Implementasi *Switch Openflow* Dengan Menggunakan Raspberry Pi Dalam Jaringan Sdn (*Software Defined Networking)*

Firman Aprianto[1], Taviv Sutisna[2], Taufik Irfan[3]

[1,2,3]*Jurusan Teknik Elektro, Politeknik Negeri Bandung, Bandung 40012*

E-mail: firman.aprianto.tkom19@polban.ac.id

E-mail: tavivpolban@polban.ac.id

E-mail: taufik.irfan@polban.ac.id

ABSTRAK

Switch adalah suatu perangkat pada jaringan komputer yang berfungsi untuk menghubungkan antar komputer, *switch* ini bekerja pada *data layer*. Seperti contoh pada suatu infrastruktur jaringan komputer seperti pada jaringan *software defined networking* (SDN). Jaringan SDN ini merupakan konsep infrastruktur jaringan yang memisahkan antara *data plane* dan *control plane*. Pada SDN ini membutuhkan perangkat *switch* yang mendukung protokol openflow atau biasa disebut *switch openflow*. Openflow merupakan protokol komunikasi antara *control plane* dan *data plane* sehingga dapat mempermudah dalam mengelola perangkat dalam suatu jaringan. Pada penelitian ini akan diimplementasikan perangkat *switch openflow* menggunakan perangkat raspberry pi dalam jaringan SDN yang sederhana dimana raspberry pi merupakan perangkat yang dapat digunakan sebagai *data plane* dan untuk bagian *controller* akan dibuat menggunakan sebuah *software* OpenDayLight *controller*. Selanjutnya akan diukur QoS pada saat *forwarding packet* yang dilakukan antar *host* pada topologi yang telah dirancang dan terhubung pada *switch openflow* ini. Hasil pengukuran QoS ini didapatkan hasil *delay* sebesar 5,2 mS, *avg. jitter* sebesar 14,42 ms, *packet loss* sebesar 0% dan *avg. throughput* sebesar 3,662 Mbps. Sehingga dapat disimpulkan bahwa raspbery pi memiliki performa yang baik sebagai pengganti *switch openflow* pada infrastruktur SDN.

Kata Kunci

Control Plane, Data Plane, OpenDayLight, Raspberry pi, Switch openflow

1. PENDAHULUAN

Semakin berkembangnya zaman menyebabkan teknologi yang ada semakin berkembang, salah satunya yaitu teknologi komunikasi jaringan. Oleh karena itu, penggunaan perangkat teknologi jaringan semakin hari semakin banyak digunakan yang menyebabkan meningkatnya kebutuhan akan perangkat jaringan dan kemudahan dalam mengkonfigurasi jaringan. Untuk meningkatkan fleksibilitas dan kompleksitas maka terdapat suatu konsep jaringan yaitu *Software Defined Networking (SDN)*. Jaringan SDN adalah konsep infrastruktur jaringan yang memisahkan antara *data plane* dan *control plane*. Pada SDN ini membutuhkan perangkat *switch* yang mendukung protokol openflow atau biasa disebut *switch openflow*. Openflow merupakan protokol komunikasi antara *control plane* dan *data plane* untuk mempermudah dalam mengelola perangkat dalam suatu jaringan.

Switch openflow yang dirancang menggunakan raspberry pi dalam jaringan SDN yang sederhana dimana raspberry pi merupakan perangkat yang dapat digunakan sebagai *data plane* dan untuk bagian *controller* akan dibuat menggunakan sebuah *software* OpenDayLight *controller*.

2. PUSTAKA TERKAIT

2.1 *Software Defined Network* (SDN)

Software-defined network (SDN) adalah arsitektur yang dirancang untuk membuat jaringan lebih fleksibel dan lebih mudah dikelola. Konsep SDN ini dengan cara memisahkan *control plane* pada *controller* dengan *data plane* untuk *forwarding data* pada perangkat jaringan**.** [1]

2.2. Openflow

OpenFlow adalah protokol yang digunakan pada SDN yang berada di antara *control plane* dan *data plane.* OpenFlow ini dapat mengatur *routing* dan pengiriman paket dalam sebuah *switch***,** dimana *switch* berfungsi hanya untuk meneruskan paket. Dengan *OpenFlow* kita dapat untuk mengakses dan memanipulasi *forwarding plane* secara langsung pada perangkat seperti router dan switch. *Controller* SDN yang digunakan harus sudah mendukung protokol *Openflow*. [2]

2.3 OpenVswitch

Open vSwitch adalah switch virtual multilayer berkualitas produksi yang dilisensikan di bawah lisensi *opensource* Apache 2.0. Open vSwitch mendukung antarmuka dan protokol manajemen standar (misalnya NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag) [3].

2.4 *Controller*

Controller atau *control plane* adalah *layer* yang berfungsi untuk mengontrol menentukan jalan yang digunakan untuk mengirim paket atau frame. Control plane bertanggung jawab untuk mengisi tabel *routing*, menggambar topologi jaringan, tabel forwarding dan karenanya memungkinkan fungsi *data plane* [4].

Controller pada SDN memiliki jenis yang berbeda -beda, diantaranya OpenDayLight *controller* [5] [6]. Selain itu terdapat juga penelitian dengan menggunakan ONOS *controller* [7]. Terdapat juga jurnal terkait dengan menggunakan Floodlight *controller* [8]. Terdapat juga jurnal penelitian tentang SDN dengan menggunakan dan POX *controller* [9].

2.4. Parameter Pengukuran

2.4.1 *Throughput*

Throughput merupakan *bandwidth* aktual saat itu juga dimana kita sedang melakukan koneksi. Satuan yang dimilikinya sama dengan *bandwidth* yaitu bps. Standar *throughput* TIPHON ditunjukkan pada tabel 1. [10]

Tabel 1. Standar *Throughput* TIPHON [10]

Kategori	***Throughput* (Mbps)**	**Indeks**
Sangat Baik	>2,1 Mbps	4
Lebih Baik	1,2 - 2,1 Mbps	3
Baik	700 – 1200 Kbps	2
Cukup Baik	338 -700 Kbps	1
Buruk	0 - 338 Kbps	0

2.4.2 *Delay*

Delay adalah waktu tunda yang disebabkan oleh proses transmisi dari satu titik ke titik lain yang menjadi tujuannya. Satuan dari *delay* yaitu mS. Berikut ini merupakan tabel kategori *delay* berdasarkan standar TIPHON ditunjukkan pada tabel 2. [10]

Tabel 2. Standar *Delay* TIPHON [10]

Kategori	***Delay* (ms)**	**Indeks**
Sangat Bagus	<15	4
Bagus	15-150	3
Sedang	150-400	2
Jelek	>400	1

2.4.3 *Jitter*

Jitter adalah selisih antara *delay* pertama dengan *delay* selanjutnya. Satuan dari *jitter* yaitu mS. Berikut ini merupakan tabel kategori *jitter* berdasarkan standar TIPHON ditunjukkan pada tabel 3. [10]

Tabel 3. Standar *Jitter* TIPHON [10]

Kategori	*Jitter* (ms)	Indeks
Sangat Bagus	0-75	4
Bagus	75-125	3
Sedang	125-225	2
Jelek	>225	1

2.4.4 *Packet Loss*

Packet loss adalah hilangnya paket pada proses transmisi data dari titik pengirim ke titik penerima. Kategori *packet loss* berdasarkan standar TIPHON ditunjukkan pada tabel 4. [10]

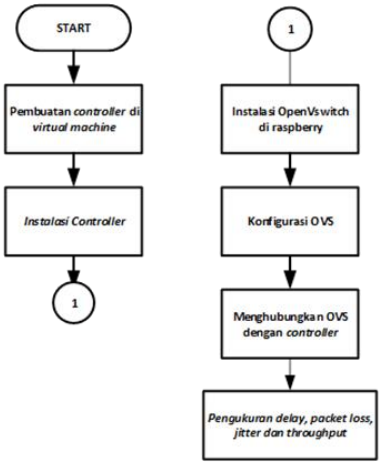
Tabel 4. Standar *Packet Loss* TIPHON [10]

Kategori	*Packet Loss* (%)	Indeks
Sangat Bagus	0 – 3 %	4
Bagus	3 – 15 %	3
Sedang	15 - 25 %	2
Jelek	>25 %	1

3. METODOLOGI PELAKSANAAN

3.1 Perancangan

Tahapan perancangan dari sistem yang akan dibuat ditunjukkan melalui diagram alir pada gambar 1.

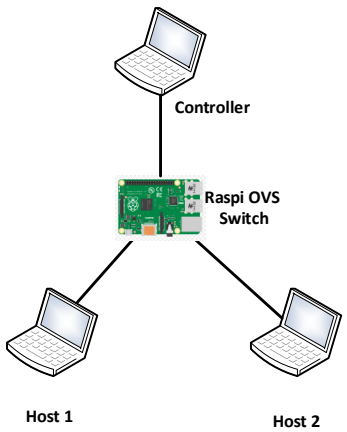


Gambar 1. Diagram Alir Perancangan

Konfigurasi pada *controller* dilakukan dengan cara menginstall OpenDayLight *controller*. Sedangkan konfigurasi pada OVS dilakukan dengan cara menginstall Open Vswitch, dimana Open Vswitch merupakan *software* yang memberikan protokol openflow sehingga raspi dapat mendukung protokol openflow.

3.1.1 Perancangan Topologi

Pada penelitian ini menggunakan topologi linear yang ditunjukkan pada gambar 2. Topologi tersebut terdiri dari 1 *switch*, 1 *controller* dan 2 *host*.



Gambar 2. Topologi Percobaan

3.2 Spesifikasi Perangkat

Agar sistem ini dapat terbangun, maka dibutuhkan *hardware* dan *software* untuk membangun sistem *software defined networking*.

Hardware yang digunakan memiliki spesifikasi yang ditunjukkan pada tabel 5 berikut.

Tabel 5. Kebutuhan *Hardware*

Hardware	Jumlah	Spesifikasi
Laptop (*controller*)	1 buah	• CPU AMD A4 • RAM 8 GB • SSD 256 GB
Laptop (*Host*)	2 buah	• CPU INTEL CORE I3

		• RAM 4 GB • HDD 500 GB
Raspberry Pi 3 b +	1 buah	• RAM 1 GB • 4 USB 2 ports

Sedangkan untuk kebutuhan *software* yang digunakan untuk membangun sistem *software defined networking* ditunjukkan pada tabel 6 berikut.

Tabel 6. Kebutuhan *Software*

Software	Keterangan
Linux Ubuntu 20.0.4 LTS	Sistem Operasi *Controller*
OpenDayLight (Berrylium)	SDN *controller*
Iperf	Monitoring Jaringan
Raspbian Jessie (OS)	Sistem Operasi pada raspberry (*switch ovs*)

4. HASIL PENGUJIAN DAN ANALISIS

Setelah melakukan perancangan dan realisasi, selanjutnya yaitu melakukan pengujian terhadap fungsionalitas sistem dan pengujian QoS sistem diantaranya *delay*, *jitter*, *packet loss* dan *throughput* dari sistem yang telah direalisasikan.

4.1 Pengujian Fungsionalitas

4.1.1 Pengujian Konektivitas

Pengujian ini dilakukan untuk mengetahui konektivitas antara *controller* dan OVS yang telah berhasil terpasang.

Gambar 3. Tampilan pada GUI controller

Berdasarkan gambar 3, OVS telah berhasil terhubung ke *controller* yang ditampilan pada tampilan GUI *controller* dengan munculnya perangkat dan mac *address* dari perangkat yang terhubung pada *switch openflow*, dimana terdapat dua PC yang ditampilan dan telah sesuai dengan jumlah perangkat yang ditambahkan pada OVS.

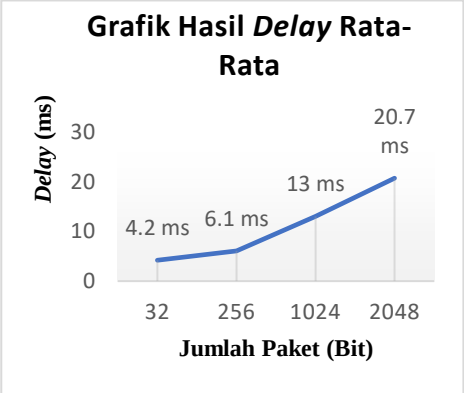
4.2 Pengujian Parameter QoS

4.1.1 Pengujian *Delay*

Pengujian terhadap *delay* dilakukan pada CMD dengan protokol ICMP. Pada pengujian ini dilakukan dengan 4 skenario berdasarkan jumlah byte yang dikirim mulai dari 32bit, 256 bit, 1024 bit, dan 2048 bit, masing-masing skenario dilakukan dengan 10 kali percobaan dan diambil nilai rata-ratanya. Besar hasil *delay* ditunjukkan pada Tabel 7.

Tabel 7. *Delay* rata-rata

Jumlah *bit*	**Rata-rata *Delay* (mS)**
32	4,2
256	6,1
1024	13
2048	20,7



Gambar 4. Grafik Delay rata-rata

Pada Gambar 4 ditunjukkan hasil *delay* rata-rata, pada pengujian dengan data 32bit, *delay* rata-rata yang dihasilkan sebesar 4,2 ms. Sedangkan pada data 256 bit, dihasilkan *delay* rata-rata sebesar 6,1 ms. Pengujian selanjutnya yaitu dengan data 1024 bit, dihasilkan *delay* rata-rata sebesar 13 ms. Terakhir yaitu pengujian dengan data 2048 bit, dihasilkan *delay* rata-rata sebesar 20,7 ms. *Delay* yang dihasilkan berada dikategori sangat baik karena berada pada rentang

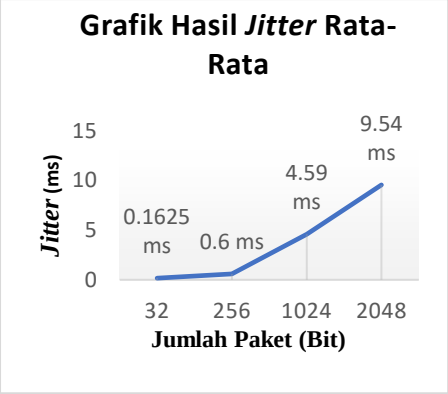
dibawah 150 mS dengan indeks 4 pada standarisasi TIPHON seperti yang ditunjukkan pada tabel 2.

4.2.2 Pengujian *Jitter*

Pengujian *jitter* dilakukan pada CMD dengan menggunakan *iperf*. Pada pengujian ini dilakukan dengan 4 skenario berdasarkan jumlah byte yang dikirim mulai dari 32 bit, 256 bit, 1024 bit, dan 2048 bit, masing-masing skenario dilakukan dengan 10 kali percobaan dan diambil nilai rata-ratanya. Besar hasil *jitter* ditunjukkan pada tabel 8.

Tabel 8. Pengukuran *Jitter*

Jumlah *bit*	Rata-rata *jitter* (ms)
32	0,1625
256	0,762
1024	4,59
2048	9,54



Gambar 5. Grafik Hasil Jitter Rata-Rata

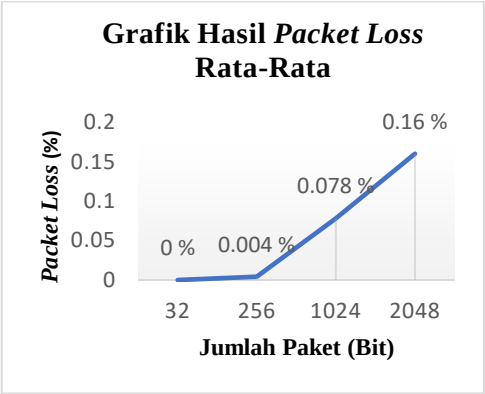
Pada Gambar 5 ditunjukkan hasil *jitter* rata-rata, pada pengujian dengan data 32 bit, *jitter* rata-rata yang dihasilkan sebesar 0,1625 ms. Sedangkan pada data 256 bit, dihasilkan *jitter* rata-rata sebesar 0,60 ms. Pengujian selanjutnya yaitu dengan data 1024 bit, dihasilkan *jitter* rata-rata sebesar 4,59 ms. Terakhir yaitu pengujian dengan data 2048 bit, dihasilkan *jitter* rata-rata sebesar 9,54 ms. *Jitter* yang dihasilkan dikategorikan baik dengan indeks 3 pada standarisasi TIPHON seperti yang ditunjukkan pada tabel 3.

4.2.3 Pengujian *Packet Loss*

Pengujian *packet loss* dilakukan pada CMD dengan menggunakan *iperf*. Pada pengujian ini dilakukan dengan 4 skenario berdasarkan jumlah byte yang dikirim mulai dari 32 bit, 256 bit, 1024 bit, dan 2048 bit, masing-masing skenario dilakukan dengan 10 kali percobaan dan diambil nilai rata-ratanya.

Tabel 9. Pengukuran *Packet loss*

Jumlah *bit*	Rata-rata *packet loss* (mS)
32	0
256	0,004
1024	0,078
2048	0,16



Gambar 6. Grafik Hasil *Packet loss* rata-rata

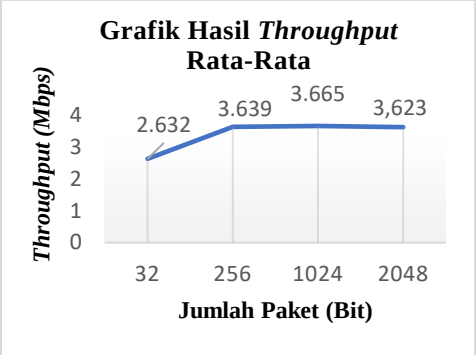
Berdasarkan gambar 8 ditunjukkan hasil *packet loss* rata-rata, pada pengujian dengan data 32 bit, *packet loss* yang dihasilkan sebesar 0% atau tidak ada paket yang hilang dalam proses pengiriman data. Sedangkan pada data 256 bit, dihasilkan *packet loss* rata-rata sebesar 0,004 %. Pengujian selanjutnya yaitu dengan data 1024 bit, dihasilkan *packet loss* rata-rata sebesar 0,008%. Terakhir yaitu pengujian dengan data 2048 bit, dihasilkan *packet loss* rata-rata sebesar 0,016%. *Packet loss* yang dihasilkan sangat baik dengan indeks 4 pada standarisasi TIPHON seperti yang ditunjukkan pada tabel 4.

4.2.4 Pengujian *Throughput*

Pengujian *throughput* dilakukan pada CMD dengan menggunakan *iperf*. Pada pengujian ini dilakukan dengan 4 skenario berdasarkan jumlah byte yang dikirim mulai dari 32 bit, 256 bit, 1024 bit, dan 2048 bit, masing-masing skenario dilakukan dengan 10 kali percobaan dan diambil nilai rata-ratanya.

Tabel 4.5 Pengukuran *Throughput*

Jumlah *bit*	**Rata-rata *throughput* (Mbps)**
32	2,632
256	3,639
1024	3,665
2048	3,623



Gambar 7. Grafik Pengujian Throughput

Berdasarkan gambar 7 ditunjukkan hasil *throughput* rata-rata, pada pengujian dengan data 32 bit, dihasilkan *throughput* rata-rata sebesar 2,632 Mbps. Sedangkan pada data 256 bit, dihasilkan *throughput* rata-rata sebesar 3,639 Mbps. Pengujian selanjutnya yaitu dengan data 1024 bit, dihasilkan *throughput* rata-rata sebesar 3,665 Mbps. Terakhir yaitu pengujian dengan data 2048 bit, dihasilkan *throughput* rata-rata sebesar 3,623 Mbps.

Berdasarkan pengujian, hasil parameter *throughput* lebih besar dari 2,1 Mbps yang menjadi standar kategori sangat baik dengan indeks 4 pada standarisasi TIPHON seperti yang ditunjukkan pada tabel 1.

5. KESIMPULAN

Dari penelitian yang telah dilakukan dapat disimpulkan bahwa penelitian in telah berhasil dirancang, hal ini ditunjukkan dengan berhasilnya konektivitas antara raspberry pi sebagai *switch openflow* dengan OpenDayLight *controller*.

Selain itu didapatkan juga hasil performansi QoS dari pengujian dengan nilai parameter *delay* rata-rata sebesar 20,7 ms. *Delay* yang dihasilkan kecil dan dibawah 150 mS, hal ini dikarenakan panjang media fisik yang tidak terlalu panjang sehingga waktu untuk mengirimkan paket tidak terlalu besar dan *host* yang dipakai sedikit. *Delay* yang dihasilkan akan semakin besar seiring dengan bertambahnya variasi beban yang dikirimkan sehingga dibutuhkan proses pengiriman data yang lebih lama. Dari pengujian tersebut diperoleh nilai *delay* yang dikategorikan sangat baik dengan indeks 4.

Lalu untuk pengujian *jitter*, diperoleh *jitter* rata-rata sebesar 9,54 ms yang dikategorikan baik dengan memiliki indeks 3. Hal ini menunjukkan bahwa semakin besar variasi beban yang dikirim maka akan semakin besar pula *jitter* yang dihasilkan.

Lalu pada *throughput*, diperoleh hasil *throughput* rata-rata diatas rentang 2,1 Mbps *throughput* dan memiliki selisih yang sangat sedikit saat diberi variasi beban yang berbeda, hal ini dipengaruhi oleh jumlah *host* yang dipakai hanya berjumlah 2 buah, sehingga *traffic* jaringan yang terjadi tidak padat. Sehingga dapat dikategorikan sangat baik dengan indeks 4.

Selanjutnya yaitu *packet loss*, diperoleh hasil *packet loss* rata-rata sebesar 0,016 %. Pada pengukuran *packet loss*, saat ditambahnya variasi beban, *packet loss* yang dihasilkan akan semakin besar, hal ini membuktikan bahwa *packet loss* dipegaruhi oleh besarnya variasi beban yang dikirimkan. Berdasarkan pengujian ini, *packet loss* yang sihasilkan termasuk dalam kategori sangat baik dengan indeks 4. Hasil parameter QoS tersebut berada pada kategori baik menurut standarisasi TIPHON sehingga raspberry pi dapat digunakan sebagai pengganti *switch openflow*.

Adapun saran untuk penelitian selanjutnya yaitu dengan menambahkan jumlah *host* dan raspberry pi, menggunakan topologi dengan skala yang lebih luas sehingga dapat mendapatkan hasil pengujian dan pengukuran parameter *QoS* yang lebih kompleks. Untuk meningkatkan kehandalan *openflow switch* dapat dibangun dengan perangkat raspberry pi yang lebih baik spesifikasinya dibandingkan dengan versi yang penulis pakai.

6. UCAPAN TERIMA KASIH

Penulis ucapkan banyak terima kasih kepada Politeknik Negeri Bandung yang telah memberikan dana dan kesempatan kepada penulis untuk berpartisipasi di seminar IRWNS.

DAFTAR PUSTAKA

[1] "VMware, inc," Software-Defined Networking, [Online]. Available: https://www.vmware.com/topics/glossary/content/software-defined-networking.html. [Accessed 6 June 2022].

[2] I. Ummah and D. Abdillah, "Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking," *Indonesia Journal Of Computing*, vol. 1, no. 1, pp. 95-106, 2016.

[3] "Linux Foundation Collaborative," Open Vswitch, 2016. [Online]. Available: https://www.openvswitch.org/.. [Accessed 6 June 2022].

[4] "geeksforgeeks," Difference between Control Plane and Data Plane , [Online]. Available: https://www.geeksforgeeks.org/difference-between-control-plane-and-data-plane/. [Accessed 2 July 2022].

[5] D. S. Wijaksa, R. Mardiati, N. Ismail and T. Juhana, "Testbed Open vSwitch Raspberry Pi Pada Skala Kecil," *SENTER : Seminar Nasional Teknik Elektro*, 2016.

[6] T. I. Bayu, "Software Defined Network (SDN) Simulation Concept Using Raspberry Pi," *Jurnal Terapan Teknologi Informasi*, vol. 2, 2018.

[7] R. C. Kurniawan, R. Tulloh and D. I. Irawati, "VPLS on Software Defined Network Using ONOS Controller Based on Raspberry-Pi 3," *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pp. 19-24, 2021.

[8] R. Kartadie, E. Utami and E. Pramono, "Prototipe Infrastruktur Software-Defined Network Dengan Protokol Openflow Menggunakan Ubuntu Sebagai Kontroler," *JURNAL DASI*, vol. 15, 2014.

[9] N. F. Amirah and B. L. Ngah, "Development of SDN Controller Testbed Using Raspberry Pi 4," *International Journal of Synergy in Engineering and Technology 2*, p. 2021.

[10] M. Riadi, "KAJIANPUSTAKA.COM," Pengertian, Layanan dan Parameter Quality of Service (QoS), 26 Mei 2019. [Online]. Available: https://www.kajianpustaka.com/2019/05/pengertian-layanan-dan-parameter-quality-of-service-qos.html. [Accessed 6 Juni 2022].

[11] M. Riadi, "Raspberry Pi (Definisi, Fungsi, Jenis, Spesifikasi dan Pemrograman)," KajianPustaka.com, 17 December 2020. [Online]. Available: https://www.kajianpustaka.com/2020/12/Raspberry-Pi.html. [Accessed 20 March 2022].

[12] R. Muchlisin, "KAJIANPUSTAKA.COM," Pengertian, Layanan dan Parameter Quality of Service (QoS), 26 May 2019. [Online]. Available: https://www.kajianpustaka.com/2019/05/pengertian-layanan-dan-parameter-quality-of-service-qos.html. [Accessed 28 05 2022].