Design High Throughput Data Center Topology Network berbasis *Random Graph*

Aziza Hayupratiwi[1], Sidik Prabowo[2], Muhammad Arief Nugroho[3] [1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]azizahayupratiwi@students.telkomuniversity.ac.id, [2]pakwowo@telkomuniversity.ac.id, [3]arif.nugroho@telkomuniversity.ac.id

Abstrak

Salah satu peningkatan kinerja dalam *Data Center* (DC) adalah dengan membuat desain topologi pada *Data Center Networks* (DCNs). Pengaruh desain *network* dalam *data center* akan berpengaruh terhadap penging- katan nilai *throughput* dalam jaringan. Dalam topologi *Data Center Networks*, desain terbagi menjadi dua yaitu *homogeneous* dan *heterogeneous*. *Homogeneous topology design* diatur dengan sejumlah *port* dan *swi- tch* yang sama dan dapat mencapai *high throughput* dengan *low cost*. Namun, pada jaringan *heterogeneous* belum dapat mencapai *high throughput*. Dalam tugas akhir ini dilakukan implementasi desain topologi pada *data center* berbasis *random graph*. Hasil yang diperoleh pada penelitian ini membuktikan bahwa ja- ringan *heterogeneous* dengan 40 dan 32 *degree* dapat mencapai *high throughput* jika dibandingkan dengan jaringan *heterogeneous* 24 dan 8 *degree* serta jaringan *homogeneous* 16*degree*.

Kata Kunci: *Throughput, Data Center Networks, Homogeneous, Heterogeneous,*

***Topology Design*. Abstract**

One of the Performance Improvements in Data Centers (DC) is to create topology designs in Data Center Networks (DCNs). The influence of network design in the data center will affect the increase in value of throughput in the network. In the Data Center Network topology, the design is divided into two hetero- geneous and heterogeneous forms. Homogeneous topology design with the same portfolios added and can achieve high throughput with low costs. However, the heterogeneous network has not been able to achieve high throughput. In this final project, the implementation of topological data design in the basis of random graphs. The results obtained in this study prove that a heterogeneous network with 40 and 32 can achieve high throughput if compared to a heterogeneous network of 24 and 8 degrees and a homogeneous network of 16 degrees

Keywords: Throughput, Data Center Networks, Homogeneous, Heterogeneous, Topology Design

1. Pendahuluan

Latar Belakang

Data Center Networks (DCNs) merupakan ruangan yang berisi komponen penting dalam infrastuktur jaring- an, seperti *rack server*, *hardisk*, *switch*, yang mendukung jaringan Internet dalam dunia digital sebagai tempat penyimpanan data terpusat. Pada DCN terdapat performansi matriks untuk menunjukkan kemampuan suatu per- angkat dalam mengatasi beban DCN. Terdapat beberapa matriks yang digunakan untuk mengukur performansi suatu perangkat, yaitu *throughput*, *frame loss*, *latency*, *connection processing rate*, dan *Concurrent Connections* [1].

Pada dasarnya, efisiensi dari *data center* bergantung pada *high capacity network* untuk memastikan bahwa komputasinya tidak menyebabkan *bottleneck* [16]. Dari karakteristik *high capacity network* adalah *high throu- ghput* yang sangat dibutuhkan pada DCN untuk *transfer rate* secepat mungkin [7]. *High throughput* dibutuhkan pada *data center* untuk menangani sebuah data yang besar pada suatu jaringan [13]. Salah satu metode untuk mencapai *throughput* yang maksimal yaitu dengan desain topologi *data center networks* [15].

Terdapat beberapa topologi pada *data center* [10] yang digunakan, antara lain *Fat-Tree* [14], *Jellyfish* [17], *F10* [11], dan *Xpander* [18]. Diantara topologi diatas, hanya Jellyfish [15] dan Xpander [18] yang dapat mencapai *high throughput* mendekati optimal. *Jellyfish* merupakan topologi yang dibangun berdasarkan *random graph* diantara *switch top-of-rack* (ToR), yang akan membangun koneksi antar

host secara acak yang dihasilkan dari probabilitas distribusi [17]. Sedangkan *Xpander* merupakan topologi turunan *Jellyfish* dengan mengelompokkan *host* men- jadi beberapa *metanode*. Setiap *metanode* memiliki jumlah ToR yang sama, namun satu *metanode* tidak saling terhubung secara langsung [18].

Penggunaan *random graph* untuk desain topologi *data center* pada jaringan konvensional sudah pernah digunakan [7, 15, 17] dikarenakan arsitektur *random graph* sangat fleksibel untuk *data center networks*, seperti memungkinkan untuk *high capacity, short path, high bandwidth* dan *high throughput*. *Random graph* merupakan grafik yang dihasilkan dengan probabilitas tepi terdistribusi. Dengan menghitung probabilitas terdistribusi, diha- rapkan mampu mengatasi desain topologi pada *data center* untuk mencapai *high throughput.*

Topik dan Batasannya

Desain topologi terbagi menjadi dua, yaitu *homogeneous topology design* dan *heterogeneous topology design* [16]. Masalah *high throughput* pada *homogeneous topology design* dapat diatasi dengan *Jellyfish* [16]. Namun, pada *heterogeneous topology design* masalah tersebut belum dapat diatasi. Hal ini disebabkan perbedaan jumlah port yang digunakan pada *switch* di jaringan heterogen yang mempengaruhi konektivitas antar *switch*.

Tugas akhir ini melakukan desain topologi pada *data center* berbasis *random graph* dengan membandingkan *homogeneous topology design* dan *heterogeneous topology design* untuk mencapai *high throughput* dengan menggunakan emulator mininet untuk simulasi jaringan *Software Defined Network* (SDN) dan *OpenDayLight* sebagai *controller*. Jumlah *host* yang digunakan pada tugas akhir sebanyak 320 host dengan 20 switch. Untuk *heterogeneous topology*, digunakan 2 *cluster switch* yang mana setiap *cluster* memiliki jumlah port berbeda, yaitu 24 dan 8 port. Setiap jalur antar host diberikan *bandwidth* sebesar 100 Mbps.

Terbatasnya jumlah host yang digunakan disebabkan karena emulator *mininet* tidak dapat menampung lebih dari 400 host. Untuk dapat menjalankan topologi pada tugas akhir ini, diperlukan waktu minimal 60 menit dan maksimal 24 jam. Hal tersebut dipengaruhi performansi dari komputer dan *controller* yang digunakan.

Tujuan

Untuk menganalisis apakah suatu topologi dapat mencapai *high throughput*, akan dibangun topologi berbasis *random graph* pada emulator mininet dengan *bandwidth* sebesar 100 Mbps pada setiap jalur host. Suatu jaringan dapat dikatakan mencapai *high throughput* apabila *throughput* yang dihasilkan mendekati *bandwidth*.

2. Dasar Teori

2.1 Data Center Topology Design

Data center topology merupakan arsitektur pembangunan *data center* yang mana tata letak dan teknologi yang digunakan untuk membantu kebutuhan *data center* dalam menangani sistem jaringan dan penyimpanan pusat data. Desain suatu topologi sangat berpengaruh terhadap performansi jaringan didalamnya, khususnya *throughput* pada *data center* [15].

2.1.1 Homogeneous Topology Design

Pada penelitian Angkit Singla (2014) [16], desain jaringan dibagi menjadi dua, yaitu *homogeneous* dan *heterogeneous*. *Homogeneous network* menggunakan *switch* yang semuanya diatur memiliki *line-speeds* dan jumlah port yang sama. *Jellyfish* merupakan contoh topologi yang dikembangkan dalam *homogeneous design*. *Jellyfish* merupakan topologi yang diadopsi dari *random graph* dan mampu menangani penambahan *host* secara bertahap karena sifatnya yang random. *Jellyfish* mampu memperbaiki kelemahan *Fat-tree* dengan penggunaan *switch* yang tidak banyak. Setiap *switch* memiliki r port yang terhubung ke *switch* lain dengan sisanya k = (n-r) terhubung ke host [10]. Jika *Jellyfish* dibandingkan dengan *degree-diameter graph*, *degree-diameter graph* lebih unggul dalam mencapai *high throughput*, namun *Jellyfish* tetap mampu mencapai *high throughput* sampai lebih 91% [17].

Kelemahan dari *Jellyfish* telah disempurnakan oleh *Xpander*, yang merupakan arsitektur baru pada *data center* secara deterministik dan diadopsi dari teori *expander graph.* Xpander menggunakan *switch* yang lebih sedikit dari *fat-tree* namun memperbaiki kelemahan yang sama [18]. *Xpander* lebih baik digunakan pada *data center* karena data tidak menimbulkan *bottleneck* [5].

2.1.2 Heterogeneous Topology Design

Heteregeneous network merupakan jaringan yang mana jumlah port pada *switch* yang digunakan berbeda. Pada penelitian LEGUP [3] mengenai *heterogeneous network* menggunakan *Clos topology*, jaringan heterogen dapat mencapai performansi melebihi *fat-tree* dan mengurangi *cost* pada *data center* ketika ada *upgrading* di *data center*.

2.2 Random Graph

Random graph merupakan grafik acak yang mana jumlah simpul, tepi, dan koneksinya ditentukan dengan beberapa cara acak, salah satunya yaitu dengan probabilitas distribusi uniform [19]. *Random graph* digunakan sebagai pemodelan jaringan seperti Internet, *World Wide Web*, dan jaringan biologis dan sosial [2, 6]

2.3 Software Defined Network

Software Defined Network (SDN) merupakan paradigma baru dalam arsitektur jaringan komputer yang memisahkan *control plane* dan *data plane* [9]. Pemisahan tersebut bertujuan untuk memudahkan perawatan dan pengendalian jaringan terpusat dengan protokol *OpenFlow* sebagai perantara komunikasi antara *controller* dan perangkat jaringan [12].

2.4 OpenDayLight

OpenDayLight merupakan *controller* pada SDN yang bersifat *open source* dengan *platform* pengontrol modular yang menyediakan monitoring jaringan. *OpenDayLight* dibangun dengan bahasa pemrograman Java sehingga dapat digunakan pada semua *platform*, baik perangkat keras maupun *operating system* yang mendukung Java [8].

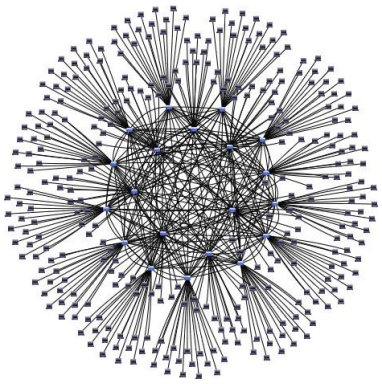
3. Sistem yang Dibangun

3.1 Jellyfish

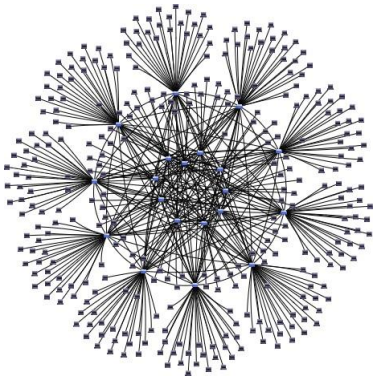
Jellyfish adalah topologi yang dibangun berdasarkan *random graph* diantara *switch top-of-rack* (ToR), yang akan membangun koneksi antar *host* secara acak yang dihasilkan dari probabilitas terdistribusi [13]. Pada umumnya, *Jellyfish* membangun koneksi antar *host* dengan jumlah derajat yang sama pada setiap *switch*, disebut *homogeneous netwrok*. Karena sifat *Jellyfish* yang fleksibel [17], dapat dimodifikasi dengan mengatur jumlah derajat yang berbeda pada setiap *switch* yang disebut *heterogeneous network*.

Homogeneous dan *heterogeneous* dibangun dengan notasi *Random Regular Graph* atau RRG(N,k,r) [15]. N *switch* akan membangun koneksi antar N sebanyak r dan memiliki jumlah *port* untuk masing-masing *switch* sebanyak k yang terkoneksi ke k server. Jumlah keseluruhan *host* pada jaringan dibangun sebanyak $N(k-r)$. *Pseudo code Jellyfish* dapat dilihat pada Lampiran.

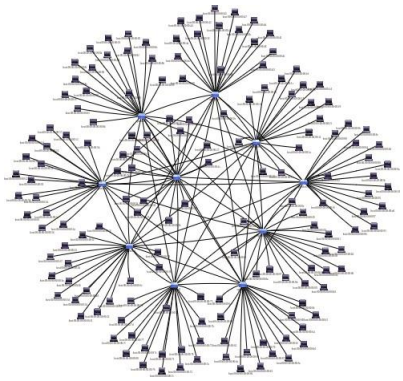
Pada penelitian ini, jumlah *host* yang digunakan yaitu 320 *host* dengan 20 *switch*, 160 *host* dengan 10 *switch*, dan 96 *host* dengan 6 *switch*. Untuk *heterogeneous network*, digunakan 2 *cluster switch* yang mana setiap *cluster* memiliki jumlah *port* berbeda, yaitu 40 dan 8 *port*, 32 dan 8 *port*, serta 24 dan 8 *port*. Setiap jalur antar host diberikan *bandwidth* sebesar 100 Mbps.



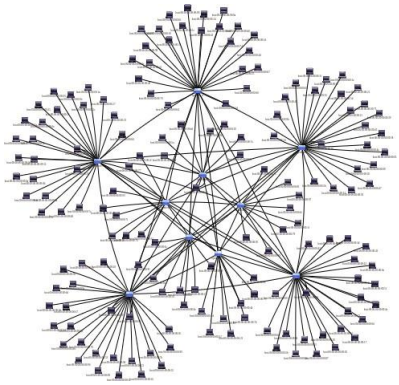
Gambar 1. Homogeneous Network 320 host



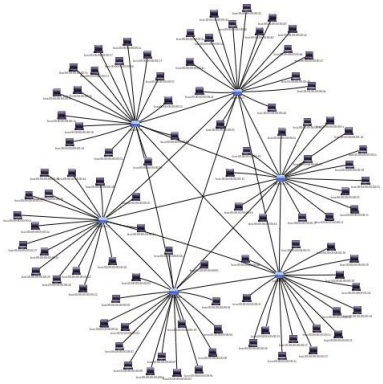
Gambar 2. Heterogeneous Network 320 host



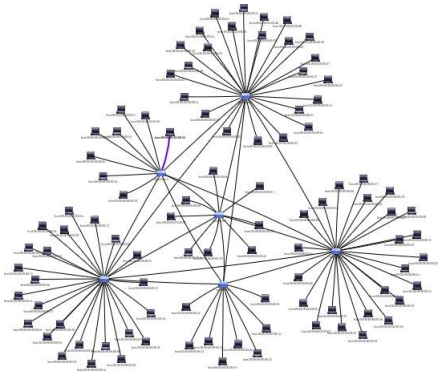
Gambar 3. Homogeneous Network 160 host



Gambar 4. Heterogeneous Network 160 host



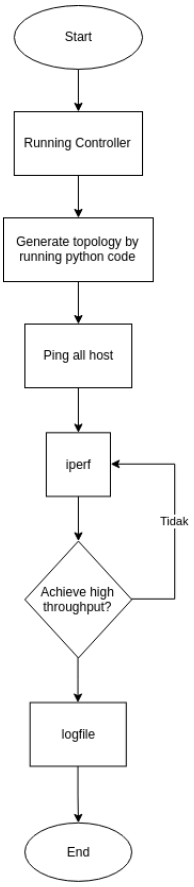
Gambar 5. Homogeneous Network 96 host



Gambar 6. Heterogeneous Network 96 host

3.2 Skema Pengujian

Berikut skema untuk melakukan simulasi pengujian *high throughput* pada topologi *data center*:



Gambar 7. Skema Pengujian

Alur pengujian dilakukan dengan menjalankan *controller* dan *generate* topologi yang menggunakan bahasa pemrograman *python* dengan emulator mininet dan terhubung dengan *controller*. Lakukan *ping* ke seluruh *host* tanpa ada *packet loss*. Jalankan *iperf* untuk mengukur *throughput* pada masing-masing *host*. Data *throughput* akan disimpan dalam bentuk *file* .txt.

3.3 Skenario Simulasi

Skenario simulasi akan dengan 3 pengujian pada topologi yang masing-masing memiliki *host* sebanyak 320 *host* dengan 20 *switch*, 160 *host* dengan 10 *switch*, dan 96 *host* dengan *6* switch. Tujuannya adalah untuk mengetahui topologi mana yang mencapai *high throughput.* Suatu jaringan dapat dikatakan *high throughput* jika nilainya mendekati *bandwidth* [4].

1. *Heterogeneous Network: 40 degree.*

   Salah satu *host* pada *switch* yang memiliki 40 *port server* digunakan sebagai *iperf server* untuk melakukan *ping* ke seluruh *host*. Seluruh *link* pada topologi diberikan *bandwidth* sebesar 100 Mbps. Hal ini bertujuan untuk mengetahui besar *throughput* yang dihasilkan ketika *iperf server* berada pada 40 *degree*.

2. *Heterogeneous Network: 32 degree.*

   Salah satu *host* pada *switch* yang memiliki 32 *port server* digunakan sebagai *iperf server* untuk melakukan *ping* ke seluruh *host*. Seluruh *link* pada topologi diberikan *bandwidth* sebesar 100 Mbps. Hal ini bertujuan untuk mengetahui besar *throughput* yang dihasilkan ketika *iperf server* berada pada 32 *degree*.

3. *Heterogeneous Network: 24 degree.*

   Salah satu *host* pada *switch* yang memiliki 24 *port server* digunakan sebagai *iperf server* untuk melakukan *ping* ke seluruh *host*. Seluruh *link* pada topologi diberikan *bandwidth* sebesar 100 Mbps. Hal ini bertujuan untuk mengetahui besar *throughput* yang dihasilkan ketika *iperf server* berada pada 24 *degree*.

4. *Heterogeneous Network: 8 degree.*

   Salah satu *host* pada *switch* yang memiliki 8 *port server* digunakan sebagai *iperf server* untuk melakukan *ping* ke seluruh *host*. Seluruh *link* pada topologi diberikan *bandwidth* sebesar 100 Mbps. Hal ini bertujuan untuk mengetahui besar *throughput* yang dihasilkan ketika *iperf server* berada pada 8 *degree*.

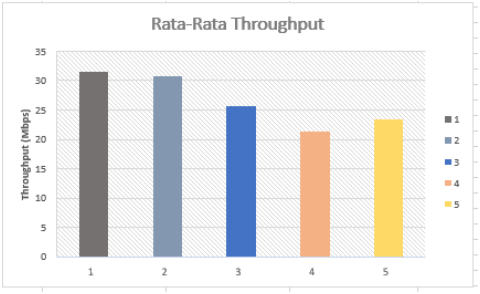
5. *Homogeneous Network.*

   Seluruh *switch* pada *Homogeneous network* memiliki 16 textitport server digunakan sebagai *iperf server* untuk melakukan *ping* ke seluruh *host*. Seluruh *link* pada topologi diberikan *bandwidth* sebesar 100 Mbps. Hal ini bertujuan untuk mengetahui besar *throughput* yang dihasilkan ketika *iperf server* berada pada 16 *degree*.

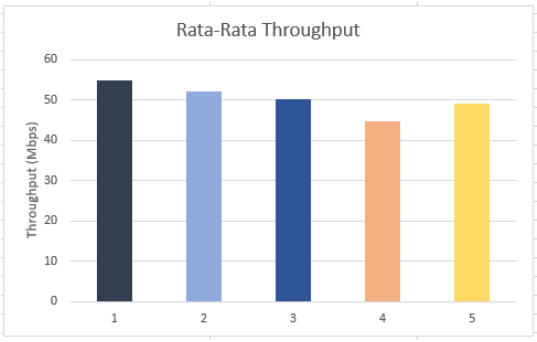
4. Evaluasi

4.1 Hasil Pengujian *Throughput*

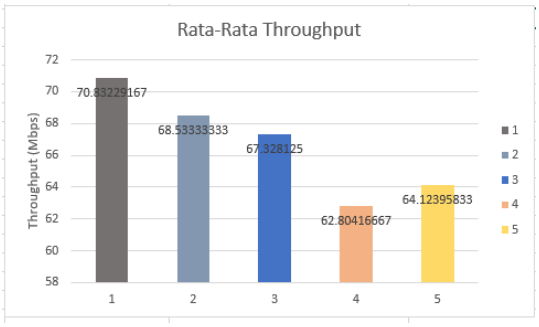
Pengambilan data *throughput* pada pengujian ini dilakukan dengan mengirim paket TCP oleh *host* yang ditentukan sebagai *test server* selama 60 detik pada setiap skenarionya. Pengujian *throughput* dilakukan sebanyak 3 kali per skenario. Gambar 8, 9, dan 10 menunjukkan nilai rata-rata *throughput* dari 320 *host*, 160 *host*, dan 96 *host*. 1, 2, 3, 4, dan 5 secara berurutan menunjukkan heterogen 40 *degree*, heterogen 32 *degree*, heterogen 24 *degree*, heterogen 8 *degree*, dan homogen 16 *degree.*



Gambar 8. Grafik Rata-Rata *Throughput* 320 *host*



Gambar 9. Grafik Rata-Rata *Throughput* 160 *host*



Gambar 10. Grafik Rata-Rata *Throughput* 92 *host*

4.2 Analisis Hasil Pengujian

Dapat dilihat pada Gambar 8, *heterogeneous network* dengan 40 dan 32 *degree* dengan 320 *host* memiliki nilai rata-rata *throughput* sebesar 31,5 dan 30,7 Mbps. Lebih besar 5,9 Mbps dibandingkan *heterogeneous network* dengan 24 dan 8 *degree* sebesar 21,3 Mbps dan hanya berbeda 8 Mbps dari *homogeneous network* yang sebesar 23,3 Mbps. Walaupun tidak menunjukkan hasil yang signifikan, perbedaan jumlah *port* suatu *switch* dapat mempengaruhi hasil *throughput* jaringan secara keseluruhan. Hal itu disebabkan setiap *host* yang berada pada *switch* yang sama bertugas sebagai *test server* akan mencapai *throughput* rata-rata sebesar 95,4 Mbps.

Berdasarkan Gambar 9, *heterogeneous network* 40 *degree* dengan 160 *host* memiliki nilai rata-rata *throughput* sebesar 54,7 Mbps, 32 *degree* sebesar 52 Mbps, dan 24 *degree* sebesar 50 Mbps. Lebih besar 10 Mbps dibandingkan *heterogeneous network* dengan 8 *degree* sebesar 44,7 Mbps dan berbeda 5,7 Mbps dari *homogeneous network* sebesar 48,9 Mbps. Sedangkan Gambar 10, *heterogeneous network* 40 *degree* 96 *host* memiliki nilai rata-rata *throughput* yang cukup tinggi mencapai 70,8 Mbps. Sedangkan*heterogeneous network* dengan 8 *degree* dan *homogeneous network* dengan 16 *degree* hanya mencapai *throughput* sebesar 62,8 dan 64,1 Mbps. Hal ini membuktikan bahwa desain heterogen topologi yang dibangun dengan 40 dan 32 *degree* mencapai *high throughput* jika dibandingkan dengan jaringan heterogen 24 dan 8 *degree*, dan jaringan *homogen* 16 degree.

5. Kesimpulan

Dengan menggunakan skema jaringan *heterogeneous*, maka *throughput* yang dihasilkan lebih baik. Semakin besar *port* yang tersedia pada suatu *switch*, maka semakin besar *throughput* yang diperoleh. Namun, penambahan *host* akan mengurangi *throughput* yang dihasilkan. Kekurangan dari penelitian ini adalah tidak menggunakan *load balancing* sebagai pemerataan paket yang dikirimkan pada setiap jalur. Sehingga hasil *throughput* yang diperoleh tidak maksimal.

Daftar Pustaka

[1] M. Arregoces and M. Portolani. *Data Center Fundamental*. CISCO, 2003.

[2] B. Bollobas. Random graphs. Cambrdige University Press, 2001.

[3] A. R. Curtis, S. Keshav, and A. Lopez-Ortiz. Legup: Using heterogeneity to reduce the cost of data center network upgrades. In *CoNEXT*. sigcomm, 2010.

[4] Datapath.io. What is network throughput?

[5] M. Dinitz. Explicit expanding expanders as datacenter topologies. Semantic Scholar, 2015.

[6] R. Durret. Random graph dynamics. Cambrdige University Press, 2006.

[7] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla. Measuring and Understanding Throughput of Network Topologies. In *29th ACM International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016.

[8] Z. K. Khattak, M. Awais, and A. Iqbal. Performance evaluation of opendaylight sdn controller. IEEE, 2014.

[9] D. Kreutz, F. M. V. Ramos, P. E. Ver´ıssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. proceeding of the ieee. IEEE, 2015.

[10] B. Lebiednik, A. Mangal, and N. Tiwari. A survey and evaluation of data center network topologies. 2016.

[11] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson. F10: A fault-tolerant engineered network. 2013.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. sigcomm, 2008.

[13] Mellanox. Big data -hadoop.

[14] Z. Qian, B. Hu, and K. L. Yeung. An efficient routing algorithm in fat-tree data center networks. 2016.

[15] A. SINGLA. *DESIGNING DATA CENTER NETWORKS FOR HIGH THROUGHPUT*. University of Illinois at Urbana-Champaign & Core, 2015.

[16] A. Singla, P. B. Godfrey, and A. Kolla. High Throughput Data Center Topology Design. In *11th USENIX Symposium on Networked Systems Design and Implementation*, 2014.

[17] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking Data Centers Randomly. In *9th USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[18] A. Valadarsky, G. Shahafy, M. Dinitz, and M. Schapira. Xpander: Towards optimal-performance datacenters. 2016.

[19] WolframMathWorld. Random graph.