BAB I

PENDAHULUAN

1.1 Latar Belakang

Secara umum, Internet dapat didefinisikan sebagai sebuah jaringan komputer dengan skala yang besar (Stringer, 2005). Internet terbentuk dari komputer-komputer di seluruh dunia, yang terhubung melalui jaringan-jaringan kecil. Jaringan-jaringan kecil ini terhubung satu sama lain sehingga membentuk sebuah jaringan yang lebih besar. Semua jaringan ini pada akhirnya akan terhubung antara yang satu dengan yang lainnya, membentuk sebuah jaringan yang sangat besar yang kita sebut dengan "Internet".

Internet telah merevolusi dunia komputer dan komunikasi dengan kemampuannya untuk menyebarkan informasi ke seluruh dunia dalam waktu yang singkat (Leiner, et al., 2009). Dengan kemampuan yang dimiliki oleh Internet, informasi menjadi sangat mudah didapatkan. Berbagai macam jenis informasi dapat diperoleh dengan mudah, mulai dari hal-hal seperti video musik, berita internasional, buku komik, hingga berbagai macam tutorial, materi pelajaran, dan karya ilmiah.

Agar komputer-komputer yang terdapat dalam jaringan-jaringan yang berbeda dapat saling terhubung antara yang satu dengan yang lainnya, dibutuhkan alat yang bernama *router*. *Router* adalah salah satu komponen penting dalam jaringan komputer, yang bertugas untuk meneruskan paket data yang diterimanya

ke jaringan di mana *host* tujuan dari paket data tersebut berada (Cole, Krutz, & Conley, 2005).

Untuk dapat menyampaikan paket data ke jaringan yang benar, *router* perlu mengetahui di jaringan mana *host* tujuan dari paket data tersebut berada. Karena itu, *router* perlu bekerja sama dengan *router-router* lain yang terhubung ke jaringan yang sama dengan *router* yang bersangkutan. *Router-router* tersebut dapat berkomunikasi antara yang satu dengan yang lainnya dengan menggunakan *routing protocol*. Secara umum, algoritma *routing protocol* dapat dibagi menjadi dua kelompok: *distance vector routing protocol* dan *link-state routing protocol*.

Dalam jaringan komputer, terdapat *server* yang berfungsi sebagai perantara antara *host client* yang mengakses layanan dengan *host server* dari layanan tersebut. *Server* yang bertugas menjadi perantara ini disebut *proxy server*. Kecepatan akses dari layanan-layanan yang diakses dari sebuah instansi melalui Internet dapat ditingkatkan dengan memanfaatkan sebuah *proxy server* yang berfungsi untuk menyimpan *cache* dari layanan-layanan yang seringkali diakses oleh pengguna jaringan milik instansi tersebut.

Dengan adanya sebuah *proxy server* yang melakukan *caching*, kecepatan akses layanan dapat ditingkatkan. Data-data yang dibutuhkan untuk layanan tersebut, yang seharusnya diambil dari *server* penyedia layanan, dapat dibuatkan *cache* di dalam *proxy server* sehingga data-data tersebut tidak perlu diambil dari *server* penyedia layanan. Dengan demikian, kecepatan akses layanan akan meningkat karena berkurangnya ukuran data yang perlu dikirimkan oleh *server* penyedia layanan ke *client* dan data-data yang sudah tersimpan dalam *cache* milik

proxy server akan diberikan kepada *client* langsung dari *cache* (Dykes & Robbins, 2001).

Kegunaan lain dari *proxy server* di antaranya yaitu memberikan anonimitas kepada *client* pada saat mengakses suatu *server*. Pada saat *client* mengakses *web server* melalui *proxy server*, yang akan tercatat sebagai *host* pengakses dalam *log* milik *web server* adalah *proxy server* yang digunakan, bukan *client* yang sebenarnya. Hal ini juga berlaku sebaliknya, pemilik layanan dapat menempatkan sebuah *proxy server* yang harus diakses oleh *client* yang ingin mengakses layanan yang terdapat dalam *server* mereka. Dengan demikian, *client* tidak perlu mengetahui alamat dari *server* yang menyediakan layanannya dan keamanan dari *server* tersebut menjadi lebih baik (Wang & Chien).

Untuk menggunakan *proxy server*, umumnya pengguna dari *host client* harus melakukan konfigurasi dan menetapkan *proxy server* yang akan digunakan oleh *host*. Jika terdapat beberapa *proxy server* yang bisa digunakan oleh *client*, maka pengguna harus memilih yang terbaik di antara *proxy server* yang tersedia. Agar dapat memilih dengan baik, pengguna harus mencoba menggunakan setiap *proxy server* untuk mengakses *server* yang diinginkannya dan menilai performa dari masing-masing *proxy server* untuk menentukan *proxy server* manakah yang terbaik. Proses pengujian *proxy server* ini dapat dilakukan oleh sebuah program yang berfungsi sebagai *proxy selector*.

Proxy selector dapat melakukan pengujian *proxy server* dan mencatat performanya dengan akurat, dibandingkan dengan percobaan yang dilakukan manusia. Kecepatan akses dapat dihitung dengan tepat dan dapat dibandingkan

dengan performa dari *proxy server* lainnya oleh *proxy selector*, sehingga *proxy selector* dapat membantu pengguna untuk memilih *proxy server* yang terbaik berdasarkan perhitungan tersebut.

Pada saat ini contoh program *proxy selector* yang cukup populer adalah sebuah *add-on* untuk *web browser* Mozilla Firefox yang bernama Proxy Selector (Proxy Selector Add-On, 2013). *Add-on* Proxy Selector ini tidak memiliki fitur untuk melakukan pengujian dan pemilihan *proxy server* yang terbaik, tetapi hanya memberikan kemudahan untuk pengguna agar bisa berganti-ganti *proxy server* dengan mudah dengan cara menyimpan beberapa konfigurasi *proxy* di dalam *file* konfigurasi dari *add-on* tersebut.

Selain *add-on* Proxy Selector untuk Mozilla Firefox, aplikasi *proxy selector* lain yang cukup populer pada saat ini adalah Proxy Switcher (Proxy Switcher Home Page, 2013), yang dapat melakukan pergantian konfigurasi *proxy* secara otomatis. Proxy Switcher adalah aplikasi berbayar yang melakukan pengecekan kecepatan *proxy server* secara terjadwal dan mengganti konfigurasi *proxy* secara otomatis sesuai dengan *proxy* yang tercepat pada saat itu.

Metode ini memiliki kelemahan, yang mana jika sebuah *proxy server* yang memiliki *history* performa yang buruk mendapatkan *throughput* yang sangat tinggi untuk beberapa detik saja, maka *proxy server* tersebut akan dipilih sebagai *proxy* terbaik sampai pada saat pengujian performa berikutnya sekalipun *proxy server* tersebut sudah kembali ke performa biasanya. Hal ini dapat diatasi dengan memperpendek interval di antara pengujian *proxy server*, tetapi cara ini akan

memberikan beban yang lebih berat kepada *host client* karena harus melakukan pengetesan secara terus menerus tanpa henti.

Dengan menggunakan adaptasi algoritma *link-state routing* pada *proxy selector* yang dibuat, diharapkan *proxy selector* yang dikembangkan dalam penelitian ini memiliki kemampuan untuk memilih *proxy* berdasarkan perhitungan probabilistik yang dibuat dari *history link-state* yang didapat dari pengujian-pengujian sebelumnya, sehingga *proxy selector* ini dapat memilih *proxy* yang terbaik dengan interval pengujian yang relatif lebih panjang.

1.2 Perumusan Masalah

1. Bagaimana cara mengadaptasikan algoritma *link-state routing protocol* yang umunya digunakan untuk *routing* agar dapat digunakan dalam *proxy selector*?
2. Bagaimana cara *proxy selector* dapat melakukan prediksi performa *proxy server* supaya *proxy selector* dapat memilih *proxy server* yang memiliki performa baik dan stabil pada saat digunakan sehingga *user* tidak perlu terlalu sering mengganti *proxy*?

1.3 Tujuan Penelitian

Penelitian bertujuan untuk mengimplementasikan algoritma *link-state* pada aplikasi *proxy selector*, dan memberikan aplikasi tersebut kemampuan untuk melakukan prediksi dengan perhitungan probabilistik.

1.4 Manfaat Penelitian

Penelitian ini menghasilkan sebuah aplikasi *proxy selector* berbasis *link-state* dengan probabilistik yang sehingga dapat memilih *proxy server* yang tepat untuk pengguna.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Bab I: Pendahuluan

   Berisi tentang latar belakang masalah, rumusan masalah, serta tujuan dan manfaat yang diharapkan dari hasil penelitian.

2. Bab II: Landasan Teori

   Teori-teori mengenai jaringan komputer, *proxy server*, pemilihan *server*, dan *moving average*.

3. Bab III: Metodologi dan Perancangan Aplikasi

   Desain aplikasi, *flowchart*, UML, spesifikasi sistem, dan mendefinisikan metode pengujian.

4. Bab IV: Implementasi dan Pembahasan

   Analisa terhadap hasil eksperimen, menentukan nilai *n* yang digunakan dalam perhitungan *moving average*, melakukan pengujian fungsionalitas dan pengujian kinerja.

5. Bab V: Kesimpulan dan Saran

Mengambil kesimpulan dari penelitian yang telah dilakukan dan memberikan saran-saran yang mungkin akan berguna untuk pengembangan aplikasi lebih lanjut.