BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Terdapat penelitian yang membahas implementasi IPS berbasis *athena* untuk mencegah serangan DDoS pada arsitektur *Software Defined Network* (SDN) [5]. Pengujian dilakukan untuk mengetahui kinerja IPS menggunakan framework *Athena* sebagai pendeteksi dan pencegah serangan yang diterapkan pada *controller* ONOS. Hasil yang didapatkan pada penelitian menyatakan bahwa IPS mampu mencegah serangan DDoS dengan dibuktikan hasil *troughput* kembali ke keadaan normal. Dimana saat keadaan normal, diserang dan diterapkan IPS didapatkan nilai *troughput* transnmit sebesar 3956 pps, 4045 pps, dan 3919 pps dan receiver sebesar 4720 pps, 4793 pps, dan 4692 pps . Selain itu penerapan IPS berbasis *athena* juga mampu mengenali karakteristik host berbahaya

Terdapat penelitian yang membahas perancangan sistem keamanan jaringan berbasis *Software Defined Network* (SDN) menggunakaan *Intrusion Detection and Prevention System* (IDPS) [7]. Pengujian ini menerapkan sistem deteksi keamanan menggunakan *snort* dan IPTable dalam memblokir serangan pada jaringan SDN. *Controller* yang digunakan adalah ONOS dan penyerangan DDoS menggunakan bonet. Dari hasil pengujian serangan setelah diintegrasi IPS didapatkan hasil rata-rata QoS troughput sebesar 414,67 bps, *delay* 0.00771 ms dan packet loss 4,14%.

Terdapat penelitian implementasi *Intrusion Prevention System* (IPS) pada *Software Defined Network* (SDN) menggunakan *Ryu Controller* [8]. Pada pengujian ini menerapkan IPS untuk mencegah serangan *DoS Syin Flood* menggunakan aplikasi *snort* untuk memblokir serangan. Dari hasil implementasi penggunaan IPS terbukti efekti dalam mendeteksi serta memblokir serangan *Daniel of Service*. Didapatkan nilai QoS *throughput* sebelum serangan dengan rata-rata 22,536 Gb/s. Saat serangan sebesar 14,163 Gb/s dan kenaikan *throughput* setelah serangan sebesar 14,926. Namun kinerja CPU saat serangan *Syn Flood* dan

saat integrasi IPS hampir mencapai 100%. Hal ini dikarenakan sistem IPS sedang aktif dalam melakukan blokir serangan. Terdapat penelitian deteksi dan mitigasi serangan *Distributed Daniel of service* pada *Software Defined Network* [9]. Dimana pada penelitian ini dibuat sebuah sistem pendeteksi serangan berbasis *snort* IDS dan pencegahan serangan dengan mengimplementasikan iptable sebagai *firewall* pada *server*. Dari hasil pengujian implementasi *snort* pada SDN mampu mendeteksi serangan DDoS dengan akurasi 95% saat serangan *slowhttptest*, 90% serangan *slowris* dan 100% serangan LOIC. Selain itu implementasi *IPTables* pada SDN mampu memblokir serangan DDoS dan sistem juga mampu mengelola koneksi volume yang besar dan menjaga ketersediaan sistem SDN.

Terdapat penelitian yang membahas implementasi *Intrusion Prevention System* untuk mencegah serangan DDoS pada *Software Defined Network* [6]. Pada pengujian ini menggunakan metode *Intrusion Detection Prevention System* berbasis *snort* sebagai deteksi serangan dan *rest_firewall* dari *ryu* untuk memblokir serangan. Dari hasil implementasi tersebut berhasil memblokir serangan DDoS dengan mengirimkan 1000 paket TCP SYN *Flood*. Namun kinerja CPU meningkat saat terintegrasi IPS yaitu 21.5% untuk proses pengguna (us) dan 53.3% untuk besaran CPU yang digunakan sistem(sy). Serta penggunaan memori saat integrasi IPS lebih banyak dibandingkan saat sistem tidak terintegrasi IPS

Pada penelitian ini akan melakukan serangan DDoS pada jaringan SDN dengan menerapkan *Intrusion Prevention System* sebagai pendeteksi dan pencegah serangan. Berbeda dengan penelitian sebelumnya yang menggunakan *Athena* dan *snort* sebagai deteksi serta *IPTable* sebagai *Firewall*. Pada penelitian ini mengembangkan sistem deteksi *snort* yang dibuat terhubung dengan Telegram. Notifikasi telegram bertujuan agar administrator mudah dalam memonitoring *server* serta *Controller Ryu* akan diterapkan sebagai *Firewall* dalam memblokir serangan. Selain itu serangan yang akan digunakan adalah DDoS tipe TCP SYN *Flood* dan UDP *Flood*. Pengujian akan dilakukan dalam kondisi normal, saat dilancarakan serangan dan saat terintegrasi IPS. Untuk mengetahui kinerja dari 3 kondisi tersebut dapat dilihat dari data yang akan diambil berupa parameter nilai QoS *throughput*, CPU *Usage* dan Memori *Usage*.

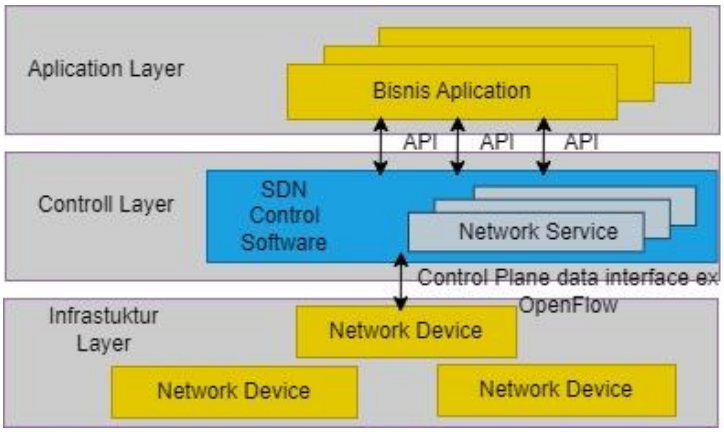
Tabel 2.1 Kajian Penelitian Sebelumnya

Year	***Author***	***Objective***	***Security system***	***Controller***	***Result***
2019	M. Farradhika, Primantara Hari dan Rakhmadhany. P	Analisa serangan DDoS menggunakan IPS berbasis *athena* pada SDN	IPS Berbasis *Athena*	*ONOS*	Integrasi IPS mampu mencegah serangan DDoS dengan dibuktikan hasil troughput kembali ke keadaan normal. Selain itu penerapan IPS berbasis *athena* juga mampu mengenali karakteristik host berbahaya
2021	Joni Padjr	Perancangan keamanan jaringan SDN menggunakan IDPS	IDPS *Snort* dan *IPTabel* sebagai *Firewall*	*ONOS*	Penerapan IDPS pada SDN dalam mencegah serangan DDoS mendapatkan hasil rata-rata QoS yang meningkat untuk troughput sebesar 414,67 bps dan penurunan *delay* 0.00771 ms serta packet loss 4,14%.
2021	M. Huda dan I Made Suartana	Implementasi IPS dalam mencegah DDoS pada jaringan SDN	IDPS *Snort* dan rest_*firewall Ryu*	*Ryu*	Integrasi IPS menjadikan jaringan normal kembali namun kinerja CPU meningkat saat terintegrasi IPS menjadi 21,5 % untuk pengguna (us) dan 53,3% untuk pengunnan sistem (sy)
2024	Dheni Yulia Dinda, Ronald Adrian	Melakukan deteksi dan mitigasi serangan DDoS pada SDN	IDS *Snort* dan *IPTabel* sebagai *Firewall*	*ONOS*	Implementasi IDS *snort* mampu mendeteksi serangan DDoS engan akurasi 95% saat serangan *slowhttptest*, 90% serangan *slowris* dan 100% serangan LOIC. Selain itu implementasi *IPTable* juga mampu memblokir serangan
2022	Bongga Arifwidodo, Bima Setyadi, Syariful	Implementasi IPS Pada SDN dalam mencegah serangan DoS TCP SYN *Flood*	IPS *berbasis snort*	*Ryu*	Implementasi IPS terbukti efektif dalam mendeteksi serangan dan memblokir serangan DoS dengan didapatkan hasil nilai QoS *Throughput* normal 22,536 Gb/s. Saat serangan sebesar 14,163 Gb/s dan kenaikan *throughput* setelah serangan sebesar 14,926. Namun kinerja CPU meningkat saat serangan dan terintegrasi IPS

2.2 DASAR TEORI

2.2.1. *Software Defined Network* (SDN)

Software Defined Network (SDN) merupakan paradigma baru dalam sistem jaringan komputer saat ini. *Software Defined Network* menawarkan konsep jaringan yang lebih efisien, fleksibel dan terpusat. Konsep SDN ini memisahkan *Control plane* dan Data *Plane* dalam jaringan. Dengan memisahkan *control plane* dan data *plane*. Logika kontrol akan diambil alih oleh *controller* sedangkan *switch* sebagai data *plane* akan bertindak sebagai forwarding device yang dihubungkan ke *controller* melalui protokol*OpenFlow*. Sehingga sentralisasi kendali jaringan dengan semua pengaturan berada pada *control plane* [9].



Gambar 2.1 Arsitektur SDN [10]

Terdapat 3 bagian yang menjadi arsitektur pada jaringan SDN yang ditunjukan pada Gambar 2.1 yaitu *aplication layer* sebagai aplikasi *service* yang dibutuhkan dalam jaringan SDN seperti *Intrusion Detection System* (IDS), *Intrusion Prevention System* dimana aplikasi ini sebagai penunjang kemanan dalam jaringan SDN yang terhubung menggunakan Rest API. *Control layer* merupakan bagian terpenting dalam arsitektur SDN karena pada layer ini yang bertugas mengatur jaringan, melakukan *controller* penuh pada jaringan dan memproses semua paket yang ada dalam jaringan tersebut. Selanjutnya adalah infrastuktur *layer* yang

merupakan sekumpulan perangkat *switch* atau network device yang bertindak sebagai forwarding device dimana *switch* tersebut akan terhubung dengan *controller* menggunakan protokolseperti *OpenFlow* untuk meneruskan paket ke yang masuk dan keluar dari host [10].

2.2.2. *Controller*

Controller adalah perangkat lunak yang berfungsi sebagai pengendali pada jaringan SDN. *Controller* bekerja berdasarkan protokolseperti *OpenFlow* yang memberitahu kemana paket akan dikirimkan. *Controller* akan bertanggung jawab dalam menentukan bagaimana menangani paket, mengelola table *flow* dan menentukan apa yang harus dilakukan pada paket baik itu mengizinkan *flow* mana yang boleh melewati jaringan atau *flow* mana yang tidak dizinkan melewati jaringan[11]. Macam – Macam *Controller* sebagai berikut:

2.2.2.1. Open Network Operating System (ONOS)

Open Network Operating System atau disingkat *ONOS* merupakan *controller* yang digunakan pada jaringan SDN bersifat *open source*. ONOS menggunakan bahasa pemograman Java. Dimana ONOS telah dilengkapi tampilan GUI (*Graphical User Interface*) yang memudahkan dalam mengontrol perangkat serta mendukung protokol*OpenFlow*. Selain itu ONOS juga dirancang untuk memenuhi kebutuhan operator dalam menerapkan jaringan yang dinamis serta atarmuka program yang sederhana [7].

2.2.2.2. Ryu

Ryu merupakan salah satu *controller* yang dirancang untuk meningkatkan kemampuan jaringan, memusatkan kontrol jaringan dalam mengatur ribuan perangkat jaringan. *Ryu* merupakan *controller* yang mendukung *Open Flow* dan bersifat *Open source*. Kata *Ryu* berasal dari bahasa Jepang yang berarti "Flow". Pada *Controller Ryu* menggunakan bahasa pemograman *python* yang mudah dalam pemakaian dan dokumentasi yang banyak sehingga mudah dalam menemukan solusi terhadap masalah. Selain itu *controller ryu* juga menyediakan banyak aplikasi yang dapat

diterapkan pada jaringan SDN salah satunya adalah *rest_firewall* yang berguna sebagai keamanan pada jaringan itu sendiri [12].

2.2.2.3. Floodlight

Floodlight Foodlight merupakan salah satu *controller* yang dikembangkan dari *controller* Beacon dan dan didukung oleh komunitas pengembang termasuk beberapa insiyur *Big Switch Network*. *Foodlight* merupakan *OpenFlow Controller* berlisensi Apache yang dirancang untuk berkerja dalam meningkatkan jumlah *switch*, router serta jalur akses yang mendukung standar *OpenFlow* [12].

2.2.2.4. Pox

POX Controller merupakan salah satu pengembangan perangkat lunak SDN berbasis *python*. Pox menyediakan cara yang efisien dalam mengimplementasikan protokol*OpenFlow*. Selain itu juga menyediakan banyak aplikasi yang dapat dijalankan dan diterapkan pada jaringan SDN seperti hub, *switch, load balancer* dan masih banyak lagi [13].

2.2.3. *Openflow*

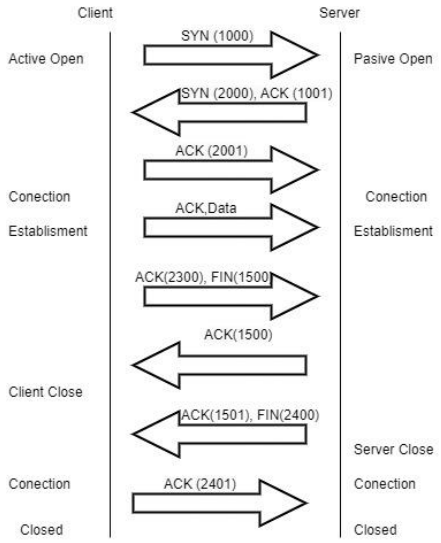
OpenFlow merupakan protokolutama dan salah satu jenis dari API yang digunakan pada jaringan SDN. Dimana protokol ini bertujuan memberikan jalur komunikasi antara *controller* dan infrastuktur *layer*. *OpenFlow* digunakan untuk mengontrol dan mengatur traffik *flow* pada *switch*, memberikan akses *forwarding* dari *switch* atau router menuju *controller*. *Switch* yang telah terpasang protokol *OpenFlow* akan mempunyai *flow-table* yang berisikan *flow-entity* sabagai rule dalam memproses dan meneruskan paket secara terpusat sekaligus sebagai koneksi dengan *controller* [10].

2.2.4. *Transmission Control Protocol* (TCP)

TCP merupakan protokolyang terletak pada layer transport dimana menyediakan *service* yang dikenal sebagai *connection oriented* dimana sebelum melakukan pertukaran data, dua aplikasi TCP harus melakukan pembentukan hubungan terlebih dahulu. Kemudian *realible* yang berarti TCP menerapkan proses

deteksi kesalahan paket yang diretransmisi. serta *Byte stream service* yang berarti paket akan dikirim dan sampai ketujuan secara berurutan [14].

Contoh sederhana pembukaan hubungan TCP antara *client* ke *server* ditunjukan pada Gambar 2.2 dimana untuk memulai pembukaan suatu hubungan, *client* harus terlebih dahulu mengirimkan paket SYN (*Synchronize*) ke *server*. Setelah *server* menerima paket tersebut maka paket akan dikirimkan kembali. SYN memiliki *acknowledge* (ACK) terhadap SYN sebelumnya. Ketika *client* menerima paket ini, maka akan *acknowledge* dan mengirimkan data miliknya. Sehingga pada saat itu terbentuklah koneksi TCP antara dua komputer yaitu *client* dan *server* [14].



Gambar 2.2 Pembentukan dan pemutusan hubungan TCP [14]

Angka dalam kurung yang mengikuti SYN pada gambar 2.2 adalah representasi dari *squence number* dimana nilai ini awalnya akan dihasilkan secara acak. Setiap ACK terhadap 1 paket harus diikuti *squence number* yang lebih tinggi dibandingkan *squence number* sebelumnya. Untuk pemutusan hubungan TCP antara kedua sisi harus mengirimkan paket yang berisi FIN (*Finish*) dimana paket ini harus di *acknowledge* lawan sebelum paket berakhir [14].

2.2.5. *USER DATAGRAM PROTOCOL* (UDP)

UDP memberikan suatu metode kepada aplikasi dalam mengirimkan data ke aplikasi di host lain tanpa harus membangun hubungan komunikasi dengan host tersebut (*connectionless*). Pada UDP tidak menjamin keberhasilan pengiriman data tersebut yang disebut sebagai datagram serta tidak menjamin adanya dumplikasi pengiriman. *Source* dan *destination* digunakan sebagai indentitas pengiriman dan dikarenakan UDP tidak memerlukan jawaban atau respon, maka *source* port sebenarnya tidak diperlukan. Sedangkan *detination* port adalah nomor yang akan dikenal oleh aplikasi di mesin remote yang akan dijadikan identitas layanan. Sebagai contoh FTP (*File Transfer Protocol*) menggunakan port 69 sebagai indentitasnya. Untuk mekanisme UDP dalam mendeteksi *error* pada pengirimaan data menggunakan *Checksum* [14].

2.2.6. Keamanan Jaringan

Kemananan jaringan menjadi perhatian utama saat kita ingin membangun sebuah jaringan. Pada dasarnya arsitektur jaringan menggunakan router dengan integrasi sistem *firewall* didalamnya. Selain itu dukungan *software* jaringan dapat memberikan kemudahan dalam mengontrol, memonitoring data paket dan penggunaan protokol yang diatur secara ketat. Selain itu kemananan jaringan juga dapat memabatasi folder dan file yang dapat dilihat oleh pengguna tertentu pada sistem jaringan itu sendiri. Salah Satu kemanan jaringan yang sering digunakan adalah *firewall* [15].

Firewall merupakan teknik yang sering diterapkan pada suatu jaringan. Hal tersebut dikarenakan sistem *Firewall* ini dapat diterapkan pada hardware, *software* maupun pada sistem itu sendiri. Pengunaan *firewall* ini bertujuan itu melindungi, menyaring, membatasi atau menolak suatu aktivitas dari jaringan pribadi dengan jaringan luar. *Firewall* atau dapat disebut tembok-api ini akan membatasi dan mengontroll akses terhadap user mana saja dapat mengakses jaringan pribadi dari

pihak luar. Sehingga penggunaan *firewall* sangat berguna dan penting dalam mengamankan jaringan [15].

2.2.7. *Intrusion Prevention System (*IPS*)*

Intrusion Prevention System (IPS) merupakan program keamanan jaringan yang digunakan untuk medeteksi serta memblokir serangan yang masuk pada jaringan. Fungsi utama dari IPS adalah mengidetifikasi aktivitas mencurigakan atau berbahaya , mencatat log semua paket yang teridentifikasi, melakukan pemblokiran terhadap *packet/traffic* yang berbaya serta melaporkan ke *network* admin. IPS dapat dianggap sebagai ekstensi IDS dikarenakan IPS mengkombinasikan *Firewall* dan metode *Intrusion Detection System* (IDS) dengan sangat baik. Sehingga dapat dikatakan IPS bertindak sebagai detektesi serta *firewall* dalam mengizinkan atau memblok paket yang tidak diiginkan [5].

2.2.8. *Intrusion Detection System* (IDS)

Secara analogi, *intrusion detection system* dapat disebut juga dengan alarm pencuri atau sebuah sistem keamanan yang melakukan proses deteksi serangan yang masuk pada jaringan. Fungsi utama IDS adalah mendeteksi dan memonitoring sebuah paket yang masuk melewati lalu lintas jaringan. Ketika sistem jaringan telah tertembus oleh serangan. Maka IDS akan memberikan pemberitahuan ke user atau sistem untuk mencegah serangan tersebut. Namun IDS hanya melakukan pendeteksi pada serangan saja. Mitigasi akan terjadi Apabila IPS dijalankan [16].

2.2.9. *Snort*

Snort merupakan sebuah perangkat lunak yang bersifat open *source* dan menjadi salah satu *tools security* terbaik dalam mendeteksi serangan pada jaringan. Ada berberapa keuntungan dari penggunaan *snort* diataranya kemudahan dalam melakukan konfigurasi, penambahan *rules* yang fleksibel serta kemampuan dalam

menganalisis data paket mentah menjadikan *snort* salah satu IDS terbaik. *Snort* berkeja dalam tiga mode diantaranya[17]:

1. *Snifer Mode : Snort* akan menangkap atau melihat semua paket yang melewati jaringan dimana *snort* diterapkan. Kemudian *snort* akan menampilkan hasil dari *sniffing* secara *realtime* namun dalam bentuk *console* [17].
2. *Packet Logger Mode*: Pada mode ini selain dapat melihat paket yang melewati jaringan, *snort* juga dapat mencatat log dari aktivitas yang melewati jaringan dan dapat disimpan pada storage *snort* dalam bentuk *output* file [17].
3. *Network Intrussion Detection System* (NIDS) mode: Mode ini dikenal degan modem IDS atau mode deteksi terhadap serangan. Mode ini akan menjalankan *snort* dengan konfigurasi pengguna yang telah ditentukan sebelumnya. File *snort*.conf akan dipanggil saat menjalankan *snort* mode IDS [17].

2.2.10. *Distributed Daniel Of Service* (DDoS)

Distributed Daniel Of Service (DDoS) merupakan salah saru jenis serangan yang digunakan penyerang untuk melumpuhkan target. DDoS memiliki konsep cara kerja yang sederhana yaitu dengan membanjiri *server* berupa paket data yang tidak berguna secara terus menerus. Membuat lalu lintas *server* berjalan dengan beban berat dan menghabiskan sumber daya sistem sehingga tidak mampu melayani *request* reguler dari user lain. Serangan DDoS berfokus pada tiga layer OSI. Diantaranya serangan pada network layer berupa *ICMP Floods* dan *Ping Of Death*. Kemudian serangan pada Transport layer berupa TCP SYN *Flood* dan UDP *Flood*. Dan serangan pada application layer berupa *Flood* HTTP *GET* [15]. Berikut beberapa pengertian dari serangan DDoS menurut Kumarasami (2021):

1. *TCP SYN Flood*: Merupakan type serangan protokol TCP diamana penyerangan akan mengirimkan paket SYN dalam jumlah yang besar secara terus menerus atau dalam waktu tertentu untuk membebani target sehingga tidak dapat merespon permintaan dari user lain dan menghabiskan sumber daya korban.

2. UDP *Flood*: Sama halnya denga SYN, pada UDP ini serangan akan memanfaatkan *service* pada protokol UDP dengan mengirimkan paket UDP palsu bervolume tinggi yang bertujuan sama yaitu melumpuhkan *server*/target.
3. *Ping (ICMP Flood):* Merupakan jenis serangan yang memanfaatkan protokol ICMP. Sama halnya dengan serangan sebelumnya. Serangan ini bertujuan menghabiskan sumber daya *server* atau target dengan mengirimkan *request* ICMP *Echo* atau yang dikenal dengan ping.

2.2.11. *Quality Of Service* (QOS)

Quality of Service atau sering disebut juga dengan QoS merupakan sebuah metode pengukuran yang digunakan untuk mengetahui kualitas suatu jaringan atau dapat didefinisikan karakteristik dan sifat dari suatu jaringan [18]. Pada penelitian ini pengkuran QoS dilakukan dengan pengambilan data *throughput* melalui *iperf*:

2.2.11.1. *Throughput*

Throughput adalah besaran kecepatan (*rate*) transfer data yang diukur dalam satuan (*bit per second*) dalam jangka waktu tertentu selama proses pengiriman suatu file. Dalam hal ini perhitungan *throughput* dapat dilakukan dengan mengitung total paket yang berhasil dikirimkan dalam interval waktu yang telah ditentukan. Adapun Klasifikasi standarisai nilai troughput berdasarkan TIPHON ditunjukan pada Tabel 2.1.

Tabel 2.2 Kategori *Throughput* berdasarkan TIPHON [18]

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

2.2.12. Memori

Memori atau dapat disebut dengan RAM (*Random Access Memory*) merupakan tempat penyimpanan data sementara. Dimana konsep kerja memori menyimpan data-data penting yang dibutuhkan *processor* dengan cepat untuk diolah menjadi informasi. Karena itulah kapsitas pada memori merupakan hal terpenting dimana semakin besarnya kapasitas pada memori kerja *processor* akan lebih cepat [19]

2.2.13. *Central Processing Unit* (CPU)

Central Processing Unit (CPU) merupakan perangkat keras komputer yang melaksanakan perintah dari data perangkat lunak atau *processor* (pengolah data). Aplikasi perangkat lunak yang dijalankan pada komputer biasanya akan memakan penggunaan CPU. Semakin besar suatu sistem yang berkerja pada perangkat tersebut maka penggunaan CPU juga akan semakin besar dan mengakibatkan perangkat lambat dalam merespon [19].

2.2.14. *Iperf*

Iperf merupakan salah perangkat lunak yang digunakan dalam dunia industri maupun ilmiah untuk melakukan pengunkuran kinerja suatu jaringan. Selain itu *iperf* merupakan alat yang dapat mengukur berbagai parameter perfomansi jaringan seperti kecepatan data transfer (*bandwidth*), *throughput* dan latensi. Adapun keuntungan menggunakan *iperf* diantaranya fleksibilitas dalam konfigurasi, bersifat *realtime*, pengukuran berbagai parameter dan uji kinerja satu arah [18].

2.2.15. *Top*

Top merupakan *software* yang berjalan pada ubuntu yang digunakan sebagai monitoring aplikasi yang berjalan baik dibelakang layar maupun yang sedang dijalankan *user*. Pada Top akan menampilkan nilai pengunaan CPU *usage* dan *system* serta penggunaan memori berupa memori yang tidak digunakan (*free*), memori *usage* dan memori *cache*.