BAB I Pendahuluan

1.1. Latar Belakang

Pada masa sekarang ini jaringan telah menjadi infrastruktur yang sangat penting bagi bisnis, perkantoran maupun kampus. Jaringan lokal atau sering disebut *Local Area Network (LAN)* juga sangat cocok untuk diimplemetasikan pada area tersebut. Oleh karena itu, sangat penting sekali untuk memiliki sistem infrastruktur jaringan yang mudah dikelola namun memiliki kemampuan kerja yang optimal dan juga fleksibel.

Namun jaringan komputer merupakan sesuatu yang kompleks dan sukar utuk diatur. Banyak peralatan yang dibutuhkan untuk merancang dan mengatur jaringan komputer, dari *router* dan *switch* hingga ke *middlebox* seperti *firewall, translator* alamat jaringan, penyeimbang beban server (*load balance*), dan lainnya. Tentu saja *routers* dan *switch* bekerja dengan sistem yang kompleks, perangkat lunak kontrol didistribusi secara khusus, tertutup dan merupakan hak milik perusahaan perangkat keras yang digunakan. Perangkat lunak yang digunakan juga mengimplementasikan protokol jaringan yang dijalankan distandarisasi dan diuji coba beberapa tahun jauh kebelakang. *Adminsitrator* jaringan secara khusus mengkonfigurasi perangkat jaringan menggunakan antarmuka konfigurasi yang *vendor-based*, yang berarti setiap perangkat jaringan memiliki konfigurasi berbeda tiap vendor, bahkan banyak produk dengan satu vendor yang memiliki antarmuka konfigurasi yang berbeda[10].

Sistem jaringan seperti ini dioperasikan dengan protokol, mekanisme dan antarmuka konfigurasi yang individual. Metode pengoperasian seperti ini mengurangi inovasi, meningkatkan kompleksitas dan menjadikan biaya operasional melambung dalam menjalankan jaringan[10].

Berdasarkan fakta-fakta yang telah disebutkan, arsitektur jaringan seperti itu sudah tidak lagi cocok dengan kebutuhan *enterprise, carriers* dan *end user* saat ini[4]. Oleh karena itu *Software Defined Network* (SDN) mengubah cara jaringan didesain dan diatur.

SDN memiliki dua karakteristik. Pertama, SDN memisahkan *control plane* (yag memutuskan bagaimana mengatasi lalu-lintas) dengan *data plane* (yang mem-*forward* lalu-lintas berdasarkan keputusan yang dibuat *control plane*). Kedua, SDN mengkonsolidasikan *control plane*, sehingga satu perangkat lunak mengendalikan beberapa elemen kontrol program *data-plane. Control-plane* SDN menggunakan kendali langsung terhadap keadaan di dalam

elemen *data-plane* jaringan (seperti *router*, *switch*, dan *middlebox)* melalui API yang terdefinisi[10].

Salah satu contoh API tersebut adalah *OpenFlow*. Sebuah *switch OpenFlow* memiliki satu atau lebih tabel aturan *packet-handling.* Setiap aturan mencocokan dengan bagian dari lalu-lintas dan menjalankan aksi tertentu pada lalul-lintas yang memiliki kecocokan dengan aturan tersebut. Aksi tersebut bisa berupa *dropping, forwarding,* atau *flooding.* Tergantung pada aturan yang terpasang pada aplikasi kendali, sebuah *switch OpenFlow* dapat menjadi *router, switch, firewall, network adress translator,* atau yang lainnya[10].

Pada *OpenFlow*, modul pengendali dilepaskan dari perangkat dan ditambahkan ke *web server* di luar perangkat. *Web server* menjalankan program pengendali utuk mengirim instruksi kepada *switch* dan mendesain aturan *forwarding.* Program pengendali juga dapat menyediakan antarmuka yang dapat diprogram bagi *manager* untuk bertanggung jawab pada *switch* dan mengurangi proses konfigurassi manual[11].

Salah satu program pengendali pada *OpenFlow* adalah *Floodlight Controller*. *FloodLight* mengadopsi arsitektur modular untuk mengimplementasi fitur pengendali dan beberapa aplikasi nya. Floodlight menyediakan dua bagian fungsi utama. Pertama, untuk menangani koneksi ke *switch* dan mengarahkan pesan *OpenFlow* kepada *event* modul lain yang dapat mendengarnya. Kedua, *floodlight* memutuskan urutan pesan OpenFlow khusus yang dikirim ke modul yang sedang mendengar. Sehingga modul dapat memutuskan untuk mengijinkan pemrosesan pesan untuk diteruskan ke *listener* selanjutnya atau tidak[11]. Modul yang dibutuhkan oleh pengguna *enterprise* dan *end user* salah satunya adalah modul *quality of service (QoS)* dan *firewall.*

Namun untuk mengoperasikan *floodlight controller* harus melalui sebuah API dimana untuk mengaksesnya harus melakukan pemrograman. Hal ini menyebabkan penggunaan *controller* tersebut menjadi cukup sulit.

Berdasarkan pemaparan di atas, SDN dapat mengurangi kompleksitas dan memudahkan pengaturan jaringan komputer. Pengimplementasian SDN pada jaringan komputer memberikan efisiensi yang cukup besar baik dari sisi penggunaan maupun biaya.

Namun karena pengembangan pengendali berbasis modular seperti *floodlight* lebih banyak pada sisi modul, sehingga penggunaannya oleh *end user* cukup sulit dikarenakan untuk mengoperasikannya harus melalui *rest* API. Oleh karena itu dibutuhkan pengembangan suatu

sistem berbasis antar muka grafis yang dapat digunakan dengan mudah oleh *end user* untuk mengkonfigurasi SDN. Sehingga diajukan sebuah tugas akhir berupa "Rancang Bangun Sistem Antar Muka Grafis *Floodlight* sebagai *Controller* pada *Software Defined Network* (SDN)".

1.2. Rumusan Masalah

Dalam penyusunan laporan ini akan dibahas beberapa permasalahan antara lain:

1. Bagaimana rancang bangun sistem Antar Muka Grafis *Floodlight* sebagai *Controller* pada *SDN* yang *user-friendly* bagi *end-user*.
2. Bagaimana implementasi fungsionalitas sistem Antar Muka Grafis *Floodlight* yang telah dibangun sebagai *Controller* pada SDN sehingga mengurangi kompleksitas pengaturan jaringan komputer.

1.3. Tujuan dan Manfaat

1.3.1. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

1. Membuat sistem Antar Muka Grafis *Floodlight* menggunakan bahasa *Java* sebagai antar muka controller jaringan berbasis *SDN* agar *SDN* dapat digunakan oleh *end-user*.
2. Menguji fungsionalitas sistem Antar Muka Grafis *Floodlight* yang telah dibuat pada simulasi jaringan berbasis SDN agar dapat dipastikan bahwa *SDN* dapat mengurangi kompleksitas pengaturan jaringan komputer.

1.3.2. Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Manfaat Akademis

   Penelitian ini erat hubungannya dengan mata kuliah jaringan komunikasi dan pemrograman sehingga diharapkan mampu menjadi referensi atau bahan kuliah untuk mata kuliah yang disebutkan. Dan juga penelitian ini dapat bermanfaat bagi pengembangan atau penelitian selanjutnya tentang *Software-Defined Network.*

2. Manfaat Praktis

   Penelitian ini dapat bermanfaat bagi pengguna infrastruktur jaringan untuk mengakses *Floodlight* sebagai controller yang berfungsi untuk mengelola lalu-lintas jaringan berbasis *Software-Defined Network.*

1.4. Batasan Masalah

Diperlukan batasan masalah dalam simulasi sistem kendali lalu-linitas jaringan dengan SDN ini sehingga dapat diperoleh hasil yang sesuai dengan tujuan pembuatan serta membatasi masalah yang akan dibahas. Adapun pembatasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem yang dibuat adalah sistem Antar Muka Grafis *Floodlight*
2. Sistem yang dibuat diimplementasikan pada jaringan *SDN* dengan simulasi *Local Area Network* yang dibangun oleh simulator jaringan bernama *mininet.*
3. *Controller* yang digunakan adalah *floodlight* + modul *QoS* versi *beta*.
4. Antar muka yang dibuat adalah antar muka untuk modul *QoS DSCP*. *QoS Queue, Firewall* dan *Controller Monitoring.*
5. Topologi jaringan yang digunakan adalah topologi *Linear*
6. Bahasa pemrograman yang digunakan pada aplikasi adalah bahasa java.
7. Pengujian dan analisis dilakukan terhadap keseluruhan fungsionalitas sistem dan kinerja modul *QoS Queue* dan *Firewall.*

1.5. Posisi Penelitian (State of the Art)

State of The Art merupakan pernyataan yang menunjukan bahwa penyelesain masalah yang diajukan merupakan hal yang berbeda dengan penelitian yang telah dilakukan pihak lain. Dalam bagian ini akan diuraikan secara singkat penelitian terdahulu yang dapat memperkuat alasan mengapa penelitian ini akan dilakukan. Adapun State of The Art penelitian dijabarkan pada bagan berikut ini :

Tabel 1.1 Riset sejenis/terdahulu

Judul	Peneliti	Konsep Model
OpenFlow: Enabling Innovation in Campus Networks	Nick McKeown, Guru Parulkar – Stanford University; Larry Peterson, Jennifer Rexford – Princeton University; Tom Anderson, Jonathan Turner – Uvicersity of Washington; Hari Balakrishnan – MIT, Scott Shenker – University of California 2008	Menggunakan metode *programmable* untuk diimplementasikan pada jaringan
		Protokol *OpenFlow* untuk menjadikan sebuah jaringan yang *programmable* (SDN)
		Dengan melihat sejarah dari ide awal hingga pengembangan terbaru. Lalu

Judul	Peneliti	Konsep Model
A Survey of Past, Present and Future of Software Defined Networking	Pritesh Ranjan, Pankaj Pande, Ramesh Oswal, Zainab Qurani, Rajneeshkaur Bedi - Computer Department MIT college of Engineering Pune – India 2014	digambarkan secara khusus dan detail terkait arsitektur *SDN* dan standar OpenFlow
		Dihasilkan sebuah *overview* dari arsitektur SDN dan penggunaannya masa kini dan yang akan datang
An SDN Approach: Quality of Service using Big Switch's Floodlight Open-source Controller	Ryan Wallner, Robert Cannistra. Marist College 2013	Menggunakan metode *DSCP* dan *Queue*
		Dihasilkan modul pada *controller* untuk menangani *QoS*
SDN-based Security in Virtualized Environments for Cloud Computing	Youngsang Shin, Kyungho Son, Haeryong Park Information Security Group, Korea Internet & Security Agency, Seoul, South Korea 2014	Menggunakan model kooperatif antara keamanan berbasis SDN dan keamanan berbasis virtual
		Dihasilkan sistem keamanan berbasis SDN pada lingkungan virtual untuk *cloud computing*

Pada paper dengan judul *"OpenFlow: Enabling Innovation in Campus Networks"* dihasilkan sebuah protokol jaringan yang disebut *OpenFlow*. *OpenFlow* ini merupakan suatu protokol yang bekerja pada *SDN* dan dapat mengurangi kompleksitas dan kesukaran manajemen pada jaringan komputer. Penggunaan SDN akan semakin banyak seiring bertambahnya pengembangan-pengembangan yang dilakukan dan pengaplikasian yang sangat luas seperti yang disajikan pada paper berjudul "*A Survey of Past, Present and Future of Software Defined Networking*". SDN memisahkan *control-plane* dengan *forwarding-plane* sehingga pengendalian lalu-lintas jaringan dapat dilakukan secara terpusat menggunakan *controller*. Berdasarkan dua paper diatas, dapat disimpulkan bahwa *SDN* dapat menggantikan jaringan tradisional yang telah digunakan.

Kemudian paper berjudul *"An SDN Approach: Quality of Service using Big Switch's Floodlight Open-source Controller"* menghasilkan sebuah modul yang dapat mengontrol *Quality of Service* pada SDN. Lalu paper berjudul *"SDN-based Security in Virtualized Environments for Cloud Computing"* menghasilkan sistem keamanan berbasis SDN. Modul-modul tersebut dapat diimplementasikan pada sebuah *controller* yang bersifat modular seperti *Floodlight* yang akan digunakan pada pengembangan sistem ini.

Dari penelitian yang dipaparkan diatas, dapat disimpulkan bahwa *SDN* sudah siap untuk diimplementasikan. Namun pengembangan *SDN controller* seperti *Floodlight* selama ini lebih terfokus pada riset sehingga menghambat pengimplementasian karena sulit unutk dioperasikan oleh *end-user*. Oleh karena itu pengembangan sistem antarmuka grafis *Floodlight* ini sangatlah diperlukan agar implementasi *SDN* dapat dilakukan. Dan juga karena pengembangan sistem ini yang lebih fokus terhadap *end user* maka penelitian ini mengandung kebaruan dan tidak menjiplak dari penelitian-penelitian atau pengembangan-pengembangan sebelumnya.

1.6. Kerangka Pemikiran