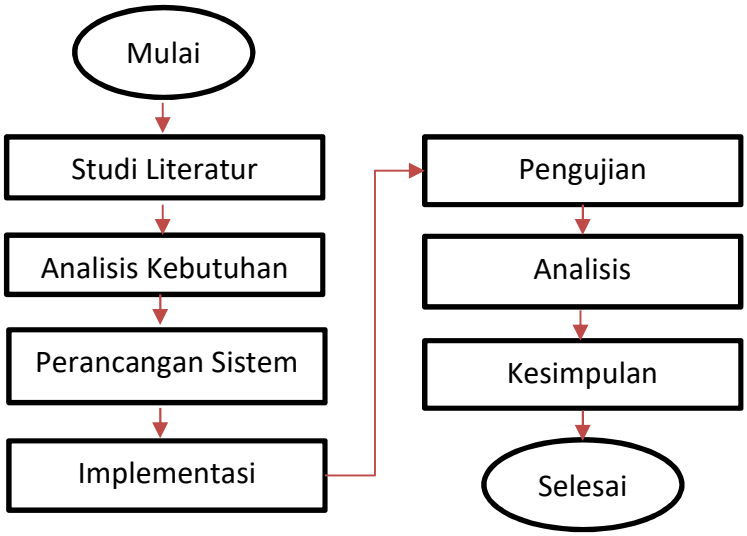
BAB 3 METODOLOGI

Pada bagian ini dijelaskan tentang studi literatur, analisis kebutuhan yang terdiri dari kebutuhan fungsional dan non-fungsional, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan. Berikut merupakan tahapan-tahapan metodologi penelitian yang tergambar pada diagram alir yang ditunjukkan gambar 3.1



Gambar 3. 1 Diagram Alir Metode Penelitian

Berdasarkan Gambar 3.1, maka dapat dijelaskan langkah-langkah metodologi penelitian yang digunakan, yaitu:

- Melakukan Studi literatur penelitian yang mencakup, *Software Defined Network, Openflow, Ryu Controller, Load Balancing, dan Mininet.*
- Analisis kebutuhan sistem meliputi kebutuhan fungsional dan non fungsional.
- Perancangan sistem load balancing dengan algoritme *shortest delay* pada *software defined network*
- Implementasi SDN menggunakan mininet.
- Pengujian sistem *load balancing pada software defined network, algoritme shortest delay dan algoritme round robin.*
- Analisis dan hasil pengujian dengan parameter *delay.*
- Penarikan kesimpulan berdasarkan hasil analisi pengujian yang dilakukan terhadap sistem.

3.1 Studi Literatur

Pada bab ini membahas tentang dasar teori yang mendukung kebutuhan terhadap perancangan dan implementasi *load balancing* pada *software defined network* dengan algoritme *shortest delay.* Berbagai macam studi literatur yang dilakukan membahas dasar-dasar SDN dan *Openflow* berserta aplikasi dan penerapannya. Kemudian dilakukan studi literatur tentang penerapan algoritme *shortest delay*, lalu dilakukan studi literatur tentang emulator dan penerapannya serta *Ryu Controller* sebangai pengembangan aplikasi-aplikasi SDN. Studi literatur yang terakhir dilakukan untuk mengetahui parameter uji yang cocok digunakan pada penelitian ini.

3.2 Analisis Kebutuhan

Metode analisis kebutuhan ditujukan untuk menganalisis beberapa kebutuhan sistem yang diperlukan pada penelitian ini. Kebutuhan pada sistem ini terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional.

3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang dapat dilakukan oleh sistem, informasi apa yang harus ada dan bisa di hasilkan oleh sistem. Kebutuhan fungsional meliputi sebagai berikut:

1. Sistem dapat memantau delay dari masing-masing server.
2. Sistem dapat membagi beban server berdasarkan algoritme *shortest delay.*
3. Sistem dapat menghasilkan nilai yang dapat diukur yang nantinya akan dibandingkan sesuai parameter uji yang telah ditentukan.

3.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan kebutuhan *hardware* maupun *software* yang digunakan untuk membangun sistem sesuai yang diharapkan pada penelitian ini. Berikut kebutuhan non-fungsional antara lain:

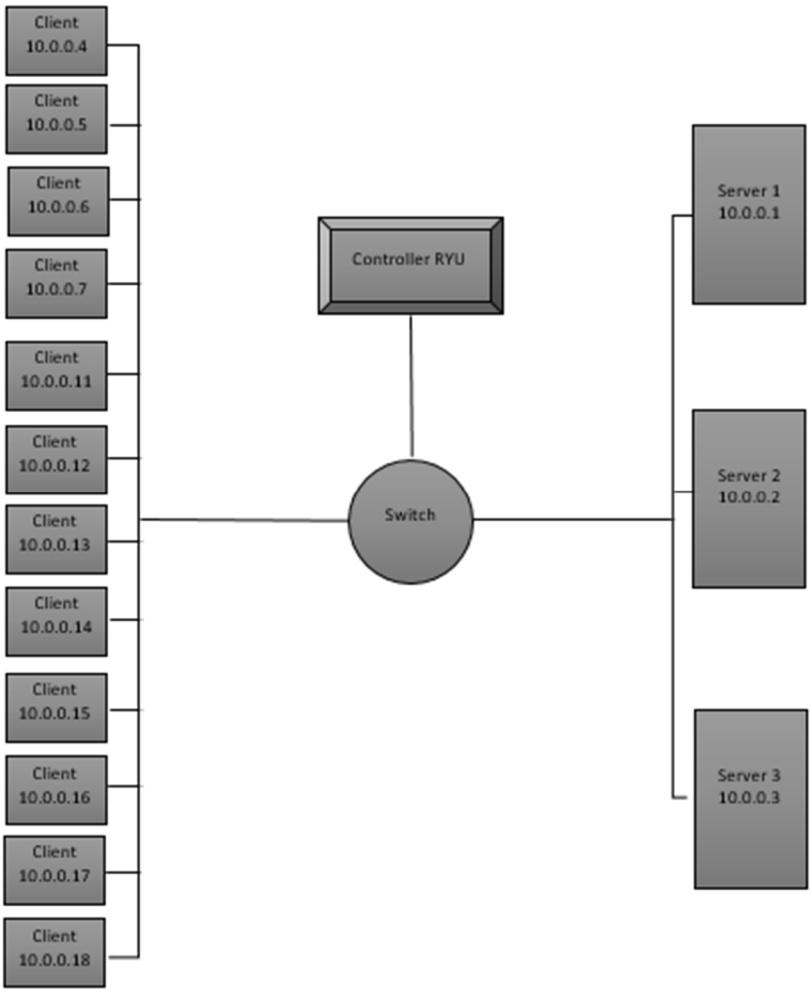
a. Kebutuhan perangkat keras (Hardware):
   - *Processor : Intel ® Core ™ i3-2330M CPU @ 2.20Ghz*
   - *Memory: 4096MB RAM*
   - *Operating Sistem : Windows 10 64-bit*

b. Kebutuhan perangkat lunak (software):
   - *Operating sistem Ubuntu 16.04 64bit = sebagai sistem operasi*
   - *Mininet emulator = emulator pembangun topologi*
   - *Ryu Controller software defined network = sebagai Controller pada Mininet*
   - *Bahasa pemrograman python*

- *Iperf, httperf = tool alat ujii*

3.3 Perancangan Sistem

Perancangan sistem dapat di artikan sebagai tahapan untuk membangun sistem setelah analisis kebutuhan sistem telah terpenuhi. Berikut langkah-langkah perancangan sistem dapat dilihat pada blok-diagram sebagai berikut:



Gambar 3. 2 Perancangan Sistem

Pada Gambar 3.2 merupakan struktur perancangan sistem yang digunakan pada mininet *server* menggunakan tiga *server* virtual yang berkomunikasi pada port 80. Controller memegang control penuh pada penelitian ini, yaitu sebagai monitor *delay*. Cara memonitoring *delay* dengan cara pengiriman paket. Ketika client mengirimkan paket, switch bertugas meneruskan paket tersebut ke Controller untuk selanjutnya diberikan action terhadap paket tersebut. Lalu, Controller mengalokasikan paket tersebut ke server dengan delay terendah. Perancangan sistem ini menggunakan 16 *host* dengan pembagian 13 *host* bertindak sebagai *client* dan 3 *host* berindak sebagai *server*. Perancangan sistem dilakukan dengan membandingkan antara algoritme *shortest delay* dan algoritme *round-robin*

3.4 Implementasi

Implementasi sistem dilakukan dengan dasar acuan kebutuhan dan perancangan sistem yang telah di buat sebelumnya. Maka, penerapan implementasi meliputi:

a. Instalasi Linux Ubuntu 16.04 64bit
b. Instalasi *Mininet, Miniedit dan Ryu Controller*
c. Membangun topologi jaringan SDN pada *Mininet dan Miniedit*
d. Membangun sistem load balancing
e. Membuat *algoritme shortest delay* sebagai mekanisme pemilihan server untuk membagi beban server.
f. Membandingkan dengan salah satu algoritme *load balancing* seperti *algoritme round robin*.
g. Melakukan analisis perbandingan dari hasil kedua algoritme tersebut berdasarkan parameter uji yang telah ditentukan.

3.5 Pengujian dan Analisis

Bab ini berisi tentang langkah-langkah untuk pengujian sistem dan analisis berdasarkan implementasi yang telah dilakukan dengan menguji apakah kinerja sistem telah sesuai. Pengujian akan dilakukan dengan cara mensimulasikan algoritme *shortest delay dan round robin.* Pengujian dilakukan terfokus pada pengujian terhadap fungsional, apakah sistem dapat berjalan sesuai dengan kebutuhan fungsional pada perancangan atau tidak. Adapun skenario yang dilakukan untuk melakukan pengujian sistem untuk performa *web server*, yaitu:

1. Melakukan request dengan 12 client secara bersamaan dengan memberikan beban traffic ke server dengan 24 *request/second*, 48 *request/second* dan 72 *request/second.*
2. Melakukan *request* dengan 12 client secara bersamaan.
3. Melakukan analisis dengan parameter yang digunakan, seperti *throughput, connection rate, reply time dan error*.

3.6 Pengambilan Kesimpulan

Pengambilan kesimpulan berdasarkan setelah melakukan semua tahapan perancangan, implementasi, pengujian dan analisis sistem yang telah selesai dilakukan. Kesimpulan diambil dari pengujian dan analisis sistem berdasarkan perbandingan antara algoritme *shortest delay* dan *round robin* untuk pedistribusian beban ke server pada jaringan Openflow. Pada Tahap akhir penulisan ini bisa ditambahkan dengan memberi masukan dan diharapkan dapat digunakan pada penelitian selanjutnya.