Analisis Peformansi *QoS* pada EasyRTC menggunakan Algoritma Distribution Hash Table

Fadma Sari Y.E.G.[1], Dodi Wisaksono Sudiharto[2], Setyorini[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]emmafadma@students.telkomuniversity.ac.id, [2]dodiws@telkomuniversity.ac.id,
[3]setyorini@telkomuniversity.ac.id

Abstrak

Komunikasi audio/video melalui internet memiliki kemungkinan *latency* yang tinggi dan rendahnya *downlink goodput* yang disebabkan karena pembatasan QoS oleh *provider*. Permasalahan tersebut dapat diatasi, salah satunya dengan menggunakan jaringan *peer-to-peer,* sebab jaringan ini dapat melakukan mekanisme perutean yang efisien seperti penyambungan, pemisahan maupun *client routing*. Namun jaringan *peer-to-peer* rentan terhadap *attacker*. Sehingga pada jaringan *peer-to-peer* secara umum, dilengkapi dengan fitur pertahanan seperti dengan menggunakan Algoritma DHT *(Distribution Hash Table)*. Pada jaringan *peer-to-peer* yang lebih maju, algoritma yang digunakan umumnya adalah *Kademlia* yang masih termasuk turunan DHT. Kademlia menjadi fitur pertahanan jaringan *peer to peer* yang populer karena dianggap memiliki *overhead* yang relatif kecil dibandingkan algoritma lama, yaitu *Chord*. Di samping itu, dengan kemajuan teknologi *web*, melahirkan suatu *framework* yang dapat digunakan untuk melakukan hubungan komunikasi audio/video. Hal ini dimungkinkan lantaran adanya teknologi *web socket* yang dikembangkan pada *framework EasyRTC*. API *(Application Programming Interface)* pada *EasyRTC* dapat digunakan sebagai fitur untuk membentuk jaringan *peer-to-peer* berbasis *web* menggunakan *Node.js*. Namun kesulitan implementasi *peer-to-peer* berbasis *web* adalah pada proses *signalling* bilamana proses tersebut perlu menerapkan algoritma *Kademlia.* Pada studi ini diimplentasikan suatu komunikasi audio/video berbasis *EasyRTC* pada jaringan *peer-to-peer* dengan fitur *Kademlia* dan membandingkannya dengan komunikasi audio/video berbasis *EasyRTC* pada jaringan *peer-to-peer* tanpa fitur *Kademlia*.

Kata kunci : QoS, EasyRTC, kademlia, peer-to-peer

Abstract

Audio or video communication via internet has possibility of high latency and low downlink goodput due to restrictions of QoS by the provider. These problems can be overcome by using a peer-to-peer network, because this network can perform efficient routing mechanisms such as join, leave and client routing. Peer-to-peer networks also has some vulnerablelity for attackers, so that in peer-to-peer networks in general, equipped with defense features such as using the DHT Algorithm (Distribution Hash Table). On more advanced peer-to-peer networks, the algorithm used is generally the Academy which is still a DHT derivative. Academia has become a popular peer to peer network defense feature because it is considered to have relatively small overhead compared to the old algorithm, Chord. In addition, with the advancement of web technology, gave birth to a framework that can be used to connect audio / video communications. This is possible because of the web socket technology developed in the EasyRTC framework. The API (Application Programming Interface) on EasyRTC can be used as a feature to form a web-based peer-to-peer network using Node.js. But the difficulty of web-based peer-to-peer implementation is in the signaling process if the process needs to apply the Kademlia algorithm. In this study an audio / video communication was implemented based on EasyRTC on a peer-to-peer network with Kademlia features and compared it with audio / video communication based on EasyRTC on a peer-to-peer network without the Kademlia feature.

Keywords: QoS, EasyRTC, kademlia, peer-to-peer

1. Pendahuluan

Latar Belakang

Perkembangan evolusi komunikasi audio dan video melalui internet menjadi teknologi alternatif untuk komunikasi jarak jauh selain teknologi konvensional menggunakan PSTN *(Public Switch Transport Network)*. Salah satu teknologi yang dapat digunakan untuk komunikasi audio dan video tersebut adalah sebuah *framework* yang memanfaatkan protokol *web socket*, yaitu *EasyRTC*. Teknologi ini memungkinkan penggunaan biaya komunikasi yang fleksibilitas [1]. Keunggulan komunikasi dengan mekanisme penggunaan *web socket* ini adalah adanya tingkat *troughput* yang lebih tinggi ketimbang komunikasi suara atau video seperti yang dilakukan pada

VoIP tradisional [2]. Nilai *throughput* merupakan salah satu parameter dari QoS *(Quality of Service),* QoS merupakan standar untuk mengukur suatu layanan. QoS memiliki peranan penting karena memberi jaminan terhadap layanan minimal yang dibutuhkan untuk layak beroperasi [3].

Salah satu cara untuk meningkatkan QoS akibat kemungkinan *latency* yang tinggi karena limitasi oleh *provider* adalah dengan menggunakan jaringan *peer-to-peer* [4]. Pada jaringan *peer-to-peer* memiliki kerentanan yang cukup tinggi karena pada proses *join* dari *node* baru tidaklah seketat jaringan yang populer lainnya seperti *client-server* [5]. Di samping itu, penemuan node pada jaringan dilakukan dengan menggunakan mekanisme *key pairing*, sebagai akibatnya perutean paket dapat menembus *firewall* [6]. Hal ini menyebabkan jaringan *peer-to-peer* rentan terhadap serangan, seperti misalnya dengan menggunakan mekanisme *DDoS Attack* [7]. Sehingga umumnya jaringan *peer-to-peer* dilengkapi suatu pertahanan, seperti misalnya dengan menggunakan Algoritma DHT *(Distribution Hash Table)* [7]. Algoritma yang populer yang digunakan pada jaringan *peer-to-peer* adalah *Kademlia* [8]. Sebab, algoritma ini memiliki tingkat *overhead* yang lebih kecil daripada algoritma sebelumnya, yaitu *Chord* [8].

Meski penggunaan *web socket* dapat meningkatkan *throughput* pada komunikasi suara atau video, namun terdapat tantangan pada *EasyRTC* ini bila diimplementasikan dalam bentuk jaringan *peer-to-peer*. Hal ini disebabkan kedua *framework* tersebut tidak mem-*bundle* fitur keamanan seperti *Kademlia* dalam mekanisme komunikasinya [9]. Sehingga implementasi *Kademlia* pada *framework* tersebut merupakan sebuah tantangan.

Pada penelitian ini akan dibangun jaringan *peer-to-peer* menggunakan *framework EasyRTC* yang menerapkan algoritma *Kademlia* pada hubungan komunikasinya. Pengukuran performansi dari sistem yang diajukan akan dibandingkan dengan performansi pada *framework EasyRTC* yang tidak menerapkan algoritma *Kademlia.*

Topik dan Batasannya

Sesuai dengan latar belakang yang dijelaskan, beberapa permasalahan yang diteliti diantaranya implementasi jaringan *peer-to-peer* yang berbasis *web socket* yang menerapkan algortima *Kademlia* dalam upaya menjaga keamanan. Jaringan *peer-to-peer* diimplementasikan dengan menghubungkan beberapa *node* dalam satu jaringan dimana salah satu *node* akan berperan sebagai *server gateway*. Akhir penelitian sistem ini dilakukan perbandingan performansi *QoS* khususnya *throughput* pada jaringan yang menerapkan algoritma *Kademlia* dengan jaringan yang tidak menerapkan algoritma *Kademlia*. Penelitian ini dibatasi oleh beberapa hal, implementasi sistem dilakukan pada sistem operasi *Ubuntu* untuk *server* sedangkan untuk *client* menggunakan sistem operasi *Windows.* Implementasi *framework EasyRTC* menggunakan aplikasi *Node.Js*, aplikasi ini digunakan untuk membantu proses *signalling,* pemilihan algoritma *Kademlia* dikarenakan pada sisi *overhead*, *Kademlia* lebih baik dibandingkan algoritma *Chord*. Pengaksesan fitur *EasyRTC* melalui *web browser firefox*, karena sistem *security* untuk mengakses *local device* tidak terhalang.

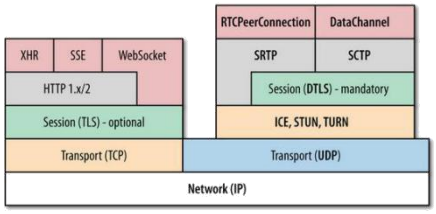
Tujuan

Pada penelitian ini diimplementasikan jaringan *peer-to-peer* dalam menggunakan layanan *EasyRTC* yang terhubung dengan protokol *web socket*. Pada jaringan *peer-to-peer* ini juga diterapkan algoritma *Kademlia* supaya dapat dibandingkan dengan jaringan *peer-to-peer* tanpa algoritma *Kademlia*. Menunjukkan perbandingan nilai *throughput* pada jaringan *peer-to-peer* dengan algoritma *Kademlia* dan jaringan tanpa *Kademlia*.

***EasyRTC* secara umum**

EasyRTC merupakan *framework open source* yang merupakan versi *simple* dan ringan dari *WebRTC* yang berfungsi sebagai *real time communication* salah satunya *video conference*. Pada *EasyRTC* menggunakan protokol *TURN server open source* pada *backend*, sehingga tidak diperlukan *setting* secara manual, hal ini yang menjadikan kenapa *EasyRTC* merupakan versi *simple* dari *WebRTC*. Selain itu protokol lain yang digunakan *EasyRTC* diantaranya *SCTP* yang membantu proses *signalling* pada *server*, dan *UDP* dalam transmisi data antar *client*. Usaha dalam mengembangkan atau menerapkan aplikasi tentu setiap *developers* memiliki hambatan baik secara umum atau pribadi, begitu pula penerapan *EasyRTC* pada penelitian ini yang juga menjumpai hambatan namun menjadi tantangan tersendiri.

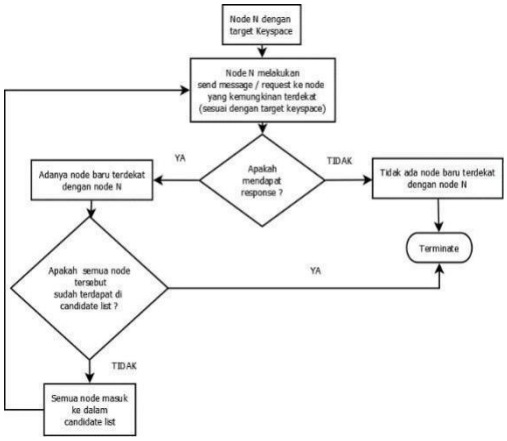
Protokol yang digunakan mengirimkan *real-time* data antar *peer* pada *transport layer* yaitu protokol UDP yang merupakan dasar fondasi dalam *real time communication*. Proses pengiriman data harus melewati beberapa halangan yang diantaranya NAT dan *firewall*, maka untuk bernegosiasi permasalahan tersebut dibutuhkan protokol ICE, STUN dan TURN untuk mempertahankan koneksi *peer-to-peer.* DLTS berperan untuk mengawasi dan me-*enkripsi* data *transfer*. Terakhir, SCTP dan SRTP digunakan untuk proses *signaling* [10]



Gambar 1. Block diagram *EasyRTC* [10]

Prinsip Kerja Algoritma *DHT*

EasyRTC dalam implementasinya memerlukan sebuah *server* yang berperan sebagai *signaling server*. *Signalling* diperlukan untuk perantara dalam masalah NAT dan *firewall* dan juga berperan untuk menghubungkan antar *node* ke dalam jaringan *peer-to-peer* mengunakan protokol *websocket*. Dengan *signalling server*, antar *node* diberikan *direct link* untuk saling berkoneksi, saat proses ini peran algoritma *DHT* dapat diterapkan pada *server* untuk menemukan *node* lain. *Distribution hash table* (*DHT)* merupakan sistem yang memungkinkan penyimpanan secara terukur dan terdistribusi, pencarian informasi secara luas (*key* dan *value*) serta pendukung pengembangan aplikasi. Konsep kerja *DHT* dengan menghubungkan *node* pada jaringan, dimana tiap *node* memiliki tabel yang terdiri dari *key* dan *value* yang berpasangan, *value* merupakan *hash file* yang berisi informasi tiap *node* yang bersangkutan [11].



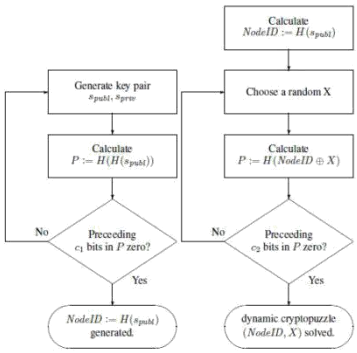
Gambar 2. Flowchart Lookup pada Algoritma *DHT* secara Umum

Step pencarian *node* terdekat (*lookup procedure*) sesuai dengan gambar 1. dilakukan dengan *node* tersebut mengirim *request* ke *node* terdekat dan apabila *node* terdekat tersebut mengirim *response*, maka diketahui adanya *node* baru terdekat namun apabila *node* pengirim tidak mendapatkan *response* diketahui tidak ada *node* lain / baru yang terdekat. Penerapan algoritma *DHT*, node menyimpan informasi mengenai lokasi tiap *peer*, rute untuk ke *peer* terdekat, *peer* mana saja yang membutuhkan informasi dan *peer* mana saja yang memiliki informasi, selain itu *DHT* juga memiliki prinsip pertahanan jaringan.Terdapat beberapa sistem yang menggunakan konsep *DHT* diantaranya *Chord*, *Kademlia* dan lainnya.

Algoritma *Kademlia*

Kademlia merupakan salah satu jenis dari Algoritma *DHT*, merupakan Algoritma yang memperbaiki C*hord* dalam hal mengurangi *overhead* [8] sebagai mana menggunakan *XOR metriks* dengan performansi tinggi karena *XOR metriks* merupakan operasi *symmetric* dan juga *unidirectional* [7]. Konsep penggunaan *XOR metriks* dimana dengan cara mencari *node* yang sama dengan *key* yang tersimpan pada *hash table* melalui *routing* . *Kademlia* memiliki panjang paket yang berbeda dibandingkan dengan panjang paket pada non DHT, karena terdapat *key* tambahan didalamnya.

Kademlia memiliki ketahanan terhadap serangan, *Sybil attack, eclipse attack* maupun *DDoS attack*. Pertahanan serangan yang dilakukan *Kademlia* berpengaruh pada performansi jaringan, untuk menghalangi serangan tersebut *Kademlia* memiliki metode yaitu *secure identifier assignment*, yaitu proses *generate identifiers* dalam cara yang aman dengan *signature. Signature* dilakukan dalam dua tahap, yaitu *hashing public key* dan *crypto puzzle* s*ignature* [7]. *Crypto puzzle* dapat menghalangi pemilihan *indentifier* secara bebas dan memastikan kerumitan untuk men*generate identifier* dalam jumlah banyak.



Gambar 3. Flowchart *Crypto Puzzle* untuk *NodeId Generation* [7]

Private key pada *node* di *decrypt* menggunakan *cryptographically secure hash function* (*P*, gambar 3), dikonfirmasi apakah nilai *complexity* dari *crypto puzzle* pada *P* tersebut *zero* dan jika memenuhi syarat tersebut maka *NodeId* berhasil di *generate*. Nilai *P* berubah setelah *nodeId* berhasil di *generate*, hasil dari *XOR operation nodeId* dengan nilai random. Konfirmasi nilai *complexity* dilakukan, jika memenuhi syarat, *crypto puzzle* yang terdiri dari *nodeId* dan nilai *random* X terbentuk.

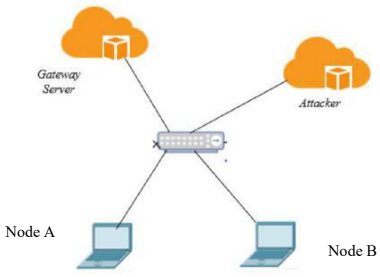
2. Studi Terkait

Pada [12] dipaparkan sebuah sistem *peer-to-peer* menggunakan algoritma DHT. Pada penelitian tersebut dikatakan sistem *peer-to-peer* yang menerapkan algortima *Kademlia* dapat meminimalkan *latency* pada *routing* sehingga dapat disimpulkan bahwa sistem dengan algoritma *Kademlia* dapat menurunkan tingkat *overhead* [13]. Pada [14] disajikan bahwa sistem *peer-to-peer* merupakan sistem yang sesuai untuk pendistribusian informasi jaringan yang dimana rentan terhadap serangan *DDoS attack* dan lainnya.

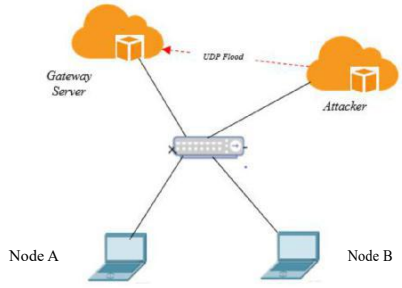
Pada [7] dipresentasikan bahwa *Kademlia* dapat menangani serangan terhadap jaringan *peer-to-peer* dengan beberapa solusi dalam upaya membuat *Kademlia* menjadi lebih tangguh. Pada [15] [16] dinyatakan bahwa *WebRTC* dan juga *EasyRTC* menggunakan *web socket* dalam komunikasi suara atau video. Hal ini dapat meningkatkan *throughput* pada komunikasi datanya.Pada [9] dipaparkan bahwa fitur seperti *Kademlia* bukan bawaan dari *framework* seperti *WebRTC* maupun *EasyRTC*. Karena kedua *framework* tersebut pada dasarnya merupakan sebuah *project open source* yang bertujuan membuat jaringan *peer-to-peer* dengan kapasitas yang tersinkronasi [16].

3. Sistem yang Dibangun

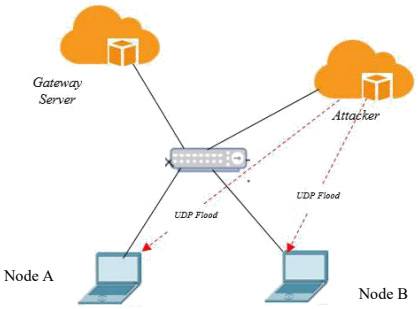
Sistem yang dibangun pada penelitian ini sesuai dengan gambar 4, terdapat *node server* yang diimplementasikan pada *cloud service* yaitu *Amazon Elastic Compute Cloud* (EC2) dengan sistem operasi *Ubuntu, node server* berperan sebagai *gateway* dari *node* yang akan melakukan panggilan.



Gambar 4. Topologi Jaringan yang dibangun pada Penelitian



Gambar 5. Topologi Jaringan pada Skenario *DDoS Attack 1*



Gambar 6. Topologi Jaringan pada Skenario *DDoS Attack 2*

Pada skenario pertama, para *client* saling terhubung terlebih dahulu dengan jaringan *peer-to-peer* tanpa fitur *Kademlia*. Sedangkan untuk skenario kedua, dilakukan dengan menerapkan fitur *Kademlia.* Skenario terakhir sebagai pembuktian algoritma *Kademlia* tidak mudah terpengaruh terhadap serangan, dilakukan *DDoS attack* dengan dua skenario di mana pada serangan pertama, *node attacker* pada *cloud server* melakukan serangan *UDP flood* pada *gateway server*. Pada serangan skenario kedua, *attack* dilakukan dari *cloud server* ke *node* yang melakukan *audio data transporting*.

4. Evaluasi

4.1 Hasil Pengujian

4.1.1 Skenario Pengujian Jaringan *Peer-to-Peer*

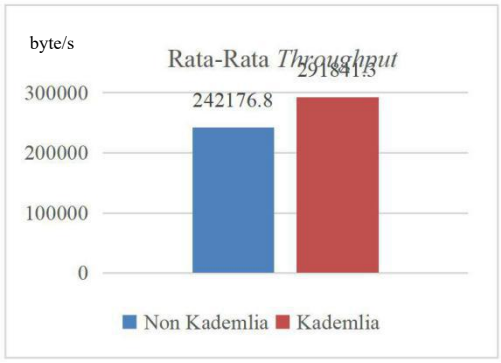
Pada skenario ini dilakukan pengujian nilai *throughput* pada jaringan *peer-to-peer* tanpa *Kademlia* dan jaringan *peer-to-peer* dengan *Kademlia*. Pengujian dilakukan sebanyak 30 kali pada masing–masing jaringan dengan kondisi yang sama dan berulang

4.1.2 Hasil Pengujian Nilai *Throughput*

Dari hasil pengujian didapat nilai subjektif rata–rata *throughput* sebagai berikut.

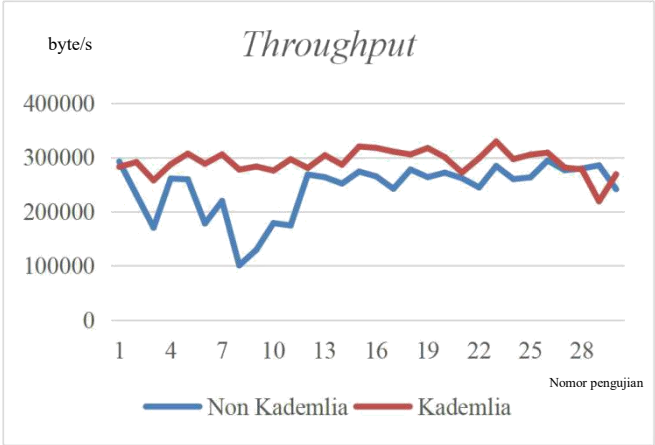
Tabel 1. Hasil Perhitungan Rata – Rata *Throughput*

Jaringan Non *Kademlia*	Jaringan *Kademlia*
242176.8 byte/s	291841.3 byte/s



Gambar 7. Trafik Perbandinngan Rata–Rata Nilai *Throughput*

Hasil pengujian dari percobaan sebanyak 30 kali pada skenario I menghasilkan rata–rata nilai *throughput* pada jaringan tanpa *Kademlia* sebesar 242176 byte/s sedangkan pada jaringan dengan *Kademlia* menghasilkan rata–rata sebesar 291841 byte/s.



Gambar 8. Grafik Nilai Throughput selama 30 kali Percobaan

Perbandingan besar *throughput* pada setiap percobaan yang dilakukan sebanyak 30 kali menampilkan bahwa *throughput* pada jaringan dengan *Kademlia* memiliki nilai yang lebih baik dibandingkan pada jaringan tanpa *Kademlia.*

4.1.3 Skenario Pengujian *DDoS Attack*

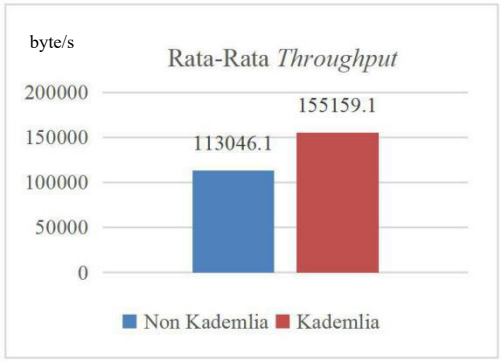
Pengujian skenario *DDoS attack* dari *cloud* ke *server gateway*

Berdasarkan hasil pengujian pada jaringan *peer-to-peer* yang menerapkan algoritma *Kademlia* dan jaringan tanpa algortima *Kademlia*, pengaruh adanya *DDoS attack* yang menyerang *server gateway*, dengan maksud mengganggu proses *signalling*, menyebabkan kedua *node* tidak dapat melakukan panggilan.

Pengujian skenario *DDoS attack* dari *cloud* ke *client*

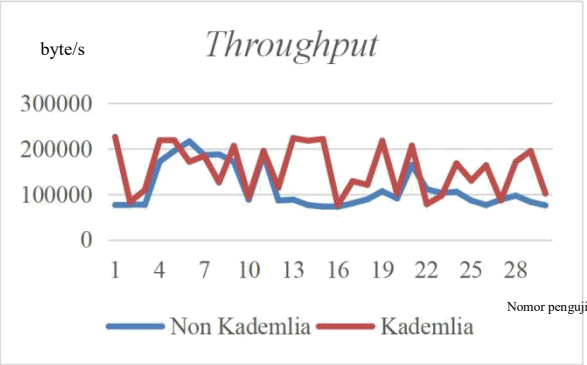
Tabel 2. Hasil Perhitungan Rata–Rata *Throughput*

Jaringan Non *Kademlia*	Jaringan *Kademlia*
113046,1 byte/s	155159,1 byte/s



Gambar 9. Trafik Perbandinngan Rata–Rata Nilai *Throughput* setelah Serangan

Hasil pengujian dari percobaan sebanyak 30 kali pada skenario II menghasilkan rata–rata nilai *throughput* pada jaringan tanpa *Kademlia* sebesar 113046,1 byte/s sedangkan pada jaringan dengan *Kademlia* menghasilkan rata–rata sebesar 155159,1 byte/s. Nilai *throughput* setelah adanya serangan mengalami penurunan jika dibandingkan dengan jaringan yang tidak ada serangan, pada jaringan tanpa *Kademlia* mengalami penurunan sebesar 31,82 % sedangkan pada jaringan dengan *Kademlia* turun sebesar 34,71 %. Serangan *DDoS attack* dimaksudkan mengirim *request* UDP sebanyak mungkin, sehingga dapat memenuhi *traffic* saat melakukan panggilan, akhirnya dapat memakan *bandwith client* yang diserang.



Gambar 10. Grafik Nilai Throughput selama 30 kali Percobaan setelah Serangan

Grafik perbandingan besar *throughput* pada setiap percobaan yang dilakukan sebanyak 30 kali menampilkan bahwa *throughput* pada jaringan dengan *Kademlia* memiliki nilai yang lebih besar setelah percobaan ke–12 dibandingkan pada jaringan tanpa *Kademlia.*

5. Kesimpulan

Berdasarkan hasil analisis seluruh percobaan pada penelitian ini, dapat ditarik kesimpulan diantaranya:

Throughput pada jaringan *peer-to-peer* dengan algoritma *Kademlia* memiliki nilai yang lebih besar, karena *Kademlia* menerapkan *iterative routing*, yaitu mengakses *multiple node* sehingga dapat memaksimalkan nilai *throughput*. Selain itu adanya perbedaan panjang paket pada algoritma *Kademlia*, karena adanya penambahan *key*, sehingga jumlah paket yang dikirimkan lebih besar dibandingkan dengan jumlah paket non *Kademlia,* di mana menjadi salah satu faktor kenapa pada rentang waktu yang sama jumlah paket yang dikirim berbeda sehingga memepengaruhi nilai *throughput* pada jaringan dengan *Kademlia*.

Setelah dilakukan serangan *DDoS* ke *server gateway*, kedua *client* tidak dapat melakukan panggilan, karena *port* UDP pada *server gateway* terpengaruh pada *DDoS attack* sehingga proses *signalling* gagal dilakukan Setelah dilakukan serangan ke *client*, nilai *throughput* pada *Kademlia* memiliki performansi lebih baik dibandingkan dengan jaringan tanpa *Kademlia* dengan peningkatan sebesar 13,72 %.

Saran yang diberikan untuk pengembangan penelitian berikutnya adalah :

a. Membandingkan performansi QoS jaringan dengan algoritma DHT lainnya.
b. Melakukan serangan selain *DDoS*, karena berdasarkan penelitian sebelumnya serangan yang menjadi masalah terbesar pada algoritma *Kademlia* adalah *Sybil attack.*
c. Membandingkan performansi dengan *framework* selain berbasis *web socket*.