SKRIPSI

ANALISIS KINERJA PROTOKOL *OSPF* DAN *RIPv2* MENGGUNAKAN *ONOS CONTROLLER* PADA JARINGAN *SOFTWARE DEFINED NETWORKS* (*SDN*)

Disusun dan diajukan oleh:

MUHAMMAD ABDUH. MF
D121 17 1505

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2024

LEMBAR PENGESAHAN SKRIPSI

ANALISIS KINERJA PROTOKOL *OSPF* DAN *RIPV2* MENGGUNAKAN *ONOS CONTROLLER* PADA JARINGAN SOFTWARE-DEFINED NETWORKS (*SDN*)

Disusun dan diajukan oleh

MUHAMMAD ABDUH. MF
D121 17 1505

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 01 Agustus 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Eng. Ir. Muhammad Niswar, S.T., M. InfoTech	**Dr. Eng. Zulkifli Tahir, S.T., M. Sc**
NIP. 19730922 199903 1 001	NIP. 19840403 201012 1 004

Ketua Program Studi,

Prof. Dr. Ir. Indrabayu., ST, MT, M. Bus. Sys., IPM, ASEAN. Eng
NIP. 19750716 200212 1 004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini ;

Nama	:	Muhammad Abduh. MF
NIM	:	D121171505
Program Studi	:	Teknik Informatika
Jenjang	:	S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

ANALISIS KINERJA PROTOKOL *OSPF* DAN *RIPV2* MENGGUNAKAN *ONOS CONTROLLER* PADA JARINGAN *SOFTWARE-DEFINED NETWORKS* (*SDN*)

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, Agustus 2024

Yang Menyatakan

Muhammad Abduh. MF

NIM. D121 17 1505

ABSTRAK

MUHAMMAD ABDUH. MF. *Analisis Kinerja Protokol OSPF Dan RIPv2 Menggunakan ONOS Controller Pada Jaringan Software-Defined Networks (SDN)* (dibimbing oleh Muhammad Niswar dan Zulkifli Tahir)

Jaringan *Software Defined Networks* (SDN) menjadi paradigma baru dalam pengelolaan jaringan dengan memisahkan *control plane* dan *data plane*. Hal ini memungkinkan kontrol terpusat dan fleksibilitas yang lebih tinggi dalam mengelola jaringan. Salah satu aspek penting dalam SDN adalah pemilihan protokol routing yang tepat. Penelitian ini membandingkan kinerja protokol routing Open Shortest Path First (OSPF) dan *Routing Information Protocol Version 2* (RIPv2) pada jaringan *SDN* menggunakan *ONOS controller*.

Parameter yang dianalisis meliputi waktu konvergensi, *throughput*, dan *packet loss*. Hasil penelitian menunjukkan bahwa *OSPF* memiliki kinerja yang lebih baik dibandingkan *RIPv2* dalam hal waktu konvergensi dan throughput. Hal ini disebabkan oleh algoritma *Dijkstra* yang digunakan *OSPF* dalam menghitung rute terpendek, serta kemampuannya dalam menangani topologi jaringan yang kompleks. Sedangkan *RIPv2* memiliki kelebihan dalam hal kesederhanaan konfigurasi dan kompatibilitas dengan perangkat jaringan yang lebih luas.

Pengambilan data pada penelitian ini menggunakan bantuan *ONOS Controller*, *Mininet* serta *Quagga Routing Suite* yang diterapkan pada topologi yang sudah dirancang beserta 4 skenarionya yakni, *2 router, 3 router, 5 router, serta 7 router*.

Kesimpulan penelitian ini yakni kedua protokol mampu diimplementasikan pada sebuah topologi virtual menggunakan bantuan *ONOS Controller*, *Mininet* dan *Quagga Routing Suite*. Selain itu hasil analisis terhadap nilai *QoS* pada variabel *Throughput, latency, packet loss*, serta *convergence time* menunjukkan bahwa *OSPF* lebih cocok digunakan pada jaringan SDN yang kompleks dan membutuhkan kinerja routing yang tinggi. Sedangkan RIPv2 dapat digunakan pada jaringan SDN yang lebih kecil dan sederhana.

Kata kunci: *SDN, OSPF, RIPv2, ONOS controller*, waktu konvergensi, *throughput*, *packet loss*.

ABSTRACT

MUHAMMAD ABDUH. MF. *Performance Analysis of OSPF and RIPv2 Protocols Using ONOS Controller in Software Defined Networks (SDN)* (supervised by Muhammad Niswar and Zulkifli Tahir)

Software Defined Networks (SDN) networks are a new paradigm in network management by separating the control plane and the data plane. This allows centralized control and greater flexibility in managing the network. One important aspect in SDN is choosing the right routing protocol. This research compares the performance of the Open Shortest Path First (OSPF) and Routing Information Protocol Version 2 (RIPv2) routing protocols on SDN networks using the ONOS controller.

The parameters analyzed include convergence time, throughput, and packet loss. The results show that OSPF has better performance than RIPv2 in terms of convergence time and throughput. This is due to the Dijkstra algorithm used by OSPF in calculating the shortest route, as well as its ability to handle complex network topologies. Meanwhile, RIPv2 has advantages in terms of simplicity of configuration and compatibility with a wider range of network devices.

Data collection in this research uses the help of ONOS Controller, Mininet and Quagga Routing Suite which is applied to the topology that has been designed along with 4 scenarios, namely, 2 routers, 3 routers, 5 routers and 7 routers.

The conclusion of this research is that both protocols can be implemented in a virtual topology using the ONOS Controller, Mininet and Quagga Routing Suite. Apart from that, the results of the analysis of the QoS values for the throughput, latency, packet loss and convergence time variables show that OSPF is more suitable for use in complex SDN networks and requires high routing performance. Meanwhile, RIPv2 can be used on smaller and simpler SDN networks.

Keywords: SDN, OSPF, RIPv2, ONOS controller, convergence time, throughput, packet loss.

DAFTAR ISI

LEMBAR PENGESAHAN SKRIPSI i
PERNYATAAN KEASLIAN..... ii
ABSTRAK iii
ABSTRACT iv
DAFTAR ISI.....v
DAFTAR GAMBAR vi
DAFTAR TABEL vii
DAFTAR SINGKATAN DAN ARTI SIMBOL viii
DAFTAR LAMPIRAN..... ix
KATA PENGANTAR x
BAB I PENDAHULUAN.....1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan Penelitian 2
1.4 Manfaat Penelitian 2
1.5 Ruang Lingkup.....3
BAB II TINJAUAN PUSTAKA.....4
2.1 *Software Defined Networks*.....4
2.2 *Controller*.....5
2.3 *Mininet* 7
2.4 *OpenFlow*.....8
2.5 Protokol *Routing* 8
2.6 *Quagga Routing Engine*.....13
2.7 Topologi *Tree*.....14
2.8 *Wireshark*.....14
2.9 *Internet Communication Message Protocol* (*ICMP*).....15
2.10 *iperf*.....16
2.11 Analisis Kinerja Jaringan 17
BAB 3 METODE PENELITIAN 21
3.1 Lokasi Penelitian.....21
3.2 Instrumen Penelitian 21
3.3 Prosedur Penelitian 22
3.4 Gambaran Umum Sistem 23
3.5 Perancangan Topologi.....34
3.6 Pengujian Kinerja Jaringan 36
3.7 Tabel Konfigurasi Jaringan.....39
3.8 Tabel *Routing OSPF* dan *RIPv2* 40
BAB 4 HASIL DAN PEMBAHASAN 41
4.1 Hasil Pengujian 41
4.2 Pembahasan.....44
BAB 5 47
KESIMPULAN DAN SARAN.....47
5.1 Kesimpulan 47
5.2 Saran.....47
DAFTAR PUSTAKA 48

DAFTAR GAMBAR

Gambar 2.1 Perbedaan *SDN* dan Jaringan Tradisional (Latah, Majd. 2021) 4
Gambar 2.2 Arsitektur Software Defined Networks (Nikesh e.t al, 2021) 5
Gambar 3.3 Arsitektur *ONOS* (*opennetworking.org*) 7
Gambar 2.4 Topologi *Mininet (mininet.org)*. 7
Gambar 2.5 Letak *openflow* pada arsitektur *Software Defined Networks* 8
Gambar 2.6 Logo Wireshark (https://www.Wireshark.org/) 14
Gambar 3.7 Lokasi Penelitian di Lab UBICON Kampus Teknik Universitas Hasanuddin (https://maps.google.com) 21
Gambar 3.8 Prosedur Penelitian 22
Gambar 3.9 *Mininet* telah berjalan pada sistem 24
Gambar 3.10 *ONOS* telah berhasil dijalankan. 26
Gambar 3.11 ONOS telah berhasil dijalankan. 26
Gambar 3.12 Tampilan setelah berhasil *login* 27
Gambar 3.13 Tampilan awal *VSCodium* 28
Gambar 3.14 *Quagga* berhasil dijalankan 29
Gambar 3.15 ONOS berjalan dengan sukses 30
Gambar 3.16 *Mininet* telah terhubung dengan *controller ONOS* 31
Gambar 3.17 Topologi *mininet* telah muncul pada *ONOS Controller* 32
Gambar 3.18 Aplikasi yang harus aktif pada *ONOS* 33
Gambar 3.19 File konfigurasi *Quagga* 34
Gambar 3.20 Topologi *OSPF* 35
Gambar 3.21 Topologi *RIPv2* 35
Gambar 3.22 Uji *Throughput* pada kedua protokol 37
Gambar 3.23 Uji *latency* kedua protokol 38
Gambar 24. *OSPF* telah terkoneksi 46
Gambar 25. *RIPv2* terkoneksi 46

DAFTAR TABEL

Tabel 1. Kategori Throughput 18
Tabel 2. Kategori *Latency* 19
Tabel 3. Kategori *Packet Loss* 19
Tabel 4. Konfigurasi kedua protokol 39
Tabel 5. Hasil *Throughput* 41
Tabel 6. Persentase *packet loss* kedua protokol 42
Tabel 7. Hasil ping protokol *OSPF* dan *RIPv2* 43
Tabel 8. Hasil convergence time 43
Tabel 9. Perbedaan *OSPF* dan *RIPv2* 44

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
ETSI	*European Telecommunications Standards Institute*
IGP	*Interior Gateway Protocol*
IP	*Internet Protocol*
OSPF	*One Shortest Path First*
QoS	*Quality of Service*
RIPv2	*Routing Information Protocol v2*
SDN	*Software Defined Network*

DAFTAR LAMPIRAN

Lampiran 1 Kode ospf.py 50
Lampiran 2 Kode ripv2.py 56
Lampiran 3 Kode router r1ospf1 – r1ospf7.conf 58
Lampiran 4 Kode router r1ripd.conf – r7ripd.conf 62
Lampiran 5 Kode r1zebra1 – r7zebra7.conf 64

KATA PENGANTAR

Assalamu'alaikum wa rahmatullaahi wa barakatuh.

Alhamdulillah wassholatu wassalamu ala rasulillah, puji syukur senantiasa penulis panjatkan atas kehadirat Allah Subhanahu wa ta'ala yang telah melimpahkan rahmat dan hidayah-Nya, serta shalawat dan salam terkhusus kepada baginda Rasulullah shallallahu 'alaihi wasallam, sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini yang berjudul "ANALISIS KINERJA PROTOKOL *OSPF* DAN *RIPv2* MENGGUNAKAN *ONOS CONTROLLER* PADA JARINGAN *SOFTWARE DEFINED NETWORKS (SDN)*" sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu (S1) pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin. Dalam proses pembuatan laporan akhir ini tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan tugas akhir ini. Oleh karena itu, penulis dari hati yang tulus mengucapkan rasa terima kasih kepada:

1. Allah Subhanahu wa ta'ala yang melalui berkat dan rahmat-Nya, hidayah dan inayah-Nya, sebagai penolong disetiap langkah penulis sehingga dapat menyelesaikan tugas akhir ini.
2. Kedua Orang Tua penulis beserta keluarga, utamanya Ayahanda tercinta Drs. Muhlis, M. Si dan Ibundaku tersayang, Fahmiyah Alwi, S. Si, M. PKim yang selalu memberikan dukungan, motivasi, dan semangat serta selalu sabar menunggu ku hingga sarjana, tidak henti-hentinya mendoakan, merawat, menyayangi, terima kasih yang tak terhingga, dan tentunya maafkan anakmu ini yang terlambat menyelesaikan studinya.
3. Bapak Dr- Eng. Ir. Muhammad Niswar, S.T., M.InfoTech., selaku pembimbing I dan bapak Dr. Eng. Zulkifli Tahir S.T., M. Sc., selaku pembimbing II yang senantiasa menyediakan waktu, tenaga, pikiran dan memberikan bimbingan dalam penyusunan tugas akhir ini.
4. Bapak Adnan, S.T, M.T, Ph. D, dan bapak Iqra Aswad, S.T, M.T selaku penguji I dan II yang telah memberikan saran terkait tugas akhir ini.

5. Bapak Prof. Dr. Ir. Indrabayu., ST, MT, M. Bus. Sys., IPM, ASEAN. Eng selaku ketua Departemen Teknik Informatika Universitas Hasanuddin
6. Segenap Dosen dan Staff Departemen Teknik Informatika Fakultas Teknik Univesitas Hasanuddin yang telah banyak membantu dan memberikan banyak ilmu selama masa perkuliahan.
7. Adik-adikku tercinta, Sitti Nur Aisyah Muhlis, S. Tr. Gz, dan Ahmad Muflih Syawal yang senantiasa memberi semangat kepada penulis untuk segera menyelesaikan studinya.
8. Seluruh Anak Teknik baik Senior maupun Junior dan terkhusus teman Angkatan 2017 yang telah menemani suka duka dalam menjalani dunia kemahasiswaan dan perkuliahan dikampus kita tercinta ini.
9. Teman-teman RECOGN17ER atas dukungan dan semangat yang telah diberikan, ingat selalu Aku, Kamu, dan Kalian.
10. Keluarga besar SMA Negeri 17 Makassar yang telah menjadi tempat mengabdi sejak menjadi siswa, yang senantiasa mendukung dan mendorong penulis untuk segera menyelesaikan studinya.
11. Orang-orang berpengaruh lainnya yang tidak sempat penulis sebutkan satu persatu yang telah memberikan dukungan, semangat dan doanya selama penyusunan tugas akhir ini.

Akhirnya dengan segala kerendahan hati, penulis menyadari bahwa masih terdapat kekurangan dalam penyusunan tugas akhir ini baik dari segi isi maupun cara penyajian. Oleh karena itu, penulis berharap adanya saran dan kritik yang bersifat membangun demi kesempurnaan tugas akhir ini. Penulis berharap semoga tugas akhir ini dapat memberikan manfaat bagi pembaca pada umumnya dan manfaat bagi penulis sendiri khususnya.

Wassalamu’alaikum wa rahmatullaahi wa barakatuh.

Makassar, Juni 2024

Muhammad Abduh. MF

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi terutama di bidang jaringan semakin pesat seiring banyaknya inovasi yang ditemukan oleh para peneliti dan insinyur di bidang jaringan komputer. Salah satu inovasi yang berkaitan dengan pengembangan jaringan yakni *Software Defined Networks* (*SDN*). *SDN* bertujuan sebagai sebuah teknologi modern yang dapat membuat sebuah jaringan dikendalikan oleh satu pusat saja sesuai dengan kebutuhan yang diinginkan, atau dengan kata lain jaringan ini menjadi lebih fleksibel dan dinamis. Konsep dasar dari *SDN* ini yakni memisahkan antara *data plane* dan *control plane* dari perangkat utama jaringan yang kita gunakan. Lapisan data plane ini hanya melakukan penerusan paket data dan satu-satunya bagian yang terletak pada perangkat inti jaringan berdasarkan instruksi *control plane*, sedangkan control plane berada di atas data plane dan bertanggung jawab untuk mengatur lalu lintas paket pada jaringan melalui sebuah *controller SDN* (Bholebawa IZ, Dalal UD. 2018).

Dalam arsitektur yang terdapat pada *SDN*, ada protokol-protokol tertentu yang mendukung konsep penggunaan *SDN*, salah satunya adalah *Openflow*. Openflow merupakan salah satu protokol komunikasi antara data plane dan control plane pada arsitektur *SDN* (Afandi, dkk. 2018). Untuk memudahkan penentuan rute jalur pada jaringan, kita dapat menggunakan protokol *OSPF* dan *RIPv2*. Kedua protokol ini termasuk kedalam jenis *Interior Gateway Protocol* (*IGP*) yang berfungsi untuk mendistribusikan informasi *routing* keseluruh jaringan yang saling terhubung. Artinya protokol *OSPF* ini hanya dapat bekerja dalam jaringan internal suatu organisasi atau perusahaan. *OSPF* (*Open Shortest Path First)* merupakan sebuah protokol *routing dynamic* dan berjenis *link state*. Adapun protokol *RIPv2* (*Routing Information Protocol*) termasuk dalam kategori *distance vector*, dimana protokol ini digunakan untuk menentukan jalur berdasarkan jumlah *hop* untuk mencapai koneksi tujuan (Supriadi, D. dkk. 2019) Dengan kata lain, kedua protokol tersebut merupakan protokol *routing* dinamis yang sama-sama berfungsi untuk mendapatkan jalur terbaik yang dapat dilalui oleh proses *networking*. Selain itu, kedua protokol tersebut memiliki peran penting khususnya pada jaringan skala

besar guna pencatatan alamat pada jaringan. Jika seluruh alamat sudah tercatat pada jaringan, hal ini disebut sebagai kondisi konvergen.

Untuk mengatur logika pada jaringan *SDN*, diperlukan sebuah *controller* khusus, Dari banyaknya *controller* pada sistem *SDN*, salah satu yang dapat digunakan yakni *controller ONOS* atau *Open Network Operating System*. *Controller* ini memiliki kelebihan yakni *high-avability*, *scalable*, dan performa sekelas AT&T pada jaringan komputer. *ONOS* juga memiliki tampilan web *responsive* yang mempermudah proses *monitoring* jaringan (Ramadhan, F. dkk. 2018)

Berdasarkan latar belakang tersebut, penulis mengangkat judul penelitian "Analisis Kinerja Protokol *OSPF* dan *RIPv2* menggunakan *ONOS Controller* pada Jaringan *Software-Defined Networks* (*SDN*)" untuk mengetahui sejauh mana kinerja dan pengimplementasian protokol *OSPF* dan *RIPv2* pada *controller ONOS* dengan menggunakan topologi *tree* berdasarkan standar *Quality of Service* (*QOS*) yang keluarkan oleh *European Telecommunications Standards Institute* (ETSI).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan pada poin sebelumnya, maka rumusan masalah pada penelitian ini antara lain :

1. Bagaimana pengimplementasian topologi protokol *OSPF* dan *RIPv2* menggunakan *ONOS Controller* pada *Software Defined Networks*?
2. Bagaimana hasil analisis kinerja protokol *OSPF* dan *RIPv2* menggunakan *ONOS Controller* pada *Software Defined Networks*?

1.3 Tujuan Penelitian

Tujuan akhir dari penelitian ini yakni :

1. Mengimplementasikan protokol *OSPF* dan *RIPv2* menggunakan *ONOS Controller* pada *Software Defined Networks*.
2. Menganalisis kinerja protokol *OSPF* dan *RIPv2* menggunakan *ONOS Controller* pada *Software Defined Networks*.

1.4 Manfaat Penelitian

Manfaat yang akan didapatkan pada penelitian ini yakni :

1) Bagi peneliti, diharapkan hasil penelitian ini menjadi referensi pengembangan *Software Defined Networks* bagi peneliti selanjutnya.

2) Bagi institusi, diharapkan hasil penelitian ini menjadi referensi bacaan yang berkaitan dengan *Software Defined Networks*.

1.5 Ruang Lingkup

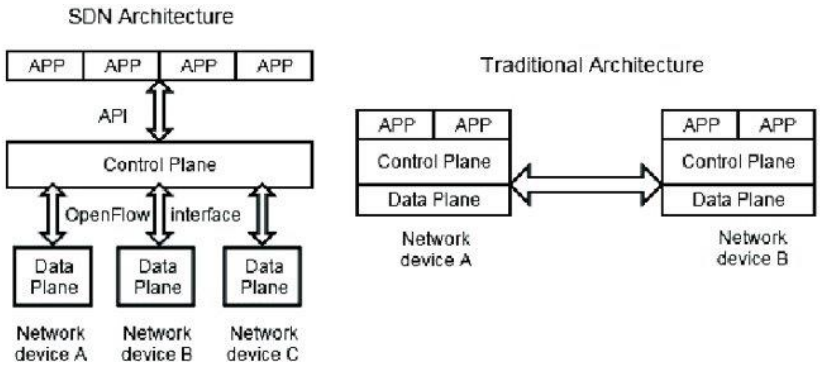
Berdasarkan beberapa hal yang dicantumkan pada rumusan masalah, maka ruang lingkup dari penelitian ini yakni :

1. Emulasi jaringan *Software Defined Networks* Menggunakan emulator Mininet dan *Controller ONOS* yang di install pada distro Ubuntu 20.04.6
2. Menggunakan protokol *OSPF* dan *RIPv2* pada router virtual dengan bantuan *Quagga Routing Engine.*
3. Kedua protokol memiliki perbedaan dari segi algoritma dan skala penggunaan pada jaringan sehingga yang diuji hanya performansi masing-masing, bukan perbandingan keduanya.
4. Parameter yang akan diuji berdasarkan *QoS* (*Quality of Service*) yang dikeluarkan oleh *ETSI* (*European Telecommunications Standards Institute*) sebagai berikut : *Throughput, Latency, Packet Loss,* dan *Convergence Time.*

BAB II
TINJAUAN PUSTAKA

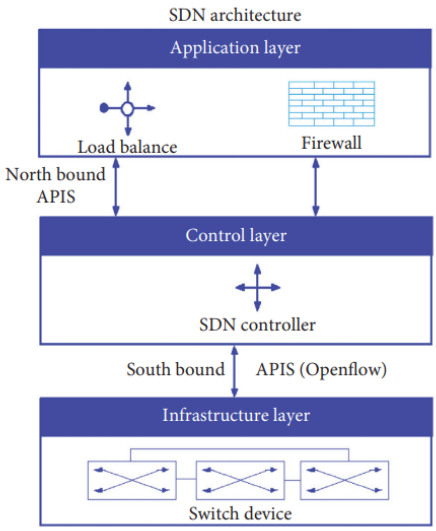
2.1 *Software Defined Networks*

Software Defined Networks (*SDN*) telah menjadi teknologi yang semakin populer untuk manajemen dan kontrol jaringan, karena fleksibilitas, skalabilitas, dan kemampuan untuk menyesuaikan pemrograman didalamnya. Konsep dasar di balik *SDN* yakni memisahkan bidang kontrol *(control plane)* dan bidang data jaringan *(data plane). Control plane* berfungsi sebagai pengelola serta untuk konfigurasi jaringan, sedangkan *data plane* bertugas untuk meneruskan dan merutekan paket data. Pada *control plane* biasanya terjadi pengimplementasian dengan menggunakan bantuan perangkat lunak yang dalam beberapa literatur disebut sebagai *controller*. *Controller* inilah yang berkomunikasi secara langsung dengan perangkat-perangkat jaringan yang saling terhubung satu sama lain. Hal ini membuat sebuah struktur jaringan lebih mudah untuk dikonfigurasi menggunakan bantuan perangkat lunak, tanpa perlu melakukan konfigurasi satu persatu terhadap alat-alat jaringan tersebut.



Gambar 2.1 Perbedaan *SDN* dan Jaringan Tradisional (Latah, Majd. 2021)

Pendekatan ini memungkinkan para *administrator* jaringan untuk secara dinamis mengalokasikan dan mengelola sumber daya jaringan, dan dengan sigap terkait perubahan pada kondisi jaringan. Sejak diperkenalkannya *SDN*, banyak penelitian telah berfokus pada kelebihan dan kekurangannya. Menurut Chen dkk. (2019), *SDN* menawarkan fleksibilitas, skalabilitas, dan kontrol jaringan yang lebih besar dibandingkan dengan arsitektur jaringan tradisional.



Gambar 2.2 Arsitektur Software Defined Networks (Nikesh e.t al, 2021)

Pada arsitektur jaringan tradisional, semua lapisan baik itu *application layer*, *control plane*, dan *data plane*, berada dalam satu alat sehingga ketika ada permasalahan, para *administrator* harus mengecek masing-masing alat untuk mengetahui bagian yang *error* ketika ada masalah pada jaringan. Berbeda dengan arsitektur pada *SDN*, masing-masing layer terpisah sehingga ketika ada masalah, kita hanya perlu mengecek melalui *layer control plane* menggunakan bantuan *controller*.

2.2 *Controller*

Controller merupakan perangkat lunak pengendali pada *SDN* yang mengelola openflow. Ada 2 tipe dari *controller* yaitu:

1. *Controller* statis, dapat berupa perangkat yang dapat menambahkan atau menghilangkan flow dari flow table secara statis.
2. *Controller* dinamis, secara dinamis memanipulasi isi dari flow sehingga cocok untuk beberapa konfigurasi. Tanggung jawab *controller* menurut Vishnoi, A., dkk, (2013) adalah :

a) Menyediakan mekanisme untuk koneksi dan interaksi dengan *Openflow switch*

b) Menginterpretasikan pesan yang dikirim oleh *switch Openflow*.

c) Menyediakan semua instruksi pada setiap spesifikasi (baik spesifikasi yang dibutuhkan, tambahan atau keduanya) untuk dapat diprogram ke dalam *switch*.

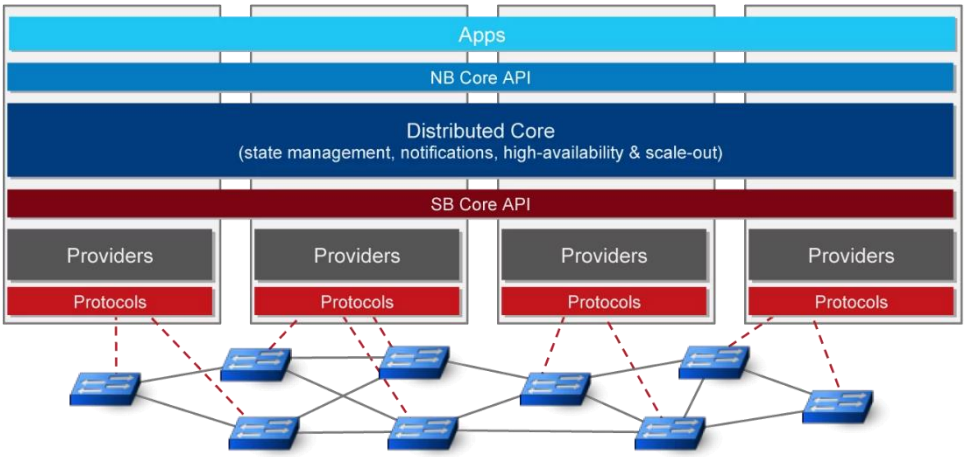
d) Memberikan mekanisme kepada *switch* untuk mendapatkan informasi/pendapat secara umum hingga ke yang detail dan harus dapat menginterpretasikan respon yang tepat.

2.2.1 *ONOS Controller*

ONOS (*Open Network Operating System*) adalah sebuah system operasi pada lingkungan *SDN* yang menyediakan sebuah *platform* untuk mengimplementasikan secara langsung pengendali atau *controller* yang terdistribusi di *SDN* untuk penyediaan layanan komunikasi, dan dirancang sedemikian rupa sehingga memiliki *scability, high performance*, dan *high avaibility. ONOS* termasuk kedalam kontroler yang bersifat *open source* yang berorientasi pada jaringan operator. *ONOS* menggunakan bahasa pemrograman java kemudian setiap fitur yang berada pada kontroler *ONOS* teraktivasi menggunakan Apache (Wahyudi, Eka d.k.k. 2022). Adapun fitur dari *controller ONOS* yang dilansir dari situs resminya (*https://opennetworking.org/ONOS/)* sebagai berikut :

1. Penyedia Layanan memerlukan ketersediaan tinggi agar pelanggan tidak mengalami downtime jaringan. *ONOS* dirancang sejak awal untuk mendukung jaringan operator yang paling menuntut dan memiliki banyak mekanisme untuk memastikan jaringan dan koneksinya dapat diandalkan.
2. Perangkat lunak yang lebih mudah dibaca, diuji, dan dirawat. Yang paling penting, ini memungkinkan mitra untuk lebih mudah menyesuaikan perangkat lunak.
3. *ONOS* menyediakan kemudahan pemrograman jaringan terotomatisasi dan kontrol inovatif yang membantu untuk menyederhanakan pembuatan, penyebaran, dan pengoperasian konfigurasi, manajemen, dan pengontrolan terhadap perangkat yang terhubung.
4. Mudah untuk adaptasi ke perangkat *legacy* (lama) atau perangkat baru (Arsitektur *Plug-in*) *ONOS* mengabstraksi karakteristik perangkat sehingga sistem operasi inti tidak harus mengetahui protokol tertentu yang digunakan untuk mengontrol atau mengkonfigurasi perangkat.
5. Kerangka GUI & UI Dasar, *ONOS*® GUI memberikan tampilan jaringan multi-layer dan memungkinkan eksplorasi elemen-elemen

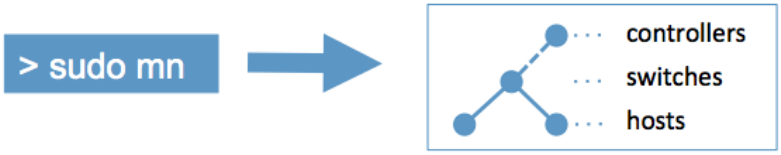
jaringan, konektivitas, keadaan jaringan, kesalahan jaringan, dan lainnya.



Gambar 3.3 Arsitektur *ONOS* (*opennetworking.org*)

2.3 *Mininet*

Mininet adalah sebuah emulator jaringan yang dikembangkan oleh beberapa kontributor, salah satunya adalah *Bob Lantz*. Mininet mempunyai fungsi sebagai emulator jaringan yang membuat jaringan host virtual, switch, *controller*, dan link. *Host Mininet* berjalan pada perangkat lunak jaringan linux standar, dan switch-nya memiliki kemampuan yang mendukung *OpenFlow* untuk *routing* yang secara fleksibel dan tentunya *SDN* itu sendiri (*Mininet Overview*, 2022).



Gambar 2.4 Topologi *Mininet (mininet.org)*.

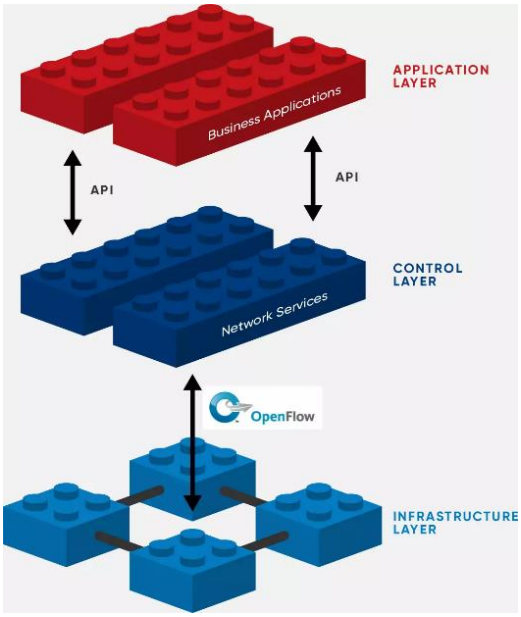
Dilansir dari situs resmi *mininet.org*, mininet mampu menjalankan sebuah topologi yang berisikan *controllers*, *switches*, dan *hosts*. Adapun beberapa topologi dasar dari mininet yang sering digunakan yaitu :

- *Single* : Topologi yang memiliki satu *switch* dan jumlah *hostnya* dapat disesuaikan dengan kebutuhan
- *Tree* : Topologi yang sesuai dengan namanya yakni berbentuk seperti pohon dan diawali dengan satu *switch* lalu bercabang dua dan seterusnya sesuai dengan keinginan dan perintah yang dimasukkan kedalam sistem.

- *Linear* : Topologi yang berbentuk seperti garis dimana setiap switchnya memiliki satu host dan *switch* terhubung seperti *line*.

2.4 *OpenFlow*

OpenFlow adalah protokol komunikasi terbuka yang memungkinkan sebuah switch atau router jaringan untuk mengontrol aliran paket jaringan secara dinamis (McKeown dkk. 2008.) *OpenFlow* memungkinkan jaringan yang didefinisikan oleh perangkat lunak (*SDN*) dengan memisahkan lapisan kontrol (*control plane*) dari lapisan pengalihan (*forwarding plane*) dalam perangkat jaringan.



Gambar 2.5 Letak *openflow* pada arsitektur *Software Defined Networks*

2.5 Protokol *Routing*

Protokol *routing* adalah komponen kunci dalam jaringan komputer yang bertanggung jawab untuk mengarahkan alur lalu lintas data dari satu titik ke titik lain dalam jaringan. Hal ini memungkinkan jaringan untuk menentukan rute terbaik guna pengiriman paket data dari sumber ke tujuan, dan berperan penting dalam mengoptimalkan kinerja jaringan serta menjaga kualitas dari koneksi. Protokol *routing* beroperasi berdasarkan prinsip-prinsip dasar yang meliputi :

- *Routing* : Ini adalah proses pengiriman paket data dari sumber ke tujuan melalui serangkaian perangkat jaringan yang disebut router. Router adalah perangkat yang memutuskan rute terbaik untuk paket data berdasarkan informasi dalam tabel routing.

- Tabel *Routing*: Setiap router dalam jaringan memiliki tabel routing yang berisi informasi tentang rute-rute yang tersedia dan metrik (nilai yang menunjukkan kualitas rute) yang berkaitan dengan setiap rute.
- Metrik: Metrik digunakan untuk menentukan kualitas rute. Rute dengan metrik yang lebih rendah dianggap lebih baik. Metrik dapat berbeda tergantung pada protokol perutean yang digunakan.

Protokol *routing* dapat diklasifikasikan menjadi dua kategori utama yakni protokol perutean statis dan protokol perutean dinamis. Protokol perutean statis (*static routing)* yakni admin jaringan secara manual mengkonfigurasi tabel *routing* pada setiap *router*. Hal ini dapat digunakan untuk jaringan kecil dengan topologi sederhana. Sedangkan protokol perutean dinamis (*dynamic routing)* merupakan protokol yang memungkinkan router untuk secara otomatis bertukar informasi perutean dan memperbaharui tabel *routing*. Contoh protokol perutean dinamis termasuk *RIP*, *OSPF*, *EIGRP*, dan *BGP*.

2.5.1 One Shortest Path First (*OSPF*)

Dalam jaringan komputer, *OSPF* (*Open Shortest Path First*) adalah protokol routing yang banyak digunakan untuk menemukan jalur terpendek antara dua titik. *OSPF* beroperasi sebagai protokol status tautan, di mana setiap perangkat yang menjalankan *OSPF* mempertahankan *database* jaringan yang diketahui beserta biaya yang terkait. Dengan pertukaran iklan status tautan (LSA), *router OSPF* membangun peta topologi komprehensif dari jaringan. Peta ini memungkinkan router *OSPF* untuk secara dinamis menghitung jalur terpendek ke jaringan tujuan berdasarkan biaya tautan di sepanjang jalur tersebut. Akibatnya, *OSPF* dapat beradaptasi terhadap perubahan topologi jaringan dan menyediakan routing yang efisien.

Salah satu keunggulan *OSPF* adalah kemampuannya untuk mendukung jaringan besar dengan banyak router. Dengan mendistribusikan informasi tentang topologi jaringan dan biaya tautan, *OSPF* memastikan setiap router memiliki pandangan penuh tentang jaringan. Pengetahuan jaringan yang komprehensif ini memungkinkan *router OSPF* membuat keputusan *routing* yang terinformasi dan menghindari perulangan routing yang dapat menyebabkan kepadatan jaringan dan perlambatan transmisi data. Selain itu, *OSPF* menawarkan dukungan untuk jalur ganda ke jaringan

tujuan. Saat menghitung jalur terpendek berdasarkan biaya tautan, *router OSPF* juga dapat menentukan dan menggunakan jalur alternatif ke jaringan yang sama. Fitur ini memfasilitasi penyeimbangan beban lalu lintas di sepanjang jalur ganda, meningkatkan kinerja jaringan secara keseluruhan dan mengurangi kepadatan.

Beberapa peneliti telah mengusulkan arsitektur dan algoritma baru untuk meningkatkan kinerja dan efisiensi *OSPF*. Selain itu, terdapat minat yang meningkat dalam memanfaatkan teknik pembelajaran mesin untuk meningkatkan keamanan dan keandalan *OSPF* dalam lingkungan Software Defined Networking (*SDN*) (Wu dkk., 2020). Adapun Prinsip kerja OSPF berdasarkan Algoritma *Dijkstra* sebagai berikut :

- Pengumpulan Informasi Topologi:

OSPF menggunakan algoritma Dijkstra untuk menemukan jalur terpendek ke setiap tujuan dalam jaringan. Setiap router OSPF mengumpulkan informasi topologi dari seluruh jaringan melalui pertukaran Link State Advertisements (LSAs) dengan router tetangga.

- Pembangunan Database Link State:

Informasi yang diterima dari LSAs digunakan untuk membangun database link state yang mencerminkan topologi jaringan secara lengkap. Setiap router memiliki salinan lengkap dari database ini .

- Algoritma Dijkstra:

Router menjalankan algoritma Dijkstra pada database link state untuk menghitung jalur terpendek dari dirinya ke semua router lain dalam jaringan. Algoritma ini bekerja dengan langkah-langkah berikut:

Inisialisasi: Tetapkan jarak ke node awal (router sendiri) sebagai 0 dan jarak ke semua node lain sebagai tak terhingga. Tetapkan node awal sebagai node saat ini.

Iterasi: Untuk node saat ini, periksa semua tetangganya. Jika jarak ke tetangga melalui node saat ini lebih pendek dari jarak yang diketahui sebelumnya, perbarui jarak ke tetangga tersebut. Setelah semua tetangga diperiksa, tandai node saat ini sebagai selesai dan pilih node dengan jarak terpendek yang belum selesai sebagai node saat ini.

Pengulangan: Ulangi langkah ini sampai semua node telah ditandai selesai .

Pembaruan Tabel Routing:

Hasil dari algoritma Dijkstra adalah pohon jalur terpendek (SPF Tree) dengan router sebagai root. Informasi dari pohon ini digunakan untuk memperbarui tabel routing OSPF dengan jalur terbaik ke setiap tujuan .

Pemeliharaan Jaringan:

OSPF terus memantau keadaan link di jaringan dan akan menjalankan ulang algoritma Dijkstra setiap kali terjadi perubahan topologi, memastikan bahwa tabel routing selalu diperbarui dengan informasi terbaru .

2.5.2 Routing Information Protocol v2 (*RIPv2*)

RIPv2 (*Routing Information Protocol version* 2) adalah protokol routing vektor jarak yang banyak digunakan untuk merutekan paket data antara jaringan. *RIPv2* memperbarui pendahulunya, RIPv1, dengan memperkenalkan beberapa perbaikan, termasuk dukungan untuk *classless inter-domain routing* (*CIDR*) dan kemampuan membawa informasi *subnet mask*. Salah satu keunggulan utama *RIPv2* adalah sederhana dan mudah dikonfigurasi. Router *RIPv2* secara periodik dalam melakukan pertukaran dalam *routing*, yang dikenal sebagai *Routing Information Protocol* (RIP), untuk memberi tahu satu sama lain tentang jaringan yang dapat mereka capai. Pembaruan ini mencakup informasi tentang alamat IP jaringan, subnet mask, dan metrik, yang mewakili biaya atau jarak untuk mencapai jaringan. *RIPv2* menggunakan metrik *hop count* untuk menentukan jalur terbaik ke jaringan tujuan. *Hop count* mewakili jumlah router yang harus dilalui paket untuk mencapai tujuan.

Meskipun sederhana, *RIPv2* memiliki beberapa keterbatasan. Salah satu kekurangan utama adalah waktu konvergensi yang lambat. Ketika terjadi perubahan dalam jaringan, seperti kegagalan *link* atau perubahan topologi jaringan, router *RIPv2* memerlukan waktu untuk memperbarui tabel routing mereka dan menyebarkan perubahan ke router lainnya. Konvergensi yang lambat ini dapat menyebabkan routing yang tidak optimal dan ketidakstabilan jaringan dalam lingkungan jaringan dinamis. Untuk mengatasi keterbatasan *RIPv2*, berbagai perbaikan dan alternatif telah diusulkan. Sebagai contoh, penggunaan protokol *routing* seperti *OSPF* (*Open Shortest Path First*) dan *IS-IS* (*Intermediate System to Intermediate System*) memberikan konvergensi

yang lebih cepat dan skalabilitas yang lebih baik dibandingkan dengan *RIPv2*. Protokol ini menggunakan algoritma canggih dan teknik flooding untuk mendistribusikan informasi routing secara efisien dan beradaptasi lebih cepat terhadap perubahan jaringan. Adapun prinsip kerja berdasarkan Algoritma *Bellman-Ford* pada protokol *RIPv2* sebagai berikut :

1. Pengumpulan Informasi Routing:
    - RIPv2 menggunakan algoritma Bellman-Ford untuk menghitung jalur terbaik ke setiap tujuan. Setiap router RIPv2 secara berkala mengirimkan seluruh tabel routingnya ke router tetangga dalam bentuk pesan update routing .
2. Algoritma Bellman-Ford:
    - Algoritma Bellman-Ford bekerja dengan prinsip berikut:
        - Inisialisasi**:** Tetapkan jarak ke node awal (router sendiri) sebagai 0 dan jarak ke semua node lain sebagai tak terhingga.
        - Iterasi**:** Setiap router menerima tabel routing dari tetangganya dan menggabungkan informasi tersebut ke dalam tabel routingnya sendiri. Untuk setiap entri dalam tabel routing tetangga, tambahkan jarak ke tetangga tersebut ke jarak dalam entri tersebut. Jika hasilnya lebih pendek dari jarak yang diketahui sebelumnya untuk tujuan tersebut, perbarui jarak dalam tabel routing.
        - Pengulangan**:** Ulangi langkah ini sampai semua router memiliki tabel routing yang stabil, di mana tidak ada lagi perubahan jarak .
3. Pembaruan Tabel Routing:
    - Tabel routing di setiap router diperbarui berdasarkan informasi yang diterima dari tetangga. RIPv2 menambahkan beberapa fitur seperti subnet mask dan otentikasi ke pesan update routing untuk meningkatkan fungsionalitas dan keamanan dibandingkan RIP versi sebelumnya .
4. Pengendalian Looping:
    - Untuk mencegah looping routing, RIPv2 menggunakan teknik seperti split horizon, route poisoning, dan hold-down timers. Ini

memastikan bahwa informasi routing tidak beredar kembali ke router asalnya, yang dapat menyebabkan looping .

5. Pemeliharaan Jaringan:
    - RIPv2 terus mengirimkan update routing secara berkala (setiap 30 detik) dan juga setiap kali terjadi perubahan topologi, memastikan bahwa tabel routing selalu diperbarui dengan informasi terbaru .

2.6 *Quagga Routing Engine*

Quagga adalah rangkaian perangkat lunak *routing* jaringan *open source* yang menyediakan implementasi beberapa protokol perutean, termasuk OSPF, RIP, dan BGP. Ini banyak digunakan karena fleksibilitas, ketahanan, dan kompatibilitasnya dengan berbagai perangkat dan sistem jaringan. *Quagga*, seperti yang dijelaskan oleh dokumentasi resminya (*Quagga*, n.d.), menyediakan rangkaian perangkat lunak perutean yang stabil dan serbaguna yang mampu mengelola berbagai protokol perutean. Ini sangat dapat diperluas dan dapat dioperasikan dengan banyak perangkat dan sistem jaringan, menjadikannya pilihan utama di antara administrator jaringan. Filsfils dkk. (2012) memberikan rincian arsitektur *Quagga* secara rinci. Mereka menjelaskan bagaimana desain modular *Quagga* memungkinkan setiap protokol routing beroperasi sebagai.

Dalam evaluasi kinerja yang dilakukan oleh Zhang dan Garcia-Luna-Aceves (2006), *Quagga* ditemukan menyediakan layanan routing yang andal dan efisien. Studi mereka juga menyoroti kemampuan *Quagga* untuk beradaptasi dengan kondisi dan kebutuhan jaringan yang berbeda. Penerapan praktis *Quagga* dibahas dalam Proyek ONOS (n.d.). Proyek ini menggunakan *Quagga* untuk mendemonstrasikan pengelolaan berbagai protokol routing yang efektif. Aplikasi dunia nyata ini menggarisbawahi kompatibilitas dan kinerja *Quagga* dalam lingkungan jaringan berskala besar. Perbandingan yang dilakukan oleh Pignataro dkk. (2012) antara *Quagga* dan rangkaian perangkat lunak perutean lainnya menemukan *Quagga* fleksibel dan dapat diandalkan. Studi tersebut menekankan kemampuan *Quagga* untuk menangani berbagai protokol routing dengan tetap menjaga kinerja yang stabil dan efisien.

2.7 Topologi *Tree*

Topologi *tree* merupakan salah satu topologi dalam jaringan yang berbentuk pohon dengan simpul pusat sebagai akarnya, mendapat perhatian karena kemampuannya mendistribusikan lalu lintas dengan efisien dan memberikan skalabilitas. Menurut Forouzan (2006), penggunaan umum topologi pohon terlihat dalam jaringan kampus, di mana hal ini membantu mengelola lalu lintas dan menyediakan akses terstruktur ke sumber daya jaringan. Simpul pusat, seperti yang dijelaskan oleh Tanenbaum (2011), berfungsi sebagai pusat distribusi lalu lintas dalam topologi pohon. Meskipun terdapat ketergantungan pada simpul pusat, struktur hierarkis ini mendukung skalabilitas dan memudahkan penambahan simpul ke dalam jaringan. Penerapan topologi pohon dalam pusat data juga menjadi sorotan. Al-Fuqaha et al. (2015) menyatakan bahwa di pusat data, topologi pohon memberikan struktur terorganisir yang mendukung aplikasi dan layanan yang kompleks. Dalam konteks ini, topologi pohon memungkinkan distribusi lalu lintas yang terkontrol di seluruh pusat data.

Sebuah penelitian oleh Garg et al. (2018) menyoroti integrasi topologi pohon dengan konsep *Software-Defined Networking* (*SDN*) untuk meningkatkan fleksibilitas dan manajemen jaringan. Dengan *SDN*, kontrol dan pemrograman sentral dapat dioptimalkan untuk mengatur lalu lintas dengan lebih efektif dalam topologi pohon. Secara keseluruhan, topologi pohon terus menjadi pilihan yang signifikan dalam desain jaringan komputer. Dukungan terhadap skalabilitas, distribusi lalu lintas, dan integrasi dengan teknologi terkini seperti *SDN* membuat topologi ini relevan dalam dinamika jaringan modern.

2.8 *Wireshark*

Gambar 2.6 Logo Wireshark (https://www.Wireshark.org/)

Wireshark adalah alat analisis paket jaringan *open-source* yang paling banyak digunakan dan populer. Dikenal sebelumnya sebagai Ethereal, *Wireshark* dikembangkan oleh Tim *Wireshark* dan merupakan proyek sumber terbuka yang mendukung sejumlah besar protokol jaringan. Alat ini mampu memonitor dan

menganalisis paket lalu lintas jaringan secara mendetail, membuatnya menjadi pilihan utama di kalangan profesional jaringan dan peneliti. Menurut Chappell (2010), *Wireshark* menyediakan antarmuka pengguna grafis yang intuitif dan kaya fitur, memungkinkan pengguna untuk menangkap dan memeriksa paket lalu lintas dengan mudah. Fitur filtrasi yang kuat memungkinkan penyaringan berdasarkan berbagai kriteria, termasuk protokol, alamat IP, dan informasi lainnya, mempermudah analisis paket yang spesifik.

Wireshark mendukung sejumlah besar protokol jaringan, dari protokol umum seperti TCP, UDP, dan HTTP, hingga protokol khusus dan proprietary. Menurut Ramachandran (2016), keunggulan ini membuat *Wireshark* menjadi alat yang sangat fleksibel dan dapat diandalkan untuk memecahkan masalah jaringan, mengidentifikasi anomali, dan memahami interaksi protokol dalam berbagai skenario. Penelitian oleh Kothari dan Rana (2014) menyoroti bahwa *Wireshark* bukan hanya alat pasif untuk memonitor lalu lintas, tetapi juga mampu melakukan analisis yang mendalam terhadap paket, menyajikan data dalam format yang mudah dipahami. Kemampuan untuk melihat struktur paket secara detail dan menganalisis interaksi antara perangkat dalam jaringan menjadikan *Wireshark* penting dalam mendeteksi dan menangani masalah jaringan. Selain itu, *Wireshark* memiliki komunitas pengguna yang besar dan aktif, yang berkontribusi pada perkembangan dan pemeliharaan alat ini. Menurut Casad (2017), forum, dokumentasi, dan sumber daya lainnya yang disediakan oleh komunitas ini memperkaya pengalaman pengguna dan memastikan pembaruan dan perbaikan terus-menerus. Meskipun keunggulan *Wireshark*, Deri et al. (2006) mencatat bahwa pemahaman mendalam tentang protokol jaringan dan analisis paket diperlukan untuk menggunakan alat ini secara efektif. Oleh karena itu, pengguna perlu memiliki pengetahuan yang cukup untuk menginterpretasikan hasil analisis yang diberikan oleh *Wireshark*.

2.9 *Internet Communication Message Protocol* (*ICMP*)

ICMP, atau yang dikenal sebagai *Internet Control Message Protocol*, adalah protokol jaringan yang digunakan untuk tujuan diagnostik dan pengendalian dalam jaringan IP. Beroperasi pada lapisan jaringan dari *Internet Protocol Suite*, *ICMP* memungkinkan perangkat untuk berkomunikasi melalui pesan-pesan kesalahan dan informasi operasional. Pesan-pesan *ICMP* biasanya dikirim dalam paket IP untuk memberi tahu perangkat jaringan tentang berbagai kondisi dan peristiwa.

ICMP memiliki beberapa fungsi dalam jaringan IP. Ini membantu melaporkan kesalahan dalam pemrosesan paket IP dengan menghasilkan pesan-pesan kesalahan yang memberikan informasi berharga untuk pemecahan masalah. Selain itu, *ICMP* mendukung manajemen dan pemecahan masalah jaringan dengan menyediakan alat seperti utilitas "ping", yang menguji konektivitas jaringan dan mengukur waktu tempuh pulang-pergi. *ICMP* juga berperan dalam penemuan path MTU, membantu menentukan ukuran paket maksimum yang dapat dikirim tanpa fragmentasi. Selain pelaporan kesalahan dan pengujian jaringan, *ICMP* mencakup berbagai jenis pesan lain yang memfasilitasi pertukaran informasi antara perangkat. Pesan-pesan ini mencakup topik seperti routing, konfigurasi, dan kondisi jaringan.

Secara ringkas, *ICMP* adalah protokol penting dalam jaringan IP yang membantu fungsi diagnostik, pengendalian, dan manajemen. Ini memungkinkan perangkat berkomunikasi melalui pesan-pesan kesalahan, menguji konektivitas dan kinerja jaringan, serta pertukaran informasi operasional. Pemahaman menyeluruh tentang *ICMP* dan jenis pesannya menjadi penting untuk pemecahan masalah dan pemantauan jaringan yang efisien.

2.10 *iperf*

Iperf adalah alat pengukuran kinerja jaringan sumber terbuka yang banyak digunakan, memungkinkan pengguna mengukur throughput, bandwidth, dan latensi koneksi jaringan. Biasanya digunakan untuk pengujian jaringan, pemecahan masalah, dan perencanaan kapasitas.

Salah satu keunggulan utama iperf adalah kemudahan dalam penggunaannya. Program ini menyediakan arsitektur klien-server, di mana satu perangkat bertindak sebagai server dan perangkat lain bertindak sebagai klien. Klien mengirim paket data ke server, dan server mengukur laju penerimaan paket. Hal ini memungkinkan pengguna menilai kinerja koneksi jaringan dengan mengukur jumlah data yang dapat ditransfer dalam jangka waktu tertentu. Iperf menawarkan berbagai opsi dan parameter yang memungkinkan pengguna menyesuaikan parameter pengujian sesuai dengan kebutuhan mereka. Misalnya, pengguna dapat menentukan durasi pengujian, ukuran paket data, jumlah koneksi paralel, dan arah transfer data. Fleksibilitas ini memungkinkan pengguna mensimulasikan kondisi jaringan yang berbeda dan menilai kinerja link, perangkat, dan aplikasi jaringan. Selain itu, Iperf mendukung pengujian baik dengan TCP (*Transmission Control Protocol*) maupun

UDP (*User Datagram Protocol*). Pengujian TCP mengukur throughput dan latensi koneksi jaringan dalam kondisi normal dan dapat diandalkan, sementara pengujian UDP berfokus pada pengukuran throughput maksimal yang dapat dicapai dan tingkat kehilangan paket. Hal ini membuat iperf cocok untuk berbagai skenario evaluasi kinerja jaringan.

Selain antarmuka baris perintahnya, iperf juga menyediakan antarmuka pengguna grafis (GUI) dan integrasi dengan alat pemantauan dan analisis jaringan lainnya. Fitur-fitur ini meningkatkan kegunaan dan memperluas kemampuan Iperf, memungkinkan pengguna memvisualisasikan dan menganalisis data kinerja jaringan dengan lebih efektif.

2.11 Analisis Kinerja Jaringan

Analisis kinerja jaringan merupakan aspek kritis dalam pengelolaan dan pemeliharaan infrastruktur jaringan. Berbagai metode dan alat telah dikembangkan untuk menyediakan wawasan mendalam tentang bagaimana jaringan beroperasi dan sejauh mana kinerjanya memenuhi kebutuhan pengguna. Tinjauan pustaka ini akan mengeksplorasi beberapa konsep utama dalam analisis kinerja jaringan serta teknik dan alat yang digunakan dalam konteks ini.

Metode analisis kinerja jaringan mencakup pendekatan analitis dan simulasi. Menurut Kurose dan Ross (2017), throughput mengukur jumlah data yang dapat dikirim atau diterima oleh jaringan dalam satu periode waktu. Latensi, di sisi lain, mencerminkan waktu yang diperlukan untuk data melakukan perjalanan dari satu titik ke titik lain dalam jaringan. Keandalan mengacu pada kemampuan jaringan untuk memberikan layanan tanpa gangguan dan kehilangan data.

Dalam konteks analisis kinerja jaringan, Mohan et al. (2020) menunjukkan tren menuju analisis real-time dan prediktif. Analisis real-time memungkinkan pemantauan langsung dan respons cepat terhadap perubahan kondisi jaringan. Sementara itu, analisis prediktif menggunakan kecerdasan buatan untuk memprediksi potensi masalah kinerja sebelum terjadi. Dengan demikian, analisis kinerja jaringan terus berkembang sejalan dengan perkembangan teknologi dan kebutuhan pengguna. Penerapan metode analitis dan simulasi, pemahaman parameter kinerja kunci, dan penggunaan alat analisis yang canggih menjadi esensial dalam memastikan kinerja optimal dan ketersediaan layanan jaringan.

2.11.1 Throughput

Throughput adalah parameter kinerja yang sangat penting dalam jaringan komputer dan sistem komunikasi. Menurut Tanenbaum dan Wetherall (2011), throughput mengukur jumlah data yang berhasil ditransfer dari satu titik ke titik lain dalam suatu jaringan dalam periode waktu tertentu. Ini dapat dianggap sebagai ukuran efisiensi jaringan dan sering diukur dalam bit per detik (bps) atau byte per detik (Bps).

Persamaan untuk menghitung *Throughput* :

$$\text{Throughput} = \frac{Jumlah\ data\ diterima}{Jumlah\ data\ terkirim} \text{ x } 100\%$$

Tabel 1. Kategori Throughput

Kategori	***Throughput***	**Indeks**
Sangat Bagus	100%	4
Bagus	75%	3
Sedang	50%	2
Jelek	<25%	1

Sumber : TIPHON

Adapun faktor yang mempengaruhi *Throughput* menurut Forouzan, B. A (2017) :

- *Bandwidth*: Kapasitas maksimum dari link jaringan.
- *Network Congestion* : Kepadatan lalu lintas yang menyebabkan penurunan kinerja jaringan.
- *Packet Size* : Ukuran paket yang lebih besar bisa meningkatkan *throughput*, tetapi juga bisa menyebabkan fragmentasi.
- *Protocol Overhead* : Protokol komunikasi menambahkan *overhead* yang dapat mengurangi *throughput* efektif.
- *Hardware Performance* : Kemampuan perangkat jaringan seperti *router* dan *switch*.
- *Signal Interference*: Gangguan sinyal pada jaringan nirkabel dapat mengurangi throughput.

2.11.2 *Latency*

Latensi adalah salah satu aspek kritis dalam kehandalan jaringan. Kurose dan Ross (2017) menjelaskan bahwa latensi mencakup sejumlah faktor, termasuk latensi propagasi, latensi transmisi, latensi pengolahan, dan latensi antrian. Pengurangan latensi penting untuk meningkatkan responsivitas dan kinerja jaringan.

Tabel 2. Kategori *Latency*

Kategori	*Latency*	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 s/d 300 ms	3
Sedang	300 s/d 450 ms	2
Jelek	>450 ms	1

Sumber : TIPHON

Adapun faktor-faktor yang mempengaruhi *Packet Loss*:

- *Network Congestion*: Kepadatan lalu lintas dapat menyebabkan paket dibuang.
- *Faulty Hardware*: Perangkat jaringan yang rusak atau tidak berfungsi dengan baik.
- *Signal Interference*: Gangguan sinyal pada jaringan nirkabel.
- *Buffer Overflow*: Ketika buffer pada perangkat jaringan penuh, paket baru akan dibuang.
- *Link Failure*: Kegagalan fisik pada link jaringan.

2.11.3 *Packet Loss*

Packet loss adalah fenomena di mana paket data hilang selama transmisi melalui jaringan. Menurut Forouzan dan Fegan (2006), faktor-faktor seperti kegagalan koneksi, beban lalu lintas yang tinggi, dan konfigurasi yang tidak benar dapat menyebabkan kehilangan paket. Pengelolaan packet loss menjadi kunci dalam memastikan integritas data dalam jaringan.

Tabel 3. Kategori *Packet Loss*

Kategori	*Packet loss*	Indeks
Sangat Bagus	0%	4

Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

Sumber : TIPHON

Adapun faktor yang mempengaruhi *Latency*:

- *Propagation Delay*: Waktu yang diperlukan untuk sinyal berjalan melalui media transmisi.
- *Transmission Delay*: Waktu yang diperlukan untuk memuat paket data ke media transmisi.
- *Processing Delay*: Waktu yang diperlukan oleh perangkat jaringan untuk memproses paket data.
- *Queuing Delay*: Waktu yang dihabiskan oleh paket data dalam antrian di perangkat jaringan.
- *Distance*: Jarak fisik antara pengirim dan penerima.

2.11.4 *Convergence Time*

Convergence adalah proses di mana jaringan atau sistem menggabungkan berbagai layanan atau teknologi untuk bekerja bersama-sama secara efisien. Atau dengan kata lain *convergence* merupakan waktu ketika seluruh perangkat yang ada pada jaringan telah terhubung. Menurut Tanenbaum dan Wetherall (2011), contoh *convergence* termasuk penyatuan layanan telekomunikasi dalam infrastruktur IP. Proses ini penting untuk meningkatkan efisiensi dan fleksibilitas jaringan. Adapun faktor-faktor yang mempengaruhi *Convergence Time* sebagai berikut:

- *Protocol Design*: Protokol routing yang berbeda memiliki mekanisme konvergensi yang berbeda (misalnya, OSPF vs. RIP).
- *Network Size*: Jaringan yang lebih besar biasanya membutuhkan waktu lebih lama untuk mencapai konvergensi.
- *Link State Changes*: Frekuensi dan jumlah perubahan link state dalam jaringan.
- *Router Processing Power*: Kemampuan perangkat keras router untuk memproses perubahan topologi dan menghitung ulang tabel routing.
- *Timer Settings*: Pengaturan waktu dalam protokol routing, seperti interval update dan hold-down timers.