Konfigurasi Jaringan Untuk Mekanisme Edge Caching Berbasis IP Menggunakan Apache Trafik Server

1st Moch. Aril Bachtiar
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
arilbachtiaralhabsy@student.telkomuniversity.ac.id

2nd Tody Ariefianto Wibowo
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
ariefianto@telkomuniversity.ac.id

3rd Sofia Naning Hertiana
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sofiananing@telkomuniversity.ac.id

***Abstrak* — Mengingat pesatnya pertumbuhan lalu lintas Internet, Cisco (2020) memprediksi bahwa akan ada lebih banyak lalu lintas IP pada tahun 2023, ketika 66% populasi dunia akan menjadi pengguna Internet. Jumlah konten di Internet terus meningkat, sehingga hampir tidak mungkin untuk melacak semua konten. Cache konten terdesentralisasi dapat mengurangi beban server backhaul sebesar 1/3-2/3 jumlah data dalam jaringan. Tentu saja, caching di edge dengan server proxy memastikan waktu respons yang lebih cepat, karena permintaan pengguna tidak perlu waktu lama untuk terhubung ke server asal. Apache Traffic Server diperlukan sebagai metode server cache untuk proxy ke cache, dan Apache Traffic Server menerima permintaan header HTTP/HTTPS dari klien. Dalam skenario parameter kinerja pengguna, ada perbedaan yang signifikan antara tidak ada cache dan cache. Misalnya, dua pengguna non-cache menghasilkan cache 0,54 Mbps dan 8,48 Mbps. Parameter delay tanpa cache menghasilkan nilai 3.82s dan cache 0.27s. Parameter RTT tanpa cache menghasilkan nilai 5.08s dan cache 0.39s. Berdasarkan hasil dan analisis konsep cache dengan menggunakan metode Apache Traffic Server Proxy terbukti lebih efektif di server asli mengurangi beban. Hasil pengukuran kinerja, latensi dan RTT menunjukkan bahwa terdapat perbedaan yang signifikan antara cache dan cache.**

Kata kunci— cache, no-cache, apache traffic server, caching

I. PENDAHULUAN

Dengan pesatnya pertumbuhan trafik data di internet, volume dan variasi konten terus bertambah, dan Cisco (2020) mencatat bahwa trafik IP meningkat sangat pesat menjelang tahun 2023, dan diprediksi akan ada lebih banyak trafik IP di tahun 2023 di mana 66 persen populasi global akan menjadi pengguna internet. Dengan meningkatnya penggunaan internet, jumlah konten *multimedia* pun berkembang pesat. Namun, jumlah halaman situs web yang diakses menurun seiring dengan waktu respons karena penundaan 1 detik menyebabkan penurunan 7% [1]. *Cisco Visual Network Index* memperkirakan bahwa trafik konten akan mencapai 82% dari penggunaan internet global pada tahun 2023 [1]. Trafik internet juga tumbuh secara signifikan dalam dua dekade terakhir karena meningkatnya jumlah pengguna internet yang menuntut layanan berkualitas tinggi.

Jumlah konten di internet terus meningkat, sehingga hampir tidak mungkin untuk menyimpan semua kontennya ke dalam *cache*. Itulah mengapa penting untuk memutuskan konten mana yang akan di-*cache*. Selain itu, sejumlah besar pengguna hanya mengakses beberapa kontetn populer yang menyebabkan peningkatan lalu lintas, sementara sebagian besar konten jarang di akses.

Caching konten terdistribusi dapat mengurangi beban pada backhaul *server* 1/3 hingga 2/3 volume data pada jaringan [2]. Dengan pengimplementasian menggunakan metode *Caching Content*, mengakses *internet* dapat dilakukan dengan lebih efisien dan efektif untuk mengurangi beban *server* agar performa *internet* lebih cepat.

II. KAJIAN TEORI

Pada penelitian ini terdapat beberapa teori yang digunakan dalam proses pengerjaan sistem yang dibangun.

A. IP Address

Alamat IP dapat disebut sebagai ID pengguna Internet, jadi satu alamat tidak boleh sama dengan yang lain. Sebelum pengembangan protokol internet, jaringan memiliki perangkat dan protokol sendiri yang digunakan untuk berkomunikasi satu sama lain. Selain itu, dibuat sebuah protokol yang biasanya digunakan untuk menstandarkan berbagai perbedaan penggunaan perangkat jaringan. (Huang & Wang, 2018). Sebagian besar lalu lintas internet didasarkan pada IPv4. Dalam paket IPv4, header dan unit data adalah bagian dari paket. Sebelum paket dikirim melalui jaringan, header minimal 20 byte dikemas dalam bagian data IPv4. Header IPv4 terdiri dari 14 bidang. Ukuran maksimum adalah 60 byte. Satu bidang adalah opsional. 4 bit pertama dari header adalah versinya. Ini menunjukkan versi IP yang digunakan. Bidang TTL 8-bit membantu mencegah paket beredar di Internet. Setiap kali sebuah paket datang dan melewati node jaringan, bidang TTL berkurang satu (Roddav et al., 2019). Pada sistem yang digunakan penulis, setiap perangkat memiliki alamat IP yang berbeda-beda tergantung kebutuhannya.

B. IP Tables

Tabel IP adalah aplikasi (berbasis Linux) yang memungkinkan administrator untuk mengonfigurasi tabel yang disediakan oleh firewall kernel Linux (diimplementasikan sebagai modul Netfilter mandiri). Berbagai modul dan program inti saat ini digunakan untuk berbagai protokol, tabel IP untuk IPV4, ip6tables untuk IPV6, tabel ARP dan tabel untuk frame Ethernet. Tabel IP memerlukan hak tinggi untuk mengoperasikan atau mengimplementasikan pengaturan yang dibuat oleh pengguna "root".

Tabel IP memiliki beberapa tabel yaitu NAT, MISS dan FILTER. Penjelasannya adalah sebagai berikut:

Table Mangle adalah tabel yang bertanggung jawab untuk memproses paket, mis. B. untuk mengubah kualitas layanan (QOS), TTL dan MARK di header TCP. Filter tabel adalah tabel yang bertanggung jawab untuk pemfilteran paket. Pada tabel ini terdapat 3 string yaitu :

Tabel NAT adalah tabel yang bertanggung jawab untuk Terjemahan Alamat Jaringan (NAT). Dengan NAT, saat perangkat di jaringan lokal mengirim permintaan ke Internet, alamat IP lokal perangkat diterjemahkan ke alamat IP publik router atau gateway sebelum mencapai tujuan akhirnya di Internet. Saat respons dikirim kembali dari internet, router atau gateway menerjemahkan alamat IP publik kembali ke alamat IP lokal yang sesuai sehingga informasi tersebut dapat dikirim kembali ke perangkat yang mengajukan permintaan.

Sistem yang penulis buat memberikan IP Tables pada klien untuk memberikan solusi efisien dan aman kepada klien untuk mengakses internet tanpa perlu mengatur pengaturan *proxy* atau *seamless*.

C. DHCP Server

Setiap komputer di jaringan membutuhkan identitas untuk berkomunikasi antar komputer. Identitas setiap komputer di jaringan ini ditentukan secara otomatis oleh server DHCP. Peran server DHCP adalah untuk menetapkan satu set alamat IP yang telah ditetapkan sebelumnya, yang kemudian didistribusikan ke komputer yang terhubung melalui server DHCP, menghilangkan kebutuhan administrator jaringan untuk mendaftarkan alamat IP setiap komputer di jaringan.. Dalam sistem yang penulis buat *DHCP server* di install didalam *apache traffic server* agar klien yang terhubung mendapatkan IP secara otomatis.

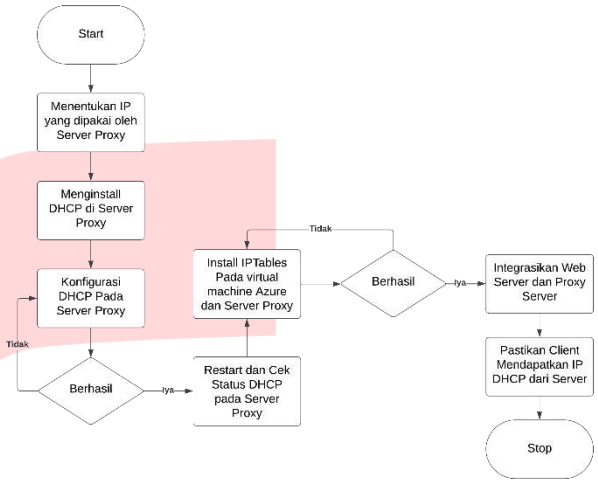
D. Edge Caching

Pada *proxy server, caching* yang dilakukan pada *edge* tentunya akan memberikan waktu respon yang lebih cepat karena tentunya permintaan dari pengguna tidak perlu memakan waktu yang banyak untuk terhubung ke *server* asli (*Website Multimedia* dan *Website Video on Demand*). Agar *proxy server* dapat melakukan *caching* pada *edge* maka diperlukan *Apache Traffic Server* sebagai metode s*erver caching proxy* dan *Apache Traffic Server* bekerja dengan cara menerima permintaan *header* HTTP/HTTPS dari klien, memeriksa *cache*-nya untuk melihat apakah sumber daya yang diminta telah di-*cache* sebelumnya, jika tidak meneruskan permintaan ke *server* tujuan, menerima respons dari *server* tujuan, melakukan *filtering* pada respons, dan akhirnya mengirimkan respons yang di*filter* tersebut kepada pengguna. Dengan demikian, *Apache Traffic Server* mempercepat waktu respon, mengurangi beban *server backend*, dan menyediakan solusi *caching* yang kuat.

III. METODE

Pada bagian ini, akan dijelaskan metode yang digunakan dalam penelitian ini, termasuk rancangan desain sistem, topologi jaringan yang digunakan, parameter pengujian ditetapkan, serta scenario pengujian yang dilakukan.

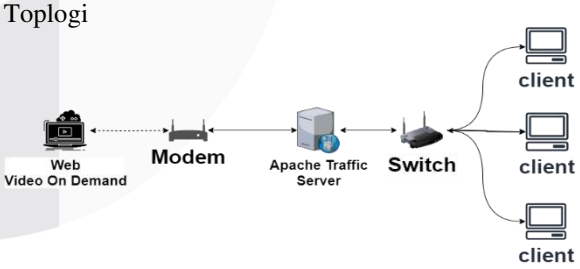
A. Desain Sistem



GAMBAR 1:
Flowchart Jaringan pada Sistem IP Edge Caching

Pada gambar 1 merupakan *Flowchart* jaringan pada sistem IP *Edge Caching* dimana administrator menentukan IP yang akan digunakan oleh *proxy server*. Setelah itu, pada *proxy server* dilakukan *install* DHCP *server* agar klien yang terhubung mendapatkan IP secara otomatis dan pada *proxy server* kita memberikan IP *Tables* yang bertujuan untuk membelokkan *port* HTTPS (443) ke *port proxy apache traffic server* HTTPS (8442) agar klien ketika mengakses internet dapat langsung terhubung dengan *proxy apache traffic server* tanpa melakukan setting secara manual.

B. Toplogi



GAMBAR 2:
Topologi IP Edge Caching dengan Web VoD

Pada gambar 2 merupakan topologi IP *Edge Caching Web* VoD dimana klien mengakses Web VoD yang penulis buat melewati *apache traffic server*. Ketika klien 1 perrtama mengakses konten maka *proxy apache traffic server* akan meminta konten langsung ke *server* VoD, dan jika klien 2 mengakses konten yang sama maka *proxy apache traffic server* akan cek didalam *cache* lokal jika ada maka akan langsung dikirm kembali ke klien tanpa perlu meminta ke *server Web* VoD.

C. Parameter Pengujian

1. *Throughput*

Throughput merupakan jumlah total kedatangan paket (data) sukses menuju tujuan selama interval waktu tertentu yang merupakan kecepatan *transfer* data yang diukur dalam satuan *MegaByte per Second* (MBps).

$$Throughput = \frac{jumlah\ data\ yang\ dikirim}{waktu\ pengiriman\ data} \quad (1)$$

Pada persamaan (1) *throughput* dapat dihitung dengan membagi jumlah data yang dikirim dengan waktu pengiriman data [3].

2. *Delay*

Delay merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal hingga dapat diterima oleh penerima.

$$Rata - rata\ delay = \frac{Total\ Delay}{Jumlah\ paket\ yang\ diterima} \quad (2)$$

Pada persamaan (2), delay dapat dihitung dengan membagi total delay yang diterima dengan jumlah paket yang diterima.

3. *Round Trip Time* (RTT)

Round trip time didefinisikan sebagai waktu yang dibutuhkan ketika *client* mengirim sebuah permintaan sampai *server* memberikan response kembali ke *client*. Untuk menghitung waktu *round trip time* bisa didapatkan dengan melakukan pengiriman paket uji seperti *ICMP (ping)*, *DNS* maupun *TCP*. Nilai waktu perjalanan pulang pergi dapat dipengaruhi oleh besarnya permintaan paket yang dikirimkan oleh klien dan perubahan kondisi lalu lintas jaringan. Saat kondisi trafik jaringan relatif stabil, nilai return time kecil, dan saat kondisi trafik jaringan lambat, nilai return time yang dihasilkan tinggi.

$$RTT = Waktu\ mengirim\ request + Waktu\ server\ merespon \quad (3)$$

Pada Persamaan (3), waktu pulang pergi dapat dihitung dari waktu respons paket (T2) dikurangi waktu paket (T1).

D. Skenario Pengujian

Untuk melakukan pengujian *caching* pada *Website Video on Demand* menggunakan protokol HTTPS menggunakan proxy apache traffic server. Berikut skenario yang digunakan untuk melakukan pengujian :

TABEL 1 :
Skenario Jumlah User

Skenario Jumlah User	
No-Cache	Cache
2 User	2 User
4 User	4 User
6 User	6 User

Pada Tabel 1 merupakan skenario jumlah user yang penulis digunakan.

TABEL 2:
Skenario Resolusi Video

Skenario Resolusi Video	
No-Cache	Cache
480p	480p
720p	720p
1080p	1080p

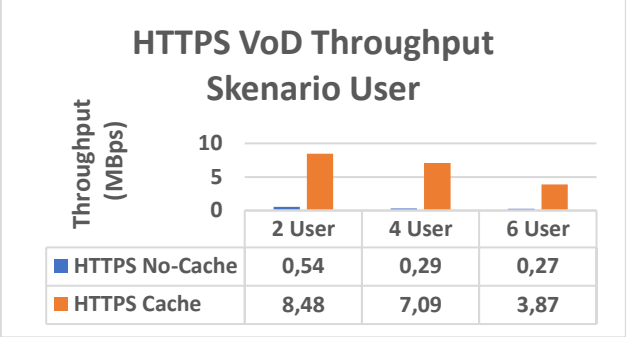
Pada Tabel 2 merupakan skenario resolusi video yang penulis digunakan.

IV. HASIL DAN PEMBAHASAN

A. Skenario Jumlah User

Dari pengujian yang dilakukan pada skneario jumlah *user* didapatkan hasil dengan parameter seperti dibawah:

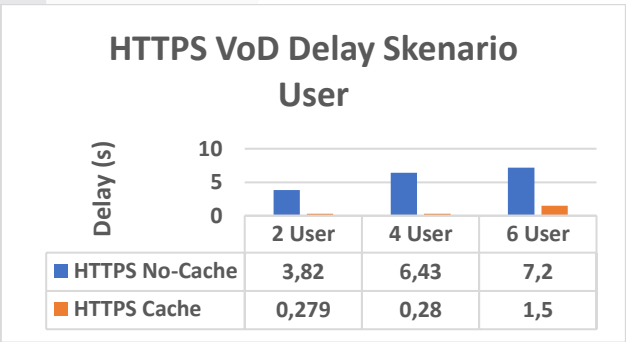
1. Throughput



GAMBAR 3:
Grafik HTTPS VoD Throughput Skenario User

Pada gambar 3 Terdapat perbedaan diatas 90% ketika menggunakan *cache* dan *no-cache*, *cache* memungkinkan pengguna atau *user* memiliki *throughput* yang besar ketika mengakses *Video Streaming* (VOD). Perbedaan yang signifikan terlihat antara penggunaan *cache* dan *no cache* dalam hal kualitas pengalaman pengguna saat menonton video. Dengan menggunakan *cache*, pengguna dapat menikmati video dengan lancar tanpa adanya jeda *buffering* atau gangguan lainnya. Hal ini dikarenakan konten video telah disimpan secara lokal dalam *cache server*, sehingga dapat diakses dengan cepat dan efisien.

2. Delay

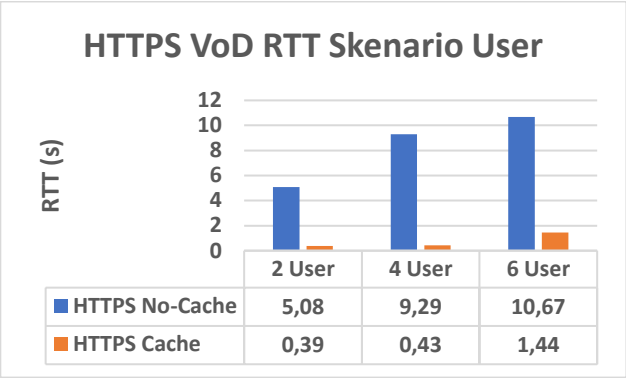


GAMBAR 4:
Grafik VoD Delay Skenario User

Pada Gambar 4 Terdapat perbedaan diatas 75% ketika menggunakan *cache* dan *no-cache* pada protokol HTTPS menunjukkan bahwa pada kondisi *Cache* untuk hasil *delay* yang didapat lebih kecil atau lebih cepat dibandingkan dengan hasil *delay* dengan kondisi *No-Cache*. *Delay* disini adalah waktu rata-rata yang dibutuhkan untuk setiap

segments dapat menjalankan atau mengaktifkan video.

3. RTT



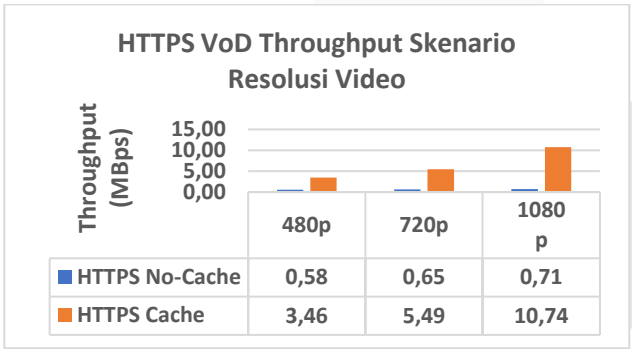
GAMBAR 5:
Grafik VoD RTT Skenario User

Pada Gambar 5 Terdapat perbedaan diatas 85% ketika menggunakan *cache* dan *no-cache* pada protokol HTTPS. Grafik menunjukkan hasil pengujian *Round Trip Time* (RTT) dengan skenario jumlah *user* menggunakan perbandingan *No-Cache* dan *Cache* pada Protokol HTTPS. Pada nilai RTT yang didapat, dapat terlihat bahwa pada kondisi *Cache* nilai RTT yang didapat lebih kecil atau lebih cepat, hasil tersebut berbeda jauh dengan kondisi *No-Cache.* Dari hasil tersebut menjelaskan bahwa fungsi *caching* bisa meningkatkan kinerja video streaming untuk setiap jumlah *user* yang mengakses.

B. Skenario Resolusi Video

Dari pengujian yang dilakukan pada skneario resolusi video didapatkan hasil dengan parameter seperti dibawah:

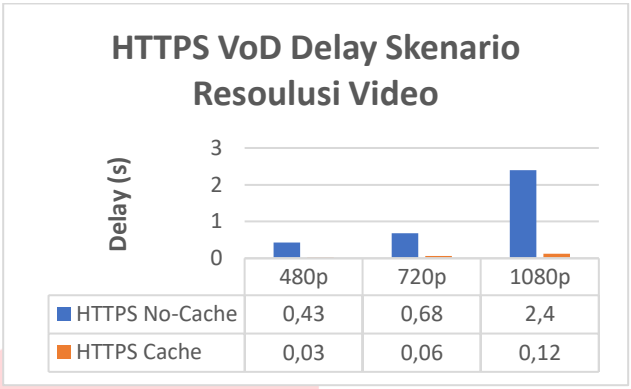
1. Throughput



GAMBAR 6:
Grafik HTTPS VoD Throughput Skenario Resolusi Video

Pada gambar 6 Terdapat perbedaan diatas 80% ketika menggunakan *cache* dan *no-cache*, *cache* memungkinkan pengguna atau *user* ketika mengakses dengan resolusi video yang berbeda memiliki *throughput* yang besar. Perbedaan yang signifikan terlihat antara penggunaan *cache* dan *no cache* dalam hal kualitas pengalaman pengguna saat menonton video. Dengan menggunakan *cache*, pengguna dapat menikmati video dengan lancar tanpa adanya jeda *buffering* atau gangguan lainnya. Hal ini dikarenakan konten video telah disimpan secara lokal dalam *cache server*, sehingga dapat diakses dengan cepat dan efisien.

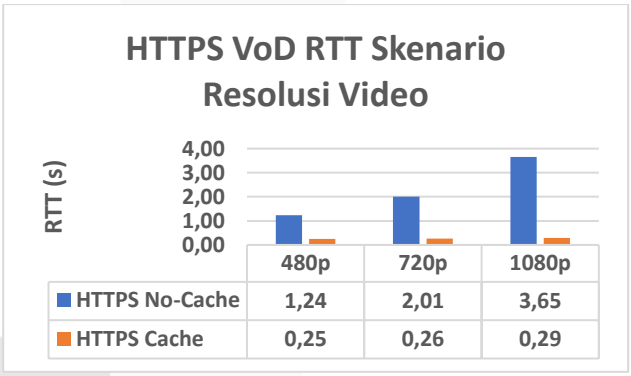
2. Delay



GAMBAR 7:
Grafik VoD Delay Skenario Resolusi Video

Pada Gambar 7 Terdapat perbedaan diatas 90% ketika menggunakan *cache* dan *no-cache* pada protokol HTTPS menunjukkan bahwa pada kondisi *Cache* untuk hasil *delay* yang didapat lebih kecil atau lebih cepat dibandingkan dengan hasil *delay* dengan kondisi *No-Cache*. *Delay* disini adalah waktu rata-rata yang dibutuhkan untuk setiap *segments* dapat menjalankan atau mengaktifkan vide.

3. RTT



GAMBAR 8:
Grafik VoD RTT Skenario Resolusi Video

Pada Gambar 8 Terdapat perbedaan diatas 80% ketika menggunakan *cache* dan *no-cache* pada protokol HTTPS. Grafik menunjukkan hasil pengujian *Round Trip Time* (RTT) dengan skenario jumlah *user* menggunakan perbandingan *No-Cache* dan *Cache* pada Protokol HTTPS. Pada nilai RTT yang didapat, dapat terlihat bahwa pada kondisi *Cache* nilai RTT yang didapat lebih kecil atau lebih cepat, hasil tersebut berbeda jauh dengan kondisi *No-Cache.* Dari hasil tersebut menjelaskan bahwa fungsi *caching* bisa meningkatkan kinerja video streaming untuk setiap jumlah *user* yang mengakses

V. KESIMPULAN

Pada sistem yang dibuat, ketika klien mengakses *website* VoD tidak perlu melakukan *setting* secara manual pada *browser* atau *seamless* dikarenakan *proxy apache traffic*

server sudah menggunakan IP Tables untuk membuat klien *transparent* dank klien juga sudah mendapatkan IP secara otomatis karena sudah diisntal DHCP *Server* pada *proxy apache traffic server.*

Berdasarkan hasil dan analisis konsep *caching* yang menggunakan metode *Proxy Apache Traffic Server* terbukti lebih efektif dalam mengurangi beban *server* asli. Hasil pengukuran pada *Throughput, Delay*, dan RTT menunjukkan bahwa *caching* memiliki perbedaan antara *no-cache* dan *cache* yang signifikan.

Pada skenario user parameter *throughput* terdapaat perbedaan yang signifikan antara *no-cache* dan *cache*. Sebagai contoh 2 *user no-cache* menghasilkan *throughput* sebesar 0,54 MBps dan *cache* sebesar 8,48 MBps dengan perbedaan sebesar 94%. Pada parameter *delay no-cache* menghasilkan nilai sebesar 3,82 s dan *cache* sebesar 0,27 s dengan perbedaan sebesar 93%. Pada parameter rtt *no-cache* menghasilkan nilai sebesar 5,08 s dan *cache* sebesar 0,39 s dengan perbedaan sebesar 92%.

Pada skenario resolusi video parameter *throughput* terdapaat perbedaan yang signifikan antara *no-cache* dan *cache*. Sebagai contoh resolusi 480p *no-cache* menghasilkan *throughput* sebesar 0,58 MBps dan *cache* sebesar 3,46 MBps dengan perbedaan sebesar 83%. Pada parameter *delay* no-*cache* menghasilkan nilai sebesar 0,43 s dan *cache* sebesar 0,03 s dengan perbedaan sebesar 93%. Pada parameter rtt *no-cache* menghasilkan nilai sebesar 1,24 s dan *cache* sebesar 0,25 s dengan perbedaan sebesar 80%.

Berdasarkan hasil dan analisis, dapat disimpulkan bahwa penggunaan *cache* memberikan hasil yang sangat baik dibandingkan dengan kondisi tanpa *cache*. Dengan menggunakan *cache*, *server* dapat menyimpan salinan *respons* sebelumnya dan mengirimkannya kembali kepada klien tanpa harus memproses ulang permintaan tersebut. Hal ini membantu mengurangi beban pada *server* utama dan meningkatkan kecepatan respon terhadap permintaan klien. Selain itu, penggunaan *cache* juga mengurangi latensi jaringan karena tidak perlu melakukan permintaan ke *server* utama untuk setiap permintaan klien.

REFERENSI

[1] K. Raja, B. Abilash, S. Anbalagan, K. Dev, and A. Ganapathisubramaniyan, “Popularity based content caching of YouTube data in cellular networks,” *Multimed Tools Appl*, vol. 81, no. 26, pp. 37165–37182, Nov. 2022, doi: 10.1007/s11042-022-13528-1.

[2] N. Krishnan, M. Karthikeyan, Tamilnadu College of Engineering, Institute of Electrical and Electronics Engineers. Madras Section. Podhigai Subsection, Institute of Electrical and Electronics Engineers. Madras Section. Signal Processing/Computational Intelligence/Computer Joint Societies Chapter, and Institute of Electrical and Electronics Engineers, *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) : 2017 December 14-16 : venue: Tamilnadu College of Engineering, Coimbatore-641 659, Tamilnadu, India*.

[3] Y. Li, H. Ma, L. Wang, S. Mao, and G. Wang, “Optimized Content Caching and User Association for Edge Computing in Densely Deployed Heterogeneous Networks,” *IEEE Trans Mob Comput*, vol. 21, no. 6, pp. 2130–2142, Jun. 2022, doi: 10.1109/TMC.2020.3033563.