* label sederhana yang terpisah dengan *routing* protokol maupun label *exchange protocol*. Tabel *Label Forwarding Information Base* (LFIB) digunakan untuk meneruskan paket berdasarkan label. Tabel LFIB dipopulasi oleh *label exchange protocol* yang digunakan, baik LDP maupun TDP atau dua-duanya.

MPLS terdiri atas sirkuit yang disebut *label-switched path* (LSP) yang menghubungkan *node-node* yang disebut *label-switched router* (LSR). LSR pertama yang merupakan awal tempat masuknya paket disebut dengan *ingress* dan LSR terakhir tempat keluar paket dari MPLS disebut *egress*. Setiap LSP dikaitkan dengan dengan sebuah *forwarding equivalence class* (FEC), yang merupakan kumpulan paket yang menerima perlakuan *forwarding* yang sama di sebuah LSR. FEC diidentifikasikan dengan pemasangan label.



Gambar 2.2 Komponen MPLS [14]

Berikut detail komponen yang ada dalam MPLS, seperti digambarkan pada gambar 2.2.

1. *Label Switched Path* (LSP)

   Merupakan jalur yang melalui satu atau serangkaian LSR dimana paket diteruskan oleh *label swapping* dari satu MPLS *node* ke MPLS *node* yang lain. MPLS menyediakan dua cara untuk menetapkan LSP yaitu.

   a. *Hop-by-hop routing*, cara ini membebaskan masing-masing LSR menentukan *node* selanjutnya untuk mengirimkan paket. Cara ini mirip seperti *Open Shortest Path First* (OSPF) dan *Routing Information Protocol* (RIP) dalam IP *routing*.
   b. *Explisit routing*, dalam metode ini LSP akan ditetapkan oleh LSR pertama yang dilalui aliran paket.

2. *Label Switching Router*

   Merupakan *router* dalam MPLS yang berperan dalam menetapkan LSP dengan menggunakan teknik *label swapping* dengan kecepatan yang telah ditetapkan.

3. MPLS *Edge Node* atau *Label Edge Router* (LER)

   Merupakan *router* MPLS yang menghubungkan sebuah MPLS domain dengan *node* yang berada di luar MPLS domain.

4. MPLS *Ingress Node*

   MPLS *node* yang mengatur trafik saat memasuki MPLS domain.

5. MPLS *Egress Node*

   MPLS *node* yang mengatur trafik saat akan meninggalkan MPLS domain.

6. MPLS label

   Merupakan deretan bit informasi yang ditambahkan pada *header* suatu paket data dalam MPLS. Label MPLS atau yang disebut juga MPLS header ini terletak di antara *header* layer 2 dan *header* layer 3.

7. MPLS *node*

   Node yang menjalankan MPLS. MPLS *node* ini sebagai *control protocol* yang akan meneruskan paket berdasarkan label. Dalam hal ini MPLS *node* merupakan sebuah *router*.

8. *Forward Equivalance Class* (FEC)

   Merupakan representasi dari beberapa paket data yang akan diklasifikasikan berdasarkan kebutuhan *resource* yang sama di dalam proses pertukaran data.

9. *Label Distribution Path* (LDP)

   Merupakan protokol yang berfungsi untuk mendistribusikan informasi yang ada pada label ke setiap LSR pada MPLS. Protokol ini digunakan untuk memetakan FEC ke dalam label untuk selanjutnya akan dicapai untuk menentukan LSP. LDP message dapat dikelompokkan menjadi:

   a. *Discovery Messages*, yaitu pesan yang memberitahukan dan memelihara hubungan dengan LSR yang baru tersambung ke MPLS.

b. *Session Messages*, yaitu pesan untuk membangun, memelihara dan mengakhiri sesi antara titik LDP.
c. *Advertisement Messages*, yaitu pesan untuk membuat, mengubah dan menghapus pemetaan label pada MPLS.
d. *Notification Messages*, yaitu pesan yang menyediakan informasi bantuan dan sinyal informasi jika terjadi *error* [14].

Struktur label MPLS memiliki struktur tertentu dengan data sebesar 32-bit seperti yang ditunjukkan pada Tabel 2.2.

Tabel 2.2 Label MPLS [14]

0 1 2 3			
01234567890123456789012345678901			
LABEL	EXP	BoS	TTL

1. Label: Merupakan *field* yang terdiri dari 20 bit pertama pada label MPLS berupa Label Nilai dan nilai 16 pertama yang dibebaskan untuk penggunaan normal dikarenakan untuk penggunaan khusus. Nilai label tersebut contohnya alamat IP, besar data, jenis data dan lain-lain. Sistem mempelajari hop berikutnya dan operasi yang akan dilakukan, setelah menerima paket berlabel dan nilai label di bagian atas.
2. *Experimental Use* (EXP): bit ini dimulai dari 20-22 yang dicadangkan untuk penggunaan eksperimental, dan digunakan hanya untuk QoS atau dapat juga merupakan hasil salinan dari bit – bit IP *precedence* pada paket IP. Secara teknis *field* ini digunakan untuk keperluan eksperimen.
3. BOS: Bit 23 dikenal sebagai *Bottom of Stack bit*. *Field* ini digunakan untuk mengetahui *label stack* yang paling bawah. *Label stack* paling bawah memiliki nilai bit 1 dan yang lain diberi nilai 0. Hal ini sangat diperlukan pada proses *label stacking*. *Stack* adalah kumpulan label dan

dapat terdiri dari satu label atau beberapa label. *Label stacking* memiliki struktur tertentu seperti ditunjukkan pada Tabel 2.3.

Tabel 2.3 Label Stacking [14]

Label	EXP	0	TTL
Label	EXP	0	TTL
……..			
Label	EXP	1	TTL

2.2.2 *Internet Protocol Address* (Alamat IP)

Internet Protocol adalah suatu protokol jaringan yang umumnya dijalankan bersama protokol TCP/IP. Untuk berkomunikasi dengan perangkat lain melalui TCP/IP di internet, setiap perangkat yang terhubung harus memiliki alamat IP sebagai tanda identifikasi. Oleh karena itu, penting bahwa alamat IP bersifat unik dan tidak boleh ada duplikasi alamat IP yang digunakan oleh dua perangkat berbeda (*host*) dalam suatu jaringan. IANA (*Internet Assigned Number Authority*) bertanggung jawab mengatur penggunaan alamat IP di seluruh dunia. Penulisan alamat IP mengikuti format A.B.C.D, dimana setiap huruf terdiri dari angka 8 bit dengan nilai yang berkisar antara 0 hingga 255. Sebagai contoh, alamat IP untuk komputer yang tidak terhubung ke jaringan adalah 127.0.0.1 juga dikenal sebagai *localhost*. Ada dua jenis alamat IP, yaitu *public* dan *private*. Alamat IP *public* digunakan untuk komunikasi di jalur umum dan harus melalui proses registrasi oleh organisasi yang menangani distribusi IP, untuk mencegah adanya duplikasi alamat IP pada dua perangkat *host* yang berbeda. Sebaliknya, alamat IP *private* digunakan dalam jaringan lokal dan tidak memerlukan proses registrasi [15].

Dengan beberapa pertimbangan terkait efisiensi alamat IP, penyelesaian masalah topologi jaringan dan kebutuhan organisasi, administrator jaringan seringkali melakukan *subnetting*. Esensi dari *subnetting* adalah memindahkan garis pemisah antara bagian *network* dan

bagian *host* dari suatu alamat IP. Beberapa bit dari bagian *host* dialokasikan sebagai bit tambahan pada bagian jaringan. Sebuah alamat jaringan sesuai dengan struktur standar, dipecah menjadi sejumlah *subnetwork*. Pendekatan ini menciptakan beberapa jaringan tambahan dengan mengurangi jumlah maksimum *host* yang dapat diakomodasi di setiap jaringan tersebut. Salah satu tujuan lain dari *subnetting* yang tidak kalah penting adalah mengurangi tingkat kepadatan (kongesti) dalam suatu jaringan. Perlu dicatat bahwa secara logika, suatu jaringan merujuk pada host-host yang terhubung ke suatu jaringan fisik.

2.2.2.1 Konsep Subnetting IP *Address*

Untuk menghindari terjadinya kongesti akibat terlalu banyak *host* dalam suatu *physical network* dilakukan segmentasi jaringan. *Subnetting* juga diterapkan untuk menangani perbedaan *hardware* dan media fisik yang digunakan dalam sebuah jaringan. Router IP memiliki kemampuan untuk mengintegrasikan berbagai jaringan dengan media fisik yang berbeda, asalkan setiap jaringan memiliki alamat jaringan yang unik. Selain itu, melalui *subnetting*, seorang administrator jaringan dapat menetapkan alokasi alamat IP untuk *host* di setiap departemen dalam suatu perusahaan besar, memfasilitasi pengaturan keseluruhan jaringan. Setelah melakukan *subnetting* secara fisik, langkah berikutnya adalah membuat subnet secara logis. Setiap subnet fisik dalam setiap departemen harus memiliki subnet logis yang berbeda, yang terdiri dari alamat jaringan yang merupakan bagian (*sub*) dari alamat jaringan perusahaan.

Tabel 2.4 Contoh Subnet Mask [15]

No	Subnet Mask (Biner)	Desimal	Hexa	Tingkat
1	11111111.11111111. 00000000.00000000	255.255.0.0	FF.FF.00.00	16 bit
2	11111111.11111111. 11111111.00000000	255.255.255.0	FF.FF.FF.00	24 bit
3	11111111.11111111	255.255.255.128	FF.FF.FF.80	25 bit

No	Subnet Mask (Biner)	Desimal	Hexa	Tingkat
	11111111.10000000			
4	11111111.11111111. 11111111.11000000	255.255.255.192	FF.FF.FF.C0	26 bit
5	11111111.11111111. 11111111.11100000	255.255.255.224	FF.FF.FF.E0	27 bit

Tabel 2.4 menunjukkan beberapa contoh *subnet mask*. Suatu *subnet* didefinisikan dengan menerapkan pembatasan bit (*subnet mask*) pada alamat IP. *Subnet mask* memiliki struktur yang sama dengan alamat IP, yaitu terdiri dari 32bit yang dibagi menjadi 4 segmen. Bentuk *subnet mask* mengikuti pola urutan bit 1 diikuti oleh bit 0. Jumlah bit 1 pada *subnet mask* menentukan tingkat *subnet mask* tersebut.

Penerapan *subnet mask* pada suatu alamat IP memiliki peran krusial dalam mengkonfigurasi dan mengidentifikasi bagian jaringan dari alamat tersebut. *Subnet mask*, yang terdiri dari serangkaian bit 1 dan 0, memainkan peran penting dalam mengaktifkan atau menonaktifkan segmen tertentu dari alamat IP, menentukan batas antara bagian jaringan dan bagian *host*. Bit 1 pada *subnet mask* menunjukkan pengaktifan, sementara bit 0 menunjukkan ketidakaktifan. Dengan cara ini, bit-bit pada alamat IP yang "ditutupi" oleh bit-bit *subnet mask* yang aktif akan diinterpretasikan sebagai bagian dari jaringan, memberikan struktur yang jelas dan terdefinisi untuk pengelompokan dan manajemen alamat IP pada suatu subnet.

Penetapan subnet dilakukan saat mengkonfigurasi *interface* yang menghubungkan router atau *host* ke dalam jaringan. Perintah yang diperlukan dapat bervariasi tergantung pada jenis komputer, sistem operasi, dan router yang sedang digunakan. Setelah subnet dikonfigurasi dengan benar, langkah selanjutnya adalah menjalankan protokol *routing* yang terdapat dalam TCP/IP untuk mengaktifkan *routing* secara otomatis dalam jaringan. Beberapa protokol *routing* yang digunakan dalam jaringan TCP/IP termasuk RIP (*Routing Information Protocol*), yang beroperasi di dalam suatu *autonomous system*.

2.2.3 *Internet Protocol Version 4* (IPv4)

Alamat IP versi 4 (IPv4) pada awalnya adalah sederet bilangan biner sepanjang 32bit yang dipakai untuk mengindentifikasikan *host* pada jaringan, Alamat IP ini diberikan secara unik pada masing-masing komputer/*host* yang terhubung ke internet. Prinsip kerjanya adalah paket yang membawa data dimuati alamat IP dari komputer pengirim data kepada alamat IP dari komputer yang akan dituju, kemudian data tersebut dikirim ke jaringan. Paket ini kemudian dikirim dari *router* ke *router* dengan berpedoman pada alamat IP tersebut menuju ke komputer yang dituju, Seluruh komputer/*host* yang tersambung ke internet, dibedakan hanya berdasarkan alamat IP ini, oleh karena itu tidak boleh terjadi duplikasi pada alamat IP untuk setiap komputer yang terhubung ke jaringan internet.

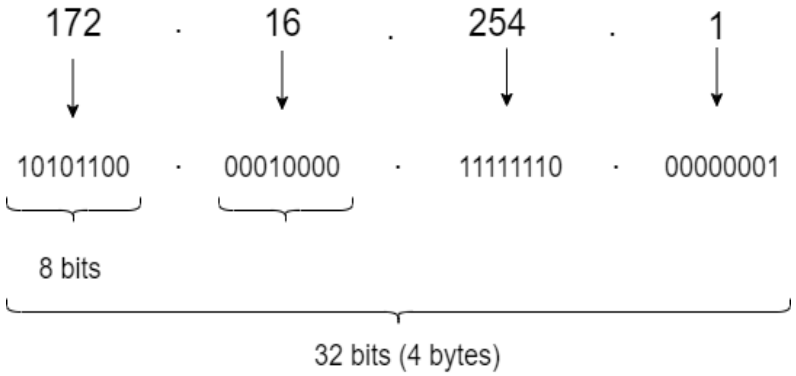
Pada IPv4 terdapat pembagian kelas pada protokol yang mencakup empat kelas utama, yaitu kelas A, kelas B, kelas C dan kelas D. Setiap kelas memiliki rentang alamat IP yang dapat digunakan dan membedakan berdasarkan jumlah bit yang dialokasikan untuk identifikasi jaringan dan identifikasi *host*. Selain pembagian kelas, penggunaan CIDR (*Classless Inter-Domain Routing*) juga diperkenalkan untuk memberikan fleksibilitas dalam alokasi alamat IP.

Alamat-alamat IP panjangnya 32bit dan dibagi menjadi dua identifikasi sebagai berikut:

1. Bagian identifikasi net ID menunjukkan identitas jaringan komputer tempat *host* (komputer) dihubungkan.
2. Bagian identifikasi *host* ID memberikan suatu pengenal unik pada setiap *host* (komputer) pada suatu jaringan komputer [16].

Alamat IPv4 umumnya ditulis dalam notasi desimal bertitik (*dotted-decimal notation*), yang dibagi menjadi empat oktet dengan masing-masing oktet memiliki ukuran 8-bit. Dalam representasi ini, nilai-nilai di setiap oktet berkisar antara 0 hingga 255, sesuai dengan ukuran bitnya. Pengalamatan IPv4 terdiri dari total 32 bit, dan setiap bitnya dipisahkan dengan notasi titik. Gambar 2.3 menunjukkan notasi pengalamatan IPv4

mencakup empat kelompok angka yang dipisahkan oleh titik, seperti: 192.168.0.1. Sistem ini memfasilitasi identifikasi dan pengelompokan alamat IP dalam jaringan dengan cara yang jelas dan terstruktur.



Gambar 2.3 Pengalamatan pada IPv4 [17]

Contoh notasi pengalamatan IPv4:

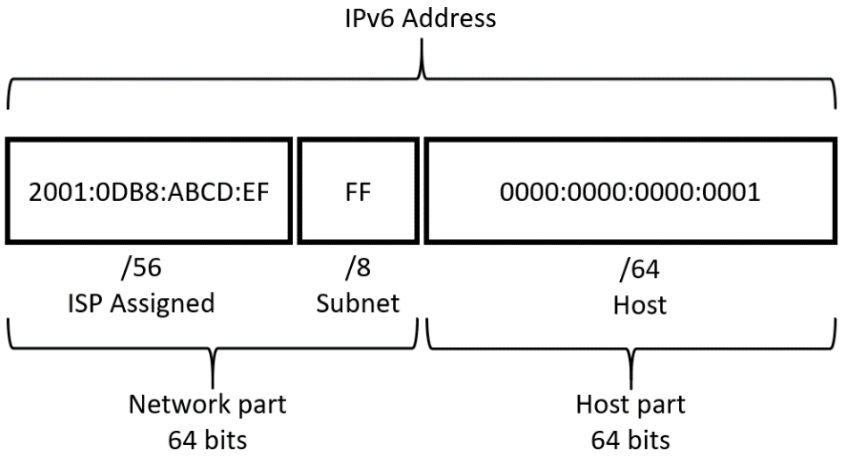
FFFFFFFF.FFFFFFFF.FFFFFFFF.FFFFFFF

Nilai F dirubah menjadi nilai biner (1 dan 0) 10101100.00010000.11111110.00000001. Sehingga jika dirubah ke dalam desimal menjadi 172.16.254.1.

2.2.4 *Internet Protocol Version* 6 (IPv6)

Internet Protocol version 6 (IPv6) atau yang sering disebut juga sebagai IPng (*Internet Protocol next generation*) adalah suatu protokol *layer* ketiga terbaru yang diciptakan untuk menggantikan IPv4 atau yang sering dikenal sebagai IP. Alasan utama dari penciptaan *Internet Protocol version* 6 (IPv6) ini adalah untuk mengoreksi masalah pengalamatan pada versi 4 (IPv4). Karena kebutuhan akan alamat internet semakin banyak, maka IPv6 diciptakan dengan tujuan untuk memberikan pengalamatan yang lebih banyak dibandingakan dengan IPv4, sehingga perubahan pada IPv6 masih berhubungan dengan pengalamatan IP sebelumnya. Konsep pengalamatan pada IPv6 memiliki persamaan dengan IPv4, akan tetapi lebih diperluas dengan tujuan untuk menciptakan pengalamatan yang bisa mendukung perkembangan internet yang semakin pesat dan penggunaan aplikasi baru di

masa depan. Perubahan terbesar pada IPv6 adalah terdapat pada *header*, yaitu peningkatan jumlah alamat dari 32bit (IPv4) menjadi 128 bit (IPv6) [16].



Gambar 2.4 Pengalamatan pada IPv6 [17]

Gambar 2.4 menunjukkan pengalamatan pada IPv6. Protokol IPv6 menyediakan ruang alamat sebesar 128bit yaitu 4 kali lipat ruang alamat yang disediakan IPv4. Format alamat yang ada pun berbeda dengan format alamat pada IPv4. Berbeda dengan IPv4, IPv6 yang disediakan sebagai pengenal pada 1 atau lebih *interface* dibedakan atas 3 tipe yaitu:

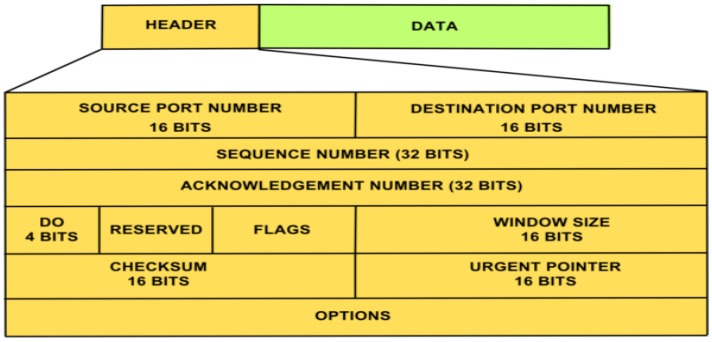
1. *Unicast address* merupakan alamat yang hanya menyediakan komunikasi secara poin-to-point, secara langsung antara dua host dalam sebuah jaringan. Digunakan sebagai pengenal untuk 1 *Network Interface Card*, dimana paket data yang dikirim ke *unicast address* hanya dikirim ke *Network Interface Card* yang bersangkutan saja. Pada alamat unicast terdiri dari :
   a. Global merupakan alamat yang hanya digunakan oleh provider dan alamat geografis.
   b. *Link local address* merupakan alamat yang dipakai di dalam satu *link*. *Link* merupakan jaringan lokal yang saling terhubung dalam satu level.
   c. *Site-local* adalah alamat yang setara dengan *private address* dan terbatas digunakan hanya dalam *site* saja.

2. *Anycast address* merupakan sebuah alamat yang diberikan pada beberapa host, untuk mendefinisikan kumpulan node. Jika ada paket yang dikirim ke alamat ini, maka *router* akan mengirim paket tersebut ke *host* terdekat yang memiliki *anycast address* sama, namun pada *anycast address* tidak disediakan ruang khusus. Jika terhadap beberapa *host* diberikan sebuah alamat yang sama, maka alamat tersebut dianggap sebagai *anycast address*. *Anycast address* digunakan sebagai pengenal untuk beberapa *Network Interface Card* sekaligus, dimana paket data yang dikirim ke *anycast address* akan dikirim ke salah satu *Network Interface Card*.
3. *Multicast address* merupakan alamat yang digunakan untuk komunikasi satu lawan banyak dengan menunjuk *host* dari *group*. *Multicast address* ini pada IPv4 didefinisikan sebagai kelas D, sedangkan pada IPv6 ruang yang 8bit pertamanya di mulai dengan "FF" disediakan untuk *multicast address*. Ruang ini kemudian dibagi-bagi lagi untuk menentukan range berlakunya. Kemudian *broadcast address* pada IPv4 yang alamat bagian *host*-nya didefinisikan sebagai "1", pada IPv6 sudah termasuk di dalam *multicast address* ini. *Multicast address* ini digunakan pengenal untuk beberapa *Network Interface Card* sekaligus, dimana paket data yang dikirim ke *multicast address* akan dikirim ke semua *Network Interface Card* yang bersangkutan.

2.2.5 Protokol TCP (*Transmission Control Protocol*)

Dalam penggunaan internet dan secara umum pada jaringan TCP/IP, setiap aplikasi berkomunikasi dengan menggunakan protokol pendukung. TCP menggunakan komunikasi byte-stream, yang berarti bahwa data dinyatalan sebagai suatu urutan-urutan byte. Protokol ini berada di dalam lapisan *transport* (*transport layer*) dalam standar OSI yang bertanggung jawab untuk memberikan efisiensi dan jaminan komunikasi *end-to-end*. TCP (*Transmission Control Protocol*) merupakan protokol yang dapat diandalkan dan dirancang khusus untuk mengelola alur data dalam jaringan internet yang seringkali dihadapi dengan kondisi yang tidak dapat diandalkan. TCP juga dirancang untuk beradaptasi dengan berbagai masalah

peralatan jaringan. Protokol TCP dirancang dengan pendekatan *connection-oriented* dalam pengiriman data, yang berarti terdapat suatu koneksi yang stabil sebelum data dapat dikirim. TCP menjamin keandalan data dengan menggunakan metode ARQ (*Automatic Repeat request*). ARQ secara otomatis mentransmisikan data ulang berdasarkan informasi kegagalan pengiriman yang diterima dari penerima data yang disebut ACK (*Acknowledgment*) [18].



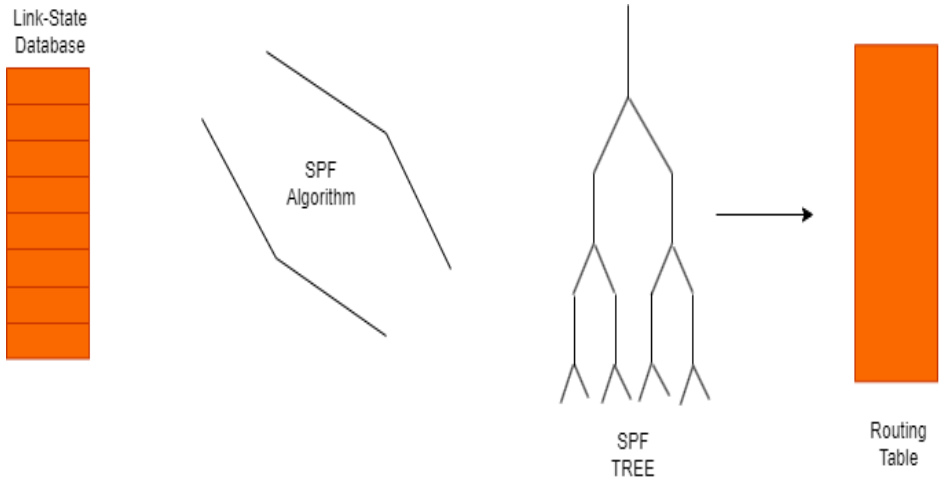
Gambar 2.5 Format Header TCP [18]

Format header TCP pada gambar 2.5 memiliki komponen sebagai berikut:

1. *Source* dan *destination port* didefinisikan sebagai nomor port dari sebuah program aplikasi yang digunakan oleh TCP.
2. *Sequence number* mengidentifikasi *byte* pertama dari data dalam suatu segment.
3. *Acknowledgment number* berfungsi untuk mengumumkan data yang sukses diterima.
4. *Header length* mengindikasikan nomor dari 32-bits kata dalam *header*.
5. *Control* mendefinisikan segmen yang digunakan atau melayani sebagai sebuah cek validasi untuk *field* lainnya.
6. *Window size* mendefinisikan ukuran dari *sliding window* yang digunakan dalam *flow control*.
7. *Checksum* digunakan sebagai deteksi kesalahan.
8. *Urgent pointer* mendefinisikan loncatan antara *urgent* data dan data normal.

2.2.6 *Open Shortest Path First* (OSPF)

OSPF merupakan suatu protokol *routing dynamic* yang dikembangkan untuk jaringan atau *network* berbasis IP oleh *Internet Engineering Task Force* (IETF). OSPF juga merupakan suatu protokol *routing link state* yang diterapkan pada suatu *intra autonomus system*. Algoritma yang digunakan pada *routing* OSPF adalah metode *link state*. OSPF menggunakan *link state* yang dibentuk untuk bekerja secara tepat berdasarkan pengiriman update informasi rute. Algoritma OSPF biasanya menggunakan metode *link state* untuk mengkalkulasi *shortest path* pada setiap *destination node*. Pada protokol *routing link state* mempunyai informasi yang lengkap dan akurat tentang *network* yang akan dilalui di semua *router* [19].



Gambar 2.6 Skema *Routing* OSPF [19]

Gambar 2.6 menunjukkan skema *routing* OSPF, dimana router menjalankan atau melakukan *broadcast* mengenai informasi *routing* kepada setiap *router* dalam suatu *Autonomus System*. Jika terjadi perubahan dalam informasi *routing* maka *router* mengirimkan *broadcast link-state* ke setiap router lainnya jika terdapat perubahan *cost* dan status *network* yang berubah.

Protokol OSPF menggunakan konsep hirarki *routing*, yang dimana protokol akan membagi area jaringan ke dalam beberapa dimensi area. Ini akan sangat berpengaruh dalam pengiriman data karena dengan hirarki ini akan lebih memudahkan dalam keteraturan pengiriman data. Data yang akan

dikirim tidak akan disebar kesana kemari karena sudah tersegmentasi. Dan juga penggunaan bandwidth akan lebih efisien karena sudah ada keteraturan dalam distribusi peroutingan.

Paket *hello* merupakan salah satu cara bagi protokol OSPF untuk mengumpulkan informasi. Ini berupa pengidentifikasian informasi *interface* sekitarnya untuk membangun sebuah hubungan sebuah *update routing* yang berguna bagi pertukaran data. Selanjutnya masuk ke dalam fase *exstart*, yaitu mempertukarkan database inisial. Peroutingan terjalin setelah mendapatkan *acknowledgment* atau *ack*. Selama waktu *loading*, *router* akan mengolah tabel peroutingan. OSPF memiliki 3 tabel di dalam *router*:

1. *Routing table* biasa juga disebut sebagai *Forwarding database*. *Database* ini berisi *the lowest cost* untuk mencapai *router-router*/*network-network* lainnya. Setiap *router* mempunyai *routing table* yang berbeda-beda.
2. *Adjecency database*, *database* ini berisi semua *router* tetangganya. Setiap *router* mempunyai *adjacency database* yang berbeda-beda.
3. *Topological database*, *database* ini berisi seluruh informasi tentang *router* yang berada dalam satu *network*nya/areanya.

OSPF sudah mengeluarkan beberapa versi, seperti OSPFv1, OSPFv2 dan yang baru OSPFv3. Perbedaan yang paling mendasar dari OSPFv2 dengan OSPFv3 adalah dari internet protokol nya. OSPFv3 menggunakan basis IPv6 sedangkan OSPF versi sebelumnya menggunakan IPv4 [20]. Beberapa perbedaan paling menonjol dari kedua versi OSPF yakni OSPFv2 dan OSPFv3 terdapat pada Tabel 2.5.

Tabel 2.5 Perbedaan OSPFv2 dengan OSPFv3 [20]

	OSPFv2	*OSPFv3*
Advertises	*IPv4 networks*	*IPv6 prefixes*
Source address	*IPv4 source address*	*IPv6 link-local address*
authentication	*Plain text and MD5*	*IPv6 authentication*

	OSPFv2	*OSPFv3*
Advertise networks	*Configured using the network router configuration command*	*Configured using the IPv6 OSPF process-id area area-id interface configuration command*
Destination address	*Choice of:* *1. Neighbor IPv4 unicast address* *2. 224.0.0.5 all-OSPF-routers multicast address* *3. 224.0.0.6 DR/BDR multicast address*	*Choice of:* *1. Neighbor IPv6 link-local address* *2. FF02;;5 all-OSPFv3-routers multicast address* *3. FF02;;6 DR/BDR multicast address*
IP unicast routing	*IPV4 unicast routing is enabled by default*	*IPv6 unicast forwarding is not enabled by default. The IPv6 unicast-routing global*
		configuration command must be configured

2.2.6 *Quality of Service* (QoS)

Quality of Service (QoS) didefinisikan sebagai suatu pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari suatu layanan. Kinerja jaringan komputer dapat bervariasi akibat beberapa masalah, seperti halnya masalah QoS mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada trafik jaringan tertentu melalui teknologi yang berbedabeda. Tujuan dari QoS adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama. Teknologi QoS ini adalah teknologi yang memungkinkan administrator jaringan untuk dapat

menangani berbagai efek akibat terjadinya kemacetan pada lalu lintas aliran paket dari berbagai layanan yaitu dengan mengatur dan memberikan prioritas pada jaringan tersebut, ini akan mengoptimalkan aplikasi yang kritis atau yang memiliki delay sensitif untuk dapat berjalan sebagaimana mestinya. Dengan implementai QOS, *network* administrator akan memiliki fleksibilitas yang tinggi untuk, mengontrol aliran dan kejadian-kejadian yang ada di aliran paket pada jaringan. Kualitas layanan (*Quality of Service*) berdasarkan sudut pandang jaringan adalah kemampuan suatu elemen jaringan, seperti aplikasi jaringan, *host*, atau *router* untuk memiliki tingkatan jaminan bahwa elemen jaringan tersebut dapat memenuhi kebutuhan suatu layanan [21].

Terdapat beberapa parameter QoS (*Quality of Service*), yaitu sebagai berikut:

1. Parameter *Throughput*

   Throughput yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam bps. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Secara umum terdapat lima kategori penurunan performansi jaringan berdasarkan nilai *Throughput* sesuai dengan standar TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*) yaitu seperti tampak pada Tabel 2.6.

Tabel 2.6 Standar *Throughput*

Kategori	*Throughput*	*Index*
Buruk	0 - 338 kbps	0
Kurang bagus	338 - 700 kbps	1
Cukup	700 - 1200 kbps	2
Bagus	1.200 kbps – 2,1 Mbps	3
Sangat Bagus	>2,1 Mbps	4

Rumus untuk menghitung nilai *throughput* ditunjukkan pada persamaan (2.1), dimana:

$$Throughput = \frac{Jumlah\ Bit\ Yang\ Dikirim}{Total\ Waktu\ Pengiriman} \quad \text{..... (2.1)}$$

2. Parameter *Delay*

Delay (*latency*) adalah waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari satu titik menuju titik lain yang menjadi tujuannya. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama. Waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari satu titik ke titik lain yang menjadi tujuannya. *Delay* diperoleh dari selisih waktu kirim antara satu paket TCP dengan paket lainnya yang direpresentasikan dalam satuan *second*. Menurut versi TIPHON, besarnya *delay* dapat diklasifikasikan seperti pada tabel 2.7.

Tabel 2.7 Standar *Delay*

Kategori	Delay	*Index*
Sangat Bagus	< 150 ms	4
Bagus	150 - 300 ms	3
Cukup	300 - 450 ms	2
Buruk	> 450 ms	1

Rumus untuk menghitung nilai *delay* seperti tertera pada persamaan (2.2), dimana:

Rata-Rata *Delay* = $\frac{Total\ Delay}{Total\ Paket\ Yang\ Diterima}$ ..... (2.2)

3. Parameter *Packet Loss*

Merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, hal ini dapat terjadi karena beberapa kemungkinan antara lain terjadinya *overload* di dalam suatu jaringan, tabrakan (*congestion*) dalam jaringan, *error* yang terjadi pada media fisik, kegagalan yang terjadi pada sisi penerima antara lain bisa disebabkan karena *router buffer over flow* atau kemacetan. Di dalam implementasi jaringan, nilai *packet loss* ini diharapkan mempunyai nilai yang minimum. Secara umum terdapat empat kategori penurunan performansi jaringan berdasarkan nilai *packet loss* sesuai dengan versi

TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*) yaitu seperti tampak pada Tabel 2.8.

Tabel 2.8 Standar *Packet Loss*

Kategori	*Packet Loss*	*Index*
Sangat Bagus	0 – 2 %	4
Bagus	3 – 14 %	3
Cukup	15 – 25 %	2
Buruk	>25 %	1

Rumus untuk menghitung *packet loss* ditunjukkan pada persamaan (2.3), dimana:

$$Packet\ Loss = \frac{Data\ Yang\ Dikirim - Data\ Yang\ Diterima}{Paket\ Data\ Yang\ Dikirim}\ x\ 100\% \quad \ldots\ldots\ldots (2.3)$$

4. Parameter *Jitter*

Jitter merujuk pada variasi waktu antara paket dalam jaringan dan dipengaruhi oleh beban serta kepadatan lalu lintas dalam jaringan. *Jitter* dapat diartikan sebagai gangguan dalam komunikasi, baik dalam konteks digital maupun analog, yang disebabkan oleh perubahan sinyal terhadap referensi posisi waktu. *Jitter* memiliki potensi untuk menyebabkan kehilangan data, terutama pada pengiriman data dengan kecepatan tinggi. Variasi *jitter* seringkali terkait dengan *delay*, *delay* menunjukkan variasi yang signifikan dalam transmisi data pada jaringan [22]. Standar *jitter* menurut TIPHON dapat dilihat pada Tabel 2.9.

Tabel 2.9 Standar *Jitter*

Kategori	*Jitter*	*Index*
Sangat Bagus	0 ms	4
Bagus	1 - 75 ms	3
Cukup	75 – 125 ms	2
Buruk	125 – 225 ms	1

Rumus untuk menghitung *jitter* seperti tertera pada persamaan (2.4), dimana:

$$Jitter = \frac{Total\ Variasi\ Delay}{Total\ Paket\ Data\ Yang\ Diterima} \quad \text{..... (2.4)}$$

2.2.6 *Graphical Network Simulator* (GNS3)

GNS3 adalah sebuah program *graphical network simulator* yang dapat mensimulasikan topologi jaringan yang lebih kompleks dibandingkan dengan simulator lainnya. Program ini dapat dijalankan pada sistem operasi *windows* atau *linux*. Banyak sekali jenis router dan *firewall* yang didukung oleh GNS3 antara lain Mikrotik, *Cisco*, Juniper dan lain sebagainya. Walaupun belum bisa menggantikan seluruh perangkat keras, namun fitur yang ditawarkan cukup beragam dan simulasinya mendekati perangkat asli karena bisa mengkombinasikan antara perangkat virtual dan perangkat asli. GNS3 menjadi pilihan karena fiturnya yang banyak, GNS3 mampu mensimulasikan jaringan yang komplek dan mendekati riil. Dengan fiturnya yang begitu lengkap, GNS3 juga memiliki kelemahan yaitu membutuhkan spesifikasi *hardware* yang tinggi, semakin banyak perangkat yang digunakan dalam simulasi maka dibutuhkan spesifikasi *hardware* yang tinggi pula.

Software GNS3 memerlukan dua aplikasi tambahan untuk dapat berjalan, yaitu *dynampis* dan *dynagen*.

1. *Dynampis* merupakan *software* yang dapat mengemulasikan *router cisco* seri 1700, 2600, 3600, 3700 dan 7200 *hardware*. *Dynampis* dikembangkan dengan maksud untuk keperluan *testing* dan eksperimen dan menguji kualitas konfigurasi IOS pada router secara *real*. *Software* ini berbasis CLI dan tidak memiliki mode GUI sehingga harus memahami perintah-perintahnya. *Dynamips* mampu berjalan dibeberapa sistem operasi seperti *linux* dan *windows* fitur-fitur IOS dan menguji kualitas konfigurasi-konfigurasi sebelum diterapkan pada router sebenarnya. Aplikasi ini menggunakan metode teks dan tidak memiliki antar muka grafis, sehingga diperlukan aplikasi dynagen sebagai *front end* bagi dynampis [23].

2. *Dynagen* merupakan program end-to-end untuk *dynampis* yang berfungsi untuk menyederhanakan konfigurasi *dynampis*.

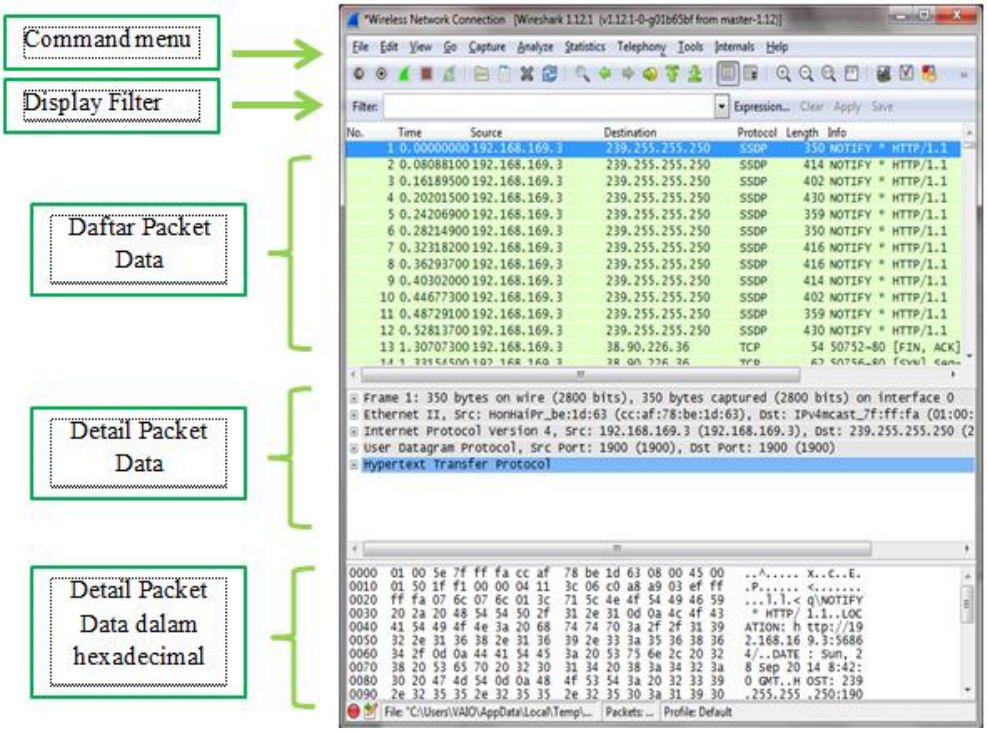
Untuk membuat suatu simulasi jaringan di GNS3 terkadang kita memerlukan keberadaan *end user device* untuk keperluan test koneksi *end to end* sehingga simulasi routing menjadi terasa lebih realistis. Qemu merupakan aplikasi emulator yang mengandalkan translasi *binary* untuk mencapai kecepatan yang layak saat berjalan di arsitektur komputer *host*. Dalam hubungannya dengan komputer *host*, Qemu menyediakan satu perangkat model yang memungkinkan untuk menjalankan berbagai *system* operasi yang belum dimodifikasi sehingga dapat ditampilkan dalam *hosted virtual machine monitor*. Qemu juga dapat memberikan dukungan percepatan modus campuran *binary translation* (untuk kernel *code*) dan *native execution* (untuk *user code*).

2.2.7 Wireshark

Wireshark merupakan perangkat lunak yang spesifik untuk melakukan analisa paket data pada jaringan secara *real time* dan menampilkan hasil analisa paket data tersebut dalam format yang dipahami oleh pengguna. *Wireshark* dapat melakukan paket *filtering*, paket *color coding*, dan fitur-fitur lain yang dapat mengizinkan untuk melihat detail *network traffic* dan inspeksi paket data secara individu. Aplikasi *wireshark* ini merupakan aplikasi yang dapat digunakan secara gratis karena aplikasi ini berbasis *open source* atau sumber yang terbuka. Aplikasi *wireshark* ini dapat dijalankan dibanyak *platform* diantaranya yaitu adalah *linux, windows* dan *mac* [24]. Dengan segala kemampuan yang dimilikinya, *wireshark* digunakan oleh *network* professional untuk keperluan analisis, *troubleshooting*, pengembangan *software* dan protokol, serta digunakan juga untuk tujuan edukasi.

Wireshark adalah *tools* yang berfungsi untuk menganalisa data terhadap lalu lintas jaringan dan mampu berfungsi meng-*capture packet* atau *frame* yang lewat dalam suatu *network* atau jaringan ketika melakukan proses pengujian terhadap jaringan. Proses *capture paket* dapat dilakukan

ketika suatu topologi jaringan berhasil dan terkoneksi dengan baik. Hasil *capture* paket dengan *tools* ini terlihat seperti pada gambar 2.7.



Gambar 2.7 Hasil *Capture* dengan *Wireshark* [24]

Keterangan:

1. *Command* Menu: terdapat berbagai menu yang dapat digunakan pada saat monitoring.
2. *Display filter*: kolom untuk mengisi sintaks-sintaks untuk memfilter paket data apa saja yang akan ditampilkan pada list paket.
3. Daftar paket: menampilkan paket-paket yang berhasil ditangkap oleh *wireshark*, berurutan mulai dari paket pertama yang ditangkap dan seterusnya.
4. Detail paket: sebuah paket tentunya membawa informasi tertentu yang berbeda-beda antar paketnya, disini akan ditampilkan detail paket yang terpilih pada daftar paket diatasnya.
5. Detail heksa: detail paket ini akan ditampilkan dalam bentuk heksa.
6. *Time*: menampilkan waktu saat paket tersebut ditangkap.
7. *Source*: menampilkan IP sumber dari paket data tersebut.

8. *Destination*: menampilkan tujuan dari paket data tersebut.
9. *Protocol*: menampilkan protokol apa saja yang dipakai sebuah paket data.
10. Info: menampilkan informasi detail tentang paket data tersebut.

Wireshark sebagai *network analyzer* memiliki kemampuan untuk melakukan pengamatan terhadap berbagai elemen seperti alamat IP, protokol jaringan, jumlah *frame*, sumber dan tujuan paket, serta dapat melakukan fungsi *filtering* dan *sniffing*. *Wireshark* memungkinkan pengguna untuk memonitor data dari jaringan yang sedang beroperasi atau dari data yang tersimpan di lokasi *disk*. Dengan alat ini, pengguna dapat segera melihat atau menyortir data yang ter-*captured*, mulai dari informasi singkat hingga rincian lengkap, termasuk di dalamnya *header* lengkap dan jumlah data yang dapat diperoleh dari paket tersebut. Fitur-fitur tersebut memberikan pengguna kemampuan untuk mendapatkan wawasan mendalam terkait lalu lintas jaringan dan aktivitas komunikasi yang terjadi.