BAB 1
USULAN GAGASAN

1.1 Deskripsi Umum Masalah

1.1.1 Latar Belakang Masalah

Pertumbuhan internet pada saat ini mengalami peningkatan, ditandai dengan semakin banyak orang yang menggunakan internet dalam keseharian mereka. Peningkatan ini didukung dengan munculnya teknologi baru seperti *cloud computing*, *Internet of Things*, 4G, dan 5G. Dengan pertumbuhan tersebut, permintaan layanan juga menjadi semakin beragam yang membuat terjadinya perubahan dalam persyaratan jaringan. Beberapa jaringan seperti *data center*, *cellular network,* dan jaringan yang mengimplementasikan konsep baru memerlukan kelincahan dan tingkat fleksibilitas yang lebih tinggi. Oleh karena itu, arsitektur *Software Defined Network* (SDN) digunakan untuk mendukung persyaratan jaringan yang baru dengan menyediakan jaringan yang bersifat cekatan dan fleksibel [1]. SDN diperkenalkan dengan konsep pemisahan antara *control plane* dan *data plane* dengan manajemen terpusat [2]. Kemudian, SDN mengalami pengembangan menjadi lebih canggih karena *data plane* yang ada pada SDN dapat dikonfigurasi dengan bahasa *Programming Protocol-independent Packet Processors* (P4) yang memungkinkan pengguna menulis protokol sesuai dengan kebutuhan, membangun *complex match/action pipeline,* serta menggabungkan fungsi yang berasal dari luar ke dalam kode [3].

Untuk memenuhi permintaan layanan yang semakin tinggi, *data center* dikembangkan untuk memberikan layanan dalam skala yang besar. Banyak perusahaan dan instansi yang menggunakan *data center* untuk mendukung operasional mereka, di antaranya Amazon, Google, Facebook, serta beberapa perusahaan Badan Usaha Milik Negara (BUMN) dan Badan Usaha Milik Swasta (BUMS). Dalam lingkup pendidikan, Universitas Telkom juga sudah memiliki *data center* yang digunakan untuk menyediakan layanan teknologi informasi yang handal bagi seluruh sivitas akademika. Layanan ini dijalankan di beberapa server aplikasi dan sistem *load balancing* mendistribusikan permintaan pengguna ke masing-masing server sesuai permintaan pelanggan dalam waktu yang bersamaan. Dalam situasi tersebut, jika server tidak mampu mengatasinya, beban server menjadi terlalu berat dan dapat menyebabkan server terkena masalah atau bahkan *crash* [4]. Masalah ini sangat dihindari karena *data center* memegang peran penting dalam operasional perusahaan atau instansi. Jika mengalami kegagalan, bisa menyebabkan gangguan layanan yang berakibat serius pada aspek operasional

hingga finansial. Oleh karena itu, *load balancer* dikembangkan untuk mendistribusikan *traffic* jaringan dengan seimbang dan memastikan penggunaan yang paling efisien agar tidak ada salah satu server atau sumber daya tertentu yang terlalu terbebani atau *overload*.

Penerapan *load balancing* yang ada pada perusahaan diawali dengan adanya ketersediaan aplikasi yang bisa diakses pengguna. Seiring dengan berjalannya waktu, jumlah pengguna semakin banyak dan perusahaan harus mempunyai sistem *load balancing* yang baik untuk membagi beban aplikasi agar tidak hanya bertumpu pada satu server saja. Di sisi lain, perusahaan tersebut juga perlu keamanan yang tinggi. Pada *load balancer* sering kali di dalamnya sudah ada *Web Application Firewall* (WAF) dan *Web Application and API Protection* (WAAP) untuk melihat apakah *traffic* yang mengarah ke aplikasi tidak berbahaya. Selain itu, *load balancing* juga bisa meningkatkan performa aplikasi karena ada sistem *caching*, sehingga *file* tidak langsung diakses ke server. Sistem *caching* membuat kinerja server menjadi lebih ringan. *Traffic* yang masuk juga bisa diarahkan ke server manapun sesuai dengan kebutuhan. Dengan menggunakan *load balancing*, ketika terjadi pemeliharaan, aplikasi tidak mengalami *downtime* karena dapat dihindari dengan adanya fitur *High Availability* (HA). Terakhir, dapat mengurangi biaya operasional karena memungkinkan *scaling* aplikasi secara horizontal. Dengan pengurangan biaya operasional, pengalokasian dana dapat digunakan untuk meningkatkan kinerja sistem dengan menambahkan lebih banyak server atau sumber daya untuk menangani permintaan yang tinggi.

1.1.2 Analisis Masalah

1.1.2.1 Aspek Keberlanjutan

Penggunaan *data center* mengalami peningkatan signifikan pada jumlah koneksi yang harus dikelola oleh server dan adanya peningkatan respons server oleh pengguna [4]. Pada arsitektur *data center*, *load balancer* memiliki peran penting dalam mendistribusikan *request* dari klien dan menjaga konsistensi per koneksi [5]. Namun, algoritma *load balancing* tradisional sering mengalami kegagalan menyeimbangkan *traffic* jaringan yang membeludak dan gagal mendeteksi adanya penghambatan data. Kegagalan traffic menimbulkan berbagai tantangan bagi *load balancer* di *data center*, termasuk kebutuhan akan deteksi penghambatan data yang *real-time* dan akurat, penanganan yang berbeda antara *flow* data pendek dan panjang, penanganan kegagalan *link*, dan ketahanan terhadap pengurutan ulang *Transmission Control Protocol* (TCP) [6].

Semakin banyaknya *request* yang ditangani oleh server akan meningkatkan kemungkinan terjadinya *overload* dan *collapse*. Permasalahan ini memotivasi perkembangan teknologi *load*

balancing [7]. Selama bertahun-tahun, banyak mekanisme telah diusulkan untuk mengatasi masalah ini, dengan fokus pada peningkatan *throughput*, *response time*, ketahanan, skalabilitas, dan efisiensi *resource*. Namun, setiap tujuan ini menghadirkan tantangan tersendiri dan membutuhkan pendekatan yang berbeda [6].

1.1.2.2 Aspek Teknologi

Programmable data plane menawarkan fleksibilitas dan skalabilitas yang lebih besar dibandingkan dengan teknologi SDN tanpa *programmable data plane* yang memiliki *fixed-function data plane*. SDN dengan *programmable data plane* memungkinkan pengguna untuk menulis protokol sesuai dengan kebutuhan mereka, membangun *complex match/action pipeline*, dan menggabungkan fungsi yang berasal dari luar ke dalam kode [8]. Salah satu keunggulan utama dari SDN dengan *programmable data plane* adalah kemampuannya untuk memeriksa dan memodifikasi *header* paket sesuai dengan kustomisasi pengguna. Selain itu, *programmable data plane* juga dapat menjadi solusi untuk meningkatkan pertahanan keamanan jaringan, seperti deteksi intrusi, enkripsi, dan mitigasi serangan *Distributed Denial of Service* (DDoS) [9].

Keunggulan-keunggulan tersebut dapat diimplementasikan berkat kemampuan pemrosesan pada *programmable data plane* dan teknologi penyaringan paket per paket. Oleh karena itu, keamanan jaringan dapat ditingkatkan oleh *programmable data plane* dengan memungkinkan peningkatan pada beberapa fungsi, seperti kontrol akses *traffic* jaringan masuk, enkripsi pada bagian *data plane*, serta peningkatan strategi pertahanan [9]. Selain itu, *programmable data plane* memungkinkan pengembang untuk mendefinisikan operasi pada *data plane* dan meningkatkan performa *load balancing* untuk memenuhi kebutuhan pada *data center* [7]. Dengan demikian, *programmable data plane* menawarkan solusi yang signifikan dan fleksibel untuk memenuhi berbagai kebutuhan dalam pengelolaan jaringan.

Pada SDN, fleksibilitas adalah kunci utama yang memungkinkan *resource* jaringan beradaptasi sebagai respons dari adanya perubahan topologi atau *network requirements*. Kemampuan beradaptasi ini semakin meningkat dengan munculnya *programmable data plane* yang menyediakan fitur *In-band Network Telemetry* (INT). INT pada *programmable data plane* dapat memberikan informasi tambahan, seperti jalur paket sehingga dapat membantu *troubleshooting* jaringan secara signifikan. *Programmable data plane* juga menawarkan fleksibilitas untuk memprogram *reactive logic* yang dapat dieksekusi dengan cepat pada *data plane* sehingga merevolusi kemampuan pengukuran dan analisis yang sangat detail pada *data plane* dengan kecepatan tinggi. *Programmable data plane* memudahkan pembuatan dan

penyesuaian struktur data khusus, seperti sketsa dan *bloom filter*, yang digunakan untuk mengumpulkan dan menganalisis metrik tertentu yang penting bagi pengguna atau administrator jaringan. Dengan fleksibilitas dan pengukuran statistik *traffic* jaringan dengan akurasi tinggi memungkinkan pemrogram untuk merespons secara *real time* jika terjadi suatu *event,* seperti kegagalan paket ketika melewati sebuah *threshold.* Keunggulan ini memungkinkan pengelolaan jaringan yang lebih responsif dan efisien [10].

Pada konteks *reliability*, *programmable data plane* dapat membantu pemulihan saat terjadi kegagalan jaringan. *Programmable data plane* dapat menyertakan jalur cadangan ke dalam *header* paket saat terjadi kegagalan *link* dan memanfaatkan informasi tersebut untuk memulihkan kegagalan *link.* Selain itu, teknologi ini juga dapat memastikan konsistensi dan akurasi data melalui teknologi pengolahan data tingkat rendah yang sistematis, yang memungkinkan rekonfigurasi langkah-langkah pengolahan jaringan. Dengan demikian, *programmable data plane* dapat membantu dalam memastikan bahwa data tetap akurat, dapat diandalkan, dan bebas dari kesalahan [11]. Perkembangan *programmable data plane* secara signifikan meningkatkan fleksibilitas dan skalabilitas SDN, serta *reliability,* sehingga membuka jalan untuk manajemen jaringan yang lebih efisien dan responsif.

1.1.2.3 Aspek Ekonomi

Algoritma *load balancing* yang kurang efektif dalam mengalokasikan beban dapat menyebabkan *resource* yang tersedia menjadi *overused* atau *underused.* Ketidakefektifan pengelolaan beban dapat menimbulkan biaya yang besar bagi *provider* telekomunikasi. Jika sebagian *resource* mengalami *overload* sementara yang lainnya *underutilized,* performa akan menurun dan *response time* akan meningkat. Penurunan performa dapat memberikan dampak negatif pada pengalaman *customer* dan berpotensi menyebabkan kehilangan *customer* [12]. Ketidakefektifan dalam *load balancing* juga dapat memengaruhi kualitas layanan (*quality of service)* yang diberikan oleh *provider*. *Overload* pada sebagian *resource* dapat menyebabkan gangguan pada *service* atau *downtime* yang dapat merusak reputasi *provider* dan menurunkan kepuasan *customer* [13].

1.1.2.4 Aspek Kebutuhan yang Harus Dipenuhi

Analisis aspek keberlanjutan, teknologi, dan ekonomi menunjukkan bahwa *load balancing* yang ideal untuk *data center* adalah *load balancing* yang dapat menangani *traffic* yang besar, mendistribusikan beban secara merata, beradaptasi dengan perubahan, meningkatkan performa, dan meminimalisir biaya. Implementasi algoritma *load balancing* yang diusulkan

pada tugas akhir ini harus memenuhi semua faktor ideal tersebut. Parameter yang akan ditingkatkan untuk mendapatkan hasil tersebut adalah *throughput*, *response time, request loss*, distribusi *request* dan tingkat kegagalan.

1.1.3 Tujuan Capstone

1) Menghasilkan performa *load balancing* yang lebih baik untuk *data center* menggunakan SDN-*programmable data plane* dengan fokus pada algoritma Resource Based, Round Robin, dan Weighted Round Robin.
2) Melakukan pembuktian solusi performa *load balancing* dengan dua tahapan: *performance test* dan *fault tolerant test.*

1.2 Analisis Solusi yang Ada

Ke, Chih-Heng dkk. mengembangkan teknologi *load balancing* dengan memanfaatkan bahasa pemrograman P4 untuk menyeimbangkan beban pada SDN. Pada pengembangan ini, permasalahan pada server dapat diidentifikasi oleh *controller* dan memberi peringatan ke sistem *load balancer* P4 untuk memblokir permintaan. Pengidentifikasian oleh *controller* dilakukan untuk menghindari server yang tidak berfungsi dan mengurangi kegagalan pengiriman paket. Meskipun begitu, kode algoritma yang digunakan tidak dapat dikonfigurasi ulang dan sistem *load balancer* dapat berhenti bekerja saat terjadi kegagalan pada *controller* [7].

Das, Rohit Kumar dkk. mengusulkan arsitektur *fault tolerance* terdistribusi untuk SDN yang disebut FT-SDN. FT-SDN tetap bertahan dengan baik walaupun terjadi kegagalan pada *control plane*. Namun, arsitektur tersebut tidak dibahas pengimplementasiannya pada jaringan nyata *data center* [14].

Neghabi, Ali Akbar dkk. mengemukakan tiga jenis algoritma, yaitu statis, dinamis, dan proaktif. Ketiga jenis algoritma tersebut dapat menyesuaikan masing-masing *traffic* mulai dari *traffic* yang tidak berubah-ubah, berubah-ubah, dan sangat fluktuatif. Walaupun demikian, penelitian ini terlalu fokus pada produk dan solusi dari vendor tertentu. Paper ini menyebutkan beberapa produk dan solusi suatu vendor untuk *load balancing* SDN, seperti OpenDaylight, ONOS, dan Ryu [15].