BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian [1] membahas mengenai *load balancing* algoritma *weighted least connection* pada *controller pox* menggunakan agen psutil. Pada *Software Defined Network*. Tujuan dari penelitian ini untuk mengetahui apakah *load balancing weighted least connection* menggunakan agen psutil lebih baik dari *load balancing least connection* menggunakan agen psutil. Pengujian dilakukan dengan menggunakan skenario *client* 1 hingga *client* 3 yang mengirimkan *request* kepada 4 *server* dengan jumlah permintaan yang berbeda-beda. Selanjutnya melakukan pengujian kinerja menggunakan parameter *response time, distribusi traffic, memory utilization*, dan *cpu utilization* dengan jumlah *request* dimulai dari 100 *req*, 300 *req*, dan 500 *req*. Hasil pada penelitian ini menunjukan bahwa kinerja dari algoritma *weighted least connection* berbasis agen dan *flow* lebih baik debandingkan dengan algoritma *least connection* berbasis agen.

Penelitian [2] membahas mengenai performa dari algoritma *least connection* dan *round robin* yang diterapkan pada *load balancing server e-learning*. Tujuan dari penelitian ini untuk meningkatkan kinerja *server e-learning* dengan melakukan perbandingan berdasarkan kondisi sebelum dan setelah *load balancing* diimplementasikan. Penelitian ini dilakukan dengan cara melakukan pengujian dengan melakukan permintaan layanan yang akan dilakukan secara bertahap. Dimulai dari jumlah koneksi 500/10 *sec*, 500/20 *sec*, dan 500/30 *sec* yang dilakukan koneksi ke *server* secara bersamaan dalam satu waktu. Pengamatan pada pengujian ini dilakukan dengan menggunakan parameter *throughput* dan *response time*.

Penelitian [3] membahas mengenai perbandingan performa dari algoritma *round robin* dan *least connection* pada *web server nginx*. Penelitian ini bertujuan untuk mengetahui perbandingan performa dari setiap algoritma yang digunakan dan nilai rata-rata setiap parameter yang diberikan. Pengujian dilakukan dengan kondisi setiap *server* telah menjalankan *web server nginx* dan dengan kondisi tidak menjalankan program aplikasi lain (*idle*). Pengujian terhadap parameter *throughput*

dan *response time* sebanyak 4 kali dengan 3 kategori 30 req/s, 60 req/s dan 120 req/s. Hasil yang didapatkan pada parameter *throughput*, algoritma *least connection* lebih unggul menghasilkan nilai *throughput* yang lebih besar dari algoritma *round robin*. Semakin besar nilai *throughput* yang dihasilkan maka semakin baik *server* pada saat mengirimkan paket kepada *client*. Pada parameter *response time* algoritma *least connection* memiliki *response time* lebih rendah dari *round robin*, hal ini menunjukkan bahwa semakin rendah nilai *response time* maka semakin bagus *server* tersebut.

Penelitian [4] meneliti mengenai perbandingan performa metode algoritma *round robin* dan algoritma *weighted round robin* untuk *load balancing* pada *web server*. Tujuan dari penelitian ini adalah untuk mengetahui dan membandingkan nilai *response time* dari *load balancer* dengan metode *round robin* dan *weighted round robin*. Pengujian dilakukan dengan mengirim sampel pengujian kepada *server load balancing* sebanyak 10 sempel dan mengirimkan *request* kepada *server load balancing* melalui *tools apache benchmark*. Berdasarkan pengujian dengan 10 sampel dengan nilai *request* yang terus bertambah, hasil yang didapatkan yaitu apabila jumlah *request* yang dikirim kepada *load balancer* semakin besar, maka akan berpengaruh pada nilai *response time* yang semakin besar. Total perbandingan nilai *response time* antara metode *weighted round robin* dan *round robin* dengan spesifikasi *server* sama dan beban 1:2, menghasilkan persentase nilai *response time* total antara 13% sampai dengan 46%.

Penelitian [5] membahas mengenai perbandingan algoritma *least connection* dan koloni semut untuk *load balancing* pada SDN (*Software Defined Network*). Tujuan dari penelitian ini untuk mengetahui algoritma yang dapat bekerja secara optimal dalam optimasi pemerataan beban *traffic*. Pengujian ini dilakukan dengan membangkitkan *traffic* dari *host* 4 ke *host* 13 dan dari *host* 12 ke *host* 2 yang bertujuan agar *traffic* padat. Pengukuran dilakukan dengan menggunakan parameter *throughput*, *delay* dan *jitter* pada dari *host* 14 sebagai *client* ke *host* 1 sebagai *server*. Pada parameter *throughput*, algoritma *least connection* memiliki nilai efektivitas *load balancing* lebih tinggi dibandingkan algoritma optimasi koloni semut. Nilai *delay* yang didapat untuk algoritma optimasi koloni semut dan *least connection* pada simulasi ini lebih tinggi dibandingkan pada simulasi tanpa menerapkan *load*

balancing. Nilai pada parameter *jitter* terus menaik berbanding lurus dengan *background* yang diaktifkan.

Berdasarkan penjabaran kajian pustaka tersebut, penelitian terdahulu yang memiliki relevansi dengan penelitian yang akan dilakukan dijelaskan dalam tabel perbandingan yang tertera pada tabel 2.1:

Tabel 2.1 Tabel perbandingan Jurnal Referensi

Penelitian Oleh	**Tujuan**	**Metode**	**Parameter**
Sumbayak, Nurwarsito dan Primananda (2019) [1]	Mengetahui *load balancing weighted least connection* menggunakan agen psutil lebih baik dari *load balancing least connection* menggunakan agen psutil.	*Weighted Least Connection*	*Response time, distribusi traffic, memory utilization,* dan *cpu utilization.*
Riskiono, Pasha (2020) [2]	Meningkatkan kinerja *server e-learning* berdasarkan kondisi sebelum dan setelah *load balancing* diimplementasikan.	*Least connection* dan *round robin*	*Throughput* dan *response time.*
Ekmawan (2021) [3]	Mengetahui perbandingan performa dari setiap algoritma yang digunakan dan nilai rata-rata setiap parameter yang diberikan.	*Least connection* dan *round robin*	*Throughput, delay, jitter* dan *packet loss.*
Oktariyadi, Ruslianto dan Bahri (2021) [4]	Mengetahui dan membandingkan nilai *response time load balancer* metode *round robin* dan *weighted round robin.*	*Round robin* dan *weighted round robin*	*Throughput* dan *response time.*
Desa, Dewanta dkk (2021) [5]	Mengetahui algoritma yang dapat bekerja secara optimal dalam optimasi pemerataan beban *traffic.*	*Least connection* dan koloni semut	*Throughput, delay* dan *jitter.*
Febrianti (2022)	Mengetahui performansi *load balancing nginx* menggunakan algoritma *least connection* pada arsitektur *software defined network* berdasarkan parameter yang digunakan.	*Least connection*	*Throughput, delay, jitter, packet loss* dan *response time.*

Berdasarkan tabel diatas terkait dengan referensi jurnal penelitian terdahulu terdapat beberapa persamaan yaitu menggunakan algoritma *least connection, web server nginx* dan diterapkan pada arsitektur SDN. Namun pada penelitian ini memiliki perbedaan dimana skenario pengujian yang dilakukan menggunakan jumlah koneksi yang meningkat 2 kali dimulai dari 500 hingga 8000 koneksi dengan *rate* 20. Penelitian ini juga menggunakan parameter QoS seperti *throughput, delay, jitter* dan *packet loss* serta *response time*.

2.2 DASAR TEORI

2.2.1 Perkembangan Jaringan Komputer

A. Pengertian Jaringan Komputer

Jaringan komputer merupakan dua perangkat komputer atau lebih yang saling terhubung sehingga dapat melakukan komunikasi dan berbagi sumber daya dengan menggunakan media penghubung dengan jenis kabel atau nirkabel.

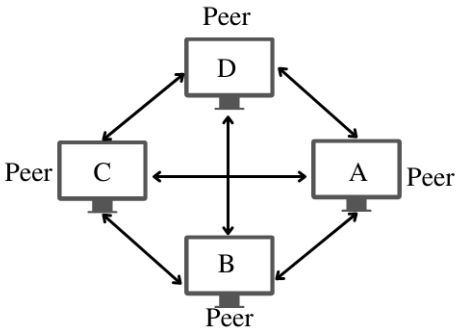
1. Konfigurasi Dasar Jaringan Komputer

Jaringan komputer memiliki beberapa konfigurasi yang dibedakan menjadi beberapa jenis yaitu berdasarkan pola pengoperasian, berdasarkan jangkauan dan berdasarkan media transmisi.

a) Berdasarkan Pola Pengoperasian

1) *Peer to Peer*

Peer to peer merupakan jaringan komputer yang terdiri dari beberapa komputer yang biasanya digunakan untuk laboratorium komputer, riset dan beberapa hal. *Peer to peer* merupakan suatu model dimana pada tiap pc dapat menggunakan *resource* pada pc yang lain atau juga memberikan *resource* nya untuk digunakan oleh pc lain atau dapat dikatakan berfungsi seperti *client* maupun *server* pada periode yang sama [6].

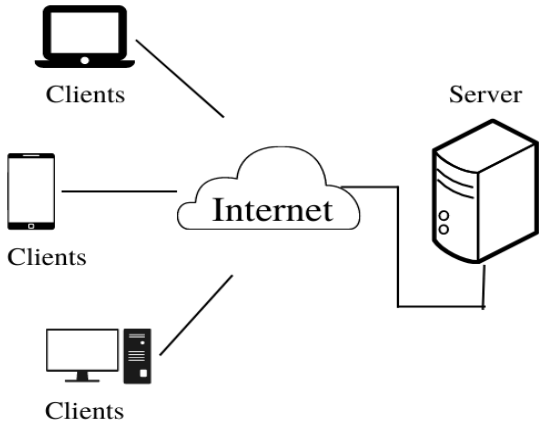


Gambar 2.1 Konfigurasi *Peer to peer*

Berdasarkan gambar 2.1, dapat dianalogikan apabila pc A dalam jaringan *peer to peer* mengambil data dari pc B, maka pc A akan bertindak menjadi *server* sehingga dapat mengakses *file* yang diinginkan dari pc B, sedangkan pc B bertindak sebagai *client*. Sehingga komputer dapat bertindak menjadi *server* dan *client* dalam jaringan *peer to peer* secara bersamaan [6].

2) *Client to Server*

Client to server merupakan jaringan komputer dimana semua komputer akan terhubung ke satu komputer yang bertugas sebagai *server*, sedangkan komputer yang dilayani disebut dengan *client* atau pengguna seperti yang ditunjukan pada gambar 2.2[6].

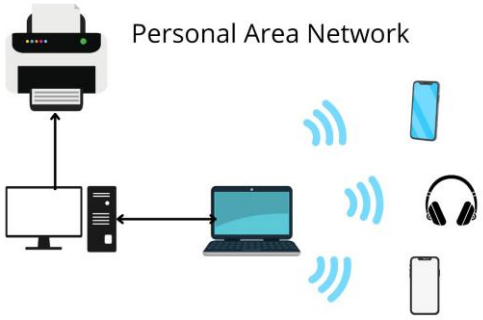


Gambar 2.2 Konfigurasi *Client to server*

b) Berdasarkan Jangkauan

1) PAN

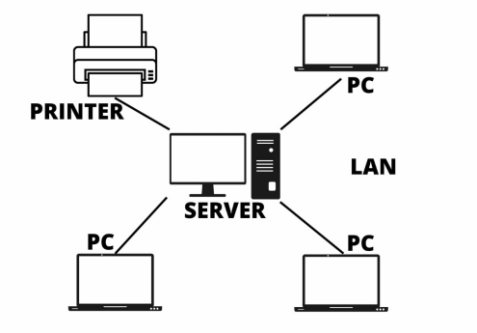
Personal Area Network merupakan suatu jaringan komputer yang bertujuan melakukan komunikasi antara komputer dengan sebuah perangkat seperti laptop, *handphone* dengan jangkauan yang sangat dekat karena tidak dapat digunakan untuk melakukan komunikasi jarak jauh. Fungsi dari jaringan PAN ini dapat juga disebut sebagai titik akses terhadap berbagai macam perangkat seperti *handphone*, komputer dan sebagainya yang terhubung menggunakan media seperti *Bluetooth*, *wifi* dan lain sebagainya seperti yang ditunjukan pada gambar 2.3[6].



Gambar 2.3 Topologi jaringan PAN

2) LAN

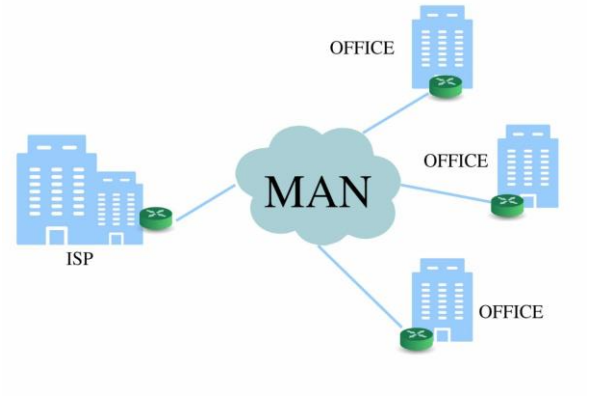
Local Area Network merupakan jaringan komputer berdasarkan jangkauannya, memiliki jaringan telekomunikasi antar perangkat komputer dan perangkat lainnya namun dalam area cakupan kecil. Biasanya cakupan area LAN ini meliputi satu gedung atau satu ruangan. Jaringan LAN ini biasanya digunakan untuk kebutuhan jaringan kecil menggunakan *resource* bersama, contohnya penggunaan printer bersama dan sebagainya. Contoh topologi jaringan LAN ditunjukan pada gambar 2.4[6]].



Gambar 2.4 Topologi Jaringan LAN

3) MAN

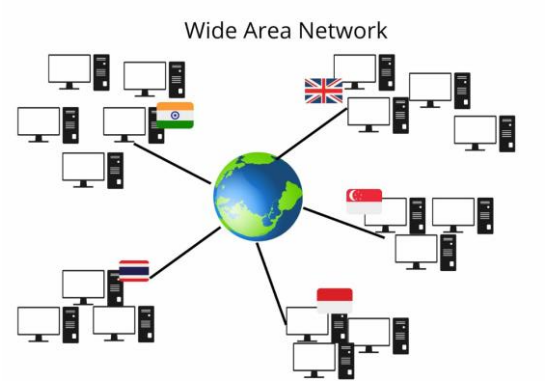
Metropolitan Area Network merupakan jenis jaringan komputer berdasarkan jangkauannya menggunakan metode sama seperti LAN hanya saja cakupan area lebih luas jika dibandingkan dengan LAN. Cakupan area MAN ini seperti besar dalam satu kota atau bahkan lebih besar seperti yang ditunjukan pada gambar 2.5[6].



Gambar 2.5 Topologi jaringan MAN

4) WAN

Wide Area Network merupakan jaringan komputer berdasarkan jangkauannya yang memiliki jaringan komunikasi dengan jangkauan cakupan area nya sangat luas bahkan dapat melewati batas provinsi, satu negara bahkan satu benua. Jaringan WAN ini pada umumnya menggunakan gabungan komunikasi seperti *fiber optic*, satelit maupun *microwave* seperti yang ditunjukan pada gambar 2.6[6].

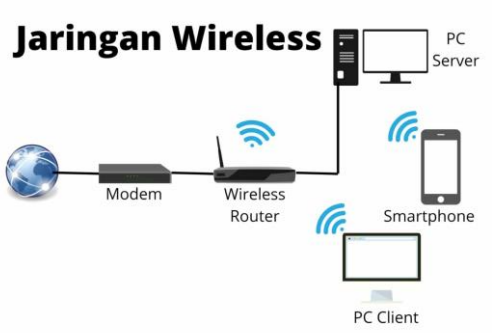


Gambar 2.6 Topologi Jaringan WAN

c) Berdasarkan Media Transmisi

1. *Wireless Network*

Jaringan komputer tanpa kabel adalah jenis jaringan komputer yang tidak menggunakan media kabel. Saat ini penggunaan jaringan *wireless* banyak digunakan karena *user* hanya cukup mengaktifkan layanan *wireless* dari perangkat baik komputer, laptop maupun *handphone* kemudian menghubungkannya ke koneksi *wireless* yang ada.



Gambar 2.7 Konfigurasi Jaringan *Wireless*

Contoh layanan jaringan *wireless* adalah *hotspot* area, fitur *thetering* yang ada pada *smartphone* dan sebagainya seperti yang ada pada gambar 2.7[6].

a. *Wifi*

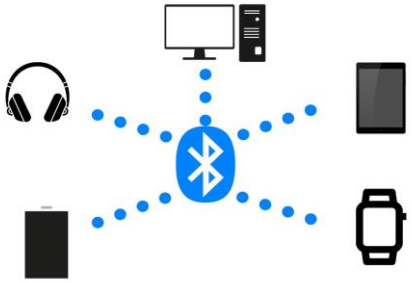
Wireless Fidelity merupakan tempat menyediakan koneksi LAN menggunakan teknologi yang biasa disebut dengan *hotspot. Wifi berada pada frekuensi 2.4 GHz menggunakan Direct Sequence Spread Spectrum* atau DSSS sehingga lebih cepat dan stabil

dibandingkan dengan *bluetooth*. Konfigurasi wifi ditunjukkan pada gambar 2.8[6].

Gambar 2.8 Konfigurasi *wifi*

b. *Bluetooth*

Bluetooth berada pada frekuensi 2.4 GHz menggunakan sinyal radio. Menggunakan *Bluetooth* dapat menciptakan koneksi antar perangkat elektronik apa saja tidak hanya komputer, dengan cakupan *bluetooth* hingga 10 m dan tidak terhalang oleh fleksibilitas media. Contoh dari penggunaan *bluetooth* seperti penggunaan piranti *wireless headset*, pc, *file exchange* dan sebagainya. Namun *bluetooth* memiliki kelemahan dimana manajemen data hanya memungkinkan 2 perangkat saja sementara perangkat yang lain menunggu. Konfigurasi penggunaan *Bluetooth* ditunjukkan pada gambar 2.9[6].

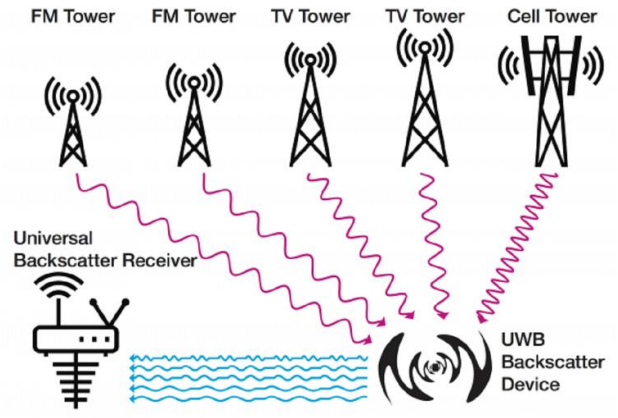


Gambar 2.9 Konfigurasi *Bluetooth*

c. Gelombang Radio

Gelombang radio menjangkau jarak yang jauh secara *omnidirectional* yang dapat digunakan untuk komunikasi dalam dan luar yang memiliki sifat yaitu tergantung pada frekuensi nya. Gelombang radio dapat menangkap sinyal dengan frekuensi yang

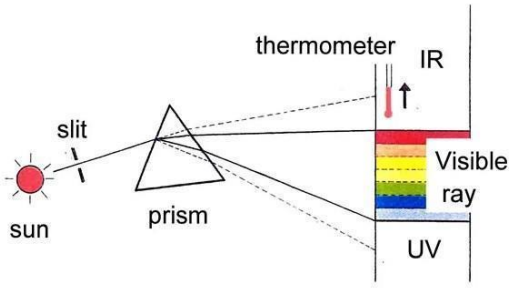
sama dan dapat menggerakan informasi menuju ke tujuannya dengan amplitudo seperti yang ditunjukkan pada gambar 2.10[6].



Gambar 2.10 Gelombang radio

d. Gelombang Inframerah

Gelombang inframerah dapat menyebar secara garis lurus dan memiliki sifat directional dengan pemancar cahaya, diterima oleh *receiver* dengan menyamakan frekuensi. Gelombang ini terbilang murah dan mudah dalam pembuatannya, tidak mengganggu sistem infrared yang lainnya. Namun disisi lain gelombang *infrared* ini memiliki kekurangan dimana gelombang ini tidak dapat tembus pada benda padat. Cara kerja gelombang radio ditunjukkan pada gambar 2.11 [6].

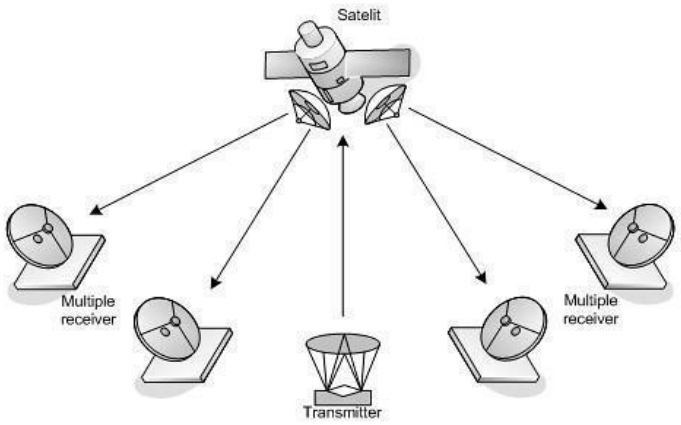


Gambar 2.11 Cara kerja *infrared*

e. Gelombang Mikro

Gelombang mikro merupakan sebuah gelombang radio berada pada frekuensi tinggi. Gelombang mikro menyebar secara garis lurus dan memiliki sifat *directional* sehingga dapat difokuskan. Gelombang radio menggunakan satu pemancar utama sehingga dapat mengalokasikan

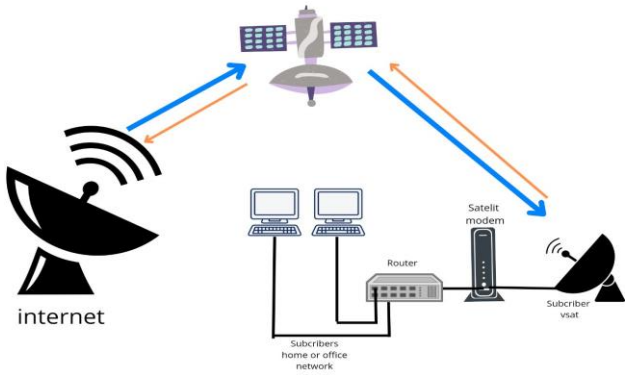
frekuensi 2.400-2484 GHz. Salah satu contoh dari penggunaan gelombang mikro ditunjukkan pada gambar 2.12[6].



Gambar 2.12 Penerapan gelombang mikro

f. Satelit

Satelit merupakan media transmisi untuk menerima sinyal yang dikirm dari stasiun bumi dan kemudian meneruskan pada stasiun belahan bumi lain, dapat berfungsi sebagai *repeater* dan membagi jalur komunikasi agar satelit dapat digunakan bersama-sama, namun data atau informasi tidak menjadi satu. Komunikasi satelit sering kali dimanfaatkan untuk menjangkau daerah yang jauh dan terpencil. Konfigurasi satelit ditunjukkan pada gambar 2.13[6].

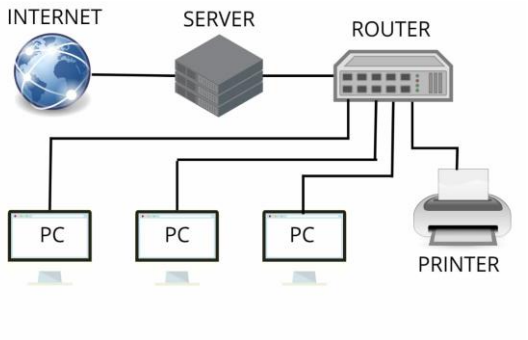


Gambar 2.13 Konfigurasi satelit

2. *Wired Network*

Wired network merupakan jaringan komputer menggunakan kabel untuk media transmisinya. Jaringan berkabel ini memiliki tingkat kestabilan bahkan kecepatan yang relatif tinggi dibandingkan dengan *wireless*. *Wired network* ini menggunakan media kabel berupa kabel

UTP, kabel *Coaxial*, dan *Fiber Optic*. Konfigurasi *wired network* ditunjukkan pada gambar 2.14 [6].



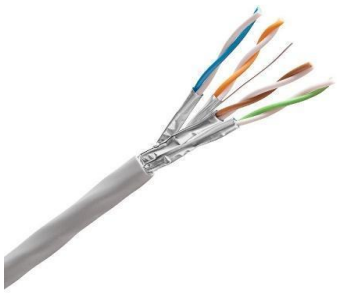
Gambar 2.14 Konfigurasi Jaringan *Wired*

a. *Twisted Pair Cable*

Twisted pair cable merupakan kabel dengan dua pasang kawat yang terpilin untuk mengurangi interferensi elektromagnetik yang dihasilkan seperti radiasi elektromagnetik, dan *crosstalk* antar kabel berdekatan. 2 jenis *twisted pair cable* diantaranya:

1) *Shielded Twisted Pair*

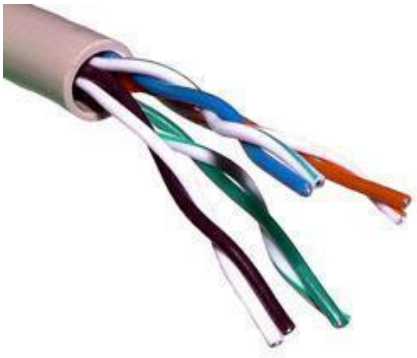
S*hielded twisted pair* merupakan kabel yang biasa untuk instalasi jaringan *Ethernet* yang memiliki resistansi interferensi elektromagnetik dan frekuensi radio tanpa meningkatkan ukuran dari kabel fisiknya. Kabel *shielded twisted pair* terdiri dari dua pasang kabel yang terpilin menjadi satu. Kabel tipe ini biasanya digunakan untuk beberapa instalasi telepon, tv dan radio. Kabel stp ini memiliki struktur yang ditunjukkan pada gambar2.15[6].



Gambar 2.15 Struktur kabel STP

2) *Unshielded Twisted Pair*

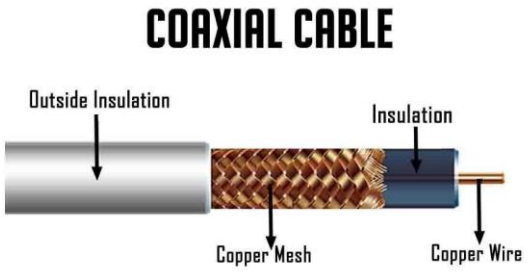
Unshielded twisted pair merupakan kabel yang digunakan untuk pemasangan kabel jaringan komputer. *Unshielded twisted pair* terdiri dari 4 pasang kawat medium yang memiliki lapisan pelindung pada setiap pasangannya. Kabel *unshielded twisted pair* memiliki ukuran yang kecil namun mudah terkena interferensi yang disebabkan dari media atau perangkat yang ada disekitarnya. Kabel utp ini memiliki struktur yang ditunjukkan pada gambar 2.16[6].



Gambar 2.16 Stuktur kabel UTP

b. *Coaxial*

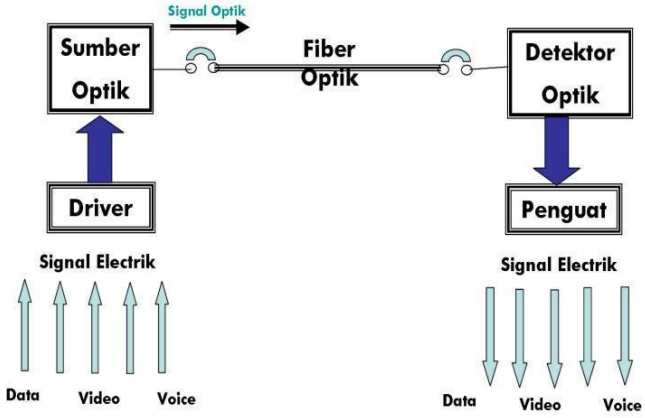
Coaxial merupakan kabel yang terdiri dari konduktor silindris yang mengelilingi kabel tembaga, biasa digunakan untuk mengirim sinyal frekuensi tinggi dari 300 kHz keatas. Dalam jaringan LAN, kabel *coaxial* dijalankan tanpa membutuhkan banyak *repeater* untuk komunikasi jarak jauh. Kabel *coaxial* ini memiliki struktur yang ditunjukkan pada gambar 2.17[6].



Gambar 2.17 Struktur kabel *coaxial*

c. Fiber Optik

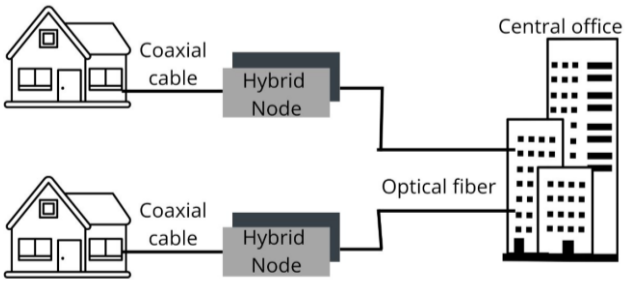
Fiber optik merupakan kabel yang berfungsi sebagai media *networking*, dapat digunakan untuk mentransmisikan modulasi. Fiber optik dapat melakukan transmisi hingga puluhan juta bit digital per detik sebuah koneksi kabel optik yang berjalan pada jaringan komersial. Penggunaan fiber optik memiliki keuntungan seperti kecepatan yang tinggi hingga *gigabits per sec* dengan panjang gelombang fiber optik yang membawa paket yang memiliki kapasitas besar, tahan interferensi elektromagnetik walaupun fiber optik memiliki harga mahal. Cara kerja dari fiber optik ditunjukkan pada gambar 2.18[6].



Gambar 2.18 Cara kerja fiber optic

d. *Hybrid Fiber Coax*

Hybrid fiber coax merupakan teknik penggabungan dari dua transmisi yaitu *fiber optic* dan *coaxial*. HFC merupakan sebuah perkembangan dari teknologi jaringan *cable* TV dengan menggunakan kabel *coaxial* sebagai media transmisi nya untuk aplikasi layanan Tv *broadcast*. HFC memiliki jenis-jenis layanan aplikasi seperti layanan video dengan fasilitas *pay per view*, *video on demand*, *game on demand* dan sebagainya, selain itu terdapat layanan data seperti *e-commerce*, *video conference* dan sebagainya. Contoh konfigurasi *hybrid fiber coaxial* ditunjukkan pada gambar 2.19[6].



Gambar 2.19 Konfigurasi *hybrid fiber coaxial*

3. Protokol Jaringan Komputer

a) TCP

Transmission Control Protocol merupakan protokol penting pada *layer transport* yang bersifat *Connection Oriented*. *Connection oriented* artinya adalah dua aplikasi pengguna TCP harus melakukan suatu pembentukan hubungan yang dilakukan dengan pertukaran *control* informasi sebelum terjadinya transmisi data. TCP memiliki layanan *reliable* yang berarti melakukan deteksi kesalahan pada paket retransmisi. Selain itu, TCP memiliki layanan *byte stream service* yang berarti paket akan dikirimkan dan sampai pada tujuan dilakukan secara berurutan. Protokol TCP dapat digunakan koneksi yang membutuhkan keandalan seperti FTP, HTTP, Telnet, SSH dan lain nya [7].

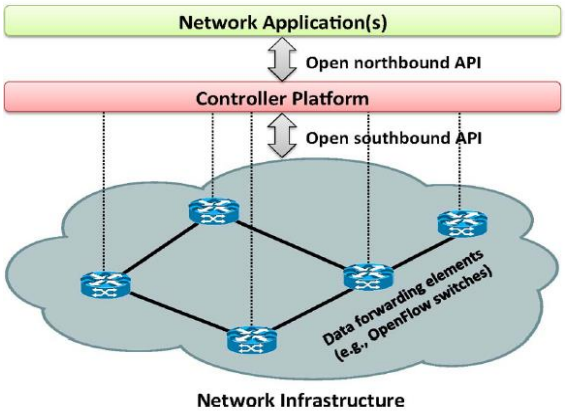
b) UDP

User Datagram Protocol merupakan protokol *layer transport* berorientasi pada *connection oriented* dan dapat diandalkan. UDP menyediakan layanan pengiriman datagram bersifat *connectionless oriented* yang artinya tidak dilengkapi dengan sistem deteksi dan koreksi kesalahan. Protokol UDP dapat digunakan pada layanan transmisi *audio* atau *video* seperti VoIP, audio atau *video streaming*. Karena UDP kurang baik apabila untuk pengiriman paket berukuran besar, karena mengakibatkan banyak paket yang *loss* [6].

A. *Software Defined Network*

Software Defined Network adalah suatu pendekatan baru untuk mendesain, mengelola, dan melakukan implementasi jaringan yang

mendukung inovasi jaringan yang semakin kompleks. Arsitektur jaringan SDN ditunjukkan pada gambar 2.20[1].

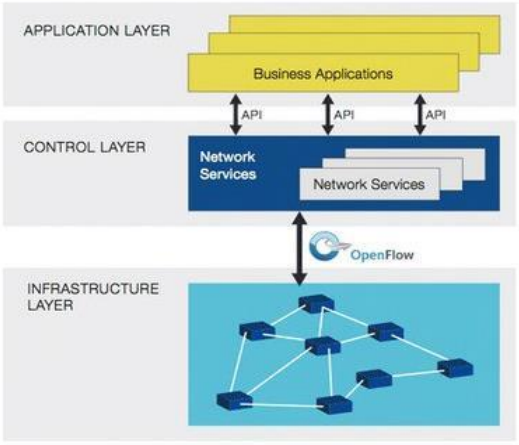


Gambar 2.20 Arsitektur jaringan SDN

SDN memiliki konsep dasar untuk melakukan pemisahan antara *control plane* dengan *forwarding plane. Control plane* merupakan komponen dari jaringan yang berfungsi mengontrol jaringan seperti konfigurasi sistem, melakukan manajemen jaringan, menentukan *forwarding table* dan *routing table*. Sedangkan untuk *data plane* merupakan komponen dari jaringan berfungsi untuk meneruskan paket, mengatur *Quality of Service*, melakukan enkapsulasi paket serta menguraikan *header* paket.

1. Struktur *Layer* pada SDN

Arsitektur jaringan *Software Defined Network* terpisah menjadi 3 *layer*, yaitu *application layer*, *control layer* dan *infrastructure layer*. *Layer* pada SDN ini memiliki tugas dan fungsi nya masing dan tersusun mulai dari *infrastructure layer*, *control layer* dan *application layer* yang ditunjukan seperti pada gambar 2.21[1].



Gambar 2.21 *Layer* pada arsitektur SDN

Penjelasan mengenai fungsi dari tiap *layer* sebagai berikut:

a. *Application Layer*

Pada *layer* ini berfungsi dalam menyediakan layanan *interface* dalam pembuatan program aplikasi. *Layer* ini juga dapat menetapkan aturan dan menawarkan layanan seperti *load balancing*, IDS, *firewall* dsb.

b. *Control Layer*

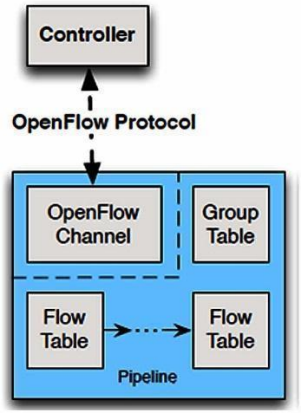
Pada *layer* ini menyediakan kebutuhan antara infrastruktur dan aplikasi dengan memberikan perintah yang sesuai. Jenis *controller* yang digunakan seperti pox, ryu, *opendaylight*, onos dan *floodlight*.

c. *Infrastructure Layer*

Pada *layer* ini berisi elemen-elemen jaringan dan perangkat keras berfungsi untuk sebagai *forwarding plane* [8].

2. Protokol Komunikasi SDN - *Openflow*

Openflow merupakan sebuah protokol komunikasi pemisahan antara *control plane* dan *forwarding plane* dari perangkat jaringan, dan menciptakan komunikasi pada SDN. *Openflow* mengizinkan dalam akses langsung menuju *data plane* dari suatu perangkat jaringan *switch* dan *router* baik secara fisik atau virtual. *Openflow* merupakan antarmuka SDN *controller* dengan *switch*, memiliki komponen seperti *flow table*, *controller channel*, dan *openflow protocol* seperti yang ditunjukan pada gambar 2.22[9].



Gambar 2.22 *Openflow Switch*

3. *Controller*

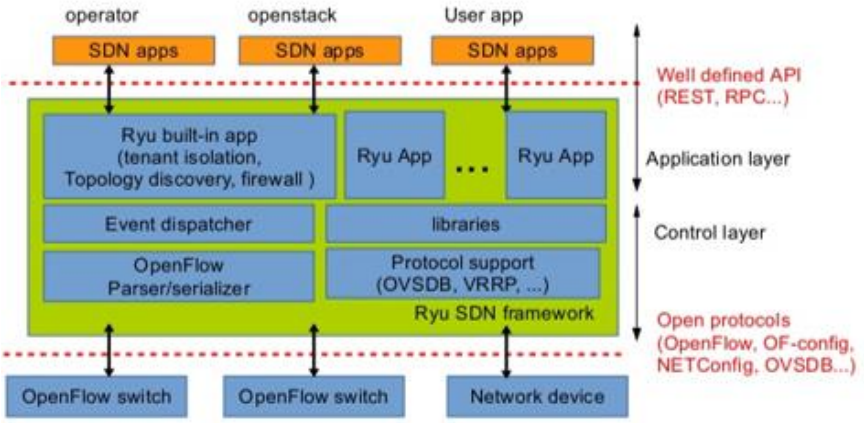
Controller merupakan otak dalam jaringan SDN yang didalamnya terdapat pengendali strategis yaitu mampu mengontrol aliran dari *switch* dan memiliki aturan agar dapat menentukan jalur terbaik untuk dipilih. *Controller* melakukan fungsi lainnya seperti *routing* dan *load balancing* [10].

a) POX

POX suatu *platform* pengembangan *open source* pada SDN menggunakan bahasa pemrograman *Phyton* dan salah satu jenis *controller Openflow*. *Controller pox* memungkinkan proses dari perancangan pembangunan jaringan lebih cepat, dan umumnya sering digunakan daripada *controller* pendahulunya yaitu NOX. POX menyediakan penggunaan dengan cara yang efisien dalam mengimplementasikan protokol komunikasi antara *control plane* dan *forward plane* [10].

b) RYU

Ryu merupakan *controller* yang disebut dengan komponen dasar, sebuah perangkat lunak yang bersifat *open source* pada *networking framework*. Implementasi dari *controller ryu* ini menggunakan bahasa *phyton*. *Controller ryu* menyediakan komponen perangkat lunak dengan API sehingga memudahkan membuat manajemen jaringan dan control aplikasi baru. *Ryu* mendukung protokol dalam perangkat jaringan seperti *openflow*, *netconf* dll. *Framework ryu* berada pada arsitektur SDN berada pada *control layer* sesuai dengan gambar 2.23[10].



Gambar 2.23 Struktur *Ryu*

c) *Floodlight*

Floodlight merupakan salah satu jenis *controller* pada kelas *enterprise* dengan lisensi *Apache* yang mendukung protokol *openflow* menggunakan bahasa pemrograman java sehingga dapat mengatur aliran data pada arsitektur SDN. *Floodlight* dibuat dengan tujuan untuk bekerja dalam meningkatkan jumlah *router, switch, switch virtual* dan jalur akses yang mendukung protokol *openflow*. *Floodlight* memiliki beberapa versi *openflow* 1.0, 1.2, dan juga 1.3[10].

d) ONOS

ONOS merupakan sistem operasi yang dirancangan dengan bertujuan membantu *service provider* jaringan dalam membangun jaringan berbasis *carrier-grade*. Selain itu, dapat berfungsi sebagai pesawat *control* SDN jaringan LAN dan pusat data [10].

4. *Software* Mininet sebagai Emulator SDN

Mininet merupakan emulator yang dapat digunakan dalam pembuatan *prototype* jaringan skala besar secara cepat pada *resource* terbatas. *Software* mininet mendukung riset pada bidang SDN [11]. Mininet dalam penggunaannya menciptakan jaringan virtual realistis, *switch* dan kode aplikasi, menjalankan *real kernel*, pada *machine* tunggal seperti *virtual machine*, *physical machine*. Mininet merupakan salah satu solusi yang unggul karena memudahkan dalam penggunaan, performansi, akurasi, skalabilitas, dan memiliki sertifikasi emulator SDN dan *Openflow* [5].

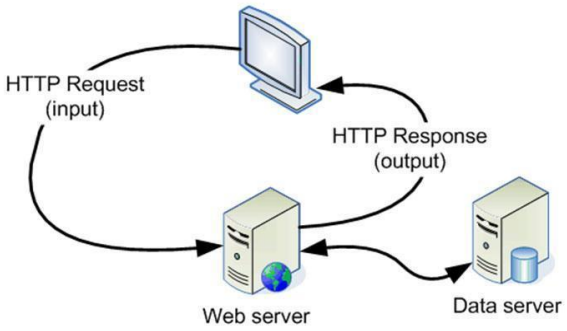
2.2.2 Web Server

A. Pengertian *Web Server*

Web server merupakan sebuah media untuk memuat informasi yang terletak pada halaman *web*. *Web server* berfungsi untuk mentransferkan berkas *request client* melalui protokol komunikasi HTTPS/HTTP atau biasa disebut dengan *browser* dan men*transfer* dalam sebuah halaman *web*.

B. Cara Kerja *Web Server*

Cara kerja *web server* sederhana yaitu menerima permintaan *client* dan kemudian mengirimkan kembali bentuk berkas kepada *client*. Cara kerja dari *web server* ditunjukkan pada gambar 2.24.



Gambar 2.24 Cara kerja *web server*

Pada saat *client* melakukan *request* kepada *web server*, *request* data tersebut akan dikemas pada TCP yang kemudian akan dikirim menuju alamat yang dibutuhkan yaitu HTTPS atau HTTP untuk dapat ditampilkan pada *browser*. Namun apabila *request* data tidak dapat ditemukan pada *web server* secara otomatis *web server* akan menolak adanya *request* tersebut dengan menampilkan *Page Not Found* atau *Error* 404 [12].

C. Aplikasi *Web Server*

Web server memiliki beberapa jenis yang paling umum digunakan seperti *Apache*, IIS, *Litespeed*, dan *Nginx*. Berikut penjelasan terkait dengan jenis *web server*.

1. *Apache*

Apache merupakan *web server* yang paling banyak digunakan. Penggunaan *web server apache* sangat aman karena proses instalasi yang mudah dalam *freeware*, konfigurasi dan cara pengaturan yang mudah, *open source* dan tidak terbatas dalam komunikasi. *Web server apache* berfungsi

untuk membuat suatu hubungan antara *server* dengan *browser* seperti opera, *chrome* dan safari milik *client* dan melakukan pengiriman *file* antar *user* dan *server* [13].

2. IIS

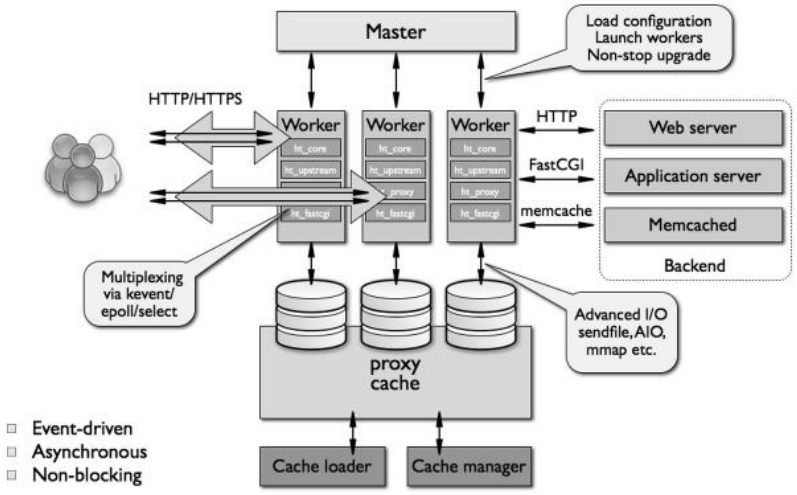
Web server IIS atau *Internet Information Service* merupakan jenis *web server*, bekerja menggunakan beberapa protokol jaringan seperti DNS, IP/TCP yang berfungsi untuk merangkai suatu situs *web*. *Web server* jenis ini cocok untuk protokol seperti HTTP, SSL dan FTP. Kelebihan *web server* IIS ini dapat diakses penuh pada sistem operasi *Windows* dan *platform* .NET [12].

3. *Litespeed*

Web server LiteSpeed merupakan sistem *web* yang bersifat *open source* mendukung sistem *Linux* dan *Unix* yang diciptakan oleh *programmer* dari jerman. *Web server* ini memiliki keunggulan dengan fitur tambahan seperti *Output-Compression, FastCGi* dan URL *Writing* [12].

4. *Nginx*

Nginx merupakan *web server open source* kinerja tinggi sebagai *server proxy* dan *http*. *Nginx* merupakan *web server* yang ringan dengan memiliki performa cepat, mampu memproses *request* dengan baik. *Nginx* dapat menerima banyak *traffic* dengan menggunakan *load balancing*, karena *nginx* dapat diakses banyak klien dalam waktu yang bersamaan [3].



Gambar 2.25 Arsitektur *web server nginx*

Pada gambar 2.26 merupakan arsitektur *nginx* yang menggunakan sistem asinkron dan *event driven*. Dimana aplikasi ini akan menangani *request* hanya dalam satu *thread* saja, tidak menjadikan *request* tersebut sebagai satu proses yang baru. *Nginx* memiliki sistem *worker process* dalam menangani sekumpulan *thread*, didalam *worker process* terdapat *worker connection* dengan unit lebih kecil yang bertugas dalam menangani *request thread* di dalam *worker process*. Satu *worker connection* dapat melayani hingga 1.024 *request* [14].

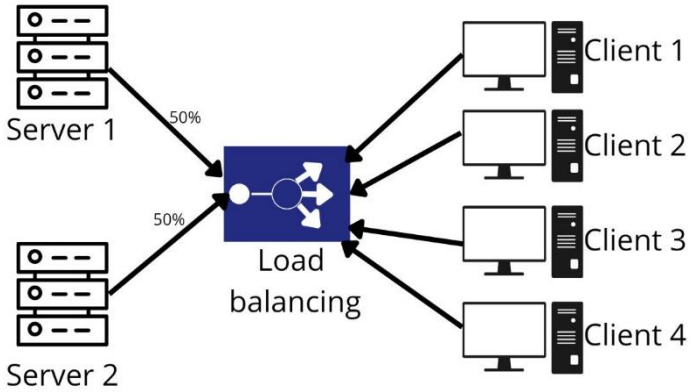
2.2.3 *Load Balancing*

A. Pengertian *Load Balancing*

Load balancing merupakan mekanisme jaringan dalam membagi beban *server* mempertimbangkan kapasitas yang dimiliki dari *server* bertujuan untuk mengoptimalkan sumber daya, memaksimalkan nilai *throughput*, meminimalkan nilai *response time* dan menghindari *server overload*. Metode ini digunakan *server* yang memiliki jumlah *client* melebihi kapasitas maksimum dari jumlah *request* yang mampu ditangani *server*. Sehingga suatu *server* membutuhkan metode *load balancing* dalam optimasi agar tidak mengalami *overload*.

B. Cara Kerja *Load Balancing*

Berdasarkan pengertian *load balancing* yaitu membagi beban pada *server* berdasarkan kapasitas yang dimiliki oleh server. Berikut merupakan salah satu contoh kasus mekanisme kerja *load balancing* apabila terdapat 2 server seperti akan ditunjukan pada gambar 2.30.



Gambar 2.27 Cara kerja *load balancing*

Berdasarkan gambar 2.27 ketika 4 *client* mengirimkan *request* ke *server*, *load balancer* akan membagi beban sebagian dengan ketentuan *client* 1 dan 2 dilayani *server* 1 sedangkan untuk *client* 3 dan 4 akan dilayani oleh *server* 2[4]. Server yang akan digunakan dapat disesuaikan dengan kebutuhan tidak hanya menggunakan 2 server saja. Pada implementasinya, *load balancing* memiliki beberapa algoritma dalam mengelola dan mendistribusikan beban jaringan diantaranya *Least Connection, Round Robin, Weighted Round Robin, Weighted Least Connection,* dan koloni semut [9].

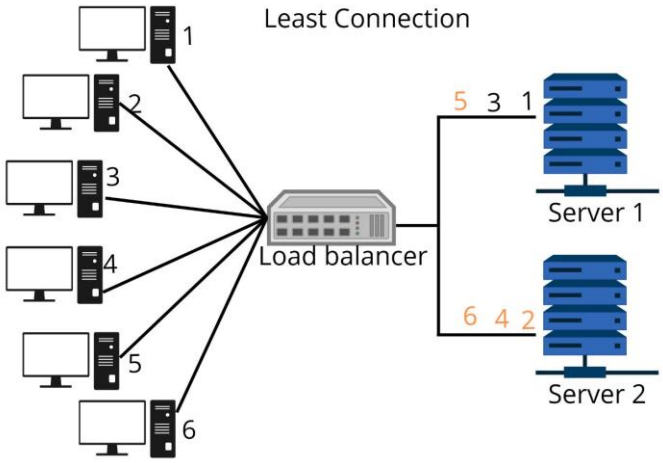
C. Algoritma *Least Connection*

1. Pengertian *Least Connection*

Algoritma *least connection* merupakan salah satu jenis algoritma yang dimiliki oleh *load balancing* dalam pembagian beban kerja *server* dengan mendistribusikan beban kerja berdasarkan banyaknya koneksi yang terhubung dan dilayani oleh *server*.

2. Cara Kerja

Pada contoh pembagian beban kerja berdasarkan algoritma *round robin* berikut menggunakan 2 buah *server* dengan 6 *client*. Penggunaan jumlah *server* dan *client* dapat disesuaikan dengan kebutuhan seperti ditunjukkan pada gambar Pembagian beban kerja berdasarkan algoritma *least connection* ditunjukkan pada gambar 2.28 [15].

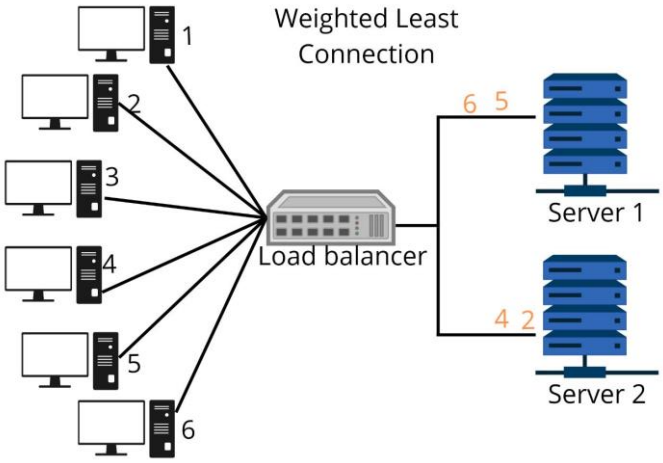


Gambar 2.28 Algoritma *least connection*

Berdasarkan gambar diatas, dalam hal ini akan menggunakan 2 *server* dan 6 buah *client* dengan kondisi *server* 1 telah menyelesaikan *request* dari *client* 1 dan 3 sedangkan pada *server* 2 sedang melayani *request* dari *client* 2,4 dan 6. Apabila *client* 5 melakukan *request*, maka *request* tersebut akan dialihkan menuju *server* 1 karena *server* tersebut memiliki koneksi aktif paling sedikit. Sesuai dengan cara kerja algoritma *least connection server* dengan koneksi paling banyak dialihkan beban kerja nya ke *server* lain dengan beban lebih rendah, begitu juga dengan *server* yang memiliki koneksi paling sedikit maka akan diberikan beban terlebih dahulu. Algoritma ini baik digunakan untuk menyeimbangkan beban server[16].

D. Algoritma *Weighted Least Connection*

Algoritma *weighted least connection* pengembangan dari algoritma *least connection* dengan membagi beban kerja berdasarkan bobot dari *server*. Bobot diberikan pada *server*, dan untuk koneksi baru dipilih *server* dengan kapasitas yang paling sedikit. Penggunaan jumlah *client* dan *server* dapat disesuaikan dengan kebutuhan. Salah satu contoh penerapan dari algoritma ini menggunakan 2 *server* dengan 6 *client* seperti yang ditunjukan pada gambar 2.29[17].

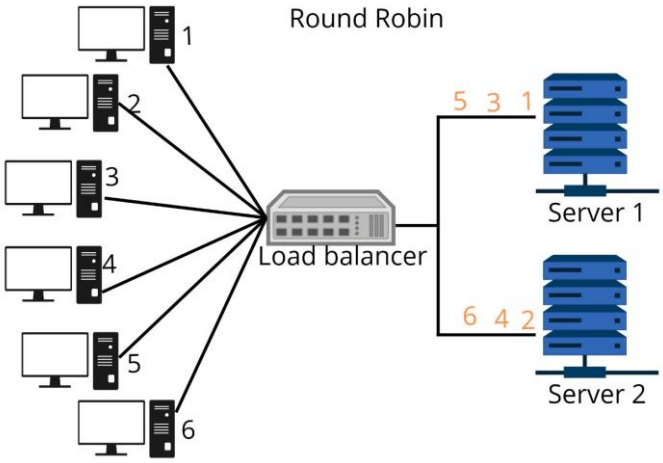


Gambar 2.29 Algoritma *weighted least connection*

E. Algoritma *Round Robin*

Algoritma *round robin* merupakan salah satu jenis algoritma yang dimiliki oleh *load balancing* dalam pembagian beban kerja *server* dengan mendistribusikan beban kerja secara rata pada seluruh *server* tanpa

melihat *capacity* atau beban permintaan yang dikirim. Algoritma ini merupakan algoritma yang paling sederhana dan banyak digunakan. Penggunaan jumlah *server* dan *client* dapat disesuaikan dengan kebutuhan tidak hanya dengan 2 *server* saja, pada contoh pembagian beban kerja berdasarkan algoritma *round robin* berikut menggunakan 2 buah *server* dengan 6 *client* ditunjukkan pada gambar 2.30[4].



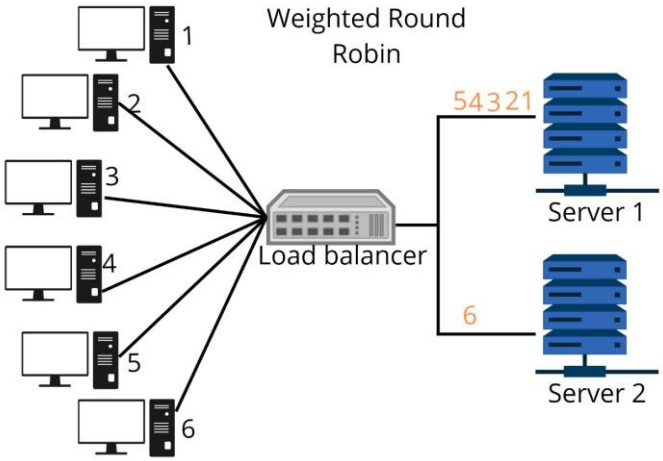
Gambar 2.30 Algoritma *round robin*

Berdasarkan gambar 2.30 Ketika *client* 1 melakukan *request*, *load balancing* meneruskan *request* menuju ke *server* 1. Kemudian ketika *request* dari *client* 2 datang, maka *load balancing* meneruskan *request* ke *server* 2. Apabila terdapat *request* selanjutnya, maka *request* diteruskan menuju *server* 1, dan seterusnya [4].

F. Algoritma *Weighted Round Robin*

Algoritma *weighted round robin* merupakan pengembangan dari algoritma *round robin*, sehingga algoritma ini memiliki cara kerja sama dengan algoritma *round robin* namun suatu kondisi baru akan diberikan pada *server* yang akan dikirimkan *request* dari *load balancer* mempertimbangkan kemampuan sumber daya *server* yang digunakan dan memberi jumlah beban lebih besar pada *server* dengan kapasitas lebih tinggi. Penggunaan jumlah server dan client dapat disesuaikan dengan kebutuhan tidak hanya dengan 2 *server* saja, pada contoh pembagian beban kerja berdasarkan algoritma *weighted round robin* berikut

menggunakan 2 buah *server* dengan 6 *client* ditunjukkan pada gambar 2.31[4].



Gambar 2.31 Algoritma *weighted round robin*

G. Algoritma Koloni Semut

Algoritma koloni semut merupakan algoritma yang terinspirasi dari perilaku koloni semut ketika mencari makanan. Dalam implementasinya, algoritma ini memiliki dua tahapan. Pada tahap pertama ketika semut maju maka proses ini bertujuan untuk mencari *rute* baru dan menemukan informasi mengenai kondisi *rute* yang akan dilewati, apabila sudah menemukan tempat tujuan, di tahap selanjutnya semut akan mundur melalui jalur yang sama [5].

2.2.4 Sistem Operasi

A. Pengertian Sistem Operasi

Sistem operasi merupakan bagian penting dalam sistem komputer, atau dapat disebut dengan jantung komputer. Sistem operasi pada komputer biasanya untuk mengendalikan sistem *input output*, membaca dan menjalankan program. Sistem operasi berisikan sekumpulan kebijakan dan mekanisme membantu mendefinisikan pengendalian *resource* yang digunakan bersama. Sistem operasi memiliki berbagai macam jenis yang berguna untuk memaksimalkan kerja komputer seperti *Unix, Windows, Linux, MAC* OS [18].

B. Jenis-Jenis Sistem Operasi

1. *Unix*

Unix merupakan sistem operasi yang didesain sebagai sistem operasi *portable*, *multitasking* dan juga *multiuser*. Sistem operasi ini menekankan pada *workstation* dan *server*. Sistem operasi *unix* ini memiliki kelebihan seperti aman dari virus dan bersifat *open source* [18].

2. *Windows*

Windows merupakan sistem operasi pengembangan dan perbaikan dari Ms.Dos, perbaikan yang cukup jelas terlihat yaitu dari sisi kemudahan dalam penggunaan, karena sistem operasi *windows* ini menggunakan *interface* berbasis GUI (*Graphical User Interface*). Namun sistem operasi ini memiliki kekurangan seperti mudah diretas karena sistem keamanan yang masih lemah, rentan terhadap virus, bersifat *close source* dan berbayar [18].

3. *MAC OS*

MAC OS atau *Macintosh Operating System* merupakan sistem operasi diciptakan oleh *Apple Computer* sehingga penggunaannya khusus untuk komputer *macintosh*, tidak kompatibel pada komputer lain. Sistem operasi ini memiliki kelebihan seperti lebih stabil, aman dari virus, memiliki keamanan yang tinggi sehingga tidak mudah untuk diretas. Namun sistem operasi ini memiliki kekurangan dimana harga yang relatif mahal dan tidak bersifat *open source* [18].

4. *Linux*

Linux merupakan sistem operasi yang bersifat *open source*, sehingga pengguna dapat melakukan modifikasi dan mendistribusikan kembali secara bebas tanpa perlu adanya lisensi. *Linux* dapat digunakan dalam pengembangan *software* dan juga sebagai *end user platform*. Ada beberapa macam *distro linux* yang digunakan seperti *debian*, ubuntu, *red hat* dll [18].

a) Ubuntu

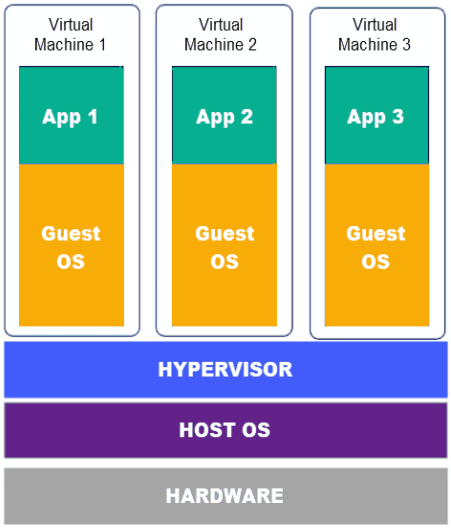
Ubuntu merupakan sistem operasi pengembangan dari program inti *Linux* yang dirilis pada tahun 2004 oleh perusahaan Canonical. Ubuntu merupakan perangkat lunak bebas dengan sumber yang terbuka. Ubuntu

digunakan dalam membangun sistem *server* yang diinstal pada komputer, mesin virtual menggunakan *virtualbox* atau *vps*. Ubuntu ini memiliki beberapa kelebihan seperti mudah digunakan, memiliki keamanan yang kuat, bersifat *open source* dan gratis, minimnya spesifikasi yang dibutuhkan perangkat, mendukung beberapa *software* di dalam *windows* [19].

2.2.5 Virtualisasi

A. *Virtual Machine*

Virtual machine merupakan sebuah program perangkat lunak atau bisa disebut dengan sistem operasi virtual yang biasanya digunakan pada sebuah perangkat keras bersamaan dengan OS asli dari perangkat tersebut. *Virtual machine* berfungsi untuk melakukan tugas yang tidak dapat dilakukan sistem operasi yang asli dari perangkat tersebut.



Gambar 2.32 Struktur *virtual machine*

Berdasarkan gambar 2.32, merupakan struktur dari teknologi *virtual machine* biasanya digunakan pada lingkup *cloud server* yang berfungsi untuk mengakomodasi berbagai tingkat kebutuhan. Sistem operasi utama yang menjalankan *virtualbox* disebut *Host OS*, sedangkan untuk sistem operasi tambahan berjalan secara *virtual* disebut *Guest OS*. Pada mesin *virtual* yang disimulasikan, terlihat bahwa simulasi yang dijalankan terisolasi baik dari mesin *virtual* lain dan dari *host os*. Apabila salah satu mesin *virtual* terkena virus, maka mesin *virtual* yang lain akan aman tidak

terpengaruh dengan adanya mesin *virtual* yang terkena virus. Setiap mesin *virtual* membutuhkan sistem operasi, *binary* dan juga aplikasi secara independen. Maka dari itu konsep dari suatu *virtualisasi* memakan sumber daya besar, karena disimulasikan independen. *Virtual machine* memiliki berbagai *software* seperti *VMWare, Hyper-V, Qemu* dan *Virtualbox* 20].

B. *Software Virtual Machine – Oracle VM Virtualbox*

Oracle VM Virtualbox merupakan salah satu jenis perangkat lunak virtualisasi berfungsi sebagai media dalam menjalankan os tambahan pada os utama [3]. Perangkat ini men*support* banyak os dapat di*instal* dan dijalankan dalam mesin virtual ini seperti *Mac OS X, Windows XP, Linux, Windows 7, OpenSolaris, Solaris, Windows Vista*. Perangkat virtual ini bersifat *open source* sehingga dapat dengan mudah mendapatkan tanpa membeli aplikasi tersebut. *Virtualbox* memiliki fitur yang cukup lengkap, mudah dan stabil dalam penggunaanya dan pengembangan aplikasi ini terhitung cepat. Cara kerja dari *oracle vm virtualbox* ini, sama dengan cara kerja dari mesin *virtual*. Karena *virtualbox* merupakan salah satu aplikasi dalam mengelola mesin *virtual* [3].

2.2.5 Aplikasi Pengujian dan Monitoring Jaringan

A. *HTTPERF*

Httperf merupakan suatu *tool* yang dimanfaatkan sebagai media untuk mengukur performansi terhadap kemampuan seperti standar deviasi, *error rate*, dan *reply rate* dari suatu *web server*. *Httperf* dibuat oleh David Mosberger yang bersifat *open source*. *Tool* ini menawarkan fitur yang fleksibel dalam pembuatan beban kerja berdasarkan parameter yang diberikan. *Httperf* merupakan suatu *tool* untuk sistem operasi turunan UNIX, dapat membangkitkan sejumlah paket *load* dan mendukung HTTP/1.0 dan HTTP/1.1. Pengukuran performansi dari sebuah *web server* dilakukan dengan mengirim permintaan pada *server* dengan *rate* tertentu dan mengukur waktu permintaan [17].

B. *WIRESHARK*

1. Pengertian *Wireshark*

Wireshark merupakan *tool* yang digunakan dalam menganalisis dan mengamati lalu lintas paket data dari suatu jaringan. *Wireshark* dapat juga digunakan dalam menangkap paket data yang melewati jaringan dan menampilkan informasi paket data secara detail, memodifikasi data yang berdasarkan pada *port* yang digunakan. Menggunakan *software* ini juga dapat mengungkap informasi alamat ip dari suatu perangkat, selain itu *wireshark* juga dapat menganalisa jaringan internet, memudahkan dalam melakukan *monitoring* dan menganalisa paket yang dilewatkan pada suatu jaringan .

2. Cara Kerja

Secara umum *wireshark* memiliki dua tahapan cara kerja. Pertama, *wireshark* akan melakukan perekaman pada semua paket yang melewati *interface* yang dipilih melalui *wifi* atau *Ethernet*. Kedua, hasil rekaman pada semua paket yang melewati *interface* dapat dianalisis dengan menggunakan protokol yang diinginkan seperti TCP, UDP atau HTTP .

2.2.6 Parameter Pengujian

A. *Quality of Service*

Quality of Service merupakan kemampuan jaringan dalam menyediakan layanan trafik data yang melewatinya, mengacu pada kemampuan suatu jaringan dalam menyediakan layanan lebih baik pada trafik tertentu seperti aplikasi jaringan, *router* atau *host* bertujuan untuk memberikan pelayanan jaringan yang baik dan terencana untuk memenuhi kebutuhan suatu layanan. Performansi atau kualitas suatu jaringan dapat dilihat melalui parameter-parameter yang ada pada *Quality of Service* seperti *Bandwidth, Jitter, Delay, Throughput, Packet Loss dan Response Time* [21].

1. *Throughput*

Throughput merupakan parameter untuk menghitung kecepatan transfer data suatu paket diamati pada tujuan selama interval waktu tertentu, kemudian dibagi dengan durasi interval waktu. Kecepatan *transfer* data diukur dalam *bps* (*bit per second*). Rumus perhitungan *throughput* ditunjukkan pada persamaan 2.1[22].

$$Throughput = \frac{Paket\ data\ diterima(bytes)}{Lama\ pengiriman(sec)} \quad (2.1)$$

Pengelompokkan standarisasi parameter *throughput* berdasarkan standarisasi TIPHON ditunjukkan pada tabel 2.2.

Tabel 2.2 Tabel kategori nilai *throughput* TIPHON

Kategori *Throughput*	*Throughput* (Mbit/s)	Indeks
Sangat Bagus	>2,1 Mbps	4
Bagus	1200 kbps – 2,1 Mbps	3
Cukup	700-1200 kbps	2
Kurang Bagus	338-700 kbps	1
Buruk	0-338 kbps	0

2. *Delay*

Delay merupakan nilai waktu keterlambatan paket data dalam menempuh jarak dari *client* ke *server* atau sebaliknya. Parameter ini merupakan faktor yang mempengaruhi jarak, waktu proses lama[21]. Rumus perhitungan *delay* ditunjukkan pada persamaan 2.2[22].

$$Delay = \frac{Total\ delay(ms)}{Total\ paket\ yang\ diterima} \quad (2.2)$$

Pengelompokkan standarisasi parameter *delay* berdasarkan standarisasi TIPHON ditunjukkan pada tabel 2.3.

Tabel 2. 3 Tabel kategori nilai *delay* TIPHON

Kategori *Delay*	*Delay* (ms)	Indeks
Sangat Bagus	<150 ms	4
Bagus	150 sd 300 ms	3
Cukup	300 sd 450 ms	2
Buruk	>450 ms	1

Selain itu pengelompokkan standarisasi parameter *delay* berdasarkan standarisasi ITU-G.1010 ditunjukkan pada tabel 2.4

Tabel 2.4 Tabel kategori nilai *delay* ITU-G

Kategori *Delay*	*Delay* (ms)	Indeks
Sangat Baik	0-150 ms	4
Baik	150-300 ms	3
Cukup	300-450 ms	2
Buruk	>450 ms	1

3. *Jitter*

Jitter atau variasi kedatangan paket yang diakibatkan oleh variasi-variasi panjang antrian, waktu pengolahan data, serta waktu penghimpunan ulang paket-paket di akhir perjalanan *jitter*. Apabila semakin kecil nilai kualitas sebuah jaringan menandakan bahwa kualitas jaringan semakin bagus. Rumus perhitungan *jitter* ditunjukkan pada persamaan 2.3[21].

$$Jitter = \frac{Total\ variasi\ delay}{Total\ paket\ diterima} \quad (2.3)$$

Nilai dari total variasi delay didapatkan melalui persamaan 2.4 berikut:

Total Variasi Delay = Delay – (rata-rata delay) (2.4)

Pengelompokkan standarisasi parameter *jitter* berdasarkan standarisasi TIPHON ditunjukkan pada tabel 2.5.

Tabel 2.5 Tabel kategori nilai *jitter* TIPHON

Kategori *Jitter*	*Jitter* (ms)	Indeks
Sangat Bagus	0 ms	4
Bagus	0 sd 75 ms	3
Cukup	75 sd 125 ms	2
Buruk	125 sd 225 ms	1

Selain itu pengelompokkan standarisasi parameter *jitter* berdasarkan standarisasi ITU-G.1010 ditunjukkan pada tabel 2.6.

Tabel 2.6 Tabel kategori *jitter* ITU-G

Kategori *Jitter*	*Jitter* (ms)	Indeks
Sangat Baik	0 ms	4
Baik	0-75 ms	3
Cukup	76-125 ms	2
Buruk	126-225 ms	1

4. *Packet Loss*

Packet loss merupakan gambaran kondisi menunjukan jumlah total paket yang hilang atau dapat juga sebagai kegagalan transmisi paket IP dalam mencapai tujuan. Kegagalan paket yang terjadi dapat disebabkan oleh beberapa hal seperti tabrakan pada jaringan, *overload* pada trafik jaringan, *error* pada media fisik, dan apabila kegagalan terjadi pada sisi *receiver*, disebabkan karena *overflow* pada *buffer*. Rumus perhitungan *packet loss* ditunjukkan pada persamaan 2.5[21].

$$Packet\ Loss = \frac{(Paket\ data\ dikirim - Paket\ data\ diterima) x 100\%}{Paket\ data\ yang\ dikirim} \quad (2.5)$$

Pengelompokkan standarisasi parameter *packet loss* berdasarkan standarisasi TIPHON ditunjukkan pada tabel 2.7.

Tabel 2.7 Tabel kategori nilai *packet loss* TIPHON

Kategori *Degradasi*	*Packet Loss* (%)	Indeks
Sangat Bagus	0%	4
Bagus	3%	3
Cukup	15%	2
Buruk	25%	1

Selain itu pengelompokkan standarisasi parameter *packet loss* berdasarkan standarisasi ITU-G.1010 ditunjukkan pada tabel 2.8.

Tabel 2.8 Tabel kategori *packet loss* ITU-G

Kategori *Packet Loss*	*Packet loss* (%)	Indeks
Sangat baik	0%	4
Baik	3%	3
Cukup	15%	2
Buruk	25%	1

5. *Response Time*

Response time merupakan pengujian untuk melihat waktu *respons*, mengacu pada waktu yang dibutuhkan satu *node* sistem dalam menanggapi setiap permintaan. *Response time* mengukur kinerja setiap *request*, waktu *response* dimulai pada saat *client* mengirimkan permintaan dan berakhir pada saat tidak terjadi koneksi atau permintaan tersebut telah selesai. Besaran *response time* diukur dalam *kilobyte* per detik (kb/*sec*) [2].