BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Ananth M.D. dan Rinki Sharma [4] pada tahun 2017. Menganalisis biaya dan performasi penggunaan NFV berbasis *cloud*. *Software cloud* yang digunakan adalah *openstack* dengan memvirtualkan layanan *web server*. Pada penelitian ini, hasil yang didapatkan dari performansi CPU dengan 4 pengguna yang mengakses layanan sebesar 0.3% pada *cloud system* dan 0.31% pada *physical system*, dengan 40 pengguna yang mengakses layanan sebesar 0.3% pada *cloud system* dan 0.31% pada *physical system*, dengan 400 pengguna yang mengakses layanan sebesar 0.5% pada *cloud system* dan 1.04% pada *physical system*, untuk 2000 pengguna yang mengakses layanan sebesar 1.11% pada *cloud system* dan 1.43% pada *physical system*, dan untuk 4000 yang mengakses layanan sebesar 0.65% pada *cloud system* dan 0.97% pada *physical system*. Berdasarkan parameter tersebut *cloud* layak untuk diimplementasikan

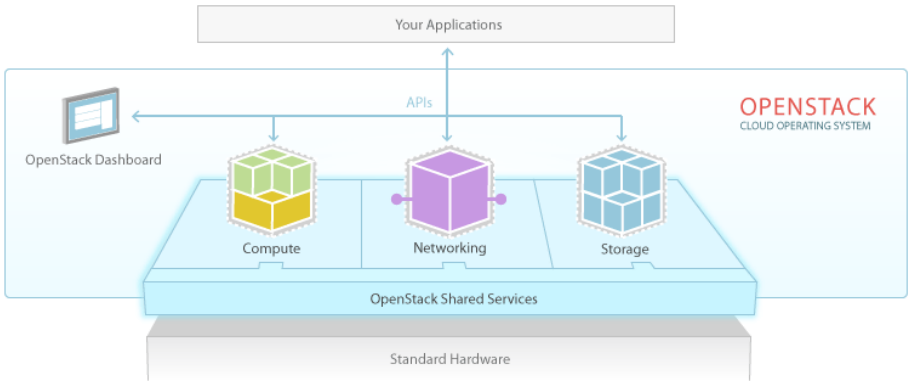
Penelitian Hafis Nurdin, Sumarna dan Felix Wuryu Handono [5] pada tahun 2018 mengimplementasikan *cluster openmeetings* guna meningkatkan efektivitas belajar mengajar jarak jauh. Pada penelitian ini dilakukan 3 pengujian, pengujian pertama yaitu pengujian Belajar Mengajar, pengujian dilakukan untuk memperhatikan kualitas suara dan gambar yang dihasilkan dan hasilnya menunjikan kualitas suara dan gambar yang layak untuk dijadikan kelas jarak jauh. Pengujian kedua adalah pengujian *server* tunggal, pengujian dilakukan dengan cara melakukan konferensi oleh seorang pengajar kepada dua siswa, hasil pengujian menunjukan bahwa semua siswa dan pengajar berada pada room dan *server* yang sama, hal ini menandakan bahwa *server Openmeetings* sudah bekerja dengan baik dan benar. Pengujian ketiga adalah pengujian *Cluster*, pengujian dilakukan dengan cara melakukan konferensi oleh tiga pengajar dengan masing-masing mempunyai seorang siswa dan materi pengajaran yang berbeda. Hasilnya menunjukan bahwa setiap pengajar dengan siswanya dan materi pengajaran yang sama akan menempati room yang

sama walaupun *server* nya berbeda, hal ini meunjukan pembagian beban ke setiap *server* tidak hanya bertumpu pada satu buah *server* saja, akan tetapi terbagi kesemua *server* yang ada, sehingga kinerja dari tiap *server* menjadi ringan.

Penelitian Muhammad Fauzan, Andrew Fiade dan Fenty Eka M. A. [6] pada tahun 2017 merancang insfrastruktur *private cloud* dengan *Openstack*. Tujuan dari perancangan tersebut untuk melakukan efisiensi serta skalabilitas yang tinggi sehingga dapat memudahkan akses, monitoring dan manajemen secara terpusat. Berdasarkan hasil penelitian tersebut, sistem infrastruktur yang dirancang dengan menggunakan *Openstack* menggunakan satu node *controller* dan satu node *compute* mampu melakukan efisiensi sumber daya infrastruktur IT sebesar 1 CPU dan 638,296 bytes RAM dan mampu melakukan *resource sharing* infrastruktur IT.

2.2 OPENSTACK

Openstack merupakan *software open source* untuk membangun *cloud computing*. *Openstack* dapat berjalan di sistem operasi linux *RedHat* / *CentOS*, *Ubuntu*, *Opensuse* [8]. Tampilan arsitektur *openstack* dapat dilihat pada gambar 2.1.

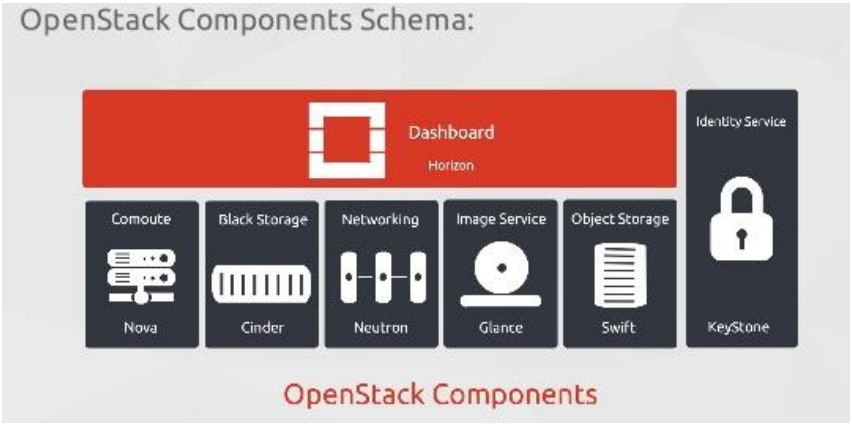


Gambar 2. 1 *Openstack Architecture*[7]

Openstack mengelola sumber daya yang terdapat pada infrastruktur fisik di dalam datacenter [8]. Sumber daya *Openstack* diantaranya adalah sebagai berikut:

A. *Controller* berfungsi sebagai pengelola layanan *identity, image service,* manajemen dari *compute* dan *network*
B. *Compute* beroperasi pada *virtual machine* atau *instances.*
C. S*torage* beroperasi untuk menjalankan komponen *objeck* dan *block storage.*
D. *Network* yang berfungsi untuk menjalankan *network plug-in*, menyediakan *virtual network, routing, Network Address Translation (NAT),* dan *Dinamic Host Configuration Protocol (DHCP).* Sumber daya *Network* juga menangani *external* konektivitas (*internet)* untuk *virtual machine* atau *instance.*

Openstack terdiri dari beberapa proses API. Semua layanan memiliki setidaknya satu proses API (*Application Programming Interface)* yang melakukan proses mereka dan meneruskan ke bagian layanan lain [9]. *Openstack* memiliki desain yang bersifat standar. Masing-masing komponen di Openstack mengatur sumber daya yang berbeda-beda yang dapat di-virtualisasikan untuk *client*. Pemisahan masing-masing sumber daya tersebut yang dapat di virtualisasikan kedalam komponen-komponen terpisah. Tampilan komponen *openstack* dapat dilihat pada gambar 2.2.



Gambar 2. 2 Komponen *Openstack*[10]

Berikut keterangan dari komponen-komponen *Openstack:*

A. *Horizon* (*Dashboard*), Menyediakan antarmuka web untuk semua komponen didalam *Openstack.*
B. *Swift (Object Storage),* Sebagai penyimpanan *storage* yang dapat menerima data sangat banyak hingga *unlimited*.
C. *Nova (Compute),* Komponen utama dalam pembentukan *Infrastructure As A Service* (IaaS) yang mengatur proses alokasi CPU untuk setiap *Virtual Machine (*VM*).*
D. *Glance (Image Service),* Untuk me-*manage image* dan layanan *virtual disk image* yang dapat di*attach* kedalam VM melaui *Nova.*
E. *Cinder (Block Storage),* Sebagai penyedia layanan penyimpanan *block* dalam bentuk *volume* yang digunakan sebagai *virtual disk* untuk VM.
F. *Neutron (Networking),* Menyediakan layanan *Network Connectivity as a Service* (NaaS) yang mengatur *network* seperti *subnet, routing, load balancer* dan *floating ip.*
G. *Heat (Orchestration),* Komponen untuk men-*deploy* aplikasi *Cloud* yang menjadi penengah antara manusia dan mesin dan digunakan unuk mengatur semua komponen infrastruktur dan aplikasi dalam *cloud openstack*.
H. *Keystone (identity),* menyediakan otentikasi dan otorisasi di seluruh layanan yang ada didalam *openstack.*

2.3 CLOUD COMPUTING

Cloud computing merupakan sebuah model *Client-server*, di mana *resources* seperti *server*, *storage*, *network* dan *software* dapat dipandang sebagai layanan yang dapat diakses oleh pengguna dimana saja dan kapan saja [11].

Terdapat tiga *service layer* atau *delivery model* yang disediakan *cloud computing* yaitu :

1. Infrastructure As A Service (IaaS)

Pengguna dapat menggunakan fundamental computing *resouces* seperti *processing power, storage, networking component.* Penguna diizinkan untuk menginsal sistem operasi, *storage*, membangun aplikasi sendiri, membuat *firewal* dan *load balancer*. Contoh IaaS yaitu Amazon *Elastic Compute Cloud.*

2. Platform As A Service (PaaS)

Pengguna dapat mengembangkan aplikasi menggunakan *application framework* atau *application engine* yang disediakan oleh *provider.* Penggina dapat secara leluasa mengontrol aplikasi, namun tidak dapat mengontrol sistem operasi, *hardware* atau *network.* Contoh PaaS yaitu Force.com dan Mikcrosoft Azure Investment.

3. Software As A Service (SaaS)

Pengguna dapat menggunakan aplikasi namun tidak dapat membuat aplikasi, tidak dapat mengontrol sistem operasi, *hardware*, dan *network.* Aplikasi dapat diakses melalui *web browser* atau *web based interface.* Contoh SaaS adalah GoogleDoc dan SalesForce.

2.4 VIRTUALISASI

Teknologi Virtualisasi adalah landasan bagi tumbuh dan berkembangnya *Opensack.* Pada dasarnya *openstack* tidak lain adalah suatu kerangka kerja bagaimana mengatur dan menggunakan sumber daya teknologi virtualisasi yang berkembang terus agar penggunaan virtualisasi tetap terkontrol dan mudah dilakukan.

Virtualisasi merupakan membuat versi virtual dari suatu sumber daya sehingga pada suatu sumber daya fisik dapat dijalankan, dengan syarat unjuk kerja masing-masing sumber daya virtual tidak berbeda signifikan dengan sumber daya fisiknya, hingga saat ini sumber daya yang tekah dapat

divirtualisasikan adalah perangkat keras (*hardware*), media penyimpanan (*storage*), *operating system* dan layanan jaringan (*networking*) [12].

2.5 VIDEO CONFEFRENCE

Komunikasi dalam *video conference* menggunakan audio serta video secara *real time* yang bisa dilakukan dalam tempat yang berbeda, bisa berupa dua lokasi (*point-to-point)* atau mengikutsertakan beberapa lokasi yang berbeda (*multi-point)* [13]. Adapun jenis-jenis video conference berdasarkan hubungan diantara pemakainya adalah sebagai berikut :

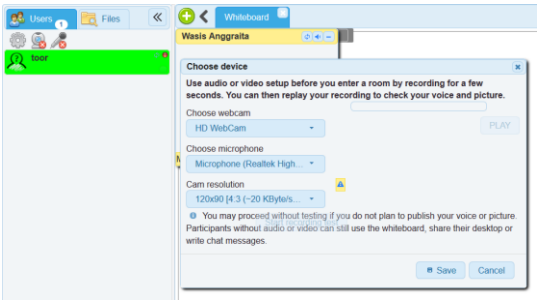
1. *Real Time Colaborative multiparty conferencing.*
2. *Active Participation User* adalah keikutsertaan pemakai yang bersifat aktif.
3. *Passive Participation User* adalah keikutsertaan pemakai yang bersifat pasif.

2.6 OPENMEETINGS

Openmeetings merupakan *software opensource* berbasis browser yang memungkinkan untuk mengatur langsung sebuah konferensi di web [14]. *Software* ini mendukung penggunaan mikrofon atau *webcam*, *sharing* dokumen, *sharing screen* atau *record* dari konferensi tersebut. Fitur dan fasilitas *Openmeetings* adalah:

1. *Conferencing audio and video.*

Tampilan untuk mengatur kualitas *webcam* dan *audio* yang digunakan dapat dilihat pada gambar 2.3.

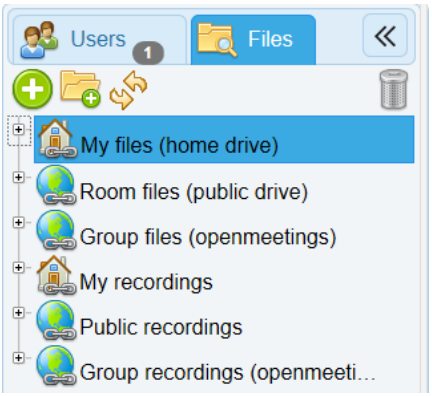


Gambar 2. 3 Conferencing audio and video

Dalam aplikasi *Openmeetings* dapat menggunakan fungsi yang bisa dipilih selama sesi *conference* yang diantaranya adalah audio dan video, hanya audio, hanya video atau hanya gambar.

2. *File Explorer.*

Tampilan fasilitas *file exploler* dalam *software openmeetings* dapat dilihat pada gambar 2.4.

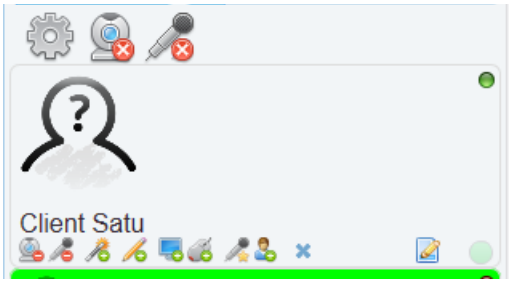


Gambar 2. 4 File Explorer

Fasilitas *file explorer* disediakan untuk setiap *conference room*, disertai dengan kemampuan *drag and drop* untuk mengatur file serta membuat struktur folder.

3. Moderating system.

Tampilan fasilitas *moderating system* dalam *software openmeetings* dapat dilihat pada gambar 2.5.



Gambar 2. 5 *Moderating System*

Selama *conference* berlangsung, moderator dapat mengatur hak akses semua pengguna secara individual.

4. *Multi Whiteboard and Chating.*

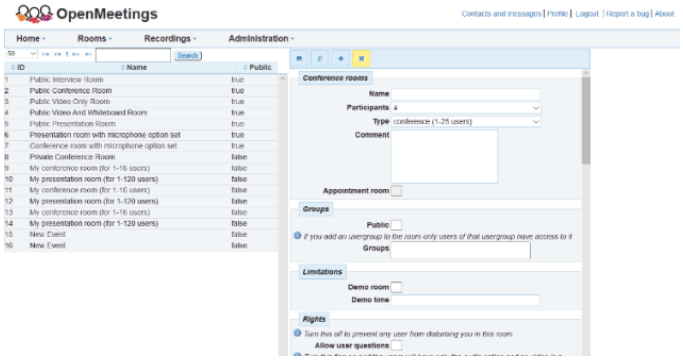
Tampilan fasilitas *Multi Wthiteboard and Chatting* dalam *software openmeetings* dapat dilihat pada gambar 2.6.

Gambar 2. 6 Multi Whiteboard and Chatting

Multi Whiteboard, fitur atau fasilitas ini memungkinkan pengguna dapat menambahkan *Whiteboard* baru, setiap *Whiteboard* dapat disimpan dalam folder tertentu dengan menggunakan fitur *Save Whiteboard*, *Whiteboard* dapat digunakan sebagai media untuk menggambar, menulis, *drag and drop*, mengubak ukuran gambar, menyisipkan gambar, symbol – symbol atau *clipart* serta fasilitas *chatting* yang dapat digunakan pengguna untuk mengirim pesan kepada pengguna lainnya.

5. Manajemen Pengguna dan Ruangan.

Tampilan fasilitas manajemen pengguna dan ruangan dalam *software openmeetings* dapat dilihat pada gambar 2.7.

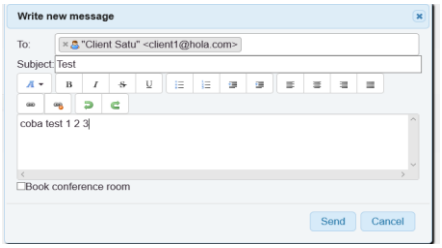


Gambar 2. 7 Manajemen Pengguna dan Ruangan

Sebagian user atau organisasi dapat diorganisir dalam satu buah objek *Openmeetings*. Setiap user secara default memiliki dua *personal room* yang dapat diakses secara eksklusif.

6. *Private Message Center.*

Tampilan fasilitas *private message center* dalam *software openmeetings* dapat dilihat pada gambar 2.8.

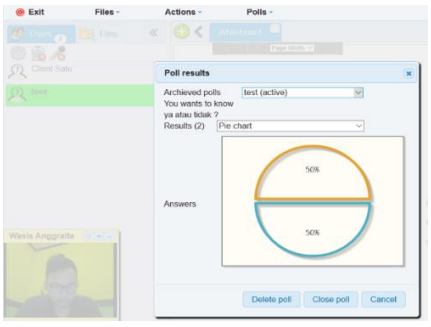


Gambar 2. 8 *Private Message Center*

Fitur ini memungkinkan untuk mengirimkan pesan ke pengguna dan mengelolanya dalam folder. *Plan Meetings with Integrated calendar*. Perencanaan *conference* dapat dibuat dan dapat mengundang pesertanya dari dalam *Openmeetings* atau dari luar. Peserta yang diundang akan menerima sebuah email dengan detail tentang *conference*.

7. *Polls and Votes.*

Tampilan fasilitas *Polls and votes* dalam *software openmeetings* dapat dilihat pada gambar 2.9.



Gambar 2. 9 *Polls and Votes*

Jejak pendapat dapat dibuat dengan jawaban ya atau tidak dengan 1 sampai 10 pertanyaan. Pengguna dapat menjawabnya dan dapat melihat hasil *voting*.

8. Backup.

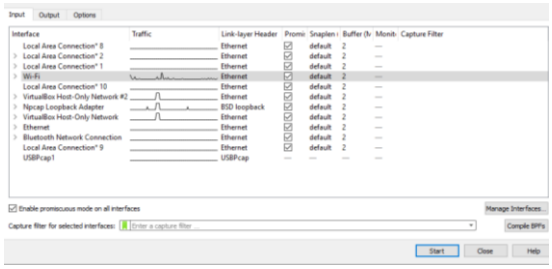
Tampilan fasilitas *backup* dalam *software openmeetings* dapat dilihat pada gambar 2.10.

Gambar 2. 10 *Backup*

Semua file dapat di*backup* dalam *Openmeetings*. Hasil *backup* dapat kembali dimasukkan kedalam *room* yang berbeda dalam *Openmeetings*.

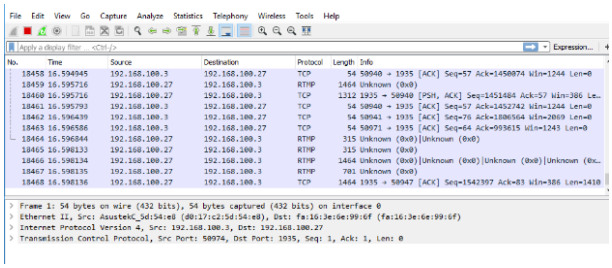
2.7 WIRESHARK

Wireshark dapat dikatakan sebagai *tool* analisa paket data jaringan yang paling sering digunakan. Sebagian fitur pada *Wireshark* adalah tersedia untuk *platform* UNIX, Linux, Windows dan Mac [15]. *Wireshark* Dapat melakukan *Capture* paket data jaringan secara *real time*, dapat menampilkan informasi protokol secara lengkap, pada Gambar 2.11 merupakan tampilan pemilihan *interface* yang akan di *capture* dan *start capture*.



Gambar 2. 11 Pemilihan *Interface* dan *start capture Paket Data*

Wireshark dapat mengkaji paket data secara *real time* dan akan mengawasi seluruh paket data yang keluar masuk melalui *interface* yang sudah diterapkan. Pada Gambar 2.12 merupakan tampilan *wireshark* yang sedang melakukan pengawasan paket data:



Gambar 2. 12 Pengawasan Paket Data

Paket data dapat disimpan menjadi file dan nantinya dapat dibuka kembali dengan cara *klik file* dan pilih *save as*, untuk Pem*filter*an paket data jaringan dan Pencarian paket data dengan kriteria spesifik bisa dicari pada kolom *Apply a display fillter*.

2.8 QUALITY OF SERVICE (QOS)

Quality of Service (Qos) Merupakan metode pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakter dan sifat dari suatu *service*. QoS digunakan untuk mengukur beberapa atribut kinerja yang telah dispesifikasikan dengan suatu *service* [16].

2.8.1 THROUGHPUT

Throughtput adalah nilai rata – rata pengiriman yang sukses melalui saluran telekomunikasi dalam suatu pengiriman. *Throughtput* diukur dalam satuan *bit per second* (bps atau bit/s). Rumus menghitung *throughtput* ditunjukkan pada persamaan 2.1 [17]

$$Throughtput\ (bps) = \frac{Packet\ Data\ yang\ diterima\ (bit)}{Waktu\ Pengiriman\ Packet\ (second)} \quad (2.1)$$

2.8.2 *DELAY*

Delay adalah permasalahan umum yang terjadi pada jaringan telekomunikasi. *Delay* merupakan waktu yang diperlukan sebuah paket untuk melakukan perjalanan dari pengiriman ke penerima. Rumus untuk menghitung *delay* ditunjukkan pada persamaan 2.2 [17]

$$Delay\ Rata - Rata = \frac{Total\ Delay}{Total\ Paket\ yang\ diterima} \quad (2.2)$$

Klasifikasi standarisasi *delay* berdasarkan TIPHON ditunjukkan pada Tabel 2.1.

Tabel 2. 1 Klasifikasi standarisasi *delay* [18]

No	Kategori	Besar (ms)
1	Sangat Baik	< 150
2	Baik	151 – 350
3	Cukup Baik	350 – 449
4	Tidak Direkomendasikan	> 450

Nilai *delay* dikategorikan Sangat Baik jika mendapatkan kurang dari 150 ms, bernilai Baik jika mendapatkan di antara 150 ms sampai dengan 300 ms, bernilai Cukup Baik jika mendapatkan di antara 300 ms sampai dengan 450 ms dan jika mendapatkan lebih dari 450 ms maka tidak direkomendasikan untuk digunakan.

2.8.3 *PACKET LOSS*

Packet loss merupakan parameter yang menggambarkan kondisi yang menunjukkan jumlah paket yang hilang. Rumus untuk menghitung *packet loss* ditunjukkan pada persamaan 2.3 [17]

$$Packet\ Loss = \frac{(Paket\ data\ dikirim - paket\ data\ diterima)\ x\ 100\ \%}{paket\ data\ yang\ dikirim} \quad (2.3)$$

Klasifikasi standarisasi *packet loss* berdasarkan TIPHON ditunjukkan pada Tabel 2.2.

***Tabel 2. 2 Klasifikasi standarisasi* packet loss** [18]

No	Kategori	Besar (%)
1	Sangat Baik	0-2
2	Baik	3-14
3	Cukup Baik	15-24
4	Tidak Direkomendasikan	>25

Nilai *packet loss* dikategorikan Sangat Baik jika mendapatkan nilai diantara 0-2%, bernilai Baik jika mendapatkan di antara 3% sampai dengan 14%, bernilai Cukup Baik jika mendapatkan di antara 15% sampai dengan 24% dan jika mendapatkan lebih dari 25% maka tidak direkomendasikan untuk digunakan.

2.8.4 JITTER

Jitter merupakan variasi *delay* yang terjadi akibat Panjang antrian dalam suatu pengelolahan data dan *reassemble* paket data di akhir pengiriman akibat kegagalan sebelumnya. Rumus menghitung *Jitter* ditunjukkan pada persamaan 2.4 [17]

$$Jitter = \frac{Total\ variasi\ delay}{Total\ Paket\ yang\ diterima - 1} \quad (2.4)$$

Klasifikasi standarisasi *Jitter* berdasarkan TIPHON ditunjukkan pada Tabel 2.3.

Tabel 2. 3 Klasifikasi standarisasi *Jitter* [18]

No	Kategori	Besar (ms)
1	Sangat Baik	0 – 74
2	Baik	75 – 124
3	Cukup Baik	124 – 224
4	Tidak Direkomendasikan	> 225

Nilai *Jitter* dikategorikan Sangat Baik jika mendapatkan 0 ms, bernilai Baik jika mendapatkan di antara 1 ms sampai dengan 75 ms, bernilai Cukup Baik jika mendapatkan di antara 76 ms sampai dengan 125 ms dan jika mendapatkan lebih dari 126 ms maka tidak direkomendasikan untuk digunakan.