REPLIKASI DATA MENGGUNAKAN DETECTION CONTROLLER MODULE UNTUK MENCEGAH CONGESTION DI DATA CENTER

Erna Auparay[1)] dan Royyana M. Ijtihadie[2)]
[1,2)] Departemen Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya, 60111, Indonesia
e-mail: ernaworabay@yahoo.com[1)], royyana@gmail.com[2)]

ABSTRAK

Replikasi data secara terdistribusi merupakan salah satu fase penting dalam proses distribusi data terpusat. Replikasi data melakukan sebuah teknik copy atau duplikasi dalam proses pendistribusian data logic. Pendistribusian ini saling berhubungan baik secara sistem ataupun secara aplikatif dengan data yang terdistribusi dalam jaringan komputer. Media penyimpanan yang terdistribusi akan melaksanakan sinkronisasi antara server sumber ke server tujuan dari satu media penyimpanan ke media penyimpanan yang lain sehingga konsistensi data dapat terjamin. Penggunaan Software Defined Network (SDN) controllers pada data center berfungsi untuk mengendalikan proses pendistribusian data ke beberapa lokasi yang menyimpan data yang sama. Hal ini sangat berguna pada saat lokasi-lokasi tersebut membutuhkan data yang sama atau memerlukan server yang terpisah dalam pembuatan aplikasi laporan. SDN controllers juga digunakan untuk mengendalikan entitas pendistribusian, yang mungkin menunjukkan behaviour yang berbeda pada event triggers, namun apabila terjadi kepadatan arus lalu lintas pendistribusian di jaringan data center yang menyebabkan terjadinya kongesti atau kemacetan sehingga perlu adanya strategi penyelesain. Oleh karena itu, paper ini akan membahas pendekatan strategi replikasi data yang digunakan untuk mengatasi kepadatan arus lalu lintas jaringan.

***Kata Kunci**: Kongesti, data center, distribusi data, replikasi data*

ABSTRACT

Distributive data replication is one of the important phases in the process of centralizing distributed data. Data Replication uses copy or duplication technique in the process of logical data distribution. This distribution will interconnect to either system or application data which is distributed in the computer network. The distribution of media storage from one storage to another storage will implement the synchronization between the source server and the destination server, so that the data consistency can be guaranteed. The use of Software Defined Network (SDN) controllers in the data center is to control the data distribution process to multiple locations which stores the same data. This is very useful when these locations require the same data or require a separate server to create reporting applications. SDN controllers is also used to control the distribution entities, which may exhibit different behaviour on event triggers, but in the case of traffic flow density distribution in the data center network causes congestion/bottlenecks so that need completion strategies. Therefore, this paper will discuss about data replication strategies that can be used to cope with the density of the network traffic flow.

***Keywords**: Congestion, data center, data distribution, data replication*

I. PENDAHULUAN

PADA era teknologi sekarang ini, kebutuhan akan data center tidak terhindarkan lagi, kebutuhannya menjadi semakin besar. Software Defined Network (SDN) *controllers* secara logis mengatur arus yang terjadi dalam *control plane*, maka data center dituntut untuk selalu siap dalam menyediakan data yang frekuensi aksesnya semakin tinggi dan jenis data yang ukuranya makin besar dan beragam [1]. Pertumbuhan data yang luar biasa telah menjadikan data center harus lebih mengembangkan sistemnya baik secara sistem ataupun aplikatif. Jutaan server dan media penyimpanan yang tak terhitung banyaknya memproses semua permintaan pengguna tanpa henti di data center.

Data center menjamin ketersediaan data yang besar dan pastinya beragam. Dan untuk mendukung data center dalam melakukan tugasnya, dibutuhkan suatu sistem Replikasi Data Terdistribusi. Replikasi data adalah suatu teknik untuk melakukan duplikasi dan pendistribusian data. Dengan menggunakan teknik replikasi ini, data dapat didistribusikan ke lokasi yang berbeda baik melalui koneksi jaringan lokal maupun internet. Replikasi data akan memelihara sistem dengan melakukan penyimpanan salinan atau duplikasi.

Beberapa pendekatan telah dilakukan salah satunya dengan menggunakan replikasi yang juga untuk mendukung kinerja aplikasi, penyebaran data fisik sesuai dengan penggunaannya, seperti pemrosesan transaksi online dan pemrosesan database terdistribusi melalui beberapa server. Banyaknya lalu lintas komunikasi internet yang terjadi di data center dan kurangnya efisiensi kualitas layanan menyebabkan terjadinya kongesti atau kemacetan pada ethernet. Manajemen kongesti atau kemacetan pada ethernet harus mempertimbangkan isu-isu seperti *delay* atau keterlambatan yang rendah, tingkat kehilangan paket *(packet loss)* yang rendah dan *throughput*

yang tinggi dari jaringan. Hal ini menyebabkan *latency* yang rendah dan *troughput* yang tinggi pada proses replikasi data. Namun, aplikasi yang berbeda mempunyai kebutuhan yang berbeda terhadap konsistensi data [1] [3] [4].

Pada paper ini dilakukan pembahasan mengenai konsep dasar replikasi data terdistribusi pada data center. Selain itu, paper ini akan membahas faktor-faktor apa saja yang mendukung replikasi data terdistribusi secara aplikatif, dan beberapa perencanaan replikasi secara aplikatif apabila terjadi kongesti. Tujuan dari penulisan paper ini adalah untuk membahas aplikasi data center yang dapat menyatukan data dari berbagai server dengan basis data yang beragam secara otomatis. Berdasarkan hal tersebut di atas, maka data center sangat dibutuhkan dimana dengan adanya sistem replikasi data diharapkan dapat membantu proses pemusatan data dan ketersediaan data secara terus-menerus.

II. STUDI LITERATUR

Secara garis besar, konsep dasar replikasi data merupakan teknik duplikasi data yang dapat mendistribusikan data ke lokasi yang berbeda melalui koneksi jaringan lokal maupun internet. Replikasi juga memungkinkan untuk mendukung kinerja aplikasi, penyebaran data fisik sesuai dengan penggunaannya, seperti pemrosesan transaksi online dan DSS (Desiscion Support System) atau pemrosesan database terdistribusi melalui beberapa server. Adapun keuntungan replikasi tergantung dari jenis replikasi tetapi pada umumnya replikasi mendukung ketersediaan data setiap waktu dan dimanapun diperlukan. Keuntungan lain yang mendukung penggunaan replikasi data adalah sebagai berikut [1] [3] [5]:

1. Aplikasi transaksi online terpisah dari aplikasi pembacaan seperti proses analisis database secara online, *smarts data* atau data warehouse.
2. Memungkinkan otonomi yang besar. Pengguna dapat bekerja dengan menduplikasi data pada saat tidak terkoneksi kemudian melakukan perubahan untuk dibuat database baru pada saat terkoneksi
3. Data dapat ditampilkan seperti layaknya melihat data tersebut dengan menggunakan aplikasi berbasis web.
4. Meningkatkan kinerja pembacaan dan membawa data mendekati lokasi individu atau kelompok pengguna. Hal ini akan membantu mengurangi masalah karena modifikasi data dan pemrosesan query yang dilakukan oleh banyak pengguna.
5. Pendistribusian melalui jaringan dan data dapat dibagi berdasarkan kebutuhan masing-masing unit atau pengguna.
6. Penggunaan replikasi data sebagai bagian dari strategi *standby server*.

A. Jenis–jenis replikasi data

Berikut ini akan dijelaskan jenis-jenis replikasi data, yaitu:

1) Snapshot replication

Mendistribusikan data yang dapat dilihat pada saat tertentu tanpa melakukan update. Biasanya digunakan pada saat memerlukan tampilan data seperti: daftar harga, katalog, dan data yang digunakan untuk pengambilan keputusan. Data-data ini sifatnya hanya '*read only*'. Replikasi ini membantu pada saat:

- Data sebagian besar statis dan tidak sering berubah.
- Sistem menerima copy data yang telah melewati batas waktu yang ditentukan.
- Datanya sedikit.

2) Transactional replication

Replikasi ini memelihara kekonsistenan transaksi yang terjadi.

3) Merge replication

Merge replication memungkinkan pengguna bekerja dan merubah data sesuai dengan wewenangnya. Pada saat server tidak dikoneksikan ke seluruh lokasi dalam topologi, replikasi merubah ke nilai data yang sama.

B. Teknik Replikasi

1) Teknik Synchronous Data

Replikasi untuk sinkronisasi data sebagai solusi replikasi data dikenal dengan istilah replikasi *synchronous* dimana tiap-tiap gambaran basis data ditulis secara cepat. Di dalam teknik ini, integritas data dijaga dalam solusi basis data tradisional dengan suatu fase ganda dilakukan di dalam transaksi yang mengakses tiap-tiap basis data unsur untuk ditulis dan diproses hanya ketika tiap-tiap basis data yang tersedia diperbaharui. Pada solusi basis data tradisional, replikasi *synchronous* menghapuskan perbedaan antara *master/slave* dan *peer-to-peer*

configurations. Bagaimanapun, replikasi *synchronous* peka terhadap kegagalan sistem. Jika satu basis data tidak tersedia akibat kegagalan *server*, keberhasilan dari transaksi akan ditunda.

Ketika replikasi *synchronous* memiliki integritas data, hal tersebut harus dikerjakan dengan biaya yang tersedia. Sistem ketersediaan dapat dikurangi dengan *synchronous replication* jika mata rantai antar basis data mudah pecah. Sifat mudah pecah ini pada kegagalan jaringan juga mencegah secara *synchronous* server replikasi basis data yang disebarkan secara geografis.

Terdapat dua teknik dasar untuk menjamin transaksi menghasilkan satu hasil dan tidak bergantung pada akses terhadap data atau replika data yang digunakan dalam perhitungan transaksi [2][5][6].

2) Teknik Asynchronous Data

Replikasi data *asynchronous* memiliki dua manfaat, yaitu meningkatkan pembacaan data dan pembaharuan basis data. Kegagalan *server* tidak mempengaruhi operasi dari *server* lain, tetapi akan menambah pekerjaan dari aplikasi untuk memelihara integritas data.

Ketika perubahan dibuat, basis data yang dituju akan diperbaharui, dan perubahan disebarkan kepada basis data sisanya. Dua mekanisme populer untuk penyebaran perubahan kepada basis data-basis data yang lain adalah *imbedded triggers* dan *passing change logs*. *Triggers* ditempatkan di dalam basis data dan ditambahkan biaya eksploitasi dengan melakukan transaksi di dalam basis data tersebut. *Passing change logs* juga meningkatkan biaya eksploitasi jaringan selama replikasi diperbaharui, namun hal tersebut akan mendapatkan lebih sedikit gangguan pada basis data lokal.

Replikasi *asynchronous* juga mengijinkan basis data untuk terisolasi secara geografis, selama masing-masing lokasi mempunyai versi sendiri tentang data tersebut. Pada replikasi *asynchronous* terdapat dua pendekatan, yaitu replikasi asinkron situs primer dan replikasi asinkron *peer-to-peer* [1].

1. Replikasi asinkron situs primer

Replikasi asinkron situs primer memiliki satu replika yang dianggap sebagai master atau data primer. Replika lainnya disebut replika sekunder. Tidak seperti replika primer, replika sekunder tidak dapat di-*update* langsung. Mekanisme pemilihan replika primer dan sekunder melalui mekanisme pendaftaran oleh pengguna dan penentuan relasi di situs tertentu yang dijadikan replika primer dan replika lainnya harus mengacunya.

2. Replikasi asinkron peer-to-peer

Pada replikasi asinkron *peer-to-peer,* beberapa replika bisa di-*update* (mungkin tidak semua) dan dijadikan replika master. Dalam hal terjadi konflik karena masalah keterlambatan propagasi, maka harus diterapkan salah satu strategi penanganan konflik. Secara umum konflik biasanya dapat diselesaikan, bahkan lebih sering tidak terjadi konflik, sehingga jenis replikasi ini banyak digunakan. Salah satu strategi pencegah konflik adalah waktu proses transaksi *update* tidak bersamaan dan pada satu saat hanya dilakukan terhadap salah satu replika (yang lain tidak dapat di-*update*), kemudian perubahan terhadap replika master itu dipropagasi ke replika yang lain. Jika ada kegagalan *update* terhadap salah satu replika, maka diambil alih oleh salah satu replika yang lain yang biasa dijadikan sebagai *backup*.

Berdasarkan uraian di atas, dapat disimpulkan bahwa dalam replikasi *asynchronous* sistem akan menyediakan suatu *copy* dari replikasi data pada beberapa node sehingga *local server* dapat mengakses data tanpa harus keluar dari jaringan lokal.

III. PENELITIAN TERKINI

Dalam mengukur tingkat pendistribusian data untuk mencegah kongesti maka dilakukan pembahsan mengenai strategi replikasi. Penanganan kemacetan arus data pada traffic jaringan, adapun beberapa contoh pendekatan yang dijabarkan dalam paper ini adalah sebagai berikut:

A. Permasalahan Kongesti

Congestion problem terjadi ketika sebuah jaringan mempunyai beban yang banyak dan mengakibatkan performanya menurun atau lambat dengan kata lain jumlah pengiriman data melebihi kapasitas perangkat keras yang ada [8]. Kontrol kemacetan berkaitan dengan pengalokasian sumber daya dalam jaringan sehingga jaringan dapat beroperasi pada tingkat kinerja yang sesuai ketika permintaan melebihi kapasitas sumber jaringan. Sumber daya ini termasuk *bandwidth, buffer* (memori) dan kapasitas pengolahan pada node *intermediate* [9]. Tanpa mekanisme kontrol kongesti yang tepat, *throughput* jaringan dapat berkurang jauh karena beban trafik yang besar. Menurut peneliti sebelumnya [1] [2] [3] [4] [5] kongesti pada lalu lintas jaringan disebabkan oleh:

1) *Terlalu banyak host yang melakukan transmisi*, *host* adalah alat yang terhubung ke jaringan dan dapat menerima dan mengirimkan informasi dari alat ke alat lainnya dalam jaringan tersebut. *Broadcast* domain adalah kumpulan dari alat-alat pada sebuah segmen jaringan yang menerima paket *broadcast* yang dikirim

oleh alat lain dalam segmen jaringan tersebut.

2) *Broadcast Storm*, terjadi karena semua alat mengirimkan paket *broadcast* ke seluruh alat lain melalui jaringan. Semakin banyak *host* maka semakin besar *broadcast storm*.
3) *Volume akses*, jika dalam satu jaringan terdapat banyak komputer dimana setiap komputer mengakses beberapa halaman situs web bervolume tinggi dalam satu waktu, maka besar kemungkinan akan terjadi kemacetan *bandwith*. Jalur yang kecil akan membuat lalu lintas jaringan padat jika dilewati oleh banyak data dalam satu periode.
4) *Data Collision*, yaitu suatu kondisi jaringan dimana sebuah alat mengirimkan paket data ke sebuah segmen jaringan yang kemudian memaksa semua alat lain yang ada di segmen jaringan tersebut untuk memperhatikan paketnya. Pada saat yang bersamaan alat yang berbeda mencoba untuk mengirimkan paket yang lain, yang mengakibatkan tabrakan *(collision)*, paket yang dikirim menjadi rusak akibatnya semua alat harus melakukan pengiriman ulang paket.
5) *Biaya overhead bandwith*, media jaringan dengan *bandwith* kecil tidak seimbang dengan banyaknya lalu lintas data yang terjadi sehingga dapat mengakibatkan *overhead*.

B. Mekanisme Pencegahan Kongesti

Beberapa mekanisme untuk mencegah kongesti dengan *SDN controller* di data center yang dibahas dalam masing-masing paper yang direview adalah sebagai berikut:

1. SDN-based Data replication Tested topology

Mekanisme yang diusulkan oleh M. Gholami, dkk [5] penggunaan *SDN controller* berawal dari kinerja jaringan data center sangat penting karena hosting aplikasi dan layanan *cloud* yang berbeda. Kinerja jaringan tersebut dipengaruhi oleh banyak faktor, seperti desain jaringan dan mekanisme kontrol kongesti yang digunakan dalam jaringan. Pada saat ini, jaringan data center memiliki volume yang tinggi dan arus lalu lintas yang dinamis karena penyebaran layanan *cloud* yang membuatnya lebih menantang untuk mengendalikan kemacetan dengan cara yang lebih efisien. Dengan munculnya Software Defined Networking (SDN), paradigma jaringan dapat diketahui untuk generasi baru dari jaringan data center melalui pemisahan kontrol dan data pesawat. Dalam tulisan ini kami M. Gholami dkk. [5], mengusulkan metode yang efisien untuk mengontrol kemacetan dalam perangkat lunak yang didefinisikan jaringan data center berdasarkan protokol OpenFlow.

Skema data center berbasis controller OpenDaylight (ODL) dimodifikasi menggunakan 2 alur dari openflow, yaitu:

a) Topologi konvensional
   Yang diterapkan pada jaringan data center adalah Fat-tree, yang mana Top-of-Rack (ToR) beberapa switches yang beralih pada lapisan akses lalu lintas rute ke lapisan inti (Core), modul deteksi kongesti dan komponen routing dari OpenFlow *controller*.

b) Komponen *Congestion Control*
   Untuk mengontol kongesti pada protokol OpenFlow terdapat dua modul: *congestion detection module* dan *control module*. Prototipe *congestion control* diimplementasikan menggunakan Floodlight *controller*, yang merupakan komponen OpenFlow controller.

Pada Openflow-Rackswitch (OF-Rswitch) dalam protokol *Openflow*, setelah menerima pesan pemberitahuan yang berisi antrian *site-site flow* dengan ukuran melebihi *bandwidth* kemudian OF-Rswitch mengidentifikasi tumpukan dan informasi port kemudian diteruskan ke kontroler melalui protokol *Openflow* [8][9].

Kelemahan metode ini hanya fokus pada biaya *bandwidth* yang mengalami *overhead*, sementara itu metode ini tidak melihat sisi lain untuk mencegah terjadinya kepadatan jaringan yang sangat besar yang akan memicu terjadinya kongesti. Dalam metode ini, kami mengusulkan, pemantauan kemacetan di link jaringan dideteksi dengan memantau pusat statistik port dari switch OpenFlow yang diaktifkan dan kemudian beberapa dari arus di link padat yang dialihkan melalui jalur dengan sumber daya lebih bebas dengan kontroler OpenFlow. Kami menerapkan metode yang diusulkan oleh Mininet. Hasil percobaan menunjukkan efisiensi metode yang diusulkan dapat menurunkan kemacetan dan meningkatkan kinerja jaringan.

Keuntungan menggunakan metode ini adalah mampu mengatasi masalah kemacetan pada jaringan yang lingkungan jalur lintasan transmisi packetnya telah ditentukan. Pendefinisian ODL untuk masalah kongesti dengan mempertimbangkan tingkat lalu lintas dan ukuran *bandwidth*. Untuk membuktikan apakah penanganan kongesti sudah efisien atau belum, digunakan metode algoritma Q-learning routing, kami

melakukan simulasi untuk metode yang diusulkan. Hasil menunjukan metode yang kami usulkan dapat memecahkan masalah kemacetan jaringan.

2. Software-defined Networking (SDN) dan OpenFlow protocol

Mekanisme yang diusulkan S Feki, dkk [3] berawal dari masalah tidak efisien dan ketahanan yang rendah pada komunikasi *multicast* di lingkungan SDN. *Multicast* menawarkan cara yang elegan dalam membangun kelompok komunikasi antar pengguna dengan menggunakan konsep kelompok *multicast*. Klien dapat bergabung dan meninggalkan kelompok untuk mengirim atau menerima data dari anggota grup lainnya. Selain itu mekanisme *multicast* memastikan bahwa strategi yang digunakan sudah efisien untuk menyampaikan paket data ke semua anggota secara bersamaan sehingga dapat mengurangi terjadinya kongesti [8]. V Mann dkk mengajukan mekanisme berbasis karakteristik pusat data. Manajemen kemacetan ethernet harus mempertimbangkan isu-isu seperti *delay* yang rendah, *packet loss* yang rendah dan *throughput* yang tinggi dari jaringan. Oleh karena itu, pusat data perlu metode yang fleksibel untuk mengontrol kongesti. Masalah administrasi menyebabkan biaya OPEX yang sangat tinggi di data center karena sejumlah besar lalu lintas komunikasi. Diantaranya, banyaknya lalu lintas komunikasi internet yang terjadi di data center dan kurangnya efisiensi kualitas layanan terutama manajemen kongesti ethernet.

a) Algoritma Dijkstra di ODL untuk masalah Routing

Salah satu kontroler pada SDN adalah Opendaylight, ODL menggunakan modul tabel routing dari algoritma Dijkstra. Algoritma Dijkstra adalah sebuah algoritma untuk menghitung jalur terpendek antara sumber dan tujuan dalam skema yang ditetapkan. Dalam ODL, penghitungan awal yaitu pada jalur terpendek antara sumber ke tujuan dan tabel arus pada Tabel OVS (OpenVSwitch) informasinya dimodifikasi menggunakan API Southbound berdasarkan protokol OpenFlow. Ketika terjadi traffic, maka jalurnya akan mengikuti jalur yang telah ditetapkan dalam Tabel Arus.

b) Pencegahan Kongesti

Untuk mencegah kongesti/kemacetan pada jaringan Q-learning yang diusulkan, mekanisme pencegahan menggunakan komposisi jaringan topologi diusulkan. Sistem pencegahan dilakukan dengan mengatur sejumlah besar lalu lintas paket yang melalui host1 ke Host6. Oleh karena itu, pada OVS1 untuk OVS3 ke jalan OVS6 dilakukan pengontrolan menggunakan algoritma Dijkstra. Namun, jika terjadi masalah serius pada saat pengiriman data pada Host3 ke Host7, maka kepadatan arus lalu lintas jaringan terjadi pada OVS3 dan OVS6, yang mengarah ke biaya *overhead bandwidth* jaringan sehingga terjadi kemacetan. Tindakan pencegahan ini untuk mempertahankan status jaringan seefisien mungkin.

c) Algorithm Q-learning Routing

Algoritma Q-learning Routing diusulkan untuk mengatasi kemacetan jaringan dengan mempertimbangkan biaya *overhead* jaringan, dalam kasus ini di mana sejumlah besar lalu lintas tiba-tiba melewati jaringan. Jalur alternatif dibuat menggunakan algoritma routing Q-learning untuk mengefisienkan jalur pengiriman paket dari sumber ke tujuan.

3. Multi-Routed Data Center Congestion Control (MRDCCC)

R Kanagavelu dkk [1] mengusulkan *MRDCCC* dengan tujuan untuk meminimalkan kongesti dalam lalu lintas jaringan pusat data, agar beban tersebar merata, sehingga mencapai *latency* yang rendah dan *troughput* yang tinggi pada proses replikasi data. Untuk mencapai tujuan ini, R Kanagavelu dkk menyiapkan ketersediaan data di pusat data yang menjadi tantangan penulis dalam paper ini, dimana sejumlah besar server dengan kinerja tinggi terhubung pada switch jaringan. Data replikasi pada pusat data merupakan cara yang efektif untuk melindungi data yang redundan, sehinnga dapat memastikan adanya salinan ketika terjadi *failure*/kegagalan [9] pada *switch* SDN. Dengan demikian, jumlah arus traffic pada server sumber ke server tujuan dapat dinilai secara akurat.

Jaringan berbasis SDN ini menggunakan *multi-routed data center* dikatakan demikian karena terdiri dari tiga switch, yaitu TOR, *aggregate*, dan *core* (inti switch). Cara replikasi melakukan operasi yaitu melalui pods, dimana salinan situs utamanya disimpan dalam server begitu pula dengan situs cadangan. Maka operasi replikasi data ini memiliki 2 (dua) aliran (flow), untuk aliran pertama itu antara host dan server utama, aliran lainnya host dan server *backup*. Untuk metode yang diusulkan dalam paper ini ada 2 metode:

a) Adaptive routing method

Metode routing yang adaptif merupakan kunci kerangka operasi dari replikasi data. Prosesnya dimulai dengan memilih rute yang tepat untuk arus berdasarkan status jaringan saat ini, lalu memantau statistik beban pada

switch dan berdasarkan pada beban pada jalan yang dilalui memilih jalur biaya minimum serta mencoba untuk menyeimbangkan beban dan mengurangi kemacetan jaringan yang mengarah ke *throughput* yang tinggi.

b) Function module

Untuk replikasi data jarak jauh digunakan jaringan yaitu *control network* (menggunakan NOX *controller*) dan pengujian memakai algoritma Adaptive Routing. Network Under Test (NUT) digunakan untuk menyimpan salinan arus data. Komponen fungsional modul menggunakan dua sistem penyimpanan yaitu *'Primary server'* dan *' Back-up Server'* dari jalur jaringan yang berbeda dari NUT ke host replikasi data. NOX *controller* memiliki tombol fungsional modul yaitu (a) memantau dan (b) routing fungsional modul. Kunci dari algoritma ini adalah penghitungan paket dan aliran informasi dari modul. Memantau secara bersama kapasitas dari setiap link sepanjang jalur yang akan digunakan untuk menghitung path yang dimuat.

Kelebihan penggunaan metode ini berada pada penanganan masalah replikasi data di pusat data center (Data Center) jaringan. Pengembangkan jaringan berbasis SDN dengan kerangka yang dibuat untuk arus rute (itu terkait dengan operasi replikasi) mempertimbangkan beban arus jaringan yang akan menyebabkan *throughput* yang tinggi. Percobaan ini memonitor link dan beban *path* yang memberikan efektivitas skema kerangka kerja yang baik untuk arus lalu lintas yang lebih efisien sehingga mencapai *throughput* yang tinggi.

4. NCP OpenFlow Controller Module Floodlight (OFCMF)

Sistem pada perusahaan menggunakan teknologi replikasi untuk menjaga beberapa aplikasinya atau menyimpan data agar tetap tersinkronisasi. Dengan metode yang diusulkan oleh V Mann, dkk [2] melalui makalahnya mengatakan NCP *OFCMF* merupakan sistem yang menggunakan jaringan berbasis replikasi untuk mengaktifkan layanan replikasi data pusat (DC) melalui Software Defined Network. Adapun teknik replikasi data pada metode ini berbasis pada penyimpanan dan biasanya menerapkan granularitas waktu. Selain itu, penyimpanan tidak dapat digunakan untuk mencapai replikasi layanan yang membutuhkan status sinkronisasi selain sinkronisasi disk.

Pada metode ini, waktu yang dibutuhkan untuk meretransmisi lebih lama sehingga rawan terjadi *overhead* atau redundansi komunikasi. Metode yang digunakan untuk mendukung sistem kerja dari NCP adalah sebagai berikut [2]:

a) *NCP OpenFlow Controller*

Kami mengembangkan layanan kontroler NCP OpenFlow sebagai modul dalam FloodLight OpenFlow controller. NCP merupakan Layanan kontroler yang memungkinkan penggunanya untuk mengidentifikasi arus (berdasarkan pada sumber dan tujuan utama alamat dan port) dan menentukan lokasi *middlebox* dan target replikasi untuk setiap aliran. Setelah pengguna menyimpan preferensinya, kontroler kemudian secara otomatis menentukan *switch* yang ideal (disebut sebagai yang 'RedirectionSwitch ") dalam jaringan yang harus mengarahkan paket untuk setiap aliran berdasarkan lokasi dari sumber utama ke tujuan, *middlebox* dan target replikasi di jaringan topologi.

b) *NCP Middlebox Appliance*

NCP *middlebox appliance* merupakan sebuah alat menangkap paket yang diarahkan ke mesin *middlebox* baik menggunakan *socket* tingkat pengguna ataupun menggunakan pustaka *capture* paket seperti libcap. *Socket* tingkat pengguna dapat digunakan hanya jika alamat tujuan dari paket yang diterima oleh *middlebox* telah dimodifikasi oleh *middlebox* sendiri untuk pengalihan oleh switch pengalihan atau sebagai alat kernel *middlebox* melalui penggunaan alat-alat seperti *iptables* dan *ebtables*. NCP *middlebox* merekonstruksi pesanan paket yang diambil pada saat sesi jaringan. Mirip dengan paket *capture*, *middlebox* memiliki pilihan untuk mengirim paket mentah TCP atau untuk mengirim paket muatan lebih dari *socket* TCP sesuai tingkat pengguna.

Keuntungan dari penggunaan metode ini adalah NCP menyajikan sebuah sistem yang menggunakan replikasi berbasis jaringan untuk mengaktifkan layanan replikasi di pusat-pusat data melalui *software defined networking*. NCP memungkinkan penggunanya untuk mengidentifikasi arus berdasarkan alamat jaringan dan port dan untuk menentukan target replikasi untuk setiap aliran arus. Evaluasi eksperimental terhadap NCP menunjukkan bahwa

TABEL I
KELOMPOK KESALAHAN YANG DIVALIDASII [4]

Jumlah Skenario	*Nature*	Kesalahan	Validasi
T_1	*Reactive*	*Either C, or N, or both*	√
T_2	*Proactive*	*Either C or N, or both but C≠N*	√
T_3	*Proactive*	*Both C and N where C = N*	√

jaringan berbasis replikasi dapat mengaktifkan replikasi layanan yang *scalable* terhadap waktu secara real dengan *overhead* yang minimal [8] [9].

5. *Openflow Controler Validation*

K Mahajan [4] mengusulkan sebuah metode SDN kontroler yang terpusat secara logis untuk mengatur arus yang terjadi dalam control plane. Tujuan dari paper ini adalah mengindentifikasi dan mengontrol beberapa gabungan golongan *fault* yang terjadi di SDN pada saat pengelompokan kontroler. Namun pada kenyataannya, SDN mengendalikan entitas pendistribusian, yang mungkin menunjukkan *behaviour* yang berbeda pada *event triggers*. Paper ini mengusulkan JURY yaitu sebuah sistem untuk memvalidasi kegiatan pengontrolan dalam mengelompokkan penyebaran SDN dengan melibatkan topologi dan status *forwarding*, tanpa membatasi *behaviour* kontroler [7].

Metode yang digunakan dalam paper yang diteliti ini menggunakan istilah SDN untuk merujuk kepada SDN berbasis *OpenFlow*, dan menggunakan istilah Cluster HA (*High Avaibility*) yang melakukan pengelompokan kontroler mengikuti standar arsitektur HA: Aktif-Pasif, dimana semua switch yang menghubungkan pengontrol tunggal dan lain-lain adalah replika pasif, dan Aktif-Aktif dimana jaringan dipartisi sedemikian rupa sehingga switch di setiap partisi terhubung ke kontroler yang berbeda di dalam *cluster*.

Pada makalah ini, peneliti telah melakukan evaluasi dengan 3 replikasi untuk menentukan konsensus dan status replikasi yang dapat digunakan. Hasilnya dapat dilihat pada Tabel 1. Begitu juga dengan formula yang digunakan dimana T_1 adalah jumlah *track action*, T_2 adalah mengamati kondisi dari T_1 yang akan berdampak padanya lalu pada arus replikasi ditempatkan pada T_3 dan *JURY* adalah melakukan validasi atas *fault* yang dideteksi. Statement problem dari klaster kontroler ditunjukkan pada Persamaan (1).

$$NON\ FAULTY\ (C) \Leftrightarrow \forall \tau, \psi\ \{A_c = CONSENSUS(A_1, \dots, A_k)\} \quad (1)$$

Untuk JURY sendiri mengikuti konsistensi pengendali jaringan yang berbeda untuk semua *triggers* yang direplikasikan agar mengalami konsensus validasi dan status transisi yang sama.

Keuntungan metode ini adalah metode yang diusulkan ini dapat menemukan kesalahan pada SDN. Hasil evaluasi menunjukkan JURY membutuhkan perubahan minimal untuk mengendalikan SDN untuk penyebaran, dan mampu memvalidasi tindakan kontroler mendekati *real time* dengan kinerja *overhead* yang rendah.

IV. KESIMPULAN

Beberapa mekanisme penanganan masalah kongesti dengan menggunakan kontroler dari SDN pada jaringan data center telah disajikan pada makalah ini. Penanganan masalah kongesti secara tepat diperlukan untuk menjamin performa dari *software-defined networking* dengan karakteristik kontrol terpusat pada data center. Mekanisme tersebut telah dibandingkan dengan kriteria yang sudah ditunjukan pada Tabel I. Sistem ini merupakan sistem yang dirancang berbasis jaringan untuk replikasi data untuk masing–masing kontroler sehingga dapat membantu dan mempermudah dalam akses data dan informasi. Sistem ini mampu melakukan replikasi data antara *openflow* NOX dan data center sehingga terjadi penyetaraan data serta pemusatan data dalam satu server. Melalui survei ini penulis berharap dapat memberi kontribusi dalam pengembangan *congestion control* lainnya dalam lingkungan SDN yang lebih efektif, efisien dan komprehensif.

Penanganan permasalahan kongesti tidak harus terpaku kepada satu skema saja. Pada beberapa penelitian sebelumnya, penanganan yang baik untuk replikasi data dapat memaksimalkan konsistensi data, meminimalkan kebutuhan jaringan dan menghindari beberapa masalah yang sudah dibahas pada paper ini. Untuk penelitian lebih lanjut, dapat menggunakan metode dengan menerapkan skema baru yaitu menggabungkan keunggulan-keunggulan dari skema yang sudah ada. Sebagai contoh penggunaan skema kontroler NOX yang digabungkan dengan ODL. Replikasi data bukanlah solusi terbaik untuk menangani kongesti pada jaringan data center karena masih memiliki kelemahan-kelemahan yang berisiko. Namun, meskipun begitu replikasi data bukanlah solusi yang buruk karena masih memiliki keunggulan-keunggulan di sisi lain.

DAFTAR PUSTAKA

[1] Renuga Kanagavelu, Bu Sung Lee, Rodel Felipe Miguel, Le Nguyen The Dat, Luke Ng Mingjie, "*Software Defined Network based Adaptive Routing for Data Replication in Data Centers*," *IEEE International Conference on Software Maintenance*, pp. 2004-2013, 2007.

[2] Vijay Mann, Kalapriya Kannan, Anilkumar Vishnoi, and Aakash S, "NCP: Service Replication in Data Centers through Software Defined Networking," *IFIP/IEEE International Symposium on Integrated Network Management (IM2013): Mini-Conference*, IBM Research-India 2013.

[3] S Feki dkk,"*Q-leraning based Data Replication for Highly Dynamic Distributed Hash Tables*," *IEEE WCRE*, pp. 112-117, 2014.

[4] K Mahajan., "JURY: *Validating Controller Actions in Software-Defined Networks*," *IEEE Transaction on Software Engineering*, vol. 39, no. 8, pp. 1144-1156, 2013.

[5] M Gholami., "*Congestion Control in Software Defined Data Center Networks Through Flow Rerouting*," *ELSEVIER The journal of Systems and Software* , pp. 2639-2653, 2013.

[6] ORACLE whitepapaer, *Data Protection Strategies in today's Data Center*, 2012.

[7] US: Open Networking Foundation. 2013. *Software-defined networking*: The new norm for networks. ONF White Paper. Palo Alto.

[8] Khondoker, R., Zaalouk, A., Marx, R & Bayarou, K. 2014. *Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers*.

[9] Dot Hill Corporation, Tech. Rep, *Secure Data Protection With Dot Hills Batch Remote Replication*, 2009