# ANALISIS DAN SIMULASI PERBANDINGAN KINERJA STATELESS DAN STATEFUL FIREWALL PADA ARSITEKTUR SOFTWARE-DEFINED NETWORK

## *PERFORMANCE COMPARISON ANALYSIS AND SIMULATION OF STATELESS AND STATEFUL FILTER ON SOFTWARE-DEFINED NETWORK*

**Rahmanda Mulia[1], Novian Anggis Suwastika, S.T., M.T.[2], Muhammad Arief Nugroho, S.T., M.T.[3]**

[1,2,3] Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom
[1]rhmulia@student.telkomuniversity.ac.id, [2]anggis@telkomuniversity.ac.id,
[3]arif.nugroho@telkomuniversity.ac.id

**Abstrak**

Sistem jaringan komputer penting untuk dijaga kualitasnya agar kinerjanya tetap optimal. Kinerja yang optimal didefinisikan sebagai sistem yang memiliki *Quality of Service* (QoS) yang baik: *round-trip time* (RTT), *packet loss*, dan *jitter* yang rendah. Salah satu pengembangan dari sistem jaringan tersebut adalah *software-defined network* (SDN). SDN adalah arsitektur jaringan yang memisahkan antara *control plane* dengan *data plane* dan perangkat jaringannya dapat diatur dengan *software* tertentu. Untuk meningkatkan keamanan SDN, digunakan *firewall. Firewall* berjenis *packet filtering* yang dibahas pada penelitian ini melakukan filterisasi *packet* dengan menge-*drop packet* berdasarkan *rules* yang telah diset. *Packet filtering* mempunyai dua jenis filter, yaitu: *stateless* dan *stateful*. Proses yang berbeda terjadi pada masing-masing filter, dan berpengaruh pada kinerja sistem.Untuk mengetahui seberapa besar pengaruh proses yang terjadi pada masing-masing filter tersebut, dan untuk mengetahui filter yang paling sesuai untuk sistem yang membutuhkan kinerja yang optimal, pada tugas akhir ini dilakukan simulasi pengujian antara *stateful* dan *stateless* filter pada *firewall* dalam arsitektur SDN menggunakan QoS sebagai parameter kinerjanya. Hasil yang didapat dari analisis hasil pengujian tersebut menunjukkan bahwa *stateful filter* memiliki peforma lebih baik dengan RTT lebih rendah sebesar 10,51% dan *packet loss* lebih rendah sebesar 70% *sehingga* cocok untuk digunakan pada sistem yang membutuhkan kinerja yang optimal.

**Kata kunci** : *stateless*, *stateful*, *firewall*, SDN

**Abstract**

Computer network system is important to be maintained in order to maintain its optimal performance. Optimal performance is defined as a system that has a good Quality of Service (QoS): low round-trip time (RTT), packet loss, and jitter. One of the development of the network system is software-defined network (SDN). SDN is a network architecture that separates between the control plane with data and the network device can be set with a software. To improve the security of the SDN system, a firewall is used. Packet filtering firewall that is discussed in this research filters the packet by dropping packets based on rules that have been set. Packet filtering has two types of filters, namely: stateless and stateful. Different process occurs on each filter, and affects the system performance. Therefore, to learn how big the influence of the process that occurs on each of the two pieces of the filter, and to find out which filters are most appropriate for the system that requires optimum performance, in this research the testing simulation of stateful and stateless filter on SDN is done using QoS as its performance parameter. The results obtained from the analysis of the test results show that the stateful filter has better performance with lower RTT of 10,51% and lower packet loss of 70% making it suitable for use on systems that require optimal performance.

**Keywords :** stateless, stateful, firewall, SDN

## 1. Pendahuluan

Sistem jaringan komputer memungkinkan masing-masing penggunanya untuk saling berkomunikasi dan berbagi informasi sehingga penting untuk memastikan sistem tersebut memiliki kinerja yang optimal [17]. Kinerja sistem dapat diukur dengan menggunakan *Quality of Service* (QoS) sebagai parameternya. Komponen QoS yang digunakan adalah *round-trip time* (RTT), *jitter*, dan *packet loss*[17]. Semakin kecil RTT, *jitter*, dan *packet loss* dalam sebuah jaringan komputer semakin baik.

Salah satu pengembangan dari jaringan komputer adalah *software-defined network* (SDN). SDN adalah sebuah arsitektur jaringan dimana perangkat jaringan dapat diatur dengan dengan cara memisahkan antara *control plane (controller)* dengan *data plane plane* (*switch, router, hub)* [5] dan dapat mengontrolnya dengan hanya menggunakan *software* tanpa harus melakukan konfigurasi alat satu persatu [1]. Untuk meningkatkan keamanan sistem di dalam SDN, salah satunya digunakan *firewall.*

*Firewall* didefinisikan sebagai kumpulan dari komponen yang bertindak sebagai *gateway* yang diletakkan antara dua jaringan untuk melindungi jaringan lokal dari serangan *host* asing [3] [5]. Terdapat tiga jenis *gateway* yaitu: *packet filtering, circuit gateway*, dan *application gateway*. Jenis *gateway* yang dibahas pada penelitian ini adalah *packet filtering*. *Packet filtering* adalah gateway yang bekerja dengan cara menge-*drop packet* berdasarkan sumber maupun tujuannya, atau berdasarkan *port number*-nya sesuai dengan *rules* yang telah diset[2].

*Packet filtering* mempunyai dua jenis filter, yaitu: *stateless* dan *stateful. Stateless* filter bekerja dengan memfilter *packet* berdasarkan *value* yang terdapat pada *header packet* tersebut seperti IP *address* atau nomor *port. Stateful* filter mengecek status dan informasi yang *packet* tersebut miliki seperti: IP *address* tujuan, nomor *port* tujuan, dan *connection state* [5] [6]. Proses yang terjadi didalam kedua filter ini berpengaruh terhadap kinerja sistem

## 2. Kajian Pustaka

### 2.1 *Software-Defined Network*

SDN didefinisikan sebagai arsitektur jaringan yang memisahkan antara *control plane (controller)* dengan *data/plane* (*switch, router, hub, dll).* Salah satu protokol yang pertama dan paling sering digunakan adalah OpenFlow yang dikembangkan oleh Open Networking Foundation [5].

### 2.2 Ryu

Ryu adalah salah satu *controller* pada SDN yang didesain untuk menambah keluwesan dalam mengatur jaringan dan mampu beradaptasi bagaimana *traffic* jaringan ditangani [16]. Ryu bersifat *open source* ditulis menggunakan bahasa pemrograman Python yang mendukung semua versi protokol OpenFlow dan teruji dengan *switch* OpenFlow, serta mendukung berbagai jenis *packet libraries* dan *well-defined* API [6].

### 2.2 *Quality of Service*

*Quality of Service* atau biasa disingkat QoS adalah salah satu parameter yang digunakan untuk mengukur kinerja dari suatu servis, seperti telepon atau jaringan komputer. Dengan menggunakan QoS, pengguna dapat terbantu untuk menjadi lebih produktif dengan memastikan bahwa jaringan yang dipakai memiliki kinerja yang baik, dan pengguna dapat mencoba memperbaikinya apabila QoS yang dihasilkan buruk. [17]

Komponen-komponen QoS yang akan digunakan adalah:

1. **Round-Trip Time** (RTT), merupakan waktu yang dibutuhkan oleh sebuah *packet* untuk dikirim oleh pengirim, diterima oleh penerima, dan kembali lagi kepada pengirim. Satuan yang digunakan adalah milisekon (ms). RTT yang biasa dites menggunakan Ping, berdasarkan *Telecommunications and Internet Protocol Harmonization over Networks* (TIPHON) memiliki kategori ukuran kinerja sebagai berikut.

**Tabel 2.1 : Kategori Ukuran Kinerja RTT**

| RTT | Packet Loss | Keterangan |
|---|---|---|
| < 50ms | 0% | Sangat Baik |
| ± 90ms | 0% | Baik |
| ± 150ms | 1% | Cukup |
| ± 300ms | 3% | Kurang Baik |
| > 500ms | 20% | Jelek |

2. **Packet Loss,** persentase jumlah *packet* yang hilang saat pengiriman dari dan menuju pengirim. Kategori ukuran kinerjanya seperti pada Tabel
3. **Jitter**, merupakan variasi delay antar *packet* yang dikirim. Jitter memiliki kategori ukuran kinerja berdasarkan TIPHON sebagai berikut

**Tabel 2.2 : Ukuran Kinerja Jitter**

| Kategori | Peak Jitter |
|---|---|
| Sangat Bagus | 0ms |
| Bagus | 75ms |
| Sedang | 12ms |
| Jelek | 225ms |

## 2.3 *Firewall*

*Firewall* adalah sebuah sistem yang menjaga jaringan lokal dari akses *host-host* yang tidak bertanggung jawab yang berasal dari jaringan luar [5]. *Firewall* bekerja dengan memfilter *packet* baik yang masuk maupun keluar dari dan ke jaringan lokal sesuai dengan *rules* yang telah ditentukan oleh *system administrator* [6]. Proses filterisasi tersebut dapat dilakukan di hampir semua *layer* jaringan. Meskipun banyaknya dapat menganalisis *packet header* hanya hingga *layer transport* saja seperti Net Filter, namun terdapat pula *firewall* yang berada pada *layer application* seperti Cloudflare yang sering dijumpai ketika gagal mengakses sebuah *website* [5].

## 2.4 Mininet

Mininet adalah sebuah emulator jaringan yang dapat membuat jaringan virtual yang didalamnya terdapat: *virtual hosts*, *switchs*, *controllers*, dan *links*. Mininet *host* dapat digunakan di sistem berbasis Linux dan *switch*-nya mendukung protokol OpenFlow sehingga dapat digunakan untuk mengemulasikan SDN untuk kebutuhan riset, pengembangan, pembelajaran, pengetesan, atau *debugging* [15].

## 3. Sistem yang Dibangun

### 3.1 Komponen Simulasi

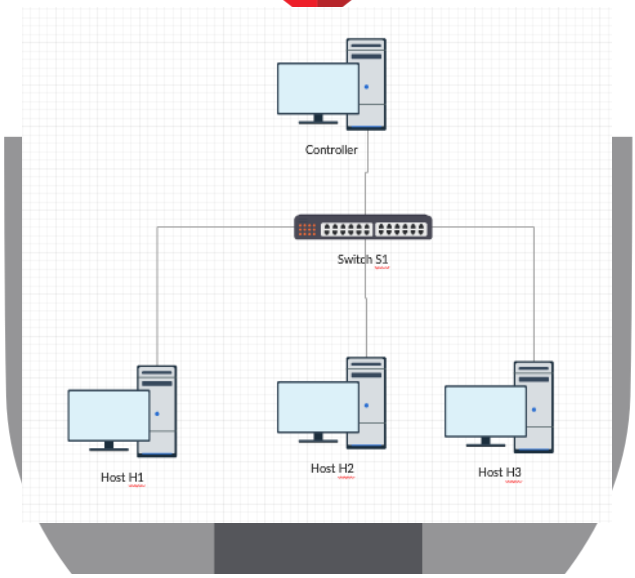
Pengujian perbandingan kinerja akan dilakukan dengan menggunakan metode simulasi menggunakan Mininet yang akan mengemulasikan SDN .

Berikut adalah komponen-komponen yang akan digunakan dalam simulasi:

1. Sistem yang disimulasikan adalah SDN
2. *Environment*-nya adalah Mininet sebagai emulator yang akan berjalan pada VirtualBox
3. *Controller* yang akan digunakan adalah Ryu *controller*
4. Atribut yang dimiliki juga berperan sebagai *Quality of* Service. Yaitu: RTT, *jitter, packet loss*
5. *Resources* yang mendukung simulasi ini adalah:
   1) OpenFlow sebagai protokol SDN
   2) Open vSwitch untuk menyediakan *switch* virtual
   3) *Rules-rules* yang akan di-*set* oleh *firewall* untuk memfilter *traffic* keluar-masuk

### 3.2 Gambaran Umum Sistem

Untuk menguji kinerja dari *stateless* dan *stateful* filter dalam jaringan virtual yang akan disimulasikan, sistem yang dibangun menggunakan topologi jaringan *linear* dengan 1 *switch* yang terhubung dengan Ryu *controller*, lalu *controller* tersebut terhubung dengan 3 *client*




***Gambar 3.1 : Skema Jaringan Simulasi***

### 3.3 Skenario Pengujian

Pengujian yang akan dilakukan akan menguji kinerja dari *stateless* dan *stateful* filter dalam SDN dengan parameter *Quality of Service* sebagai para parameter kinerjanya. Pengujian ini dilakukan untuk mendapatkan data kinerja masing-masing dan seberapa besar perbedaan kinerja diantara keduanya.

Tahap pengujian dibagi menjadi 2 berdasarkan jenis filternya: pengujian *stateless* filter dan pengujian *stateful* filter. Masing-masing jenis pengujian tersebut memiliki 5 skenario pengujian yang dilakukan berdasarkan jumlah *rules* yang diberikan, dengan jumlah *rules* yang terus bertambah pada setiap skenarionya. Hal ini bertujuan untuk mengukur perbedaan kinerja yang dihasilkan apabila *rules* yang ditetapkan oleh *firewall* terus bertambah skenarionya.

Skenario untuk pengujiannya adalah sebagai berikut:

1. a. Diberikan 600 rules lalu diuji menggunakan hping3 dengan intensitas packet sebesar 2500 packet per second

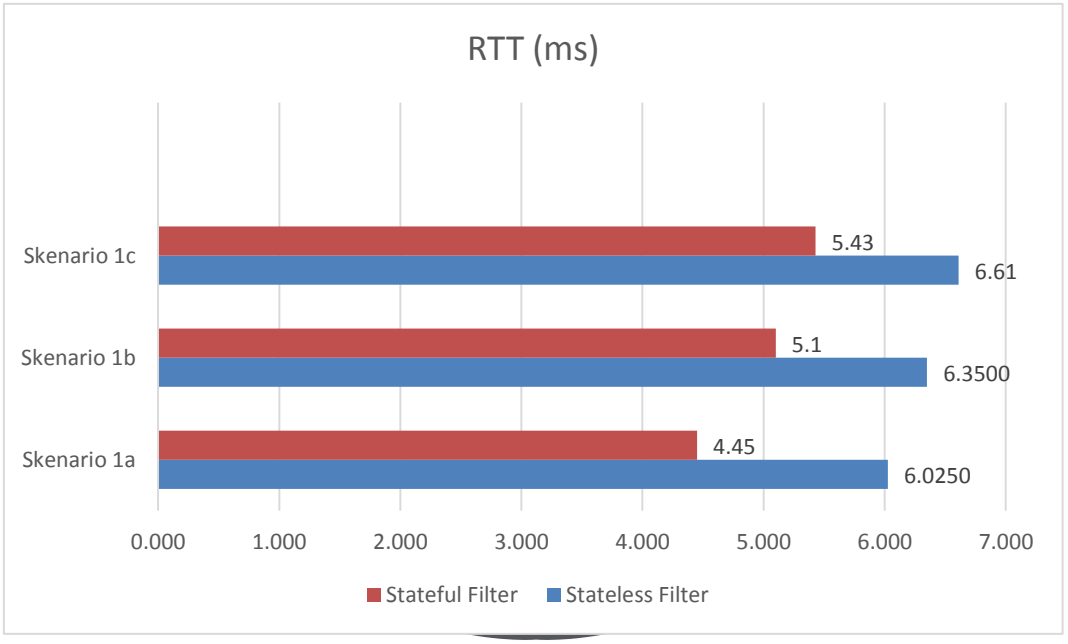
b. Diberikan 600 rules lalu diuji menggunakan hping3 dengan intensitas packet sebensar 5000 packet per second
c. Diberikan 600 rules lalu diuji menggunakan hping3 dengan intensitas packet sebensar 7500 packet per second

2. a. Diberikan 250 rules lalu diuji menggunakan hping3 dengan intensitas packet sebesar 5000 packet per second, lalu diuji lagi menggunakan iperf dengan bandwidth sebesar 1 Gigabits
b. Diberikan 500 rules lalu diuji menggunakan hping3 dengan intensitas packet sebesar 5000 packet per second, lalu diuji menggunakan iperf dengan bandwidth sebesar 1 Gigabits
c. Diberikan 750 rules lalu diuji menggunakan hping3 dengan intensitas packet sebesar 5000 packet per second, lalu diuji menggunakan iperf dengan bandwidth sebesar 1 Gigabits

Skenario tersebut akan sama-sama diterapkan untuk *stateless* dan *stateful filter* lalu hasil dari keduanya akan dibandingkan. Untuk mengukur QoS seperti: *jitter* RTT, dan *packet loss* digunakan *command hping3* dan *iperf*. Untuk mengukur RTT sekaligus menentukan besaran *packet per second* menggunakan hping3 sebanyak 4x, sedangkan untuk mengukur *jitter*, dan *packet* loss dilakukan *command* iperf yang diukur melalui *port* UDP dan dilakukan sebanyak 4x.

## 4. Pengujian Sistem

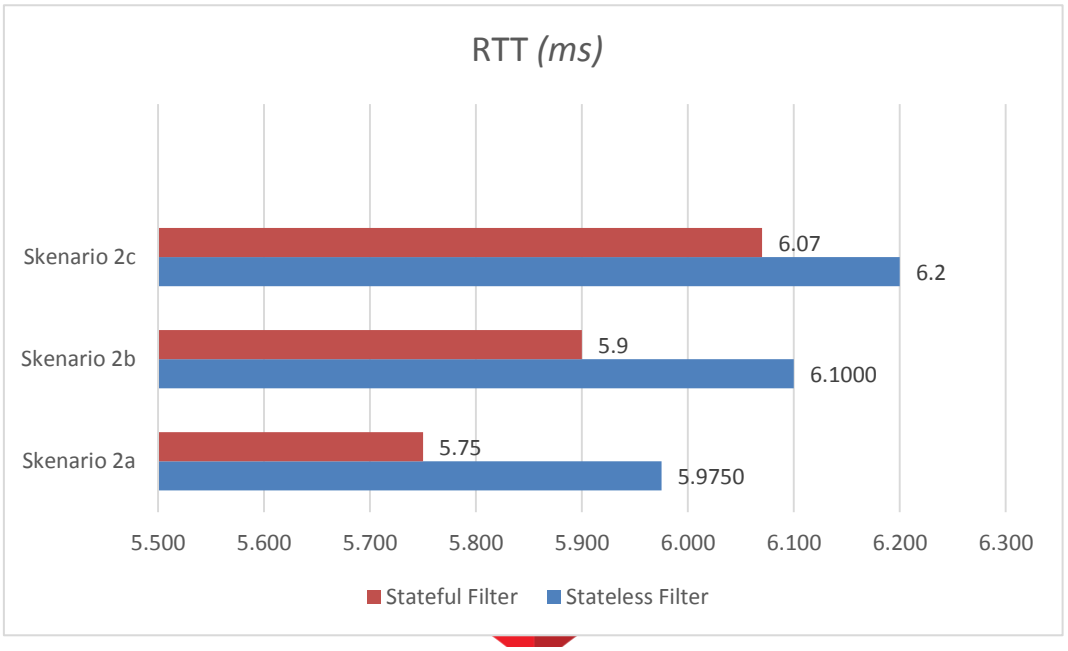
### 4.1 Skenario 1




***Gambar 4.1 : Perbandingan RTT antara stateless filter dan stateful filter pada scenario 1***

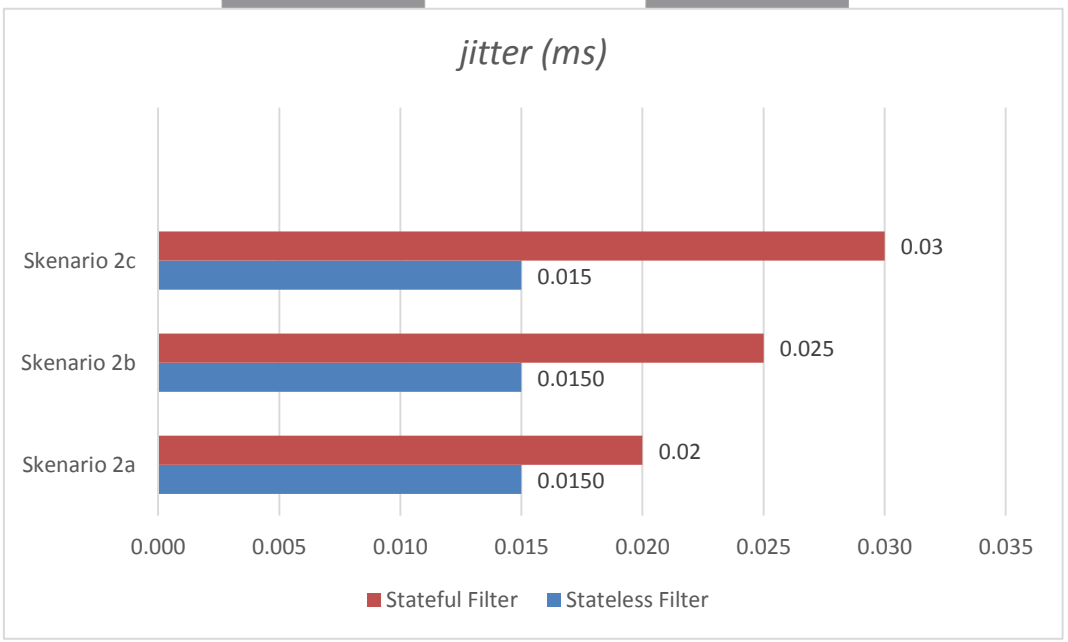
Pada *chart* diatas dapat dilihat bahwa baik pada skenario 1a, 1b, maupun 1c *stateful filter* memiliki RTT yang lebih rendah dibandingkan dengan *stateless* filter kurang lebih sebesar 21%. Ini menandakan bahwa *packet* yang dikirimkan melalui *stateful filter* lebih cepat sampai dibandingkan dengan *stateless filter*

### 4.2 Skenario 2



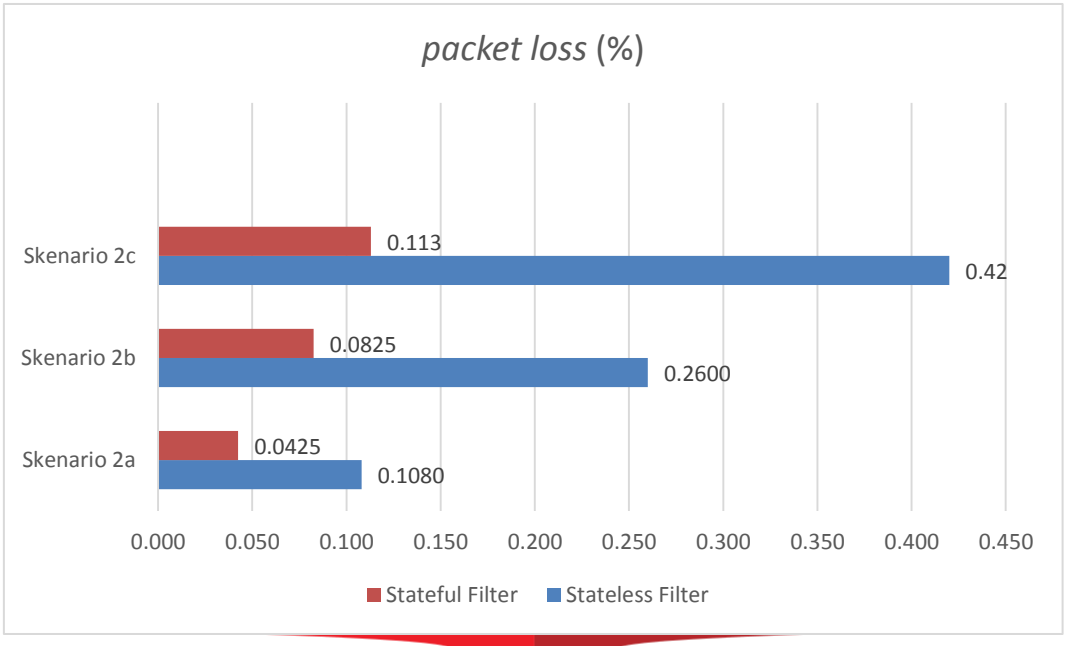

***Gambar4.2 : Perbandingan RTT antara stateless filter dan stateful filter pada scenario 2***

Sama seperti *chart* RTT yang didapat dari skenario 1, pada *chart* RTT diatas yang didapat dari skenario 2 pun menunjukkan bahwa baik pada skenario 2a, 2b, maupun 2c, *stateful filter* memiliki RTT yang rendah dibandingkan dengan *stateless filter kurang lebih sebesar 0,03%*. Apabila dirata-ratakan dengan pengujian skenario 1, perbedaan RTT kedua *filter* tersebut sebesar 10,51%




***Gambar 4.3 : Perbandingan jitter antara stateless filter dan stateful filter pada scenario 2***

Walaupun RTT *stateful filter* selalu lebih rendah, namun ia memiliki *jitter* yang lebih tinggi dibandingkan dengan *stateless filter*, namun tidak dalam jumlah yang signifikan. Apabila *stateless filter* memiliki *jitter* yang relatif stabil, *jitter* yang dimiliki *stateful filter* cenderung meninggi. Ini menandakan *stateful filter* memiliki jeda kedatangan antar *packet* yang sedikit lebih tinggi dibandingkan dengan *stateless filter*




***Gambar 4.4 : Perbandingan packet loss antara stateless filter dan stateful filter pada scenario 2***

Pada *chart* diatas dapat dilihat bahwa *stateless filter* memiliki *packet loss* yang lebih tinggi dibandingkan dengan *stateful filter* baik pada skenario 2a, 2b, maupun 2c. Ini menandakan paket yang dikirimkan melalui *stateful filter* memiliki tingkat keberhasilan sampai tujuan lebih tinggi dibandingkan dengan *stateless filter* kurang lebih sebesar 70%

## 5. Kesimpulan dan Saran

### 5.1 Kesimpulan

1. *Stateful filter* memiliki RTT yang lebih rendah kurang lebih sebsar 10,51% , dan memiliki *packet loss* yang lebih rendah dibandingkan dengan *stateless filter sebesar 70%*. Walaupun *jitter stateful filter* memang cenderung lebih besar, namun selisihnya sangat kecil dibandingkan dengan perbedaan RTT dan *packet loss* dengan milik *stateless filter*
2. Pada sistem yang lebih mengutamakan kecepatan, *stateful filter* lebih cocok digunakan sebagai *filter firewall* sistem tersebut

### 5.2 Saran

1. Pengujian yang telah dilakukan hanya berupa simulasi menggunakan Mininet saja. Alangkah lebih baiknya apabila dapat diujikan dengan jaringan SDN secara *real*
2. Skenario untuk pengujiannya dapat lebih diperbanyak lagi untuk menambah keakuratan data yang didapat

## Daftar Pustaka

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, H. Xie. A Survey on Software-Defined Networking. *IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 16, NO. 1, FIRST QUARTER.* 2015

[2] S. M. Bellovin, W. R. Cheswick. Network FIrewalls. *IEEE COMMUNICATIONS MAGAZINE, SEPTEMBER.* 1994

[3] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. *ONF White Paper*. 2012

[4] H. Guesmi, R. Tourki. Design of a QoS-Based Reconfigurable Priority Active Queue Management for IP Networks. *Electrical and Electronic Engineering.* 2012

[5] H. Francois, L. Dolberg, O. Festor, T. Engel. Network Security Through Software Defined Networking: a Survey. *IIT Real-Time Communications (RTC) Conference - Principles, Systems and Applicationf of IP Telecommunication (IPTComm).* 2014

[6] J. Shah. Implementation and Performance Analysis of Firewall on Open vSwitch. *Technische Unversitate Munchen.* 2015

[7] C. Monsanto, J. Reich, N. Foster, J. Rexfor, D. Walker. Composing Software-Defined Networks. *10th USENIX Symposium on Networked Systems Design and Implementation.* 2013

[8] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, R. Smeliansky. Advanced Study of SDN/OpenFlow Controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*. 2013

[9] T. Lin, J. M. Kang, H. Bannazadeh, A. L. Garcia. Enabling SDN Application on Software-Defined Infrastructure. *Network Operations and Management Symposium IEEE.* 2014

[10] C. Metter, S. Gebert, S. Lange, T. Zinner, P. T Gia, M. Jarschel. Investigating the Impact of Network Topology on the Processing Times of SDN Controllers. *The Seventh International Workshop on Management of the Future Internet IEEE/IFIP*.

[11] S. Sezer, S. S. Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, N. Rao. Are We Ready for SDN? Implementation Challenges for Software-Defined Networks. *IEEE Communications Magazine.* 2013

[12] Open Networking Foundation. SDN in the Campus Environment. *ONF Solution Brief.* 2013

[13] sdxcentral. What is OpenFlow? Definition and how it relates to SDN [Online]. Available: https://www.sdxcentral.com/sdn/definitions/what-is-openflow/ [Acessed 17th October 2016]

[14] Open Networking Foundation. OpenFlow Switch Specification. *ONF White Paper.* 2012

[15] Mininet. Mininet Overview [Online]. Available: http://mininet.org/overview/ [Acessed 20th October 2016]

[16] sdxcentral. What is Ryu Controller? [Online]. Available: https://www.sdxcentral.com/sd/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/ [Acessed 22nd October 2016]

[17] Imam Riadi, Wahyu Prio Wicaksono. Implementasi Quality of Service Menggunakan Metode Hierarchical Token Bucket. *JUSI Vol. 1, No. 2*. 2011

[18] T. Hayajneh, B. J. Mohd, A. Itradat, A. N. Quttoum. Performance and Information Security Evaluation with Firewalls. *Internal Journal of Security and Its Applications Vol.7, No.6.* 2013