BAB III

ANALISIS DAN PERANCANGAN

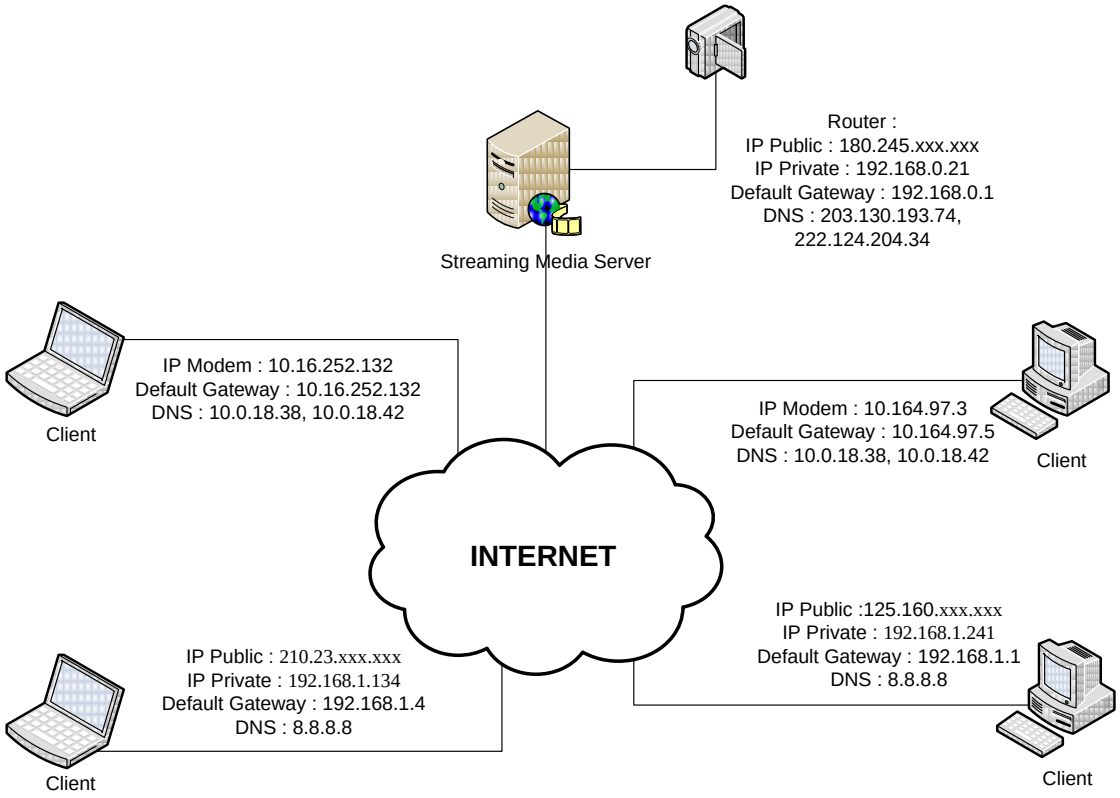
3.1 Analisis

Analisis sistem (system analysis) dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dangan maksud untuk mengidentifikasikan dan mengevaluasikan permasalah-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan. Analisis dapat juga diartikan memahami sistem pemikiran yang kompleks dengan memecahnya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur-unsur itu menjadi jelas. [3]

3.1.1 Analisis Masalah

Teknologi internet sudah menjadi kebutuhan pokok bagi sebagian besar masyarakat dalam menunjang aktifitas kerja sehari-hari. Kebutuhan masyarakat akan informasi sudah mengalamai suatu revolusi dimana tidak hanya informasi berupa text dan gambar tetapi juga melibatkan semua aspek multimedia khususnya video. Informasi yang masih berupa gambar dan teks, dirasakan kurang representatif sedangkan masyarakat lebih membutuhkan media informasi yang interaktif dan efisien sehingga informasi tersebut sampai kepada masyarakat lebih cepat dan akurat. Dengan memanfaatkan teknologi video *streaming* dan internet sebagai mediatornya maka pada saat ini beberapa informasi misalnya seperti arus lalu lintas, aktifitas terkini dari

gunung berapi dan event tertentu dapat dinikmati dalam bentuk pemutaran video visual secara langsung. Teknologi yang menyediakan layanan untuk mendapatkan informasi terkini yang diinginkan kapan dan dimana saja yang kemudian ditayangkan dalam bentuk video dikenal dengan sebutan *Live Video Streaming* .

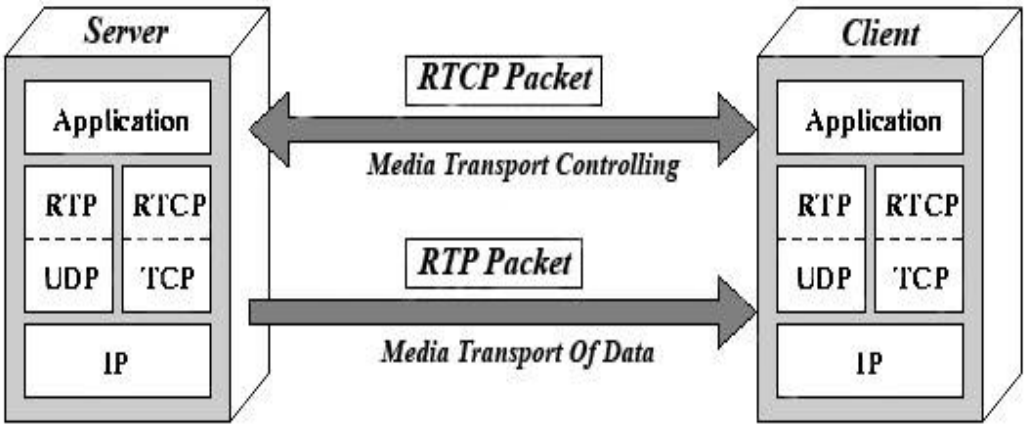


Gambar 3.1 Gambaran Umum Sistem Video *Live Streaming*

3.1.2 Analisis Protokol

Layanan media *streaming* ini pada dasarnya adalah berbasis web pada sisi *client*, maka sebagai portal akses informasinya dapat memanfaatkan protokol HTTP. Selain itu diperlukan protokol lain yang dapat membangun layanan yang interaktif, maka digunakanlah protokol yang mengatur koneksi status antara media *server* dengan *client*, protokol yang dimaksud adalah *Real Time Transport Protocol* (RTP) yang

berfungsi sebagai media *transport* berupa video secara *real-time* pada jaringan IP. Dengan protokol tersebut pengguna dapat mengakses proses *streaming* yang sedang berlangsung. RTP biasanya digunakan untuk mengangkut data melalui *User Datagram Protocol* (UDP), bersama dengan UDP menyediakan fungsionalitas *transport* protokol.



Gambar 3.2 Peran RTP Pada Teknologi *Streaming*

Protokol RTP merupakan standar untuk transportasi real-time melalui jaringan IP yaitu untuk *streaming* video, *streaming server* memotong media data untuk masing-masing track menjadi paket RTP yang kemudian ditransmisikan melalui UDP pada jaringan.

3.1.2.1 RTP *Header*

Header RTP memiliki ukuran minimum 12 *byte*. Setelah *header*, *extension header* opsional mungkin ada. Hal ini diikuti oleh RTP *payload*, format yang ditentukan oleh kelas tertentu aplikasi.

Field dalam header adalah sebagai berikut :

1. V (*Version*)

   Memiliki ukuran 1 bit, berfungsi mengindikasikan versi protokol. Versi saat ini adalah 2.

2. P (*Padding*)

   Memiliki ukuran 1 bit, digunakan untuk mengindikasikan apakah ada *byte padding* tambahan pada akhir paket RTP. *Padding* A dapat digunakan untuk mengisi sebuah blok ukuran tertentu, misalnya sebagai diperlukan oleh sebuah algoritma enkripsi.

3. X (*Extension*)

   Memiliki ukuran 1 bit, berfungsi mengindikasikan adanya *header extension* antara *header* standar dan data *payload*. Ini adalah aplikasi atau profil tertentu.

4. CC (*Contributing Source Count*)

   Memiliki ukuran 4 bit, mengandung jumlah pengenal CSRC yang mengikuti *header* tetap.

5. M (*Marker*)

   Memiliki ukuran 1 bit, digunakan pada tingkat aplikasi dan didefinisikan oleh profil. Jika diset, ini berarti bahwa data yang saat ini memiliki beberapa relevansi khusus untuk aplikasi.

6. PT (*Payload Type*)

   Memiliki ukuran 7 bit, berfungsi mengindikasikan format *payload* dan menentukan interpretasinya oleh aplikasi. Hal ini ditentukan oleh profil RTP. Sebagai contoh, untuk konferensi audio dan video dengan kontrol yang minimal.

7. *Sequence Number*

   Memiliki ukuran 16 bit, nomor urutan bertambah satu untuk setiap paket data yang dikirim dan RTP yang akan digunakan oleh penerima untuk mendeteksi kehilangan paket dan untuk mengembalikan urutan paket. RTP tidak mengambil tindakan apapun terhadap kehilangan paket, melainkan diserahkan kepada aplikasi untuk mengambil tindakan yang diinginkan. Sebagai contoh, aplikasi video mungkin memainkan frame terakhir yang diketahui di tempat dari frame yang hilang. RTP tidak memberikan jaminan pengiriman, namun keberadaan nomor urut memungkinkan untuk mendeteksi paket yang hilang.

8. *Timestamp*

   Memiliki ukuran 32 bit, digunakan untuk memungkinkan penerima untuk memutar ulang sampel yang diterima pada interval yang tepat. Ketika beberapa media stream muncul, cap waktu yang independen dalam setiap aliran, tidak dapat diandalkan untuk sinkronisasi media.

9. SSRC (*Synchronization Source*)

   Memiliki ukuran 32 bit, mengidentifikasi tanda pengenal yang unik dari sumber stream. Sumber sinkronisasi dalam sesi RTP yang sama akan menjadi unik.

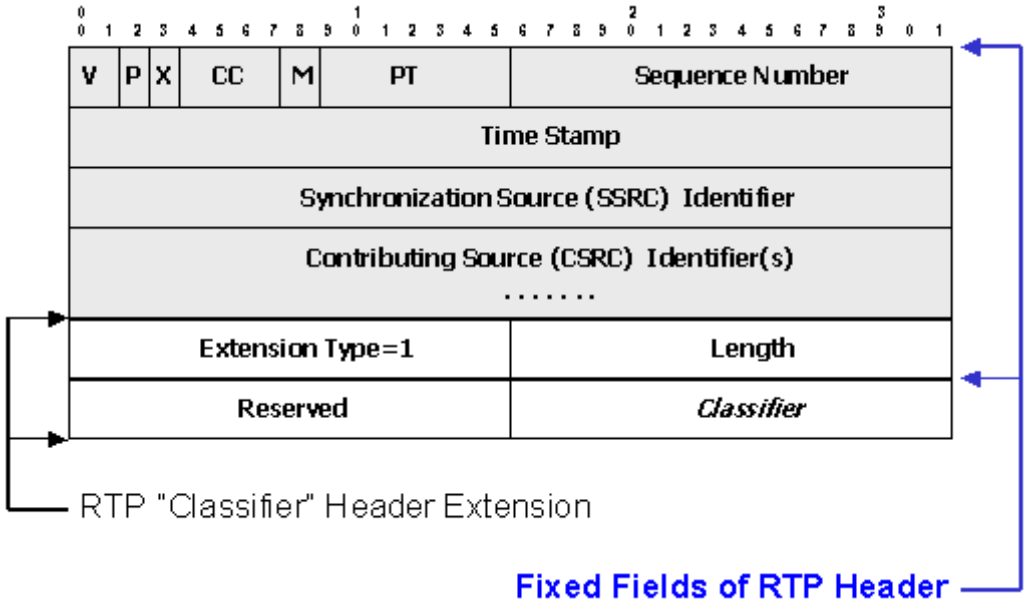
10. CSRC (*Contributing Source*)

    Memiliki ukuran 4 bit, menghitung sumber kontribusi untuk stream yang telah dihasilkan dari berbagai sumber.

11. *Extension Header*

    Ukuran opsional, 32-bit pertama berisi profil pengidentifikasi (16 bit) dan sebuah penspesifikasi panjang (16 bit) yang menunjukkan panjang perpanjangan (EHL =

extension header length) dalam satuan 32-bit , tidak termasuk 32 bit dari *header* ekstensi.



Gambar 3.3 RTP Header

3.1.2.2 Validasi Paket RTP

Hal ini penting untuk memvalidasi apakah paket yang diterima benar-benar paket RTP. Aturan pengepakan, memungkinkan paket RTP menjadi sangat ketat untuk divalidasi. Keberhasilan validasi aliran RTP memberikan jaminan tinggi bahwa aliran RTP yang sesuai juga berlaku, meskipun tidak meniadakan kebutuhan untuk validasi dari paket RTP.

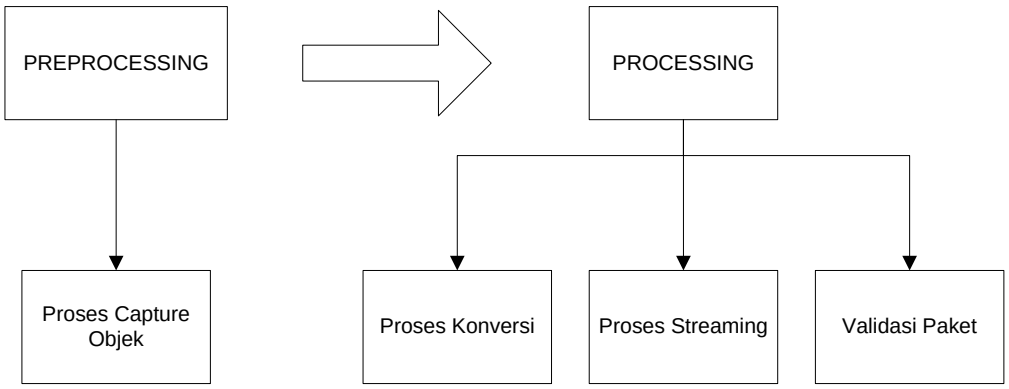
Dalam menampilkan *pseudocode* untuk proses validasi, terdapat beberapa hal penting yang harus diperhatikan antara lain :

1. Semua paket harus merupakan urutan paket RTP.
2. Nomor versi pada semua *field* pada semua paket harus sama dengan 2.

3. Jika *padding* diperlukan, akan ditambahkan hanya pada paket terakhir. *Bit padding* harus bernilai nol untuk semua paket lainnya dalam urutan paket RTP.

3.1.3 Analisis Proses Video *Streaming*

Server sebagai penyedia layanan video *streaming* merekam suatu *event* menggunakan webcam yang terhubung pada komputer *server*, hasil gambar yang terekam kemudian akan dikirimkan secara *real time.* Hasil *capture* video dari objek ditransmisikan secara langsung menggunakan protokol *RTP* menuju *web server* sekaligus agar bisa diakses oleh *client* yang terhubung pada jaringan.



Gambar 3.4 Alur Proses *Streaming*

Berdasarkan gambar diatas terdapat 2 tahap dalam alur proses *streaming* yaitu *preprocessing* dan *processing.* Tahapan *preprocessing* adalah proses *capture* objek dari video *device* yaitu webcam yang merekam *event* yang akan menjadi media *streaming* yang nantinya akan ditransmisikan secara langsung kepada pengguna.

#Pseudocode *webcam* merekam dan menampilkan suatu objek

```
namespace WebCam
{
    public partial class MainPage : UserControl
    {
        CaptureSource _captureSource = null;

        public MainPage()
        {
            InitializeComponent();

        }

        void StartWebcam()
        {
            if (_captureSource != null &&
                _captureSource.State != CaptureState.Started)
            {
                if (CaptureDeviceConfiguration.AllowedDeviceAccess
                 ||
CaptureDeviceConfiguration.RequestDeviceAccess())
                {
                    VideoCaptureDevice device =
CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();
                    if (device != null)
                    {
                        _captureSource = new CaptureSource();
                        _captureSource.VideoCaptureDevice = device;
                        _captureSource.Start();
                        VideoBrush brush = new VideoBrush();
                        brush.Stretch = Stretch.Uniform;
                        brush.SetSource(_captureSource);
                        videoRect.Fill = brush;
                    }
                }
            }
        }
        private void buttonStartWebcam_Click(object sender,
                                             RoutedEventArgs e)
        {
            StartWebcam();
        }
    }
}
```

Setelah *webcam* telah berhasil melakukan proses *capture* suatu objek maka proses selanjutnya adalah masuk ke tahap *preprocessing.* Pada tahap ini terdapat beberapa proses antara lain :

1. Proses konversi, proses setelah proses *capture* dari video *device* dimana objek telah dikonversikan menjadi data digital yang siap dilakukan proses *encoding* agar data dapat ditransmisikan kepada pengguna layanan.
2. Proses *streaming*, proses mentransmisikan video yang telah dilakukan proses *encoding* pada jaringan sehingga pengguna dapat mengakses video tersebut melalui URL video *streaming* yang telah disediakan oleh penyedia layanan.

#Pseudocode proses video *streaming*

```
/dev/video_device :width=x :height=y :fps=z
     -- sout '#transcode {vcodec}, duplicate {protocol} ,
        {
          mux = encoding_engine {video_format}
          dst = # {IP_Address} : # {Port} / video
        }
```

3. Proses validasi paket, proses untuk menjamin bahwa paket video sampai kepada pengguna. Akan tetapi dalam penelitian ini menggunakan protokol UDP yang sifatnya tidak menjamin paket sampai kepada pengguna atau tidak. Oleh karena itu, dilakukan upaya terbaik dengan memanfaatkan protokol TCP/IP dalam melakukan proses ini dalam membantu terjaminnya paket sampai kepada pengguna.

#Pseudocode untuk validasi paket RTP

```
validate_rtp(rtp_t *packet, int length)
    rtp_t*end = (rtp_t *)(((char *)packet)+length);
    rtp_t*r   = packet;
    intl = 0;
    intp = 0;
    // Semua paket harus urutan paket RTP

    if ((packet->length+ 1)*4) == length) {
        ... error: bukan urutan paket
    }
    // Cek versi RTP, tipe paket, dan padding pertama pada
    // urutan paket RTP...

    if (packet->version != 2) {
```

```
        ...error: nomor versi != 2 pada subpaket
    }

    if (packet-> p != 0) {
        ...error: bit padding harus diatur pada paket pertama
    }

    if ((packet->pt != RTP_SR) && (packet->pt != RTP_RR)) {
        ...error: urutan paket tidak boleh diawali SR (Sending
        Reception) atau RR (Receiver Report)
    }
    // periksa semua bagian pada urutan paket RTP.
    // nomor versi harus 2, dan nilai bit padding harus 0 pada
       semua
    // paket kecuali paket terakhir.

    do {
        if (p == 1) {
            ...error: padding sebelum paket terakhir dalam urutan
        }
        if (r-> p) {
            p = 1;
        }
        if (r-> version != 2) {
            ...error: nomor versi != 2 dalam subpaket
        }
        l += (r->length + 1) * 4;
        r  = (rtp_t *) (((uint32_t *) r) + r->length + 1);
    } while (r < end);
    // Cek jika panjang paket RTP cocok dengan panjang paket UDP
    yang
    // diterima...
    if ((l != length) || (r != end)) {
        ...error: panjang paket tidak cocok dengan panjang paket
UDP
    }
      ...paket valid
}
```

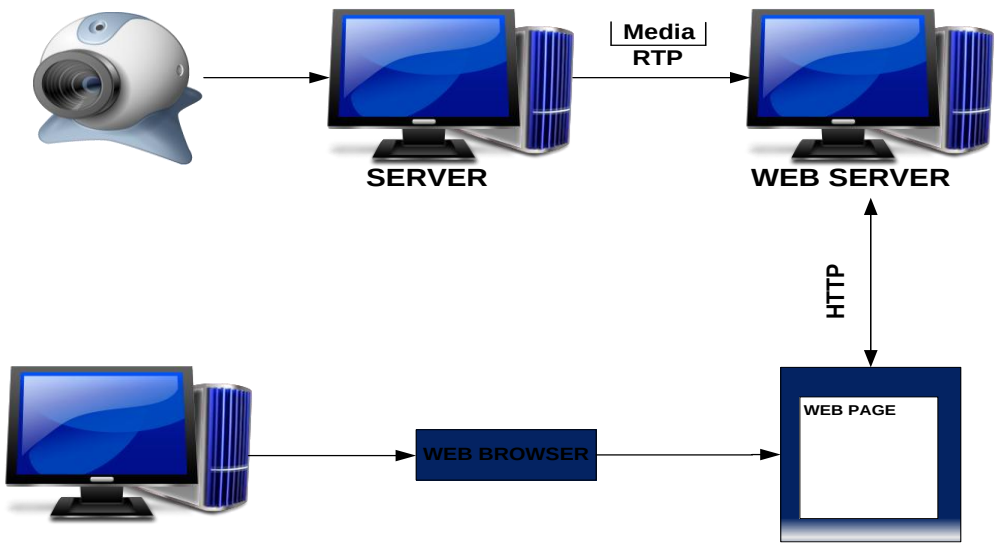
Apabila proses *preprocessing* dan *processing* dilakukan dengan benar maka komputer *server* sebagai penyedia layanan telah berhasil menyediakan sesi video *streaming* dan pengguna dapat mengakses video tersebut. Hal terpenting yang harus dilakukan oleh pengguna agar dapat mengakses video yang di*streaming*kan oleh *server* pada jaringan antara lain yaitu :

1. Harus menginstall *web browser* terlebih dahulu seperti *internet explorer, mozilla firefox, google chrome, opera,* dan *safari.*

2. Konfigurasi jaringan agar terhubung dengan komputer *server* (implementasi pada jaringan lokal).
3. Konektifitas internet agar dapat melakukan *live streaming* dari suatu *event* yang telah direkam oleh penyedia layanan.

Jika komputer *client* telah terhubung pada jaringan, hal selanjutnya yang harus dilakukan yaitu memasukkan URL *streaming* pada web browser untuk menyaksikan layanan *streaming* secara langsung dari suatu *event*. Sedangkan dalam implementasi pada jaringan internet komputer *server* harus mempunyai *IP Public* yang akan menjadi *Media Resource Locator* media *streaming* dan diinisialisasikan menjadi URL agar dapat diakses oleh pengguna layanan dimanapun dalam koneksi internet.

Adapun proses dari sistem *live streaming* ini mulai dari request *client* ke *server* hingga informasi video dapat di*streaming*kan adalah :



Gambar 3.5 Analisis Proses

3.1.4 Analisis Sistem Aplikasi Yang Dibangun

Sistem aplikasi penyedia layanan *streaming* pada komputer *server* dibangun menggunakan bahasa pemrograman *Ruby*. Aplikasi yang akan dibangun diharapkan mampu mentransmisikan video dari objek pada jaringan dan dapat menampilkan objek tersebut pada komputer *client* menggunakan *web browser*. Hal terpenting yang harus bisa dilakukan oleh aplikasi ini antara lain yaitu mampu mentransmisikan secara langsung video dari suatu objek yang sedang direkam oleh *server* pada jaringan agar dapat diakses oleh pengguna layanan.

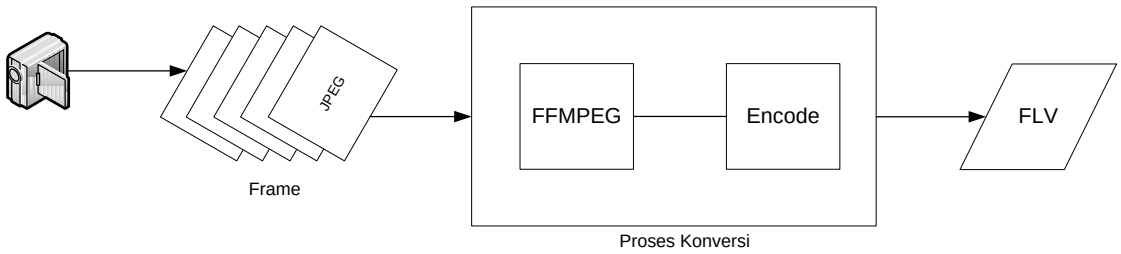
Pada komputer *server* sebagai penyedia layanan harus mempunyai media pustaka untuk proses deteksi perangkat masukan yang akan digunakan dalam proses *streaming* yang nantinya akan mempermudah aplikasi untuk mengenali perangkat multimedia yang telah terhubung. Jika perangkat video sudah berhasil terdeteksi, diharapkan aplikasi mampu menampilkan gambar dari objek. Proses selanjutnya yang menjadi proses inti yaitu proses mengirimkan objek yang sedang direkam secara *real time* pada jaringan, objek yang sedang direkam tersebut diharapkan dapat ditampilkan di *web browser* pada *client*. Jika semua proses yang telah disebutkan dapat dilakukan dengan baik dan benar tanpa terjadi kesalahan maka proses *streaming* sudah berhasil.

3.1.5 Analisis Video

Pada tahap ini terdapat beberapa hal yang akan dianalisis dalam pembangunan aplikasi multimedia dalam melakukan proses video *streaming*, antara lain :

3.1.5.1 Video *Codec*

Pengkodean video merupakan pengalihan kode analog menjadi kode digital agar video dapat dikirim dalam jaringan komputer. Hal ini dikenal dengan istilah *codec*, singkatan dari *compressor-decompressor*. Terdapat berbagai jenis *codec* yang dikembangkan untuk memampatkan/mengkompresi video agar dapat menggunakan *bandwidth* secara lebih hemat tanpa mengorbankan kualitas video (video yang keluar masih dapat dilihat dengan baik). Dalam penelitian ini menggunakan *ffmpeg* (*Fast Forward* MPEG) yang bertugas merekam dan mengkonversi video *streaming* secara *real-time* ke dalam berbagai format, dalam penelitian ini menggunakan format dengan ekstensi *FLV* (*flash video*). Teknik kompresi ini tidak mempunyai batas untuk *frame rate*, resolusi gambar, kualitas gambar atau target *bit rates* tetapi tergantung dari ketersediaan bandwidth dan kapasitas transfer dari kamera atau video *server*.



Gambar 3.6 Analisis Video *Codec*

#Pseudocode FLV *encoding*

```
void ff_flv_encode_picture_header(MpegEncContext *
s,int_picture_number)
{
      int format;
      align_put_bits(&s->pb);
      put_bits(&s->pb, 17, 1);
      put_bits(&s->pb, 5, (s->h263_flv-1));
      put_bits(&s->pb, 8, (((int64_t)s->picture_number * 30 * s-
      >avctx-time_base.num) /s->avctx->time_base.den) & 0xff);
```

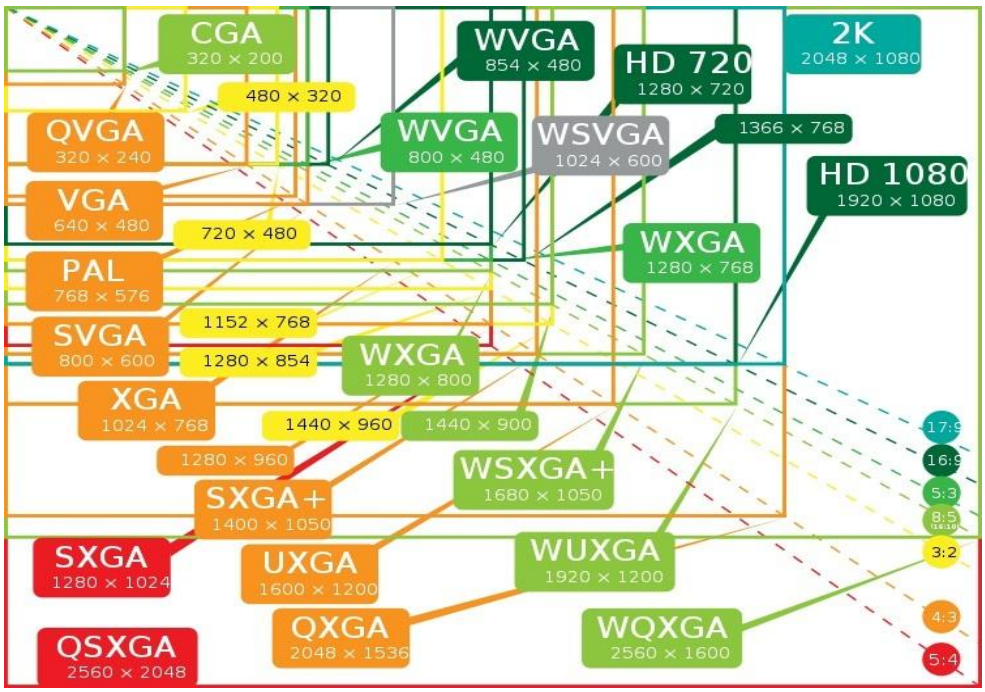
```
      if (s->width == 1024 && s->height == 768)
        format = 1;
      else if (s->width == 640 && s->height == 480)
        format = 2;
      else
        format = 3;

      put_bits(&s->pb, 2, format); /* PictureSize */
      put_bits(&s->pb, 1, s->pict_type == AV_PICTURE_TYPE_P);
      /* PictureType */
      put_bits(&s->pb, 1, s->qscale); /* Quantizer */
      put_bits(&s->pb, 1, 0); /* ExtraInformation */

      if(s->h263_aic){
        s->y_dc_scale_table=
          s->c_dc_scale_table= ff_aic_dc_scale_table;
      }else{
        s->y_dc_scale_table=
          s->c_dc_scale_table= ff_mpeg1_dc_scale_table;
      }
}
AVCodec ff_flv_encoder = {
      .name           = "flv",
      .type           = AVMEDIA_TYPE_VIDEO,
      .id             = CODEC_ID_FLV1,
      .priv_data_size = sizeof(MpegEncContext),
      .init           = MPV_encode_init,
      .encode         = MPV_encode_picture,
      .close          = MPV_encode_end,
      .pix_fmts= (const enum PixelFormat[]){PIX_FMT_YUV420P,
       PIX_FMT_NONE},
      .long_name= NULL_IF_CONFIG_SMALL("Flash Video (FLV) /
       Sorenson Spark / Sorenson H.263"),
      };
```

3.1.5.2 Resolusi Video

Pada penelitian ini resolusi video adalah hal yang sangat penting dalam mencapai tingkat kepuasan pengguna dalam melakukan akses video *streaming*, akan tetapi semakin tinggi resolusi video yang digunakan semakin besar pula penggunaan *bandwidth* sehingga perlu dilakukan proses pengkodean pada video menggunakan teknik konversi video.



Gambar 3.7 Jenis resolusi video

Dalam penelitian ini akan digunakan ukuran resolusi video jenis VGA (640x480) dan XGA (1024x768) dikarenakan media perekam yang digunakan yaitu *webcam* memiliki resolusi maksimal dalam proses pengambilan gambar sebesar 1024x768 (XGA).

3.1.6 Analisis Parameter *Quality of Service*

3.1.6.1 *Bandwidth*

Pada saat sekarang pengiriman paket-paket dalam sebuah jaringan dilakukan dengan upaya terbaik, dimana paket akan dikirimkan secara FIFO paket yang pertama sampai node akan diteruskan ke node berikutnya secara berurutan berdasarkan kedatangannya. Hal ini tidak ada salahnya, tetapi aplikasi-aplikasi yang berada di jaringan semakin lama semakin meminta *bandwidth* yang besar sedangkan harga *bandwidth* ini masih realtif mahal untuk saat sekarang ini. Sehingga dibutuhkan

langkah-langkah yang efektif yang harus dilakukan oleh seorang *network administrator* untuk menerapkan kebijakan-kebijakan tertentu agar performansi jaringan mencapai kinerja yang optimal. Salah satu cara untuk meningkatkan performansi jaringan yaitu dengan cara menerapkan metode *priority queing*. *Router* akan meneruskan paket sesuai dengan urutan kedatangan paket tersebut sehingga paket-paket tersebut akan berebut *bandwidth* yang ada. Dalam perancangan teknologi *streaming*, *bandwidth* merupakan suatu yang harus diperhitungkan agar dapat memenuhi kebutuhan yang dapat digunakan menjadi parameter untuk menghitung jumlah peralatan yang di butuhkan dalam suatu jaringan. Perhitungan ini juga sangat diperlukan dalam efisiensi jaringan dan biaya serta sebagai acuan pemenuhan kebutuhan untuk pengembangan di masa mendatang. Kebutuhan *bandwidth* akan sangat tergantung pada pemilihan *codec* yang digunakan agar proses *streaming* berjalan lancar sehingga pengguna dapat menerima informasi tanpa ada gangguan.

3.1.6.2 *Delay*

Delay adalah jarak waktu yang terjadi antar kedatangan tiap paket. Pengukuran ini bertujuan untuk mengetahui besarnya *delay* pada setiap kondisi sistem. Pada *Router* dan jaringan IP memiliki karakteristik khusus yang menyulitkan pengontrolan. Paket IP yang datang diperlakukan sama dan dilayani sesuai dengan urutan kedatangan. Ukuran paket IP sendiri bervariasi, sehingga *delay* dan variasi *delay* di jaringan menjadi besar dan tidak menentu. *Delay* dan variasi *delay* ini dapat berakibat buruk bagi kualitas video. Hal ini terjadi karena informasi video memiliki karakteristik *timing*. Agar jaringan IP ini dapat digunakan untuk menangani paket video, maka baik *delay* maupun variasinya harus dapat dikontrol dan ditekan serendah

mungkin. Besarnya *delay* maksimum yang direkomendasikan oleh ITU-T untuk aplikasi video adalah 150 ms, sedangkan *delay* maksimum dengan kualitas video yang masih dapat diterima pengguna adalah 250 ms.

3.1.6.3 *Jitter*

Jitter adalah variasi waktu kedatangan paket. *Jitter* dapat diukur dari waktu antara paket yang diterima sekarang dengan paket yang diterima sebelumnya. *Jitter* diakibatkan oleh lintasan tempuh yang berbeda-beda antar paket, variasi-variasi dalam panjang antrian, waktu pengolahan data, dan waktu penghimpunan ulang paket-paket di akhir perjalanan *jitter*. Secara umum *jitter* merupakan masalah *slow speed links*. Dalam layanan *streaming* multimedia, nilai *jitter* yang kecil dan cenderung stabil sangat dibutuhkan. Karena hal ini akan memberikan laju kedatangan paket yang bersifat kontinu, sehingga paket-paket yang datang ke dalam *buffer* tidak berlebih atau tidak kurang. Cisco menetapkan bahwa *jitter* untuk komunikasi *real-time* tidak boleh melebihi 30 ms.

3.1.7 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional disini membahas kendala pada pelayanan atau fungsi sistem, yaitu :

1. *Availability*: ketersediaan aplikasi, misalnya harus terus menerus beroperasi melakukan proses perekaman suatu event selama hari kerja tanpa adanya kegagalan pada sistem, maka sistem ini dapat di akses setiap hari, selama komputer *client* terhubung ke *server*.
2. *Reliability:* keandalan, misalnya tidak pernah boleh gagal atau kegagalan yang ditolerir dibatasi hingga berapa persen, sehingga harus dipikirkan solusi atau

tolerant fault. Pada sistem ini, dilihat dari aspek *reliability* nya karena berupa aplikasi *online,* kegagalan yang mungkin di tolerir adalah delay video yang di*streaming*kan untuk sampai ke user, jika delay video terlalu besar atau lama maka akan dilakukan konfigurasi ulang pada *server* hal-hal apa saja yang mempengaruhi, apakah teknik kompresi atau bandwidth.

3. *Ergonomy*: kenyamanan pemakaian bagi pengguna.

   Sistem ini akan di bangun sedemikian rupa agar *user friendly* dan *easy learning,* sehingga *user* akan sangat nyaman untuk menggunakan aplikasi ini.

   Selain membahas kendala pada pelayanan atau fungsi sistem, terdapat kebutuhan pendukung sistem yang akan dijalankan.

3.1.7.1 Analisis Kebutuhan Perangkat Keras

Pada sistem *live streaming* ini, perangkat keras yang digunakan terdiri dari dua jenis yaitu perangkat keras *server* dan perangkat keras *client*. Adapun spesifikasi yang digunakan dalam penelitian ini yaitu :

1. Perangkat Keras *Server*

   Minimum requirement :

   a. Processor : Intel Pentium Dual Core atau sekelasnya
   b. Memory : 2 GB
   c. VGA : 256 MB
   d. Hardisk : 500 GB
   e. Webcam

2. Perangkat Keras *Client*

   Minimum requirement :

   a. Processor : Intel Pentium Dual Core atau sekelasnya
   b. Memory : 1 GB
   c. VGA : 128 MB
   d. Hardisk : 40 GB

3.1.7.2 Analisis Kebutuhan Perangkat Lunak

Kebutuhan akan perangkat lunak yang digunakan dalam perancangan sistem Live Streaming ini yaitu :

1. Perangkat Lunak *Server*
   a. Sistem operasi Linux.
   b. NetBeans dalam membangun aplikasi *server streaming*.
   c. *Chatango* sebagai aplikasi *livechat*.
   d. Adobe Photoshop CS3 sebagai editor gambar.
   e. Wireshark sebagai aplikasi paket *sniffing*
2. Perangkat Lunak *Client*
   a. Web browser : Internet Explorer, Mozilla firefox, Opera, Google chrome, dan Safari.
   b. Flash player untuk plugin video.

3.1.7.3 Analisis Kebutuhan User

Untuk menggunakan layanan *Live Streaming* ini dibutuhkan spesifikasi user yang dapat menjalankan aplikasi ini, baik dalam sisi *client* maupun administrator. Adapun spesifikasi user yang dibutuhkan :

Tabel 3.1 Spesifikasi User

Syarat	Administrator	Client
Mengerti konsep jaringan	✓	
Menguasai aplikasi multimedia komputer	✓	✓
Bisa menggunakan aplikasi internet seperti web browser.	✓	✓

3.1.8 Analisis Kebutuhan Fungsional

Dengan memodelkan proses menggunakan Activity Diagram, Use Case Diagram, Sequence Diagram dan Class Diagram, penjelasannya sebagai berikut :

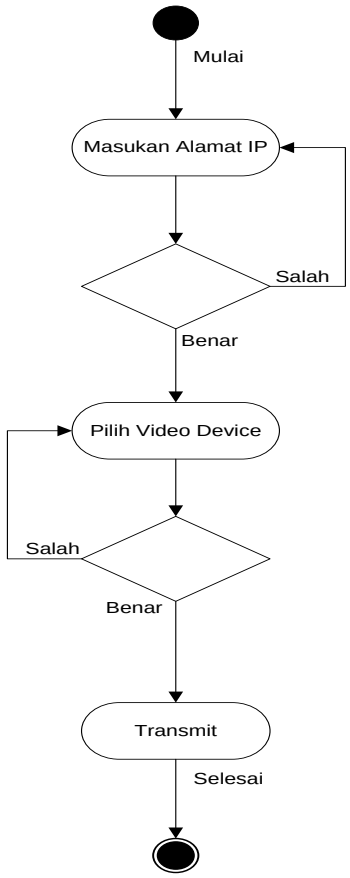
1. *Activity Diagram* adalah diagram yang menggambarkan sequence dari aktivitas - aktivitas dalam sebuah proses, termasuk sekuensial dan paralel aktivitas, dan tujuan yang telah dibuat.Sebuah activity diagram biasanya dibuat untuk satu use case dan memperlihatkan perbedaan skenario - skenario yang memungkinkan.
2. *Use Case Diagram* menjelaskan sistem secara fungsional yang terlihat user. Biasanya dibuat pada awal pengembangan dalam menggambarkan kebutuhan sistem.
3. *Sequence Diagram* bisa memperlihatkan sebuah ilustrasi yang sukses dari interaksi – interaksi diantara class atau objek. *Sequence Diagram* kadang – kadang digunakan untuk mengilustrasikan mendeskripsikan proses – proses dalam use case skenario.
4. *Class Diagram* Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain

berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

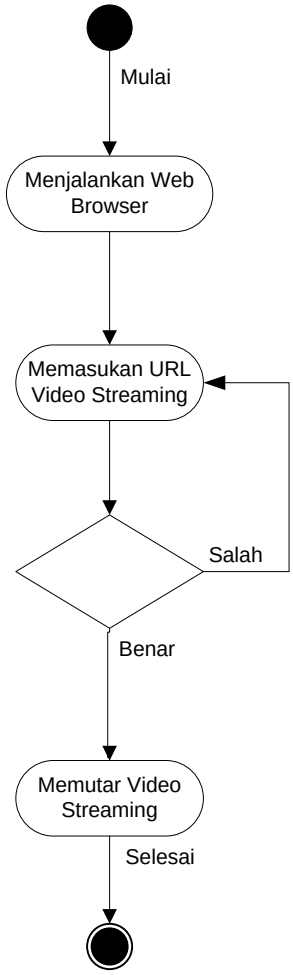
Penjelasan tentang syarat – syarat fungsional yang terdapat pada sistem ini yaitu :

3.1.8.1 *Activity* Diagram

Diagram yang menggambarkan aktivitas - aktivitas dalam sebuah proses, termasuk sekuensial dan paralel aktivitas, dan tujuan yang telah dibuat. Dalam hal ini menjelaskan aktifitas *server* dalam melakukan proses *streaming* video dari suatu *event* secara langsung dan aktifitas *client* dalam mengkases video *streaming*.



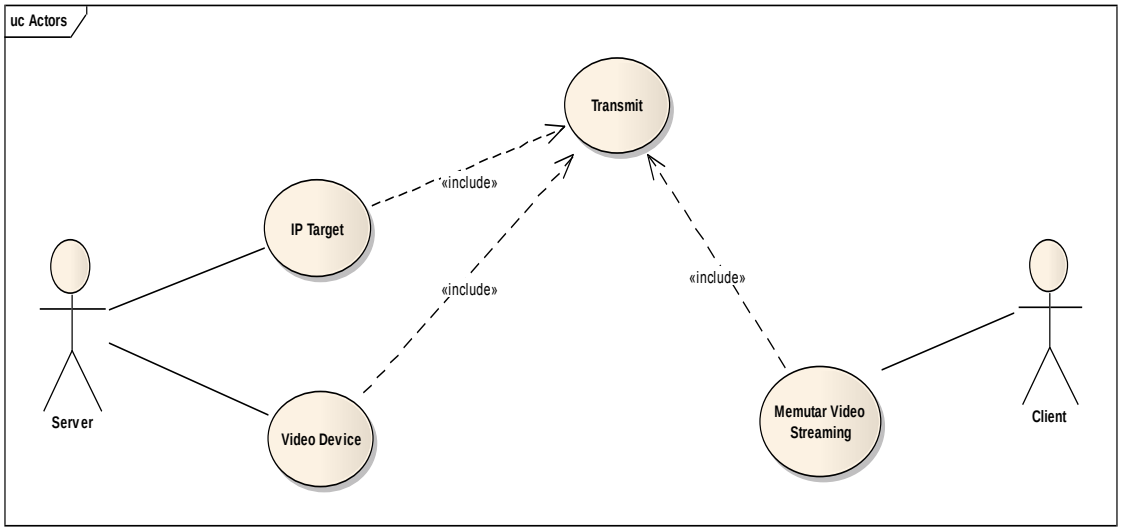
Gambar 3.8 *Activity* Diagram *Server*



Gambar 3.9 *Activity* Diagram *Client*

3.1.8.2 *Usecase* Diagram Sistem

Analisis perilaku dalam Aplikasi *Live Streaming* menggunakan protokol RTP yaitu berupa analisa *Use Case*. Terdapat beberapa *use case* yang didalamnya mencangkup proses – proses yang ada dalam Sistem Aplikasi *Live Streaming*. Use case diagram, sistem skenario dan activity diagram sebagai berikut:

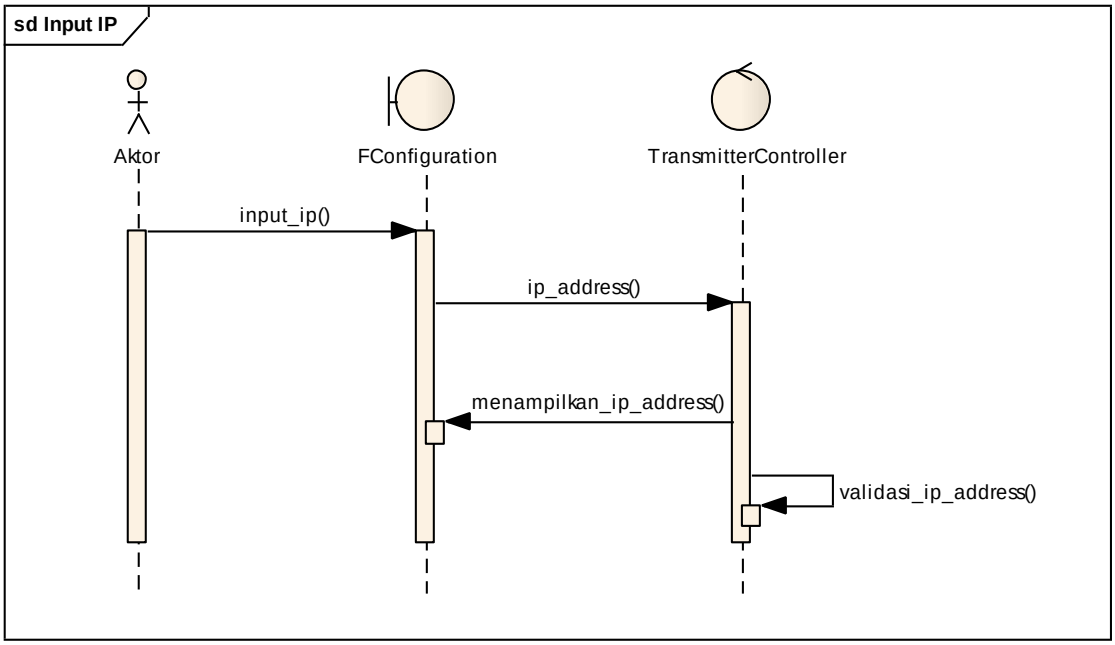


Gambar 3.10 *Usecase* Diagram *Live Streaming*

1. *Usecase* Skenario IP Target

Tabel 3.2 *Usecase* Skenario IP Target

IDENTIFIKASI	
Nama	IP Target
Deskripsi	Proses pada aplikasi untuk mengirimkan media *stream* menuju IP tujuan.
Aktor	*Server*
Pre-condition	Aktor telah menjalankan aplikasi *streaming* pada komputer *server*.
Post-condition	Alamat IP dapat dikenali
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Mengisi alamat IP	2. Menerima *request* dari aktor
	3. Melakukan validasi terhadap alamat yang diisikan.

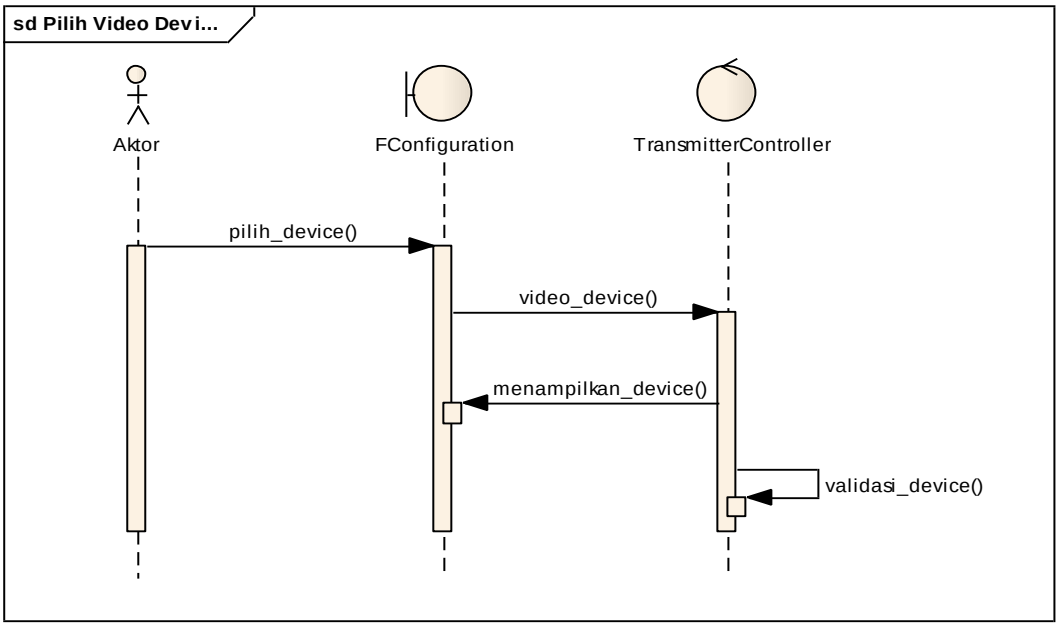


Gambar 3.11 *Sequence* Diagram IP Target

2. *Usecase* Skenario Video Device

Tabel 3.3 *Usecase* Skenario Video *Device*

IDENTIFIKASI	
Nama	Video *Device*
Deskripsi	Proses untuk memilih media masukan berupa *webcam* yang akan menjadi media *streaming*
Aktor	*Server*
Pre-condition	Aktor telah menjalankan aplikasi *streaming* pada komputer *server*.
Post-condition	*Webcam* mampu menampilkan objek yang sedang direkam.
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Memilih *device*	2. Menentukan *device* yang dipilih



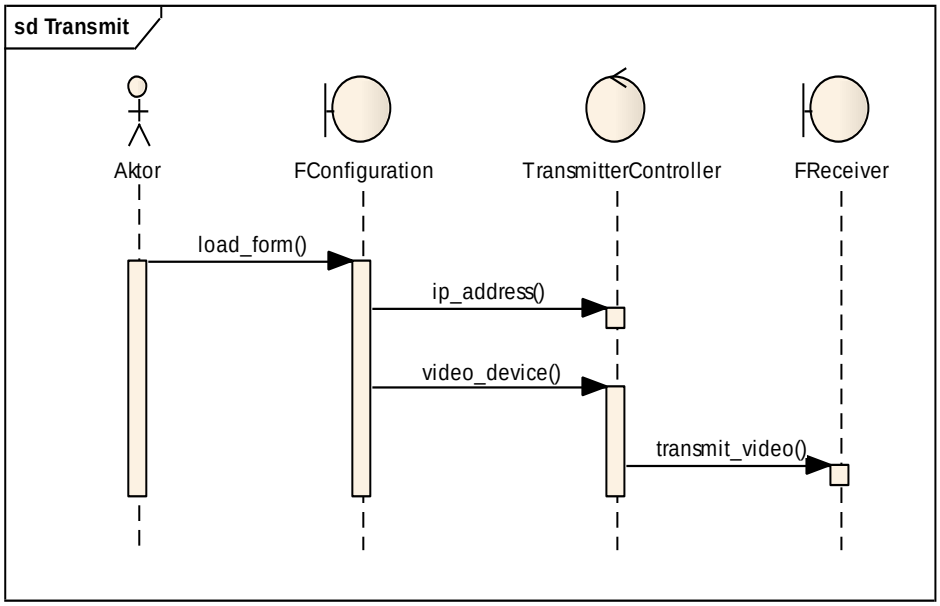
Gambar 3.12 *Sequence* Diagram Video *Device*

3. *Usecase* Skenario Transmit

Tabel 3.4 *Usecase* Skenario Transmit

IDENTIFIKASI	
Nama	Transmit
Deskripsi	Proses untuk mentransmisikan media stream berupa video dari objek secara digital.
Aktor	*Server*
Pre-condition	Aktor telah menjalankan aplikasi *streaming* pada komputer *server*.
Post-condition	Objek yang sedang direkam dapat ditampilkan secara langsung pada web browser disisi pengguna.
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Klik tombol Transmit	2. Menerima request dari aktor

	3. Mentransmisikan media *streaming* berupa video secara langsung pada jaringan, sehingga pengguna dapat mengakses video tersebut secara langsung.



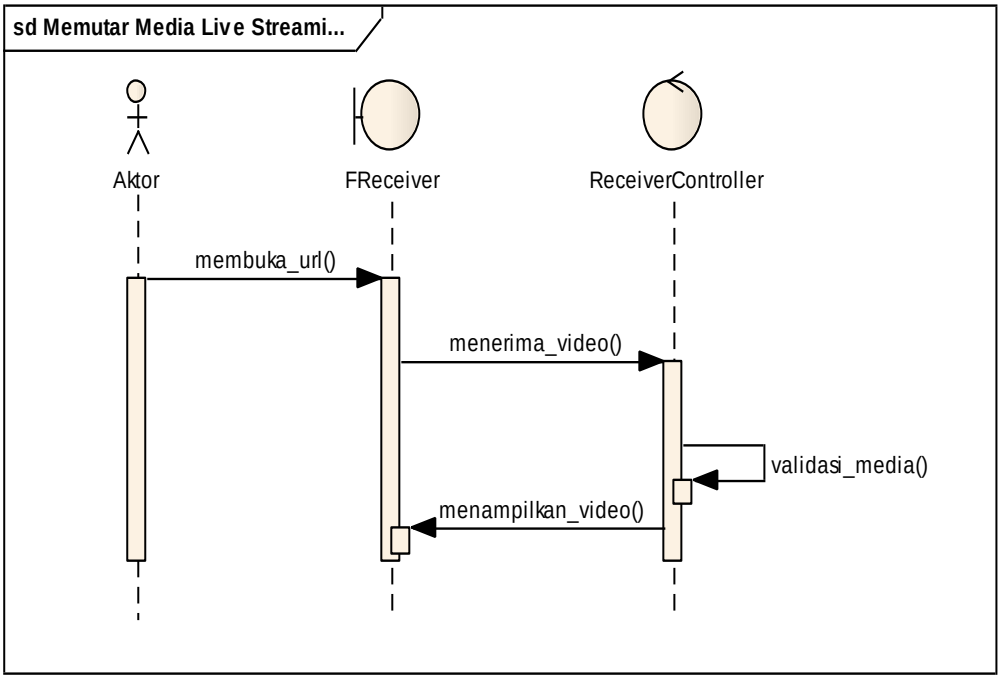
Gambar 3.13 Sequence Diagram Transmit

4. *Usecase* Skenario Memutar Video *Live Streaming*

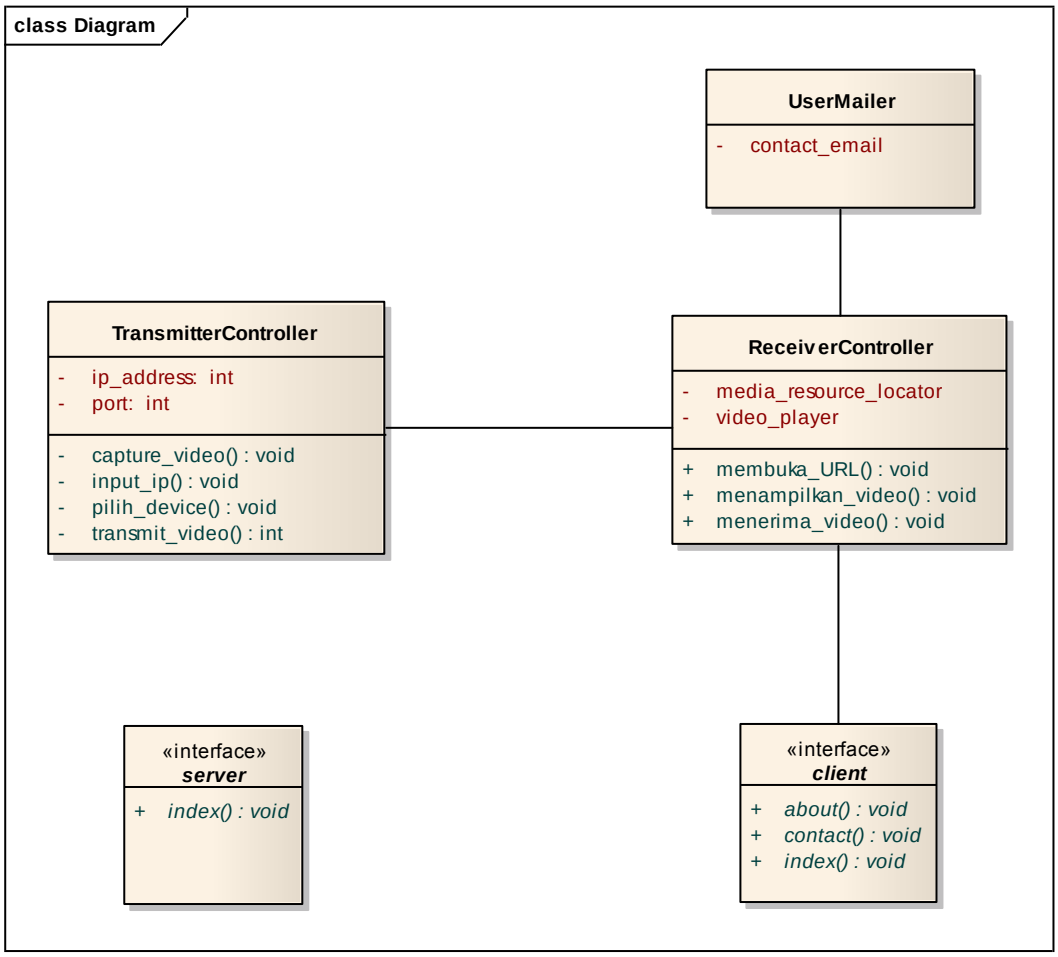
Tabel 3.5 *Usecase* Skenario Memutar Video *Live Streaming*

IDENTIFIKASI	
Nama	Usecase Memutar video *streaming*
Deskripsi	Jika *server* telah men*streaming*kan video pada jaringan maka pengguna dapat mengaksesnya melalui web browser.
Aktor	*Client*

Pre-condition	*Server* telah mentransmisikan video *streaming* pada jaringan dan aktor telah menjalankan web browser.
Post-condition	Memutar media *streaming* menggunakan web browser.
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Membuka URL video *streaming*	2. Menerima request dari aktor
	3. Memutar media *streaming*.



Gambar 3.14 *Sequence* Diagram Memutar Video *Live Streaming*



Gambar 3.15 *Class* Diagram Aplikasi Video *Streaming*

3.2 Perancangan Sistem

Perancangan akan dimulai setelah tahap analisis terhadap sistem selesai dilakukan. Perancangan dapat didefinisikan sebagai proses aplikasi berbagai teknik dan prinsip bagi tujuan pendefinisian suatu perangkat, suatu proses atau sistem dalam detail yang memadai untuk memungkinkan realisasi fisiknya. Perancangan digambarkan sebagai proses multi-langkah dimana representasi struktur data, struktur program, dan karakteristik interface, disintesis dari persyaratan informasi.

3.2.1 Perancangan Server

Dalam membangun aplikasi video *streaming* ini terdapat beberapa langkah-langkah yang harus dilakukan dalam membangun sebuah komputer server sebagai penyedia layanan. Langkah-langkah yang harus dilakukan antara lain :

1. Proses instalasi sistem operasi adalah hal pertama yang harus dilakukan, Linux Ubuntu 11.04 diinstalkan ke komputer *server*.
2. Instalasi aplikasi pendukung *streaming server*

```
$ sudo apt-get install update
$ sudo apt-get install vlc
$ sudo apt-get install ffmpeg
```

3. Instalasi aplikasi *webcam*

```
$ sudo apt-get install cheese
```

4. Instalasi aplikasi *Ruby Virtual Machine* (RVM) sebagai *streaming server*

```
$ sudo apt-get install curl git-core
$ bash < <( curl https://rvm.beginrescueend.com/releases/rvm-
  install-head )
# Tambahkan script dibawah ini pada file .bashrc
# Ganti baris kode seperti dibawah ini
[ -z "$PS1" ] && return
# Dengan
if [[ -n "$SP1" ]]; then
# Setelah itu tambahkan kode berikut pada baris terakhir
if [[ -s $HOME/.rvm/scripts/rvm ]] ; then source
$HOME/.rvm/scripts/rvm ; fi
```

```
fi
$ sudo apt-get install build-essential bison openssl
$ sudo apt-get install libruby1.8 zliblg-dev libssl-dev
libreadline5-dev build essential
$ sudo apt-get install 1.9.2-head
$ rvm –default 1.9.2-head
$ gem install rails
$ gem install sqlite3-ruby
```

5. Instalasi *Netbeans* sebagai aplikasi bahasa pemrograman dalam membangun aplikasi penyedia layanan video *streaming.*

```
$ sudo apt-get install sun-java6-jdk
# Download terlebih dahulu file netbean.sh
$ chmod 777 netbeans-6.9.1-ml-linux.sh
$ sudo ln netbeans-6.9.1-ml-linux.sh
$ sudo ./netbeans-6.9.1-ml-linux.sh
# Tunggu sampai proses instalasi selesai
```

6. Konfigurasi IP *Forwading* PC *router*

IP *public* sangat dibutuhkan karena akan diimplementasikan dalam jaringan internet, IP *public* akan diakses oleh pengguna oleh karena itu fasilitas IP *Forwading* harus diaktifkan dengan kode berikut :

`$ sysctl –w net.ipv4.ip_forward=1` (nilai 1 aktif, nilai 0 non aktif)

7. Membuat file *Transmitter* sebagai *server* video *streaming* dan file *Receiver* sebagai penerima video *streaming* bagi pengguna.

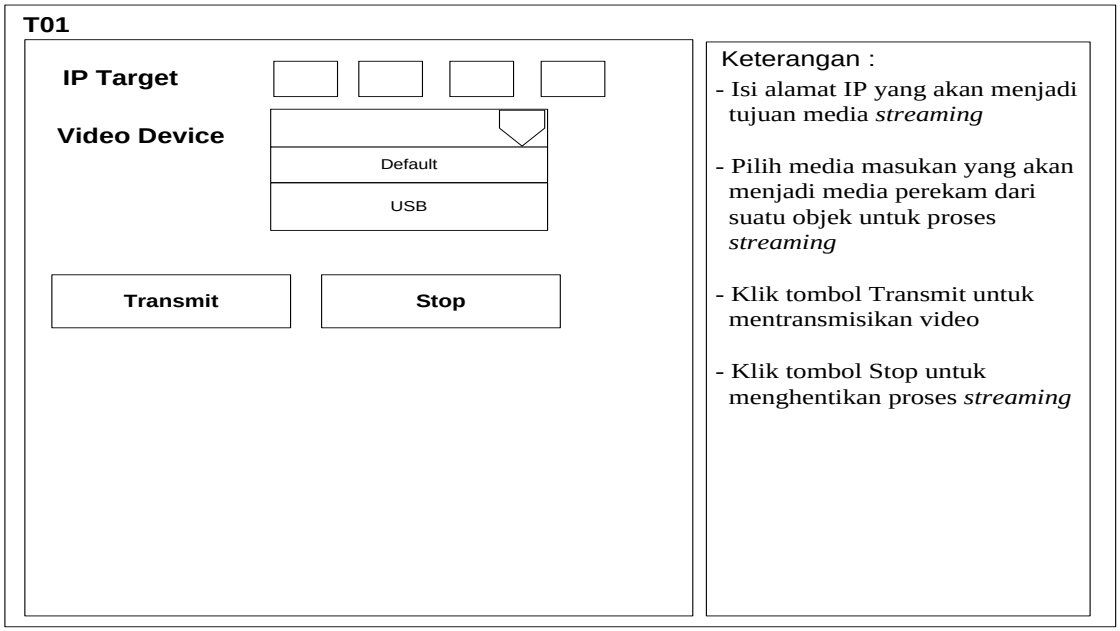
Video *streaming* dilakukan dengan mengambil gambar dari *webcam* yang terhubung dengan komputer *server*. Video *streaming* ditampilkan dengan menggunakan aplikasi multimedia di sisi *server* dan *web* browser di sisi *client*. Pada aplikasi tersebut *server* menentukan alamat IP yang akan menjadi tujuan objek yang akan di*streamingkan* dan dapat memilih perangkat video yang akan menjadi media masukan untuk proses *streaming*, proses pemilihan perangkat hanya dapat memilih salah satu perangkat yang terhubung pada komputer.

3.2.2 Perancangan Client

Jika *server streaming* sudah dibangun maka hal selanjutnya adalah merancang tampilan web untuk *client* agar bisa melakukan proses *streaming* dari *server* menggunakan *web browser*. Antarmuka pada *client* cukup sederhana, hanya menampilkan video yang di*streaming*kan oleh *server*, navigasi hanya berupa *button pause, play, dan fullscreen.* Hal ini memang benar adanya dikarenakan proses *live streaming* tidak bisa dilakukan proses *backward* dan *forward* seperti pada *video on demand*.

3.3 Perancangan Antarmuka

3.3.1 Perancangan Antarmuka Aplikasi *Server*



Gambar 3.16 Perancangan Antarmuka Menu Utama

3.3.2 Antarmuka Tampilan *Web Client*



Gambar 3.17 Antarmuka Menu Utama *Client*