BAB II
DASAR TEORI

2.1 KAJIAN PUSTAKA

Dalam penelitian ini, kajian pustaka berisikan tentang informasi dari sumber jurnal penelitian yang telah dilakukan. Jurnal yang digunakan adalah jurnal yang relevan sebagai acuan penelitian. Data yang diperoleh pada penelitian sebelumnya dapat digunakan sebagai pembanding penelitian yang akan dilakukan.

Tabel 2. 1 Kajian Pustaka.

Nama	Tahun	Judul	Controller	Metode	Hasil
Mochammad Ridhwan Nurfalah, Ali Akbar Rismayadi	2020	Implementasi Vlan pada Software Defined Network Menggunakan Protokol Openflow	Opendaylight	Melakukan pengujian dengan melakukan ping *host* ke *server* menggunakan *bandwidth* secara berkala setiap beberapa detik akan menambah *bandwidth*, pengujian ini dilakukan selama 30 menit.	Hasil dari *throughput* yaitu 50,552 Kbps, dari *packet loss* yaitu 0%, *delay* yaitu 51,541573 ms dan Hasil dari *jitter* yaitu 51,523267 ms hasil dari keselusurhan nilai dari pengujuian mennutunujukan sesuai

					standar TIPHON.
Rohmat Tulloh	2017	Analisis Performasi VLAN Pada Jaringan *Software Defined Network* (SDN)	Ryu dan Pox	Mentrasnfer data yang dikirmkan ke h1 dengan layanan UDP dan TCP dengan distribusi Weibull, dengan parameter data transfer dan throughput. Pengujian dilakukan dengan cara dilakukan pengamatan terhadap panjang paket karena penambahan paket header 802.1Q. Paket yang dilewatkan adalah TCP, UDP. dilakukan pengamatan terhadap pengaruh dari setting dua VLAN yang berbeda, dilakukan pengamatan terhadap jaringan VLAN dengan non VLAN dengan skenario yang sama seperti pengujian dua, lalu diukur parameter data transfer dan throughput.	Nilai *throughput* dan data transfer yang diperoleh dengan menggunakan VLAN menghasilkan nilai yang lebih besar daripada non VLAN seiring dengan adanya pemberian *traffic*.

Rohmat Tulloh, Ridha Muldina Negara, dan Arif Nur Hidayat	2015	Simulasi Virtual local Area Network (VLAN) Berbasis Software Defined Network (SDN) Menggunakan POX Controller	Pox	SDN dengan menggunakan openflow versi 1.0 untuk memasang *header* VLAN untuk mengetahui performa *forwarding* VLAN yang memanfaatkan openflow sebagai *control plane.* Pada pengujian ini, dilakukan ping dari host 1 menuju host 9 dimana kedua host tersebut tergabung dalam satu domain VLAN ID yang sama. dilakukan 18 kali pengujian. Pengujian konektifitas dilakukan dengan menggunakan protokol ICMP dimana dari setiap host akan mencoba meminta paket menuju host lainnya.	Nilai RTT untuk tiga topologi yang digunakan didapatkan hasil yang stabil berkisar 0,2 – 6 *second*. Jika melihat standard *latency* (VoIP <50ms, *Streaming* < 150ms) memang ini masih tinggi.
Abhimata Zuhra Pramudita dan I Made Suartana	2020	Perbandingan Performa Controller OpenDayLight dan Ryu pada Arsitektur	OpenDayLight dan Ryu	Host mengirimkan data ke host yang lain, diukur QoS dengan simulator D-ITG mensimulasikan sejumlah *switch* dan *host* yang akan	Ryu controller memiliki performa yang lebih baik

		Software Defined Network		mengirimkan data dengan besar *traffic* yang berbeda mulai dari 100Mb,500Mb,1Gb,5Gb hingga 10Gb. Pada pengujian Resource akan dilakukan Utilization simulator Phoronix-Test-Suite mensimulasikan penggunaan hardware Ketika controller berjalan,	berdasarkan paramer QoS dan performa *memory*, pada pengujia, *Resource Utilization* pada performa CPU kedua *controller* menghasilkan kestabilan yang sama baik.

Pada penelitain yang berjudul "IMPLEMENTASI VLAN ADA SOFTWAE DEFINED NETWORK MENGGUNAKAN PROTOKOL OPENFLOW" oleh Mochammad Ridhwan Nurfalah dan Ali Akbar Rismayadi pada tahun 2020. Penelitian ini menganalisis kinerja dari jaringan VLAN berbasis SDN menggunakan Opendaylight controller. Topologi jaringan yang digunakan berupa topologi *linier* dengan 4 *host* dan 1 buah *router* yang terhunbung dengan Openvswtich. Kinerja jaringan ini akan dibuktikan dengan cara melakukan ping seluruh *host* ke *server* dan dilihat juga dari sisi *Quality of Service* (QoS) adalah *Throughput*, *delay, Packet loss* dan *jitter*. Hasil penelitian yang diperoleh nilai rata-rata dari QoS dikategorikan bagus menurut standarisasi TIPHON maka jaringan berbasis *Sofware Defined Network*

menggunakan protokol openflow dapat menggantikan jaringan konvensional yang masih digunakan pada saat ini [1].

Penelitian Oleh Rohmat Tulloh yang berjudul "Analisis Performasi VLAN Pada Jaringan *Software Defined Network* (SDN)" pada tahun 2017, menganalisis performansi konsep jaringan VLAN berbasis SDN berdarkan hasil *transfer rate* dan *throughput* yang diperoleh dengan mengubah *traffic* pada jaringan VLAN, kemudian dibandingkan dengan performa jaringa non VLAN yang dibangun berbasis NFV. Jaringan pada penelitian ini menggunakan 2 buah topologi Ring yang terhubung yang terdiri dari 2 buah *switch* dan 4 buah *host*. Pengujian dalam penelitian ini dengan cara pengamatan terhadap panjang paket karena penambahan paket header 802.1Q. Paket yang dilewatkan adalah TCP, UDP dan ICMP serta pengamatan terhadap pengaruh dari *setting* dua VLAN yang berbeda yaitu VLAN 10 dan 20, lalu ukur parameter data transfer dan *throughput*. Skenario pengujian dilakukan selama 60 detik dengan memberikan traffic yang berbeda dan membandingkan VLAN dengan non VLAN *over Netwok Functions Virtualization* (NFV). Hasil dari pengujian ini 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 [2].

Pada penelitian Rohmat Tulloh “Simulasi Virtual local Area Network (VLAN) Berbasis Software Defined Network (SDN) Menggunakan POX Controller” pada penelitiannya dilakukan simulasi jaringan VLAN yang berbasis SDN dengan menggunakan openflow versi 1.0 untuk memasang *header* VLAN untuk mengetahui performa *forwarding* VLAN yang memanfaatkan openflow sebagai *control plane* dapat berfungsi dengan baik. Penggunaan Arsitektur SDN pada penelitian ini dikarenakan pada jaringan VLAN masih terdapat kendala dalam mengkonfigurasi, untuk mencocokkan fleksibilitas virtualisasi *server*, pengelola jaringan harus mampu untuk secara dinamis menambahkan, *drop* dan mengubah jaringan. Proses ini sulit dilakukan dengan *switch* konvensional, sebab logika kontrol untuk setiap *switch* terletak dalam logika *switching* yang sama. Proses pengelolaan jaringan, tingkat QoS (*Quality of Service*) dan tingkat keamanan dapat sangat memakan waktu jika jaringan

perusahaan besar yang menggunakan perangkat jaringan dari beberapa vendor, sebab administrator jaringan harus mengkonfigurasi peralatan masing-masing vendor secara terpisah dan menyesuaikan kinerja serta parameter parameternya. Pemilihan Pox controller sebagai *controller* dikarenakan Pox controller memiliki banyak fitur khususnya dalam VLAN sehingga memudahkan para peneliti melakukan penelitian mengenai SDN pada jaringan VLAN.

Hasil penelitian ini menunjukan bahwa pengujian konektifitas, verifikasi dan keamanan telah berjalan dengan baik. Untuk pengujian penambahan jumlah VLAN ID didapatkan bahwa *set-up time* akan bertambah seiring meningkatnya jumlah *host*. Dari pengujian yang dilakukan bahwa *setup time* untuk jumlah host yang semakin bertambah maka *setup time* akan semakin besar. Hal ini dikarenakan *controller* menggunakan mode *proactive* sehingga semakin banyak *host* maka *rule* yang dibutuhkan juga semakin banyak sehingga kerja dari controller untuk memberikan informasi kepada *switch* semakin banyak [3].

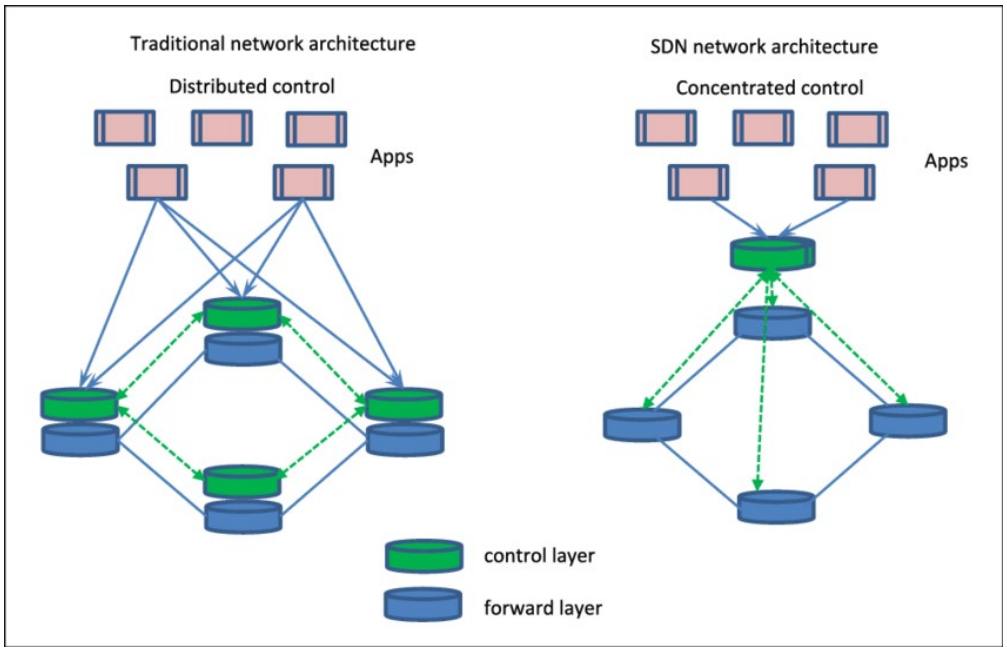
Penelitian Abhimata Zuhra Pramudita dan I Made Suartana di tahun 2020 yang berjudul "Perbandingan Performa Controller OpenDayLight dan Ryu pada Arsitektur Software Defined Network" bertujuan membandingkan performa dari dua *controller* jaringan SDN antara opendaylight dan ryu controller pada beban *traffic* besar berdasarkan parameter *throughput*, *delay* dan *packet loss* serta Pengujian *Resource Utilization* untuk mengetahui kinerja dari *memory* dan CPU. Simulasi perancangan pada penelitian ini menggunakan skema topologi *linier* dengan 8 *switch* dan 160 *host*, dengan 1 *switch* memiliki 20 *host*. Berdasarkan hasil penelitian ini ryu controller memiliki performa lebih baik dibandingkan opendaylight pada parameter *throughput*, *delay* dan *packet loss* dan untuk pengujian *Resource Utilization* pada performa CPU kedua *controller* menghasilkan kestabilan yang sama baik, sedangkan pada performa *memory* ryu menghasilkan kestabilan yang lebih baik dibandingkan opendaylight dikarenakan adanya peningkatan yang signifikan dalam performa *memory* [7].

Dari referensi tersebut [1][2][3][7] Konsep SDN tersebut dapat diaplikasikan pada berbagai mekanisme jaringan komputer, salah satunya adalah pada

konsep *Virtual Local Area Network* (VLAN), oleh karena itu penulis tertarik meneliti penggunaan Pox controller pada jaringan *Virtual local area network* (VLAN).

2.2 DASAR TEORI

2.2.1 *Software Defined Network* (SDN)

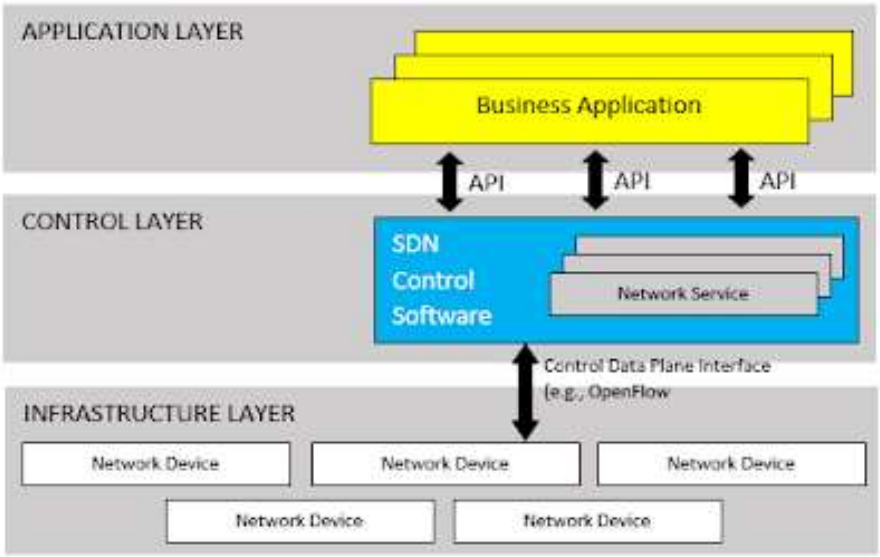


Gambar 2. 1 Perbedaan jaringan konvensional dan SDN [4].

Pada jaringan konvensional yang bersifat tertutup serta terdistribusi merupakan jaringan terdiri dari sejumlah node yang terhubung melalui media transmisi yang berbeda seperti kabel tembaga atau serat optik. Setiap node pada jaringan konvensional terdistribusi bertanggung jawab untuk melakukan tugas *routing* dan *switching* data secara mandiri, tanpa melibatkan komponen jaringan lainnya. Hal ini mengakibatkan konfigurasi jaringan menjadi kompleks dan sulit untuk dikelola, terutama ketika jaringan semakin berkembang. Dengan adanya *Software Defined Network* yang merupakan sebuah konsep pada pendekatan dalam jaringan, menyusun serta mengelola jaringan komputer dengan membuat fungsi kontrol jaringan (*control plane*) dan fungsi *forwarding data* (*data plane*) dipisahkan dengan perangkat kerasnya, menggabungkan semua *control plane* dari setiap perangkat menjadi satu *controller* yang *programmable* agar tersentralisasi sehingga mempermudah administrator untuk mendesain dan

mengoperasikan jaringan [1][8][9]. [4]. SDN juga memiliki kelebihan berupa fleksibilitas yaitu pengaturan jaringan dapat dilakukan secara terpusat, sehingga memudahkan dalam melakukan perubahan konfigurasi jaringan, seperti pengaturan *routing* dan manajemen *bandwidth*, efesiensi yang memungkinkan pengaturan jaringan yang lebih efisien dan optimal, seperti pemanfaatan *bandwidth* yang lebih baik dan penghindaran *congestion* pada jaringan, pengaturan keamanan dapat dilakukan secara terpusat, sehingga memudahkan dalam memonitor dan mengontrol akses jaringan serta memungkinkan pengembangan aplikasi yang memanfaatkan informasi dari jaringan, seperti aplikasi monitoring dan analisis jaringan, dengan mudah [1].

Hal ini yang membuat SDN bersifat *manageable*, dimana jika ada suatu *maintenance* seorang admin tidak perlu untuk mengurus setiap *switch*, cukup dengan konfigurasi pada *controller*-nya saja[10]. Sistem konfigurasi yang terpusat pada perangkat *controller* menjadikan waktu konfigurasi perangkat menjadi lebih cepat. *Controller* bertanggung jawab dalam menentukan bagaimana menangani paket dan mengelola tabel *flow* dengan menghapus dan menambahkan isi tabel *flow* melalui *secure channel*. Perangkat *switch* dalam jaringan SDN akan menerima informasi *rute* berupa tabel *flow* dari *controller*, tabel tersebut akan digunakan oleh perangkat *switch* untuk melakukan fungsi *routing* [11].



Gambar 2. 2 Arsitektur SDN [12].

Arsitektur SDN membagi jaringan menjadi 3 layer yaitu *application layer*, *control layer* dan *infrastructure* (*data layer*). *Application layer* merupakan *interface* terhadap seorang *admin* atau peneliti dalam mengelola atau mengembangkan jaringan SDN. *Control layer* berfungsi sebagai kontrol seluruh aktivitas jaringan, dalam *layer* ini fungsi dari *control plane* bekerja. *Infrastructure layer* merupakan lapisan dasar dari arsitektur SDN, lapisan ini berisi perangkat keras *forwarding plane* yang berfungsi sebagai *data plane*. Perangkat keras ini bisa berupa *switch* atau *router*.

Seperti yang terlihat pada gambar diatas Pada arsitektur SDN, sebagian besar layanan jaringan seperti protokol dan kontrol jaringan terpusat di perangkat lunak SDN. Pada arsitektur SDN terdapat beberapa controller yang dapat digunakan yaitu Floodlight, Maestro, Ryu, Pox, Onos dan beberapa jenis lainnya [13]. SDN *controller* mengelola semua infrastruktur dasar yang merupakan *logical switch virtual*. Operator jaringan dapat mengontrol jaringan melalui *interface* standar misalnya OpenFlow secara independen dari vendor perangkat jaringan. Perangkat pada SDN hanya memiliki fungsi *forwarding* yang dikendalikan oleh pengontrol SDN. Hal ini membuat operasi jaringan menjadi sederhana dan lebih efisien.

Pada Arsitektur SDN terdapat lapisan aplikasi dan lapisan control yang dihubungkan oleh API yang memberikan virtualisasi lingkungan jaringan dan sarana sebagai tempat menerapkan berbagai kebijakan seperti *routing*, *access control*, rekayasa *traffic* dan manajemen daya pada SDN [12][14].

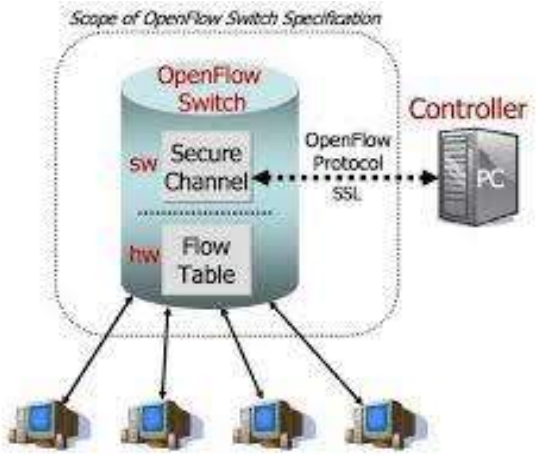
2.2.2 Protokol *Openflow*

Merupakan protokol yang digunakan agar *controller* dan *switch* dapat saling berkomunikasi. Openflow menyediakan protokol terbuka untuk memprogram *flow table* pada *switch* dan membuat administrator jaringan dapat terlibat langsung ke dalam *traffic* data untuk meneliti aliran data, dengan ini administrator mengontrol aliran data mereka sendiri dengan memilih *rute* paket daya yang mereka ikuti dan memproses yang akan diterima oleh *receiver* [1][8].

OpenFlow memungkinkan mengakses langsung dan manipulasi perangkat *forwarding plane* seperti *switch* dan *router*, baik fisik dan virtual =(*hypervisor-based*). OpenFlow diimplementasi dari kedua sisi antarmuka baik dari sisi infrastruktur

jaringan dan kontrol software SDN. Sebuah OpenFlow *switch* terdiri dari *flow table* atau *grup table*. Setiap *flow table* dalam *switch* berisi satu *set flow entri*, yang terdiri dari *header field*, *counter*, dan *set of instructions* atau *actions*[14].

OpenFlow memungkinkan *server* memberitahukan *switch* jaringan tempat mengirim paket. Dalam jaringan konvensional, setiap *switch* memiliki perangkat lunak berpemilik yang memberitahukan apa yang harus dilakukan. Openflow mendefinisikan infrastruktur *flow-based forwarding* dan *Application Programmatic Interface* (API) standar yang memungkinkan *controller* untuk mengarahkan fungsi dari *switch* melalui saluran yang aman (*secure channel*). Bisa dilihat pada Gambar bahwa fungsi openflow sebagai penghubung antara *controller* yaitu termasuk dalam *control plane* dengan *data plane* melewati *secure channel* lalu ke *flow table* dan diteruskan ke *user*.



Gambar 2. 3 Openflow Arsitektur.

OpenFlow awalnya mendukung VLAN di mode yang terbatas. Dengan peningkatan OpenFlow 1.1 ini sekarang memiliki dukungan kuat untuk VLAN, QinQ, dan tag MPLS. Kemampuan ini memberikan fleksibilitas tambahan dalam forwarding plane dengan aturan yang bisa cocok dengan lebih banyak informasi yang terkandung di dalamnya paket Ethernet [15].

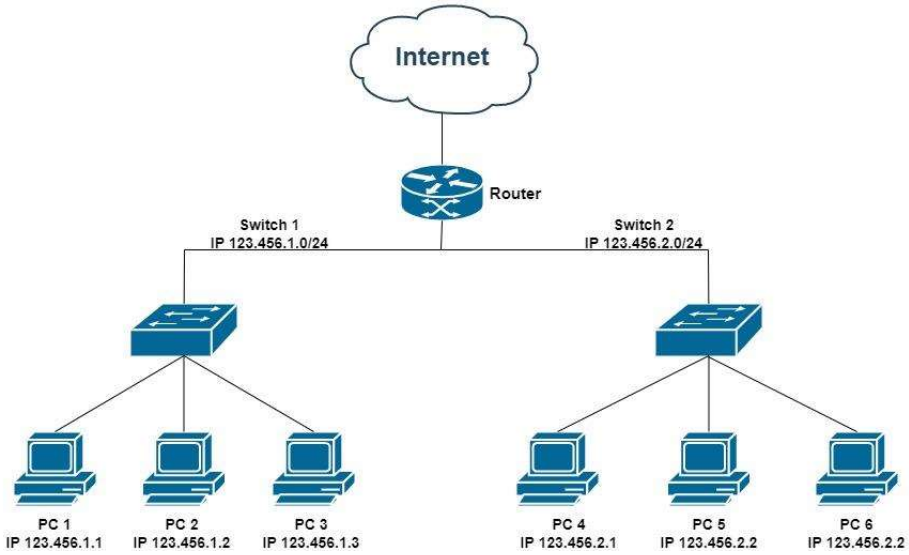
Terdapat beberapa protokol yang digunakan agar controller dan switch dapat saling berkomunikasi (Southbound API) yaitu OpenFlow, gRPC, REST, NETCONF dan SSH/CLI. OpenFlow yang dikembangkan oleh ONF (*Open Networking Foundation*) merupakan protokol yang paling dikenal dalam Southbound API dari

SDN [16][13]. Protokol OpenFlow telah didukung oleh penyedia jaringan seperti Google, Microsoft, Amazon, dan vendor peralatan seperti NEC, Cisco, adalah salah satu mekanisme yang sesuai dengan konsep tersebut dari SDN [17].

2.2.3 Local Area Network (LAN)

LAN adalah singkatan dari Local Area Network adalah sekelompok perangkat jaringan yang memungkinkan komunikasi antara perangkat yang terhubung. Local Area Network (LAN) adalah kumpulan perangkat yang terhubung bersama di satu lokasi fisik, seperti gedung, kantor, atau rumah. LAN bisa kecil atau besar, mulai dari jaringan rumah dengan satu pengguna hingga jaringan perusahaan dengan ribuan pengguna dan perangkat di kantor atau sekolah. Terlepas dari ukurannya, karakteristik penentu tunggal LAN adalah menghubungkan perangkat yang berada dalam satu area terbatas. Sebaliknya, Wide Area Network (WAN) atau Metropolitan Area Network (MAN) mencakup wilayah geografis yang lebih luas. Beberapa WAN dan MAN menghubungkan banyak LAN secara bersamaan.

Keuntungan dari LAN adalah sama dengan kelompok perangkat apa pun yang terhubung bersama. Perangkat dapat menggunakan satu koneksi Internet, berbagi file satu sama lain, mencetak ke printer bersama, dan diakses dan bahkan dikontrol oleh satu sama lain [18].

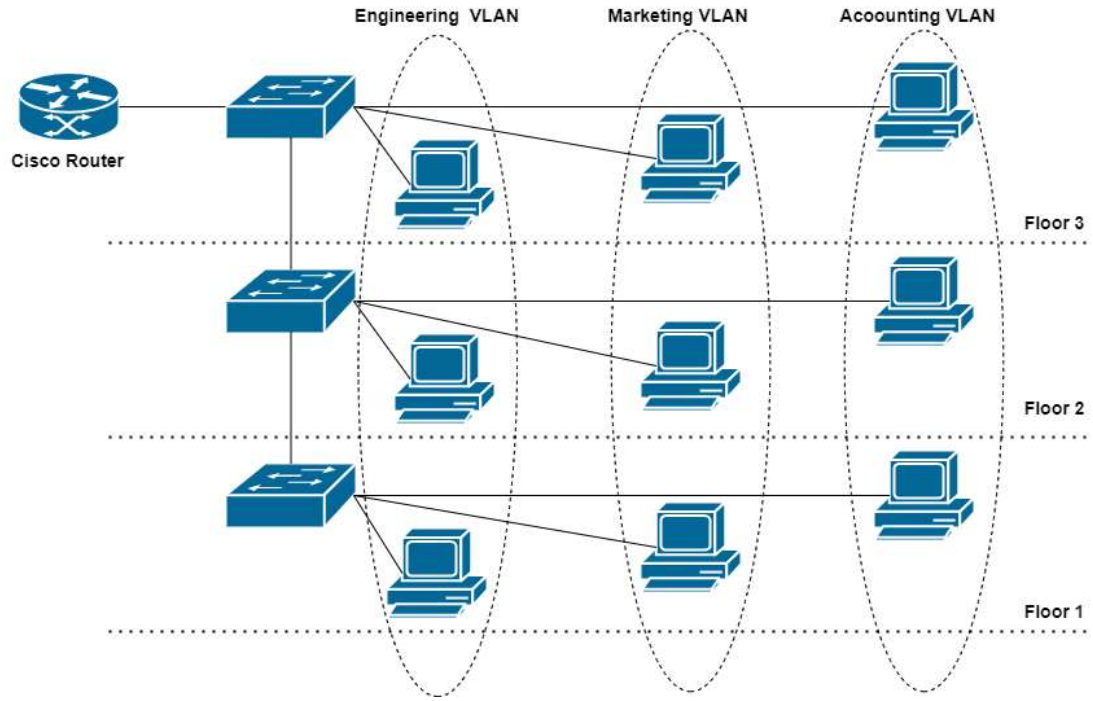


Gambar 2. 4 *Local Area Network* (LAN)

2.2.4 *Jaringan Virtual Local Area Network* (VLAN)

Teknologi VLAN (*Virtual Local Area Network*) bekerja dengan cara melakukan pembagian jaringan secara logik ke dalam beberapa *subnet* VLAN adalah kelompok *device* dalam sebuah LAN yang dikonfigurasi (menggunakan software manajemen) sehingga mereka dapat saling berkomunikasi seakan- akan dihubungkan dengan jaringan yang sama walaupun secara fisikal mereka berada pada segmen LAN yang berbeda. Jadi VLAN dibuat bukan berdasarkan koneksi fisikal namun lebih pada koneksi logikal, yang tentunya lebih fleksibel. Secara logika, VLAN membagi jaringan ke dalam beberapa subnetwork. VLAN mengijinkan banyak subnet dalam jaringan yang menggunakan switch yang sama [1][3].

VLAN biasanya dikaitkan dengan subnetwork IP. Misalnya, semua stasiun akhir dalam subnet IP tertentu milik VLAN yang sama. Untuk berkomunikasi antar VLAN, Anda harus merutekan lalu lintas [19].



Gambar 2. 5 VLAN sebagai Jaringan yang Didefinisikan Secara Logis.

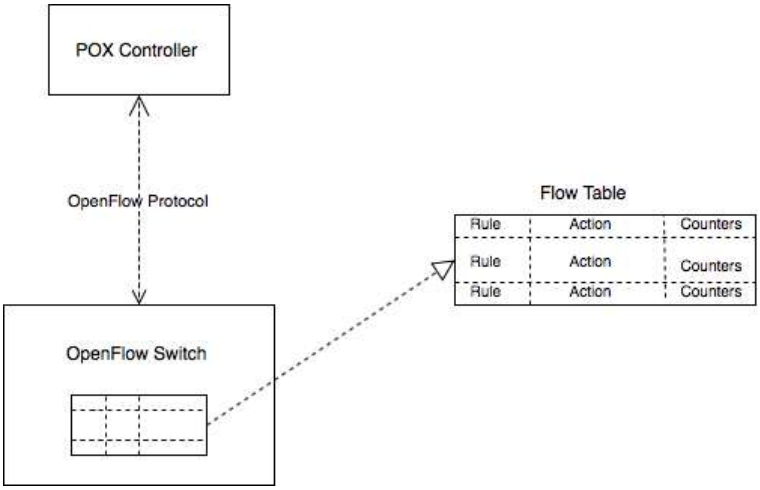
2.2.5 Pox Controller

POX adalah sebuah platform yang digunakan untuk pemodelan dan pengembangan pada SDN, menggunakan bahasa *python* dalam bahasa pemrogramannya. POX controller mempunyai komponen yang dapat digunakan dan di gabungkan untuk membentuk SDN *controller* sesuai kebutuhan pengguna [20].

POX controller menyediakan cara yang efisien untuk mengimplementasikan protokol OpenFlow yang merupakan protokol komunikasi antara *control plane* dan *data plane*. POX dapat menjalankan aplikasi yang berbeda seperti *hub*, *switch*, *load balancer* dan *firewall*. Alat *capture packet Tcpdump* bias digunakan untuk menangkap dan melihat paket yang mengalir di antaranya POX controller dan perangkat OpenFlow [3][21].

POX adalah versi berbasis *python* dari NOX (NOX di *python*). Latar belakang pengembangan POX adalah mengembalikan NOX ke C++dan mengembangkan *platform python* (*python* 2.7) secara terpisah [22].

Berikut penjabaran singkat proses POX Controller:



Gambar 2. 6 Proses pada POX Controller.

Ketika switch on, maka akan otomatis terhubung ke OpenFlow Controller. Awalnya tabel *flow* di seluruh *switch* akan kosong dan ketika paket tiba di *switch*, tabel flow ini tidak tahu apa yang dilakukan kepada paketnya. Maka paket tersebut akan mengirim pesan ke *controller*. Agar paket dapat teratasi, *controller* akan menginput *flow entries* ke tabel *flow* di *switch*[23].

Flow entries yang di tabel *flow* terbagi menjadi tiga bagian yaitu *rule*, *action* dan *counters*. Untuk setiap paket harus dapat melewati *switch* dan *flow entry* ini harus terinstall didalam *switch*, agar dapat melewati *traffic* di jaringan tanpa ada intervensi lebih jauh ke *controller*.

POX dapat dipanggil jika menjalankan pox.py atau debug-pox.py. Dalam menjalankan POX dengan sendirinya tidak akan bisa. Target dari penggunaan POX ini adalah pada para programmer yang ingin mengembangkan strukturnya sendiri. POX memiliki beberapa komponen yaitu:

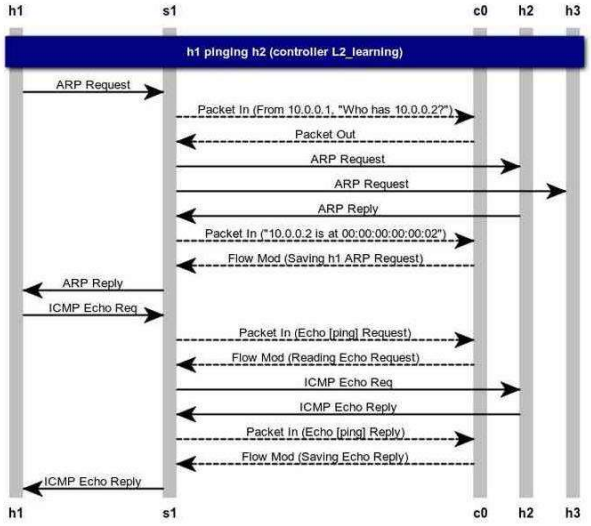
- *forwarding.l2_learning*
- *forwarding.hub*
- *forwarding.l2_pairs*
- *forwarding.l3_learning*
- *forwarding.l2_multi*
- *forwarding.l2_nx*
- *forwarding.topo_proactive*.

Jika kita ambil contoh ***forwarding.l2_learning*** maka komponen ini akan membuat OpenFlow *Switch* beroperasi seperti *switch* di *layer* 2. Untuk menjalankan komponen tersebut cukup mengetikkan (pastikan sudah masuk ke *folder* pox itu sendiri)

./pox.py forwarding.l2_learning

Kita dapat menentukan komponen mana yang mau digunakan, dimana ini disesuaikan dengan *code* yang telah kita buat dan *output* apa yang ingin kita hasilkan. Tidak semua komponen dapat bekerja sama dengan baik tetapi bukan berarti tidak dapat bekerja sama sekali.

POX *controller* memberikan kemudahan pengguna untuk membuat program yang fungsinya yang berbeda-beda, salah satu fungsinya terdapat *built-in functions* yang disediakan POX untuk membuat sebuah fungsi VLAN bekerja, salah satu fungsinya tersebut adalah f.ofp_action_vlan_vid(vlan_vid) yang berfungsi memberikan *header* pada VLAN pada paket yang masuk oleh karena itu penelitian ini berfokus pada POX controller [3].



Gambar 2. 7Proses Pengiriman Data pada Pox.

2.2.6 Mininet 2.2.2

Mininet adalah emulator jaringan yang menciptakan jaringan *virtual host*, *switch*, *controller*, dan *link*. Mininet merupakan emulator jaringan yang bersifat *open source* yang mendukung protokol OpenFlow untuk digunakan mengemulasi *data plane* atau *infrastructure layer* dalam arsitektur SDN. Tampilan pada mininet berupa *Command Language Interpreter* (CLI), hal tersebut memudahkan dalam melakukan skenario jaringan. Mininet merupakan emulator yang sudah teruji untuk mengimplementasikan konsep SDN. Mininet dapat dijalankan dalam sebuah laptop yang menggunakan sistem operasi berbasis Linux. Mininet memiliki kekurangan yaitu mininet tidak dapat berjalan pada sistem operasi non-linux yang tidak mendukung *switch* OpenFlow [3][21]. Mininet dapat mengemulsi konfigurasi jaringan SDN yang terdiri dari *controller*, *switch* dan *host* [24]. Berberapa keunggulan pada mininet:

1. Memulai jaringan sederhana hanya membutuhkan beberapa detik saja. Ini berarti loop run-edit-debug Anda dapat sangat cepat.
2. Dapat membuat topologi kustom: satu switch, topologi Internet yang lebih besar, pusat data, atau apapun.
3. Dapat menjalankan program yang sebenarnya: apa pun yang berjalan di Linux tersedia untuk dijalankan, dari server web hingga alat pemantauan jendela TCP hingga Wireshark.

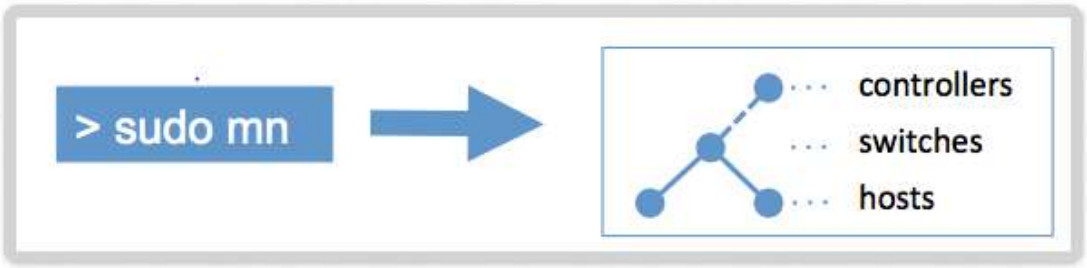
4. Dapat menyesuaikan pengalihan paket: switch Mininet dapat diprogram menggunakan protokol OpenFlow. Desain Jaringan Terdefinisi Perangkat Lunak kustom yang berjalan di Mininet dapat dengan mudah ditransfer ke switch OpenFlow perangkat keras untuk pengalihan paket pada tingkat garis.
5. Dapat menjalankan Mininet di laptop, di server, di VM, pada kotak Linux asli (Mininet disertakan dengan Ubuntu 12.10+!), Atau di cloud (mis. Amazon EC2).
6. Dapat berbagi dan mereplikasi hasil: siapa saja dengan komputer dapat menjalankan kode Anda setelah Anda mempaketkannya.
7. Dapat menggunakannya dengan mudah: dapat membuat dan menjalankan eksperimen Mininet dengan menulis skrip Python sederhana (atau kompleks jika diperlukan) [25].

Penggunaan mininet 2.2.2 dikarena pada penelitian ini OS ubuntu menggunakan python 2. Python 2 masih didukung oleh Mininet 2.3.0, tetapi mungkin menjadi lebih sulit untuk digunakan dari waktu ke waktu karena penghentian resmi proyek CPython serta mengikis dukungan dari distribusi Linux seperti Ubuntu [26]. Mininet mendukung penelitian, pengembangan, pembelajaran, pembuatan prototipe, pengujian, debugging, dan tugas lain apa pun yang dapat diuntungkan dengan memiliki jaringan eksperimental lengkap di laptop atau PC lain.

1. Menyediakan testbed jaringan yang sederhana dan murah untuk mengembangkan aplikasi OpenFlow.
2. Memungkinkan beberapa pengembang bersamaan untuk bekerja secara independen pada topologi yang sama.
3. Mendukung uji regresi tingkat sistem, yang dapat diulang dan dikemas dengan mudah.
4. Mengaktifkan pengujian topologi yang kompleks, tanpa perlu menyambungkan jaringan fisik.
5. Termasuk CLI yang sadar topologi dan sadar OpenFlow, untuk debugging atau menjalankan tes di seluruh jaringan.
6. Mendukung topologi kustom arbitrer, dan mencakup seperangkat topologi parametrized dasar.

7. Dapat digunakan di luar kotak tanpa pemrograman. juga Menyediakan API Python yang mudah dan dapat diperluas untuk pembuatan dan eksperimen jaringan [27].

Pada simulasi jaringan SDN dapat menggunakan beberapa emulator dalam melakukan simulasi jaringan yaitu Mininet, EstiNet, EMANE dan lainya.



Gambar 2. 8 *Single Command Mininet* [3].

2.2.7 *Quality of Service* (QoS)

Kemampuan untuk menjamin kebutuhan jaringan dalam memenuhi SLA antara provider aplikasi dan *end user* [8]. Tujuan utamanya adalah untuk mengetahui nilai yang terdapat pada parameter QoS (*throughput*, *delay*, *jitter* dan *packet loss*) sesuai dengan nilai rekomendasi yang dikeluarkan oleh suatu lembaga seperti ETSI (*European Telecommunications Standards Institute*) yang mengeluarkan standarisasi TIPHON (*Telecommunication and Internet Protocol Harmonization Over Networks*).

2.2.9.1 *Throughput*

Merupakan jumlah bit per detik yang dapat diterima dengan sukses melalui jaringan dari pengirim ke tujuan yang diamati dengan menghitung jumlah paket yang diterima selama interval waktu tertentu dan dibagi dengan waktu pengamatan. Persamaan menunjukan rumus perhitungan *throughput* [1][28].

$$Throughput = \frac{Paket\ yang\ dikirim\ (bit)}{Durasi\ pengiriman\ (s)}$$

2.2.9.2 *Delay*

Total waktu yang diperlukan untuk mengirim paket data dari pengirim ke penerima [1][8][28]**.** Persamaan menunjukan rumus perhitungan *delay*.

$$Delay = \frac{Total\ delay}{Total\ paket\ yang\ diterima}$$

Total delay pada persamaan didapat dari perhitungan menggunakan persamaan.

$$Total\ delay = waktu\ penerimaan\ paket - waktu\ pengiriman\ paket$$

2.2.9.3 *Jitter*

Merupakan variasi *delay* diakibatkan panjangnya antrian (dalam mengolah data) dalam jaringan[1][8]. Persamaan menunjukan rumus perhitungan *jitter*.

$$Jitter = \frac{Total\ variasi\ delay}{Total\ paket\ yang\ diterima}$$

Variasi *delay* di persamaan didapat dari perhitungan menggunakan persamaan.

$$Variasi\ delay\ = delay - (rata - rata\ delay)$$

2.2.9.4 *Packet loss*

Suatu parameter yang menggambarkan suatu kondisi dimana jumlah total paket yang gagal mencapai tujuan, kegagal paket tersebut dapat disebabkan karena *collision* dan *congestion* [1][8][28]. Persamaan menunjukan rumus perhitungan packet loss.

$$Packet\ loss = \frac{(Paket\ data\ dikirim - paket\ data\ yang\ diterima) \times 100\%}{paket\ data\ yang\ dikirim}$$

2.2.8 TIPHON TR 101 329

TIPHON (*Telecommunications and Internet Protocol Harmonization Over Network*). TIPHON merupakan standar penilaian parameter QoS yang dikeluarkan oleh badan standar ETSI (*European Telecommunications Standards Institue*). Standarisasi yang digunakan yaitu TIPHON TR 101 329 [29].

Tabel 3. 1 Standarisasi Nilai *Throughput* TIPHON TR 101 329

No	Kategori	*Througput* (%)
1	Tidak Bagus	100
2	Kurang	75
3	Sedang	50
4	Bagus	<25

Tabel 3. 2 Standarisasi Nilai *Delay* TIPHON TR 101 329

No	Kategori	*Delay* (ms)
1	Baik	<100
2	Sedang	<150
3	Kurang	<400

Tabel 3. 3 Standarisasi Nilai *Jitter* TIPHON TR 101 329

No	**Kategori**	***Jitter* (ms)**
1	Bagus	< 75
2	Sedang	< 125
3	Kurang	< 225

Tabel 3. 4 Standarisasi Nilai *Packet Loss* TIPHON TR 101 329

No	**Kategori**	***Packet Loss* (%)**
1	Sangat Baik	*0 - 2*
2	Bagus	3 - 14
3	Sedang	15 - 25
4	Kurang	> 25