Pengembangan *Video Broadcasting Server* Untuk *Live Streaming* Menggunakan *Nginx* dan RTMP Dengan Studi Kasus *Teleconference*

Tommy Andreas Susanto[1], Henry Novianus Palit[2], Agustinus Noertjahyana[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
Email: tommy.andreas@outlook.com[1], hnpalit@petra.ac.id[2], agust@petra.ac.id[3]

ABSTRAK

Era globalisasi telah mendorong komunikasi lintas negara yang lebih intensif. Komunikasi yang disediakan oleh teknologi tersebut bahkan bisa dilakukan secara *realtime.* Globalisasi juga mendorong terjalinnya kerja sama antar universitas atau institusi pendidikan lintas negara. Untuk menjalin kerja sama tentunya dibutuhkan komunikasi, dan *media* komunikasi yang terbaik untuk hal ini adalah *teleconference.* Pengembangan *broadcast server* ini dibuat menggunakan Nginx dan modul RTMP, *server* ini bisa digunakan untuk melakukan *live streaming* secara *private*, atau umum, melalui *broadcast server* yang dimiliki sendiri dan pada jaringan sendiri, sehingga tidak memerlukan *platform-platform* yang sudah ada. Pengguna dapat menggunakan server untuk melakukan *live streaming, teleconference*, dll. *Server* juga dapat melakukan *archiving* dari sesi *live streaming* yang sudah dilakukan dan hasil dari *archive* dapat diputar secara *video on demand, server* juga bisa menjadi *relay* untuk menampilkan *live streaming* ke platform-platform yang sudah ada atau ke server *Nginx* lain. Dan setelah dilakukan pengujian server sudah dapat menjalankan semua fungsi-fungsi yang sudah dikerjakan dan sudah bisa menjadi *broadcast server* yang dimiliki sendiri untuk kebutuhan *live streaming*, terutama untuk *teleconference*.

Kata Kunci: *Nginx, Live Streaming, Broadcast, Teleconference, RTMP*

ABSTRACT

Globalization Era has pushed the communication around the world to be more intensive. The communication that has been provided by technology can even be used real time. Globalization also pushed the cooperation of universities or institution between countries, Communication is needed to establish cooperation, and teleconference is the best method of communication to establish this cooperation. Nginx and RTMP is used to develop this broadcast server, the server can be used for live streaming both publicly and privately, via self-owned hardware and network, so the usage of existing platform can be negated. Users are able to use the server for teleconference and other real-time communication activities. Server can also archive the live streaming session, and the file from archive can be played on demand. The server can also become a relay to transmit a stream that its currently receiving to another platform or other server. After a series of test, server is able to use the features than had been worked on, and can be used as self-hosted server for live streaming, especially teleconference.

Keywords: *Nginx, Live Streaming, Broadcast, Teleconference, RTMP*

1. PENDAHULUAN

Era globalisasi telah mendorong komunikasi lintas negara yang lebih intensif. Komunikasi yang disediakan oleh teknologi tersebut bahkan bisa dilakukan secara *realtime.* Globalisasi juga mendorong terjalinnya kerja sama antar universitas atau institusi pendidikan lintas negara. Untuk menjalin kerja sama tentunya dibutuhkan komunikasi, dan *media* komunikasi yang terbaik untuk hal ini adalah *teleconference. Live streaming* biasanya hanya bisa dilakukan secara *one-way* yang berarti hanya ada satu pihak yang terlibat dalam proses *broadcast* tersebut.

Untuk melakukan *multi-way* diperlukan konfigurasi dan infrastruktur tambahan, *live stream* secara *multi-way* ini bisa digunakan untuk melakukan *teleconference, online lecture,* dan lain-lain. *Teleconference* adalah istilah yang digunakan untuk menghubungkan orang yang berlokasi di tempat yang berbeda dengan *media* elektronik.

Komunikasi antar pihak dalam teleconference bisa menggunakan *audio, video* atau langsung kedua-duanya. Dengan menggunakan *teleconference* penyaluran informasi akan menjadi lebih *efficient.* Data yang digunakan pada saat *teleconference* di kirim secara *realtime* atau di *stream* secara *live*.

2. DASAR TEORI

2.1 Teleconference

Teleconference adalah istilah yang digunakan untuk menghubungkan orang yang berlokasi di tempat yang berbeda dengan *media* elektronik. Komunikasi antar pihak dalam *teleconference* bisa menggunakan *audio, video* atau langsung kedua-duanya.

Dengan menggunakan *teleconference* penyaluran informasi akan menjadi lebih *efficient.* Data yang digunakan pada saat *teleconference* di kirim secara *realtime* atau di *stream* secara *live.* Dengan menggunakan *teleconference* penyaluran informasi akan menjadi lebih *efficient*, manfaat yang didapatkan dengan menggunakan *teleconference* adalah: [7]

- Menghemat waktu.
- Biaya operasional yang lebih rendah.
- *Accessible.*
- *Audience* yang lebih besar.
- *Adaptable*.
- Interaktif.

Syarat-syarat yang diperlukan untuk melakukan *teleconference* yang baik adalah sebagai berikut: [3]

- Ruangan khusus digunakan untuk *teleconference*
- Koneksi internet minimal 2mbps
- Terminal untuk partisipasi individual, misal komputer
- Peralatan audio dan video seperti *webcam* dan *microphone*

2.2 Nginx Server

Dibuat oleh *Igor Sysoev* pada tahun 2002 di Russia, *Nginx* [*engine-x*] adalah *HTTP* dan *reverse proxy*, *mail proxy server*, dan *TCP/UDP server* yang sudah gunakan oleh *website-website* terkenal di Russia seperti *Yandex*, *Mail.Ru*, *VK*, dan *Rambler*. Menurut *Netcraft*, *Nginx* telah melayani dan menjadi *proxy* 25,14 persen dari *website-website* tersibuk di dunia pada tahun 2016. *Nginx* juga digunakan oleh website-website ternama seperti *Netflix*, dan *Wordpress*. *Nginx* dirilis dengan lisensi *2-clause BSD-like*, dan bersifat *multi-platform* sehingga bisa digunakan dengan bermacam-macam sistem operasi. [9]

Server administrator menyukai nginx karena sifatnya yang *modular* [8] sehingga bisa lebih di kembangkan dengan berbagai fitur tambahan.

Nginx telah dikenal sebagai *general purpose* web server yang kokoh dan *scalable*. Menjadi pilihan utama untuk *webmaster*, dan *startup founder*, karena arsitektur *nginx simple* tetapi *expandable*. *Nginx* memberikan banyak fungsi yang berguna seperti kompresi *on the fly* dan *caching*. *Nginx* juga di dukung oleh komunitas aktif yang besar, dan serta didukung juga oleh perusahaan konsultan, oleh karena itu *nginx* secara aktif didukung dan dikembangkan. *Nginx* terintegrasi dengan teknologi web yang sudah ada seperti *Apache web server* dan *PHP*. [6]

Nginx bisa digunakan sebagai *standalone web server*, atau sebagai *proxy* untuk *web server* lain, sehingga beban *network* dari *webserver* lain bisa di *offload* ke *Nginx*, mengurangi beban dan meningkatkan konsistensi jaringan. [2]

2.3 RTMP

RTMP atau *Real-Time Messaging Protocol* adalah *open standard* yang menjadi sistem untuk mengirimkan *media* secara *on demand* atau *live* di antara aplikasi yang menggunakan Adobe *Flash*.

Awalnya diciptakan oleh Macromedia yang sekarang sudah di akuisisi oleh Adobe, Adobe mempublikasi sebagian spesifikasi *RTMP* untuk penggunaan publik, Adobe tidak mempublikasi semua bagian dari RTMP untuk melindungi bagian *security* dari RTMP dan *customer* yang menggunakan *feature security* ini. [1]

Adobe real time messaging protocol (RTMP) menyediakan sebuah layanan *bidirectional message multiplex* diatas *stream transport* yang *reliable*, seperti TCP, dimaksudkan untuk membawa *stream parallel* dari *video, audio, dan data message*, serta informasi terkait waktu, antar sepasang *peer* yang berkomunikasi.

Implementasi biasanya memberikan prioritas yang berbeda tergantung dari *message class* yang berbeda, yang dapat mempengaruhi urutan *message* apa yang akan dikirimkan ke *stream transport* yang mendasari komunikasi ketika kapasitas *transport* dibatasi. [10]

RTMP mendukung *container MP4 dan FLV streaming* secara *live* atau *video on demand* dan mempunyai beberapa keuntungan dari *HTTP* yaitu:

- *RTMP* bisa melakukan *live streaming.*
- *RTMP* bisa melakukan *dynamic streaming*, sehingga kualitas gambar bisa disesuaikan dengan kapasitas *bandwidth.* [5]

2.4 Flash

Dimulai sebagai *Future Splash Animator software* pada tahun 1995, pada tahun 1996 *software* ini dibeli oleh *macromedia* dan diperkenalkan kembali sebagai *flash*, *software* ini fokus digunakan untuk membuat animasi dan konten dinamis yang dapat ditampilkan di internet, terkenal digunakan oleh *youtube* untuk *video streaming*, *flash* menjadi format standar untuk *online multimedi*a. Pada tahun 2005 *macromedia* diakuisisi oleh *Adobe*. [11]

2.5 H.264

Sekarang ini harga untuk *processing power* dan *memory* sudah berkurang, *network support* untuk *video* sudah mulai menyebar dan perkembangan dalam *video* teknologi juga sudah dimulai, kebutuhan akan *standard* untuk teknologi *video* kompresi dengan efisiensi yang tinggi dan kokoh dalam *network environment* pun semakin tinggi. Dengan ini maka *ITU-T Video Coding Experts Group* dan *ISO/IEC Moving Picture Experts Group* membuat *Joint Video Team* pada tahun 2001 untuk menciptakan *standard* baru, yaitu H.264. [4].

3. DESAIN SISTEM

Dalam bab ini akan dijelaskan mengenai desain sistem untuk pengembangan *broadcast server*.

3.1 Desain Arsitektur

Desain arsitektur untuk pengembangan *server*. Bagian yang akan dikembangkan dari skripsi ini adalah *broadcast server*, HTTP *server* dan *website* yang nantinya akan di *serve* oleh HTTP *server*.

3.2 Desain Server

Akan menggunakan 2 buah server yang fungsinya di bagi menjadi *broadcast server* dan HTTP *server*.

3.2.1 Desain Broadcast Server

Broadcast server berfungsi untuk memproses *video stream* yang nantinya akan di gunakan oleh *broadcast controller* atau operator untuk menggabungkan *stream* menjadi satu untuk di teruskan ke HTTP *server* untuk di *view.*

Broadcast server akan di buat dengan spesifikasi sebagai berikut:

- *Intel Core* i7 3610 QM
- 4GB DDR3 RAM
- *Nvidia GeForce* GT650M
- 1 TB HDD
- *Linux Mint*

Spesifikasi server yang digunakan didasarkan oleh *system* ASUS-N46VZ, dan *linux mint* digunakan karena *user interfacenya* yang familiar, dan juga berbasis *debian* dan *ubuntu* yang mudah digunakan dan *mudah* untuk penggunaan melalui *command line*, serta *software backend* yang akan digunakan yaitu *Nginx* harus dilakukan *compile* dari *source* agar bisa digunakan dengan *nginx-rtmp module*.*Broadcast serve*r akan menggunakan *Nginx server* sebagai *backend*, dan untuk melakukan RTMP *stream*, *nginx-rtmp module* akan di integrasikan ke dalam *nginx* sewaktu proses instalasi.

Broadcast server akan di kembangkan dengan fungsi-fungsi sebagai berikut:

- Melakukan *transcoding stream video footage* menjadi *format* mp4 (h264) dengan resolusi 480p. *(Transcoding).*
- Menjadi penghubung antara *stream* ke *content distribution* atau *server Nginx* lain *(Relay).*
- Menjadi *broadcast server* yang berdiri sendiri tanpa bergantung kepada *platform streaming* yang sudah ada.
- Bisa melakukan *archiving* terhadap *stream* yang ada.
- Bisa melihat *stream* secara *video on demand*.

3.2.2 Desain HTTP Server

HTTP *server* berfungsi sebagai sistem yang akan menyediakan *website* yang nantinya berfungsi untuk membuka *live streaming* atau membuka *video* arsip dan melakukan *streaming* secara *on-demand*.HTTP *server* dan *website* akan di buat menggunakan spesifikasi sebagai berikut:

- Menggunakan *Nginx* / *Apache* HTTP *server* sebagai *backend* untuk *website.*
- Menggunakan standar HTML 5.
- Menggunakan PHP.
- Menggunakan *plugin flash* untuk mendukung RTMP.
- Menggunakan *web player* berbasis *flash*.

3.3 Desain Client

Client akan mempunyai dua fungsi, yaitu menjadi *participant* dalam *broadcast*, atau menjadi *broadcast controller* yang akan bertanggung jawab untuk mengontrol apa yang akan ditampilkan dalam *broadcast stream*.

Participant bisa melakukan *broadcast* dengan menggunakan *software* apapun dengan ketentuan *software* tersebut bisa melakukan *streaming* dengan protokol RTMP.

Broadcast controller berfungsi untuk mengambil *stream* yang di lakukan oleh *participant* dan menggabungkan beberapa stream yang berbeda tersebut menjadi sebuah *stream* yang sudah tergabung,.

Software yang direkomendasikan untuk *broadcast controller* adalah OBS (*Open Broadcaster Software*) yang tersedia untuk sistem operasi *Linux, Windows,* dan *MacOS.*Spesifikasi minimum yang dibutuhkan untuk *client* adalah sebagai berikut:

- *4th Generation (Haswell) intel core i3 series processor.*
- 4GB DDR 3 RAM.
- *Intel* HD *Graphics.*
- *Linux, Windows, or MacOS*.

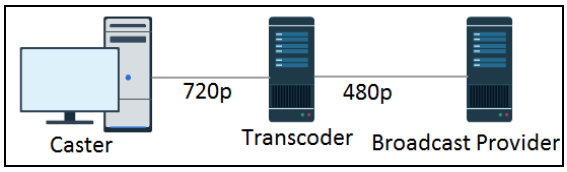
Spesifikasi didasarkan dengan menggunakan minimum *processor* dari intel generasi *haswell* karena pada generasi *haswell.*

Intel mengenalkan teknologi *quicksync, dimana didalam chip intel* pada saat itu ditanamkan *hardware decoding* dan *encoding video* sehingga *video processing* bisa dilakukan secara *hardware* untuk mengurangi beban CPU, serta generasi ke empat yaitu *haswell* memberikan *support* untuk *hardware* H.264.

Untuk system yang mempunyai *dedicated* GPU Nvidia bisa menggunakan *hardware encoder* dari *dedicated* GPU. Dengan menggunakan GPU *encoder*, beban CPU bisa dikurangi.

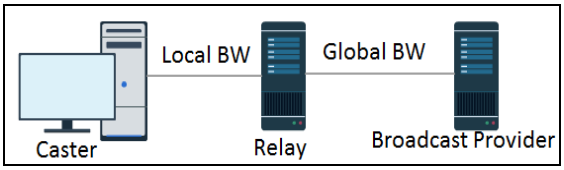
3.4 Skenario Penggunaan

Mengikuti fungsi-fungsi dari *broadcast server* yang sudah diuraikan, maka penggunaan *server* akan diilustrasikan pada gambar 1, 2, dan 3.



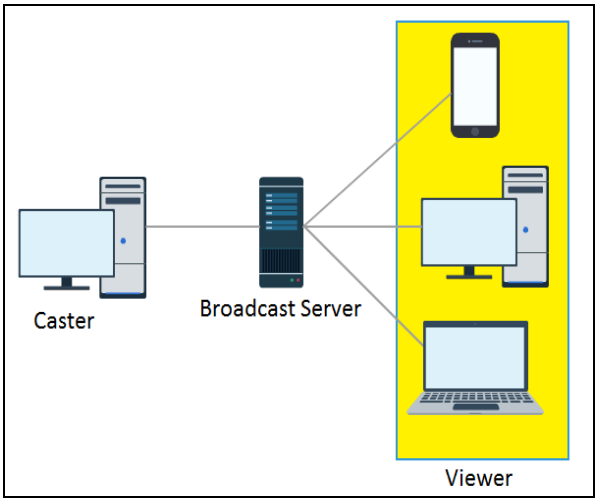
Gambar 1 Fitur *Transcoding*

Fitur ini akan mengubah resolusi *video* yang di *stream* menjadi 480p.



Gambar 2 Fitur *Relay*

Fitur ini akan membuat *broadcast server* meneruskan *live stream* ke *server* atau *platform streaming* lain.



Gambar 3 Fitur *Independent Broadcast Server*

Fitur ini akan membuat *broadcast server* menjadi *platform* sendiri dan bisa langsung digunakan untuk *live streaming* ke penonton.

4. PENGUJIAN SYSTEM

Memulai pengujian *system*, pengujian yang akan dilakukan adalah *pengujian teleconference*, *delay* yang didapat dalam *teleconference* yang dilakukan oleh dua sampai empat orang, pengujian fitur-fitur *server*, dan pengujian konfigurasi.

4.1 Pengujian Teleconference

Pada pengujian ini akan dicoba menggunakan *client* untuk melakukan *teleconference*, jumlah participant yang akan di uji adalah dua sampai empat *participant*, tetapi karena keterbatasan jumlah *system* saat penulisan buku ini, yang akan ditampilkan hanyalah *ujicoba* penggunaan *teleconference* dengan dua participant.

Mengikuti konfigurasi dari implementasi *client participant*, tiap *client* bisa terlebih dahulu membagi untuk melakukan RTMP *stream* ke bagian *application stream* satu atau dua, dan menentukan *stream code* yang akan digunakan untuk lebih mudah melakukan proses penggabungan *stream*.Setelah kedua *participant* sudah melakukan stream ke *broadcast server*, *broadcast controller* bisa menggabungkan *stream* dengan *scene collection* untuk dua *participant* dan mengubah *properties* dari setiap sources dengan menggunakan URL yang berisi *rtmp application* dan *stream code* yang sudah ditentukan sebelumnya, dan melakukan stream ke RTMP *application final*, dan *stream code* di kirimkan kembali kepada *client* dan pengguna yang akan menonton.

Perlu diketahui bahwa *bitrate* akan mempegaruhi *bandwidth* yang digunakan untuk membuka dan melihat stream, jadi *bitrate* disesuaikan dengan kecepatan dari koneksi yang digunakan oleh *system*. Tampilan stream yang sudah di gabung oleh obs pada gambar 4.

Gambar 4 OBS Dua Participant Yang Sudah Di Gabung.

Pengguna bisa menggunakan *website*, *software*, atau *apps* untuk membuka *stream*, *stream* bisa dibuka melalui *PC*, dan *smartphone*.

Untuk membuka *stream* melalui *website*, hanya tinggal membuka halaman *website* dan mengisi *stream code* yang ingin dilihat. Untuk membuka *stream* dengan resolusi yang rendah bisa memilih *low resolution stream* pada *homepage website* dan melakukan hal yang sama seperti membuka *stream* yang normal.

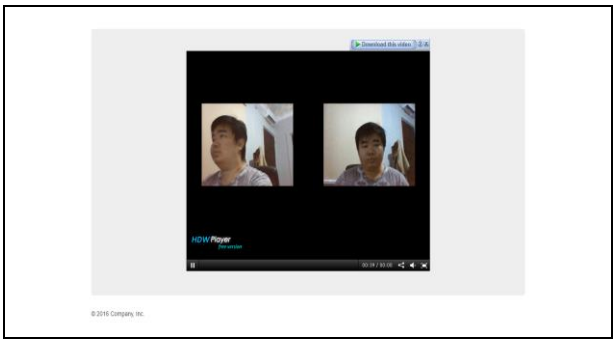
Untuk melihat *stream* menggunakan *software* bisa menggunakan *software* seperti *video player* VLC yang bisa digunaan melalui *PC* atau *smartphone android* dan IOS.

Melakukan uji coba membuka *stream* menggunakan *browser*, semua *web browser* yang mendukung *flash* dapat digunakan untuk membuka *browser*, karena *web video player* yang digunakan dibuat berbasis *flash*.

Dari 4 browser yang dilakukan ujicoba dalam membuka *stream*, yaitu *Microsoft Edge*, *Google Chrome*, *Mozilla Firefox*, dan *Opera*, semua browser bisa membuka *stream* dengan baik, tetapi untuk *Firefox* dan *Opera*, pengguna perlu melakukan instalasi *plugin adobe flash player*, karena kedua *browser* ini tidak mempunyai *plugin flash* secara *default* atau mempunyai *plugin flash* yang *outdated* saat melakukan instalasi.

Untuk melihat *stream* menggunakan *VLC*, bisa membuka *network stream* dari *software* dan *aplikasi* tersebut, dan mengisikan URL RTMP *stream*.

Tampilan stream pada web bisa dilihat pada gambar 5.



Gambar 5 Contoh Tampilan Streaming Pada Website.

4.2 Pengujian Delay Teleconference

Pada bagian ini akan dilakukan ujicoba untuk *delay* dalam *stream teleconference*, dan mencari aspek yang mempengaruhi *delay*. Ujicoba dilakukan dengan cara menggerakan *object* pada *web cam* dan menghitung waktu *delay* yang didapat ke *broadcast server*, dan *delay* yang didapat setelah *broadcast controller* menggabungkan *stream*.

Agar tidak membebani *server* dan menjaga konsistensi *stream* pada jumlah dua sampai empat *participant*, *stream* akan dibuat dengan *bitrate* 1000kbps, dan menggunakan jaringan nirkabel. Hasil dari ujicoba akan di uraikan pada tabel 1.

Tabel 1 Hasil Ujicoba Delay Teleconference

Jumlah Participant	Delay Broadcast Server	Delay Final	Overhead
2 (Base)	3 Detik	6,5 Detik	-
3	3,2 Detik	6,6 Detik	1.5 %
4	3,3 Detik	6,8 Detik	2.9 %

Dari hasil ujicoba bisa ditarik kesimpulan bahwa jumlah *participant* tidak berpengaruh signifikan terhadap *delay* yang didapat pada *stream* dan perbedaan *delay* yang didapat dari hasil ujicoba memperlihatkan perubahan *delay* dibawah 10%, melihat bahwa ujicoba dilakukan pada jaringan nirkabel, dengan menggunakan *empat participant* bisa dilihat bahwa ada pengaruh dari kualitas jaringan pada *delay* yang didapat, serta semua *stream* yang didapat oleh *broadcast server* juga akan diikuti dengan proses transcoding yang bisa menjadi aspek yang mempengaruhi *delay stream*, karena CPU dari *broadcast system* akan bekerja *transcoding* dan melakukan *transmit* bersamaan.

4.3 Pengujian Fitur Server

4.3.1 Pengujian Transcoding

Dalam pengujian ini, *server* akan dikonfigurasi untuk melakukan transcoding yang berupa menurunkan resolusi dari *source* menjadi resolusi 480p dan setelah itu akan di *publish* pada *stream small*, pengujian juga dilakukan bersamaan dengan saat ujicoba *teleconference*, karena *stream* hanya bisa dilakukan ke satu tujuan saat melakukan stream, maka *Nginx* dan *FFmpeg* lah yang akan melakukan *stream* ke *application small*, untuk konfigurasi lebih lanjut dari *transcoding* bisa mengikuti dokumentasi dari *encoder FFmpeg*. Untuk melihat apakah hasil transcoding berhasil, setelah melakukan stream, perlu membuka halaman *statistic* RTMP, apabila *stream code* yang digunakan muncul pada *application small* dan resolusi dari *video* sudah menurun, maka *transcoding* sudah berhasil dilakukan.

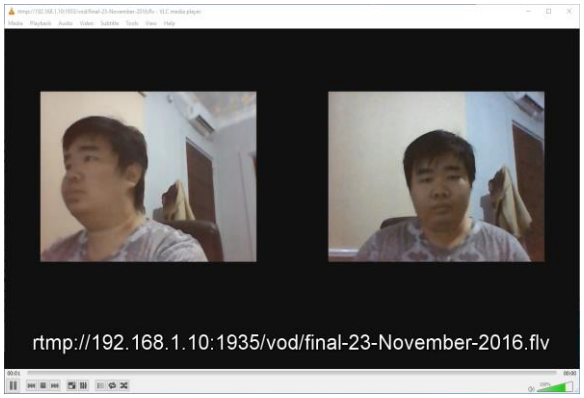
4.3.2 Pengujian Stream Archiving

Pengujian dilakukan untuk mencoba fitur *recording* dari *broadcast server*, pada pengujian ini *broadcast server* dengan RTMP *application final* akan digunakan untuk melakukan *record* dari *stream* yang *aktif*, recording akan berupa *video format flv* dan di simpan dengan format nama <*streamkey-tanggal-bulan-tahun.flv*.

Setelah melakukan sesi *stream* selama beberapa menit, *stream di shut down* dan bisa membuka *folder* tempat menyimpan *archiving*, apabila *file* dengan format nama file tadi ada dalam folder dan isi *video* sudah sama dengan sesi *stream* yang dilakukan sebelumnya, maka *archiving* sudah berhasil.

4.3.3 Pengujian Video on Demand

Dalam ujicoba ini akan dicoba untuk memutar *file archive* melalui *website* dan VLC, untuk *website*, bisa memilih VoD pada menu navigasi di *home page viewer*, dan mengisi *textbox* dengan nama *file* dari hasil *recording* dengan menggunakan *ekstensi format* atau hanya nama *file* saja., Tampilan untuk VLC pada gambar 6.



Gambar 6 Contoh Tampilan VoD Pada VLC.

4.3.4 Pengujian Relay

Untuk menguji *relay*, *stream* akan langsung dibuka pada *application final*, dan dengan mengamati *application relay* yang akan berjalan, pada *application final* sudah dikonfigurasi untuk mengubah *stream code* yang di *relay* menjadi *final relay*, tetapi masih dalam satu *system*.

Fungsi ini dapat digunakan untuk menghubungkan beberapa RTMP *server* atau digunakan untuk menerbitkan *stream* ke *platform* lain yang menerima RTMP.

Apabila *stream* sudah berjalan dan pada *rtmp stats* terdapat stream dengan *code final relay*, maka fitur *relay* sudah berfungsi

4.4 Pengujian Stream

Pada bagian ini akan di ujicoba konfigurasi *stream*, dampak dari *konfigurasi* terhadap kualitas *stream*, dan beban yang diberikan pada *system*.

4.4.1 Pengujian Bitrate Terhadap System

Bitrate adalah jumlah *stream data* yang akan digunakan untuk *live streaming*, semakin tinggi *bitrate* maka semakin tinggi pula *data* yang harus dikerjakan oleh CPU, dan *bandwidth* yang digunakan untuk mengirimkan dan membuka *stream* tersebut.

Percobaan akan di ujicoba dengan cara melakukan satu *stream* kepada *broadcast server* dengan menggunakan *bitrate* kelipatan 500kbps, menggunakan *encoder software* x264 dan *audio* AAC 160kbps pada resolusi 720p.

Spesifikasi *system* yang akan digunakan oleh *client* pada ujicoba ini adalah *Intel Celeron N2840* yang berupa CPU *dual core* kelas *entry level*.

Spesifikasi ini digunakan untuk mencari tahu *bitrate* seperti apa yang dapat digunakan untuk pengguna dengan *hardware* yang sudah lama atau *entry level* seperti *system client* yang di ujicoba, dan juga mencari tahu pengaruh *bitrate* terhadap CPU dan kualitas *FPS stream* yang akan didapatkan nantinya.Yang di amati dari ujicoba ini adalah, presentase penggunaan CPU, minimum dan maksimum *framerate* dari *stream*, presentase dari *dropped frame* atau *frame* yang tidak terkirim dengan *total frame* dari *stream*, dan penggunaan *bandwith*.

Data didapat dengan mengamati *statistic stream* dari OBS. *Stream* akan dijalankan selama dua menit untuk ujicoba. Hasil dari ujicoba bisa dilihat pada tabel 2

Tabel 2 Hasil Ujicoba Delay Teleconference

Bitrate	CPU Usage	Min FPS	Max FPS	Drop Frame	Bandwitdh
500	85~ %	27	30	0 %	550 ~ kbps
1000	85~ %	25	30	0 %	1000 ~ kbps
1500	85~ %	21	30	0 %	1500 ~ kbps

Dari hasil ujicoba bisa di Tarik kesimpulan bahwa semakin tinggi *bitrate*, maka semakin tinggi beban yang akan dikerjakan oleh CPU, dan karena CPU yang terlalu terbebani maka kualitas dari *stream* juga akan turun, ini bisa dilihat dari *minimum* FPS yang semakin menurun ketika *bitrate* di tambah. *Bitrate* juga mempengaruhi *bandwidth* yang akan digunakan untuk pengiriman *stream*.

Solusi yang bisa digunakan terutama pada system ini untuk mengurangi beban CPU adalah menggunakan *hardware encoder* yang ternyata didukung oleh *Intel Celeron N2840* ini, hardware *encoder* yang digunakan adalah *Intel Quick Sync Technology*.

4.4.2 Uji Bitrate Terhadap Detail Gambar

Ujicoba dilakukan untuk membandingkan kualitas gambar *stream* yang didapat dengan *bitrate* yang berbeda. Pengujian akan dilakukan dengan cara membuat *stream* dengan *display capture* dari *video game*, alasan menggunakan *capture* dari *video game* adalah karena *video game* berupa *video* yang selalu *bergerak* dan baik digunakan untuk membandingkan kualitas gambar.

Pengujian akan digunakan dengan konfigurasi bitrate 1000, 5000, dan 10000. Pembandingan gambar dilakukan dengan cara membuka *file recording* dari tiap *bitrate* dan dibandingkan gambar *video* yang didapat.

Pada *bitrate* 1000 detail pada gambar *video* banyak yang tampil berbentuk agak kotak atau *pixelated*, dan pada *bitrate* 5000 gambar sudah mulai tampil dengan baik tapi masih ada beberapa tempat yang kekurangan detail, untuk 10000 gambar yang ditampilkan sangat bagus, meskipun masih belum bisa seperti pada tampilan asli *monitor* tapi semua *detail* gambar sudah terlihat.

Karena game yang digunakan adalah game yang bergerak dengan cepat maka pada *bitrate* 1000 tidak dapat menampilkan semua *detail* yang ada, tetapi untuk konten yang tidak bergerak cepat seperti *teleconference*, masih baik untuk digunakan.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

- Karena konfigurasi *out of the box Nginx* yang dikompilasi dari *source* berbeda dengan *Nginx* yang di instalasi melalui *repository distro* maka terjadi kendala dalam implementasi komponononen *FastCGI* PHP, oleh karena itu solusinya menggunakan *Apache* sebagai HTTP *server*, Agar *website* bisa dibuat sesuai
- *Server* sudah bisa berjalan dengan baik dan fitur-fitur sudah bisa digunakan.

- Pada *video on demand*, *stream video* ditampilkan pada *aspect ratio 4:3* apabila distream melalui *website*, hal ini tidak diketahui penyebabnya, tetapi apabila distream melalui *software* seperti *vlc*, *video* tampil sesuai dengan *aspect ratio* yang dimiliki oleh *video* tersebut.
- Jumlah *participant* dalam *teleconference* akan menambah *delay*, tetapi tidak berpengaruh signifikan tambahan karena *delay* dari dua ke empat *participan*t hanya 2.9 %, ini dikarenakan *server* harus men*transcoding stream* dari participant, menggabungkan, dan melakukan *broadcast* lagi *stream* yang sudah digabungkan tersebut.
- Semakin tinggi *bitrate* yang digunakan dalam menjalankan *stream*, maka semakin tinggi beban yang harus diproses oleh *cpu* dan semakin banyak pula *bandwidth* yang akan digunakan untuk mengirimkan *video* tersebut.
- Fitur *relay* hanya dilakukan ujicoba secara *local* dan sudah berfungsi dengan benar.

5.2 Saran

- Simplifikasi dari proses konfigurasi dibutuhkan, ini dikarenakan untuk melakukan *stream, client* harus melakukan berbagai konfigurasi.
- Perlu ada fitur untuk melakukan *export* dan *import* konfigurasi untuk mempermudah proses konfigurasi pada *client* yang baru.
- Perlu ada dibuat program yang dikhususkan untuk *RTMP teleconference*, dikarenakan OBS biasanya digunakan bukan untuk *teleconference* tapi lebih untuk *broadcast* seperti televisi.

6. DAFTAR PUSTAKA

[1] Adobe. n.d. *Real-Time Messaging Protocol (RTMP) specification*. URI=http://www.adobe.com/devnet/rtmp.html

[2] Garret, O. 2015, October 9. *NGINX vs. Apache: Our View of a Decade-Old Question* URI=https://www.nginx.com/blog/nginx-vs-apache-our-view/

[3] ITU-T. 1996, July 1. *F.702 : Multimedia conference services.*URI=https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-F.702-199607-I!!PDF-E&type=items

[4] ITU-T. 2016, February 13. *H.264 : Advanced video coding for generic audiovisual services.* URI=https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-201602-S!!PDF-E&type=items

[5] JW Player. n.d. *About RTMP Streaming*. URI= https://support.jwplayer.com/customer/portal/articles/1430349-about-rtmp-streaming

[6] Kholodkov, V. 2015. *Nginx Essentials.* Birmingham: Packt Publishing.

[7] Lane, C. n.d. *The Distance Learning Technology Resource Guide*. URI= http://www.tecweb.org/eddevel/edtech/teleconf.html

[8] Nedelcu, C. 2015. *Nginx HTTP Server Third Edition.* Birmingham: Packt Publishing.

[9] Nginx. n.d. URI=http://nginx.org/en/

[10] Parmar, H., & Thornburgh, M. 2012, December 21. *RTMP Specification.* URI= http://www.adobe.com/devnet/rtmp.html

[11] Rocheleau, J. 2015, September 5. *A History Lesson on the Rise and Fall of Adobe Flash*. URI= https://speckyboy.com/a-history-lesson-on-the-rise-and-fall-of-adobe-flash/