Jurnal TIME , Vol. III No 1 : 37-44, 2014
ISSN : 2337 – 3601

Aplikasi E-learning Dengan Menggunakan IP-TV Berbasis Opencaster

Robet, M.Kom

Sekolah Tinggi Manajemen Informatika Dan Komputer TIME

robertdetime@gmail.com

Abstrak

Perkembangan teknologi jaringan informasi yang berkembang pesat saat ini memungkinkan konten digital untuk disampaikan lebih cepat dari satu tempat ke tempat lainnya. Dalam dunia pendidikan, teknologi ini kemudian dimanfaatkan untuk e-learning. E-learning yang diterapkan adalah penggunaan teknologi video-audio streaming pada protocol TCP/IP yang disebut IP-TV. Dalam penerapan e-learning berbasis IP-TV ini terdapat multi konten video-audio streaming yang dibuat menggunakan aplikasi Opencaster yang kemudian dapat dipancarkan melalui sebuah jaringan LAN/WAN, sehingga pengguna dapat melakukan pemilihan terhadap konten apa yang diinginkan.

Kata Kunci :E-learning, IP-TV, Opencaster

1. Pendahuluan

1.1 Latar belakang Masalah

E-learning merupakan suatu media yang menggunakan sarana dan prasarana elektronik untuk proses pembelajaran. Saat ini, dengan berkembangnya teknologi *multimedia* sangat membantu dalam dunia pendidikan. Banyak sekali *e-learning* yang diterapkan secara *online* dan berbasis *web*, dan pembelajaran menggunakan *e-learning* ini sudah berkembang sejak disahkannya Undang-Undang Sistem Pendidikan Nasional tentang pembelajaran boleh menggunakan alat bantu elektronik.

Tetapi sistem pembelajaran yang ingin dikaji lebih dalam yaitu *e-learning* yang berbasis *video*, dimana proses *e-learning* dijalankan menggunakan teknologi *video streaming* pada protocol *TCP/IP* yang disebut *IP-TV*. *IP-TV* adalah suatu pengembangan baru dalam *software* komunikasi *client-server* yang mem-*broadcast video* yang berkualitas tinggi (secara *real time full motion video* secara simultan) ke *user window* melalui jaringan data yang ada sekarang.

Di sisi lain *IP-TV* dapat memberikan layanan yang ekonomis namun dengan tidak mengorbankan kualitas layanan, serta lebih efisien karena tidak perlu membayar instruktur, biaya *print* materi relatif lebih sedikit, tidak perlu menyewa ruang seminar khusus karena dapat diakses oleh orang yang ada di setiap meja selama terkoneksi dalam satu *LAN/WAN*).

Dalam pembuatan konten *multivideo-audio streaming* akan digunakan *software open source Opencaster* yang berbasis pada sistem operasi *Linux*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka ada beberapa permasalahan yang harus dirumuskan yaitu :

1. Bagaimanakah mengaplikasikan *e-learning* dalam media *IP-TV?*
2. Bagaimanakah membuat konten *multivideo* dan *audio streaming* dengan menggunakan perangkat lunak sumber terbuka (*foss) opencaster*?

1.3 Batasan Masalah

Agar penelitian ini tidak menghasilkan pemahaman yang bias, maka perlu ditegaskan ruang lingkup pembahasannya. Sesuai dengan perumusan masalah di atas, maka batasan pembahasan pada aplikasi *IP-TV* yang berbasis *Opencaster* yang bisa diterapkan dalam aplikasi e-*learning* sebagai media pembelajaran maupun pelatihan yang dilakukan secara *broadcast*.

1.4 Tujuan Penelitian

Dengan melakukan analisis dan penerapan *IP-TV* dengan *Opencaster*, maka tujuan yang ingin dicapai adalah mengaplikasikan *e-learning* dengan menggunakan multi konten berbasis *video-audio streaming* sehingga pengguna dapat melakukan pemilihan terhadap konten yang diinginkan.

2. Landasan Teori

2.1 Pengertian *E-learning*

E-learning merupakan suatu perangkat belajar mengajar yang memungkinkan tersampaikannya bahan ajar ke siswa dengan menggunakan media *internet*, *intranet* atau media jaringan komputer lain. Dulu mungkin kita berpikir bahwa belajar harus dalam ruang kelas. Dengan kondisi dimana guru atau dosen mengajar di depan kelas sambil sesekali menulis materi pelajaran di papan tulis. Beberapa puluh tahun yang lalu pun juga telah dikenal pendidikan jarak jauh. Walaupun dengan mekanisme yang dibilang cukup "sederhana" untuk ukuran sekarang, tetapi saat itu model tersebut sudah dapat membantu orang-orang yang butuh belajar atau mengenyam pendidikan tanpa terhalang kendala geografis. Memang kita akui, sejak ditemukannya teknologi *internet*, hampir segalanya menjadi mungkin. Kini dapat belajar tak hanya *anywhere*, tapi sekaligus *anytime* dengan fasilitas *e-learning* yang ada.

2.2 Komponen *E-learning*

Adapun komponen-komponen yang membentuk *e-learning* adalah

1. Infrastruktur *e-learning*, dapat berupa *personal computer*, jaringan komputer, *internet* dan perlengkapan *multimedia*. Termasuk di dalamnya peralatan *teleconference* apabila kita memberikan layanan *synchronous learning* melalui *teleconference*.
2. Sistem dan aplikasi *e-learning*, **s**istem perangkat lunak yang mem-virtualisasi proses belajar mengajar konvensional. Bagaimana manajemen kelas, pembuatan materi atau konten, forum diskusi, sistem penilaian, sistem ujian *online* dan segala fitur yang berhubungan dengan manajemen proses belajar mengajar. Sistem perangkat lunak tersebut sering disebut dengan *Learning Management System (LMS)*. *LMS* banyak yang *opensource* sehingga bisa kita manfaatkan dengan mudah dan murah untuk dibangun di sekolah dan universitas.
3. Konten *e-learning***,** konten dan bahan ajar yang ada pada *e-learning* sistem. Konten dan bahan ajar ini bisa dalam bentuk *Multimedia-based Content* (konten berbentuk *multimedia* interaktif) atau *Text-based Content* (konten berbentuk teks). Biasa disimpan dalam *LMS* sehingga dapat dijalankan oleh siswa kapanpun dan dimanapun.

2.3 Streaming

Streaming adalah sebuah teknologi untuk memainkan file *video* atau *audio* secara langsung ataupun dengan *pre-recorded* dari sebuah *server*. Dengan kata lain, file *video* atau *audio* yang terletak pada sebuah *server* dapat secara langsung dijalankan pada komputer *client* sesaat setelah ada permintaan dari *user* sehingga proses *download* file *video* atau *audio* yang menghabiskan waktu cukup lama dapat dihindari.

Saat file *video* atau *audio* di-*stream* maka akan terbentuk sebuah *buffer* di komputer *client* dan data tersebut akan mulai di-*download* ke dalam *buffer* yang telah terbentuk pada *client*. Dalam waktu sepersekian detik, *buffer* telah terisi penuh dan secara otomatis file akan dijalankan oleh sistem. Sistem akan membaca informasi dari *buffer* sambil tetap melakukan proses *download* file sehingga proses *streaming* tetap berlangsung ke *client*. *Delay* waktu sesaat sebelum file *video* atau *audio* dijalankan berkisar antara 10-30 detik.

2.4 Masalah Dasar Dalam *Video Streaming*

Masalah dasar dalam *video streaming*, khususnya untuk implementasi pada jaringan *internet* yang bersifat *global*, adalah *bandwidth*, *delay jitter*, *loss rate*. Ketersediaan *bandwidth* antar dua titik pada jaringan *internet* secara umum tidak diketahui. Jika pengirim mengirimkan data lebih cepat dibandingkan dengan *bandwidth* yang tersedia maka akan terjadi kongesti pada jaringan, paket hilang, dan kualitas *video* akan buruk. Jika pengirim mengirimkan paket data *video* lebih lambat dari *bandwidth* yang tersedia, maka kualitas *video* yang sampai ke penerima juga kurang optimal.

Salah satu ide untuk mengatasi masalah *bandwidth* adalah dengan mengestimasi *bandwidth* kanal yang tersedia kemudian mencocokkannya dengan *bit rate video* yang akan ditransmisikan. Masalah kedua pada *video streaming* adalah *delay jitter*, di mana paket-paket yang ditransmisikan ke *client* memiliki *delay* yang bersifat fluktuatif. Variasi dari *delay* paket ini disebut dengan *delay jitter*. *Delay jitter* ini menjadi masalah karena penerima harus men-*decode* dan menampilkan *frame-frame* pada *rate* yang konstan, dan akumulasi dari keterlambatan *frame* akan menyulitkan untuk rekonstruksi *video* yang diterima.

Masalah ketiga dalam *video strea ming* adalah *loss rate*. *Loss rate* berbeda-beda untuk jaringan *fixed* dan pada jaringan *wireless*. Pada jaringan *fixed*, *loss rate* disebabkan oleh paket data yang hilang. Sedangkan pada jaringan *wireless*, *loss rate* dapat disebabkan oleh *bit error* dan *burst error*. *Loss rate* ini dapat menimbulkan penurunan kualitas *video* hasil rekonstruksi. Untuk mengatasi *loss rate* ini, sistem *video streaming* dapat didesain dengan fasilitas *error control*.

2.5 Protokol Protokol *Streaming*

Protokol-protokol *streaming* terdiri dari

1. *IP*: *internet protocol*, protokol terbawah yang digunakan untuk mentransmisikan sinyal informasi pada jaringan *internet*.
2. *TCP*: *transport control protocol*, protokol ini berada di atas lapisan *internet* yang berfungsi untuk mengatasi kongesti dan bersifat *reliable*.
3. *RTP*: *real time transport protocol*, lapisan yang berada di atas *IP* dan mendukung pengiriman data secara *real time*.
4. *HTTP*: *hypertext transport protocol*, protokol ini digunakan untuk transmisi informasi melalui *web page*. Protokol ini berada di atas lapisan*TCP*.
5. *RTSP*: *real time transport streaming protocol*, protokol ini berada di atas lapisan *RTP* yang digunakan untuk media *streaming*.

2.6 *VLC CLIENT-SERVER*

Video LAN adalah sebuah *sofware* aplikasi yang diperuntukkan bagi *streaming video*. Pembuatan *sofware* ini diprakarsai oleh mahasiswa *Ecole Centrale* Paris dan kemudian dikembangkan oleh *developer* dari seluruh dunia. *Sofware* ini berlisensi *GNU General Public License* (*GPL*).

Video Streaming dapat menggunakan dua macam *software Video LAN*, yaitu :

1. *Video LAN Server* (*VLS*), dapat digunakan *untuk streaming* file dalam format *MPEG*-1, *MPEG*-2 dan *MPEG*-4, *DVD*, *TV Channel*, dan lain lain. *VLS* ini hanya dapat bertindak sebagai *server* bukan sebagai *client* sehingga untuk c*lient* dibutuhkan *sofware* tambahan yakni *VLC*.
2. *Video LAN Client* (*VLC*) memiliki fungsi yang sama dengan *VLS*, namun dapat bertindak sebagai *server streaming* dan *client*. *VLC Media Player* bisa diperoleh secara gratis dan bersifat *opensource*. Program ini tersedia dalam berbagai *platform* sistem operasi, mulai dari *Microsoft Windows*, beragam distro *Linux*, *Mac OS*, dan beberapa sistem operasi lainnya.

2.7 *Opencaster*

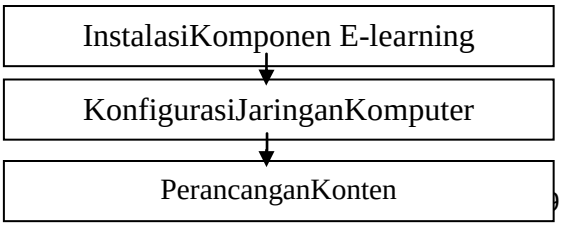
Perangkat lunak *Opencaster* adalah sebuah perangkat lunak berbasis *opensource* yang dikembangkan oleh Andrea Venturi (*Electronic Engineer Administrator*) dan Lorenzo Pallara (*Komputer Science Technical Director*) dari *Avalpa Digital Engineering* di Italia yang sampai saat ini telah dikembangkan sampai versi 3.2.2. *Opencaster* adalah sebuah perangkat lunak untuk menghasilkan, memproses, *multiplex*, menjalankan, dan menyiarkan sebuah *MPEG-2 transport stream*. *Transport stream* (*.TS*, *.TP*, *MPEG-TS*, atau *.M2T*) adalah sebuah protokol komunikasi untuk *video*, *audio*, dan data di mana dispesifikasikan *dalam MPEG-2 (ISO/IEC standar 13818-1).*

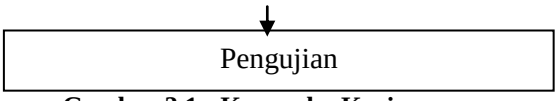
Tujuan *Opencaster* dirancang untuk melakukan *multiplexing* pada *digital video* dan *audio* dan untuk mensinkronkan hasil *output*. Fitur-fitur unik dari *Opencaster* di antara perangkat yang lain adalah dapat mendukung *Advanced Data Casting* melalui *DSMCC* dan *IP* melalui *transport stream(IP-MPE). Transport stream* menawarkan fitur-fitur untuk koreksi kesalahan untuk *transport* melalui media yang tidak *reliable* dan digunakan dalam *broadcast* jaringan seperti *Digital Video Broadcast (DVB). Opencaster* berjalan di sistem operasi *Linux* dan biasanya dikembangkan dalam *Debian Stable Distribution* untuk arsitektur 32bit X86. *Transport stream* yang dihasilkan dapat dipancarkan melalui banyak jaringan dan media, maupun di-*multicast* dalam sebuah protokol *UDP/IP*.

3. Metode Penelitian

3.1 Kerangka Kerja (*Frame Work*)

Dalam penelitian ini, kerangka kerja yang dipakai terdiri dari beberapa tahapan, adapun tahapan tersebut adalah :





Gambar 3.1 : Kerangka Kerja

3.2 Instalasi Komponen *Elearning* Berbasis *Linux*

Sebelum melakukan langkah-langkah instalasi komponen *e-learning* dilakukan pengumpulan data sesuai dengan topik yang dibahas yang bersumber dari berbagai sumber yang ada. Adapun metode yang digunakan dalam pengumpulan data adalah

1. Studi Literatur
   Merupakan tahap pertama yang dilakukan, dimana data diperoleh dengan cara membaca, mengumpulkan referensi dan kajian literatur tentang *linux ubuntu, e-learning, vlc, ffmpeg, multimedia video-audio streaming, IP-TV, opencaster*.
2. Kemudian langkah selanjutnya adalah instalasi komponen yang dimulai dari instalasi sistem operasi, instalasi aplikasi, dan sampai pada konfigurasi aplikasi.
3. Observasi
   Observasi yang dilakukan yaitu melalui pendekatan kualitatif, di mana pengamatan dilakukan secara optimal, karena pengamatan didasarkan atas pengalaman langsung yang memungkinkan mencoba, mengamati,dan kemudian mencatat hal-hal yang telah diuji.

3.3 Konfigurasi Jaringan Komputer

Pada tahap ini mulai dilakukan konfigurasi sistem jaringan komputer berbasis *LAN* yang terdiri dari sebuah *server* yang terinstal sistem operasi *linux ubuntu* dan beberapa *client* dengan sistem operasi *microsoft windows* dengan aplikasi *vlc player*.

1. Di sisi *server*
   a. Perangkat keras yaitu PC dengan spesifikasi yang harus cukup tinggi
   b. Perangkatlunak
      - Sistemoperasi*Linux Ubuntu* 9.04
      - Aplikasi*Opencaster* 3.2.2
      - Aplikasi *ffmpeg* sebagai *encoder* dan *decoder*
2. Di sisi *client* (1 buah *PC* dan 1 buah *Laptop*) dengan spesifikasi yang harus cukup tinggi dan perangkat lunak sistem operasi dan aplikasi *VLC.*

Selain di sisi *server* dan *client* ada tambahan alat jaringan yaitu : 1 buah *switch* 8 *port* dan 3 buah kabel *UTP* + *RJ*-45 dengan panjang masing-masing 2 meter.

3.4 Perancangan Konten

Pada tahap ini akan dilakukan proses pembuatan konten yang dimulai dengan mempersiapkan beberapa data *video-audio*, yang kemudian dikonversi ke dalam file *transport stream*

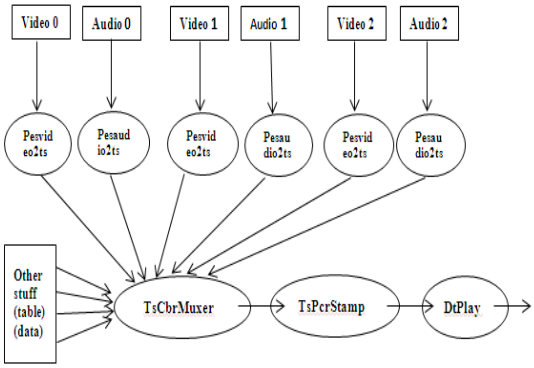
(TS), setelah file *video-audio* berektensi **.TS* selesai dikonversi, maka langkah selanjutnya adalah melakukan proses *multiplexer* yaitu proses penggabungan beberapa file *transport stream* menjadi sebuah file *transport stream* yang multi konten.

3.5 Pengujian

Pengujian dilakukan dalam sebuah sistem jaringan *LAN* yang kemudian file multi konten *video-audio* di-*streaming* melalui sebuah *server* dengan menggunakan aplikasi *Opencaster*. Dan di sisi *client* dengan menggunakan aplikasi *vlc* dan sedikit konfigurasi, langsung dapat melihat tayangan yang telah dipancarkan.

4.1 Analisa Proses

Dalam pembuatan *video-audio* multi konten tentunya diperlukan sebuah perangkat lunak yang bisa menunjang proses pembuatan *video-audio* tersebut. Di samping itu ada satu hal yang perlu diperhatikan yaitu mengenai biaya yang harus dikeluarkan jika menggunakan perangkat lunak yang berbayar. Maka dari itu menggunakan perangkat lunak *Opencaster* yang berbasis *opensource* di dalam penerapannya. *Opencaster* ini memang merupakan perangkat lunak yang sudah dirancang khusus di dalam melakukan pembuatan *video-audio* multi konten dan sekaligus juga untuk pemancarannya melalui sistem jaringan berbasis *IP-TV*. *Video-audio* multi konten ini dibuat dalam bentuk file *MPEG-2 Transport Stream* atau dalam bentuk paket *TS*.

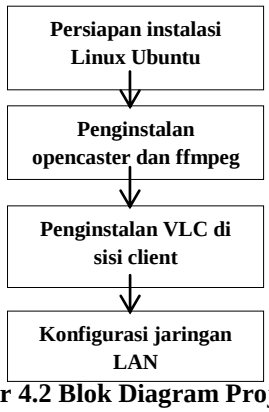


Gambar 4.1 *Video-Audio* Multi Konten Sistem

Pada gambar di atas menjelaskan bagaimana proses data yang berupa beberapa file *video-audio* dikonversi ke dalam file berekstensi *transport stream*. Setelah itu dari hasil konversi kemudian dilakukan proses *multiplexer* yang menjadikan sebuah file *transport stream* menjadi banyak konten di dalamnya, yang akhirnya siap untuk dipancarkan.

4.2 Spesifikasi Perangkat

Adapun blok diagram dari sistem yang akan direalisasikan yaitu terdiri dari instalasi sistem operasi *linux ubuntu*, instalasi aplikasi *video-audio streaming Opencaster, ffmpeg, dan vlcclient*, dan konfigurasi jaringan komputer,.



Gambar 4.2 Blok Diagram Projek

Gambar di atas menjelaskan langkah-langkah yang harus dilakukan sebelum dapat melakukan perancangan sebuah sistem *e-learning.*

4.3 Perancangan Sistem *E-learning*

Penggunaan jaringan *intranet* maupun *internet, Opencaster*, dan *video lan client* sebagai media pendistribusian *video-audio streaming* multi konten untuk mendukung proses belajar mengajar sangatlah efisien dan efektif.

Dalam penelitian ini akan memaparkan tentang bagaimana perancangan v*ideo-audio streaming* multi konten dengan aplikasi *Opencaster*, dan bagaimana pendistribusiannya ke *client* melalui protocol *TCP-IP* berbasis *IP-TV*, yang terdiri dari penginstalan *Opencaster* yang digunakan sebagai *server*, *VLC* untuk *client*, penyetingan *VLC* di *Microsoft Windows XP*, dan konfigurasi sistem jaringan *LAN*. Dalam perancangan dan pembuatan sistem *e-learning* ini diperlukan pendalaman materi dan bahan yang membahas mengenai *hardware* dan *software* mengenai sistem jaringan, teknik konversi file *video-audio* dengan *ffmpeg*, *multiplexing transport stream* file, *video-audio streaming* dengan *Opencaster*, dan *video lan client*.

Dalam perancangan sistem *e-learning* berbasis *video-audio streaming* multi konten ini cukup menarik untuk dijelajahi karena relatif baru dengan biaya yang cukup murah. Dengan semakin murahnya peralatan elektronik dan aplikasi yang bersifat *opensource*, maka penerapannya juga akan semakin nyata untuk di lingkungan pendidikan. Perancangan sistem *e-learning* berbasis *video* memang membutuhkan *bandwidth* yang besar, dikarenakan proses *streaming* yang dilakukan harus *real time*, sehingga tidak menyebabkan terjadinya gambar yang patah-patah. Proses di dalam menghantarkan *video* membutuhkan *bandwidth* yang lebar (sangat banyak *byte* per detik yang dikirimkan), yang oleh karenanya sangat membutuhkan teknologi kompresi *video* untuk mengurangi kebutuhan *bandwidth* sebelum

dikirimkan melalui saluran komunikasi. Sekedar gambaran singkat, sebuah kanal *video* yang baik tanpa dikompresi akan mengambil *bandwidth* sekitar 9 *Mbps*. Dengan teknik kompresi yang sudah ada pada saat ini, kita dapat menghemat sebuah kanal *video* sekitar 30 *Kbps*. Itu berarti sebuah saluran *internet* yang tidak terlalu cepat sebetulnya dapat digunakan untuk menyalurkan *video*. Beberapa hal yang perlu diperhatikan dalam pengiriman *video* adalah :

1. Jika kita menggunakan *video* hitam-putih akan memakan *bandwidth* lebih kecil dari pada jika kita menggunakan *video* berwarna.
2. Jika kita menggunakan kecepatan pengiriman. Kecepatan pengiriman *frame per second* (*fps*) *video* yang rendah, akan memakan *bandwidth* yang lebih rendah dibandingkan *frame per second* (*fps*) yang tinggi. *Video* yang cukup baik biasanya dikirim dengan kecepatan *frame per second* (*fps*) sekitar 30 *fps*. Jika dikirimkan tanpa kompresi, sebuah *video* dengan 30 *fps* akan mengambil *bandwidth* kira-kira 9 *Mbps*, amat sangat besar untuk ukuran kanal komunikasi data.
3. Perlu diketahui pada sebuah *streaming video* adalah bagaimana data tersebut dikompres serta format isi dari data yang di-*codec*-kan. Semua data baik *video* ataupun *audio* yang akan di-*stream* harus melewati kedua proses di atas. Kompresi dilakukan untuk mengurangi ukuran data yang akan di-*stream* walaupun mengakibatkan kualitas data yang sampai ke *client* tidak persis sama dengan kualitas data asli. Data yang dikompres untuk *streaming video* ini dapat dibedakan dalam dua bagian yakni *video* dan suara. Algoritma kompresi yang sering digunakan adalah *MPEG-1*, *MPEG-2*, *MPEG-4*, *MPEG-7*, *vorbis*, *Divx*, dan lain-lain.

5.1 Konfigurasi Perangkat *Client-Server*

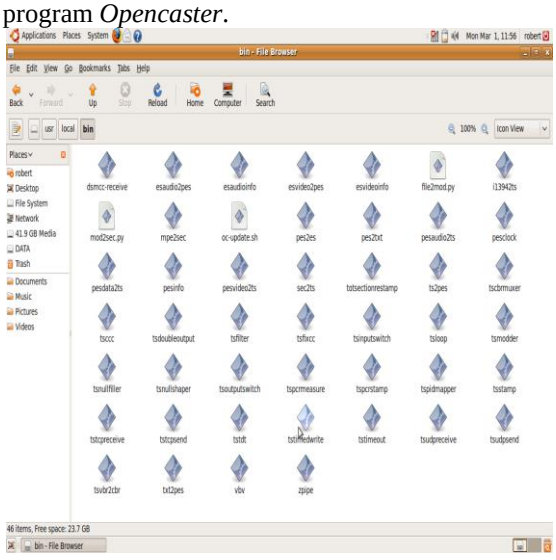
Dalam melakukan pengujian sistem *e-learning video-audio streaming* ini, kecepatan *processor*, *memory*, dan kapasitas *hard disk* sangat diperlukan (khususnya di bagian *server database video-audio* multi konten), karena yang diolah adalah suatu gambar bergerak atau *video*. Dalam ujicoba yang dilakukan terdapat dua bagian yang terdiri dari *server*, dan *client*, di mana dengan memanfaatkan beberapa perangkat keras yang tidak memiliki spesifikasi yang begitu tinggi, dan beberapa perangkat lunak yang terdiri dari sistem operasi dan aplikasi pendukung dalam pembuatan *video-audio streaming* dan *broadcasting*, yang kemudian dihubungkan ke dalam sebuah jaringan *LAN* sederhana.

5.2 Menjalankan *Opencaster*

Proses implementasi sistem *e-learning* ini dilakukan dengan tahapan sebagai berikut :

1. Instalasi program *Opencaster*2.2 disisi*server*

Untuk dapat menggunakan *Opencaster* dalam melakukan *multiplexing video audio*, perangkat lunak *Opencaster* tersebut dapat *download* di situs http://www.avalpa.com. Setelah selesai di-*download* file *Opencaster* yang berbentuk file kompresi, maka file tersebut harus di-*extract* terlebih dahulu sebelum bisa dijalankan di sistem operasi *Linux Ubuntu*. *Opencaster* hanya dapat dijalankan di sistem operasi *Linux* yang telah diuji di *Linux Ubuntu* ataupun di *Debian Linux X86 32bit*. Biasanya sebelum *Opencaster* bisa diinstal, ada paket-paket aplikasi seperti *python-dev*, *libcap-dev*, *zlib1g-dev* yang dibutuhkan dan harus diinstal terlebih dahulu. Tetapi aplikasi-aplikasi tersebut belum tersedia di dalam *Linux*. Untuk menginstal paket-paket tersebut kita bisa mengambil langsung di repository *Linux* di *internet* dengan perintah-perintah seperti *sudo apt-get install python-dev*, *sudo apt-get install libcap-dev*, dan *sudo apt-get install zlib1g-dev* yang ada di sistem operasi *Linux*. Perlu diingat bahwa untuk aplikasi *python* diharuskan versi 2.4 atau 2.5. Setelah semua langkah di atas telah dilakukan, maka untuk menginstal *Opencaster* diharuskan mengetik perintah "*make install*" di *level root* dari sistem operasi *Linux* melalui sistem terminal. Dan untuk mengetahui instalasi *Opencaster* telah berhasil, maka akan tampak sebuah *folder* di *Usr/ Local/ Bin* yang berisi program *Opencaster*.



Gambar 5.1 Tampilan File *Opencaster*

2. Mengkonversi file *video-audio* ke dalam bentuk file *transport stream* Dengan aplikasi *ffmpeg* sebagai aplikasi *encoder* dan *decoder*, maka bisa dihasilkan file dalam bentuk *transport stream*. Langkah- langkah yang harus dilakukan adalah menyediakan file-file *video-audio* yang ingin dibuat menjadi multi konten, dalam penelitian ini disediakan file *video-audio* sebanyak 6 (enam) file dalam bentuk **.vob*, setelah itu file harus di-*encode*

terlebih dahulu dengan aplikasi *ffmpeg*. Berikut ini adalah langkah-langkah dalam melakukan *encode* file *video-audio*.

File1

- Ffmpeg –vn –ab 128k –ar 480000 –i elearning1.vob –acodec mp2 –ac 2 suara1.mp2
- Ffmpeg –I elearning1.vob –an –f mpeg2video –vcodec mpeg2video –b 2600k –maxrate 2600k –minrate 2600k –bf 2 –bufsize 1835008 belajar1.m2v
- Esvideo2pes belajar1.m2v > belajar1.video.pes>belajar1.pes.length
- Esaudio2pes suara1.mp2 1152 48000 384 3600>suara1.audio.pes
- Pesvideo2ts 2064 25 112 2900000 0 belajar1.video.pes>belajar1.ts
- Pesaudio2ts 2070 1152 48000 384 0 suara1.audio.pes>suara1.ts

Untuk file *video-audio* kedua sampai ke enam juga sama cara konversinya seperti di atas, tetapi yang perlu diperhatikan adalah mengenai *pid video* dan *audio*-nya, dimana *pid video* di file pertama adalah 2064, maka di file kedua dan seterusnya adalah 2065, 2066, 2067, 2068, 2069, dan untuk *pid audio* pada file pertama adalah 2070, dan untuk file kedua dan seterusnya adalah 2071, 2072, 2073, 2074, 2075, 2076. Setelah semua file telah dikonversi maka akan menghasilkan file yang berekstensi *.ts (belajar1.ts).

3. Membuat file *video-audio transport stream* menjadi multi konten.

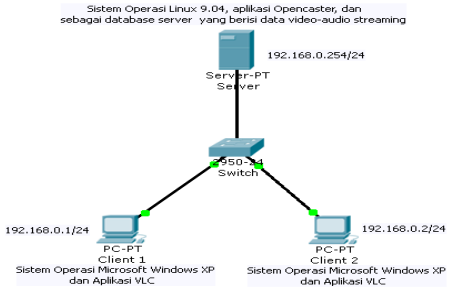
   Setelah semua file *video-audio* tersebut dikonversi ke dalam file *transport stream*, maka sebelum bisa membuat file *.ts menjadi multi konten, ada sebuah *listing program* dari aplikasi *Opencaster* yang harus diubah terlebih dahulu yaitu file yang berekstensi *mpts config.py*. Dan setelah diubah harus dieksekusi, melalui menu terminal di *Linux Ubuntu* dengan cara mengetik **./mptsconfig.py**. Adapun hasil eksekusi file tersebut akan muncul file yang berfungsi untuk membuat file *video-audio* multi konten tersebut. File-file yang dihasilkan yaitu *mptspmt1 .ts*sampai *mptspmt6.ts*, *mptsnit.ts*, dan *mptssdt.ts* karena sesuai dengan konten yang ingin kita buat yaitu 6 (enam) konten. Setelah ada file-file diatas maka proses pembuatan multi konten atau *multiplexer* dapat dilakukan. Dalam melakukan proses *multiplexer* digunakan perintah sebagai berikut :

   tscbrmuxer b:2800000 belajar1.ts b:1880000 suara1.ts b:2800000 belajar2.ts b:1880000 suara2.ts b:2800000 belajar3.ts b:1880000 suara3.ts b:2800000 belajar4.ts b:1880000 suara4.ts b:2800000 belajar5.ts b:1880000 suara5.ts b:2800000 belajar6.ts b:1880000 suara6.ts b:3008 mptspat.ts b:3008 mptspmt1.ts b:3008 mptspmt2.ts b:3008 mptspmt3.ts b:3008 mptspmt4.ts b:3008 mptspmt5.ts b:3008 mptspmt6.ts b:1400 mptsnit.ts b:1500 mptssdt.ts b:10174084 null.ts>belajar.ts

   dalam melakukan proses *multiplexer* untuk menghasilkan file belajar .ts diharapkan untuk menghentikan prosesnya dengan menekan tombol Ctrl+C jika file yang kita inginkan telah cukup untuk disimpan, karena kalau tidak dihentikan maka bisa menyebabkan kapasitas *hard disk* kita menjadi penuh.

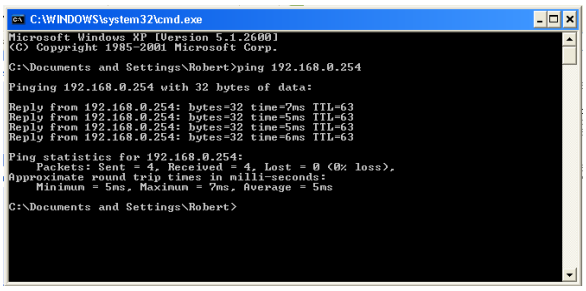
4. Jaringan Komputer

   Pada tahap ini digunakan sistem jaringan komputer secara *LAN*



Gambar 5.2 Sistem Jaringan *LAN* Dalam Implementasi

Gambar di atas merupakan sistem jaringan sederhana yang digunakan dalam melakukan implementasi sistem *e-learning video-audio streaming*, di mana masing-masing perangkat keras pada kartu jaringannya diberikan nomor *IP* yang unik yaitu pada bagian *server IP* yang diberikan adalah 192.168.0.254 dengan *subnet mask* 255.255.255.*0*, kemudian di sisi *client IP* yang diberikan adalah 192.168.0.1 dan 192.168.0.2 dengan *subnet mask* yang sama pada *server*. Dan untuk melakukan pengujian apakah setiap komputer sudah dapat terhubung dalam jaringan, dapat menggunakan perintah *ping* pada *command prompt* yang ada di sistem *windows*. Jika ada tulisan *reply from* 192.168.0.254, maka komputer sudah terhubung ke dalam sistem jaringan yang telah dibuat.



Gambar 5.3 Tampilan Menguji Koneksi Jaringan Dengan Perintah *Ping*

5.3 Melakukan *Streaming* Dari *Server* Dengan Aplikasi *Opencaster*

Dalam melakukan *streaming* dapat dilakukan dengan 2 cara yaitu dengan perintah *tstcpsent* atau *tsudpsend*. Contoh *streaming* yang dilakukan dengan *tscudpsend*

Tsudpsend belajar .ts 192.168.0.2 7001 290000.
Dimana :

- belajar.ts merupakan file yang akan di-*broadcast*
- 192.168.0.2 merupakan *IP client*
- 7001 merupakan *port* koneksi yang diberikan
- 290000 merupakan *bit rate* dari tranmisi data

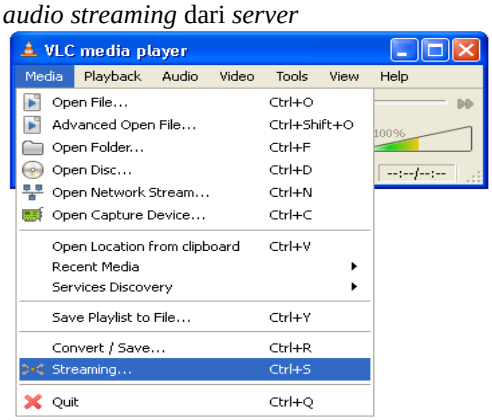
5.4 Menjalankan Sistem *E-learning* Dari *Client*

Pada tahap ini dilakukan akses sistem *e-learning* dari *client* dengan menggunakan *VLC* yang ada pada masing-masing *client*.

1. Tampilan awal *VLC*
   Berikut ini merupakan tampilan dari aplikasi VLC yang akan digunakan untuk pemilihan konten *video-audio streaming*.

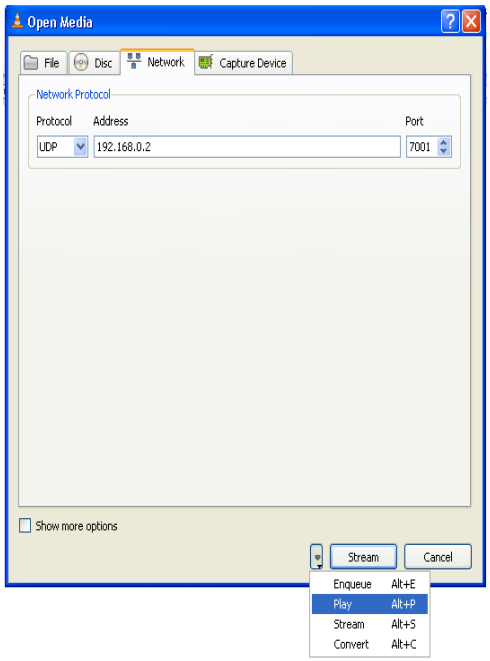
Gambar 5.4 Tampilan Awal Aplikasi *VLC*

2. Tampilan sewaktu mau mengambil file *video-audio streaming* dari *server*



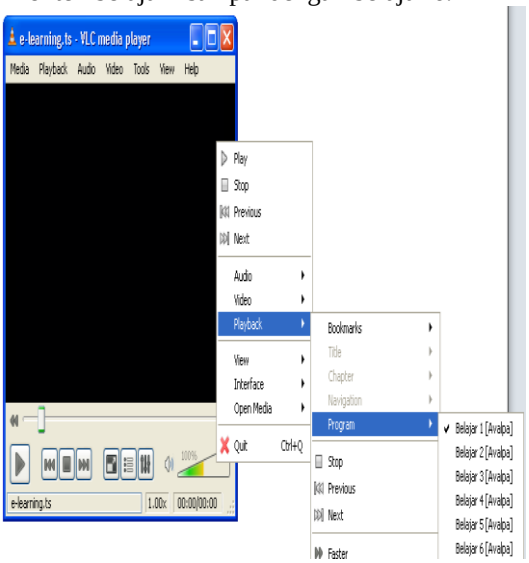
Gambar 5.5 Tampilan Untuk Mengambil File *Video-Audio Streaming*

3. Tampilan konfigurasi untuk membuka file *video-audio* yang sudah di-*streaming* dari *server*, di mana proses streaming yang dilakukan menggunakan protocol *udp* yang kemudian ditujukan ke alamat 192.168.0.2 dengan *port* koneksi 7001. Setelah itu tinggal memilih menu *play* untuk menjalankan file *video-audio*-nya



Gambar 5.6 Tampilan Konfigurasi Untuk Menjalankan File *Video-Audio*

4. Tampilan *video-audio streaming* multi konten yang bisa dipilih oleh *client*, di mana dengan melakukan klik kanan pada aplikasi *VLC*, dan kemudian pilih *playback* → *program* → dan kemudian pilih konten yang ingin dilihat. Dalam hal ini boleh dipilih konten belajar1 sampai dengan belajar 6.



Gambar 5.7 Tampilan Untuk Memilih Konten Yang Diinginkan

6. Kesimpulan

6.1 Kesimpulan

Dari ujicoba yang dilakukan, dapat diambil beberapa kesimpulan yaitu:

1. Dengan adanya aplikasi *Opencaster* dan*VLC Client* yang *opensource*, makabisaditerapkansistem *e-learning* berbasis *IP-TV* sebagai media yang bisa memberikan sebuah terobosan dalam dunia pendidikan untuk meningkatkan kualitas belajar siswa.
2. Di dalam pembuatan konten *multivideo* dan *audio streaming*, terdapat kendala sewaktu proses *multiplexer* berlangsung, di mana prosesnya harus dihentikan sesuai dengan yang kita inginkan, jika tidak maka akan menyebabkan kapasitas media penyimpanan menjadi penuh.

6.2. Saran

Adapun saran yang ingin disampaikan yaitu pengembangan sistem *e-learning* sebaiknya memasukkan factor interaktif agar pengguna bisa langsung memberikan respon yang akan diterima oleh *server*.

Daftar Pustaka

Askari Azikin danYudha Purwanto, ST, 2005, Video / TV Streaming dengan Video LAN Project, PenerbitAndi, Yogyakarta

Fischer, W, 2008, Digital Video Audio Broadcasting Technology, Springer, Germany

Held, G, 2006, Understanding IPTV, Auerbach Publication, New York

Simpson, W dan Greenfield, H, 2007, IPTV and Internet Video, Elsevier Inc, UK

http://www.avalpa.com

http:// ffmpeg.org

http://www.videolan.org.