ANALISIS KINERJA VIDEO *STREAMING* BERBASIS *DYNAMIC ADAPTIVE STREAMING OVER HTTP* (DASH) PADA AREA JARINGAN KAMPUS (STUDI KASUS: PTIIK UNIVERSITAS BRAWIJAYA)

SKRIPSI

LABORATORIUM JARINGAN KOMPUTER

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:
PUTRANTI PUJI PURNOMO
105060801111071

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
MALANG
2014

LEMBAR PERSETUJUAN

ANALISIS KINERJA VIDEO *STREAMING* BERBASIS *DYNAMIC ADAPTIVE STREAMING OVER HTTP* (DASH) PADA AREA JARINGAN KAMPUS

(STUDI KASUS: PTIIK UNIVERSITAS BRAWIJAYA)

SKRIPSI

LABORATORIUM JARINGAN KOMPUTER

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

PUTRANTI PUJI PURNOMO

105060801111071

Telah diperiksa dan disetujui oleh:

Pembimbing I	**Pembimbing II**
Achmad Basuki, ST., M.MG., Ph.D.	**Eko Sakti Pramukantoro, S.Kom, M.Kom.**
NIP. 19741118 200312 1 002	**NIK. 860805 06 1 1 0252**

LEMBAR PENGESAHAN

ANALISIS KINERJA VIDEO *STREAMING* BERBASIS *DYNAMIC ADAPTIVE STREAMING OVER HTTP* (DASH) PADA AREA JARINGAN KAMPUS

(STUDI KASUS: PTIIK UNIVERSITAS BRAWIJAYA)

SKRIPSI

LABORATORIUM JARINGAN KOMPUTER

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh:

PUTRANTI PUJI PURNOMO

NIM. 105060801111071

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 29 Desember 2014
dan dinyatakan memenuhi syarat untuk memperoleh
gelar sarjana dalam bidang Ilmu Komputer

Penguji I	Penguji II
Kasyful Amron, ST., M.Sc.	**R. Arief Setyawan, ST., MT.**
NIP. 19750803 200312 1 003	**NIP. 19750819 199903 1 001**

Penguji III

Eko Setiawan, ST., M.Eng.
NIK. 870610 06 1 1 0256

Mengetahui,

Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Januari 2015

Mahasiswa,

PUTRANTI PUJI PURNOMO

NIM. 105060801111071

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan tugas akhir dengan judul "Analisis Kinerja Video *Streaming* Berbasis *Dynamic Adaptive Streaming over HTTP* (DASH) Pada Area Jaringan Kampus" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan tugas akhir ini dengan baik dan tepat waktu.

Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Kedua orang tua yang telah memberikan dukungan, baik moril dan materil.
2. Bapak Achmad Basuki, ST., MMG., Ph.D., selaku dosen pembimbing I yang telah banyak memberikan ilmu dan saran untuk laporan skripsi ini.
3. Bapak Eko Sakti Pramukantoro, S.Kom., M.Kom., selaku dosen pembimbing II yang juga banyak memberikan ilmu dan saran untuk laporan skripsi ini.
4. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
5. Semua teman-teman PTIIK yang telah membantu memberi saran dan kritik atas laporan ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa PTIIK Universitas Brawijaya.

Malang, Desember 2014

Penulis

ABSTRAK

Putranti Puji Purnomo. 2014. Analisis Kinerja Video *Streaming* Berbasis *Dynamic Adaptive Streaming Over HTTP* (DASH) Pada Area Jaringan Kampus (Studi Kasus: PTIIK Universitas Brawijaya). Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing Achmad Basuki, ST., MMG., Ph.D. dan Eko Sakti Pramukantoro, S.Kom., M.Kom.

Video s*treaming* saat ini sudah menjadi suatu hal yang biasa bagi setiap orang untuk saling bertukar informasi secara audio-visual. Namun tidak semua pengguna memiliki kemampuan untuk mengakses konten-konten *streaming* dengan lancar, karena adanya perbedaan kondisi jaringan (*bandwidth*) di sisi pengguna. Adanya solusi yang mampu beradaptasi dengan kondisi dinamis di sisi pengguna bernama *Dynamic Adaptive Streaming over HTTP* (DASH), maka kendala-kendala yang sering muncul saat proses *streaming* dapat terselesaikan.

DASH merupakan sebuah layanan *client-server* dengan definisi konten DASH dinyatakan di dalam bentuk file XML (MPD). DASH membutuhkan *web server*, file MPD, dan sebuah *player* untuk dapat beradaptasi dengan kondisi jaringan yang dinamis (fluktuatif). Peran DASH *player* pada sisi pengguna sangat besar karena dapat menginterpretasikan secara mandiri berdasarkan ketersediaan konten di dalam *server*, yang dapat diminta sesuai dengan kondisi akses jaringan (*bandwidth*) pengguna.

Hasil eksperimen dan analisis pada penelitian ini menunjukkan bahwa berdasarkan jumlah rata-rata waktu *delay* yang lebih kecil di sisi pengguna, maka dapat dikatakan bahwa layanan video *streaming* menggunakan DASH lebih baik daripada tanpa menggunakan DASH. Selain itu, DASH juga dapat beradaptasi pada kondisi jaringan (*bandwidth*) yang dinamis, salah satunya pada area jaringan kampus PTIIK dengan rata-rata representasi *bandwidth* yang digunakan oleh DASH adalah 4191 Kbps, atau dengan id pertama atau yang paling baik.

Kata Kunci: video *streaming*, HTTP *streaming*, *dynamic adaptive streaming*, DASH

ABSTRACT

Putranti Puji Purnomo. 2014. Analisis Kinerja Video *Streaming* Berbasis *Dynamic Adaptive Streaming Over HTTP* (DASH) Pada Area Jaringan Kampus (Studi Kasus: PTIIK Universitas Brawijaya). Information Technology and Computer Science Program, Brawijaya University, Malang. *Advisor*: Achmad Basuki, S.T., MMG., Ph.D. and Eko Sakti Pramukantoro, S.Kom., M.Kom.

Video streaming is now becoming a common thing for people to exchange audio-visual's information. However, not all users have the ability to access streaming content smoothly, due to the differences in network conditions (bandwidth) in the client side. The existence of a solution that is able to adapt to a dynamic condition on the user side, which is known as Dynamic Adaptive Streaming over HTTP (DASH), the constraints that often arise during the streaming process can be resolved.

DASH is a server-client service where definition of DASH content expressed in the XML file (MPD). DASH requires web server, MPD file, and player to be able to adapt with dynamic network conditions (fluctuating). The role of player DASH on the user side is very large, because it can be interpreted independently by the availability of content on the server, which is adapted to the conditions of access network (bandwidth) the current user.

The experiment and analysis results in this study showed that based on the smaller average delay on the user side, it can be said that video streaming service using DASH is better than video streaming without DASH. Additionally, DASH also can adapt to the network condition dynamically, such as in the campus area network PTIIK, with an average representation of bandwidth used by DASH is 4191 Kbps, or the first id or the best.

Keywords: *video streaming, HTTP streaming, dynamic adaptive streaming, DASH*

DAFTAR ISI

KATA PENGANTAR i
ABSTRAK ii
ABSTRACT iii
DAFTAR ISI iv
DAFTAR GAMBAR vi
BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 4
1.6 Sistematika Penulisan 4
BAB II 7
TINJAUAN PUSTAKA 7
2.1 Kajian Pustaka 7
2.2 Video *Streaming* 10
2.2.1 *Adaptive Streaming* 10
2.2.2 *Video on Demand* (VoD) 11
2.2.3 Protokol HTTP 11
2.2.4 *Dynamic Adaptive Streaming over HTTP* (DASH) 13
2.2.4.1 *Media Presentation Description* (MPD) 18
2.2.4.2 Format Segmen 25
2.3 Emulasi Jaringan 26
2.3.1 NetBalancer 26
2.3.2 Trickle 27
BAB III 29
METODE PENELITIAN 29
3.1 Analisis Kebutuhan 29
3.2 Instalasi *Web Server* 32

3.3 Alur Proses DASH *Player* 33
3.4 Implementasi Lingkungan Berbasis DASH 36
3.5 Lingkungan Pengujian 37
3.5.1 Pengumpulan Dataset 39
3.5.2 Skenario Pengujian 42
3.6 Hasil dan Pembahasan 44
3.7 Pengambilan Kesimpulan 45
BAB IV 46
IMPLEMENTASI 46
4.1 Jaringan (*Bandwidth*) 46
4.2 Instalasi dan Konfigurasi *Web Server* 46
4.3 File MPD (*Media Presentation Description*) 48
4.4 *Player* 49
BAB V 53
HASIL DAN PEMBAHASAN 53
5.1 Uji Coba DASH Pada Kondisi Simulasi 53
5.1.1 DASH Pada Kondisi *Bandwidth* Dinamis (Fluktuatif) 53
5.1.2 Perbandingan *Delay* 55
5.1.3 Perubahan (*Switching*) Representasi Video 57
5.2 Uji Coba DASH Pada Jaringan Kampus 59
5.2.1 Gedung A 59
5.2.2 Gedung B 61
5.2.3 Gedung C 63
5.2.4 Gedung D 64
5.2.5 Gedung E 65
BAB VI 67
PENUTUP 67
6.1 Kesimpulan 67
6.2 Saran 68
DAFTAR PUSTAKA 69
LAMPIRAN 2 75

DAFTAR GAMBAR

Gambar 2.1 Cara Kerja HTTP 12
Gambar 2.2 Arsitektur DASH 14
Gambar 2.3 Lingkup MPEG DASH 16
Gambar 2.4 Struktur MPD 18
Gambar 2.5 Tampilan Aplikasi Total Video Converter 19
Gambar 2.6 Tampilan Aplikasi My MP4Box GUI 21
Gambar 2.7 *Script* Pembuatan MPD 22
Gambar 2.8 Isi File MPD 23
Gambar 2.9 Tampilan Aplikasi MP4Client (Windows) 25
Gambar 2.10 Aplikasi *Bandwidth Limiter* NetBalancer 27
Gambar 2.11 *Script* Trickle 28
Gambar 3.1 Alur Proses DASH *Player* 34
Gambar 3.2 Uji Coba DASH Kondisi Simulasi 38
Gambar 3.3 Uji Coba DASH Pada Jaringan Kampus 38
Gambar 3.4 Pengumpulan File Video 40
Gambar 4.1 Tampilan Awal Apache *Web Server* 47
Gambar 4.2 Tampilan File `car.mpd` 49
Gambar 4.3 Tampilan Ekstrak File *Player* HTML5 50
Gambar 4.4 Tampilan Aplikasi *Player* HTML5 Berbasis DASH 51
Gambar 4.5 Tampilan Pesan *Log* Pada *Player* 52
Gambar 5.1 Uji Coba DASH Kondisi *Bandwidth* Dinamis (Fluktuatif) 54
Gambar 5.2 Perbandingan Waktu *Delay Streaming* DASH dan Tanpa DASH 56
Gambar 5.3 Perbandingan Perubahan (*Switching*) Representasi Video *Streaming* DASH dan Tanpa DASH 57
Gambar 5.4 Uji Coba DASH Ketika Diakses di Gedung A (PTIIK-UB) 60
Gambar 5.5 Uji Coba DASH Ketika Diakses di Gedung B (PTIIK-UB) 62
Gambar 5.6 Uji Coba DASH Ketika Diakses di Gedung C (PTIIK-UB) 63
Gambar 5.7 Uji Coba DASH Ketika Diakses di Gedung D (PTIIK-UB) 64
Gambar 5.8 Uji Coba DASH Ketika Diakses di Gedung E (PTIIK-UB) 66

DAFTAR TABEL

Tabel 3.1 Karakteristik Video untuk Simulasi DASH 41
Tabel 3.2 Karakteristik Audio untuk Simulasi DASH 41
Tabel 3.3 Skenario Pengujian 44

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi saat ini berkembang semakin pesat dengan adanya fasilitas Internet. Dahulu, orang bertukar informasi hanya dapat melalui teks ataupun gambar, tetapi sekarang hal tersebut sudah mengalami transformasi menjadi suatu bentuk yang lebih atraktif, yaitu audio dan video (audio-visual). Data yang berupa audio dan video tersebut juga dapat diunduh dan diunggah menggunakan satu antar muka berupa *web browser*. Tren aplikasi yang saat ini banyak digunakan salah satunya adalah Youtube yang mengijinkan penggunanya untuk mengakses video baik *progressive/adaptive download*, dan layanan seperti ini umumnya dikenali sebagai video *streaming* melalui protokol HTTP (Lederer, 2012).

Salah satu layanan Internet yang dapat diterapkan pada area jaringan kampus adalah layanan video *streaming*. Video *streaming* adalah sebuah teknologi yang memungkinkan sebuah file audio atau video dimainkan, dan pada saat itu pula terjadi proses perpindahan data (*playing while downloading*) (Hartanto, 2009). Video *streaming* konvensional tidak berjalan dengan sukses karena terhalang biaya yang terbilang tidak murah dan terhalang oleh infrastruktur video *streaming* konvensional yang mengakibatkan tidak semua klien dapat menikmati layanan video *streaming* dengan nyaman (Stockhammer, 2011). Penerapan layanan video *streaming* konvensional pada area jaringan kampus seringkali terkendala oleh kondisi ketersediaan *bandwidth* jaringan yang selalu dinamis setiap saatnya.

Alternatif lain yang dibutuhkan agar dapat menutupi kekurangan pada teknologi video *streaming* konvensional yaitu dengan menggunakan *adaptive streaming*. *Adaptive streaming* adalah teknologi pengiriman media yang memberikan jaminan kepada penggunanya dengan keberagaman kondisi jaringan saat itu (Miller, 2012). DASH (*Dynamic Adaptive Streaming over HTTP*) merupakan bentuk nyata dari *adaptive streaming* berbasis HTTP. DASH adalah suatu standar baru yang digunakan dalam hal pengiriman konten media, dan

mampu menangani berbagai kondisi jaringan (*bandwidth*) terutamanya di sisi klien selama *streaming* berlangsung (Muller, 2012). DASH juga tidak terbatas hanya untuk kalangan tertentu saja, dan tidak menutup kemungkinan klien untuk dapat mengaksesnya. DASH dapat mengatasi berbagai macam klien dengan beragam kondisi jaringan dan mengijinkan klien untuk dapat memainkan konten DASH yang telah disediakan. Klien tidak terbatas hanya pada kondisi jaringan yang stabil saja, melainkan DASH juga mampu untuk beradaptasi dengan kondisi jaringan yang tidak stabil (Lederer, 2012).

DASH menggunakan protokol HTTP (TCP) dan bukan RTP (UDP) untuk pengiriman kontennya, karena RTP sering kali memiliki kendala *firewalls* yang memblokir paket-paket UDP (Sodagar, 2011). Namun, HTTP juga tidak selamanya lebih baik daripada RTP, karena pengiriman konten menggunakan HTTP juga terhalang oleh kendala *delay*. Pengguna yang melakukan video *streaming* dengan menggunakan protokol HTTP (HTTP *adaptive streaming*) bisa beradaptasi dengan aliran/*stream* pada berbagai kondisi jaringan, dan juga pengiriman konten menggunakan HTTP secara signifikan memakan biaya yang lebih rendah dibandingkan dengan infrastruktur video *streaming* sebelumnya (RTP) (Rainer, 2012).

Hal penting yang harus diperhatikan di dalam DASH adalah penggunaan *player*. Pada dasarnya cara kerja *player* DASH memang sama dengan *player* HTML5 yang lain. Perbedaan keduanya hanya terletak pada penggunaan *script* pada *player* DASH untuk mengestimasi ketersediaan *bandwidth* yang akan digunakan, sedangkan pada *player* HTML5 lain tidak ada. Berdasarkan alasan tersebut, maka bukan tidak mungkin jika *player* DASH dapat memutar video dengan kualitas sesuai dengan ketersediaan *bandwidth* saat itu. Kondisi inilah yang mengakibatkan DASH dikatakan *adaptive*, karena ketika nilai *bandwidth* dibandingkan dengan nilai-nilai yang sudah ditetapkan di dalam file MPD, maka *player* akan memutar video sesuai dengan *bandwidth* yang ada. Dengan kata lain, DASH dapat membandingkan ketersediaan *bandwidth* saat itu dengan nilai *bandwidth* yang sudah ditentukan sebelumnya, berdasarkan pada kualitas video yang sudah disimpan di dalam *web server*. Hal ini berakibat pada berkurangnya

delay, karena nilai *bandwidth* yang akan digunakan telah sesuai dengan kualitas video yang akan diputar.

DASH merupakan sebuah solusi yang dapat digunakan untuk mengatasi permasalahan-permasalahan yang sering terjadi pada video *streaming* konvensional. Penelitian ini menghasilkan pengetahuan tentang sejauh mana kinerja video *streaming* berbasis DASH dapat dijalankan pada kondisi jaringan kampus (PTIIK) yang memiliki kondisi jaringan dengan ketersediaan *bandwidth* yang berubah secara dinamis (fluktuatif) setiap saat.

1.2 Rumusan Masalah

Berdasarkan paparan yang telah disebutkan pada bagian latar belakang, maka didapatkan rumusan masalah yang meliputi:

1. Bagaimana membangun lingkungan untuk sistem DASH?
2. Bagaimana mengevaluasi kinerja DASH untuk kondisi jaringan yang beragam?

1.3 Batasan Masalah

Penelitian ini dibatasi oleh hal-hal sebagai berikut:

1. Dataset yang didistribusikan hanya dalam bentuk video dan audio, dengan *bitrate* dan jenis atau tipe video yang ditentukan pada saat pengumpulan dataset.
2. Konfigurasi *web server* untuk video *streaming* menggunakan Apache *Web Server* yang dijalankan pada Ubuntu versi 12.04.
3. *Player* yang digunakan sesuai dengan format *codecs* yang terdapat di file MPD yaitu avc1, dengan format video mp4.
4. Pendistribusian file hanya dilakukan dengan menggunakan DASH, dan pada saat proses pengujian akan diberikan suatu alat uji (*bandwidth limiter)* guna mengatur kondisi *bandwidth* di sisi klien.

1.4 Tujuan

Tujuan dari penelitian yang dilakukan ini adalah:

Agar dapat menghasilkan sebuah analisis tentang kinerja video *streaming* berbasis *Dynamic Adaptive Streaming over HTTP* (DASH) pada area jaringan kampus.

1.5 Manfaat

Adapun manfaat yang diperoleh dari hasil penelitian ini adalah sebagai berikut:

1. Agar dapat menambah pemahaman baru tentang teknologi video *streaming* berbasis DASH pada area jaringan kampus (PTIIK-UB).
2. Agar penerapan teknologi video *streaming* berbasis DASH dapat digunakan sebagai referensi lebih lanjut mengenai pengembangan di bidang jaringan.

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini terdiri dari bab-bab yang memuat beberapa sub-bab. Untuk memudahkan pembacaan dan pemahaman, maka skripsi ini dibagi menjadi beberapa bab yaitu:

BAB I PENDAHULUAN

Pada bab I ini akan dijelaskan mengenai latar belakang mengapa mengambil judul "Analisis Kinerja Video *Streaming* Berbasis *Dynamic Adaptive Streaming Over* HTTP (DASH) pada Area Jaringan Kampus". Kemudian terdapat rumusan masalah yang fokus menjelaskan tentang pembuatan lingkungan yang sesuai dengan sistem DASH serta menguji sistem DASH agar dapat menghasilkan suatu kinerja DASH yang akan direpresentasikan dalam bentuk grafik. Selain itu juga terdapat tujuan yang berisikan hal yang nantinya ingin dicapai sesuai dengan rumusan masalah yang telah dituliskan dan judul yang diambil, manfaat yang ditujukan bagi penulis dan masyarakat luas dan sistematika penulisan yang berisi alur penulisan laporan skripsi dari awal hingga akhir.

BAB II TINJAUAN PUSTAKA

Pada bab II ini terdiri dari sub-sub bab mengenai tinjauan pustaka dan dasar teori. Tinjauan pustaka berisi tentang penjelasan singkat mengenai penelitian-penelitian yang sudah dilakukan sebelumnya dan terkait dengan topik yang diambil dalam penelitian ini, diantaranya yaitu "*Dynamic Adaptive Streaming over HTTP in Vehicular Environments*" oleh Christopher Muller, "*Dynamic Adaptive Streaming Over HTTP Dataset*" oleh Stefan Lederer, "*The MPEG-DASH Standard for Multimedia Streaming Over the Internet*" oleh Iraj Sodagar dan "*Studying Streaming Video Quality: From An Application Point of View*" oleh Zhiheng Wang**.** Kemudian pada dasar teori berisi teori-teori penunjang dalam hal penyusunan laporan, serta pembahasan mengenai konsep DASH seperti *adaptive streaming*, HTTP *streaming*, MPD (*Media Presentation Description*), format segmen dan emulasi jaringan beserta *tools* pendukung di dalamnya.

BAB III METODE PENELITIAN

Pada bab III ini terdiri dari analisis kebutuhan, baik kebutuhan perangkat keras dan perangkat lunak agar sistem DASH dapat terwujud dengan baik. Lalu instalasi *web server*, alur proses DASH *player*, implementasi lingkungan berbasis DASH yang berisi bagaimana mewujudkan suatu lingkungan berbasis DASH dari awal hingga akhir, lingkungan pengujian yang di dalamnya membutuhkan dataset dan skenario pengujian, hasil dan pembahasan, serta pengambilan kesimpulan yang didapatkan dari hasil penelitian yang telah dilakukan.

BAB IV IMPLEMENTASI

Pada bab IV ini berisi tentang proses implementasi agar suatu lingkungan berbasis DASH dapat terwujud dengan baik. Diawali dengan membutuhkan kondisi jaringan dengan ketersediaan *bandwidth* yang dinamis (fluktuatif), instalasi *web server*, file MPD dan *player* DASH yang

berbasis *web* (HTML5 DASH) dengan format *codecs* yang telah disesuaikan dengan format *codecs* yang dimiliki oleh file MPD.

BAB V HASIL DAN PEMBAHASAN

Pada bab V ini berisi tentang pengujian, hasil pengujian dan pembahasan. Pengujian dilakukan dengan menggunakan 2 skenario utama, di antaranya pengujian DASH pada kondisi simulasi dan pengujian DASH pada lingkungan jaringan kampus (PTIIK-UB). 2 skenario utama tersebut menghasilkan 8 buah kombinasi skenario pengujian. Setelah hasil pengujian didapatkan, selanjutnya akan diambil suatu hasil pengujian terhadap kinerja video *streaming* berbasis DASH pada area jaringan kampus. Kemudian yang terakhir adalah pembahasan yang berisi rangkuman dari hasil pengujian beserta analisis di dalamnya.

BAB VI PENUTUP

Pada bab VI ini membahas tentang kesimpulan dari hasil dan pembahasan yang telah dilakukan pada bab V, serta saran yang didapatkan dari penelitian yang berguna untuk kehidupan yang akan datang. Kesimpulan berisikan hal-hal yang dapat menjawab semua pertanyaan yang terdapat pada rumusan masalah, dan saran digunakan untuk pengembangan penelitian lebih lanjut sesuai dengan bidang ilmu terkait.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai kajian pustaka dan dasar teori. Kajian pustaka akan membahas penelitian-penelitian yang relevan dan pernah dilakukan sebelumnya. Sedangkan dasar teori dalam penelitian ini meliputi konsep video *streaming*, konsep *Dynamic Adaptive Streaming over HTTP* (DASH) dan emulasi jaringan beserta beberapa aplikasi pendukung (*bandwidth limiter*) yang digunakan.

2.1 Kajian Pustaka

Beberapa penelitian yang relevan tentang konsep *Dynamic Adaptive Streaming over HTTP* (DASH), antara lain adalah penelitian dari Muller (2012), "*An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments*" menjelaskan empat metode (desain) yang digunakan untuk pengiriman media (*multimedia streaming*). Empat metode tersebut diantaranya adalah *Microsoft Smooth Streaming* (MSS), *Adobe HTTP Dynamic Streaming* (ADS), *Apple HTTP Live Streaming* (HLS) dan *Dynamic Adaptive Streaming over HTTP* (DASH). Pada penelitian ini dilakukan percobaan dengan melakukan pengaksesan video menggunakan tiga jalur (*track*), dan video eksperimen yang digunakan adalah BigBuckBunny yang telah ditulis ke dalam 14 ukuran *bitrate* yang berbeda, yaitu 100, 200, 350, 500, 700, 900, 1100, 1300, 1600, 1900, 2300, 2800, 3400 dan 4500 Kb/s. Lalu dilakukan pengukuran terhadap *bitrate* rata-rata, jumlah pergantian kualitas video yang diakses, dan jumlah *unsmooth second metric*. Setelah diuji dengan menggunakan 3 parameter uji dan 4 metode (desain), dihasilkan tabel perbandingan dimana metode DASH dianggap lebih baik daripada ketiga metode yang lainnya. Hasilnya adalah MSS menghasilkan jumlah *switches* dan *bitrate* lebih kecil daripada ADS dan DASH, ADS menghasilkan rata-rata *unsmoothness* terbanyak dan DASH menghasilkan rata-rata *bitrate* terbaik kedua setelah MSS, 0 detik jumlah rata-rata *unsmoothness* dan menghasilkan jumlah rata-rata *switches* terbanyak. Yang dapat penulis ambil dari paper ini adalah keuntungan penggunaan teknologi video *streaming* berbasis

DASH yang diterapkan pada area jaringan kampus yang memiliki keberagaman jaringan (*bandwidth*) di sisi klien.

Penelitian kedua diambil dari paper milik Lederer (2012), "*Dynamic Adaptive Streaming over HTTP Dataset*" yang menjelaskan tentang dataset DASH, dan termasuk di dalamnya terdapat *tools* DASHEncoder yang merupakan sebuah *tools* untuk DASH yang bersifat *free*. Tujuannya karena dataset yang telah dibuat oleh beberapa peneliti baru atau pengembang tidak dapat diakses dan dipublikasikan. Selain itu, paper juga menjelaskan tentang evaluasi dasar dari perbedaan panjang segmen, pengaruh pengaturan *server* HTTP, dan keuntungan dari ukuran segmen yang lebih pendek. Paper ini melakukan pengujian terhadap dataset video BigBuckBunny dengan perbedaan panjang segmen dari 1, 2, 4, 6, 10 dan 15 detik yang masing-masing memiliki 21 representasi (antara 50 Kbps hingga beberapa Mbps), dan diuji menggunakan *web server* dengan koneksi tetap (*persistent*) dan tidak tetap (*non-persistent*). Hasil dari paper ini adalah evaluasi *streaming* yang menggunakan koneksi tetap membutuhkan *bandwidth* ganda untuk mencapai sebuah kinerja yang cukup baik, dan jika *streaming* dengan koneksi tidak tetap, membutuhkan permintaan *bandwidth* yang lebih besar dan rata-rata *bitrate* pada segmen detik kedua sekitar 34% lebih rendah daripada menggunakan koneksi tetap. Ukuran segmen yang lebih besar daripada 6 detik tidak berpengaruh banyak pada penganturan koneksi *web server*, dan ukuran segmen yang optimal untuk pengaturan jaringan seperti yang dilakukan paper ini adalah 2 dan 3 detik dengan menggunakan koneksi yang tetap, serta 5 dan 8 detik dengan menggunakan koneksi yang tidak tetap, dan *bitrate* dari ukuran segmen yang optimal (baik koneksi tetap maupun tidak tetap) adalah sekitar 50 Kbps.

Paper ketiga diambil dari Sodagar (2011), "*The MPEG-DASH Standard for Multimedia Streaming Over the Internet*" yang berisikan penjelasan-penjelasan tentang komponen-komponen yang terdapat di dalam teknologi *Dynamic Adaptive Streaming over HTTP* (DASH). DASH menggunakan protokol HTTP karena hampir semua *firewall* dapat melewati HTTP, dan *streaming* menggunakan HTTP dapat membuat klien mengatur sendiri proses *streaming* yang dilakukan. Di dalam DASH, konten multimedia dikirimkan menggunakan HTTP. Konten yang terdapat pada *server* terdiri dari 2 bagian, yaitu *Media Presentation Description*

(MPD) dan format segmen. MPD digunakan oleh klien untuk memutar konten. MPD berupa dokumen XML yang berisi informasi-informasi penting dari konten seperti URL, alamat dan karakteristik lain. Sedangkan segmen adalah kumpulan dari tindakan HTTP GET pada klien DASH. Konten multimedia dapat diakses sebagai sebuah kumpulan segmen. Selain itu, paper ini juga menjelaskan bahwa masing-masing *adaptive* set dapat menggunakan 1 perlindungan konten (*content-protection descriptor*) untuk mendeskripsikan skema pendukung DRM, karena sebuah *adaptive* set menggunakan beberapa skema perlindungan konten. Dan yang terakhir, paper ini memberikan penjelasan fitur yang terdapat di dalam DASH, diantaranya adalah perpindahan dan pemilihan aliran video, penambahan iklan antara periode atau segmen, pembagian file *manifest* ke dalam beberapa elemen yang dapat diunduh dengan beberapa step, dll.

Paper selanjutnya adalah paper milik Wang (2003), "*Studying Streaming Video Quality: From An Application Point of View*" yang mengulas tentang aspek-aspek penting agar kualitas *streaming* menjadi lebih baik. Peningkatan kualitas tersebut dilakukan dengan proses evaluasi. Paper ini memberikan sekumpulan set metrik pengujian untuk kualitas video yang sudah disesuaikan dengan pengembangan skala besar. Penerapan metrik bertujuan untuk mengukur kualitas video *streaming* di bawah karakteristik jaringan yang beragam. Metrik-metrik tersebut diantaranya adalah rata-rata kualitas layanan *streaming*, jumlah kejadian *delay*, rata-rata waktu *delay* dan jumlah paket yang hilang. Untuk layanan *streaming* yang digunakan adalah *Windows Media Player* (WMP) dan untuk simulasi kondisi jaringan yang beragam, paper ini menggunakan NITSNet, yang merupakan paket emulasi jaringan, dimana *host* berperan sebagai *router* yang dapat mensimulasi kondisi jaringan seperti *bandwidth* dan *delay*. Paper ini memiliki kesamaan dengan penelitian yang dikerjakan, karena sama-sama membahas teknologi video *streaming* ditambah pengujian dan pembahasan yang berguna untuk mengukur kualitas dari video *streaming* yang diterapkan saat itu. Paper ini menjadi dasar peneliti untuk menguji sistem DASH lebih lanjut. Penelitian ini mengambil dua parameter uji yang disebutkan dalam paper, yaitu jumlah kejadian *delay* dan rata-rata waktu *delay*.

2.2 Video *Streaming*

Video *streaming* adalah sebuah teknologi yang memungkinkan sebuah file audio atau video dimainkan, serta pada saat yang sama terjadi pula proses perpindahan data (Hartanto, 2009). Ada dua metode video *streaming* yang paling sering digunakan, yaitu *on-demand* dan *live*. Pada penggunaan *streaming on-demand*, media yang dimainkan akan dialirkan dari file yang telah disimpan sebelumnya. Klien dapat melakukan *fast-forward*, *rewind* ataupun *pause* terhadap media yang sedang dimainkan, baik audio maupun video. Berbeda halnya *streaming* dengan metode *live*, dimana klien tidak dapat melakukan *fast-forward*, *rewind* ataupun *pause*, karena data dikirimkan dari sumber yang melakukan *capture* file secara *real-time*.

Cara kerja video *streaming* secara umum adalah sebagai berikut (Zenhadi, 2013):

Data sumber (audio atau video) akan ditangkap dan disimpan pada sebuah *buffer* yang berada pada memori komputer (bukan *harddisk*), dan kemudian dikompresi sesuai dengan format yang diinginkan. Dalam proses kompresi, *user* dapat mengkompresi data agar ukurannya tidak terlalu besar (bersifat *optional*). Pada aplikasi *streaming* dengan menggunakan jaringan, biasanya data akan dikompresi terlebih dahulu sebelum dilakukan *streaming*, karena terhalang oleh ketersediaan *bandwidth* jaringan yang terbatas. Setelah dikompresi, data akan dialirkan ke pengguna. Pengguna akan melakukan *decode* data dan menampilkan hasilnya ke layar pengguna.

2.2.1 *Adaptive Streaming*

Berdasarkan keberagaman komunikasi jaringan saat ini, adaptifitas merupakan salah satu hal penting yang dibutuhkan untuk klien yang melakukan proses *streaming* (Thang, 2012). *Adaptive streaming* adalah suatu teknologi yang memungkinkan pengiriman media yang memberikan jaminan kepada *user* dengan keberagaman kondisi jaringan saat itu (Miller, 2012). *Adaptive streaming* juga memungkinkan penyesuaian *bitrate* secara dinamis dari berbagai kondisi *stream* jaringan.

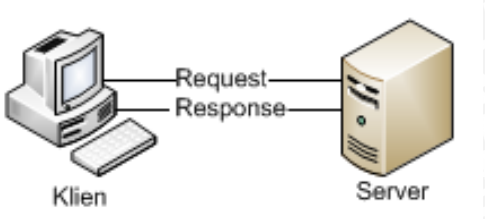
2.2.2 *Video on Demand* (VoD)

Video on Demand (VoD) adalah sebuah istilah penyajian video yang dapat diakses secara *online* melalui jaringan, dimana klien dapat melihat tayangan kapanpun, serta dapat mengulang kembali tayangan yang diinginkan. Video dapat disajikan langsung secara *streaming* ataupun dengan cara diunduh (Kusumandari, 2010). Fungsi VoD adalah untuk mempermudah klien dalam mengakses dan memilih video mana yang ingin diputar (Basuyoga, 2011).

Terdapat perbedaan yang mendasar antara VoD *streaming* dengan *live streaming*, yaitu ketersediaan waktu untuk segmen yang akan diputar. Pada *live streaming*, waktu antara 2 permintaan sesuai dengan waktu pertama kali permintaan dilakukan. Jadi, jika segmen memiliki durasi yang sama (t detik), maka jarak antara masing-masing permintaan harus mempunyai durasi yang sama pula (t detik). Sedangkan pada VoD *streaming*, permintaan dapat langsung mengisi penuh pada *buffer*. Inisialisasi *buffer* dibutuhkan pada awal sebelum memulai pemutaran konten multimedia lebih lanjut, sehingga klien dapat memutuskan berapa jumlah *bitrate* yang akan diterimanya (Basuyoga, 2011).

2.2.3 Protokol HTTP

HTTP merupakan sebuah protokol yang menjembatani file *website* menuju *World Wide Web* (WWW) dengan *user*. HTTP menangani hampir semua *website* yang tersedia dalam Internet seperti gambar dan file html, php dan xhtml (Michalos, 2012). HTTP memiliki 2 pesan diantaranya adalah *reponse* dan *request*. Untuk memahami cara kerja HTTP dapat dilihat pada **Gambar 2.1** berikut. Klien akan mengirimkan pesan *request* kepada *server*. Pesan *request* tersebut berisi segala permintaan yang akan diselesaikan oleh *server*. Setelah pesan *request* diterima oleh *server*, *server* akan menjawabnya dengan mengirimkan kembali pesan *response* yang akan diterima oleh klien.



Gambar 2.1 Cara Kerja HTTP
Sumber: (Michalos, 2012)

Di dalam HTTP terdapat hal-hal yang menyebabkan DASH muncul sampai saat ini, diantaranya adalah (Michalos, 2012):

a. *HTTP Downloading* merupakan cara paling sederhana bagi *user* yang ingin memutar video dengan cara pemilihan video oleh *user,* lalu mengunduhnya dan kemudian melihatnya. Meskipun terlihat sederhana, tetapi di balik ini semua terdapat sebuah kerugian, karena pertama-tama *user* harus mengunduh keseluruhan video baru dapat menikmatinya.
b. *HTTP Progressive Downloading* dikenalkan untuk menutupi ketidaknyamanan yang disebabkan oleh *HTTP downloading*. *HTTP Progressive Downloading* mengijinkan *user* untuk mengunduh konten multimedia ketika file lainnya sudah dapat dinikmati atau secara keseluruhan telah diputar. Hal ini yang membuat *HTTP Progressive downloading* membutuhkan koneksi Internet yang super cepat untuk mengunduh konten atau hanya memutar konten (*playback*) berulang-ulang.
c. *HTTP Pseudo-streaming* dapat mengatasi koneksi klien yang beragam seperti kapasitas *hardware* (CPU, memori RAM, kompatibel HD, dll), dan dapat mengirimkan kualitas video terbaik yang dimilikinya berdasarkan faktor keberagaman kondisi klien tersebut.

Keuntungan HTTP diantaranya adalah (Michalos, 2012):

a. HTTP adalah *stateless protocol* dimana dapat mengatur *user* yang hendak memulai koneksi untuk menonton video hingga melakukan penutupan koneksi ketika *user* selesai mengakses video, tanpa menyimpan informasi apapun ke dalam *server*.

b. HTTP adalah sebuah protokol yang secara luas dapat digunakan oleh klien hampir di seluruh dunia, serta memberikan kontrol penuh terhadap prosedur *streaming*.
c. HTTP juga dapat menangani berbagai *firewall* yang datang (Sodagar, 2011).
d. HTTP termasuk protokol yang handal dan dapat dipercaya (Stockhammer, 2011).
e. Pengiriman menggunakan HTTP memberikan kemampuan kepada klien untuk memilih secara otomatis *content rate* yang sesuai dengan ketersedian *bandwidth* saat itu, tanpa membutuhkan negosiasi terlebih dahulu terhadap *server streaming* (Stockhammer, 2011).
f. Pengiriman menggunakan HTTP dapat merubah *content rate* secara halus ketika ada perubahan terhadap ketersediaan *bandwidth* (Stockhammer, 2011).

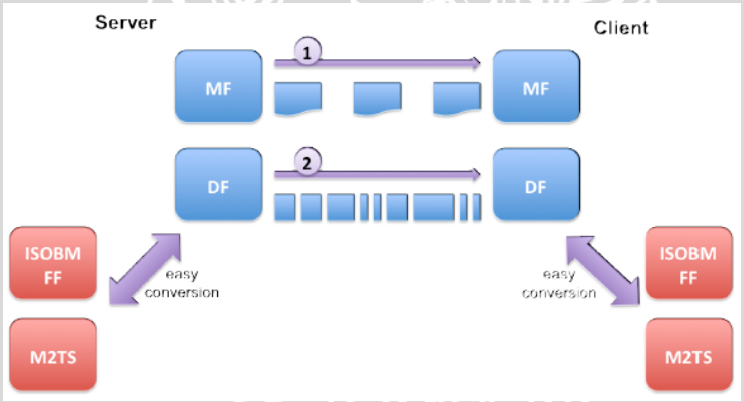
Pada HTTP *streaming*, *web server* hanya mengetahui sedikit tentang kondisi jaringan klien saat itu, dan hal ini yang membuat klien untuk membuat keputusan tentang pengiriman konten agar dapat memenuhi *Quality of Service* (QoS) (Thang, 2012). Ketika menggunakan *streaming* konvensional, pensinyalan metadata dan media biasanya menggunakan protokol yang berbeda (RTSP dan RTP). Sedangkan untuk HTTP *streaming*, pensinyalan metadata dan media dilakukan dengan satu protokol yang sama yaitu HTTP. HTTP *streaming* dapat diaplikasikan baik untuk *on-demand streaming* maupun *live streaming* (Thang, 2012).

2.2.4 *Dynamic Adaptive Streaming over HTTP* (DASH)

MPEG-DASH merupakan suatu standar yang dirancang untuk pengiriman media melalui infrastuktur yang sudah ada sebelumnya, dan dapat mengatasi berbagai kondisi jaringan (*bandwidth*) selama proses *streaming* berlangsung (Muller, 2012). DASH dapat digunakan atau diterapkan pada *browser* karena tidak membutuhkan *plugin* apapun, sedangkan disisi lain banyak *platform* komersial yang cenderung menutup sistemnya dengan format *manifest* khusus,

format konten dan protokol *streaming* (Stockhammer, 2011). Dan DASH tidak membatasi klien dengan *server* yang berasal dari beragam vendor.

Dalam DASH, video dikompresi ke dalam beberapa versi, dengan masing-masing versinya mempunyai *bitrate* dan level kualitas video yang berbeda (Kurose, 2001). Klien dapat melakukan permintaan terhadap konten yang telah disediakan di dalam web server. Ketika jumlah *bandwidth* yang tersedia tinggi, maka klien dapat secara otomatis memilih potongan video dari versi yang tinggi, dan sebaliknya jika *bandwidth* yang tersedia rendah, maka klien akan memilih video dari versi yang rendah pula. Klien memilih potongan video yang berbeda pada waktu yang bersamaan dengan menggunakan pesan *request* HTTP GET. DASH memperkenalkan beberapa representasi untuk masing-masing kontennya, dan inilah yang mengurangi efisiensi dari *cache* serta meningkatkan lalu lintas pada jaringan (Sanchez, 2011). Arsitektur sistem DASH dapat diilustrasikan seperti **Gambar 2.2** di bawah ini.



Gambar 2.2 Arsitektur DASH

Sumber: (Mueller, 2011)

Pada **Gambar 2.2**, file *manifest* (MF) dapat diibaratkan seperti *playlist*, MPD, atau apapun yang terdapat berdasarkan pada dokumen XML, sedangkan *delivery format* (DF) adalah ekstensi file dari ISOBMFF dan M2TS. Step 1 gambar di atas menjelaskan bahwa pada saat permintaan dilakukan oleh klien, file *manifest* diproses secara langsung oleh *server* dan diberikan ke klien untuk inisisasi *session*. Sedangkan step 2 menjelaskan bahwa file *manifest* akan dibagi-

bagi dan dilakukan sebuah permintaan untuk *single* segmen menggunakan HTTP yang didasari pada informasi dalam file *manifest.*

Di dalam file *manifest* terdapat file MPD yang berisi urutan dari 1 atau lebih urutan periode *non-overlapping* untuk 1 atau lebih representasi yang telah disediakan. Representasi tunggal menjelaskan sebuah media yang spesifik sesuai dengan karakteristik tertentu, seperti *bitrate, framerate*, resolusi, dsb. Masing-masing representasi berisi 1 atau lebih segmen yang dideskripsikan pada media/metadata untuk dipecah dan ditampilkan pada konten media (Muller, 2011).

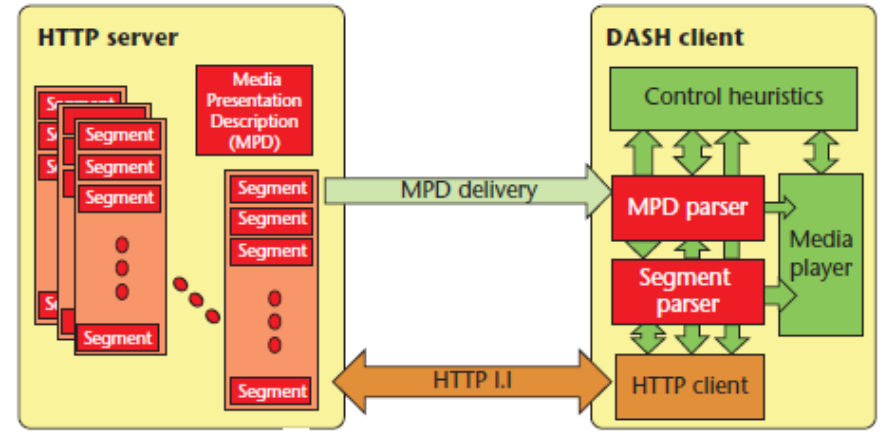
DASH mengijinkan klien dengan beragam akses Internet untuk memainkan video dengan berbagai *encoding* yang berbeda. Klien dengan kecepatan yang rendah dapat menerima versi video dengan *bitrate* yang rendah, dan klien dengan menggunakan koneksi *fiber* akan mendapatkan versi video dengan *bitrate* tinggi. Disisi lain, DASH mengijinkan kliennya untuk beradaptasi sesuai dengan *bandwidth* yang tersedia jika *bandwidth* selalu berubah-ubah selama *streaming* berlangsung.

Masing-masing versi video yang terdapat di dalam DASH disimpan dalam sebuah *web server* HTTP yang memiliki URL yang unik dan sebuah file *manifest* (MPD) yang mendeskripsikan URL untuk masing-masing versi dan *bitrate*. Selain itu, *web server* HTTP tidak hanya menyimpan banyaknya versi video, melainkan juga menyimpan banyak versi dari audio. Masing-masing versi audio memiliki level kualitas tersendiri dan *bitrate* khusus.

Cara kerja DASH adalah sebagai berikut (Kurose, 2001):

Pertama-tama klien melakukan permintaan terhadap file *manifest* (atau yang biasa disebut dengan MPD) dan mempelajari beragam versi yang telah disediakan di dalam *web server*. Kemudian klien memilih satu potongan video dengan menentukan sebuah URL dan *byterange* pada setiap permintaan HTTP GET. Hal ini dilakukan untuk tiap-tiap potongan video yang tersedia di dalam *web server*. Saat mengunduh potongan video, klien juga memastikan *bandwidth* yang diterima agar dapat memilih potongan video yang diminta selanjutnya. Jika klien memiliki banyak *buffer* video dan jika dipastikan *bandwidth* yang diterima adalah tinggi, maka akan memilih potongan video dari versi yang tinggi, begitu juga sebaliknya.

Lingkup pada MPEG-DASH dapat dilihat pada **Gambar 2.3** berikut.



Gambar 2.3 Lingkup MPEG DASH

Sumber: (Sodagar, 2011)

Pada **Gambar 2.3** dijelaskan bahwa pertama-tama konten multimedia ditangkap dan disimpan pada sebuah *web server* HTTP dan dikirimkan ke klien dengan menggunakan HTTP pula. Konten di dalam *server* HTTP berisi 2 bagian, yaitu *Media Presentation Description (MPD)* dan segmen. MPD mendeskripsikan file *manifest* dari konten yang sudah tersedia di dalam *web server*, keberagaman alternatif yang dapat dipilih oleh klien, alamat (URL) dan karakteristik-karakteristik yang lainnya. Sedangkan segmen berupa *chunks* dalam satu atau beberapa file, yang berisikan *bitstream* multimedia yang sebenarnya (Sodagar, 2011).

Di sisi klien, konten DASH diperlukan agar klien dapat memutarnya dengan baik. Pengiriman konten oleh klien dapat dilakukan dengan menggunakan HTTP, *email*, *thumb drive*, *broadcast* ataupun dengan cara yang lainnya. Disini file MPD dipecah-pecah dengan tujuan agar klien dapat mempelajari tentang informasi di dalamnya, seperti durasi, ketersediaan konten media, tipe, resolusi, *bandwidth* minimum dan maksimum serta keberagaman alternatif *encoding* yang tersedia, serta karakteristik yang lain. Dengan informasi karakteristik ini, klien DASH dapat memilih alternatif *encoding* yang paling sesuai, serta dapat memulai proses pengiriman konten DASH dengan mengambil segmen menggunakan permintaan

HTTP GET (Sodagar, 2011). Klien dengan *server* akan terus berkomunikasi mengenai representasi video apa yang akan dikirimkan ke klien, karena melihat kondisi jaringan (*bandwidth*) saat itu yang selalu tidak tetap (fluktuatif). Di sisi klien sudah ditanamkan sebuah file *Javascriot* yang bertujuan untuk melakukan proses pembacaan *bandwidth* jaringan saat itu. Penanaman file *Javascript* ini terdapat pada bagian *control heuristics* di sisi klien.

Setelah mengetahui variasi kondisi pada jaringan, klien akan terus melakukan pemilihan segmen selanjutnya dan juga memantau pergerakan *bandwidth* pada jaringan saat itu. Maka dari itu, klien dapat memutuskan bagaimana beradaptasi dengan *bandwidth* klien yang tersedia saat itu dengan cara tetap melakukan pemilihan sub-sub segmen dengan *bitrate* yang tinggi maupun rendah. Spesifikasi MPEG-DASH hanya berisi MPD dan format segmen saja. Selebihnya seperti pengiriman MPD, bagaimana *encoding* format segmen, bagaimana kinerja klien dalam melakukan pemilihan segmen serta memainkan kontennya, semuanya termasuk diluar kendali lingkup MPEG-DASH (Sodagar, 2011).

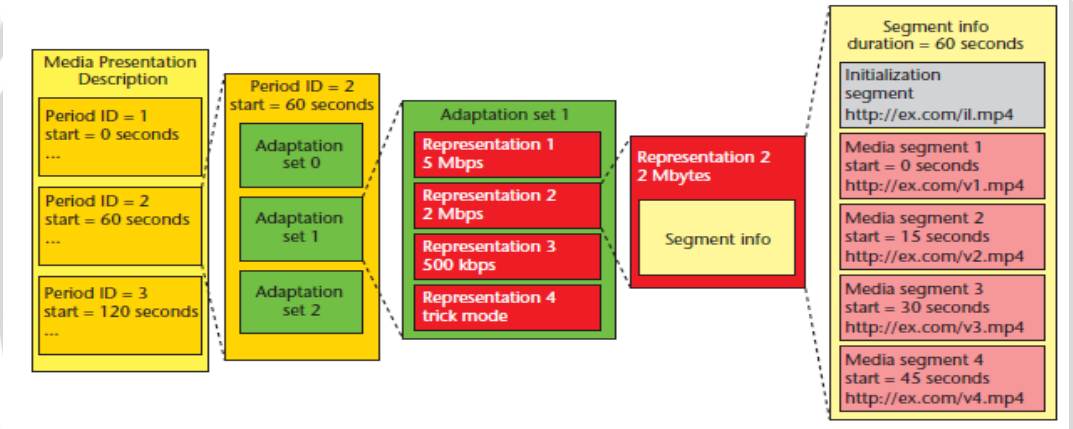
Web server bekerja dengan mengatur permintaan tepat pada waktunya, pemutaran konten dengan baik (*smoothly*) serta menyesuaikan *bitrate* atau atribut yang lainnya. Hal ini dilakukan agar klien dapat beradaptasi dengan perubahan, baik yang disebabkan oleh perangkat yang digunakan oleh klien, kondisi jaringan klien saat itu maupun keinginan dari klien sendiri (Stockhammer, 2011).

Keuntungan menggunakan sistem layanan *streaming* berbasis DASH adalah (Sodagar, 2011):

a. Dapat melakukan perubahan dan pemilihan aliran *streaming*.
b. Dapat menyisipkan iklan, baik untuk layanan *streaming on-demand* maupun *live streaming*.
c. Standar DASH memiliki kumpulan kualitas metrik agar klien dapat melaporkan kembali ke *server* tentang kegiatan *streaming* menggunakan DASH yang telah dilakukan, terutama terkait kondisi jaringan klien saat itu.
d. dll.

2.2.4.1 *Media Presentation Description* (MPD)

Streaming HTTP yang adaptif membutuhkan beragam alternatif *bitrate* dari konten multimedia untuk disediakan di dalam *web server*. Konten multimedia tersebut berisi beberapa komponen media seperti audio, video ataupun teks yang masing-masing memiliki karakteristik yang berbeda. Karakteristik ini dalam MPEG-DASH disebut dengan *Media Presentation Description* (MPD). MPD adalah sebuah dokumen dengan format XML, dan menyediakan metadata untuk mengidentifikasi komponen video *streaming* serta membangun bagian dan urutannya (Sodagar, 2011). Contoh karakteristik yang terdapat di dalam file MPD terlihat pada **Gambar 2.4** berikut.



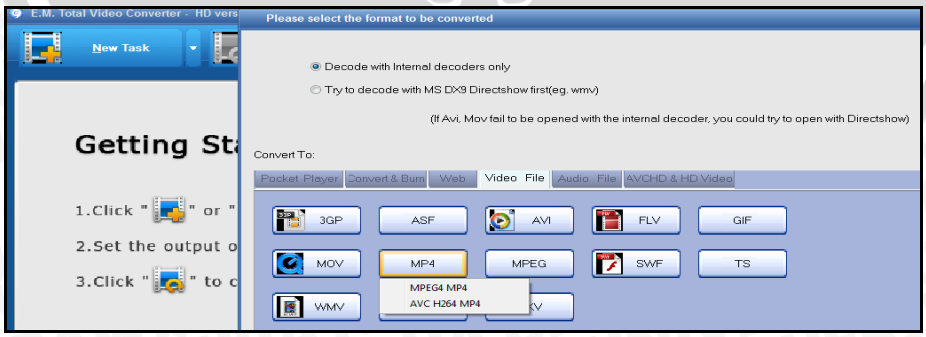
Gambar 2.4 Struktur MPD
Sumber: (Sodagar, 2011)

Pada **Gambar 2.4** dijelaskan bahwa sebuah MPD berisi 1 atau lebih periode, dimana 1 periode merupakan sebuah interval program sepanjang *temporal axis.* Di dalam periode berisi waktu mulai, durasi dan 1 atau lebih *adaptation set*. *Adaptation set* memberikan informasi tentang komponen media dan alternatif *encode*, serta berisi *bitrate* yang berbeda-beda untuk konten multimedia yang sama. *Adaptation set* juga berisi banyak representasi. Representasi adalah sebuah alternatif *encode* dari komponen media yang sama, bermacam-macam representasi *bitrate*, resolusi, jumlah *channel* atau karakteristik yang lainnya (Sodagar, 2011).

Hal-hal di dalam suatu sistem DASH yang dapat membuat DASH dikatakan sebagai suatu sistem layanan video *streaming* yang dapat secara dinamis beradaptasi dengan lingkungan di sekitarnya adalah dengan keberadaan karakteristik-karakteristik di dalam file MPD ini. Keberadaaan file MPD ini juga merupakan perbedaan yang amat mendasar jika dibandingkan dengan layanan *streaming* biasa (tanpa DASH). Klien DASH hanya mengakses URL yang telah disediakan oleh *web server*, bukan file MPD. Dan secara otomatis klien akan menerima representasi video yang sesuai dengan kondisinya (*bandwidth*) saat itu. Jadi, representasi video dan audio merupakan salah satu hal yang cukup penting yang harus ada di dalam DASH, yang dapat membuat sistem DASH dapat adaptif terhadap kondisi jaringan.

Tools untuk membuat file MPD sudah banyak ditemukan di Internet, tetapi tidak semuanya dapat digunakan dan harus membutuhkan *player* yang sesuai pula. Pada bab ini akan dijelaskan mengenai tahapan-tahapan untuk menghasilkan sebuah file MPD menggunakan aplikasi MP4Box dari GPAC (Concolato, 2014). Sebelum pembuatan file MPD, perlu dilakukan tahapan seperti konversi video dan pemecahan file video dari audionya.

Tahapan pertama yaitu konversi video dengan menggunakan aplikasi Total Video Converter. Aplikasi ini dapat dijalankan pada sistem operasi Windows. Proses konversi video bertujuan agar video utama dengan kualitas HD tersebut dapat diubah ke sub-sub video yang memiliki kualitas, *bitrate*, resolusi dan format yang sesuai dengan format yang diinginkan.



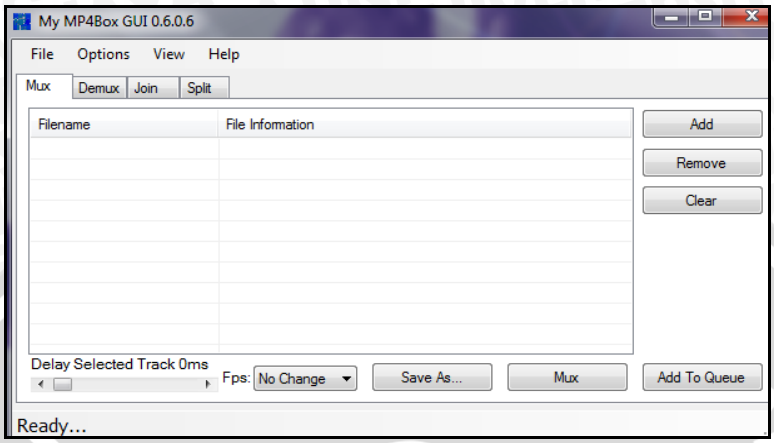
Gambar 2.5 Tampilan Aplikasi Total Video Converter

Gambar 2.5 di atas merupakan tampilan menu dari aplikasi Total Video Converter. Percobaan kali ini menggunakan aplikasi Total Video Converter yang membutuhkan sebuah file video dengan format mp4. Sebenarnya DASH mendukung format-format lain selain mp4, seperti webm. Namun, dibutuhkan kesesuaian format *codecs* yang digunakan untuk file MPD dan *player*. Melalui aplikasi Total Video Converter, video asli dapat diubah menjadi beragam tipe, salah satunya pada perobaan kali ini file video diubah menjadi tipe AVC H264 MP4.

Proses konversi di atas akan menghasilkan file dengan format ekstensi .h264. Setelah proses konversi berhasil, tahapan selanjutnya adalah pemisahan antara file video dengan audionya. Pemisahan ini dilakukan dengan menggunakan aplikasi *muxer*. Dalam penelitian kali ini, aplikasi *muxer* yang digunakan adalah My MP4Box GUI. Hal ini bertujuan untuk menghasilkan file *samplemuxer* yang nantinya akan digunakan untuk membuat file MPD. Di dalam aplikasi My MP4Box GUI terdapat menu *demux* dan *mux*.

Pertama-tama file video yang lengkap dengan audionya dimasukkan ke dalam menu *demux* yang berguna untuk memisahkan/memecah kesatuan file (video dan audio) menjadi file video dan file audio yang terpisah satu sama lainnya. Setelah dipecah menggunakan menu *demux*, maka dihasilkan suatu file video tanpa file audio di dalamnya. Kemudian file tersebut dimasukkan ke dalam menu *mux* yang bertujuan untuk menyisipkan kembali informasi ke dalam file multimedia tersebut. Kedua proses ini sangatlah penting dalam membuat file MPD, karena file video tanpa melalui menu *mux* terlebih dahulu tidak dapat diproses lebih lanjut menjadi sebuah file MPD. Dan pada akhirnya, file yang digunakan untuk membuat MPD adalah file hasil *mux*, karena DASH hanya mendukung pemutaran video yang *singleplexing*. Hal ini telah dibuktikan ketika file yang digunakan untuk pembuatan file MPD bukan file *singleplexing*, maka pembuatan file MPD tidak dapat berjalan sukses (*error*).

Berikut **Gambar 2.6** di bawah ini merupakan tampilan aplikasi My MP4Box GUI yang dijalankan pada sistem operasi Windows.



Gambar 2.6 Tampilan Aplikasi My MP4Box GUI

Pada **Gambar 2.6** dijelaskan bahwa proses pemisahan file video dengan audionya digunakan suatu aplikasi yang berjalan di sistem operasi Windows yang bernama My MP4Box GUI. Aplikasi ini dapat memproses file dengan format h264 (hasil dari konversi menggunakan Total Video Converter). Setelah proses ini berhasil dilakukan, tahapan yang terakhir adalah pembuatan file MPD. Cara untuk pembuatan file MPD sudah banyak ditemukan di Internet, diantaranya adalah DASHEncoder (Lederer, 2012), *Web*M DASH (Galligan, 2014) dan MP4Box GPAC (Lederer, 2013). Tetapi tidak semua *tools* pembuatan MPD dapat digunakan dan sukses diaplikasikan pada *player*.

Pembuatan file MPD kali ini menggunakan *tools* MP4Box dari GPAC (Lederer, 2013). Aplikasi MP4Box mendukung video dengan format mp4 dan *codecs* avc1. Aplikasi MP4Box dari GPAC dapat dijalankan oleh banyak sistem operasi, dan untuk penelitian kali ini digunakan di sistem operasi Ubuntu 12.04. Contoh *script* pembuatan MPD dapat dilihat pada **Gambar 2.7**.

```
putranti@putranti-labjar:~/gpac$ sudo MP4Box -dash 10000
high.mp4#trackID=3:role=video
normal.mp4#trackID=2:role=video
```

```
low.mp4#trackID=1:role=video
audio.mp4#trackID=4:role=audio
```

Gambar 2.7 *Script* Pembuatan MPD

Pada **Gambar 2.7**, dijelaskan bahwa terdapat 4 *track*, dimana 3 *track* dengan *role* video dan 1 track dengan *role* audio. 3 *track* video yang digunakan yaitu `high.mp4`, `normal.mp4` dan `low.mp4`. Masing-masing video tersebut memiliki kualitas yang berbeda satu dengan yang lainnya. Kualitas tertinggi dimiliki oleh video `high.mp4`, lalu kualitas kedua dimiliki oleh video `normal.mp4`, dan kualitas paling rendah dimiliki oleh video `low.mp4`.

Script di atas dijalankan guna menghasilkan file MPD yang nantinya akan digunakan untuk layanan video *streaming* berbasis DASH. Contoh potongan isi file MPD dapat dilihat pada **Gambar 2.8**.

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.1-DEV-
rev5261  on 2014-06-03T07:35:14Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011"
minBufferTime="PT1.500000S" type="static"
mediaPresentationDuration="PT0H4M5.30S"
profiles="urn:mpeg:dash:profile:full:2011">
 <ProgramInformation
moreInformationURL="http://gpac.sourceforge.net">
  <Title>titanium_track3_dash.mpd generated by
GPAC</Title>
 </ProgramInformation>
 <Period id="" duration="PT0H4M5.30S">
  <AdaptationSet segmentAlignment="true"
bitstreamSwitching="true" maxWidth="854" maxHeight="480"
maxFrameRate="25" par="854:480" lang="und">
   <Role schemeIdUri="urn:gpac:dash:role:2013"
value="video"/>
   <SegmentList>
```

```
    <Initialization sourceURL="high.mp4"/>
   </SegmentList>
   <Representation id="1" mimeType="video/mp4"
codecs="avc3.42c033" width="854" height="480"
frameRate="25" sar="1:1" startWithSAP="0"
bandwidth="622064">
    <BaseURL>video/high.mp4</BaseURL>
    <SegmentList timescale="25000" duration="250000">
     <SegmentURL mediaRange="964-366845" indexRange="964-
1008"/>
     <SegmentURL mediaRange="366846-744132"
indexRange="366846-366890"/>
      ......
  </AdaptationSet>
 </Period>
</MPD>
```

Gambar 2.8 Isi File MPD

Pada **Gambar 2.8** menunjukkan potongan isi file MPD setelah berhasil dibuat. File MPD dibuat dengan menggunakan *tools* dari GPAC dengan versi 0.5.1. File MPD tersebut dibuat pada tanggal 3 Juni 2014 pukul 07.35. *MediaPresentationDuration* menjelaskan durasi video yang akan dijalankan. *minBufferTime* menjelaskan jumlah durasi waktu inisialisasi sebelum memulai presentasi. Besaran untuk *buffer* pada tiap MPD adalah sama yaitu 1,5 detik. *Base* URL menunjukkan alamat direktori tempat disimpannya file video dan audio. *Segmentbase index range* menunjukkan panjang video yang akan diputar. *ContentComponent* terdiri dari *ContentType* yang menunjukkan tipe konten baik untuk konten video maupun audio. Format (*mimeType*) video dan audio yang digunakan adalah mp4, dengan jenis *codecs* untuk file video yaitu avc1 dan *codecs* untuk audio adalah mp4a.

File MPD bersifat statik, jika ingin merubahnya harus mengulang tahapan dengan mengetikkan *script* seperti yang telah dijelaskan pada **Gambar 2.7**. Keseluruhan isi file MPD seperti telah dijelaskan pada **subbab 2.2.4.1**. Setelah

file MPD sukses dibuat, *player* pun disiapkan. *Player* ini bertujuan agar klien dapat memutar konten DASH dengan baik. Dalam DASH, banyak *player* yang disediakan, tetapi tidak semua *player* dapat memutar MPD dengan baik.

Terdapat beberapa aplikasi yang dapat digunakan sebagai *player* DASH, antara lain VLC (Lederer, 2012), libDASH (Mueller, 2013) dan MP4Client (Concolato, 2014). Beragam *tools* tersebut dapat digunakan untuk memutar konten DASH, asalkan format *codecs* dari *player* dan MPD harus sama (sesuai). *Player* yang digunakan dalam pecobaan kali ini adalah MP4Client. *Player* ini berbasis *dekstop* dan tidak akan mendapatkan pengaruh apapun ketika dilakukan suatu pembatasan *bandwidth*. *Player* ini dipilih karena MP4Client merupakan aplikasi bawaan dari MP4Box (GPAC) yang digunakan untuk menjalankan video (file MPD) hasil *generate* dari MP4Box. MP4Client adalah *player* yang dikeluarkan oleh GPAC yang dijalankan dengan menggunakan perintah *command line*.

Aplikasi MP4Client dapat diakses oleh klien dengan sistem operasi Windows menggunakan cmd (*command prompt*), sedangkan klien dengan sistem operasi Ubuntu menggunakan terminal.

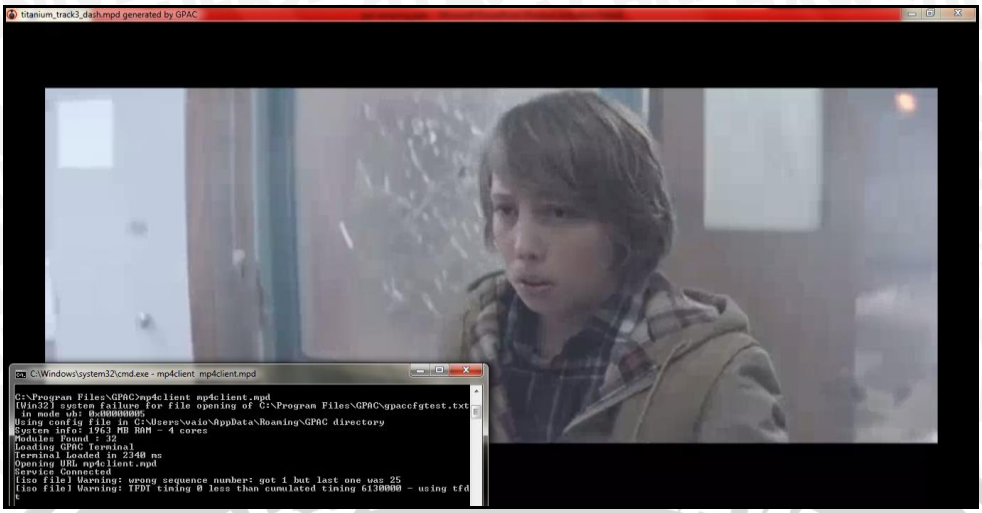
Klien dengan sistem operasi Windows :

```
C:\Program Files\GPAC>mp4client mp4client.mpd
```

Klien dengan sistem operasi Ubuntu :

```
putranti@putranti-labjar:~/gpac$ mp4client mp4client.mpd
```

Berikut pada **Gambar 2.9** merupakan tampilan *command prompt* pada sistem operasi Windows pada saat mengakses aplikasi MP4Client. Nama MPD yang diputar adalah `mp4client.mpd`.



Gambar 2.9 Tampilan Aplikasi MP4Client (Windows)

Gambar 2.9 di atas menunjukkan bahwa file MPD yang dibuat dengan menggunakan aplikasi MP4Box dari GPAC dapat berjalan dengan baik dalam sistem operasi Windows. Tampilan di atas menunjukkan proses menjalankan konten DASH (`mp4client.mpd`) dari awal hingga akhir dengan menggunakan *player* berbasis *dekstop*, yaitu MP4Client.

2.2.4.2 Format Segmen

Konten multimedia dapat diakses sebagai suatu kumpulan dari segmen. Segmen dapat didefinisikan sebagai satu kesatuan jawaban HTTP GET dari klien DASH atau sebagian dari HTTP GET. Komponen media dikonversi ke dalam beberapa segmen. Segmen pertama dapat berupa sebuah segmen inisialisasi yang berisi informasi yang dibutuhkan untuk inisialisasi media *decoder* klien DASH (tidak mencakup data-data atau media yang aktual).

Aliran media kemudian dibagi menjadi satu atau lebih media segmen yang berurutan. Tiap-tiap segmen media ditandai dengan sebuah URL yang unik, indeks, waktu mulai serta durasi baik secara implisit maupun eksplisit. Masing-masing segmen media berisi sedikitnya 1 akses *stream*, yang merupakan sebuah akses random atau *switch-to-point* pada *stream* media, dimana *decoding* dapat dimulai hanya dengan menggunakan data dari *point forward*.

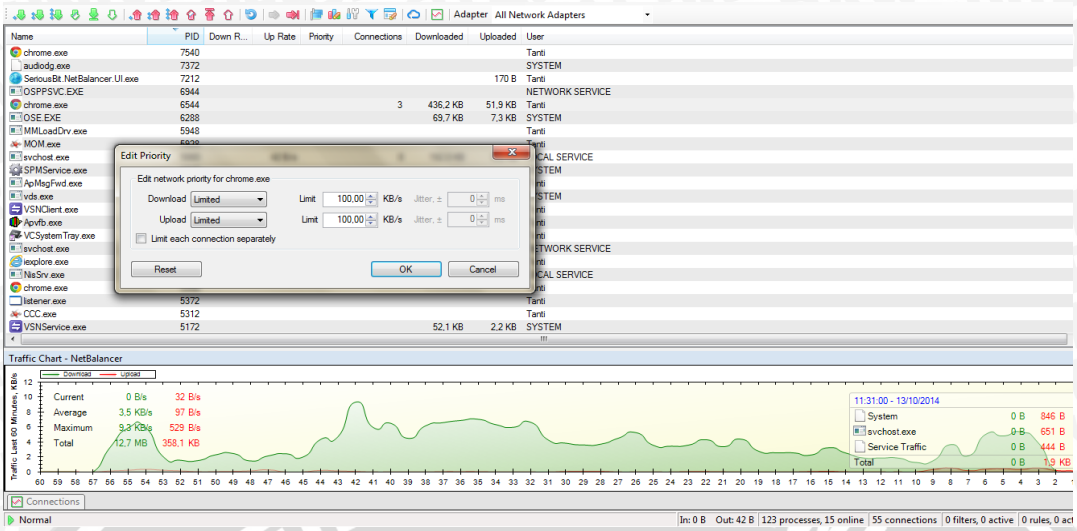
Pengunduhan segmen pada beberapa bagian dilakukan dengan suatu metode pensinyalan subsegmen menggunakan sebuah kotak indeks segmen. Kotak ini menjelaskan subsegmen dan aliran *access point* dalam segmen, dengan pensinyalan berdasarkan durasi dan pengimbang *byte* (*byte offsets*). Klien DASH dapat menggunakan informasi pada indeks untuk meminta subsegmen menggunakan perintah HTTP GET. Informasi indeks dari sebuah segmen dapat diletakkan pada sebuah *single box* pada awal segmen tersebut, atau disebarkan pada *box-box* indeks segmen.

2.3 Emulasi Jaringan

Emulasi jaringan merupakan sebuah teknik yang dilakukan pada suatu lingkungan agar dapat memenuhi kondisi jaringan yang diinginkan. Dalam hal ini emulasi jaringan dibutuhkan untuk melihat bagaimana kinerja dari sistem yang sudah diterapkan, serta memperkirakan dampak sesungguhnya dari perubahan-perubahan yang dilakukan oleh jaringan saat itu (Wardana, 2014). Pada sebuah penelitian yang tidak dapat dilakukan secara langsung di lapangan, maka emulasi jaringan sangatlah dibutuhkan untuk mendapatkan sifat-sifat jaringan yang dapat mewakili kondisi jaringan yang sebenarnya (Wardana, 2014). NetBalancer dan Trickle merupakan 2 dari banyak aplikasi emulasi jaringan yang dapat digunakan untuk mengemulasikan suatu kondisi jaringan tertentu.

2.3.1 NetBalancer

NetBalancer adalah suatu alat pemantauan trafik jaringan yang sengaja dirancang untuk sistem operasi Windows. NetBalancer dapat mengontrol dan memantau lalu lintas Internet, dan dapat mengatur rata-rata kecepatan transfer untuk proses unduh maupun unggah untuk masing-masing aplikasi yang sedang digunakan (Tom, 2014). NetBalancer memiliki 3 prioritas dalam hal pembatasan proses unduh dan unggah, diantaranya adalah prioritas tinggi, prioritas normal dan prioritas rendah, serta fitur lainnya yaitu memblok lalu lintas, mengabaikan lalu lintas dan membatasi lalu lintas. Contoh tampilan aplikasi NetBalancer dapat dilihat pada **Gambar 2.10**.

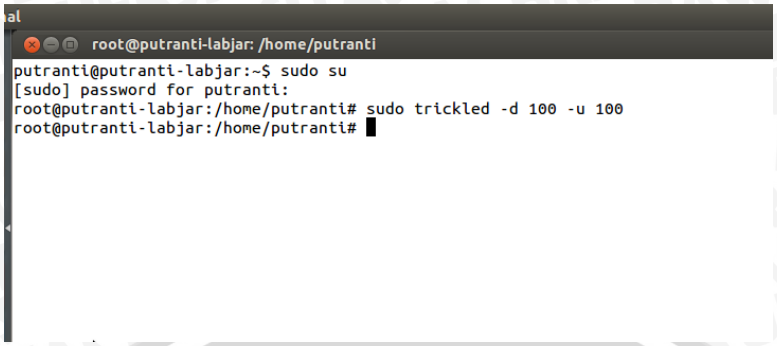


Gambar 2.10 Aplikasi *Bandwidth Limiter* NetBalancer

Pada **Gambar 2.10** dijelaskan tentang cara untuk melakukan pembatasan *bandwidth* untuk aplikasi NetBalancer. Pembatasan *bandwidth* terbagi menjadi 2 macam, yaitu pembatasan *bandwidth* untuk *upload* dan pembatasan *bandwidth* untuk *download.* Satuan yang digunakan oleh aplikasi NetBalancer ini adalah KB/s. Pada aplikasi NetBalancer juga disediakan pilihan untuk membatasi masing-masing koneksi secara terpisah. Pada penelitian ini, pembatasan *bandwidth* untuk proses *download* lebih diutamakan dibandingkan proses *upload.* Di dalam aplikasi NetBalancer juga terdapat NetTray yang digunakan untuk memantau ketersediaan *bandwidth* jaringan saat itu, untuk masing-masing koneksi yang sedang dijalankan.

2.3.2 Trickle

Trickle adalah suatu alat yang dapat menghilangkan lonjakan jaringan yang besar dengan cara melakukan proses pelimitan pada koneksi TCP (Ghobadi, 2011). Trickle membagi aliran video dengan cara menempatkan batas atas pada TCP. Sama halnya dengan aplikasi NetBalancer, pada Trickle dapat dilakukan pemilihan prioritas kecepatan proses unduh dan unggah untuk beberapa aplikasi yang sedang dijalankan. *Script* trickle dapat dilihat pada **Gambar 2.11** di bawah ini.



Gambar 2.11 *Script* Trickle

Pada **Gambar 2.11** di atas, terdapat simbol –d dan –u, dimana –d berarti besaran nilai *download* dan –u adalah besaran nilai *upload*. Besaran nilai *bandwidth* yang dituliskan ke dalam aplikasi trickle adalah dalam satuan Kbps. Pada gambar dapat dilihat bahwa besaran nilai yang dimasukkan sebagai pembatas untuk *bandwidth download* dan *upload* adalah sebesar 100 Kbps.

BAB III
METODE PENELITIAN

Pada bab metode penelitian dijelaskan langkah-langkah yang akan dilakukan pada penelitian ini, yakni sebagai berikut:

1. Analisis Kebutuhan
2. Instalasi *Web Server*
3. Rancangan Analisis Kinerja DASH
4. Implementasi Lingkungan Berbasis DASH
5. Lingkungan Pengujian
6. Hasil dan Pembahasan
7. Pengambilan Kesimpulan

3.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menentukan semua kebutuhan yang diperlukan dalam rangka menerapkan serta melakukan sebuah pengujian terhadap layanan video *streaming* berbasis DASH di area jaringan kampus (PTIIK). Analisis kebutuhan berisikan penjelasan tentang hal-hal yang dibutuhkan agar lingkungan berbasis DASH dapat terwujud, serta perangkat keras dan perangkat lunak yang dibutuhkan selama proses penelitian berlangsung. Adapun hal-hal yang dibutuhkan oleh DASH diantaranya adalah sebagai berikut:

1. Dataset

   Hal paling utama yang dibutuhkan dalam proses video *streaming* berbasis DASH adalah ketersediaan dataset berupa video, audio dan file MPD di dalam *web server*. Dalam konsep DASH, video yang disediakan lebih dari 1 kualitas. Untuk proses pengujian di dalam penelitian ini, terdapat 6 representasi video, 3 representasi audio dan 1 file MPD yang sudah ditempatkan pada *web server* yang nantinya akan diakses oleh klien DASH. Masing-masing representasi video dan audio memiliki kualitas dan *bitrate* yang beragam. Dataset yang diuji didapatkan dari Googleapis (2013). Pengumpulan dataset berupa video, audio dan MPD ini dilakukan dengan

cara mengunduh satu persatu melalui menu representasi yang telah disediakan di dalam *player* HTML yang sudah berbasis DASH (Sosro, 2014).

2. Jaringan (*Bandwidth*)

Hal kedua yang dibutuhkan agar DASH dapat berjalan dengan baik adalah dengan memantau kondisi jaringan (*bandwidth*) saat itu. Kegiatan video *streaming* membutuhkan besaran *bandwidth* yang tidak sedikit. Untuk kebutuhan layanan video *streaming* menggunakan DASH, di dalam penelitian ini sedikitnya digunakan *bandwidth* kurang lebih sekitar 5.0 MB agar dapat menjalankan video dengan lancar dan video dengan kualitas yang paling baik.

3. *Web Server*

Hal ketiga yang dibutuhkan untuk dapat mewujudkan layanan video *streaming* baik berbasis DASH maupun tanpa DASH adalah pengimplementasian *web server*. *Web server* ini digunakan sebagai media penyimpanan terhadap konten-konten DASH, yang berupa audio, video dan file MPD. Sudah banyak jenis *web server* yang ditawarkan, tetapi untuk penelitian kali ini cukup menggunakan Apache *Web Server* yang terbilang mudah pada saat proses konfigurasinya.

4. *Player*

Hal terakhir yang dibutuhkan adalah ketersediaan *player* yang digunakan oleh klien untuk memutar konten DASH yang sudah tersimpan di dalam *web server*. *Player* DASH yang digunakan harus memiliki kesesuaian dengan format *codecs* yang digunakan pada file MPD. Banyak format yang digunakan oleh masing-masing *player* DASH, dan untuk penelitian kali ini *player* yang digunakan sudah disesuaikan dengan karakteristik file MPD yaitu dengan *codecs* avc1 dan format mp4.

Semua hal yang sudah dijelaskan di atas juga membutuhkan perangkat keras dan perangkat lunak. Adapun spesifikasi dari perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut.

a. Perangkat Keras

Dalam setiap sistem komputer selalu terdapat perangkat keras dan perangkat lunak yang harus dipenuhi. Untuk membangun layanan video *streaming* dengan menggunakan DASH ini, dibutuhkan perangkat keras berupa komputer yang akan bertindak sebagai *web server*, yang berfungsi untuk menyimpan video dan audio dengan bermacam-macam representasi, serta file MPD yang nantinya akan digunakan pada saat proses pengaksesan konten. Di dalam penelitian ini, pc *server* yang digunakan memiliki spesifikasi seperti berikut:

OS	: Ubuntu 12.04 LTS i686
CPU	: Intel® Core™ i3 550
Processor	: 3.20 GHz
RAM	: 2 GB
Harddisk	: 500 GB
Network Adapter	: LAN Card

Adapun spesifikasi dari sisi klien yang nanti didalamnya akan diterapkan aplikasi *bandwidth limiter* adalah sebagai berikut:

OS	: Windows 7 Home Premium (64-bit)
CPU	: Intel® Core™ i3-2330M
Processor	: 2.20 GHz
RAM	: 2 GB
Harddisk	: 500 GB
Network Adapter	: WLAN *Card*

b. Perangkat Lunak

Layanan video *streaming* berbasis DASH merupakan layanan penyedia jasa *streaming* video dan audio. Perangkat lunak yang dibutuhkan oleh DASH diantaranya adalah Apache *Web Server* yang digunakan sebagai media penyimpanan untuk konten-konten DASH, termasuk di dalamnya file video, audio dan MPD. Implementasi dan konfigurasi *server* selanjutnya akan dijelaskan pada **subbab 3.2**. Selain itu, *player* juga dibutuhkan oleh klien agar dapat memutar konten DASH.

Untuk penelitian ini, digunakan *player* berbasis HTML5 yang sudah mendukung DASH di dalamnya.

Selain perangkat lunak yang dibutuhkan untuk membangun lingkungan berbasis DASH, dibutuhkan juga suatu aplikasi yang berfungsi untuk melakukan pembatasan kondisi *bandwidth* di jaringan agar memenuhi kondisi lingkungan pengujian yang diinginkan. Perangkat lunak yang digunakan adalah NetBalancer dan Trickle. Kedua *tools* tersebut digunakan pada sistem operasi komputer yang berbeda, untuk Netbalancer digunakan untuk klien dengan OS Windows, sedangkan Trickle digunakan untuk klien dengan OS Ubuntu. Aplikasi *bandwidth limiter* ini digunakan untuk memenuhi tujuan penelitian, agar mendapatkan suatu hasil analisis kinerja video *streaming* berbasis DASH.

3.2 Instalasi *Web Server*

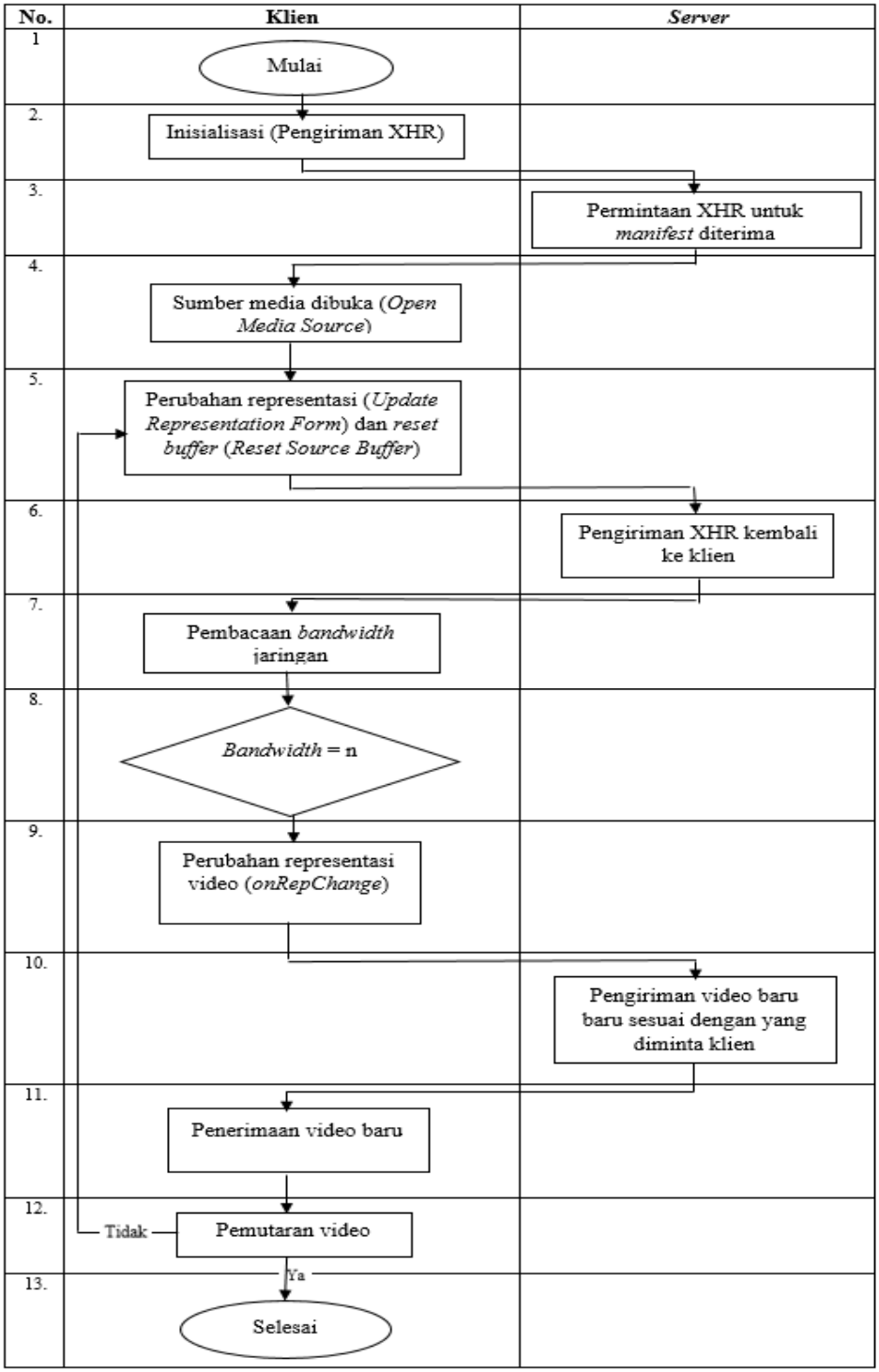
Hal pertama yang dilakukan untuk mewujudkan suatu kegiatan video *streaming* adalah dengan membuat atau mengimplementasikan sebuah *web server* yang nantinya digunakan untuk penyimpanan file-file yang dibutuhkan dalam proses *streaming*. *Web server* diimplementasikan pada salah satu PC di gedung B Laboratorium Jaringan Komputer PTIIK-UB. Pada saat proses implementasi *web server*, diperlukan suatu perangkat keras yang dapat menampung jumlah video yang cukup banyak dengan ukuran masing-masing video yang cukup besar. Instalasi *web server* juga bertujuan sebagai media penyimpanan file-file video dan file MPD yang nantinya diperlukan untuk pengaksesan file multimedia (DASH). Banyak referensi-referensi *web server* yang dapat digunakan, tapi khusus untuk penelitian ini digunakan Apache *Web Server*.

Apache adalah sebuah nama *web server* Internet terkemuka yang bertugas untuk menangani berbagai bentuk respon dan permintaan oleh protokol HTTP dan *logging* informasi secara detail. Apache merupakan salah satu dari banyak *web server* yang sangat digemari, karena bersifat *open source* dan mudah untuk mengeditnya. Apache dapat mendukung dalam mewujudkan suatu lingkungan DASH karena DASH hanya membutuhkan *server* yang dapat menyimpan konten

video beserta audio dan file MPD saja. Apache juga digunakan karena proses pengimplementasiannya yang cukup mudah (Lederer, 2012).

3.3 Alur Proses DASH *Player*

Setelah mengetahui kebutuhan apa saja yang digunakan untuk membangun suatu layanan video *streaming* berbasis DASH seperti yang telah dijelaskan pada **subbab 3.1**, maka dibuatlah suatu alur tentang proses DASH yang dapat menggambarkan cara kerja DASH yang dilihat dari sisi *player* pada klien. Berikut pada **Gambar 3.1** akan digambarkan alur proses DASH dari klien menuju ke *web server* dan begitu juga sebaliknya, serta semua penjelasan proses baik yang terjadi antara *web server* maupun klien.



Gambar 3.1 Alur Proses DASH *Player*

Gambar 3.1 di atas menggambarkan bahwa dalam suatu alur proses DASH, dibutuhkan sebuah *web server* yang berguna untuk menyimpan dataset berupa file MPD, video dan audio dengan beragam kualitas dan representasi. Pada

penelitian kali ini, digunakan 6 buah representasi video dan 3 file audio, dimana file video dan audio tersebut diwakilkan oleh 1 buah file MPD. Sedangkan di sisi klien harus mempunyai aplikasi *player* yang sudah mendukung format DASH. Peran DASH *player* di sisi pengguna sangatlah besar karena dapat menginterpretasikan sendiri (mandiri) antara ketersediaan konten di dalam *web server* dengan kondisi jaringan (*bandwidth*) pada saat itu.

1. Alur proses untuk DASH *player* dimulai.
2. Pertama-tama, inisialisasi selalu dibutuhkan pada awal proses, baik proses video *streaming* maupun proses yang lainnya. Setelah proses inisialisasi dilakukan, klien memulai untuk mengirimkan XHR (*XML HTTP Request*) untuk file *manifest* ke *web server*.
3. *Web server* menerima permintaan dari klien mengenai permintaan file *manifest* melalui XHR tadi.
4. Lalu dilakukan pembukaan sumber media (*media source*) di sisi klien.
5. Kemudian akan dilakukan perintah untuk *update representation form* yang dilakukan pada sisi klien. Selain *update representation form*, maka dilakukan pula *reset source buffer* yang terjadi di sisi klien
6. *Web server* akan mengirimkan kembali XHR berupa file *manifest* yang telah sesuai dengan permintaan ke sisi klien. Pengiriman ini menggunakan URL= http://172.21.3.55/videoitec/youtubedash/car-20120829.mp4/.
7. Klien akan melakukan pembacaan *bandwidth* jaringan saat itu.
8. *Bandwidth* ditunjukkan dengan variabel n. Hal ini dimaksudkan karena *bandwidth* pada jaringan selalu cenderung dinamis.
9. Ketika terjadi perubahan kondisi *bandwidth*, maka klien juga akan melakukan proses perubahan representasi video (*onRepChange*).
10. Lalu *web server* akan bersiap-siap untuk mengirimkan kembali video baru yang baru yang diminta oleh klien dimana telah sesuai dengan kondisi jaringan (*bandwidth*) klien saat itu.
11. Klien menerima video baru dari *web server*.
12. Klien akan memutar video yang baru tersebut, dimana video baru tersebut telah sesuai dengan kondisi jaringan (*bandwidth*) klien saat itu. Namun, jika pada saat pemutaran konten terjadi perubahan kondisi jaringan (*bandwidth*),

maka proses akan diulang pada bagian perubahan representasi dan mereset kembali *buffer* di sisi klien. Setelah itu, barulah dilakukan pembacaan *bandwidth* kembali hingga klien dapat memutar video kembali dengan kondisi jaringan yang baru juga.

13. Proses yang terjadi antara klien dan *web server* akan berhenti ketika video telah selesai dijalankan secara keseluruhan, dan *web server* akan langsung menutup koneksi video *streaming* sampai klien melakukan permintaan kembali ke *web server* untuk melakukan proses *streaming.* Semua proses yang terjadi dilakukan oleh klien menuju ke *web server*. Namun, semua perubahan yang ada di klien murni dikendalikan oleh klien sendiri, dan *web server* hanya menyediakan hal-hal yang dibutuhkan oleh klien, atau dapat dikatakan bahwa *web server* bersifat pasif, sedangkan klienlah yang bersifat aktif.

Tahapan analisis dimulai ketika terjadi perubahan kondisi jaringan (*bandwidth*) yang akan berdampak pada representasi video (*update representation*) dan *reset source buffer*. Ketika representasi video berubah, maka akan secara otomatis *buffer* di sisi klien juga akan ikut diubah (*reset*). Dampak yang diterima oleh klien ketika terjadi perubahan kondisi jaringan adalah klien mau tidak mau menghentikan sementara proses *streaming* hingga klien mendapatkan representasi video terbaru dari *web server*. Waktu yang dibutuhkan sampai klien dapat memutar kembali video tadi, dinamakan dengan waktu *delay*. Pencatatan waktu *delay* diperoleh dari pesan *log* yang terdapat di sisi bawah aplikasi *player* yang digunakan.

3.4 Implementasi Lingkungan Berbasis DASH

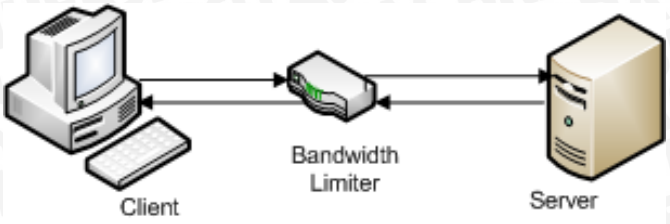
Implementasi lingkungan berbasis DASH membutuhkan suatu lingkungan jaringan dengan kondisi *bandwidth* yang selalu berubah secara dinamis (fluktuatif). Kondisi jaringan dengan *bandwidth* dinamis inilah yang dapat mewujudkan suatu lingkungan yang mendukung sistem DASH. Sesuai dengan kelebihannya, DASH dapat beradaptasi secara dinamis terhadap kondisi jaringan (*bandwidth*) yang dinamis. Pembangunan lingkungan berbasis DASH juga membutuhkan *web server* yang sebelumnya telah dijelaskan pada **subbab 3.2**.

Selain itu, sistem DASH juga membutuhkan suatu dokumen atau file yang dapat dibaca oleh *player* DASH, yakni sebuah file MPD. Dan kebutuhan terakhir yang tidak kalah pentingnya agar lingkungan DASH dapat terwujud adalah keberadaan *player* yang telah disesuaikan format *codecs*nya dengan format *codecs* dari file MPD yang digunakan. Untuk implementasi lingkungan berbasis DASH secara lebih rinci, selanjutnya akan dijelaskan pada **Bab IV Implementasi**.

3.5 Lingkungan Pengujian

Pada tahapan ini akan dijelaskan simulasi terhadap lingkungan pengujian yang akan diterapkan di dalam penelitian. Lingkungan pengujian dibangun untuk proses pengujian lebih lanjut terhadap kinerja dari layanan video *streaming* berbasis DASH. Lingkungan pengujian dalam penelitian ini dilaksanakan pada area jaringan kampus PTIIK-UB. Terdapat 2 skenario pengujian utama di dalam penelitian ini. Skenario pengujian yang pertama yaitu pengujian DASH pada kondisi simulasi. Pengujian pertama dapat dilihat pada **Gambar 3.2** berikut.

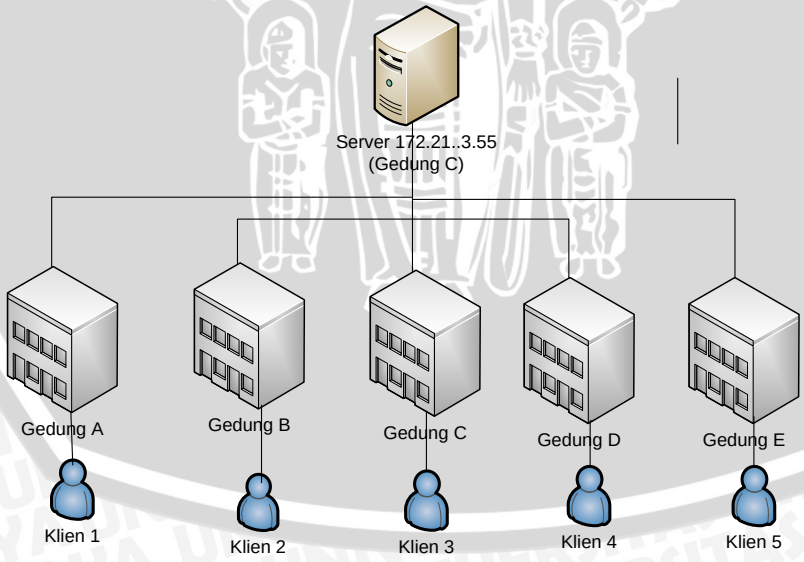
Pengujian pertama ini menghasilkan 3 buah kombinasi pengujian, diantaranya DASH pada kondisi *bandwidth* dinamis, perbandingan *delay* antara *streaming* DASH dan tanpa DASH, serta perubahan (*switching*) representasi video baik untuk *streaming* DASH maupun tanpa DASH. Ketika pengujian *delay* dan representasi video baik untuk *streaming* DASH maupun tanpa DASH, keduanya menggunakan 2 buah *player* yang berbeda, yaitu *player* berbasis HTML5 yang belum mendukung DASH dan *player* HTML5 yang sudah mendukung DASH. *Player* HTML5 yang sudah mendukung DASH inilah yang akan digunakan untuk pengujian-pengujian selanjutnya yang ada di dalam penelitian ini.



Gambar 3.2 Uji Coba DASH Kondisi Simulasi

Pada **Gambar 3.2** di atas, disediakan sebuah *web server* yang berfungsi untuk menyimpan file video, audio dan MPD, serta sebuah klien yang didalamnya telah terinstal aplikasi *bandwidth limiter*. Kondisi ini digunakan untuk keseluruhan dalam pengujian pertama. Aplikasi *bandwidth limiter* yang digunakan pada penelitian ini yaitu NetBalancer dan Trickle. Kedua aplikasi *bandwidth limiter* ini sudah dijelaskan pada **subbab 2.3 Emulasi Jaringan**. Pengujian pertama ini dilakukan di gedung B (Laboratorium Jaringan Komputer) PTIIK-UB.

Lalu pengujian kedua dapat dilihat pada **Gambar 3.3** berikut.



Gambar 3.3 Uji Coba DASH Pada Jaringan Kampus

Pengujian yang terlihat pada **Gambar 3.3** dilakukan berdasarkan kondisi jaringan kampus PTIIK-UB saat itu. Dengan kata lain adalah kondisi nyata.

Pengujian dilakukan di 5 tempat di kampus PTIIK-UB, yaitu pada gedung A, gedung B, gedung C, gedung D dan gedung E. *Web server* ditempatkan pada gedung B (Laboratorium Jaringan Komputer) PTIIK-UB. Pengujian dilakukan dengan mengakses konten DASH yang sudah disimpan di dalam *web server*, dimana pengaksesan dilakukan pada tempat yang berbeda-beda.

2 skenario pengujian di atas menghasilkan 8 buah kombinasi hasil pengujian diantaranya adalah pengujian DASH pada kondisi *bandwidth* dinamis, perbandingan *delay* antara *streaming* DASH dan tanpa DASH, perbandingan perubahan (*switching*) representasi video antara *streaming* DASH dan tanpa DASH, pengujian pada jaringan kampus (gedung A), pengujian pada jaringan kampus (gedung B), pengujian pada jaringan kampus (gedung C), pengujian pada jaringan kampus (gedung D) dan yang terakhir adalah pengujian pada jaringan kampus (gedung E).

Sebelum semua proses pengujian dilakukan, dibutuhkan sebuah dataset yang layak dan dapat diuji. Selain itu juga dibutuhkan skenario pengujian agar pengujian dapat berjalan sesuai dengan yang diharapkan. Kedua proses ini sangat berkaitan karena untuk menjalankan skenario pengujian yang telah dibuat, diperlukan suatu dataset yang sesuai dengan kondisi lingkungan pengujian saat itu dan kesesuaian dengan sistem yang telah dibangun (sistem DASH).

3.5.1 Pengumpulan Dataset

Pengumpulan dataset sebelumnya sudah dijelaskan pada **subbab 3.1**. Proses video *streaming* berbasis DASH menggunakan sebuah file *manifest* dimana pada *streaming* yang lain tidak ada. File *manifest* (MPD) ini memiliki peranan penting karenaa di dalamnya berisikan karakteristik-karakteristik tentang konten yang akan diputarnya. Pentingnya file MPD seperti sudah dijelaskan pada **subbab 2.2.4.1** dan pada **Gambar 2.4**. Berikut pada **Gambar 3.4** akan dijelaskan bagaimana mengumpulkan file video yang telah disediakan di dalam menu representasi *player*.

Gambar 3.4 Pengumpulan File Video

Pada **Gambar 3.4** merupakan tampilan *player*, dimana terdapat berbagai menu di dalamnya. Cara mendapatkan file video dengan berbagai representasi adalah dengan cara mengunduh satu persatu dari menu representasi yang terdapat di dalam *player*. Selain itu terdapat kolom adaptasi yang didalamnya terdapat 3 pilihan adaptasi, yaitu otomatis, acak dan non aktif. Adaptasi otomatis bertujuan untuk mengaktifkan adaptasi agar dinamis sesuai dengan kondisi jaringan saat itu. Adaptasi acak digunakan untuk membuat representasi video yang diterima oleh klien secara acak, tetapi juga bergantung pada kondisi jaringan klien saat itu. Sedangkan untuk adaptasi non aktif digunakan ketika klien ingin memilih sendiri representasi video dengan kualitas apa yang ingin diputar, yang juga disesuaikan dengan kondisi jaringan (*bandwidth*) saat itu.

Berikut pada **Tabel 3.1** dan **Tabel 3.2** akan ditampilkan karakteristik file video dan audio yang telah dikompresi menjadi beberapa versi ukuran dan juga *bitrate*.

Tabel 3.1 Karakteristik Video untuk Simulasi DASH

Nama File	Resolusi/ *Bitrate*	Ukuran Resolusi	Representasi *Bandwidth*	Id Video
car-20120827-89.mp4	1080	1920x1080	4191 Kbps	1
car-20120827-88.mp4	720	1280x720	2074 Kbps	2
car-20120827-87.mp4	480	854x480	869 Kbps	3
car-20120827-86.mp4	360	640x360	686 Kbps	4
car-20120827-85.mp4	240	426x240	265 Kbps	5
car-20120827-160.mp4	144	256x144	100 Kbps	6

Tabel 3.2 Karakteristik Audio untuk Simulasi DASH

Nama File	*Sample Rate*	Representasi *Bandwidth*	Id Video
car-20120827-8c.mp4	44100	127 Kbps	7
car-20120827-8d.mp4	44100	255 Kbps	8
car-20120827-8b.mp4	22050	31 Kbps	9

Karakteristik yang terdapat pada **Tabel 3.1** dan **Tabel 3.2** di atas didapatkan dari file MPD yang telah diunduh secara manual melalui Googleapis (2014). File MPD tersebut berisi semua karakteristik yang dibutuhkan untuk pemutaran konten DASH dari awal hingga akhir. Karakteristik tersebut diantaranya adalah mengenai jenis file, ukuran resolusi, *bitrate*, representasi *bandwidth*, *sample rate* dan juga id video. Id video menunjukkan tingkatan untuk file video dan juga audio dari yang paling tinggi hingga yang paling rendah dari segi kualitasnya. Id disini juga dapat dikatakan sebagai penanda untuk masing-

masing file video dan audionya. Id dibuat berurutan untuk memudahkan pembacaan baik oleh klien maupun oleh *server*, karena id 1 menunjukkan representasi video dengan kualitas yang paling baik, begitu juga dilanjutkan hingga representasi audio yang ditempatkan di akhir. Id disini dituliskan secara unik, atau dengan kata lain tidak sama dengan yang lainnya karena id juga digunakan sebagai pembeda antara file yang satu dengan file yang lainnya.

3.5.2 Skenario Pengujian

Penelitian ini menggunakan 2 buah skenario pengujian, yang pertama yaitu pengujian DASH pada kondisi simulasi dan yang kedua yaitu pengujian DASH pada kondisi jaringan kampus (PTIIK-UB). Pengujian pertama dilakukan dengan cara melakukan pembatasan *bandwidth* pada saat proses pemutaran konten yang sudah tersedia di dalam *web server*. Sedangkan pengujian kedua dilakukan tanpa menggunakan aplikasi *limiter* apapun, dan pengujian dilakukan di masing-masing gedung di area kampus PTIIK-UB.

Lingkungan pengujian secara detail telah dijelaskan pada **subbab 3.5**. Terdapat 2 skenario pengujian utama yang dilakukan, diantaranya yaitu pengujian DASH pada kondisi simulasi dan pengujian DASH pada jaringan kampus (PTIIK-UB). Dari 2 buah skenario utama tersebut, dihasilkan 8 buah kombinasi skenario pengujian yang dilakukan di dalam penelitian ini, diantaranya pengujian DASH pada kondisi *bandwidth* dinamis, perbandingan *delay* antara *streaming* DASH dan tanpa DASH, perbandingan perubahan (*switching*) representasi video antara *streaming* DASH dan tanpa DASH, pengujian pada jaringan kampus (gedung A), pengujian pada jaringan kampus (gedung B), pengujian pada jaringan kampus (gedung C), pengujian pada jaringan kampus (gedung D) dan yang terakhir adalah pengujian pada jaringan kampus (gedung E).

Teknis pelaksanaan pengujian terbagi menjadi 2, yaitu untuk pengujian DASH pada kondisi simulasi menggunakan bantuan dari aplikasi *bandwidth limiter* yang dipasangkan di sisi klien yang akan mengakses layanan video *streaming* DASH. Aplikasi *bandwidth limiter* yang digunakan dalam penelitian adalah NetBalancer. Sedangkan untuk pengujian DASH pada kondisi jaringan kampus, dilakukan dengan mengakses konten DASH secara langsung tanpa

membutuhkan aplikasi *bandwidth limiter* apapun. Pencatatan *delay* pada kedua pengujian ini dilakukan dengan cara memantau besaran rata-rata *download* dan *upload* pada aplikasi NetTray (bagian dari aplikasi NetBalancer yang dapat digunakan untuk memantau lonjakan *bandwidth* jaringan). Selain menggunakan NetTray, perubahan *delay* dan representasi video juga dapat dilihat dari pesan *log* yang terdapat di sisi bawah *player* HTML5 DASH yang digunakan.

Hal yang akan diuji pada pengujian pertama untuk pengujian DASH pada kondisi simulasi adalah dari segi sifat dinamis DASH yang dapat mengikuti perubahan yang terjadi pada jaringan saat itu, besar perubahan *delay* baik untuk *streaming* DASH maupun tanpa DASH dan perubahan (*switching*) representasi video baik untuk *streaming* DASH maupun tanpa DASH. Kemudian pengujian kedua dilakukan di area jaringan kampus (PTIIK-UB) dilakukan pada 5 tempat yang berbeda yaitu gedung A, gedung B, gedung C, gedung D dan gedung E. Pengujian kedua ini menghasilkan rata-rata nilai *bandwidth* estimasi (BE) dan *bandwidth* representasi (BR) yang digunakan oleh sistem DASH.

Skenario pengujian di atas dilakukan dengan menggunakan 1 buah file MPD *sample*. Proses pencatatan dilakukan ketika terjadi proses perubahan kondisi jaringan (*bandwidth*) yang berakibat pada perubahan (*switching*) representasi video yang sudah tersimpan di dalam *web server*. Hal ini juga mengakibatkan adanya *delay* yang terjadi ketika ada perubahan kondisi jaringan (*bandwidth*) dengan perubahan (*switching*) representasi video. Proses perubahan (*switching*) representasi video terjadi ketika *bandwidth* yang digunakan untuk mengakses layanan tersebut dibatasi (diubah-ubah), maka video akan merujuk ke id masing-masing sesuai yang dituliskan dan ditentukan pada file MPD.

Sistem dianggap dapat berjalan dengan baik ketika layanan tetap berhasil menyediakan konten yang diminta oleh klien, meskipun dalam keadaaan *bandwidth* yang terbatas (*bandwidth* minimum). Oleh karena itu, pengujian dilakukan dengan menentukan parameter-parameter yang nantinya dapat menghasilkan suatu analisis kinerja DASH pada lingkungan jaringan kampus. Parameter-parameter yang dicatat antara lain adalah jumlah perubahan (*switching*) representasi video, perubahan untuk nilai *bandwidth* estimasi (Kbps) yang tersedia pada jaringan saat itu dan *bandwidth* representasi (Kbps) yang digunakan oleh

DASH. Parameter lainnya yang tidak kalah penting untuk diuji adalah besar nilai *delay* yang terjadi saat dilakukan pemutaran konten DASH dari awal hingga akhir, dan jumlah kejadian *delay* dari awal hingga akhir pemutaran konten (Wang, 2003).

Pencatatan semua perubahan yang terjadi dilakukan dari awal pemutaran video hingga berakhirnya video di sisi klien. Data didapatkan dari pengujian proses pengunduhan dan pemutaran konten dengan ukuran maksimal konten yaitu 90,8 MB, dengan representasi *bandwidth* yang digunakan yaitu 4191 Kbps. Skenario pengujian dibuat dengan tujuan melihat kinerja layanan video *streaming* berbasis DASH, baik jika diterapkan pada kondisi simualasi maupun diterapkan pada jaringan kampus, dimana memiliki kondisi *bandwidth* yang terbatas dan seringkali berubah-ubah (sering terjadi lonjakan/interferensi).

Tabel 3.3 Skenario Pengujian

Pengujian	**Skenario 1**	**Skenario 2**	**Lingkungan Pengujian**
DASH	Uji Coba DASH Pada Kondisi Simulasi	Uji Coba DASH Pada Jaringan Kampus	- *Bandwidth* 100 Mbps - *Latency* 64 ms

3.6 Hasil dan Pembahasan

Untuk mengukur kinerja dari layanan video *streaming* berbasis DASH yang diterapkan di area jaringan kampus, ada beberapa parameter hasil pengujian yang diukur, dianalisa serta dijadikan acuan dalam pembahasan, diantaranya yaitu:

1. Sifat DASH ketika diuji coba pada kondisi *bandwidth* yang dinamis (fluktuatif), serta jumlah waktu *delay* yang terjadi dan perubahan (*switching*) representasi video yang menunjukkan kualitas video yang digunakan, baik untuk layanan video *streaming* tanpa menggunakan DASH dan layanan video *streaming* menggunakan DASH.
2. Jumlah rata-rata besaran nilai *bandwidth* representasi, jumlah kejadian *delay* yang terjadi dan kualitas video DASH yang sering digunakan di lingkungan jaringan kampus PTIIK-UB.

3.7 Pengambilan Kesimpulan

Pengambilan kesimpulan dapat diperoleh setelah pengujian, hasil analisis dan pembahasan selesai dilakukan. Pengambilan kesimpulan merupakan tahapan akhir dan juga merupakan hasil akhir dari penelitian yang telah dilakukan. Kesimpulan disusun berdasarkan hasil dari pembahasan mengenai kinerja video *streaming* berbasis DASH di area jaringan kampus. Isi pada kesimpulan diharapkan dapat menjadi acuan untuk pengembangan penelitian pada bidang terkait selanjutnya. Jika terdapat ketidaksesuaian antara hasil pengujian dengan teori yang ada, ataupun terdapat masukan mengenai penelitian terkait selanjutnya, maka akan ditambahkan ke dalam bagian saran.

BAB IV
IMPLEMENTASI

Pada bab implementasi akan dijelaskan mengenai langkah-langkah untuk membangun sebuah layanan video *streaming* berbasis DASH. Hal-hal yang dibutuhkan agar lingkungan berbasis DASH dapat terwujud diantaranya adalah sebagai berikut:

1. Jaringan (*bandwidth*) yang dinamis (fluktuatif).
2. Instalasi dan Konfigurasi *Web Server*.
3. File MPD.
4. *Player*, yang formatnya telah disesuaikan dengan format yang digunakan oleh file MPD.

4.1 Jaringan (*Bandwidth*)

Hal yang paling utama agar dapat membangun lingkungan berbasis DASH adalah memastikan bagaimana kondisi jaringan (*bandwidth*) saat itu. Kondisi *bandwidth* yang digunakan untuk DASH adalah kondisi *bandwidth* yang berubah-ubah secara dinamis (fluktuatif). Kondisi *bandwidth* seperti inilah yang digunakan oleh DASH karena sesuai dengan namanya, dimana DASH dapat secara dinamis beradaptasi dengan kondisi jaringan (*bandwidth*) yang berubah secara dinamis (fluktuatif) setiap saatnya.

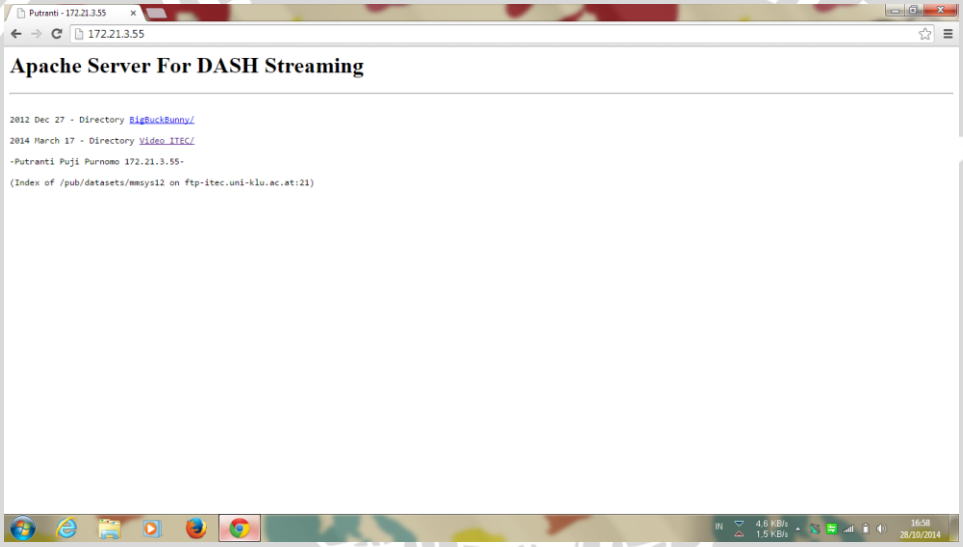
Penelitian ini juga menggunakan bantuan dari aplikasi *bandwidth limiter* agar pengkondisian jaringan dapat berjalan sesuai dengan yang diinginkan. Cara melimit *bandwidth* sudah dijelaskan sebelumnya pada **subbab 2.3**. Pelimitan *bandwidth* ini berfungsi untuk melihat bagaimana performa DASH jika kondisi jaringan (*bandwidth*) saat itu berubah-ubah (fluktuatif).

4.2 Instalasi dan Konfigurasi *Web Server*

Web server pada sistem ini dibangun dengan menggunakan 1 buah komputer. Komputer untuk *web server* tersebut menyimpan berbagai macam video dengan beragam representasi beserta audionya, serta file MPD yang

digunakan oleh klien untuk mengakses konten video dan audionya. *Web server* diimplementasikan pada komputer dengan sistem operasi Ubuntu 12.04 (32-bit).

Penelitian ini menggunakan Apache *Web Server* karena Apache tergolong mudah untuk proses pengimplementasiannya, dan layanan video *streaming* berbasis DASH hanya membutuhkan suatu *web server* sederhana dengan kapasitas yang cukup besar agar dapat menyimpan semua konten-konten yang dibutuhkan oleh klien DASH. Berikut pada **Gambar 4.1** akan disertakan bagaimana tampilan awal dari Apache *Web Server* untuk layanan video *streaming* berbasis DASH.



Gambar 4.1 Tampilan Awal Apache *Web Server*

Pada **Gambar 4.1** diatas merupakan gambar tampilan awal dari Apache *Web Server* yang digunakan sebagai media penyimpanan konten DASH (berupa video, audio dan file MPD). Konten DASH sudah dijelaskan sebelumnya pada **subbab 3.5.1** bahwa konten DASH yang digunakan di dalam penelitian ini diambil dengan cara melakukan pengunduhan satu persatu dari *player*. *Web server* ditempatkan pada alamat http://172.21.3.55/.

Pada gambar terdapat 2 buah folder utama, yaitu dengan nama BigBuckBunny dan video ITEC. Penamaan ini tidak berpengaruh terhadap konten yang terdapat di dalamnya. Untuk penelitian ini, digunakan folder bernama video ITEC, sedangkan folder BigBuckBunny merupakan dataset yang diambil

dari ITEC (ITEC, 2014) yang tidak dapat diputar di dalam *player* HTML5 DASH yang digunakan dalam penelitian. Namun file di dalam folder ini dapat diputar pada aplikasi *player* berbasis *dekstop*, yaitu VLC dengan versi 2.1.3 ke atas. Versi inilah yang sudah dapat memutar konten DASH.

4.3 File MPD (*Media Presentation Description*)

Di dalam *web server* sudah terdapat beragam video dan audio dengan beragam representasi dan kualitas. File yang beragam ini di dalam konteks DASH diwakilkan oleh sebuah file, dimana di dalamnya juga terdapat sifat dinamis yang dimiliki oleh konten tersebut yang dinyatakan dalam bentuk file XML, yang di dalam DASH lebih dikenal dengan nama MPD (*Media Presentation Description*). File MPD harus memiliki kesesuaian format *codecs* dengan format *codecs* yang dimiliki oleh *player*. Hal ini harus disesuaikan karena bisa jadi MPD yang telah dibuat tidak dapat diputar hanya karena terdapat perbedaan format *codecs* antara keduanya.

Berikut pada **Gambar 4.2** akan ditampilkan potongan file MPD yang digunakan di dalam penelitian. File MPD ini akan ditempatkan pada *web server* yang beralamatkan di http://172.21.3.55/. Untuk tampilan file MPD secara keseluruhan dapat dilihat pada **Bab Lampiran**.

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S"
minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760"
codecs="avc1.640028" height="1080" id="1"
mimeType="video/mp4" width="1920">
        <BaseURL>car-20120827-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
```

```
            <Initialization range="0-673" />
          </SegmentBase>
        </Representation>
.......
      </AdaptationSet>
  </Period>
</MPD>
```

Gambar 4.2 Tampilan File `car.mpd`

Pada **Gambar 4.2** merupakan tampilan file MPD yang juga merupakan sebuah dokumen XML, yang di dalamnya berisikan informasi-informasi mengenai karakteristik dari konten multimedia yang akan diakses. *MediaPresentationDuration* menjelaskan durasi video yang akan dijalankan. *minBufferTime* menjelaskan jumlah durasi waktu inisialisasi sebelum memulai presentasi. Besaran waktu minimal untuk pengisian *buffer* pada tiap MPD adalah 1,5 detik. *Base* URL menunjukkan alamat direktori tempat disimpannya file video dan audio. *Segmentbase index range* menunjukkan panjang video yang akan diputar. *ContentComponent* terdiri dari *ContentType* yang menunjukkan tipe konten, baik untuk konten video maupun audio, serta id yang digunakan sebagai penanda, seperti id = 1 untuk video, dan id =2 untuk audio. Format (*mimeType*) video dan audio yang digunakan adalah mp4, dengan jenis *codecs* untuk file video yaitu avc1 dan *codecs* untuk audio adalah mp4a.

4.4 *Player*

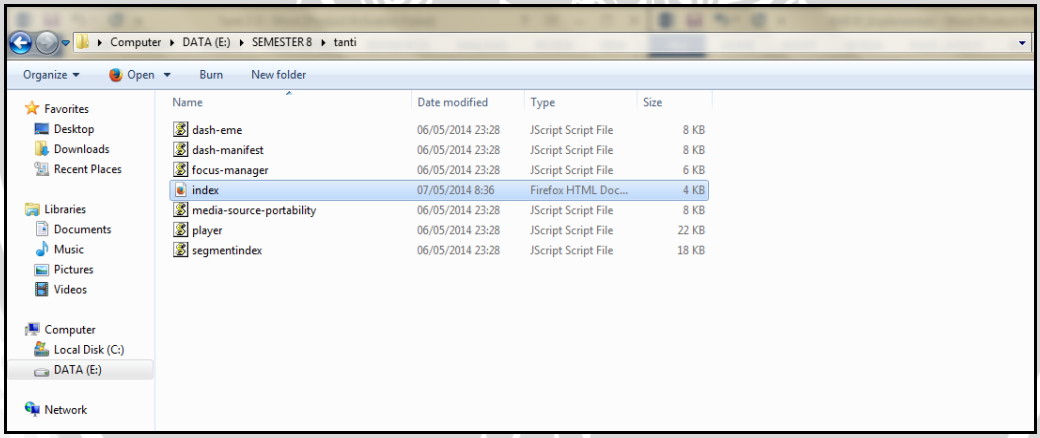
Setelah tahapan instalasi Apache *Web Server* sampai pembuatan file MPD selesai dilakukan, maka untuk mengecek apakah MPD tersebut dapat diputar atau tidak adalah dengan cara memutarnya pada sebuah *player*. Untuk DASH, banyak *player* yang disediakan, tetapi tidak semua *player* dapat memutar MPD dengan baik.

DASH memberikan keleluasaan dengan menyediakan beragam *script* pembuatan *player* DASH di Internet. *Player* tersebut diantaranya adalah DASH-JS (Rainer, 2012), *Web*M DASH *Player* (Galligan, 2014) dan HTML5 DASH

(Sosro, 2014). HTML5 merupakan sebuah layanan video *streaming* yang sudah banyak digunakan. Tetapi tidak semua *player* HTML5 sudah mendukung teknologi DASH.

Player yang sudah dijelaskan di atas tidak semuanya dapat memutar semua file MPD hasil *generate* dengan *tools* yang beragam. Kesesuaian antara format *codecs* yang dimiliki oleh file MPD dengan *player* harus diperhatikan, karena jika format *codecs* berbeda, maka *player* pun tidak akan dapat memutar konten DASH sesuai dengan yang diinginkan oleh klien.

Penelitian ini menggunakan *player* dari HTML5 yang sudah berbasis DASH. *Player* diambil dari situs Internet (Sosro, 2014). Pertama-tama dilakukan pengunduhan terhadap file zip untuk *player* yang akan digunakan dalam penelitian. Setelah itu dilakukan pengekstrakan isi dari file zip tersebut untuk mengetahui apa saja isi di dalamnya. Pada **Gambar 4.3** berikut merupakan hasil ekstrak dari file zip (Sosro, 2014).

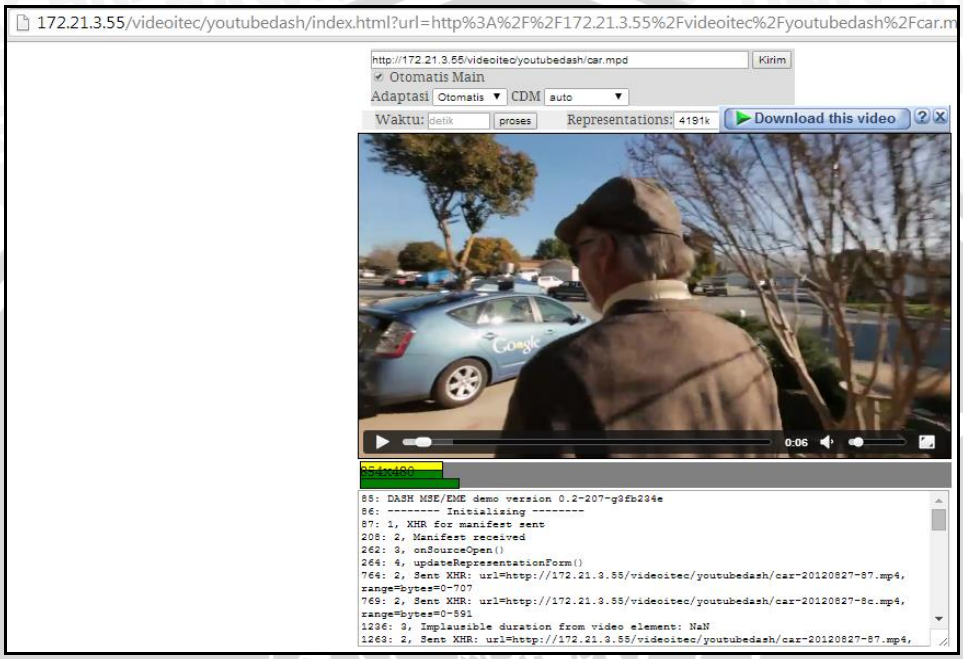


Gambar 4.3 Tampilan Ekstrak File *Player* HTML5

Pada **Gambar 4.3** di atas menunjukkan tampilan dari file-file setelah berhasil diekstrak. File tersebut berasal dari Sosro (2014). Di dalamnya terdapat file index.html yang merupakan tampilan awal *player*, lalu file-file Javascript lainnya seperti dash-eme.js, dash-manifest.js, focus-manager.js, media-source-portability.js, player.js dan segmentindex.js. Di dalam player.js ditanamkan *script* untuk pembacaan *bandwidth*. Dengan adanya *script* tersebut, maka DASH sudah

dapat dikatakan sebagai suatu layanan video *streaming* yang bersifat adaptif terhadap kondisi jaringan (*bandwidth*) di sekitarnya.

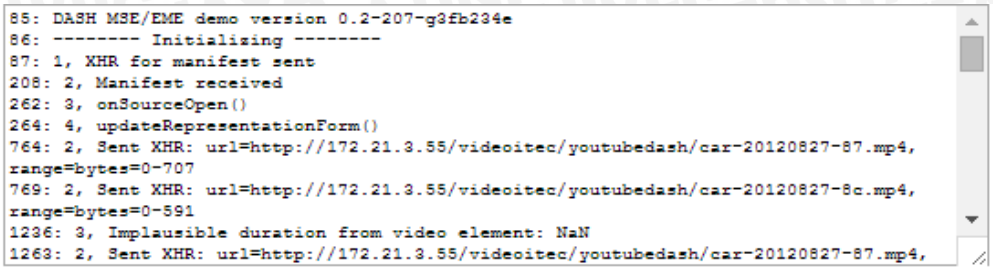
Pada **Gambar 4.4** di bawah ini merupakan tampilan dari aplikasi HTML5 berbasis DASH yang berhasil memutar konten DASH dengan baik di sisi klien.



Gambar 4.4 Tampilan Aplikasi *Player* HTML5 Berbasis DASH

Pada **Gambar 4.4** di atas menunjukkan bahwa klien dapat memutar konten DASH dengan baik. Kualitas konten DASH yang diputar disesuaikan dengan kondisi jaringan (*bandwidth*) di sisi klien saat itu. Jika ketersediaan *bandwidth* saat itu tinggi, maka kualitas yang diputar juga kualitas yang paling baik, dan begitu juga sebaliknya. Ketika alamat URL selesai diinputkan dan ditekan tombol *submit*, maka video perlahan akan dimuat. Hal pertama yang secara cepat dapat dimuat sampai habis adalah untuk representasi audionya. Setelah itu, perlahan representasi video akan dimuat juga per bagiannya. Warna kuning menunjukkan video dan audio yang sedang diputar oleh klien. Warna hijau pertama di bawah warna kuning menunjukkan representasi video yang berhasil dimuat, sedangkan warna hijau kedua (warna paling bawah pada tampilan *player*) menunjukkan representasi audio yang sudah berhasil dimuat. Seiring berjalannya waktu, warna-warna indikasi tersebut akan bertambah, dan saling menyesuaikan dengan kondisi

bandwidth jaringan saat itu. Dengan tampilan seperti **Gambar 4.4** di atas, maka tahapan implementasi dari awal hingga persiapan *player* dapat dikatakan selesai.



Gambar 4.5 Tampilan Pesan *Log* Pada *Player*

Pada **Gambar 4.5** di atas menunjukkan pesan *log* yang muncul ketika melakukan pengaksesan konten DASH di awal. Pesan *log* terjadi pada saat pengaksesan video dari awal hingga berakhirnya video. Macam-macam pesan *log* yang muncul pada saat pemutaran video dari awal hingga akhir yaitu diantaranya NaN yang berarti *Not a Number*. NaN ditampilkan jika tidak ada audio atau video yang diset. Lalu *appending init* berarti menambahkan suatu catatan informasi ke dalam file init. Kemudian ketika dilakukan proses membuka file audio dan video akan keluar peringatan *onSourceOpen*. Ketika pada tengah-tengah pemutaran video terjadi perubahan kondisi jaringan (*bandwidth*) yang mengakibatkan perubahan pula pada representasi video yang ditampilkan, maka *log* akan memunculkan *onRepChange*. *onRepChange* dimaksudkan untuk merubah representasi video maupun audio ke representasi lain yang tersedia di dalam *web server* sesuai dengan kondisi jaringan (*bandwidth*) saat itu. Selain itu juga terdapat *reset source buffer* untuk menampilkan jumlah *buffer* yang dibutuhkan untuk memutar 1 konten video dan audio tersebut, dan juga *update representation form* yang menunjukkan perubahan tampilan representasi ketika terdapat perubahan di sisi jaringan.

BAB V
HASIL DAN PEMBAHASAN

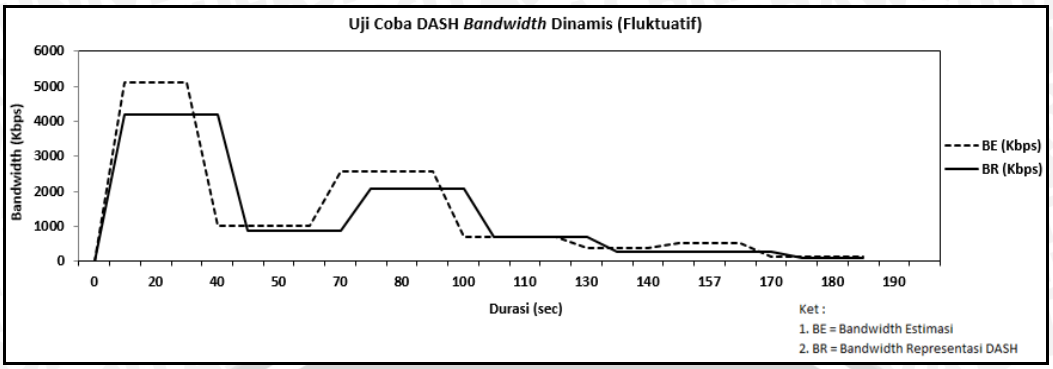
Pada bab ini akan dijelaskan mengenai hasil yang didapatkan setelah menerapkan lingkungan pengujian yang dirasa sesuai mewakili dengan kondisi yang sebenarnya. Teknis pengujian dapat dilihat pada **subbab 3.5.2 Skenario Pengujian**. Setelah hasil didapatkan, maka dilakukanlah suatu pembahasan yang berguna untuk merangkum dari hasil-hasil pengujian yang telah dilakukan.

5.1 Uji Coba DASH Pada Kondisi Simulasi

5.1.1 DASH Pada Kondisi *Bandwidth* Dinamis (Fluktuatif)

Pengujian pertama didapatkan pada saat pengaksesan DASH pada kondisi simulasi, yaitu dengan kondisi *bandwidth* dinamis (fluktuatif). Dalam pengujian yang pertama ini, pengukuran untuk *bandwidth* estimasi (BE) didapatkan dengan cara melakukan pemantauan trafik jaringan dengan menggunakan NetTray, yang merupakan bagian dari aplikasi NetBalancer. Lalu untuk *bandwidth* representasi (BR) didapatkan melalui pemantauan terhadap kualitas video yang diputar, selain itu juga didapatkan dari pesan *log* yang disediakan oleh aplikasi *player* DASH yang digunakan di dalam penelitian ini. Untuk lebih jelasnya, sudah dijelaskan pada **subbab 3.5.2 Skenario Pengujian.**

Hasil dapat dilihat pada **Gambar 5.1** yang menunjukkan bahwa DASH dengan dinamis dapat berubah-ubah sesuai dengan kondisi *bandwidth* yang ditentukan saat itu. Pada gambar terlihat bahwa garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth*, dan *second* (detik) untuk satuan durasi video.



Gambar 5.1 Uji Coba DASH Kondisi *Bandwidth* Dinamis (Fluktuatif)

Pembahasan pada sub-bab mengenai pengujian DASH pada kondisi simulasi, terutama pada kondisi *bandwidth* dinamis (fluktuatif) adalah sebagai berikut:

a. Berdasarkan pada **Gambar 5.1** bahwa dari awal pemutaran konten DASH hingga berakhirnya pemutaran video, terlihat bahwa *bandwidth* estimasi diset fluktuatif. Kondisi fluktuatif yang ditunjukkan dengan BE didapatkan dari pengaturan nilai *bandwidth* pada aplikasi *bandwidth limiter* (NetBalancer), dimana dipasangkan pada komputer klien yang sedang mengakses konten DASH. Hal ini untuk menguji apakah proses perubahan *bandwidth* estimasi tersebut juga berdampak pada *bandwidth* representasi yang digunakan oleh DASH, serta melihat apakah DASH dapat beradaptasi terhadap lingkungan jaringan (*bandwidth*) sekitarnya yang senantiasa berubah-ubah tiap waktunya.
b. Representasi video tidak langsung berubah ketika terjadi perubahan pada sisi *bandwidth* jaringan. Video akan terus menghabiskan jatahnya pada *bandwidth* sebelum terdapat perubahan. Dan setelah beberapa detik video berjalan pada *bandwidth* sebelumnya untuk menghabiskan jatah video dalam *buffer* (sedangkan *bandwidth* jaringan sudah berubah), barulah dia mengisi *buffer* dengan representasi video yang baru sesuai dengan *bandwidth* yang baru juga.
c. Kondisi *bandwidth* yang fluktuatif ini juga berdampak pada video yang diterima oleh klien. Meskipun BE di sisi klien saat itu mencapai 128 Kbps, klien tetap dapat menikmati video DASH. Video dengan representasi *bandwidth* kecil, kualitasnya pun juga rendah dan tidak sebaik kualitas video

dengan representasi *bandwidth* yang besar. Jadi, DASH tetap dapat melakukan pemutaran video di sisi klien meskipun kondisi *bandwidth* di sisi klien saat itu berubah-ubah (fluktuatif), hanya saja untuk kualitas video disesuaikan dengan kondisi *bandwidth* dan ketersediaan video di dalam *web server*.

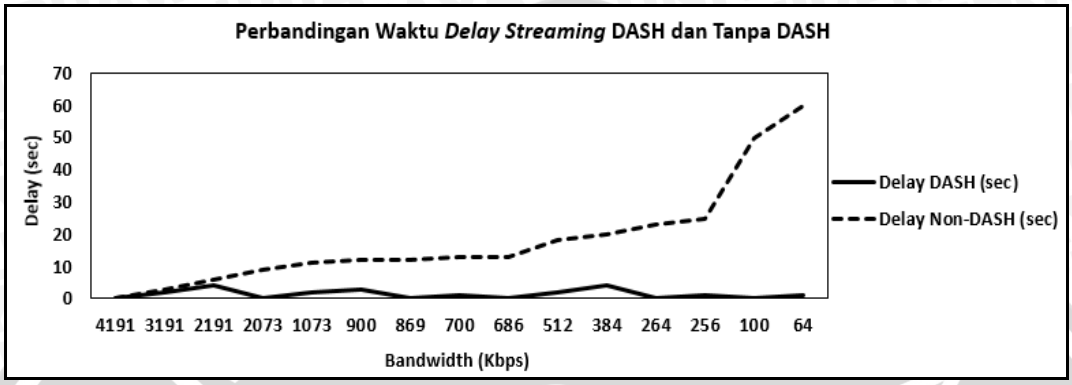
5.1.2 Perbandingan *Delay*

Setelah mendapatkan hasil mengenai DASH pada kondisi simulasi terutamanya pada kondisi *bandwidth* dinamis (fluktuatif), selanjutnya akan dilakukan pengujian untuk membandingkan kinerja layanan video *streaming* dengan DASH dan tanpa DASH, dari segi perbandingan *delay* yang terjadi. Video *streaming* tanpa DASH dilakukan dengan mengakses layanan video *streaming* HTML5 yang belum mendukung DASH, dimana di dalamnya sudah terdapat video yang telah siap diputar oleh klien. Sedangkan video *streaming* dengan DASH dilakukan dengan cara mengakses konten DASH yang sudah terdapat di dalam *web server*, dimana *player* yang digunakan berasalah dari *player* HTML5 yang sudah mendukung DASH di dalamnya.

Hasil yang didapatkan terkait perbandingan *delay* antara layanan video *streaming* dengan menggunakan DASH dan tanpa menggunakan DASH tampak pada **Gambar 5.2** berikut. Pada **Gambar 5.2** menunjukkan grafik perbandingan *delay* antara video *streaming* tanpa DASH dan video *streaming* dengan DASH. Parameter *delay* didapatkan ketika terjadi perubahan kondisi jaringan yang berdampak pula pada berubahnya kualitas representasi video yang diterima oleh klien. *Delay* disini terjadi ketika *server* ingin menyesuaikan kualitas output video yang akan dikirim ke klien dengan kondisi perubahan jaringan saat itu, dimana *server* membutuhkan waktu untuk mengisi ulang *buffer* dengan kualitas video yang baru dan juga disesuaikan oleh kondisi jaringan. *Delay* juga dapat dikatakan sebagai waktu yang dibutuhkan oleh sumber (*server*) untuk mengirimkan kontennya menuju ke tujuan (klien) (Abror, 2010). *Delay* disini juga dapat dikatakan sebagai proses *buffering*.

Pada **Gambar 5.2** di bawah ini, garis berbentuk patah-patah menunjukkan *delay* untuk video *streaming* tanpa menggunakan DASH, dan garis berbentuk

lurus menunjukkan *delay* untuk video *streaming* menggunakan DASH. Grafik menunjukkan adanya perbedaan *delay* yang cukup besar antara video *streaming* DASH dan tanpa DASH.



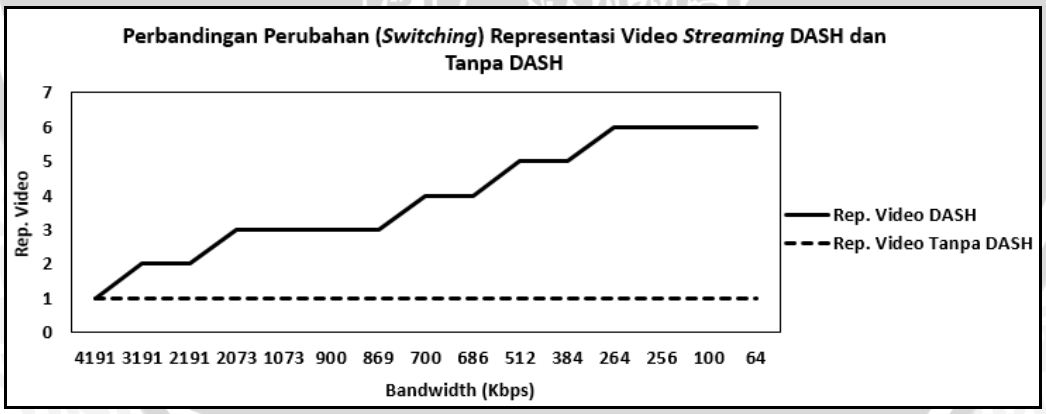
Gambar 5.2 Perbandingan Waktu *Delay Streaming* DASH dan Tanpa DASH

Pembahasan pada sub-bab mengenai perbedaan besar nilai *delay* untuk *streaming* DASH dan tanpa DASH adalah sebagai berikut:

a. Berdasarkan **Gambar 5.2** terlihat bahwa *delay* DASH lebih kecil dibandingkan dengan *delay* tanpa DASH. *Delay* yang naik dan turun tersebut terjadi karena adanya perubahan dari sisi *bandwidth* yang mulanya 4191 Kbps menjadi 64 Kbps. *Delay* pada DASH maksimal 4 detik karena DASH dapat secara dinamis menyesuaikan dengan kondisi representasi *bandwidth* lainnya yang sesuai dengan kondisi *bandwidth* di jaringan saat itu. Berbeda dengan video *streaming* biasa tanpa DASH yang mencapai *delay* hingga 60 detik, karena waktu yang digunakan untuk beradaptasi dengan perubahan *bandwidth* pada jaringan terlalu lama, sedangkan layanan tidak dapat memberikan video lain yang sementara dapat diputar oleh klien selagi menunggu *bandwidth* kembali seperti keadaan awal.
b. Dengan menggunakan DASH, dapat diputuskan *bandwidth* berapa yang sesuai dengan kebutuhan, dan juga disesuaikan dengan video yang akan ditampilkan. Dan secara penggunaannya, *delay* yang terjadi ketika ada perubahan *bandwidth* adalah sedikit atau dapat dikatakan tidak sebanyak *delay* yang terjadi pada saat video *streaming* tanpa DASH.

5.1.3 Perubahan (*Switching*) Representasi Video

Hasil yang didapatkan terkait perubahan representasi video, baik untuk *streaming* DASH dan tanpa DASH tampak pada **Gambar 5.3** berikut. Perubahan (*switching*) representasi video di dalam DASH juga menunjukkan perubahan kualitas video. Teknis pengujian dapat dilihat pada **subbab 3.5.2 Skenario Pengujian**. Pada **Gambar 5.3** menunjukkan grafik perbandingan perubahan (*switching*) representasi video yang digunakan antara video *streaming* tanpa DASH dan *streaming* dengan DASH. Garis berbentuk patah-patah menunjukkan perubahan representasi video yang digunakan untuk *streaming* tanpa DASH dan garis berbentuk lurus menunjukkan perubahan representasi video yang digunakan untuk *streaming* dengan menggunakan DASH. Grafik menunjukkan bahwa terjadi perbedaan perubahan representasi video antara *streaming* DASH dan tanpa DASH. Representasi video yang digunakan untuk *streaming* DASH ada 6 level (indeks), sedangkan untuk *streaming* tanpa DASH hanya 1 level (indeks) saja. Level (indeks) ini juga dapat dikatakan sebagai id video, seperti telah dijelaskan pada **Tabel 3.1** dan **Tabel 3.2**.



Gambar 5.3 Perbandingan Perubahan (*Switching*) Representasi Video *Streaming* DASH dan Tanpa DASH

Pembahasan pada sub-bab mengenai perbandingan perubahan representasi video yang digunakan untuk *streaming* DASH dan tanpa DASH adalah sebagai berikut:

a. Berdasarkan **Gambar 5.3** terlihat bahwa representasi video yang digunakan untuk *streaming* DASH lebih banyak dibandingkan *streaming* tanpa menggunakan DASH. Video *streaming* dengan menggunakan DASH dalam penelitian ini menggunakan 6 representasi video dan 3 representasi audio, sedangkan video *streaming* tanpa menggunakan DASH hanya menggunakan 1 representasi video dan audio saja. Saat terjadi perubahan *bandwidth*, layanan video *streaming* tanpa DASH tidak akan bisa berganti representasi video dengan kualitas lainnya, karena video yang disediakan oleh *server* untuk klien hanya 1 buah saja. Berbeda halnya dengan video *streaming* DASH. Ketika terjadi perubahan *bandwidth*, maka dengan cepat DASH akan mengganti representasi video pada *bandwidth* sebelumnya dengan representasi video yang baru, dimana representasi video yang baru tersebut juga sesuai dengan kondisi *bandwidth* pada saat itu. Representasi video yang digunakan oleh DASH memiliki beragam kualitas, ukuran dan *bitrate*, sedangkan jika tanpa DASH hanya terhenti pada 1 kualitas representasi video dan audio saja.

b. Berdasarkan **Gambar 5.3**, untuk representasi *bandwidth* 4191 Kbps, representasi video DASH yang digunakan adalah yang paling baik (kualitas pertama). Lalu terdapat perubahan *bandwidth* menjadi 3191 Kbps dan 2191 Kbps yang merubah kualitas dari representasi video ke kualitas kedua. Setelah itu *bandwidth* turun ke 2073 Kbps, 1073 Kbps, 900 Kbps dan 869 Kbps. Ketika terjadi penurunan 4 jenis *bandwidth* yang berbeda, maka representasi juga turun ke kualitas ketiga. *Bandwidth* terus menurun ke 700 Kbps dan 686 Kbps dan representasi video juga turun ke kualitas keempat. *Bandwidth* terus diturunkan agar dapat mencapai ke kualitas yang kelima, yaitu dengan representasi *bandwidth* 512 Kbps, dan 384 Kbps. Dan kualitas video yang terakhir (keenam) didapatkan ketika *bandwidth* diturunkan menjadi 264 Kbps, 256 Kbps, 100 Kbps dan 64 Kbps. Sedangkan untuk video *streaming* tanpa DASH, representasi video hanya menggunakan 1 buah video saja, yang memiliki kualitas paling baik dan tidak mempunyai video-video lain dengan kualitas yang lebih rendah. Jadi tidak ada perubahan representasi video untuk *streaming* tanpa DASH.

c. Hal yang ingin dipaparkan dari grafik yaitu DASH dapat selalu mengirimkan representasi video ke klien, meskipun saat itu kondisi jaringan (*bandwidth*) di sisi klien sangat terbatas. Hal ini ditunjukkan dengan banyaknya representasi video yang digunakan oleh DASH dibandingkan representasi video yang digunakan oleh *streaming* tanpa DASH, yang hanya 1 buah representasi video saja. Hal ini merupakan salah satu dampak positif yang didapatkan ketika menggunakan video *streaming* DASH, karena DASH dapat secara dinamis menyesuaikan kualitas video dengan kondisi ketersediaan *bandwidth* saat itu, dan dapat mengurangi jumlah *delay* yang terjadi di sisi klien. Sedangkan dampak negatif terjadi pada kebutuhan media penyimpanan yang cukup besar, karena DASH membutuhkan lebih dari 1 kualitas untuk file videonya.

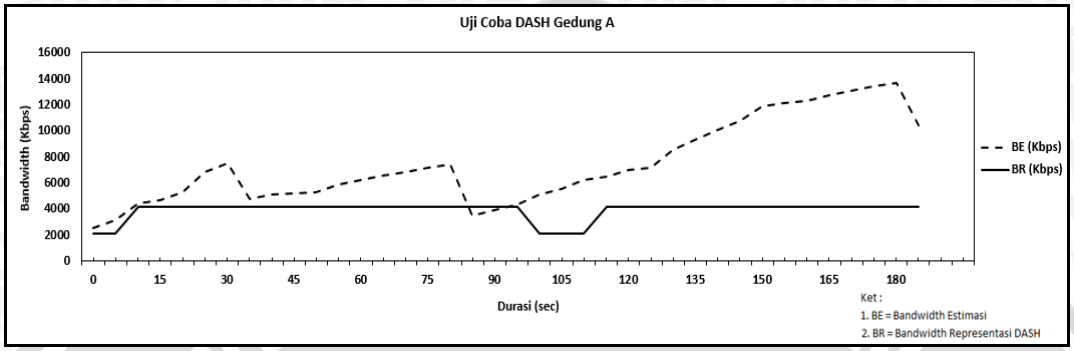
5.2 Uji Coba DASH Pada Jaringan Kampus

5.2.1 Gedung A

Setelah mendapatkan hasil pengujian DASH pada kondisi simulasi yang telah diuraikan sebelumnya, DASH juga perlu diuji pada kondisi lingkungan yang sebenarnya, yaitu jaringan kampus PTIIK-UB. Pengujian kali ini dilakukan untuk mengetahui kondisi jaringan pada gedung-gedung yang terdapat di kampus PTIIK-UB ketika mengakses konten DASH. Pengujian dilakukan dengan mengakses konten DASH dari masing-masing gedung yang berbeda, yaitu gedung A, gedung B, gedung C, gedung D dan gedung E. Hasil yang didapatkan ketika DASH diakses pada kondisi jaringan kampus pada gedung A terlihat pada **Gambar 5.4**. Garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth*, dan *second* (detik) untuk satuan durasi video.

Pengukuran untuk *bandwidth* estimasi (BE) didapatkan dengan cara melakukan pemantauan trafik jaringan dengan menggunakan NetTray, yang merupakan bagian dari aplikasi NetBalancer. Lalu untuk *bandwidth* representasi

(BR) didapatkan melalui pemantauan terhadap kualitas video yang diputar, selain itu juga didapatkan dari pesan *log* yang disediakan oleh aplikasi *player* DASH yang digunakan di dalam penelitian ini.



Gambar 5.4 Uji Coba DASH Ketika Diakses di Gedung A (PTIIK-UB)

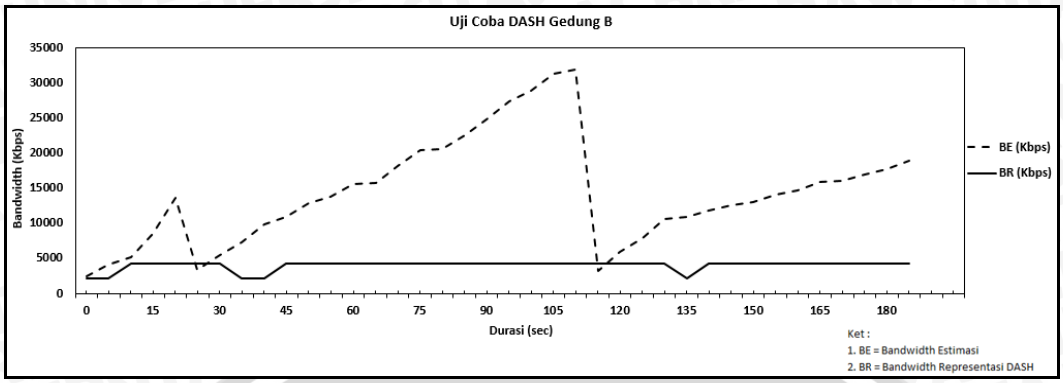
Pembahasan pada sub-bab mengenai pengaksesan konten DASH ketika dilakukan pada tempat yang berbeda, khususnya kali ini pada gedung A PTIIK-UB adalah sebagai berikut:

a. Berdasarkan **Gambar 5.4** di atas menunjukkan pengaksesan DASH yang dilakukan di gedung A PTIIK-UB. Awal mula video berjalan, *bandwidth* video yang digunakan adalah 2074 Kbps yang sesuai dengan kondisi jaringan yang saat melakukan pemutaran awal video menggunakan *bandwidth* 2500 Kbps. Setelah itu perlahan *bandwidth* pun meningkat hingga mencapai 10400 Kbps, dan secara otomatis akan merubah representasi video menjadi representasi paling baik (pertama) karena representasi video yang paling baik menggunakan representasi *bandwidth* lebih dari atau sama dengan 4191 Kbps. Sesekali pemutaran video terhenti beberapa saat karena memang terjadi penurunan pada kondisi *bandwidth* jaringan. Maka DASH secara dinamis langsung berubah juga. *Bandwidth* representasi (BR) yang digunakan oleh DASH tidak selalu mengikuti perubahan yang dialami *bandwidth* estimasi (BE) jaringan hingga mencapai maksimal, karena representasi video tertinggi yang dimiliki oleh DASH dan yang sudah tersimpan di dalam *web server* yaitu 4191 Kbps.

b. Perubahan *bandwidth* yang terjadi saat pertengahan pemutaran video dikarenakan oleh ketersediaan *bandwidth* yang pada saat itu selalu berubah-ubah (dinamis) dan keterbatasan *resource* dari komputer yang melakukan pengaksesan. Ketika terjadi perubahan *bandwidth*, tidak langsung berdampak pada representasi video yang sedang dijalankan. Video akan terus berjalan untuk menghabiskan nilai pada *bandwidth* sebelumnya, dan akan berubah kualitasnya ketika video yang di-*load* pada *bandwidth* sebelumnya telah habis. Pada gedung A rata-rata pemutaran video DASH yang digunakan adalah dengan representasi paling baik yang menggunakan representasi *bandwidth* 4191 Kbps dengan kualitas video pertama yang merupakan kualitas video paling baik juga.

5.2.2 Gedung B

Melihat bahwa masing-masing gedung memiliki *bandwidth* dan jumlah pengakses yang tidak sama satu dengan yang lainnya, maka **Gambar 5.5** menunjukkan pengujian kedua yang dilakukan pada gedung B. Sama halnya pada gedung A, pengujian dilakukan dengan mengakses DASH dimana *web server* DASH terletak di gedung B juga, tepatnya di Laboratorium Jaringan Komputer PTIIK-UB. Garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth*, dan *second* (detik) untuk satuan durasi video.



Gambar 5.5 Uji Coba DASH Ketika Diakses di Gedung B (PTIIK-UB)

Pembahasan pada sub-bab mengenai pengaksesan konten DASH ketika diakses pada gedung B PTIIK-UB adalah sebagai berikut:

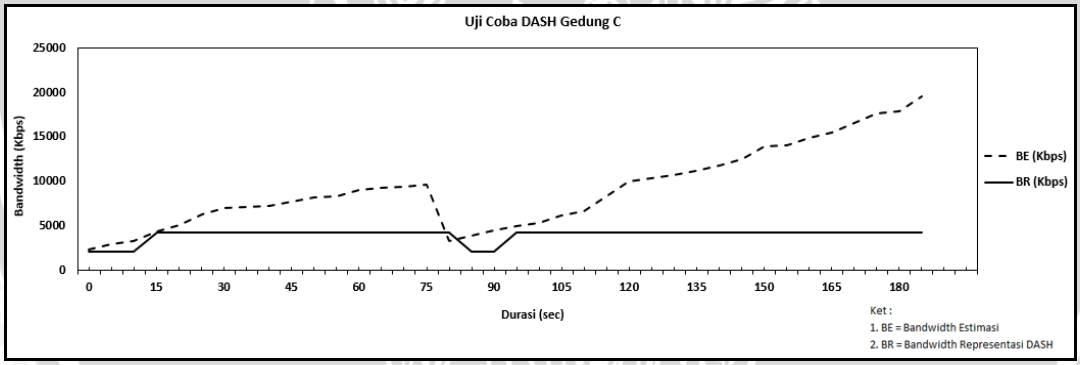
a. Berdasarkan **Gambar 5.5** menunjukkan bahwa *bandwidth* pertama kali yang digunakan untuk mengakses konten DASH adalah 2400 Kbps yang mengharuskan DASH untuk memutar video representasi kedua yang memerlukan *bandwidth* 2074 Kbps. Pada saat pemutaran video kali ini, terjadi 2 kali penurunan kondisi jaringan (*bandwidth*) yang mengharuskan DASH untuk menjalankan video dengan representasi kedua yang menggunakan *bandwidth* 2074 Kbps. Terjadinya perubahan *bandwidth* sebayak 2 kali disebabkan karena ketersediaan *bandwidth* yang pada saat itu selalu berubah-ubah (dinamis) dan adanya keterbatasan masing-masing *resource* yang dimiliki oleh komputer yang mengakses DASH. Pada gedung B, rata-rata representasi *bandwidth* DASH yang digunakan adalah 4191 Kbps, dengan video yang diputar memiliki kualitas pertama yaitu yang paling baik.

b. Pada gedung B, penggunaan video dengan representasi 4191 Kbps cukup lama, karena *bandwidth* estimasi (BE) jaringan saat itu selalu lebih dari sama dengan 4191 Kbps. Maka mau tidak mau, *server* akan memberikan video dengan representasi tertingginya, yang pada penelitian kali ini kualitas video teringgi memiliki representasi *bandwidth* 4191 Kbps. Jadi, ketika BE saat itu mencapai 30000 Kbps, maka klien akan tetap memutar video dengan representasi *bandwidth* 4191 Kbps, karena representasi *bandwidth* 4191 Kbps

merupakan representasi *bandwidth* tertinggi yang disimpan di dalam *web server* DASH.

5.2.3 Gedung C

Selain dilakukan pengaksesan pada gedung A dan gedung B, pengujian dilanjutkan ke gedung C PTIIK-UB. Tidak jauh berbeda dengan gedung A, pada gedung C ini terdapat sekali lonjakan *bandwidth* pada pertengahan pemutaran video. Garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth*, dan *second* (detik) untuk satuan durasi video. Hasilnya dapat dilihat pada **Gambar 5.6**.



Gambar 5.6 Uji Coba DASH Ketika Diakses di Gedung C (PTIIK-UB)

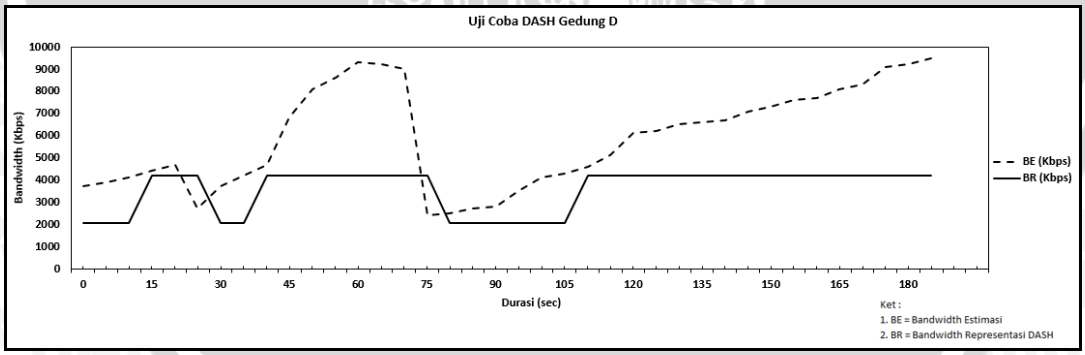
Pembahasan pada sub-bab mengenai pengaksesan konten DASH ketika diakses pada gedung C PTIIK-UB adalah sebagai berikut:

a. Berdasarkan **Gambar 5.6** menunjukkan bahwa *bandwidth* pertama kali yang digunakan adalah 2300 Kbps. Dengan *bandwidth* estimasi (BE) sebesar 2300 Kbps, maka representasi video yang harus dijalankan adalah representasi video kedua dengan representasi *bandwidth* (BR) 2074 Kbps. Pada saat pengaksesan di gedung C ini, terjadi sekali penurunan kondisi *bandwidth* saat detik ke-80 pemutaran video. Penurunan ini terjadi karena ketersediaan

bandwidth yang pada saat itu selalu berubah-ubah (dinamis) dan perbedaan *resource* yang dimiliki oleh komputer yang mengakses konten DASH ini. Rata-rata video yang diputar pada gedung C ini adalah video pertama dengan representasi *bandwidth* 4191 Kbps. Meskipun BE jaringan saat itu melebihi 4191 Kbps, maka DASH akan tetap menjalankan dan mengirimkan representasi video tertingginya (4191 Kbps) ke klien.

5.2.4 Gedung D

Jumlah pengguna Internet di kampus PTIIK-UB terutama di gedung D yang cukup banyak, maka berdampak pula pada saaat dilakukannya pengaksesan DASH. Kali ini terjadi 2 kali lonjakan pada awal dan pertengahan pemutaran video. Garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth*, dan *second* (detik) untuk satuan durasi video. Hasilnya dapat dilihat pada **Gambar 5.7**.



Gambar 5.7 Uji Coba DASH Ketika Diakses di Gedung D (PTIIK-UB)

Pembahasan pada sub-bab mengenai pengaksesan konten DASH ketika diakses pada gedung D PTIIK-UB adalah sebagai berikut:

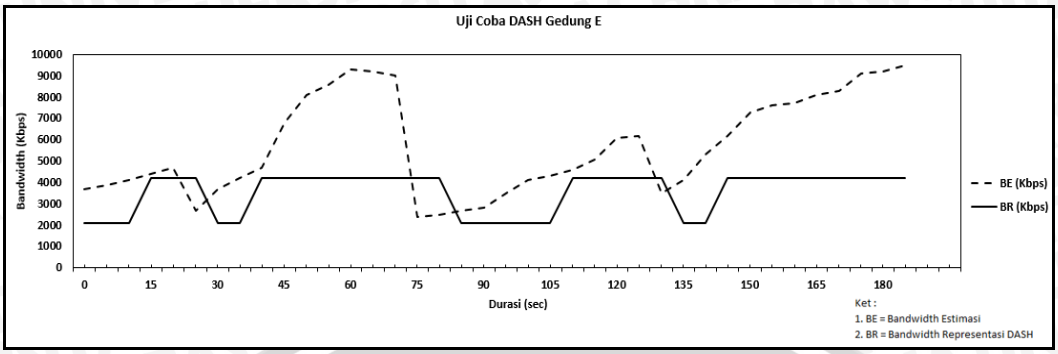
a. Berdasarkan **Gambar 5.7** menunjukkan bahwa pengaksesan DASH yang dilakukan pada gedung D dengan besar *bandwidth* estimasi (BE) awal

pemutaran video yang didapatkan adalah 3700 Kbps. Terjadi 2 kali penurunan kondisi jaringan di gedung D kali ini. Hal ini disebabkan karena ketersediaan *bandwidth* yang pada saat itu selalu berubah-ubah (dinamis) dan perbedaan kemampuan atau perbedaan spesifikasi komputer ketika mengakses konten DASH. Penurunan kondisi jaringan ini yang membuat klien harus mau tidak mau menerima video kualitas kedua, dimana kualitasnya lebih rendah dibanding video pertama. Durasi ketika menjalankan video dengan representasi *bandwidth* (BR) 2074 Kbps cukup lama, yaitu 5 menit di awal pemutaran video dan 20 menit di pertengahan video. Rata-rata *bandwidth* DASH yang digunakan pada gedung D juga sama seperti gedung-gedung sebelumnya yaitu 2074 Kbps dan 4191 Kbps, dengan 4191 Kbps yang lebih dominan.

b. Pada gedung D, BE jaringan dapat mencapai lebih dari 9000 Kbps, tetapi mau tidak mau klien juga akan menerima video dengan representasi *bandwidth* 4191 Kbps, yang dalam hal ini representasi *bandwidth* 4191 Kbps merupakan representasi video tertinggi sekaligus dengan kualitas yang terbaik pula. Hal ini dikarenakan *web server* DASH hanya menyimpan 6 buah representasi video yang dapat diakses oleh klien sesuai dengan kondisi jaringan klien saat itu, dengan kualitas yang tertinggi yaitu ditunjukkan oleh representasi *bandwidth* 4191 Kbps.

5.2.5 Gedung E

Hasil terakhir yang didapatkan dari pengujian DASH pada kondisi jaringan kampus terlihat pada **Gambar 5.8** yaitu pada gedung E kampus PTIIK-UB. Terlihat pada **Gambar 5.8** bahwa garis berbentuk lurus menunjukkan *bandwidth* representasi atau di dalam grafik disingkat menjadi BR, dimana merupakan nilai *bandwidth* yang digunakan oleh DASH, sedangkan garis berbentuk patah-patah menunjukkan *bandwidth* estimasi atau disingkat BE, dimana merupakan nilai *bandwidth* yang digunakan jaringan saat itu. Semuanya menggunakan satuan Kbps untuk *bandwidth,* dan *second* (detik) untuk satuan durasi video.



Gambar 5.8 Uji Coba DASH Ketika Diakses di Gedung E (PTIIK-UB)

Pembahasan pada sub-bab mengenai pengaksesan konten DASH ketika diakses pada gedung E PTIIK-UB adalah sebagai berikut:

a. Berdasarkan **Gambar 5.8** terlihat bahwa *bandwidth* estimasi (BE) pertama yang dibutuhkan untuk memutar konten DASH adalah 3700 Kbps, dan durasi pemutaran video dengan kualitas kedua ini sekitar 15 menit hingga video akhirnya benar-benar menggunakan kualitas video yang pertama. Terjadi 3 kali penurunan kondisi jaringan pada saat pengaksesan konten, yaitu pada detik ke 30, 85 dan 135. Penurunan ini terjadi karena ketersediaan *bandwidth* yang pada saat itu selalu berubah-ubah (dinamis), jumlah pengguna yang cukup padat dan keterbatasan *resource* yang dimiliki oleh komputer yang berbeda. Namun, rata-rata video yang sering digunakan pada saat pengaksesan di gedung E adalah video pertama dengan kualitas paling baik dan representasi *bandwidth* 4191 Kbps.
b. Dari semua gedung yang dimasukkan ke dalam pengujian, hampir semuanya menggunakan representasi *bandwidth* untuk video DASH adalah 2074 Kbps dan 4191 Kbps, dengan video yang paling dominan diputar dari awal hingga akhir adalah 4191 Kbps. Nilai 4191 Kbps sering muncul, dikarenakan nilai 4191 Kbps ini sudah diatur sebagai batas maksimum representasi *bandwidth* untuk video dengan kualitas terbaik (tertinggi) di dalam *web server*.

BAB VI
PENUTUP

Pada bab ini berisi kesimpulan dari hasil analisis kinerja layanan video *streaming* berbasis DASH dan juga saran untuk pengembangan layanan video *streaming* berbasis DASH pada kondisi jaringan lingkungan yang beragam ke depannya.

6.1 Kesimpulan

Dari hasil pengujian dan analisis kinerja video *streaming* berbasis *Dynamic Adaptive Streaming over HTTP* (DASH) pada area jaringan kampus, maka dapat disimpulkan bahwa:

1. Pembangunan lingkungan berbasis DASH membutuhkan *web server* sebagai media penyimpanan konten DASH berupa file MPD dan file video dengan beragam format *codecs* yang bersesuaian, serta DASH *player* di sisi pengguna untuk dapat beradaptasi dengan kondisi jaringan yang dinamis (fluktuatif). Peran DASH *player* pada sisi pengguna sangat besar karena dapat menginterpretasikan secara mandiri berdasarkan ketersediaan konten di dalam *web server*, yang disesuaikan dengan kondisi akses jaringan (*bandwidth*) pengguna saat itu.
2. Kinerja DASH yang didapatkan dari hasil pengujian yang sudah dilakukan adalah:
   a. Berdasarkan pada jumlah rata-rata waktu *delay* yang lebih kecil di sisi pengguna, maka dapat dikatakan bahwa layanan video *streaming* menggunakan DASH lebih baik daripada tanpa menggunakan DASH, yaitu sebesar 4 detik.
   b. *Bandwidth* estimasi pada area jaringan kampus selalu dominan tinggi, dengan ditunjukkan pada klien yang selalu mendapatkan video dengan kualitas terbaik (id pertama).

6.2 Saran

Saran yang dapat disampaikan penulis untuk pengembangan penelitian ini lebih lanjut adalah sebagai berikut:

1. Perlu dilakukan penelitian lebih lanjut tentang layanan video *streaming* berbasis DASH pada perangkat yang lebih beragam.
2. Perlu dilakukan penelitian lebih lanjut tentang pengembangan DASH, karena sampai saat ini dataset video yang dipublikasikan dan dapat diuji masih terbatas.
3. Perlu dilakukan pengembangan lebih lanjut terhadap media pemutaran konten DASH, baik *player* yang berbasis *dekstop* maupun *web*.

DAFTAR PUSTAKA

MULLER, C., et al. 2012. An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. *Proceedings of the 4th Workshop on Mobile Video*. ACM.

SODAGAR, I. 2011. Industry and Standards: The MPEG-DASH Standard for Multimedia Streaming Over the Internet. IEEE *Computer Society*.

LEDERER, S., et al. 2012. Dynamic Adaptive Streaming over HTTP Dataset. *Proceedings of the 3rd Multimedia Systems Conference*. ACM.

RAINER, B., et al. 2012. A Seamless *Web* Integration of Adaptive HTTP Streaming. *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE.

KUROSE, J. F., et al. 2001. Computer Networking: A Top-Down Approach Featuring the Internet. Vol. 2. *Reading*: Addison-Wesley, 587-601.

MULLER C., et al. 2011. A Test-bed for the Dynamic Adaptive Streaming over HTTP Featuring Session Mobility. *Proceedings of the second annual ACM conference on Multimedia systems*. ACM.

MICHALOS, M., et al. 2012. Dynamic Adaptive Streaming over HTTP. *Journal of Engineering Science and Technology Review* 5.2: 30-34.

THANG, T, C., et al. 2012. Adaptive Streaming of Audiovisual Content Using MPEG DASH. *Consumer Electronics, IEEE Transactions on* 58.1: 78-85.

WANG, Z., et al. 2003. Studying Streaming Video Quality: From An Application Point of View. *Proceedings of the eleventh ACM international conference on Multimedia*. ACM.

MILLER, K., et al. 2012. Adaptation Algorithm for Adaptive Streaming over HTTP. *Packet Video Workshop (PV), 2012 19th International*. IEEE.

GHOBADI, M., et al. 2012. Trickle: Rate Limiting YouTube Video Streaming. *USENIX Annual Technical Conference*.

STOCKHAMMER, T. 2011. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. *Proceedings of the second annual ACM conference on Multimedia systems*. ACM.

ZENHADI. 2013. Video Streaming dengan HTML5.

BASUYOGA, G. 2011. Pengertian Istilah-istilah dalam Internet. DESAIN, FAKULTAS SENI RUPA.

KUSUMANDARI, A. 2010. Rancang Bangun Sistem Set Top Box untuk Internet.

WARDANA, R, S. 2014. Analisis dan Implementasi Distributed Cloud Storage *Server* dengan Jaringan Peer-to-Peer.

SANCHEZ, Y., et al. 2011. Scalable Video Coding based DASH for Efficient Usage of Network Resources. *Third W3C Web and TV Workshop*.

LEDERER, S., et al. 2013. Distributed DASH Dataset. *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM.

MUELLER, C., et al. 2013. Demo Paper: Libdash-An Open Source Software Library for the MPEG-DASH Standard. *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on. IEEE*.

HARTANTO. 2009. Implementasi *Real-Time Streaming Protocol* untuk Aplikasi Radio Internet.

ABROR, A, A. 2010. Rancang Bangun dan Analisa QOS Audio dan Video *Streaming* Pada Jaringan MPLS VPN.

GOOGLEAPIS. 2014. DASH Dataset. http://yt-dash-mse-test.commondatastorage.googleapis.com/media/car-20120827-manifest.mpd, diakses tanggal 7 Mei 2014.

CONCOLATO, C. 2014. *Web*vtt, Mp4 Files, DASH and GPAC. http://concolato.wp.mines-telecom.fr/2013/07/28/*web*vtt-mp4-files-dash-and-gpac/, diakses tanggal 15 Mei 2014.

GALLIGAN, F. 2014. Instructions to Playback Adaptive *Web*M using DASH. http://wiki.*web*mproject.org/adaptive-streaming/instructions-to-playback-adaptive-*web*m-using-dash, diakses tanggal 4 April 2014.

SOSRO, N. 2014. Re-Posting Membuat DASH Player Sederhana. http://www.4shared.com/rar/bFGQxii-ce/dash.html, diakses tanggal 5 Mei 2014.

TOM. 2014. Download NetBalancer 7.0 For Windows. http://www.tomsguide.com/us/download/NetBalancer,0301-46134.html, diakses tanggal 20 Juni 2014.

ITEC, 2014. DASH. http://www.itec.uni-klu.ac.at/dash, diakses tanggal 2 Februari 2014.

LAMPIRAN

Isi File MPD *Sample* (`car.mpd`)

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S"
minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760"
codecs="avc1.640028" height="1080" id="1"
mimeType="video/mp4" width="1920">
        <BaseURL>car-20120827-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="2073921"
codecs="avc1.4d401f" height="720" id="2"
mimeType="video/mp4" width="1280">
        <BaseURL>car-20120827-88.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460"
codecs="avc1.4d401e" height="480" id="3"
mimeType="video/mp4" width="854">
        <BaseURL>car-20120827-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
```

```
      </Representation>
      <Representation bandwidth="686521"
codecs="avc1.4d401e" height="360" id="4"
mimeType="video/mp4" width="640">
        <BaseURL>car-20120827-86.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835"
codecs="avc1.4d4015" height="240" id="5"
mimeType="video/mp4" width="426">
        <BaseURL>car-20120827-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="100000"
codecs="avc1.4d4015" height="144" id="5"
mimeType="video/mp4" width="256">
        <BaseURL>car-20120827-160.mp4</BaseURL>
        <SegmentBase indexRange="671-1146">
          <Initialization range="0-670" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2"
sampleRate="44100">
        <BaseURL>car-20120827-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
```

```
        </SegmentBase>
      </Representation>
      <Representation bandwidth="255236" codecs="mp4a.40.2"
id="7" mimeType="audio/mp4" numChannels="2"
sampleRate="44100">
        <BaseURL>car-20120827-8d.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
id="8" mimeType="audio/mp4" numChannels="1"
sampleRate="22050">
        <BaseURL>car-20120827-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

LAMPIRAN 2

Isi File MPD Hasil *Generate* Menggunakan MP4Box GPAC (`mp4client.mpd`)

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.1-DEV-
rev5261  on 2014-06-03T07:35:14Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011"
minBufferTime="PT1.500000S" type="static"
mediaPresentationDuration="PT0H4M5.30S"
profiles="urn:mpeg:dash:profile:full:2011">
 <ProgramInformation
moreInformationURL="http://gpac.sourceforge.net">
  <Title>titanium_track3_dash.mpd generated by
GPAC</Title>
 </ProgramInformation>

 <Period id="" duration="PT0H4M5.30S">
  <AdaptationSet segmentAlignment="true"
bitstreamSwitching="true" maxWidth="854" maxHeight="480"
maxFrameRate="25" par="854:480" lang="und">
   <Role schemeIdUri="urn:gpac:dash:role:2013"
value="video"/>
   <SegmentList>
    <Initialization sourceURL="high.mp4"/>
   </SegmentList>
   <Representation id="1" mimeType="video/mp4"
codecs="avc3.42c033" width="854" height="480"
frameRate="25" sar="1:1" startWithSAP="0"
bandwidth="622064">
    <BaseURL>video/low.mp4</BaseURL>
    <SegmentList timescale="25000" duration="250000">
     <SegmentURL mediaRange="964-366845" indexRange="964-
1008"/>
```

```
      <SegmentURL mediaRange="366846-744132"
indexRange="366846-366890"/>
      <SegmentURL mediaRange="744133-1183698"
indexRange="744133-744177"/>
      <SegmentURL mediaRange="1183699-1601386"
indexRange="1183699-1183743"/>
      <SegmentURL mediaRange="1601387-2034986"
indexRange="1601387-1601431"/>
      <SegmentURL mediaRange="2034987-2657473"
indexRange="2034987-2035031"/>
      <SegmentURL mediaRange="2657474-3462465"
indexRange="2657474-2657518"/>
      <SegmentURL mediaRange="3462466-4527530"
indexRange="3462466-3462510"/>
      <SegmentURL mediaRange="4527531-5745994"
indexRange="4527531-4527575"/>
      <SegmentURL mediaRange="5745995-7129910"
indexRange="5745995-5746039"/>
      <SegmentURL mediaRange="7129911-8428571"
indexRange="7129911-7129955"/>
      <SegmentURL mediaRange="8428572-9034411"
indexRange="8428572-8428616"/>
      <SegmentURL mediaRange="9034412-9537527"
indexRange="9034412-9034456"/>
      <SegmentURL mediaRange="9537528-9940117"
indexRange="9537528-9537572"/>
      <SegmentURL mediaRange="9940118-10391449"
indexRange="9940118-9940162"/>
      <SegmentURL mediaRange="10391450-10925989"
indexRange="10391450-10391494"/>
      <SegmentURL mediaRange="10925990-11483561"
indexRange="10925990-10926034"/>
      <SegmentURL mediaRange="11483562-12903144"
indexRange="11483562-11483606"/>
      <SegmentURL mediaRange="12903145-14192524"
```

```
indexRange="12903145-12903189"/>
      <SegmentURL mediaRange="14192525-15401066"
indexRange="14192525-14192569"/>
      <SegmentURL mediaRange="15401067-16648151"
indexRange="15401067-15401111"/>
      <SegmentURL mediaRange="16648152-17593701"
indexRange="16648152-16648196"/>
      <SegmentURL mediaRange="17593702-18428331"
indexRange="17593702-17593746"/>
      <SegmentURL mediaRange="18428332-19005196"
indexRange="18428332-18428376"/>
     <SegmentURL mediaRange="19005197-19066263"
indexRange="19005197-19005241"/>
    </SegmentList>
   </Representation>
   <Representation id="2" mimeType="video/mp4"
codecs="avc3.42c033" width="854" height="480"
frameRate="25" sar="1:1" startWithSAP="0"
bandwidth="674298">
    <BaseURL>video/normal.mp4</BaseURL>
    <SegmentList timescale="25000" duration="250000">
      <SegmentURL mediaRange="964-651958" indexRange="964-
1008"/>
      <SegmentURL mediaRange="651959-1351602"
indexRange="651959-652003"/>
      <SegmentURL mediaRange="1351603-2078881"
indexRange="1351603-1351647"/>
      <SegmentURL mediaRange="2078882-2706109"
indexRange="2078882-2078926"/>
      <SegmentURL mediaRange="2706110-3326490"
indexRange="2706110-2706154"/>
      <SegmentURL mediaRange="3326491-4067824"
indexRange="3326491-3326535"/>
      <SegmentURL mediaRange="4067825-4889759"
indexRange="4067825-4067869"/>
```

```
      <SegmentURL mediaRange="4889760-5969877"
indexRange="4889760-4889804"/>
      <SegmentURL mediaRange="5969878-7188525"
indexRange="5969878-5969922"/>
      <SegmentURL mediaRange="7188526-8572441"
indexRange="7188526-7188570"/>
      <SegmentURL mediaRange="8572442-9871102"
indexRange="8572442-8572486"/>
      <SegmentURL mediaRange="9871103-10477586"
indexRange="9871103-9871147"/>
      <SegmentURL mediaRange="10477587-10993032"
indexRange="10477587-10477631"/>
      <SegmentURL mediaRange="10993033-11443369"
indexRange="10993033-10993077"/>
      <SegmentURL mediaRange="11443370-11914980"
indexRange="11443370-11443414"/>
      <SegmentURL mediaRange="11914981-12488765"
indexRange="11914981-11915025"/>
      <SegmentURL mediaRange="12488766-13054952"
indexRange="12488766-12488810"/>
      <SegmentURL mediaRange="13054953-14474535"
indexRange="13054953-13054997"/>
      <SegmentURL mediaRange="14474536-15763915"
indexRange="14474536-14474580"/>
      <SegmentURL mediaRange="15763916-16972457"
indexRange="15763916-15763960"/>
      <SegmentURL mediaRange="16972458-18219542"
indexRange="16972458-16972502"/>
      <SegmentURL mediaRange="18219543-19180622"
indexRange="18219543-18219587"/>
      <SegmentURL mediaRange="19180623-20015252"
indexRange="19180623-19180667"/>
      <SegmentURL mediaRange="20015253-20606168"
indexRange="20015253-20015297"/>
     <SegmentURL mediaRange="20606169-20667235"
```

```
indexRange="20606169-20606213"/>
    </SegmentList>
   </Representation>
   <Representation id="3" mimeType="video/mp4"
codecs="avc3.42c033" width="854" height="480"
frameRate="25" sar="1:1" startWithSAP="0"
bandwidth="774556">
    <BaseURL>video/high.mp4</BaseURL>
    <SegmentList timescale="25000" duration="250000">
      <SegmentURL mediaRange="964-933838" indexRange="964-
1008"/>
      <SegmentURL mediaRange="933839-1963265"
indexRange="933839-933883"/>
      <SegmentURL mediaRange="1963266-2983861"
indexRange="1963266-1963310"/>
      <SegmentURL mediaRange="2983862-3849139"
indexRange="2983862-2983906"/>
      <SegmentURL mediaRange="3849140-4711133"
indexRange="3849140-3849184"/>
      <SegmentURL mediaRange="4711134-5749837"
indexRange="4711134-4711178"/>
      <SegmentURL mediaRange="5749838-6849953"
indexRange="5749838-5749882"/>
      <SegmentURL mediaRange="6849954-8095484"
indexRange="6849954-6849998"/>
      <SegmentURL mediaRange="8095485-9350681"
indexRange="8095485-8095529"/>
      <SegmentURL mediaRange="9350682-10741383"
indexRange="9350682-9350726"/>
      <SegmentURL mediaRange="10741384-12040044"
indexRange="10741384-10741428"/>
      <SegmentURL mediaRange="12040045-12696841"
indexRange="12040045-12040089"/>
      <SegmentURL mediaRange="12696842-13336316"
indexRange="12696842-12696886"/>
```

```
      <SegmentURL mediaRange="13336317-13916128"
indexRange="13336317-13336361"/>
      <SegmentURL mediaRange="13916129-14567581"
indexRange="13916129-13916173"/>
      <SegmentURL mediaRange="14567582-15298368"
indexRange="14567582-14567626"/>
      <SegmentURL mediaRange="15298369-16007220"
indexRange="15298369-15298413"/>
      <SegmentURL mediaRange="16007221-17439914"
indexRange="16007221-16007265"/>
      <SegmentURL mediaRange="17439915-18730982"
indexRange="17439915-17439959"/>
      <SegmentURL mediaRange="18730983-19941767"
indexRange="18730983-18731027"/>
      <SegmentURL mediaRange="19941768-21196797"
indexRange="19941768-19941812"/>
      <SegmentURL mediaRange="21196798-22219993"
indexRange="21196798-21196842"/>
      <SegmentURL mediaRange="22219994-23072154"
indexRange="22219994-22220038"/>
      <SegmentURL mediaRange="23072155-23679092"
indexRange="23072155-23072199"/>
     <SegmentURL mediaRange="23679093-23740159"
indexRange="23679093-23679137"/>
    </SegmentList>
   </Representation>
  </AdaptationSet>
 </Period>
</MPD>
```