SIMULASI DAN ANALISIS SISTEM *GATEWAY* TERDISTRIBUSI UNTUK *FIRST-HOP REDUNDANCY* DAN *LOAD BALANCING*

Haidlir Achmad Naqvi[1], Sofia Naning Hertiana, ST., MT.[2], Ridha Muldina Negara, ST., MT.[3]

[1,2,3] Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1] haidlir@acm.org, [2] sofiananing@telkomuniversity.ac.id, [3] ridhanegara@telkomuniversity.ac.id

Abstrak

Internet diharapakan untuk selalu tersedia sehingga layanan-layanan di atasnya dapat digunakan setiap saat. Untuk mendapatkan jaringan Internet yang memiliki *availability* yang tinggi, maka pada jaringan digunakan lebih dari satu *gateway* atau redundansi. Sehingga bila terjadi *down* pada salah satu *gateway*, *gateway* yang lain dapat mengambil tugas dari *gateway* yang *down* tersebut. Pada lapisan akses, penggunaan lebih dari satu *gateway* disebut *first-hop redundancy* (FHR). Dalam penelitian ini, Penulis mengusulkan sebuah rancangan FHR dan fungsi *load balancing*. *First-hop redundancy* dirancang menggunakan paradigma *software defined networking*. Dalam perancangan sistemnya, POX dipilih sebagai SDN *controller* serta Openflow sebagai protokol antar-muka antara *switch* dan *controller*. *FHR* yang dirancang disimulasikan di atas mininet. Melalui simulasi tersebut diuji kemampuan dari *first-hop redundancy* tersebut, antara lain *delay fail-over, load distribution* serta *resource utilization* dan *overhead* pada *controller* pada topologi bipartit. Dari hasil pengujian dan analisis dapat disimpulkan bahwa sistem dapat menjalankan fungsi FHR dengan menghasilkan *fail-over delay* yang stabil di bawah 140 ms untuk aliran tunggal. Fungsi *load balancing* dapat berjalan dan menaikkan performa jaringan sejalan dengan pertambahan jumlah *gateway* yang digunakan, meskipun proporsi persebaran beban pada *gateway* masih belum rata. Selain itu, Skalabilitas sistem masih perlu ditingkatkan.

Kata kunci : *first-hop redundancy, software defined networking, openflow.*

Abstract

Internet is expected to be available every time, so that the services above it are able to be used any time. To achieve high availability Internet network, redundancy is used. So, if one gateway is down, the other gateway will take the job of the dead gateway. In the access layer, we use first-hop redundancy (FHR) as the redundancy system. In this research, author propose a design of FHR which able to apply load balancing function. The FHR is designed using software defined networking paradigm. This research uses POX as controller and Openflow as Controller Datapath Protocol Interface between switches and controller. The system is simulated on the mininet. By this simulation we will examine fail over delay, load distribution, resource utilization and overhead on the bipartite topology. Of the examination results and analysis, the system is able to run the first-hop redundancy which yields fail-over delay below 140 ms for single flow. The load balancing function is also able to run and leverage the the performance of the network, even though the distribution of the load is not quite fair yet. The scalability of the system should be improved.

Kata kunci : *first-hop redundancy, software defined networking, openflow.*

1. Pendahuluan

Saat ini, terdapat lebih dari satu setengah milyar pengguna aktif Internet di seluruh dunia [2]. Mereka mencari informasi, berkomunikasi dan berkolaborasi dengan rekan kerja mereka menggunakan teknologi media sosial yang berjalan di atas Internet. Hal tersebut menunjukkan bahwa Internet sangat berpengaruh besar terhadap kehidupan manusia. Internet diharapakan untuk selalu tersedia sehingga layanan-layanan di atasnya dapat digunakan setiap saat. Konsekuensinya, dalam setiap proses desain jaringan Internet, jalur cadangan (*redundant*) selalu ditambahkan untuk melengkapi jalur utama. Sehingga bila jalur utama terganggu, lalu lintas data dapat dialihkan ke jalur cadangan, proses ini disebut *network redundancy*. Diharapkan dengan adanya *network redundancy*, sebuah jaringan dapat mencapai nilai ketersedian (*availability*) hingga 99.999% (*five nines*). *Network redundancy* dapat dilakukan di jaringan lapis inti (*core network*), jaringan distribusi (*distribution network*), dan jaringan akses (*access network*). Pada *core network*, *networks redundancy* dilakukan dengan menggunakan mekanisme *routing Internet Protocol* (L3) atau IP antar perangkat *router*, seperti RIP, OSPF, dan BGP. Di *distribution network*, selain memanfaatkan mekanisme *routing Internet Protocol*, *layer 2 network redundancy* (L2) digunakan bila hubungan antar perangkat pada jaringan ini menggunakan teknologi L2 yang sama, hal ini sering dilakukan di *data-center network* dan *campus network*. Begitu juga dengan *access network*, pada jaringan ini digunakan *layer 2 network redundancy*.

Saat ini teknologi layer 2 yang paling banyak diadopsi adalah Ethernet. *Network redundancy* di Ethernet memiliki mekanisme yang berbeda dengan IP, karena pengalamatan di Ethernet tidak memiliki tingkatan hierarki. Oleh karena itu, terdapat beberapa jenis mekanisme *redundancy* Ethernet yang dikelompokkan dalam protokol-protokol *first-hop redundancy*. Beberapa protokol *first-hop redundancy* yang digunakan pada jaringan saat ini

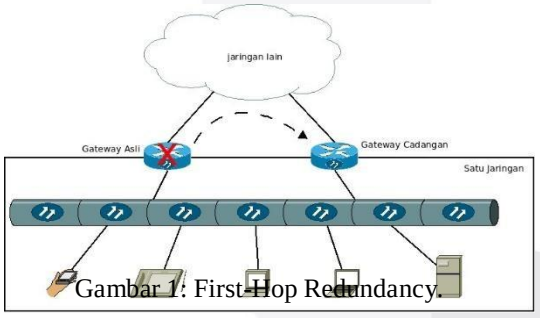
seperti Proxy Address Resolution Protocol dan Virtual Router Redundancy Protocol dibuat oleh Internet Engineering Task Force; Hot Standby Router Protocol dan Gateway Load Balancing Protocol yang dipatenkan Cisco; Common Address Reduncancy Protocol yang merupakan bagian dari Berkeley Software Distribution. Protokol-protokol tersebut bekerja dalam sistem terdistribusi, setiap perangkat mengatur sistem kerja dan kedudukan atau *state*-nya masing-masing, sehingga sistem yang ada sangat kompleks. Selain itu komunikasi antar *gateway* (mengirimkan *hello packet*) menggunakan infrastruktur jaringan yang sama dengan jaringan data sehingga menambah beban data yang ada di jaringan. Jadi, semakin banyak *gateway* yang terlibat di dalam *first-hop redundancy* maka sistem menjadi lebih kompleks dan semakin menambah beban jaringan (*overhead traffic*).

Di lain hal, saat ini sedang berkembang sebuah paradigma baru jaringan, yaitu *software defined networking* (SDN). *Software defined networking* menggunakan kendali terpusat (*logically centralized controller*) untuk mengatur perangkat infrastruktur jaringan secara men-detail (*fine-grained management*). Hubungan antara infrastruktur dan *controller* dilakukan melalui kanal khusus (*control channel*) yang dapat dilewatkan melalui kanal yang terpisah dengan kanal data, sehingga trafik *control* tidak mengurangi kapasitas yang digunakan oleh kanal data. Oleh karena itu, dalam tugas akhir ini, Penulis mengusulkan sebuah sistem yang menjalankan fungsi *first-hop redundancy* yang menerapkan paradigma *software defined networking*. Selain menjalan fungsi *first-hop redundancy*, sistem ini juga dapat melakukan fungsi *load balancing* sehingga memaksimalkan utilitas setiap *gateway*. Sistem ini menggunakan *least-load per-flow load balancing*.

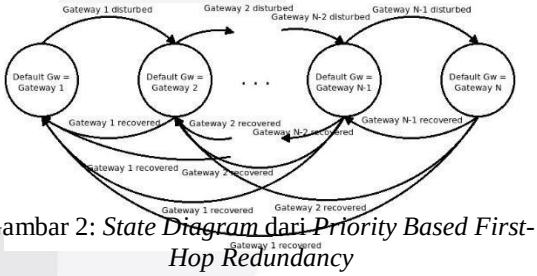
2. Dasar Teori

2.1 First-Hop Redundancy

Untuk mendukung layanan Internet dengan *availability* dan *reliability* yang tinggi, pada jaringan dapat digunakan lebih dari satu *gateway* atau redundansi. Sehingga bila terjadi *down* pada salah satu *gateway*, *gateway* yang lain dapat mengambil tugas dari *gateway* yang down tadi. Penggunaan lebih dari satu *gateway* ini dapat dimanfaatkan pada jaringan lapisan *core*, *distribution*, dan *access*. Pada lapisan *core* dan *distribution* dapat dimanfaatkan protokol *routing* untuk redundansinya. Pada lapisan akses, penggunaan lebih dari satu *gateway* disebut *first-hop redundancy*.



Gambar 1: First-Hop Redundancy.



Gambar 2: *State Diagram* dari *Priority Based First-Hop Redundancy*

First-hop redundancy merupakan sebuah mekanisme untuk menjadikan salah satu *router* sebagai default gateway saat terdapat dua atau lebih *gateway* pada jaringan yang menggunakan teknologi layer 2 seperti Ethernet, yang biasanya dijadikan mekanisme *network redundancy* di *distribution network* dan *access network*, dan diaplikasikan di *data-center network* maupun *campus network*. Selain itu jika terjadi *down* pada *router* maupun *link* yang menuju ke *gateway* tersebut, *gateway* dapat dialihkan ke *gateway* lainnya. Protokol ini tidak menambahkan kemampuan *routing*. Terdapat beberapa protokol *first-hop redundancy* yang digunakan pada jaringan saat ini: Proxy Address Resolution Protocol dan Virtual Router Redundancy Protcol dibuat oleh Internet Engineering Task Force; Hot Standby Router Protocol dan Gateway Load Balancing Protocol yang dipatenkan Cisco; Common Address Router Protocol yang merupakan bagian dari Berkeley Software Distribution.

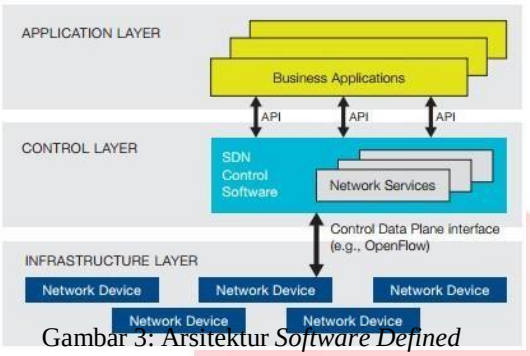
2.2 Load Balancing

Load balancing merupakan metode untuk menyebarkan beban kerja antara lain proses, komputer, jaringan, media penyimpanan atau sumber daya lainnya, sehingga tidak ada sumber tunggal sumber daya yang sibuk, di mana sumber sumber daya lainnya masih tersedia. *Load balancing* dapat juga dipandang sebagai pendistribusian barang ke dalam ember atau keranjang, data ke lokasi memori, berkas ke media penympanan, tugas ke pemproses, paket ke antar-muka jaringan dan permintaan ke server. Tujuannya adalah pendistribusian yang sesuai dengan parameter yang diinginkan.

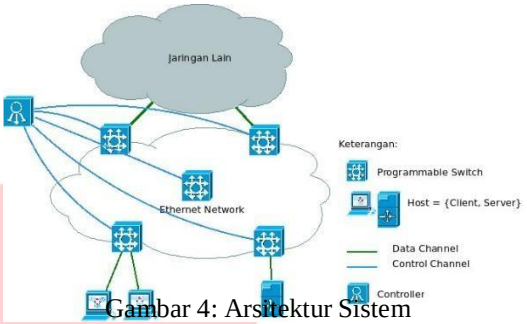
2.3 Software Defined Networking

Dikutip dari *white paper* yang diterbitkan oleh Open Network [5,6,7] *software defined networking* (SDN) merupakan sebuah arsitektur jaringan yang sedang berkembang di mana *control plane* dipisahkan dari *forwarding plane* dan dapat secara langsung diprogram. Pemindahahan pengendali, yang sebelumnya terikat pada setiap perangkat jaringan, menjadi dapat diakses oleh perangkat komputasi memungkinkan infrastruktur jaringan dapat

diringkas sebagai aplikasi dan layanan jaringan, yang dapat memperlakukan jaringan sebagai sebuah entitas logika atau maya.



Gambar 3: Arsitektur *Software Defined Networking*. [5,6,7]



Gambar 4: Arsitektur Sistem

3. Perancangan Sistem

3.1 Arsitektur Sistem

Sistem ini dijalankan di atas arsitektur *software defined networking* seperti pada gambar 4. Perangkat jaringan yang digunakan adalah *programmable switch* yang mendukung protokol Openflow 1.0. Semua *switch* dihubungkan ke sebuah *controller*. Jaringan yang menghubungkan seluruh *switch* adalah jaringan Ethernet. Antar-muka jaringan (*network interface*) di setiap *switch* tidak memiliki alamat IP, karena alamat IP *gateway* dan *network interface* yang terhubung keluar tersimpan di dalam *controller*.

Host dapat berupa komputer *client*, server, dan perangkat lainnya; mendapatkan alamat IP dinamis yang diatur oleh *dynamic host control protocol service* (DHCP) yang dijalankan oleh *controller*. *Switch* berposisi sebagai *relay* antara *host* dengan *controller* dalam proses DHCP *discover-offer-request*. Alamat IP yang digunakan hanya satu subnet dalam satu jaringan L2 *broadcast multi-access* Ethernet.

Sistem yang dibuat dalam bentuk aplikasi SDN diletakkan di dalam *controller*. Sistem ini mengatur proses *forwarding*, *monitoring*, *DHCP service*, dan *Proxy ARP service*. Proses-proses ini dijalankan baik sebagai fungsi dasar jaringan maupun untuk mendukung fungsi utama dari sistem yang dibuat, yaitu *first-hop redundancy* dan *gateway load balancing* yang akan dijelaskan di sub-bab selanjutnya.

3.2 Fungsi Sistem

Sistem yang dibuat memiliki dua fungsi utama, yaitu first-hop redundancy dan gateway load balancing.

3.2.1 First-Hop Redundancy

Mekanisme *first-hop redundancy* secara normal dapat digambarkan dalam *state diagram* gambar 2. Namun, karena pada sistem yang dibuat, fungsi *first-hop redundancy* akan digabung dengan *fungsi load balancing*, maka *diagram state* seperti pada gambar 2 tidak lagi digunakan. Pada sistem yang dibuat fungsi *first-hop redundancy* yang digunakan memiliki jumlah *gateway* sesuai dengan *transition state diagram* pada gambar 5.

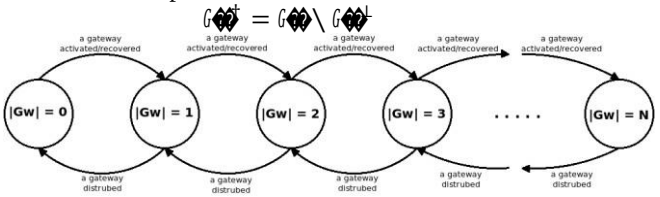
3.2.2 Gateway Load Balancing

Selain dapat melakukan *first-hop redundancy*, ketersediaan lebih dari satu *gateway*, memungkinkan metode *load balancing* juga dapat dilakukan oleh sistem ini. *Load Balancing* yang digunakan adalah *least-load per-flow load balancing*, di mana setiap *flow* merupakan kombinasi alamat IP dan alamat *port* sumber-tujuan. *Controller* akan memilih *gateway* dengan beban trafik yang paling rendah pada saat itu, sebagai *gateway* yang ditugaskan untuk meneruskan aliran data *flow* tersebut. Bila *Gw* merupakan himpunan dari seluruh *gateway* di mana $Gw = \{gw1, gw2, \ldots, gw_n\}$ dan gateway yang hidup adalah $Gw^{\dagger}$, maka *gateway* yang terpilih adalah *gw* sesuai persamaan 1, di mana *load*(*x*) merupakan fungsi yang menghitung beban trafik pada *gateway* tersebut.

$$gw' \Leftrightarrow load(gw') = (load(Gw^{\dagger})) \quad (1)$$

Karena pada sistem ini fungsi *first-hop redundancy* dan *load balancing* dijalankan secara simultan, maka tidak terdapat *default gateway* dan dalam proses pemilihan *gateway* untuk setiap *flow*, hanya *gateway-gateway* yang tidak terganggu yang ikut dalam proses seleksi. Sementara *gateway-gateway* yang mati tidak akan diikutkan dalam proses seleksi tersebut. Bila $Gw^{\perp}$ merupakan himpunan dari seluruh *gateway* yang mati, di mana $Gw^{\perp} \subseteq Gw$. Gateway yang hidup adalah $Gw^{\dagger}$ sesuai persamaan 2.

$$Gw^{\dagger} = Gw \setminus Gw^{\perp} \quad (2)$$



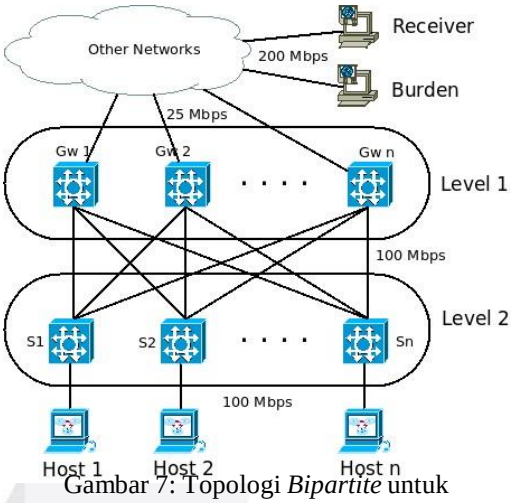
Gambar 5: *State Diagram* dari *First-Hop Redundancy* dengan *Load Balance*

4. Evaluasi Sistem

4.1 Perangkat Simulasi

Dalam tugas akhir ini, *simulator* yang digunakan dalam pengujian adalah Mininet. Mininet dipasang di dalam komputer A dan *controller* berada di komputer B, sesuai dengan gambar 6. Komputer A dan B dihubungkan ke jaringan lokal Telkom University dengan menggunakan kabel UTP Cat 5 100 Mbps, sehingga mendapatkan alamat IP dari DHCP Server Sisfo. Dalam setiap simulasi yang dilakukan, Mininet menggunakan *remote controller* yang diarahkan ke alamat IP komputer B, sehingga setiap *switch* yang berada di dalam Mininet terhubung ke *controller* di komputer B. Komputer A merupakan sebuah *personal computer* dengan *processor* Intel Core i3-3240T CPU @ 2.90 GHz x4 dan RAM sebesar 3.8 GB. Komputer A menggunakan sistem operasi Ubuntu 12.04 LTS 64 bit. Selain Mininet, di dalam komputer A juga dipasang *Distributed Internet Traffic Generator* (D-ITG) dan Iperf sebagai pembangkit trafik paket data. Dalam setiap menjalankan simulasi, komputer A hanya menjalankan mininet dan perangkat-perangkat lunak pendukung yang dibutuhkan. Komputer B merupakan sebuah *laptop* dengan *processor* Intel Core i3-2350 CPU @ 2.30 GHz x4 dan RAM 3.7 GB. Komputer B menggunakan sistem operasi Ubuntu 14.04 LTS 64 bit. Selain sebagai *controller*, komputer B juga digunakan sebagai tempat pengamatan dan pengambilan data, baik data yang dikirim dari *switch* ke *controller*, maupun data yang diambil langsung dari *controller*.

Gambar 6: Perangkat Simulasi

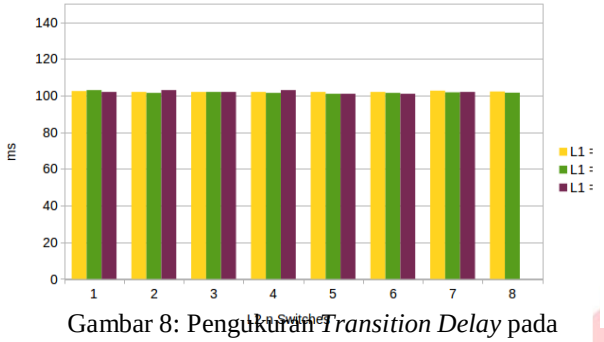


Gambar 7: Topologi *Bipartite* untuk Pengukuran Fail-Over Delay

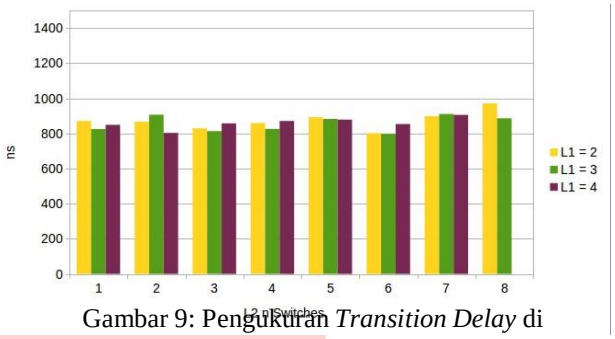
4.2 Pengukuran *Fail-Over Delay*

Fail-over delay merupakan waktu tunda yang diukur setelah sebuah *gateway* mati hingga terdapat *gateway* lain yang mengambil tugas dari *gateway* yang mati tersebut. Satuan yang digunakan adalah milidetik (ms). Pengukuran dilakukan di *host* dan di *controller*. Pengukuran di kedua titik tersebut dilakukan untuk mengetahui *delay* yang dibutuhkan setelah respon dari *controller* diterbitkan hingga *flow* yang baru terpasang di *switch*. Pengukuran parameter ini dilakukan dengan menggunakan topologi *bipartite*, dengan nilai *level* 1 dan *level* 2 yang berbeda-beda, di mana *level* 1 merupakan *gateway* dan *level* 2 merupakan *switch* yang terhubung ke *host*, sesuai dengan gambar 7. Dalam pengujian ini digunakan satu hingga empat *gateway* dan satu hingga delapan *level 2 switch*.

Pengujian dilakukan menggunakan paket ICMP Ping tunggal sebagai paket yang dikirimkan dari *host* ke *receiver*. Selanjutnya dilakukan pengecekan untuk mengetahui di *gateway* mana paket-paket ICMP tersebut lewat. Selanjutnya *uplink* pada *gateway* yang dilewati paket-paket ICMP tersebut diputus, maka *host* akan menerima *time out reply* selama beberapa waktu hingga mendapat *ping reply* lagi. Jumlah *time out* tersebut yang dijadikan tolak ukur berapa lama *transition delay* yang dirasakan oleh *host*. Sementara itu *controller* mencatat waktu ketika laporan *link* putus masuk ke *controller* dan waktu ketika respon untuk menangani *link* putus tersebut dikirimkan dari *controller* ke *switch*.



Gambar 8: Pengukuran *Transition Delay* pada Infrastruktur



Gambar 9: Pengukuran *Transition Delay* di Sistem

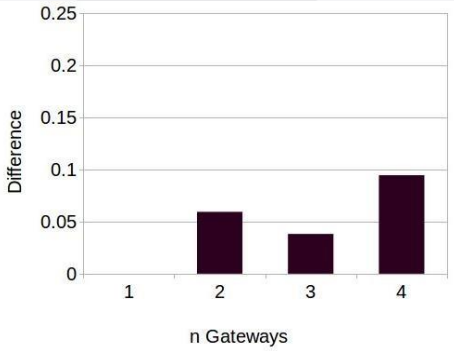
Dari hasil pengujian pada gambar 8 dan 9, kinerja sistem untuk menjalankan fungsi *first-hop redundancy* dapat dikatakan stabil. Kestabilan kinerja tersebut didasarkan pada waktu transisi di sisi infrastruktur yang stabil di bawah 140 ms untuk aliran tunggal. Sementara itu, waktu transisi di sisi sistem atau *controller* masih berada di bawah 1000 ns, sehingga tidak terlalu berpengaruh terhadap waktu transisi di sisi infrastruktur. Waktu transisi di sisi infratruktur ini merupakan waktu transisi yang dirasakan oleh pengguna. Kinerja fungsi *first-hop redundancy* sistem ini juga dapat dikatakan baik, hal ini bila waktu transisi 140 ms dibandikan dengan waktu transisi VRRP yang berkisar di 145 ms pada *backup priority* 220 sesuai dengan penilitian yang dilakukan oleh Bernat [1] dan 1,1 sekon sesuai dengan penelitian yang dilakukan oleh Yudiani [9].

Pada gambar 4.3 dan 4.4 tidak terdapat data yang lengkap pada L2 = 8. Sistem tidak dapat menjalankan kombinasi L1 = 4 dan L2 = 8 karena kebutuhan sumber daya memori yang dibutuhkan dalam perhitungan *routing* menggunakan teknik rekursif tidak dapat dipenuhi oleh perangkat keras di *controller*. Hal ini dapat dianggap sebagai kekurangan dari sistem yang harus diperbaiki. Perbaikan yang dilakukan mungkin dapat menggunakan teknik non-rekursif untuk perhitungan *routing* maupun menggunakan algoritma *routing* lainnya.

4.3 Pengukuran *Load Distribution*

Pengukuran *load distribution* dilakukan untuk mengetahui proporsi pembagian beban di setiap *gateway*. Gambar 10 menampilkan selisih antara proporsi terbesar dan terkecil dari setiap *gateway* yang digunakan sesuai persamaan 3. Pengukuran dilakukan di *controller* dengan memantau besarnya *trafik* di setiap *gateway*. Pengukuran parameter ini dilakukan dengan menggunakan topologi *bipartite*, dengan nilai *level* 1 sesuai dengan jumlah *gateway* yang digunakan dan *level* 2 adalah satu, di mana *level* 1 merupakan *gateway* dan *level* 2 merupakan *switch* yang terhubung ke *host*, sesuai dengan gambar 7.

$$selisih = proporsi_{max} - proporsi_{min} \quad (3)$$



Gambar 10: Selisih Proporsi Maksismum dan Minimum pada *Gateway*

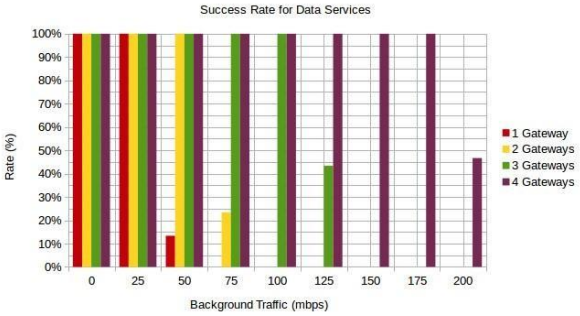
Dari gambar 10, dapat dikatakan bahwa kinerja fungsi *load balancing* masih belum konsisten. Hal ini terlihat dari tren selisih proporsi maksismum dan minimum yang meningkat bersamaan dengan bertambahnya jumlah *gateway* yang digunakan. Kinerja fungsi *load balancing* dapat diperbaiki dengan merubah metode *load balancing* yang digunakan.

Untuk melengkapi data pengukuran *load distribution*, pada tugas akhir ini juga dilakukan pengukuran performa fungsi *load balancing* sistem yang dibuat. Dalam melakukan pengukuran ini, dibangkitakan tiga jenis trafik [9] UDP, antara lain:

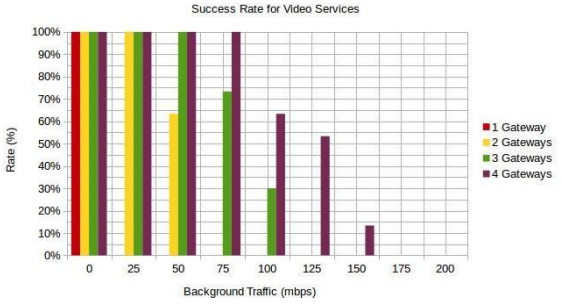
1. Data, dengan *inter-departure time* (IDT) konstan 100 pps, ukuran paket terdistribusi *poisson* dengan μ= 48 bytes, sehingga secara teoritis membutuhkan *throughput* 38,4 kbps.

2. Video Streaming 24 *frame* (satu paket per-frame) per-sekon, ukuran paket terdistribusi normal dengan μ = 27791 bytes dan σ^2 = 6254 bytes, sehingga dibutuhkan sekitar 5,4 Mbps *throughput*.
3. VoIP dengan G.711 *codec* tanpa *voice activation detection* (VAD) sebanyak 100 pps dan memiliki ukuran paket 80 bytes, sehingga dibutuhkan *throughput* sekitar 70,4 kbps.

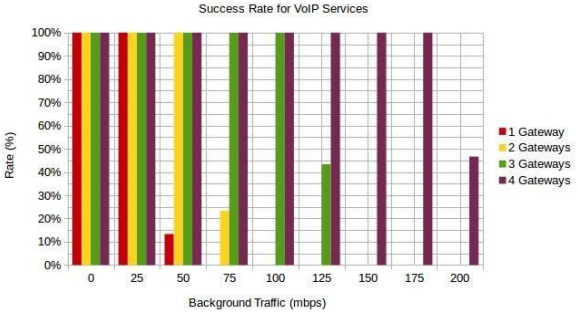
Tiga jenis trafik tersebut dibangkitkan secara bersamaan dan dikirim ke *receiver* sesuai gambar 7. Pada saat yang bersamaan pula dibangkitkan lalu lintas data sebagai beban yang nilainya merentang dari 0 Mbps sampai dengan 200 Mbps. Hal ini dilakukan untuk mengetahui seberapa besar perbaikan performa yang didapat ketika menggunakan lebih dari satu *gateway*.



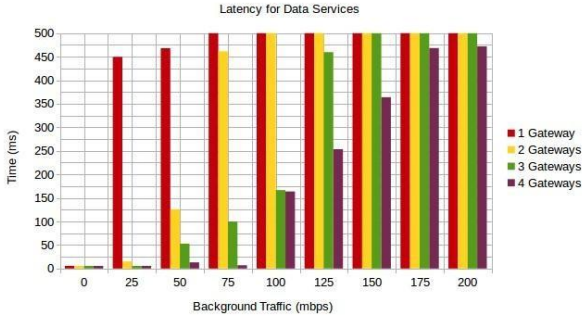
Gambar 11: *Success Rate* Servis Data



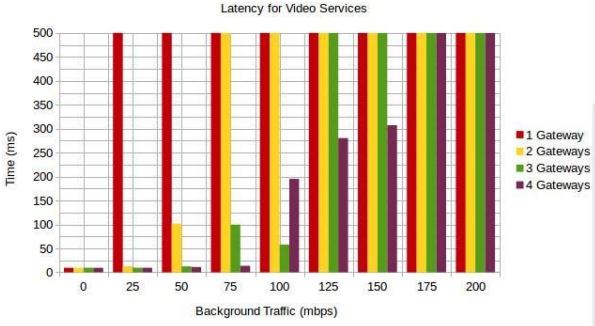
Gambar 12: *Success Rate* Servis Video



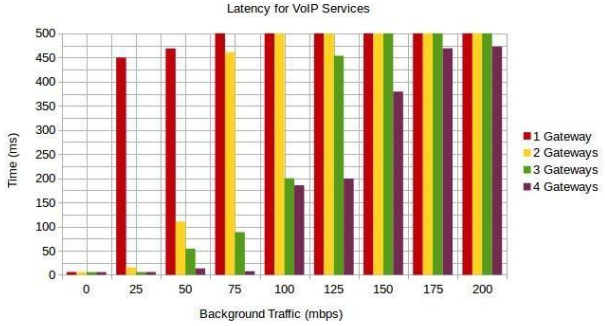
Gambar 13: *Success Rate* Servis VoIP



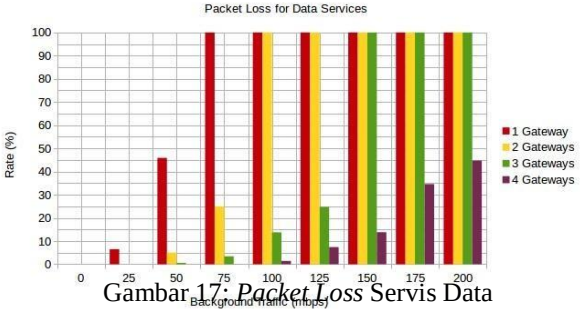
Gambar 14: Latensi Servis Data



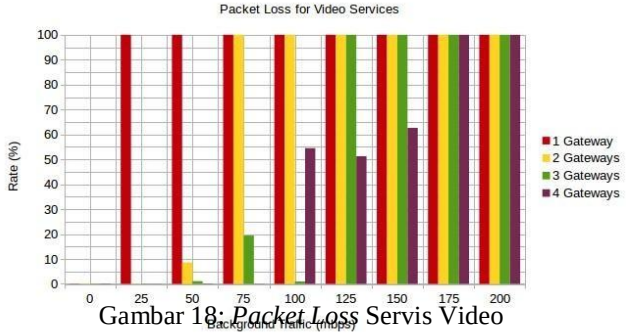
Gambar 15: Latensi Servis Video



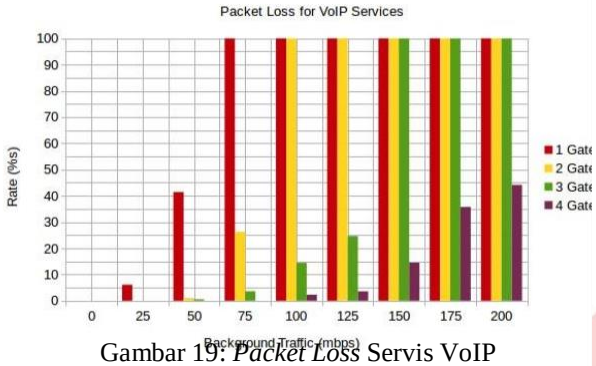
Gambar 16: Latensi Servis VoIP



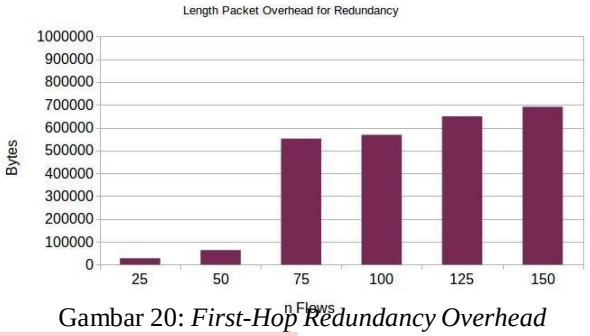
Gambar 17: *Packet Loss* Servis Data



Gambar 18: *Packet Loss* Servis Video



Gambar 19: *Packet Loss* Servis VoIP



Gambar 20: *First-Hop Redundancy Overhead*

Gambar 11, 12, 13 menampilkan tingkat kesuksesan pengiriman trafik data dari *host* ke *receiver* dari tiga puluh kali percobaan, semakin besar *background traffic* yang dibebankan ke *gateway*, maka tingkat kesuksesan semakin rendah. Tingkat kesuksesan trafik data identik dengan tingkat kesuksesan trafik VoIP, namun untuk trafik video memiliki tren yang berbeda dibandingkan dengan kedua trafik lainnya. Dari tiga gambar tersebut dapat disimpulkan bahwa bertambahnya *gateway* menaikkan tingkat kesuksesan pada kolom *background traffic* yang sama.

Gambar 14, 15, dan 16 menampilkan latensi serta gambar 17, 18, dan 19 menampilkan *packet loss percentage* untuk setiap jenis trafik dan sampel yang sukses saja. Pada trafik data dan VoIP dapat disimpulkan bahwa semakin bertambahnya *gateway* maka latensi dan *loss* yang didapatkan semakin rendah pada kolom *background traffic* yang sama. Pada trafik video, terjadi sebuah anomali di mana latensi dan *loss* saat *background traffic* 100 Mbps lebih kecil daripada saat *background traffic* 75 Mbps. Namun apabila dikorelasikan dengan gambar 12 dapat dilihat bahwa tingkat kesuksesan trafik video saat *background traffic* 100 Mbps hanya 30%, sedangkan saat *background traffic* 75 Mbps tingkat kesuksesannya hampir 75% atau dua kali lipat. Hal ini menunjukkan bahwa walaupun latensi lebih rendah pada *background traffic* 100 Mbps, namun hal ini dikompensasi dengan tingkat kesuksesan yang lebih rendah dibandingkan saat *background traffic* 75 Mbps.

Dari analisis-analisis di atas, dapat disimpulkan bahwa fungsi *load balancing* perlu diperbaiki untuk mendapatkan persebaran beban yang lebih baik lagi. Selain itu dengan menggunakan fungsi *load balancing*, maka pertambahan *gateway* dapat menaikkan performa dari jaringan dilihat dari meningkatnya tingkat kesuksesan trafik, menurunnya latensi trafik, dan menurunnya tingkat kehilangan paket. Salah satu alternatif yang dilakukan untuk memperbaiki fungsi *load balancing* antara lain dengan menggunakan mekanisme *load balancing* lain. Alternatif lainnya adalah mengubah parameter *flow*.

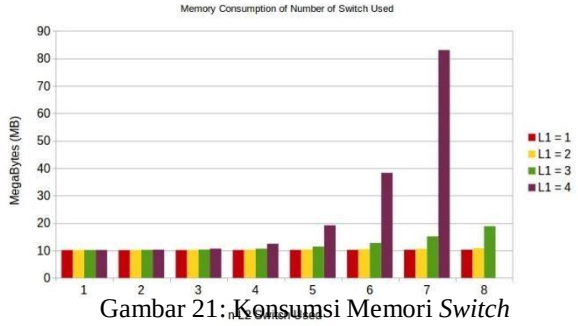
4.4 Pengukuran *Overhead*

Overhead dalam pengukuran ini adalah trafik kontrol yang dibutuhkan dalam proses *fail-over*. Satuan yang digunakan adalah bytes. Pengukuran dilakukan dengan membangkitkan trafik dengan jumlah aliran tertentu, lalu setelah beberapa saat, *uplink gateway* yang dilewati oleh aliran tersebut diputus. Perekaman paket dilakukan di *controller* menggunakan perangkat lunak *wireshark*. Besar *overhead* dihitung dari laporan *switch* saat *link* putus hingga paket modifikasi aliran (*flow mod* - bertugas mengalihkan jalur aliran) yang terakhir. Pada pengukuran ini, untuk memudahkan proses pengambilan data maka fungsi *load balancing* tidak diaktifkan, sehingga dapat lebih mudah mengetahui *gateway* yang digunakan.

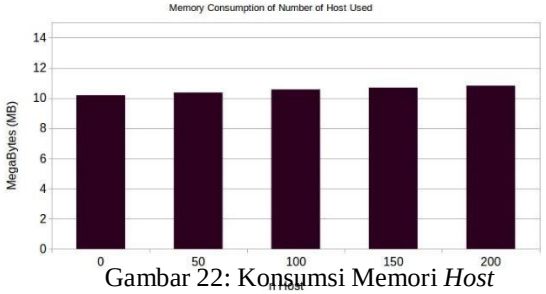
Gambar 20 menunjukkan grafik *overhead* untuk setiap jumlah aliran tertentu. Terlihat bahwa semkin banyak jumlah aliran maka ukuran *overhead* semakin besar. Selain itu, pada gambar terlihat perbedaan ukuran *overhead* yang sangat jauh diantara aliran dengan jumlah 50 dan 75. Hal ini terjadi karena, pada saat jumlah aliran 50 dan 25, hanya terdapat jenis paket *port status* dan *flow mod*. Namun saat aliran 75 dan setelahnya, selain terdapat jenis paket *port status* dan *flow mod*, juga terdapat jenis paket *barrier request* yang bertugas untuk memastikan tidak terdapat *flow-entry* yang saling tumpang tindih atau ganda.

4.5 Pengukuran *Resource Utilization*

Pengukuran ini dilakukan untuk mengetahui berapa besar memori yang digunakan dalam *first-hop redundancy* yang dirancang. Satuan yang digunakan adalah Megabytes (MB). Pengukuran dibagi menjadi dua bagian: konsumsi memori oleh *switch* dan konsumsi memori oleh *host*. Kombinasi jumlah *switch* sama seperti skenario pertama, yaitu L1 merentang dari satu sampai dengan empat, sedangkan L2 merentang dari satu sampai dengan delapan. Pengukuran dilakukan di *controller*.



Gambar 21: Konsumsi Memori *Switch*



Gambar 22: Konsumsi Memori *Host*

Gambar 21 secara umum menunjukkan bahwa semakin banyak *switch* yang digunakan maka konsumsi memori yang digunakan semakin besar. Lebih dari itu, dapat dilihat pada L1 = 4, kenaikan konsumsi memori-nya sangat tinggi. Hal ini dikarenakan pada saat L1 = 4, link yang digunakan semakin banyak serta sirkuit yang dihasilkan juga semakin banyak, sehingga jumlah hasil perhitungan *routing* juga semakin banyak. Hal ini dapat dianggp sebagai kekurangan dari sistem yang harus diperbaiki. Hal yang dapat dilakukan adalah menggunakan teknik nonrekursif dalam perhitungan *routing* maupun mengganti mekanisme *routing* yang digunakan.

Gambar 22 menunjukkan bahwa semakin banyak *host* yang terhubung ke jaringan maka konsumsi memori yang digunakan semakin besar. Namun kenaikkan konsumsi memori ini tidak terlalu besar, tidak lebih dari 4 KB per-host. Hal ini artinya sistem dapat menangani 5000 host hanya dengan menggunakan memori 30 MB.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Sistem yang dibuat menggunakan paradigma software defined networking dapat simulasikan pada jaringan yang menggunakan dua atau lebih gateway, dengan hasil evaluasi sebagai berikut:

1. Fungsi first-hop redundancy dapat mengatasi kegagalan link yang terjadi pada gateway dan mempunyai kinerja yang stabil bila dilihat dari waktu transisi di sisi infrastruktur yang berada dibawah 140 ms, atau dengan kata lain masih di bawah waktu transisi VRRP yang berkisar di 145 ms (pada backup priority 220).
2. Fungsi load balancing dapat mendistribusikan beban ke beberapa gateway, namun mempunyai kinerja yang belum baik, didasarkan pada nilai selisih proporsi beban maksimum dan minimum di gateway mendekati 0,1 dan kecenderungan nilai tersebut meningkat bersamaan dengan bertambahnya jumlah gateway.
3. Skalabilitas dari sistem yang dibuat masih belum baik, didasarkan bahwa sistem tidak dapat menangani jaringan bipartit dengan kombinasi L1 = 4 dan L2 = 8.

5.2 Saran

Penerapan algoritma routing DFS pada sistem membatasi skalabilitas dari sistem tersebut, sehingga disarankan untuk menggunakan sistem ini pada topologi yang memiliki sirkuit kurang dari jumlah sirkuit yang dihasilkan topologi bipartit dengan kombinasi L1 = 4 dan L2 = 8

Daftar Pustaka:

[1] Bernat, Joaquim S. 2011.*Redundancy and Load Balancing at IP layer in Access and Aggregation Networks*. Master Thesis. Aalto University.

[2] Chui, Michael, et al 2012. *The Social Economy: Unlocking Value and Productivity Through Social Technologies*. Report. McKinsey Global Institute.

[3] Ejaz, Syed K. 2011. *Analysis of the trade-off between performance and energy consumption of existing load balancing algorithms*. Groer Beleg. Technische Universitt Dresden.

[4] Lantz, Bob, Brandon Heller dan Nick McKeown. 2010. *A Network in a Laptop: Rapid Prototyping for Software-defined Networks*. dalam *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York. ACM.

[5] Open Network Foundation.2009. *Openflow Switch Specification version 1.0.0 (Wire Protocol 0x01)*.

[6] Open Network Foundation.2012. *Software-Defined Networking: The New Norm for Networks*.

[7] Open Network Foundation.2013. *SDN in the Campus Environment*.

[8] Avallone, Sefano, et al 2004. *A Practical Demonstration of Network Traffic Generation*. dalam *Proceedings of the 8th IASTED International Conference*. Hawaii.

[9] Yudiani, Rd. Amanda. 2013. *Implementasi dan Analisis Virtual Router Redundancy Protocol (VRRP) dan Hot Standby Router Protocol(HSRP)*. Tugas Akhir. Bandung. Institut Teknologi Telkom.