Pengaruh Metode *Classful Queuing Disciplines* terhadap Efisiensi Penggunaan Bandwidth Aplikasi *Video Conference*

Jericho P. Tarigan[1], Ghania Yuntafa Putri[2], Nabila Alia Zahra[3], Usman B. Hanafi[4], Taufik Irfan[5], Griffani Megiyanto R.[6]

[1,2,3,4,5,6]*Jurusan Teknik Elektro, Politeknik Negeri Bandung, Bandung 40012*
[1]*E-mail : jericho.p.tcom417@polban.ac.id,*
[2]*E-mail : ghania.yuntafa.tkom19@polban.ac.id,*
[3]*E-mail : nabila.alia.tkom420@polban.ac.id*
[4]*E-mail : usmanb@polban.ac.id*
[5]*E-mail : taufik.irfan@polban.ac.id*
[6]*E-mail : griffani.megiyanto@polban.ac.id*

ABSTRAK

Dewasa ini, manusia memiliki banyak kegiatan yang mengharuskan berinteraksi langsung. Sekarang, terdapat teknologi *video conference* yang memungkinkan manusia bertukar informasi suara dan gambar secara *real-time* tanpa batasan jarak. Namun seringkali layanan ini tidak berjalan dengan baik akibat bandwidth yang tersedia pada jaringan terbatas. Hal ini berdampak terhadap kualitas layanan memburuk akibat parameter *delay* dan *packet loss* yang berubah menjadi tinggi. Oleh karena itu sistem yang digunakan perlu diterapkan manajemen antrian trafik yang baik. Terdapat penelitian-penelitian terdahulu yang melakukan evaluasi manajemen trafik menggunakan berbagai skenario, tetapi kebanyakan hanya menggunakan satu metode ataupun mengevaluasi layanan hanya pada simulator. Pada penelitian ini, dibandingkan performansi metode manajemen trafik *Classful Queuing Disciplines* yang terdiri dari *Hierarchical Token Bucket* (HTB), *Class Based Queuing (CBQ),* dan *Hierarchical Fair Service Curve (HFSC)* yang diimplementasikan di layanan *video conference* pada jaringan WAN. Pengumpulan data berupa paket-paket informasi dilakukan dengan *packet sniffing* terhadap komunikasi salah satu *client* dengan server. Kumpulan data diolah menggunakan Wireshark untuk mendapatkan parameter *Quality of Service* (QoS) yang terdiri dari *throughput, delay, jitter,* dan *packet loss.* Hasil pengujian menunjukkan bahwa metode CBQ memberikan performansi yang paling baik dengan nilai *throughput* sebesar 200-250 kbit/s, *delay* sebesar 10-20 ms, *jitter* sebesar 10-20 ms, dan *packet loss* sebesar 0-3%.

Kata Kunci
CBQ, HFSC, HTB, video conference, dan QoS

1. PENDAHULUAN

Saat ini dunia masih digemparkan oleh pandemi Covid-19. Pandemi memaksa masyarakat mengurangi aktivitas di luar rumah sehingga menimbulkan berbagai dampak pada sektor kesehatan, ekonomi, maupun sosial [1]. Dampak terbesar pada sektor sosial yaitu adanya pembatasan kegiatan masyarakat yang memerlukan pertemuan langsung seperti proses belajar-mengajar, seminar, ataupun bekerja. Hal ini mengakibatkan adanya peningkatan penggunaan aplikasi *video conference* untuk berkomunikasi secara virtual [2]. Salah satu perusahaan telekomunikasi 3® melaporkan bahwa pengguna aplikasi *video conference* naik hingga mencapai 693% pada tiga bulan awal penerapan *work from home* [3].

Agar kualitas layanan baik, setidaknya jaringan harus memiliki bandwidth sebesar 3 Mbps untuk sejumlah peserta *video conference*. Dengan penambahan minimal bandwidth secara linear setiap penambahan peserta [4]. Persyaratan tersebut tidak mudah dicapai sebagian masyarakat akibat adanya keterbatasan infrastruktur jaringan ataupun kecepatan internet. Bandwidth yang rendah akan berdampak terhadap kualitas layanan dimana parameter *delay* dan *packet loss* menjadi tinggi [5]. Dalam keadaan seperti ini, parameter *Quality of Service* (QoS) layanan *video conference* menjadi tidak sesuai dengan standar yang ditentukan sehingga mengakibatkan kualitas layanan menurun bahkan mengakibatkan peserta gagal bergabung ke *video conference* tersebut.

Salah satu cara meningkatkan kualitas layanan adalah dengan menerapkan manajemen antrian trafik yang baik pada server aplikasi [6]. Dengan penerapan ini, efisiensi penggunaan bandwidth dapat meningkat sehingga mampu menurunkan parameter *delay* dan *packet loss*.

1.1 Penelitian Terdahulu

Pada penelitian terdahulu telah dilakukan berbagai modifikasi manajemen trafik terhadap server layanan yang berbeda-beda. Terdapat penelitian yang menggunakan manajemen trafik *Classless Queueing Disciplines*, seperti metode FIFO, PQ, atau WFQ. Akan tetapi penelitian tersebut hanya dilakukan pada simulator, perbedaannya penelitian yang satu dilakukan untuk jaringan local (WiFi) [7] sedangkan penelitian yang satunya dilakukan pada jaringan yang lebih besar yaitu WAN [8]. Selain itu terdapat juga penelitian yang menggunakan manajemen trafik *Classful Queueing*

Disciplines. Penelitian ini dilakukan dengan menggunakan metode HTB pada aplikasi yang berbeda. Terdapat penelitian yang dilakukan dengan menjalankan aplikasi *video streaming* [9] dan juga penelitian yang menjalankan fungsi upload-download file [10]. Kedua penelitian ini melakukan metode pengamatan yang lebih baik dari penelitian sebelumnya karena menjalankan aplikasinya pada jaringan riil. Terdapat juga penelitian yang dilakukan menggunakan metode lain seperti HFSC, PRIQ, dan CBQ akan tetapi hanya menjalankan fungsi yang ringan seperti upload-download file [11].

2. LANDASAN TEORI

Quality of Service

Quality of Service (QoS) mengacu kepada kemampuan suatu jaringan menyediakan layanan yang baik bagi pelanggannya. Jaringan melakukan ini dengan menerapkan teknologi apapun untuk mengelola lalu lintas sehingga dapat mengurangi *packet loss*, *latency*, dan *jitter* pada jaringan [9]. QoS mengontrol dan mengelola sumber daya jaringan dengan menetapkan prioritas untuk tipe data tertentu pada jaringan. QoS jaringan komputer dapat bervariasi yang diakibatkan berbagai masalah, seperti halnya masalah bandwidth, *latency*, dan *jitter* yang mampu memberikan efek yang cukup signifikan bagi banyak aplikasi. Tingkat performansi dari suatu jaringan komputer dapat ditentukan dari beberapa parameter berikut ini :

a) *Throughput*
Throughput merupakan kecepatan efektif transfer data yang diukur dalam satuan bit per second (bps). Berbeda dengan bandwidth yang merupakan besar kanal yang dapat dilewati oleh paket, *throughput* adalah besar paket yang berhasil melewati kanal tersebut. *Throughput* merupakan jumlah total kedatangan paket yang sukses diterima pada sisi tujuan selama interval waktu tertentu [5].

b) *Delay*
Delay merupakan waktu yang dibutuhkan suatu jaringan untuk mengirimkan paket dari asal ke tujuan paket. Jaringan yang baik tentunya menghasilkan *delay* yang kecil. *Delay* dipengaruhi banyak hal diantaranya jarak transmisi, media fisik, waktu proses yang lama, beban paket yang hendak dikirimkan, dan lain-lain [5].

c) *Jitter*
Jitter adalah jeda waktu antar paket yang dikirim, atau variasi dari kedatangan paket yang disebabkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan dalam waktu penghimpunan ulang paket-paket yang dikirim [5].

d) *Packet Loss*
Packet loss merupakan total paket terbuang yang disebabkan oleh paket yang tidak terkirim, hilang, ataupun karena alasan lainnya. Hal ini mungkin terjadi akibat adanya *collision* dan *congestion* pada jaringan sehingga mengakibatkan aplikasi akan mentransmisi ulang paket yang tidak terkirim sebelumnya [5].

Hierarchical Token Bucket (HTB)

Hierarchical Token Bucket atau HTB merupakan salah satu metode manajemen trafik yang termasuk kedalam jenis *Classful Queueing Disciplines.* HTB berfungsi mengatur pembagian bandwidth secara hirarki dengan membagi-bagi bandwidthnya ke dalam kelas untuk mempermudah pengaturan bandwidth. Metode ini diklaim menawarkan kemudahan pemakaian dengan teknik peminjaman dan implementasi pembagian trafik yang lebih akurat. Teknik antrian HTB memberikan fasilitas pembatasan trafik pada setiap level maupun klasifikasi, akan tetapi bandwidth yang tidak terpakai oleh suatu klasifikasi bisa digunakan oleh klasifikasi yang lebih rendah [10].

Class Based Queuing (CBQ)

Class Based Queuing atau CBQ merupakan teknik klasifikasi paket data yang memungkinkan *sharing* bandwidth antar *class* dan memiliki fasilitas *user interface.* CBQ mengatur pemakaian bandwidth yang dialokasikan untuk tiap *user*, pemakaian bandwidth yang melebihi nilai set akan dipotong (*shaping*), CBQ juga dapat diatur untuk *sharing* dan meminjam bandwidth antar *class* [11]. Konsep kerja CBQ dimulai saat *classifier* menentukan paket yang datang dan menempatkan ke kelas yang tepat. Kemudian *general scheduler* menentukan bandwidth yang diperuntukkan untuk suatu kelas. Selanjutnya *estimator* akan memeriksa apakah kelas-kelas yang mendapatkan bandwidth sesuai dengan yang dialokasikan. Jika suatu kelas kekurangan maka dengan *link-sharing scheduler*, kelas yang memiliki bandwidth tidak terpakai akan dipinjamkan ke kelas yang membutuhkan tambahan bandwidth. *CBQ* membagi *user traffic* ke dalam hirarki class berdasarkan ip address, protokol dan tipe aplikasi.

Hierarchical Fair Service Curve (HFSC)

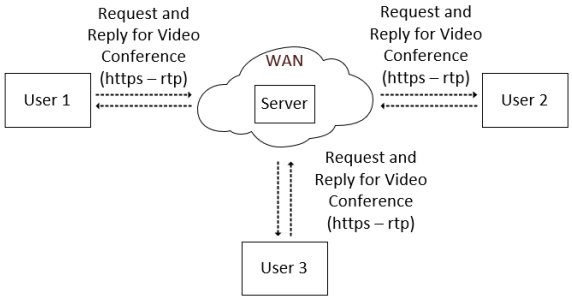
Hierarchical Fair Service Curve atau HFSC memiliki arsitektur internal yang sama dengan CBQ tetapi memiliki beberapa kelebihan. *Scheduler* HFSC memiliki kemampuan untuk mendefinisikan dua jenis paket *scheduler*, yaitu *realtime* dan *linkshare*. Kriteria *real-time* digunakan untuk menjamin kurva layanan (*service curve*) untuk semua *leaf class*, sedangkan kriteria *linkshare* digunakan untuk memenuhi kurva layanan dari kelas-kelas interior dan mendistribusikan kelebihan bandwidth secara adil. Mekanisme HFSC dikontrol oleh parameter *real-time*, *linkshare*, dan *upperlimit* [11].

3. METODE PENELITIAN

3.1 Perancangan Sistem

3.1.1 Blok Diagram Sistem

Sistem yang digunakan sebagai *environment* pada penelitian ini dapat dilihat seperti pada Gambar 1.

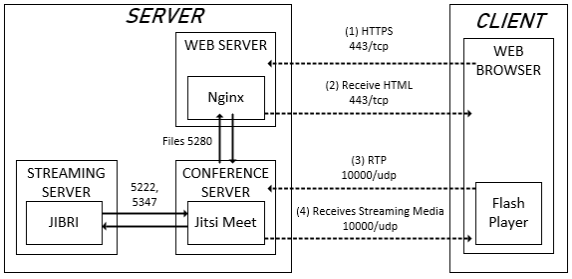


Gambar 1. Blok Diagram Sistem

Sistem *video conference* yang digunakan model "*client-server".* Server dibangun pada komputer tersendiri dengan sistem operasi Ubuntu Server 20.04 LTS. Karena digunakan pada skala kecil, komputer server cukup menggunakan *virtual machine.* Server berfungsi untuk menyimpan aplikasi *video conference* yang memiliki fungsi untuk menentukan kualitas dari video, mengatur jumlah *user* yang dapat bergabung, dan juga menghubungkan antar *user* dalam satu ruang *video conference* secara *real-time.* Komunikasi akan terjalin ketika masing-masing *user* sudah melakukan *request* dan menerima *reply* akses *video conference* ke server.

3.1.2 Sekuensial *Video Conference*

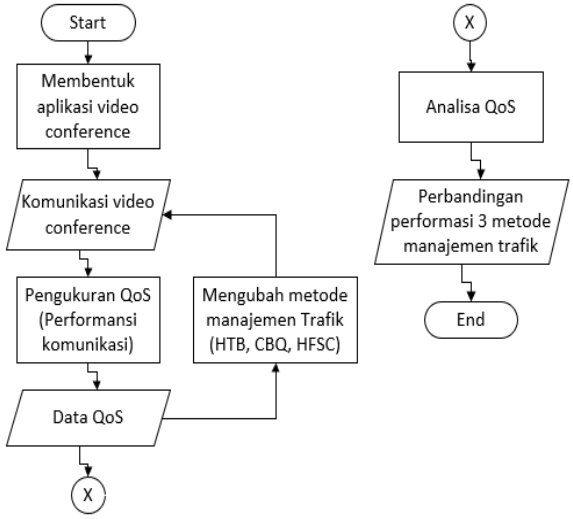
Server *video conference* terdiri dari *conference server*, *streaming server*, dan *web server*. *Conference server* berfungsi sebagai penerima *capture* (gambar atau video) dan melakukan proses encoding dari kamera yang harus tersedia pada setiap *user*. Pada penelitian ini, *conference server* yang digunakan adalah perangkat lunak *open source* Jitsi Meet. *Streaming server* berfungsi sebagai penerima video dari *conference server* yang kemudian didistribusikan secara *streaming* ke *user*. *Web server* bertanggung jawab mengatur interaksi langsung dengan *user* pada *interface* sistem yang dibangun. Proses komunikasi pertukaran paket yang terjalin di antara server dan *user* dapat dilihat dari proses sekuensial yang ada pada Gambar 2 [12].



Gambar 2. Proses Sekuensial Video Conference

Pertama, *user* melakukan permintaan HTML ke *web server* dengan menggunakan protokol HTTPS. Kedua, *web server* mengirimkan HTML sesuai permintaan *user*. Selanjutnya *user* akan mengirimkan protokol RTP agar terbentuk komunikasi server dan *user*. Setelah itu file *streaming* dikirim dari server ke *user,* maka komunikasi *video conference* sudah sudah terjalin. Proses manajemen trafik dilakukan pada *conference server* dengan mengkonfigurasi masing-masing metode yang diuji.

3.1.3 Diagram Alir



Gambar 3. Diagram Alir Pengerjaan Penelitian

Tahapan penelitian dilakukan seperti diagram alir pada Gambar 3. Tahap pertama merealisasikan server aplikasi *video conference* dengan membuat *virtual machine*. Agar komputer server dapat berjalan dengan baik, terdapat spesifikasi yang perlu dipenuhi saat membuat *virtual machine.* Spesifikasi tersebut ditunjukkan pada Tabel 1.

Tabel 1. Spesifikasi Komputer Server

Spesifikasi	Keterangan
Sistem Operasi	Ubuntu Server 20.04 LTS
Prosesor	B1ms Virtual Machine Microsoft Azure (1 vCPUs)
RAM	2 GB
Penyimpanan	30 GB

Setelah membentuk *virtual machine*, dilakukan berbagai konfigurasi seperti untuk merealisasikan server aplikasi *video conference.* Setelah server terbentuk, selanjutnya adalah membuat script konfigurasi ketiga manajemen trafik yang akan diuji. Script ini bergantian dijalankan pada server.

Pengujian dilakukan saat server melayani komunikasi *video conference* di antara 3 *user/client.* Pada saat komunikasi berlangsung, dilakukan pengumpulan data dengan melakukan pengukuran QoS dari salah satu *client*. Dari komputer *client* ini, dilakukan proses *packet sniffing* paket-paket yang terbentuk di antara client-

server. Proses ini dilakukan selama 2 menit untuk mendapatkan data yang valid. Perubahan bandwidth masing-masing *client* juga dilakukan dengan 10 variasi yang besarnya di antara 100-3000 kbit/s. Terdapat tiga puluh kumpulan paket data yang berbeda berdasarkan metode manajemen trafik dan bandwidth dari proses ini. Kumpulan data ini kemudian dikonversi menjadi parameter *throughput*, *delay*, *jitter*, dan *packet loss* dengan menggunakan rumus yang ditunjukkan pada Tabel 2.

Tabel 2. Rumus Konversi Parameter QoS

	Rumus Konversi
Throughput	$= \frac{\text{Paket data diterima}}{\text{lama waktu pengamatan}}$
Delay	$= \frac{\text{Total } Delay}{\text{Total paket yang Diterima}}$
Jitter	$= \frac{\sum \text{Variasi } Delay}{\sum \text{Paket yang Diterima} - 1}$
Variasi Delay	$= Delay - (\Delta delay)$
Packet Loss	$= \frac{(\text{paket data dikirim} - \text{paket data diterima})}{\text{paket data yang dikirim}}$

Selanjutnya data hasil konversi dianalisis dengan membandingkan satu dengan yang lainnya. Dari proses analisis ini diketahui metode manajemen trafik yang memberikan performansi paling baik.

3.2 Realisasi Sistem

3.2.1 Server Aplikasi *Video Conference*

Realisasi server diawali dengan membuat *virtual machine* pada Microsoft Azure. Dikarenakan server dibangun pada layanan *Cloud Computing*, maka *inbound port* 22 (ssh) pada server diaktifkan. Hal ini dilakukan agar dapat melakukan *remote server* untuk proses *maintenance* maupun konfigurasi server. Server diakses secara *remote* dengan memanfaatkan port ssh melalui aplikasi PuTTY. Dengan menginputkan *public* ip address server maka akan terbentuk *session* dengan server. *Public* ip address yang digunakan pada server ini yaitu 13.76.25.228 dan ditranslasi ke DNS name server jericho.southeastasia.cloudapp.azure.com. IP tersebut diatur menjadi ip *static* agar tidak perlu konfigurasi ulang ip address server setiap kali server dimatikan. Setelah terbentuk *session*, maka server sudah dapat diakses dan digunakan. Secara ringkas berikut ini langkah-langkah yang dilakukan merealisasikan server :

a) Memastikan server siap digunakan,
b) Setup fully qualified domain name (FQDN),
c) Menambahkan FQDN pada /etc/hosts,
d) Set firewall untuk Jitsi Meet,
e) Check the firewall status,
f) Instal OpenJDK JRE 8,
g) Instal Nginx web server,
h) Instal Jitsi Meet, dan
i) Jalankan Jitsi Meet.

3.2.2 Script Manajemen Antrian Trafik

```
#mengubah queuing port eth0 ke htb
$TC qdisc add dev eth0 root handle 1:0 htb default 30

#membuat parent class 1:1
$TC class add dev eth0 parent 1:0 classid 1:1 htb rate $LIMIT

#membuat anak class 1:10 dari parent class 1:1
$TC class add dev eth0 parent 1:1 classid \
1:10 htb rate $START_RATE2 ceil $CHILD_LIMIT2
#membuat anak class 1:30 dari parent class 1:1
$TC class add dev eth0 parent 1:1 classid \
1:30 htb rate $START_RATE ceil $CHILD_LIMIT1
#membuat anak class 1:50 dari parent class 1:1
$TC class add dev eth0 parent 1:1 classid \
1:50 htb rate $START_RATE ceil $CHILD_LIMIT1
```

Gambar 4. Script Metode HTB

Script metode HTB dibuat dengan skema yang terdiri dari 1 *root class* dengan id 1:0, 1 *parent class* dengan id 1:1, dan 3 buah *child class* yang masing-masing memiliki id 1:10, 1:30, dan 1:50 dengan satu diantaranya berupa *leaf class*. Terdapat 2 bandwidth yang bisa dikonfigurasi yang pertama untuk parameter *rate*, parameter ini dibuat sama pada semua *child class* meskipun jumlah *client* pada setiap *class* berbeda. Yang kedua untuk parameter *ceil*, di sini pembagian bandwidth dibuat 2 kali lebih besar bagi *child class* 1:10 karena digunakan untuk melayani *client* yang lebih banyak. Penulisan script dilakukan seperti pada Gambar 4.

```
#mengubah queuing port eth0 ke cbq
$TC qdisc add dev eth0 root handle 1: cbq allot $ALLOT avpkt $AVG bandwidth $RATE

#membuat class 1:10 & limitasi bandwidth client yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:10 cbq allot $ALLOT bandwidth $RATE_CLASS2 \
rate $RATE_CLASS2 weight $WEIG prio 3 minburst $MINBURST maxburst $MAXBURST \
avpkt $AVG_CHILD

#membuat class 1:30 & limitasi bandwidth client yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:30 cbq allot $ALLOT bandwidth $RATE_CLASS1 \
rate $RATE_CLASS1 weight $WEIG prio 5 minburst $MINBURST maxburst $MAXBURST \
avpkt $AVG_CHILD

#membuat class 1:50 & limtiasi bandwidth client yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:50 cbq allot $ALLOT bandwidth $RATE_CLASS1 \
rate $RATE_CLASS1 weight $WEIG prio 7 minburst $MINBURST maxburst $MAXBURST \
avpkt $AVG_CHILD
```

Gambar 5. Script Metode CBQ

Script metode CBQ dibuat dengan skema yang terdiri dari 1 *root class* dengan id 1:, pada *root class* digunakan parameter *allot* sebesar 1514 bytes, avpkt sebesar 1024 bytes, dan bandwidth yang bervariasi di antara 100-3000 kbit/s. Selanjutnya dibuat 3 *class* dengan 1:10, 1:30, dan 1:50. Pada ketiga *class* digunakan beberapa parameter yang sama seperti weight sebesar 6 kbit, minburst sebanyak 32 paket, maxburst sebanyak 1024 paket, dan parameter avpkt sebesar 512 bytes. Pada parameter bandwidth dan *rate* selalu diatur bernilai sama dengan nilai bervariasi di antara 100-3000 kbit/s. Penulisan script dilakukan seperti pada Gambar 5.

```
#mengubah queuing port eth0 ke hfsc
$TC qdisc add dev eth0 root handle 1: hfsc default 30

#membuat class 1:10 & limitasi bandwidth yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:10 hfsc \
sc rate $LIMIT11 ul rate $LIMIT22
#membuat class 1:30 & limitasi bandwidth yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:30 hfsc \
sc rate $LIMIT1 ul rate $LIMIT2
#membuat class 1:50 & limitasi bandwidth yang berada dalam class
$TC class add dev eth0 parent 1: classid 1:50 hfsc \
sc rate $LIMIT1 ul rate $LIMIT2
```

Gambar 6. Script Metode HFSC

Script metode HFSC dibuat dengan skema yang terdiri dari 1 *root class* dengan id 1:0, dan 3 buah *child class* dengan id 1:10, 1:30, dan 1:50. Script ini tidak menggunakan parameter rt dan ls karena kedua nilai tersebut dapat direpresentasikan menggunakan parameter sc. Nilai parameter yang digunakan pada *class* 1:10 selalu diatur 2 kali lebih besar karena digunakan untuk melayani *client* yang lebih banyak. Penulisan script dilakukan seperti pada Gambar 6.

4. HASIL DAN PEMBAHASAN

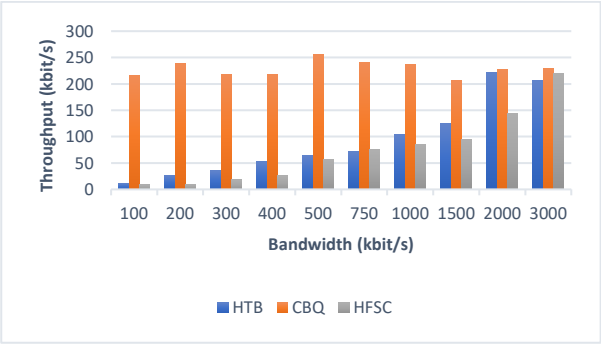
4.1 Throughput

Berdasarkan pengujian, didapatkan data *throughput* seperti yang ditunjukkan pada Tabel 3 dan grafik Gambar 7.

Tabel 3. Data Parameter *Throughput*

Bandwidth (kbit/s)	Throughput (kbit/s)		
	HTB	CBQ	HFSC
100	12	216	9.322
200	27	238	9.7
300	36	218	18
400	52	217	26
500	65	255	57
750	71	241	75
1000	104	237	86
1500	125	206	95
2000	221	228	144
3000	206	230	219

Dari Gambar 7, diketahui bahwa penggunaan metode CBQ menghasilkan *throughput* yang paling stabil. *Throughput* yang stabil di antara 200-250 kbit/s cukup untuk melakukan komunikasi *video conference* yang lancar dengan fitur kamera, *microphone*, dan juga chat. Pada penggunaan metode HTB dan HFSC ada kalanya komunikasi menjadi terganggu bahkan mengakibatkan *client* sampai terputus dari komunikasi. Hal ini bisa terjadi karena nilai *throughput* yang relatif lebih kecil.



Gambar 7. Grafik *Throughput*

Pada penggunaan bandwidth yang lebih besar, metode HTB mampu lebih cepat untuk mencapai nilai *throughput* yang lebih tinggi dibanding HFSC. Untuk mencapai *throughput* 200 kbit/s, metode HTB akan mencapainya pada saat *client* diberi bandwidth sebesar 2000 kbit/s sedangkan pada metode HFSC nilai ini akan tercapai ketika *client* diberi bandwidth sebesar 3000 kbit/s.

Berdasarkan analisis parameter *throughput*, dapat diketahui bahwa metode CBQ menghasilkan nilai *throughput* paling tinggi pada setiap variasi bandwidth sehingga mampu melakukan komunikasi *video conference* dengan baik meskipun dengan bandwidth kecil. Hal ini bisa terjadi karena pada metode CBQ memiliki *shaping algoritm* yang baik.

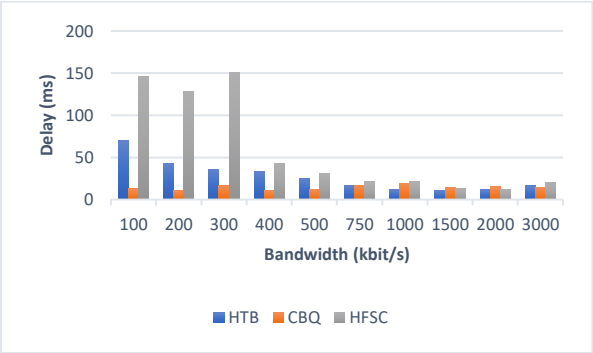
4.2 Delay

Berdasarkan pengujian, didapatkan data *delay* seperti yang ditunjukkan pada Tabel 4 dan grafik Gambar 8.

Tabel 4. Data Parameter *Delay*

Bandwidth (kbit/s)	Delay (ms)		
	HTB	CBQ	HFSC
100	69.9	12.98	145.71
200	42.84	10.89	128.77
300	36.27	16.33	150.45
400	33.37	10.66	43.06
500	25.22	12.16	31.42
750	16.59	16.9	21.08
1000	11.62	19.25	20.98
1500	11.29	14.86	13.06
2000	12.05	15.1	12.33
3000	16.18	14.92	19.77

Dari Gambar 8 diketahui dengan menggunakan metode CBQ, server hanya menghasilkan *delay* sebesar 10 – 20 ms. Hal ini berbanding lurus dengan besar parameter *throughput* sebelumnya yang bernilai stabil sehingga tidak menghasilkan parameter *delay* yang fluktuatif.



Gambar 8. Grafik *Delay*

Dari gambar tersebut dapat juga dilihat bahwa parameter *delay* cenderung bernilai sama ketika bandwidth *client* sebesar 750 kbit/s hingga bandwidth 3000 kbit/s. Akan tetapi perbedaan yang cukup signifikan terjadi pada saat bandwidth *client* di antara 100 kbit/s hingga 500 kbit/s. Pada rentang bandwidth 100 kbit/s hingga 500 kbit/s parameter *delay* metode HFSC lebih buruk dibandingkan dengan *delay* metode HTB. Hal ini bisa terjadi akibat kondisi komunikasi

yang lebih buruk saat penggunaan metode HFSC. Nilai *delay* metode HFSC meningkat pada saat *client* yang digunakan untuk pengujian mengalami putus koneksi dengan *room video conference.*

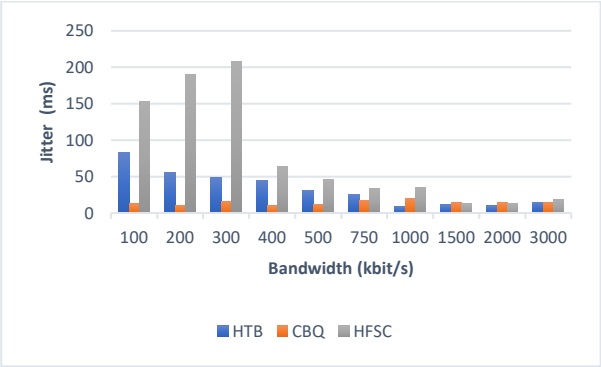
4.3 Jitter

Berdasarkan pengujian, didapatkan data *jitter* seperti yang ditunjukkan pada Tabel 5 dan grafik Gambar 9.

Tabel 5. Data Parameter *Jitter*

Bandwidth (kbit/s)	Jitter (ms) HTB	CBQ	HFSC
100	82.76	12.47	153.07
200	55.39	10.46	189.85
300	49.15	16.35	207.87
400	45.19	9.78	63.46
500	30.91	12.34	46.37
750	25.68	17.54	33.3
1000	9.02	20.52	35.57
1500	11.48	13.8	12.75
2000	10.04	14	12.9
3000	14.84	14.54	18.84

Pada Gambar 9 terlihat bahwa bentuk grafik *jitter* tidak jauh berbeda dengan grafik *delay* yang ditunjukkan oleh Gambar 8. Meskipun memiliki bentuk grafik yang sama akan tetapi nilai kedua grafik tersebut berbeda. Pada penggunaan metode HTB dan HFSC, besarnya nilai *jitter* selalu lebih besar jika dibandingkan dengan nilai *delay*-nya. Dari data pada Tabel 4 dan Tabel 5 diketahui bahwa nilai *jitter* menjadi cenderung sama dengan nilai *delay* pada saat besar nilai *delay* sekitar 10-20 ms. *Delay* sebesar itu didapatkan ketika komunikasi *video conference* berjalan dengan lancar. Pada perhitungan parameter *jitter* saat penggunaan metode CBQ didapatkan nilai yang stabil berada pada 10-20 ms. Hal ini didapatkan karena selama percobaan komunikasi selalu berjalan dengan lancar.



Gambar 9. Grafik *Jitter*

Berdasarkan pengujian, dapat diketahui bahwa bandwidth sebesar 100-750 kbit/s dengan metode HTB dan bandwidth sebesar 100-1000 kbit/s dengan metode HFSC tidak stabil dalam melakukan proses *forwarding* data sehingga dapat memperbesar kemungkinan terjadinya *packet loss*.

4.4 Packet Loss

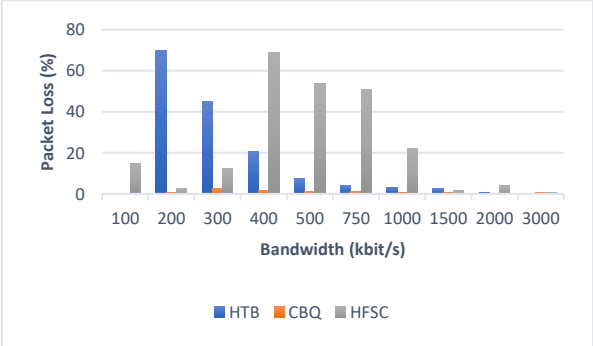
Berdasarkan pengujian, didapatkan data *packet loss* seperti yang ditunjukkan pada Tabel 6 dan grafik Gambar 10.

Berdasarkan pengujian, pola data *packet loss* metode HTB dan HFSC sama meskipun tidak selalu bernilai rendah. Sedangkan saat server menggunakan metode CBQ, parameter *packet loss* tetap menunjukkan nilai yang hampir sama yaitu sekitar 0-3% pada setiap variasi bandwidth.

Tabel 6. Data Parameter *Packet Loss*

Bandwidth (kbit/s)	Packet Loss (%) HTB	CBQ	HFSC
100	0	0.34	15.04
200	69.65	0.77	2.63
300	45.11	2.53	12.45
400	20.5	1.69	68.66
500	7.32	1.37	53.91
750	4.03	1.12	50.93
1000	3.2	0.55	22.32
1500	2.8	0.64	1.62
2000	0.64	0.26	4.06
3000	0.35	0.67	0.7

Dari Gambar 10, terlihat bahwa saat *client* memiliki bandwidth yang rendah dan server menggunakan metode HTB dan HFSC, parameter *packet loss* yang didapatkan juga relatif rendah. Padahal pada bandwidth sekecil itu komunikasi *video conference* tidak dapat dilakukan dengan lancar. Hal ini diakibatkan minimnya paket UDP yang sampai ke *client* karena *throughput* yang sangat kecil. Sehingga paket-paket tersebut tidak dapat analisis atau dihitung *packet loss*-nya.



Gambar 10. Grafik *Packet Loss*

Pada metode HTB, *Packet loss* kategori sedang-buruk terjadi ketika *client* memiliki bandwidth 200-500 kbit/s. Sedangkan pada metode HFSC, *packet loss* kategori sedang-buruk ketika bandwidth 400-1000 kbit/s. Hal ini sesuai dengan apa yang terjadi saat pengujian, dimana saat metode dan bandwidth tersebut digunakan hal yang sering terjadi yaitu kamera salah satu atau dua *client* menjadi *blank* atau mati selama beberapa detik sebelum akhirnya muncul kembali.

5. KESIMPULAN

Server aplikasi *video conference* Jitsi Meet yang dapat digunakan untuk melayani komunikasi *video conference* pada jaringan WAN telah berhasil direalisasikan pada penelitian ini. Dari tiga metode manajemen antrian trafik *Classful Queuing Disciplines* yang digunakan server, performansi QoS paling baik didapatkan ketika server menggunakan metode CBQ. Berdasarkan data yang didapat, penggunaan metode ini menghasilkan komunikasi *video conference* yang baik/lancar meskipun menggunakan bandwidth yang minim diantara 100 hingga 3000 kbit/s. Performansi QoS yang dihasilkan ketika server menggunakan metode ini stabil dengan parameter *throughput* lebih besar dari 200 kbit/s, *delay* lebih kecil dari 20 ms, *jitter* lebih kecil dari 20 ms, dan *packet loss* lebih kecil dari 3%. Maka untuk mendapatkan penggunaan bandwidth yang paling efisien, server aplikasi *video conference* sebaiknya menggunakan metode manajemen antrian trafik CBQ.

Pengembangan yang dapat dilakukan terhadap penelitian ini diantaranya, melakukan pembatasan bandwidth dari sisi *client*, melakukan *packet sniffing* dari sisi *client*, menggunakan pengujian dengan jumlah *client client* yang lebih banyak, serta mencoba menggunakan manajemen antrian trafik lainnya yang termasuk ke dalam metode *Classless Queuing Disciplines.*

DAFTAR PUSTAKA

[1] A. Syahputra, "Pengaruh Covid-19 Terhadap Aktivitas Sosial dan Ekonomi Masyarakat Lhokseumawe," *ETNOREFLIKA*, vol. 9, no. 3, pp. 226-237, 2020.

[2] M. U. Fajrin, "Faktor Yang Memengaruhi Minat Perilaku Penggunaan Teknologi," dalam *Industrial Research Workshop and National Seminar*, Bandung, Indonesia, 2020.

[3] A. Kurniawan, "Aplikasi Video Conference dan Data Komunikasi di Indonesia," 2020. [Online]. Available: https://aqi.co.id/news/aplikasi-video-conference-dan-data-komunikasi-di-indonesia. [Diakses 7 Maret 2021].

[4] Budarsa, "Berapa Kecepatan Internet Ideal Untuk VidCon?," 2020. [Online]. Available: https://blibagus.com/berapa-kecepatan-internet-ideal-untuk-vidcon/. [Diakses 8 Februari 2021].

[5] O. A. Melala, "Analisis Kualitas Layanan Video Call Menggunakan Aplikasi Skype pada Jaringan Long Term Evolution (LTE)," *KITEKTRO: Jurnal Online Teknik Elektro* , vol. 5, no. 1, pp. 38-44, 2020.

[6] A. H. Fauzan, "Analisis Manajemen Bandwidth Dengan Metode Hierarchical Fair Service Curve (HFSC) Pada Layanan Video Streaming," Universitas Telkom, Bandung, 2011.

[7] E. F. Cahyadi dan P. U. E. Sakti, "Analisis Karakteristik Teori Antrian Pada Aplikasi Wireless Fidelity Menggunakan Opnet Modeler 14.5," *Jurnal Buana Informatika*, vol. 6, no. 4, pp. 321-328, 2016.

[8] D. Suprijatmono, "Simulasi Perancangan Model Skema Antrian Pada Jaringan WAN Untuk Peningkatan Layanan Kinerja Multimedia," *BINA TEKNIKA*, vol. 15, no. 2, pp. 137-146, 2019.

[9] D. H. Janius, *Analisis QoS Video Streaming Pada Jaringan Wireless Menggunakan Metode HTB (Hierarchical Token Bucket)*, Pekanbaru: UIN SULTAN SYARIF KASIM RIAU, 2013.

[10] S. Budin dan I. Riadi, "Traffic Shaping Menggunakan Metode HTB (Hierarchical Token Bucket) pada Jaringan Nirkabel," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 1, no. 3, pp. 144-152, 2019.

[11] R. H. Syahputra, "Perbandingan Manajemen Bandwidth Dengan Metode HFSC, PRIQ, dan VBQ Pada Pfsense," *Jurnal Manajemen Informatika*, vol. 11, no. 1, pp. 41-49, 2020.

[12] Z. Maruf, *Implementasi Aplikasi Video Conference Pada E-Pesantren Berbasis OpenMeetings*, Depok: Departemen Teknik Elektro Universitas Indonesia, 2011.