BAB II
DASAR TEORI

1.1.Kajian Pustaka

Pada penelitian Rifaldi dan Helmi Faisal di tahun 2019, melakukan implementasi *load balancing* menggunakan algoritma *round robin* pada *load balancer* F5 BIG-IP LTM dengan parameter uji adalah *respon time*. Hasil penelitian tersebut menunjukan hasil yang baik karena *availability* yang tetap terjaga tanpa terjadi *overload* pada ketiga *server* dalam pengujian beban *traffic* 10 *request*, 100 *request*, dan 1000 *request*.

Pada penelitian yang dilakukan oleh Ilham Reza Wijaya pada penelitianya di tahun 2019, melakukan analisis kinerja *load balancing* menggunakan algoritma *dynamic ratio* dan membandingkannya dengan algoritma *round robin* pada *load balancer* F5 BIG-IP 13 dengan empat parameter uji yaitu *throughput*, *response time, error,* dan CPU *utilization.* Dari hasil pengujian menunjukan dengan menggunakan 3 *server* tidak terjadi *overload* pada *server* pada saat pengujian algoritma *dynamic ratio* maupun *round robin.* Nilai rataan pada algoritma *dynamic ratio* sebesar 57,83 KB/s dan pada *round robin* sebesar 55,27 KB/s, nilai rataan *response time* pada algoritma *dynamic ratio* sebesar 2.64 detik dan *round robin* sebesar 0,0% dan algoritma *round robin* 0,9% sedangkan nilai CPU *utilization* pada algoritma *dynamic ratio* sebesar 93,5% dan algoritma *round robin* sebesar 93,0%. Nilai *Fairnes index* pada algoritma *dynamic ratio* tidak mencapai angka 1 sedangkan pada algoritma *round robin* dapat mencapai angka 1.

Pada penelitian yang dilakukan irmawati, dkk pada tahun 2017 [2] melakukan penelitian dengan membandingkan antara simulasi jaringan SDN dan implementasi jaringan SDN menggunakan protokol *routing* OSPF. Topologi yang digunakan *partial mesh* . Nilai QoS dan *convergence time* yang didapat pada simulasi hasilnya tidak jauh berbeda pada nilai implementasinya. Penelitian yang dilakukan Nugroho, K dan Setyanugroho, D.P tahun 2019 [1] melakukan simulasi jaringan SDN menggunakan protokol *routing* OSPF dan teknologi yang digunakan *full mesh.* Hasil pengukuran kualitas jaringan menunjukkan bahwa penggunaan protokol UDP menghasilkan nilai parameter QoS yang lebih baik dibandingkan dengan TCP. Nilai waktu konvergensi pada ukuran data sebesar 64 Byte adalah 12,18 ms.

Pada penelitian Ridha Muldina Negara yang berjudul "Analisis Simulasi Penerapan Algoritma OSPF Menggunakan *RouteFlow* pada Jaringan *Software Defined Network* (SDN)" membahas mengenai analisis dan simulasi protokol *routing* OSPF pada teknologi SDN dengan *RouteFlow* sehingga mempermudah dalam pengontrolan jaringan dengan sistem terpusat, untuk melihat apakah peroutingan OSPF dapat berjalan dengan baik di jaringan *Software Defined Network* (SDN), *RouteFlow* yang digunakan dalam penelitian ini adalah *RouteFlow* dengan *Controller POX* dan *OpenFlow*. Pada penelitian ini parameter yang diukur adalah *time convergence* dan *Quality of Service* (QoS) menggunakan standar nilai performasi sesuai ITU.T G1010 dengan menggunakan empat topologi dengan jumlah *switch* berbeda, yaitu 4,6,8 dan 10 *switch*. Hasil pengujian QoS pada topologi 4,6,8 dan 10 *switch* parameter *delay, jitter, packet loss,* dan *throughput* masih dalam rentang nilai yang ditetapkan pada ITU-T G.1010 [4].

Pada penelitian Roni Fernando Simarmata yang berjudul "Simulasi Jaringan *Software Defined Network* Menggunakan Protokol *Routing* OSPF dan RYU *controller*" membahas mengenai penerapan kinerja routing OSPF pada SDN dengan menggunakan RYU *controller*. Penelitian ini mensimulasikan *routing* OSPF dengan menggunakan RYU *controller* dengan simulasi 8 buah *switch openflow*, dan dilakukan pengukuran QoS pada jaringan ini untuk melihat perbandinganya dengan jaringan konvensional dan analisis *background traffic* yang didapat jaringan SDN lebih handal dapat dilihat dari nilai *packet loss, jitter*, dan *troughput* yang lebih baik dari jaringan konvensional dikarenakan *control plane* dan *data plane* jaringan SDN ini terpisah tidak seperti pada jaringan konvensional yang masih menyatu *control plane* dan *data plane* [3].

2.2 Jaringan Komputer

Jaringan komputer merupakan Kumpulan beberapa perangkat seperti PC, printer, LAN Card, *Router* dan lain-lain yang membentuk sebuah jaringan yang terintegrasi satu sama lain. Pada dua buah komputer dapat dikatakan terhubung jika keduanya dapat mengirimkan data atau informasi dan dapat membalasnya[31].

1. *Local Area Network* (LAN)

   Local Area Network (LAN) adalah sebuah jaringan perangkat atau node yang memiliki lingkup *local* yang terbatas contohnya pada satu Gedung komplek atau Gedung kampus bertujuan untuk berbagi resource (*Resource Sharing*). Dalam sebuah jaringan

LAN memiliki minimal perangkat yang digunakan terdapat dua buah perangkat agar dapat saling bertukar data adtara keduanya dimana kedua perangkat tersebut harus memiliki kartu jaringan atau LAN Card.

2. *Metropolitan Area Network* (MAN)

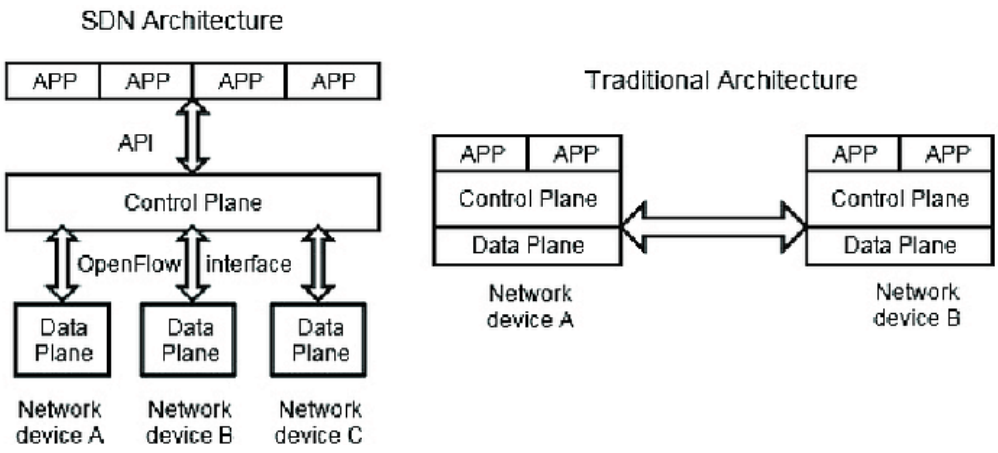
   Metropolitan Area Network (MAN) merupakan versi lain dari LAN dengan jangkauan wilayah yang lebih luas, tetapi cakupan areanya lebih kecil dibandingkan dengan WAN. Teknologi yang biasanya digunakan oleh MAN sama dengan teknologi yang digunakan oleh LAN. MAN dapat menghasilkan komunikasi berupa data dan suara, MAN juga bisa terhubung dengan tv kabel.

3. *Wide Area Network* (WAN)

   Wide Area Network (WAN) adalah kumpulan jaringan LAN yang terhubung dengan alat telekomunikasi, biasanya menggunakan modem sebagai alat perantaranya. Untuk bisa terhubung pada jaringan antar kantor pusat-cabang ataupun antar cabang. Jaringan WAN juga dapat terhubung dengan memanfaatkan jaringan umum namun dengan memperlihatkan masalah keamanan[31].

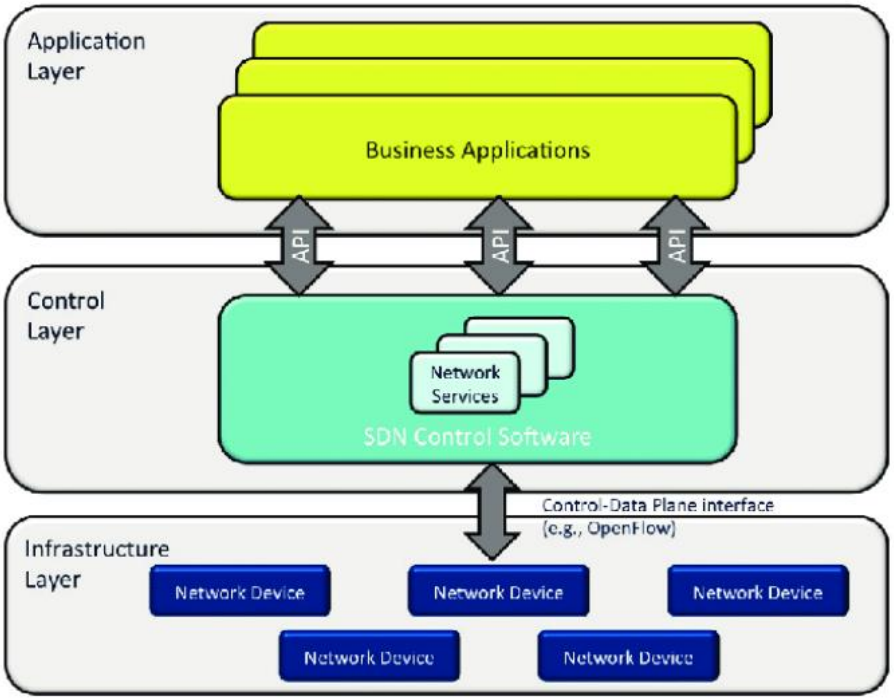
2.3 Software Defined Network

Software defined Network (SDN) adalah sebuah arsitektur baru dalam bidang jaringan komputer memiliki karakteristik yang *dinamis*, *manageable*, *cost-effective*, dan *adaptable*[11] sehingga sangat cocok untuk memenuhi kebutuhan aplikasi saat ini yang bersifat dinamis dan memiliki *bandwitdh* tinggi[13]. SDN juga merupakan sebuah pendekatan baru untuk membangun, mendesain serta untuk mengatur jaringan komputer.*Open Networking Foundation* (organisasi pengembang SDN) mendefinisikan SDN adalah sebuah arsitektur jaringan dimana *control network* dipisahkan dari *forwaringnya system*. sehingga network control tersebut dapat digunakan secara langsung[20]. *Software Defined Network* (SDN) adalah konsep pendekatan jaringan komputer dimana sistem pengontrol (*control plane*) dari arus data dipisahkan dari perangkat kerasnya (*data plane*)(Sanjoyo et al., 2020).



Gambar 2. 1 Arsitektur SDN

SDN adalah suatu metode atau paradigma yang memisahkan antara *Control Plane* dan *Forwarding Plane*. Pada jaringan tradisional atau jaringan non-SDN, kedua fungsi diatas berada dalam satu perangkat yang sama. Pada teknologi SDN terdapat protokol *openflow* dan *netconf*, protokol tersebut digunakan untuk mengontrol *traffic flows*, namun *openflow* dan *netconf* ini sudah ditentukan dan tidak dapat ubah atau dimodifikasi. *Programming Protocol-independent Packet Processors* (P4) adalah sebuah bahasa pemograman untuk *top down* programming yang dapat menentukan bagaimana *pipelines* pada *switch* bekerja dan bagaimana paket-paket ini dapat diproses [18]. *Controller Plane* merupakan yang utama dalam jaringan SDN, dapat dijalankan secara terpisah dari *Data Plane*. Sedangkan *data plane* adalah *hardware* jaringan yang terprogram secara khusus dan dikendalikan penuh oleh *Data Plane*. pada Software Defined Network (SDN) tersedia *open interface* yang sebuah *entitas software* atau aplikasi untuk mengendalikan konektifitas dari sumber daya jaringan, mengendalikan *traffic* jaringan, serta memodifikasi di *traffic* jaringan tersebut.



Gambar 2. 2 Layer SDN

1. Application Layer

   Application layer berperan sebagai antarmuka untuk memudahkan pengelolaan jaringan dalam melakukan fungsi konfigurasi, fungsi kontrol dan fungsi evaluasi. Layer ini terbentuk dari integritas dengan control layer yang memberikan informasi tentang jaringan kepada pengelola jaringan.

2. Control Layer

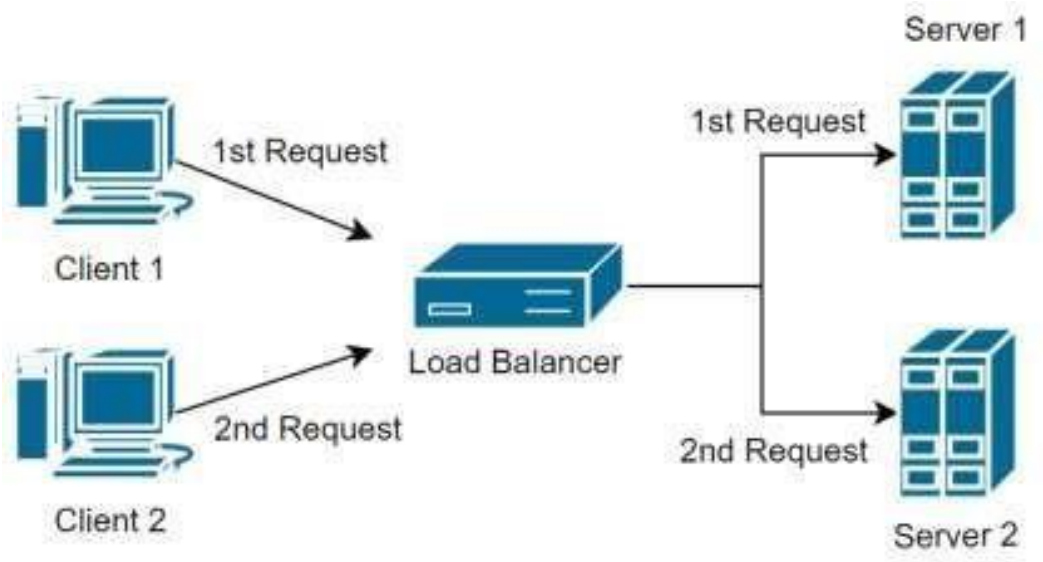
   Control layer berfungsi sebagai *control* seluruh aktifitas jaringan.layer ini fungsi dari *control plane* bekerja.

3. Infrastruktur Layer

   Infrastruktur layer merupakan lapisan dasar dari arsitektur SDN. Lapisan ini berisi perangkat keras *forwading plane* yang berfungsi sebagai *data plane*. Perangkat keras ini berupa *switch* dan *router*[20].

2.4 Load Balancing

Load Balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server.[8] Load balancing bertujuan untuk optimalisasi sumber daya, memaksimalkan *troughput*, meminimalkan waktu repon, dan menghindari pembebanan berlebih pada satu sumber daya. Menggunakan beberapa sumber daya komputasi juga dapat mengurangu kurangnya berfungsinya suatu layanan karena disetiap sumber daya dapat saling menggantikan (*redundant*). *Load balancing* juga dapat membagi trafik jaringan dengan adil dan mengurangi terjadinya *overload* pada salah satu *server*[6], namun sering terjadi diskoneksi untuk aplikasi realtime dikarenakan perpindahan *gateway* pada setiap jaringan yg menuju ke *server*. *Load Balancing* juga dikenal teknik untuk mempersingkat waktu tanggap dan menghindari *overload* pada salah satu koneksi[17].



Gambar 2. 3 Gambaran Umum *Load Balancing*

Gambar 2.1 merupakan arsitektur komputer yang digabungkan dengan *Load Balancing* yang kemudian dibagi menjadi beberapa *device* yaitu *client, load balancer*, dan *server*.

Ada beberapa keuntungan yang diperoleh dari teknik *Load Balancing* sebagai berikut :

1. *Flexibility*

   Server tidak lagi menjadi inti sistem dan resource utama, tetapi menjadi bagian dari banyak server yang membentuk cluster. Hal ini membuat bahwa performa per-unit dari *cluster* tidak terlalu diperhitungkan, tetapi performa *cluster* secara keseluruhan. Sedangkan untuk

meningkatkan performa dan *cluster, server* atau unit baru dapat ditarnbahkan tanpa mengganti unit yang lama.

2. *Scalability*

   Sistem tidak memerlukan desain ulang karena seluruh arsitektur sistem dapat mengadaptasikan sistem tersebut ketika terjadi perubahan pada komponen sistem. Untuk semua trafik yang melewati *Load Balancer*, aturan keamanan dapat dimplementasikan dengan mudah. Dengan *Private Network* digunakan untuk *Real server*, alamat Ipnya tidak akan diakses secara langsung dari luar sistem *cluster*.

3. *Hight-availability*

   Load balancer dapat mengetahui kondisi *Real server* dalam sistem secara otomatis, jika terdapat real server yang mati maka akan dihapus dari daftan *Real server*, dan jika *Real server* tersebut kernbali aktif maka akan dimasukkan ke dalam daftar *Real server*.[14] *Load Balancer* juga dapat dikonfirmasikan redundant dengan *Load Balancer* yang lain.

2.5 POX Controller

POX merupakan sebuah platform pengembangan *open source* untuk aplikasi *Software Defined Network* (SDN) berdasarkan pada bahasa *Python* dan merupakan *controller Open Flow*[10]. Dengan menggunakan POX dapat memungkinkan proses perancangan dan pembangunan jaringan yang dapat berjalan dengan cepat, dan juga lebih umum untuk digunakan dari pada pendahulunya.

Gambar 2. 4 POX

Daftar fitur POX sebagai berikut :

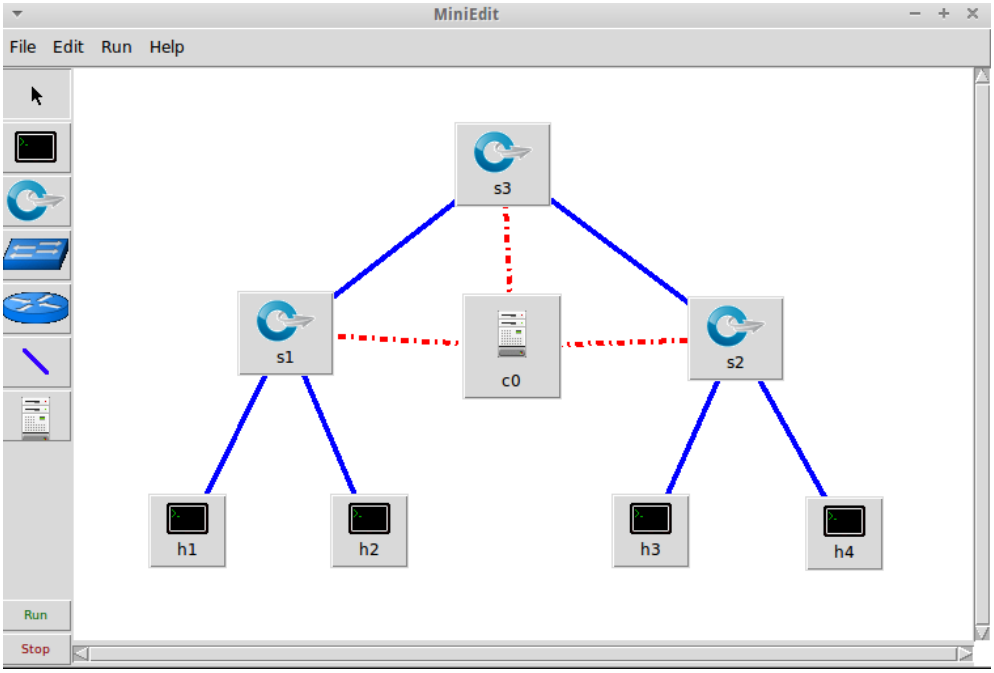
1. Tampilan menyerupai *python* digunakan untuk antarmuka *OpenFlow* pada grafik kinerja dari POX.
2. Dapat dijalakan disemua sistem. Secara khusus dapat dioperasikan untuk sistem operasi *Windows, Linux,* dan *Mac Os.*
3. Mendukung tampilan GUI dan visualisasi yang sama seperti NOX [27].

2.6. Mininet

Mininet merupakan sebuah emulator berbasis *Command Line Interface* (CLI) untuk membuat *prototype* jaringan yang berskala besar secara cepat pada sumber daya yang sangat terbatas. Mininet menggunakan virtualisasi yang berbasis proses untuk menjalankan banyak *host* dan *switch* pada satu *kernel OS*. Mininet memanfaatkan virtual *software switch OpenvSwitch* untuk membangun topologi *OpenFlow* SDN (*Software Defined Network*). Mininet juga digunakan

untuk membuat simulasi jaringan SDN dengan berbagai macam teknologi.[19] Mininet dibuat dengan tujuan untuk mendukung riset dibidang SDN. Mininet sendiri dapat memungkinkan untuk menjalankan sebuah kode secara interaktif tanpa harus memodifikasi kode tersebut[10]. Mininet juga dapat digunakan sebagai solusi yang dianggap paling unggulmdalam berbagai hal seperti kemudahan penggunaan, akurasi dan skalabilitas. Mininet dapat menyediakan konfigurasi yang realitas dan mudah dengan harga yang murah. Dibandingkan dengan *hardware testbed* yang cenderung lebih mahal, sangat sulit untuk dikonfigurasi ulang, namun lebih akurat dibandingkan dengan menggunakan emulator Mininet.

Mininet merupakan solusi yang dianggap paling unggul dalam hal kemudahan penggunaan, performansi, akurasi, dan skalabilitas. Mininet juga mampu menyediakan lingkungan yang reakistis dan nyaman dengan harga yang lebih murah. Dengan mininet kita dapat menggunakan alternatif lain seperti hardware test-bed untuk simulasi jaringan, dimana dapat berjalan kencang dan akurat, tetapi harganya mahal dan harus di share oleh penggunanya.



Gambar 2. 5 Mininet

Salah satu alasan pengguna lebih menggunakan mininet karena dengan mininet dapat membantu dalam topologi sesuai dengan kebutuhan atau keinginan pembuat, banyak yang sudah menggunakanya, lebih besar dari topologi internet. Fitur lain yang bagus dari mininet yaitu

memungkinkan untuk kostuminsasi packet forwading sepenuhnya [29]. Berikut adalah kelebihan mininet :

1. memungkinkan beberapa pengembang untuk bekerja sama secara independent pada topologi yang sama.
2. Menyediakan testbed jaringan yang sederhana dan murah untuk mengembangkan aplikasi OpenFlow
3. Memungkinkan pengujian topologi yang kompleks tanpa perlu memasang jaringan fisik.
4. Mendukung system-level regression test yang berulang dan mudah dikemas
5. Mendukung pembuatan topologi custom secara acak, sehingga dapat memudahkan pengguna dalam membuat topologi berdasarkan kebutuhan pengguna.
6. Menggunakan CLI pada sebuah topologi dan OpenFlow untuk debugging atau melakukan tes pada jaringan luas[29].

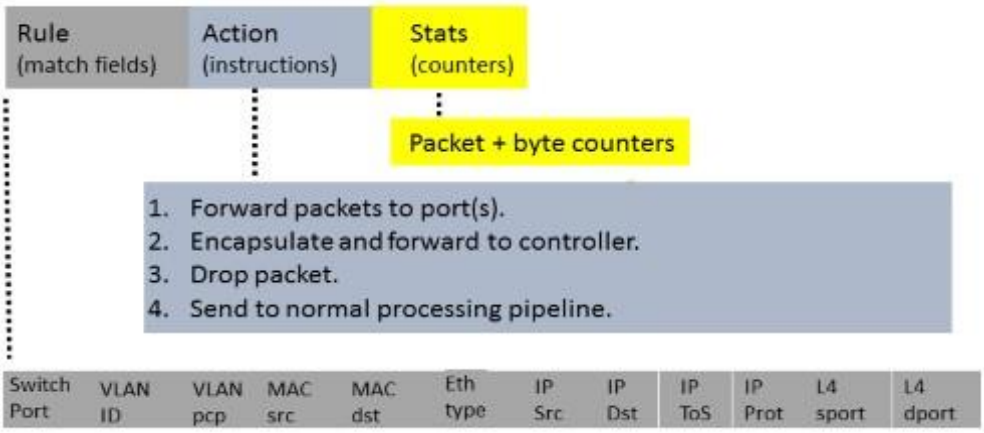
2.7. HYBRID SDN

Hybrid SDN adalah proses transisi jaringan tradisional ke SDN yang menggabungkan kontrol pusat terprogram dengan distribusi *routing*. Operator dapat menyeimbangkan beban dan mengatur jaringan lebih mudah. *Controller* dalam jaringan SDN hanya dapat difokuskan pada arus atau lalu lintas data. Sementara Sebagian dari paket dikelola oleh *protocol* tradisional. *Operator* hanya perlu melakukan migrasi perangkat akses lama ke *switch SDN*[7]. Dalam jaringan *hybrid* SDN yang ditunjukan pada gambar, lalu lintas diteruskan berdasarkan keputusan mekanisme *switching* dan *routing* terdistribusi, kontrol lalu lintas di *hybrid* SDN tergantung pada berbagai strategi penyebaran seperti (*Island-based, service-based, controller-based, Hal-based, Agent-based dan overlay-based*) dalam hal ini, Kontrol lalu lintas akan diteruskan melalui SDN dan sejumlah lalu lintas akan dikendalikan melalui SDN, lalu lintas yang lain dikendalikan dengan mekanisme tradisional, dan pilihan tentang cara meneruskan dan memproses lalu lintas jaringan diambil oleh pengontrol yang terpusat secara logis. Dalam *hybrid* SDN dan *legacy network* masih menggunakan *protocol* tradisional seperti ECMP *(Equal Cost Multipath*), *Open Shortest Path First* (OSPF) untuk meneruskan paket, hanya

perangkat *Software Defined Network* yang akan dikendalikan oleh pengontol terpusat. Hal ini disebut dengan *hybrid SDN*. SDN memiliki perbedaanya sendiri dan tantangan yang tidak ada di *Pure SDN*.

2.8. OPEN FLOW

Open Flow merupakan sebuah sandar *interface* dari *protocol* komunikasi jaringan yang memiliki fungsi untuk menghubungkan antara lapisan controller dengan lapisan forwading pada arsitektur SDN. *OpenFlow* dapat mengijinkan akses langsung dan dapat memanipulasi *forwading plane* dari sebuah perangkat keras jaringan seperti *switch* dan *router* baik fisik maupun virual. Secara sederhana *OpenFlow* merupakan sebuah antarmuka antara SDN *controller* dengan perangkat *switch.*[11]



Gambar 2. 6 *Open Flow*

2.9. QUALITY OF SERVICE

Quality Of Service (QoS) merupakan kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan bandwidth, mengatasi *jitter* dan *delay*. Tujuan QoS adalah untuk menyediakan kualitas layanan yang berbeda beda untuk beragam kebutuhan akan layanan di dalam jaringan IP, sebagai contoh untuk menyediakan *bandwidth*, menurunkan hilangnya paket-paket, menurunkan waktu jeda di dalam proses transmisinya[27].

2.10. THROUGHPUT

Bandwidth merupakan suatu nilai transfer data yang terhitung dalam bit/detik antara *server* dengan *client* pada waktu tertentu. Tujuan dengan digunakanya *bandwidth* ini untuk menghitung banyaknya pertukaran data [16]. Sedangkat *troughput* merupakan *bandwidth* aktual yang terukur

pada suatu waktu yang menggunakan rute internet spesifik ketika sedang melakukan transaksi data. Dengan menggunakan troughput dapat diukur efektifitas penggunaan *bandwidth* pada jaringan. Semakin tinggi nilai *average troughput* yang dihasilkan maka semakin tinggi pula efektifitas *bandwidth* [19].

2.11. DELAY

Delay merupakan waktu tunggu yang terbuang untuk menunggu antrian yang dibutuhkan dalam pengiriman paket data dari pengirim ke penerima. *Delay* yang terjadi dipengaruhi oleh banyak factor seperti jarak, media perantara, proses transmisi dan lainya [11].

2.12. PAKET LOSS

Paket loss didefinisikan sebagai kegagalan dari transmisi paket ip yang mencapai tujuanya. paket loss memiliki satuan %. Ada beberapa factor yang mempengaruhi kegagalan paket untuk mecapai tujuan, sebagai berikut :

a. Tabrakan dalam jaringan
b. Terjadinya kelebihan *traffic* didalam jaringan
c. Error yang terjadi pada media fisik
d. Kegagalan yang terjadi pada sisi penerima bisa disebabkan karena *overflow* yang terjadi pada *buffer* [9].

2.13. SWITCH

Switch merupakan perangkat yang secara tipikal mempunyai beberapa *port* untuk menghubungkan beberapa *segmen* LAN lain yang mempunyai kecepatan rendah. *Switch* pada prinsripnya sama seperti *hub*. Perbedaanya adalah *switch* dapat beroperasi dengan mode *half-duplex* dan mampu mengalihkan jalur dan memfilter informasi ke dan dari tujuan yang spesifik. Ada dua jenis arsitektur dasar yang digunakan pada *switch*, yaitu *cut-trough* dan *store-and-forward.* Saat paket datang, *switch* hanya memperhatikan alamat tujuanya sebelum meneruskan paket ke segmen tujuan. Sedangkan pada *switch store-and-forward*, saat menerima paket, isi paket akan dianalisa terlebih dahulu sebelum meneruskanya ke alamat tujuan, sehingga memungkinkan *switch* untuk mengetahui adanya kerusakan pada paket dan mencegahnya agar tidak mengganggu kerja jaringan[24]

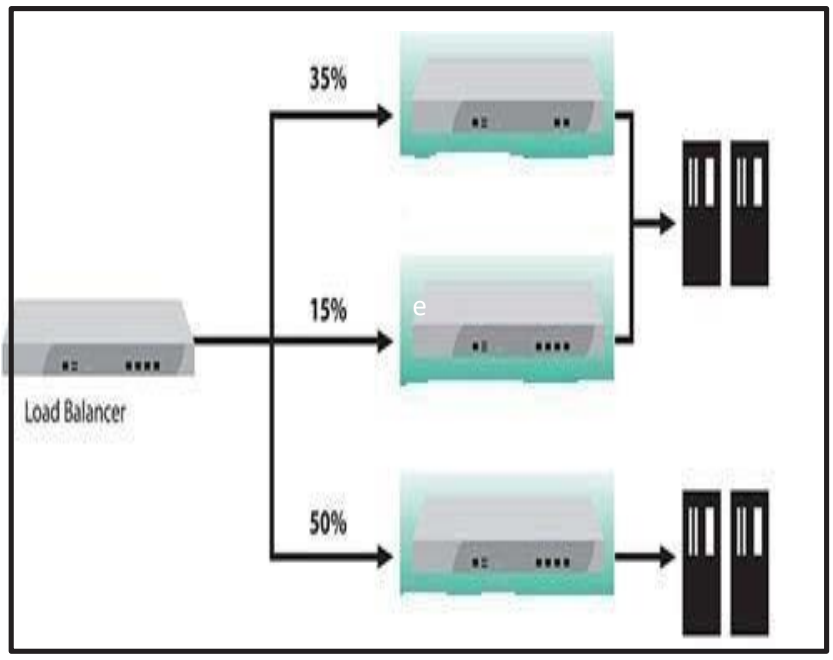
Gambar 2. 7 Switch

2.14. CLIENT

Komputer *client* adalah komputer yang digunakan untuk meminta layanan tertentu dari komputer *server*. Layanan tersebut bisa jadi data, *file*, gambar, *printer*, maupun yang lainya. Pada komputer *client* dibutuhkan suatu aplikasi tertentu agar dapat mengakses layanan dari komputer *server*. Akses yang diberikan komputer *client* pun cukup cepat karena tidak melakukan tugas lain dalam waktu yang bersamaan, seperti halnya komputer *server*. Oleh karena itu, komputer cukup baik dugunakan sebagai sebuah sistem keamanan dan administrasi Perusahaan, sebab akses yang dilakukan dapat dibatasi[24].

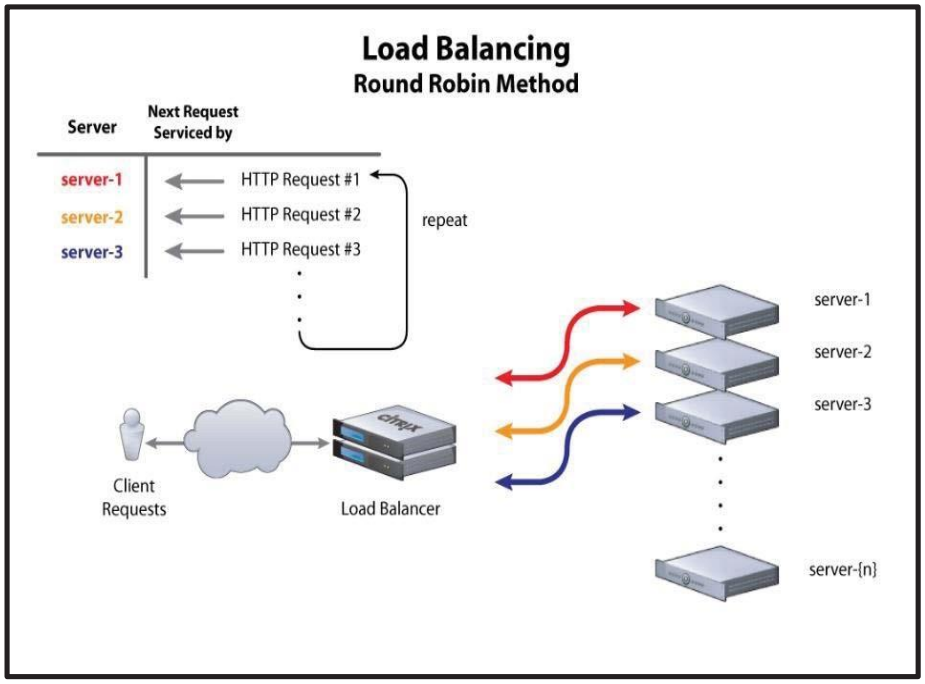
2.15. ALGORITMA RATIO

Algoritma *ratio* menggunakan parameter *ratio* pada masing masing server di dalam sistem *load balancing*. Parameter *rasio* ini akan menjadi landasan pembagi beban pada *server-server* yang terlibat. *Server* dengan *rasio* terbesar diberi beban besar, sebaliknya *server rasio* kecil akan lebih sedikit diberi beban seperti yang ditunjukan pada gambar



2.16. ALGORITMA ROUND ROBIN

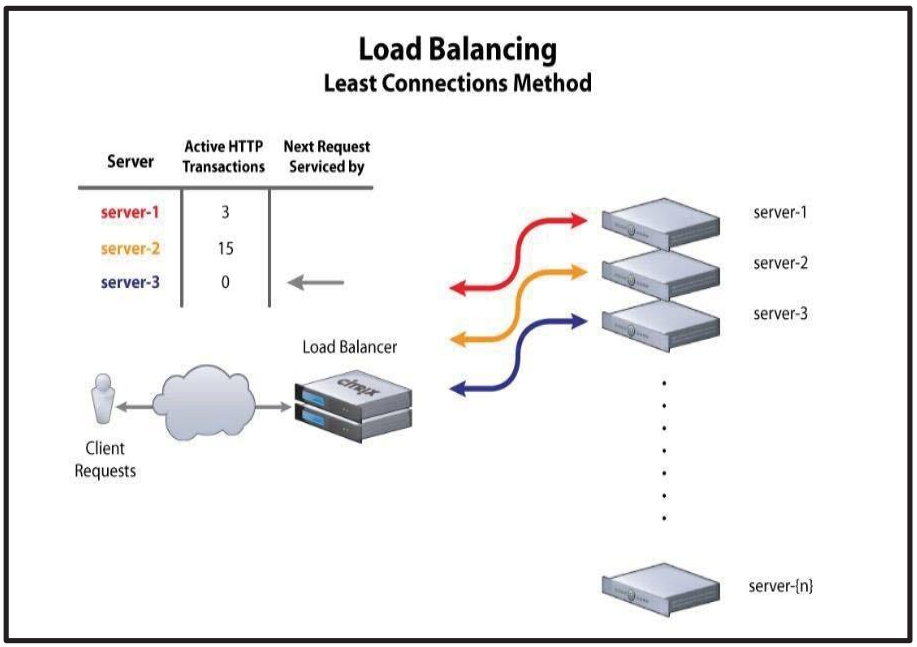
Round Robin merupakan algoritma paling sederhana dan paling banyak digunakan oleh perangkat *load balancing*. Algoritma *round robin* bekerja dengan cara membagi beban secara berurutan dan bergiliran dari satu *server* ke *server* lain. Konsep dasar dari algoritma *round robin* ini adalah dengan menggunakan *time sharing*, karena algoritma ini memproses antrian secara bergiliran[17].



2.17. ALGORITMA LEAST CONNECTION

Pada gambar , ditunjukkan algoritma *least connection* dimana algoritma ini melakukan pembagian beban *traffic* berdasarkan pada banyaknya koneksi yang sedang dilayani oleh *server*. *Server* dengan minim koneksi akan diberikan beban berikutnya, dan juga apabila *server* dengan koneksi sibuk akan dialihkan bebanya kepada *server* yang koneksi lebih rendah.

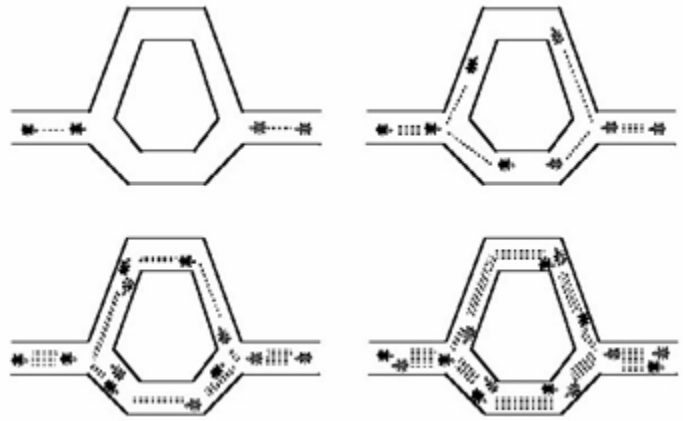
Algoritma ini termasuk dalam algoritma penjadwal dinamis, karena memerlukan perhitungan koneksi aktif untuk masing-masing *real server* secara dinamik. Mekanisme algoritma ini apabila merujuk pada gambar . terdapat 3 *server*, contoh terdapat dua *service-HTTP* dari 3 *server* tersebut. *Service HTTP* tersebut berada di *server* 1 dan *server* 2. Apabila *client request* dan proses *request* masih menerima *server* 2 akan dialihkan ke *server* 1 dikarenakan *server* 2 masih dalam proses [17].



Gambar proses least connection

2.18. ALGORITMA KOLONI SEMUT

Koloni semut merupakan algoritma yang bersifat heuristik untuk menyelesaikan masalah optimasi. Algoritma ini diinspirasikan oleh lingkungan koloni semut pada saat mencari makanan. Semut dapat mencari lintasan terpendek dari sumber makanan menuju sarangnya tanpa harus melihatnya secara langsung. Semut mempunyai cara penyelesian yang unik dan maju, yaitu dengan menggunakan jejak pheromon pada satu jalur untuk berkomunikasi dan membangun solusi, semakin banyak jejak *pheromon* ditinggalkan maka akan diikuti semut lain[15].



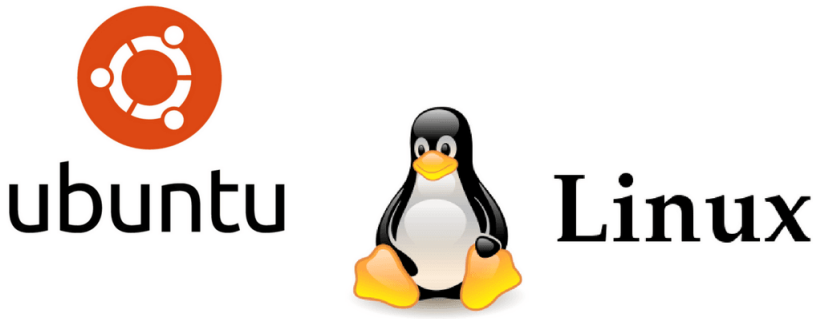
2.19. SISTEM OPERASI

Sistem operasi merupakan software yang digunakan untuk mengalokasikan *control* eksekusi program dan aplikasi sekaligus sebagai tampilan *interface* antara user pengguna *computer* dengan *hardware computer*. Sistem operasi memiliki catatan sistematis seperti pemrosesan data, perhitungan penggunaan memori, penyimpanan data dan lainya[12].

Gambar 2. 8 Sistem Operasi

2.20. LINUX UBUNTU

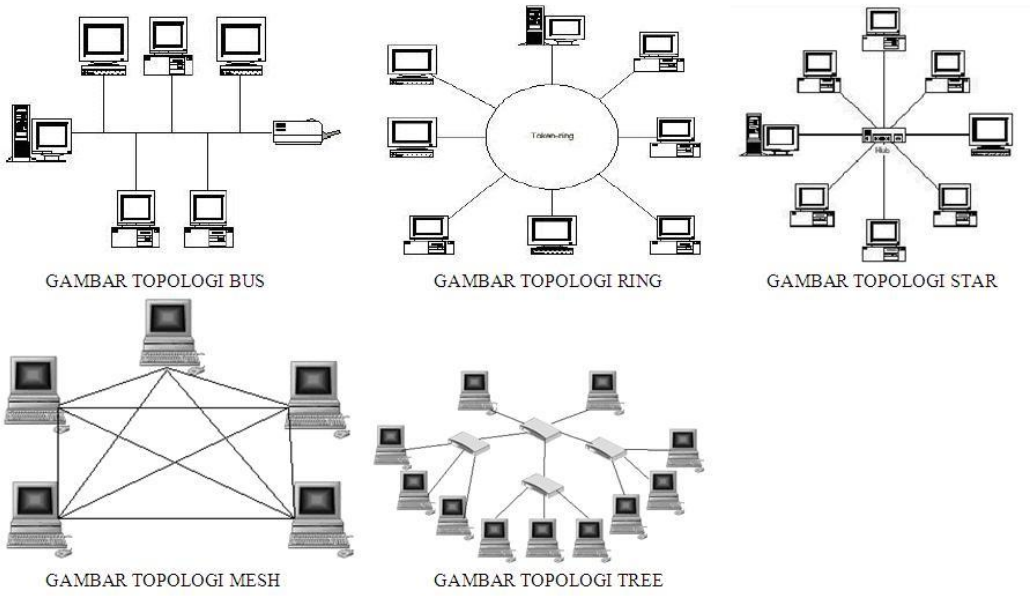
Linux merupakan aplikasi atau sebuah program yang menggunakan kernel sebagai sistem operasi yang bersifat *opensource* dan memiliki banyak varian atau distro seperti *Debian, ubuntu, redhat, mint* dan lainya. *Ubuntu* merupakan salah satu distribusi dari *LINUX* yang basisnya pada *Debian* dan didistribusikan secara *open source. Ubuntu* dirancang untuk menunjang kepentingan pengguna pribadi, *server* maupun pengguna umum[12].



Gambar 2. 9 Linux Ubuntu

2.21. TOPOLOGI

Topologi merupakan istilah yang digunakan untuk menguraikan cara computer terhubung satu sama lain dalam suatu jaringan. Ada 3 topologi yang biasa digunakan yaitu *ring, bus, mesh, dan star*[24].



Gambar 2. 10 Topologi Jaringan

A. Topologi ring

Topologi *ring* igunakan dalam jaringan dengan performa yang tinggi, karena membutuhkan *bandwidth* yang besar untuk beberapa fitur yang digunakan. Pada topologi *ring*, masing-masing titik memiliki fungsi sebagai *repeater*.

B. Topologi Bus

Topologi *Bus* merupakan topologi jaringan yang pertama kali digunakan untuk menghubungkan komputer. Media transmisi yang digunakan untuk menghubungkan yaitu dengan menggunakan sebuah kabel panjang dengan beberapa terminal yang nantinya akan terhubung ke masing-masing komputer, dan pada ujung kabel harus diakhiri dengan satu *terminator*. Topologi jaringan ini sudah sangat jarang dipakai di suatu perusahaan maupun instasi karena resiko yang ditimbulkan terlalu besar. Mulai dari tingginya resiko tabrakan data, dan jika ada suatu perangkat komputer yang rusak, maka jaringan langsung tidak bisa berfungsi sebelum diperbaiki.

C. Topologi Mesh

Topologi *Mesh* gabungan dari topologi *ring* dan topologi *star*. Secara pengertian topologi *mesh* adalah suatu bentuk hubungan antar perangkat/pc dimana masing-masing perangkat terhubung secara langsung ke perangkat lainya dalam jaringan.

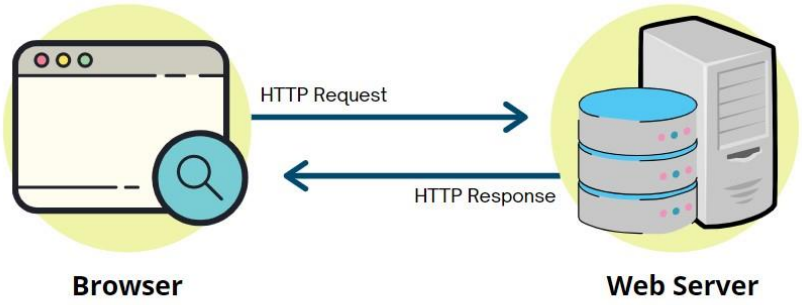
D. Topologi Star

Topologi *Star* memiliki bentuk yang sama seperti bintang, dengan *hub* sebagai media penghubung ke setiap perangkat komputer seperti gambar topologi *star* diatas. Topologi *Star* biasa digunakan dalam lab komputer di suatu sekolah.

2.22. WEB SERVER

Web *server* merupakan perangkat lunak yang melayani *request HTTP* baik dari *client* maupun dari *server* dan web browser. Kemudian, *web server* akan mengirimkan akan mengirimkan kode-kode dinamis ke *server* aplikasi. *Server* aplikasi inilah yang menerjemahkan dan memproses kode-kode dinamis menjadi kode-kode statis HTML dalam suatu halaman statis yang kemudian dikirimkan ke *browser* oleh *web server* seperti yang ditunjukkan pada gambar. .

Web server biasanya disebut juga dengan *HTTP server*, karena basisnya menggunakan *protocol HTTP*.[25]



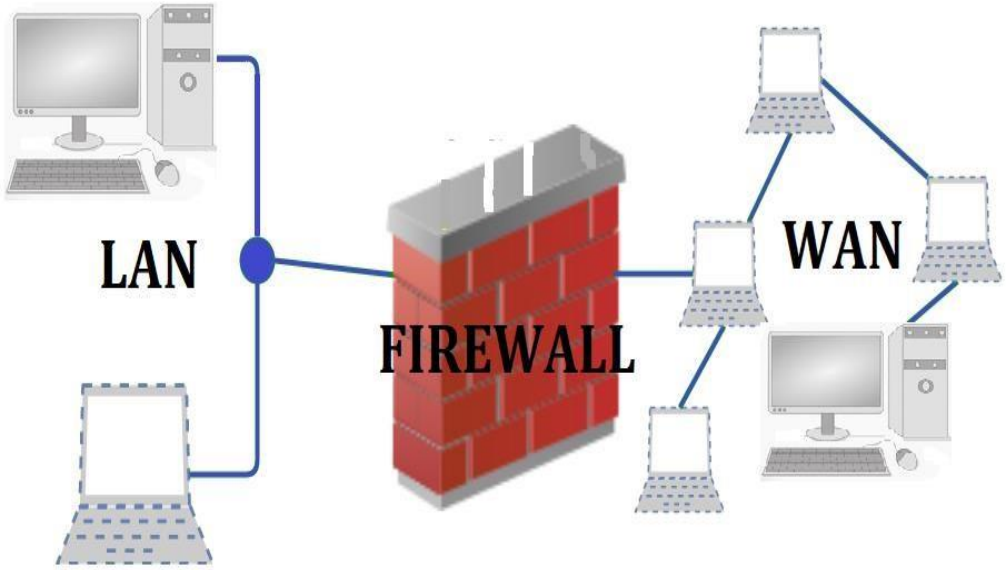
Gambar 2. 11. Cara Kerja *Web Server*.

2.23. Firewall

Pada implementasi keamanan infrastruktur jaringan yaitu dengan menjalankan kebijakan it yang dibuat dan dikonsultasikan oleh vendor yang punya reputasi baik dalam menangani masalah keamanan sistem informasi sebelumnya, sehingga paling tidak dapat meminimalisir dan memperkecil peluang peretas untuk masuk ke dalam jaringan yang ada salah satu kebijakan yang dituangkan dalam mengamankan sebuah infrastruktur jaringan, yaitu :

1. Firewall

   Digunakan untuk mengamankan jaringan dan merupakan sistem baik pada perangkat *software* maupun *hardware* dengan membatasi, menyaring atau bahkan dapat menolak satu atau semua jaringan lokal maupun jaringan yang ada diluar. Pada umumnya *firewall* diterapkan dalam suatu *gateway* (gerbang) antara jaringan lokal dengan jaringan internet, digunakan dan dipakai untuk mengontol atau membatasi akses ke jaringan. *Firewall* bagaikan melihat dan mendengar aktifitas lalu lintas informasi siapa saja yang boleh dan tidak boleh lewat.



Gambar 2. 12 Cara kerja Firewall