BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Dalam penulisan skripsi ini penulis memperoleh referensi tentang pembahasan topik dari penelitian -penelitian sebelumnya. Hal ini dapat di jadikan sebagai bahan perbandingan baik dari sisi kekurangan maupun kelebihan di tampilkan dalam table 2.1 keterkaitan penelitian.

Table 2.1 Keterkaitan Penelitian.

No	Nama Penulis	Tahun	Judul	Hasil
1	Alaurin Attari	2019	Analisis Performansi *High availability* Jaringan pada *Virtual Private LAN Service* Legasi dan Berbasis *Software Defined Network*	*failover delay* dari penerapan SDN dengan kontrol terpusat 44.7% lebih rendah dibandingkan dengan VPLS legasi tanpa SDN dengan kontrol menyebar
2	Rohmat Tulloh	2015	Analisis Performansi *High availability* Jaringan pada *Virtual Private LAN Service* Legasi dan Berbasis *Software Defined Network*	*set-up time* akan bertambah seiring meningkatnya jumlah *host* dan dengan menggunakan protokol *OpenFlow*, *latency* yang terjadi di jaringan dapat dipantau dengan parameter *round trip time* (RTT) yang stabil direntang 0,2 sampai 6 second walaupun jumlah vlan_id dan *background traffic* bertambah

No	Nama Penulis	Tahun	Judul	Hasil
3	Rohmat Tulloh	2017	Simulasi *Virtual Local Area Network* (VLAN) Berbasis *Software Defined Network* (SDN) Menggunakan POX *Controller*	nilai data transfer dan *throughput* yang diperoleh pada VLAN lebih besar hampir 30%. Hasil analisis dari seluruh pengujian penambahan *traffic* terlihat bahwa kinerja VLAN pada SDN akan membebani kinerja jaringan pada VLAN yang berbeda
4	Haris Setyo Sudarmojo	2019	Analisis Performansi VPLS *(Virtual Private LAN Service)* Dengan *VLAN Encapculation* Berbasis SDN *(Software Defined Network)* Menggunakan ONOS *Controller*	*Bandwidth* 10 *Mbps* ketika *throughput* melebihi besar *bandwidth*, maka *throughput* yang dihasilkan akan semakin kecil disisi *destination* sebesar 4831.1 *Kbps*, *delay* yang semakin meningkat sebesar 16.610 *ms*, meningkatkan nilai *jitter* sebesar 3.773 *ms*. *Bandwidth* sudah tidak dapat menampung paket data maka terjadinya *overload* di dalam jaringan tersebut mengakibatkan adanya *packet loss* sebesar 24.94 %. Hasil pengukuran *throughput bandwidth unlimited* ketika diberi penambahan beban trafik semakin meningkat secara perlahan.

Penelitian yang dikaji oleh Alaurin Attari pada tahun 2019 yang berjudul "Analisis Performansi *High availability* Jaringan pada *Virtual Private LAN Service* Legasi dan Berbasis *Software Defined Network*" pada penelitian ini mengunakan ONOS *Controller* untuk simulasi dan menguji *Platform* SDN (*Software Defined Network*) serta menggunakan *OpenFlow* versi 1.0. penelitian ini difokuskan untuk *Availabilitas* jaringan yang baik berhubungan dengan waktu *failover delay* ketika terjadi kegagalan system yang memanfaatkan *OpenFlow* sebagai *control plane* dapat berfungsi dengan baik. Pada penelitian ini di lakukan pengujian *high availability* terhadap jaringan VPLS legasi dan VPLS SDN menggunakan ONOS *controller* untuk melihat pengaruh kontrol terpusat dan kontrol menyebar terhadap *high availability* jaringan VPLS. Jenis kegagalan sistem yang diujikan adalah dengan melakukan pemutusan link dengan parameter pengujian *failover delay*. Berdasarkan hasil pengujian didapatkan kesimpulan bahwa *failover delay* dari penerapan SDN dengan kontrol terpusat 44.7% lebih rendah dibandingkan dengan VPLS legasi tanpa SDN dengan kontrol menyebar. Berdasarkan parameter uji ini, menjadikan VPLS SDN lebih unggul dalam *high availability* (HA).[4]

Penelitian yang dikaji oleh Rohmat Tulloh pada tahun 2015 yang berjudul "Simulasi *Virtual Local Area Network* (VLAN) Berbasis *Software Defined Network* (SDN) Menggunakan POX *Controller*" pada penelitian ini mengunakan POX *Controller* untuk simulasi dan menguji *Platform* SDN (*Software Defined Network*) serta menggunakan *OpenFlow* versi 1.0 untuk memasang *header* VLAN sehingga penelitian ini difokuskan untuk mengevaluasi *performa forwarding* VLAN yang memanfaatkan *OpenFlow* sebagai *control plane* dapat berfungsi dengan baik. Hasil penelitian ini mengusulkan penerapan karakteristik teknologi VLAN dengan skenario penambahan jumlah VLAN ID didapatkan bahwa *set-up time* akan bertambah seiring meningkatnya jumlah *host* dan dengan menggunakan protokol *OpenFlow*, *latency* yang terjadi di jaringan dapat dipantau dengan parameter *round trip time* (RTT) yang stabil direntang 0,2 sampai 6 second walaupun jumlah vlan_id dan *background traffic* bertambah.[5]

Penelitian yang dikaji oleh Rohmat Tulloh pada tahun 2017 yang berjudul "Analisis Performansi VLAN Pada Jaringan *Software Defined Network* (SDN)" Penelitian ini menganalisis performansi VLAN di jaringan SDN. Terjadi penurunan

hampir 60% pada angka jumlah paket yang dapat terkirim (data transfer) dan nilai *throughput* pada sebuah VLAN ID karena terdapat pengaruh dari VLAN ID yang berbeda. Pada pengujian membandingkan VLAN dengan non VLAN *over Netwok Functions Virtualization* (NFV) didapatkan bahwa nilai data transfer dan *throughput* yang diperoleh pada VLAN lebih besar hampir 30%. Hasil analisis dari seluruh pengujian penambahan *traffic* terlihat bahwa kinerja VLAN pada SDN akan membebani kinerja jaringan pada VLAN yang berbeda.[6]

2.2 DASAR TEORI

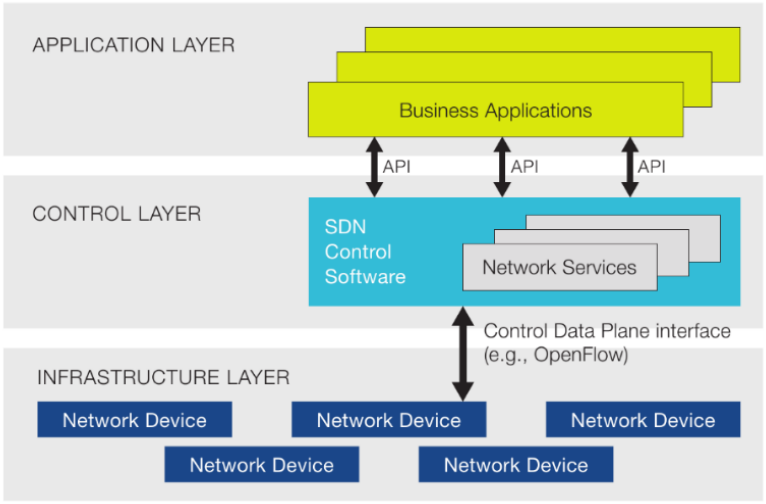
2.2.1 *SOFTWARE DEFINED NETWORK*

Software Defined Network (SDN) adalah sebuah metode baru dalam sebuah struktur jaringan, dalam arsitektur SDN, *control plane* dan data planes dipisahkan, dan infrastruktur jaringan di atur oleh sebuah aplikasi. Sehingga perusahaan dan operator memperoleh program, otomatis, dan mengkontrol jaringannya sendiri. SDN juga merupakan solusi masalah jaringan yang lebih effisien karena dapat disediakan baik secara lokal ataupun terpusat bahkan dapat melalui *cloud,* Jaringan SDN mengkonfigurasi semua perangkat-perangkat dalam jaringan hanya melalui satu media *controller server*.[7]

Tujuan utama dari SDN adalah untuk mencapai pengelolaan jaringan yang lebih baik dengan tingkatan dan kompleksitas yang besar serta memastikan bahwa semua keputusan dari sistem kontrol yang dimana seluruh perangkat dapat diolah melalui satu media *controller*. Dimana *control plan* dapat mengirim perintah ke *data plane* dimana mempermudah dalam konfigurasi *router, switch* dan seluruh perangkat jaringan yang telah terhubung pada sebuah aplikasi. [8]

2.2.2 ARSITEKTUR *SOFTWARE DEFINED NETWORK*

Dalam Arsitektur *Software Defined Network* (SDN) memiliki 3 layer yaitu *application layer*, *control layer*, dan *Infrastructure layer*. Dari ketiga layer tersebut memliki peran atau tugas penting yang berbeda-beda. Pada Gambar 2.1 arsitektur *Software Defined Nework*:



Gambar 2.1 Arsitektur *Software Defined Nework*.[9]

1. *Application layer*

Application layer merupakan *layer* yang berperan sebagai antar muka untuk memudahkan pengelola jaringan dalam melakukan fungsi konfigurasi, fungsi kontrol, dan fungsi evaluasi. Pada lapisan ini terbentuk dari integrasi dengan *control layer* yang memberikan informasi tentang jaringan yang selanjutnya ditampilkan dalam bentuk yang dapat dipahami oleh pengelola jaringan. *Application Programming Interface* (API) untuk menghubungkan *application layer* dan *control layer*, dimana diperlukan bahasa pemrograman tertentu untuk mengkonfigurasi *controller* yang nantinya akan terhubung ke infrastruktur jaringan yang dibuat.[9]

2. *Control layer*

Control layer terdiri dari satu set pengontrol SDN, masing-masing memiliki kontrol eksklusif atas seluruh aktifasi jaringan. Dimana sebagai penerjemah perintah dari *Application layer* di teruskan oleh *Control plane* yang berguna untuk mengontrol jaringan.[10]

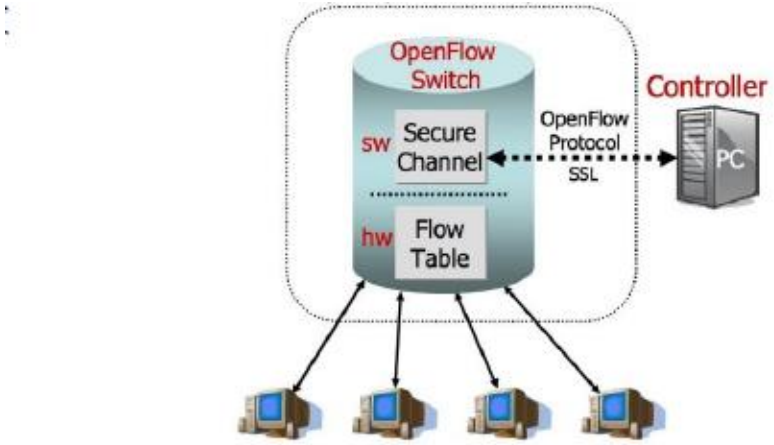
3. *Infrastucture* Layer

Infrastucture Layer berada dibagian lapisan terbawah dari arsitektur *Software Defined Network*. Didalam lapisan tersebut terdapat berbagai perangkat

jaringan yang berfungsi sebagai *data plane*, untuk meneruskan sebuah packet dan menentukan packet mana yang diterima atau dilewatkan. Perangkat jaringan yang biasanya digunakan berupa *router* dan *switch*.[11]

2.2.3 *OPENFLOW*

OpenFlow adalah sebuah *control interface* yang memungkinkan untuk memprogram *switch* pada data plan sehingga administrator dapat mengontrol secara langsung lalu lintas paket pada *forward plan* atau data plan melalui *interface OpenFlow* ini. *OpenFlow* mendefinisikan infrastruktur *flow-based forward*ing dan *Application Programmatic interface* (API) standar yang memungkinkan *controller* untuk mengarahkan fungsi dari *switch* melalui saluran yang aman (*secure chanel*). Dimana *interface* yang memungkinkan terjadinya komunikasi antara *layer controller* dan *data plane* pada arsitektur SDN.[12]



Gambar 2.2 Arsitektur *OpenFlow*.[12]

Pada gambar 2.2 dalam arsitektur *OpenFlow* Cara kerjanya yaitu memungkinkan akses langsung dan memanipulasi *forward*ing plane pada perangkat jaringan seperti *switch* dan *router*. Dalam lingkungan *OpenFlow*, perangkat apapun yang ingin berkomunikasi dengan *controller* SDN harus didukung dengan protokol *OpenFlow*. Melalui *interface* ini, *controller* SDN dapat melakukan perubahan pada *switch*/*router flowtable* yang memungkinkan administrator jaringan untuk memanajemen *traffic*, mengontrol aliran data untuk kinerja yang optimal.[9]

2.2.4 VIRTUAL PRIVATE LAN SERVICE(VPLS)

Layanan *Virtual Private LAN Service* (VPLS) pada aplikasi ONOS yang memungkinkan operator untuk membuat jaringan *overlay broadcast* di Layer 2 sesuai permintaan, di atas infrastruktur *OpenFlow*.[13]

Broadcast traffic dari *single-point* to *multi-point*, penyebaran konektivitas disiaran melalui *single-point to multi-point* di dalam VPLS yang sama, untuk setiap *host* (sumber lalu lintas siaran). Disetiap titik masuk port tempat *host* pengirim terhubung dengan Titik penerimana *port* manapun yang terhubung dengan *host* tujuan syaratnya harus miliki VPLS yang sama.[13]

Aplikasi terhubung ke *host* jaringan *overlay* yang terhubung ke bidang data *OpenFlow*.Menggunakan VPLS dengan VLAN *encapsulation*, pada *ethernet interfaces* dalam mode VLAN dapat memiliki beberapa *ethernet interfaces* logis. Dalam mode VLAN nomor ID VLAN dapat digunakan dari 1 hingga 511 VLAN, tetapi jika menggunakan VPLS nomor ID VLAN dapat di gunakan 512 hingga 4094. Untuk *interfaces Fast Ethernet 4-port*, dapat menggunakan VLAN ID 512 hingga 1024 untuk VPLS VLAN.[14]

2.2.5 VIRTUAL LOCAL AREA NETWORK(VLAN)

Prinsip kerja sebuah jaringan LAN (*Local Area Network*) adalah, semua device yang berada pada satu LAN berarti berada pada satu *broadcast domain*. Sebuah *broadcast domain* mencakup semua *device* yang terhubung pada satu LAN dimana jika salah satu *device* mengirimkan *frame broadcast* maka semua *device* yang lain akan menerima kopi dari *frame* tersebut. Tanpa VLAN, sebuah *switch* akan menganggap semua *interface* (*port*) nya berada pada satu *broadcast domain*, dengan kata lain, semua komputer yang terhubung ke *switch* tersebut akan di anggap berada pada satu LAN yang sama. Dengan menggunakan teknologi VLAN, *switch* bisa mengelompokan beberapa *interface* (*port*) *switch* ke dalam satu *broadcast domain* dan beberapa *interface* yang lain ke dalam *broadcast domain* yang berbeda, sehingga tercipta *multiple broadcast domain*. *Broadcast domain* yang dibuat oleh *switch* inilah yang kita sebut sebagai Virtual LAN.[15]

2.2.5.1 VLAN ID

VLAN ID adalah sebuah identitas yang digunakan untul membedakan antar wilayah VLAN. Identitas (ID) di sini akan dikodekan dalam bentuk nomor. Misalnya VLAN dengan ID 1 (VLAN 1) akan berbeda wilayah dengan VLAN dengan ID 2 (VLAN).[3]

Namun seperti penjelasan diawal, semua *port switch* akan masuk ke dalam wilayah VLAN dengan ID yang sama yaitu VLAN 1. Sehingga tentu saja VLAN dengan ID = 1 (VLAN 1) akan berbeda wilayah *broadcast* dengan VLAN dengan ID = 2 (VLAN 2). Identitas (ID) yang digunakan untuk membedakan wilayah VLAN dibagi menjadi dua bagian; normal dan extended.[3]

Nomor ID yang digunakan untuk membedakan wilayah VLAN terletak pada rentang nomer 1 sampai 1001. Sedangkan nomor dengan rentang ID 1002 sampai 1005 hanya bisa digunakan pada implementasi jaringan *Token Ring* atau FDDI.[3]

2.2.6 ONOS *CONTROLLER*

Open Network Operating System (ONOS) adalah *open source* SDN *operating system* yang pertama kali yang menargetkan secara spesifik kepada *Service Provider* dan *mission critical network*. ONOS dibuat untuk memberikan *high availability, scale-out,* dan *performance* pada jaringan jika dibutuhkan.[16]

Sebagai tambahannya, ONOS telah menciptakan abstraksi *Northbound* dan API untuk memungkinkan pengembangan aplikasi lebih mudah dan abstraksi *Southbound* dan *interface* untuk memungkinkan untuk kontrol *OpenFlow* pada perangkat keras.[16] Gambar 2.3 logo ONOS *controller*.

Gambar 2.3 Logo ONOS *Controller*.[16]

Berikut adalah kelebihan kontroler ONOS:

- Membawa fitur kelas *carrier provider* (skala, ketersediaan, dan performa) untuk pesawat *control* SDN
- Memungkinkan untuk memonitoring dengan *Web-Style*
- Membantu *Service Provider* untuk memigrasikan jaringan yang ada ke jaringan yang baru.

2.2.7 QUALITY OF SERVICE(QoS)

Quality of Service (QoS) merupakan kemampuan suatu jaringan untuk menyediakan layanan yang memadai dengan menyediakan *bandwidth*, mengatasi *jitter* dan *delay*. QoS sangat ditentukan dengan kualitas jaringan yang digunakan. Tujuan dibuatnya QoS yaitu untuk memenuhi banyaknya kebutuhan layanan yang berbeda, yang berada di infrastruktur jaringan yang sama serta untuk membantu *end user* menjadi lebih produktif dan memastikan *user* mendapatkan performansi jaringan yang handal dari berbagai aplikasi berbasis jaringan.[17]

1. ***Throughput***

   Throughput merupakan jumlah total kedatangan bit yang berhasil diamati pada *destination* selama interval waktu tertentu dibagi oleh durasi waktu pengiriman *bit*. Biasanya *throughput* selalu dikaitkan dengan *bandwidth*. Karena *throughput* memang bisa disebut juga dengan *bandwidth*

dalam kondisi yang sebenarnya. *Bandwidth* lebih bersifat *fix*, sementara *throughput* sifatnya adalah dinamis tergantung trafik yang sedang terjadi karena *throughput* kecepatan sebenenarnya di dalam jaringan.[17] Persamaan 2.1 perhitungan untuk mencari nilai *throughput:*

Persamaan perhitungan *throughput*:

$$Throughput = \frac{(Jumlah\ bit\ data\ yang\ diterima\ benar)}{Waktu\ pengiriman\ bit}\ Bps \quad \text{.....(2.1)}$$

2. *Delay*

Delay merupakan durasi waktu transmit yang dibutuhkan paket untuk sampai ke tujuan, panjang pendeknya waktu yang dibutuhkan di pengaruhi beberapa faktor yaitu, adanya antrian pengiriman paket yang panjang, proses pengambilan rute lain dan media fisik atau juga waktu proses yang lama.[17] Pada tabel 2.1 adalah kategori jaringan berdasarkan nilai *delay versi TIPHON.*

Persamaan 2.2 perhitungan untuk mencari nilai *delay*:

$$Delay = \frac{Total\ Delay}{Total\ paket\ yang\ di\ terima} \quad \text{.....(2.2)}$$

Tabel 2.1 Standar *Delay* berdasarkan TIPHON TS 101 329-2.[18]

Kategori *Delay*	Besar *Delay*
Sangat bagus	< 150 ms
Bagus	< 250 ms
Sedang	< 350 ms
Jelek	< 450 ms

3. *Packet Loss*

Packet loss merupakan banyaknya paket yang hilang selama proses transmisi data berjalan. *Packet loss* disebabkan oleh beberapa faktor yaitu disebabkan karena tabrakan (*collision)* atau kemacetan trafik data (*congestion*). pada jaringan paket yang *corrupt* yang menolak untuk *transit*, kesalahan *hardware* jaringan sehingga paket tersebut tidak sampai pada penerima dan jaringan yang terputus sehingga tidak dapat sampai kekomputer

yang dituju.[17] Pada tabel 2.2 adalah kategori jaringan berdasarkan nilai *packet loss versi TIPHON*. Persamaan 2.3 untuk mencari nilai *packet loss*.[17] Persamaan perhitungan *packet loss*:

$$Packet\ loss = \frac{(Packets\ transmitted - Packets\ received)}{Packet\ transmitted}\ x\ 100\ \% \quad \text{.....(2.3)}$$

Tabel 2.2 Standar *Packet loss* berdasarkan TIPHON TS 101 329-2.[18]

Kategori *Packet Loss*	*Packet Loss*
Sangat bagus	0 %
Baik	3 %
Cukup	15 %
Buruk	25 %

4. Jitter

Jitter merupakan variasi *delay* pengiriman paket yang terjadi pada jaringan IP antara *source* dan destination. Besarnya nilai *jitter* yang dihasilkan dipengaruhi oleh variasi beban trafik dan besarnya tumbukan (congestion) antar paket pada jaringan IP.[13]Tabel 2.3 adalah kategori jaringan berdasarkan nilai *jitter versi TIPHON*.

$$Jitter\ = \frac{\text{Variasi}\ Delay}{Total\ paket} \quad \text{.....(2.4)}$$

Variasi delay = d2-d1 + d3-d2 +d4-d3 ….dn.....(2.5)

Tabel 2.3 Standar *Jitter* berdasarkan TIPHON TS 101 329-2.[18]

Kategori *Packet Loss*	*Peak Jitter*
Sangat bagus	0 ms
Baik	75 ms
Cukup	125 ms
Buruk	255 ms

2.2.8 VMWARE

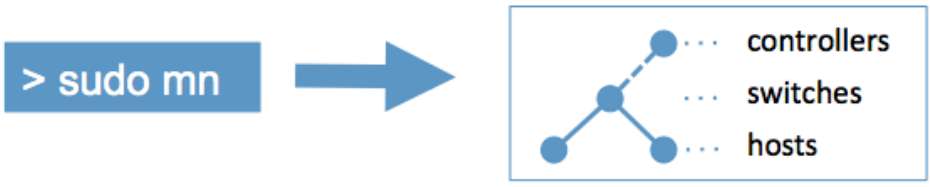
Virtual machine(VM) adalah suatu *environment*, biasanya sebuah program atau system operasi, yang tidak ada secara fisik tetapi dijalankan dalam *environment* lain. Dalam konteks ini, VM disebut "*guest*" sementara *environment* yang menjalankannya disebut "*host*". Ide dasar dari *Virtual machine* adalah mengabtraksi perangkat keras dari satu komputer (CPU, memori, disk, dst) kebeberapa *environment* eksekusi, sehingga menciptakan illusi bahwa masing-masing *environment* menjalankan komputernya [terpisah] sendiri.[19] Gambar 2.4 tampilan apalikasi VMware ketika sudah terinstall.

Gambar 2.4 Tampilan Aplikasi VMware.[20]

VMware memungkinkan bebarapa sistem operasi dijalankan pada satu mesin PC tunggal secara bersamaan. Dengan cara ini maka pengguna dapat memboot suatu sistem operasi (misal *Linux*) sebagai *host operating system* (sistem operasi tuan rumah) dan lalu menjalankan sistem operasi lainnya misal MS *Windows*.[19]

2.2.9 MININET

Mininet merupakan sebuah *emulator* jaringan yang dapat menggambarkan jaringan yang besar dengan menggunakan sebuah PC/laptop.



Gambar 2.5 *Mininet emulator*.[12]

Gambar 2.5 arti dari *Mininet* yang memiliki tujuan menciptakan jaringan virtual yang realistis, menjalankan *real kernel*, *switch* and *application code* pada satu mesin (VM, *cloud* atau asli), dalam hitungan detik, dengan satu perintah. *Mininet* bersifat *open-source*, sehingga proyek yang telah dilakukan berupa *source code*, *scripts*, dan dokumentasi yang dapat dikembangkan oleh siapa saja.[21] *Mininet* sendiri memiliki kelebihan sebagai berikut:

1. *Prototype* jaringan yang telah dibuat dengan menggunakan *emulator Mininet* ini memiliki perilaku jaringan yang sama dengan jaringan yang sebenarnya
2. Karena bersifat *open-source*, *prototype* yang telah dibuat dengan menggunakan *Mininet* ini dapat dikembangkan oleh semua orang secara bebas dan bisa juga dikembangkan bersama-sama.
3. Dengan menggunakan *Mininet*, *prototype* jaringan yang dibuat dapat menggambarkan jaringan dengan jumlah *switch* yang banyak hanya dalam satu perangkat PC/laptop.

Selain memiliki kelebihan, adapula kekurangan yang dimiliki *Mininet* itu sendiri yaitu *Mininet* tidak dapat berjalan pada sistem operasi non-*Linux* yang tidak mendukung *switch OpenFlow*.[21]

2.2.10 WIRESHARK

Wireshark adalah salah satu *Network analysis tool* / *packet sniffer* yang digunakan untuk membantu *Network administrator* dalam mengatasi *troble shooting* serta analisis jaringan. Aplikasi *Wireshark* memungkinkan menangkap informasi dan paket – paket data yang berjalan dalam jaringan. [22]

Aplikasi *Wireshark* adalah salah satu *packet sniffer* yang diprogram sedemikaian rupa agar dapat mengenali berbagai macam protokol dalam jaringan.

2.2.11 PUTTY

PuTTY adalah aplikasi terminal *emulator* untuk klien yang bersifat *open source*. Dengan aplikasi *PuTTY* kita dapat *login* dan meremote *server* melalui protokol SSH, *Telnet* dan *protocol FTP*.[23]

Gambar 2.6 Tampilan Aplikasi *Putty*.[23]

Pada gambar 2.7 tampilan aplikasi *putty* setelah terinstall. Secara keseluruhan, kata *PuTTY* masih belum memiliki penjelasan. Namun kata "*tty*" adalah nama untuk sebuah terminal dalam sistem operasi *Unix*. Aplikasi *PuTTY* dibangun awal tahun 1999, dan digunakan sebagai SSH klien sejak Oktober 2000. Pada awalnya *PuTTY* ditulis untuk berjalan di sistem operasi *Windows*, agar dapat meremote *server Linux* atau *Unix*. Akan tetapi saat ini telah berkembang ke berbagai sistem operasi lain.[23]

2.2.12 XMING

Apilikasi *Xming* adalah aplikasi yang menyediakan fungsi sederhana dari X11 *Forward*ing untuk digunakan pada system operasi *Windows* OS. Biasanya digunakan bersamaan dengan *putty* untuk menampilkan *host* atau *guest* sebuah jaringan. tampilan aplikasi *Xming* jika di jalan, maka akan muncul pada taskbar.

2.2.13 D-ITG

D-ITG *(Distributed Internet Traffic Generator)* adalah *platform* yang mampu menghasilkan lalu lintas IPv4 dan IPv6 dengan secara akurat mereplikasi beban kerja aplikasi Internet saat ini. Pada saat yang sama D-ITG juga merupakan alat pengukuran jaringan yang dapat mengukur metrik kinerja yang paling umum (*Throughput, delay, jitter, packet loss*) di tingkat paket.[24]

Pada lapisan *transport*, D-ITG saat ini mendukung TCP (*Transmission Control Protoco*l), UDP (*User Datagram Protocol*), SCTP1 (*Stream Control Transmission Protocol*), dan DCCP1 (*Datagram Congestion Control Protocol*).

juga mendukung ICMP (*Internet Control Message Protocol*). Di antara beberapa fitur yang dijelaskan di bawah ini, mode pasif seperti FTP juga didukung untuk melakukan eksperimen di hadapan NAT, dan dimungkinkan untuk mengatur bidang *header* TOS (DS) dan TTL IP.[24]

2.2.14 IPERF

iPerf adalah alat untuk pengukuran batas maksimum bandwidth yang dapat dicapai pada jaringan tersebut. mendukung penyetelan berbagai parameter yang terkait dengan waktu, buffer dan protokol (TCP, UDP, SCTP dengan IPv4 dan IPv6).