PERBANDINGAN KINERJA PROTOKOL RFCOMM DAN OBEX UNTUK IMPLEMENTASI *PSEUDO VIDEO STREAMING* PADA *SMARTPHONE* MELALUI BLUETOOTH

SKRIPSI

Oleh

FEBRI KURNIAWAN WIDYANTORO
04 03 03 039 X

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GANJIL 2007/2008

PERBANDINGAN KINERJA PROTOKOL RFCOMM DAN OBEX UNTUK IMPLEMENTASI *PSEUDO VIDEO STREAMING* PADA *SMARTPHONE* MELALUI BLUETOOTH

SKRIPSI

Oleh

FEBRI KURNIAWAN WIDYANTORO
04 03 03 039 X

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GANJIL 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

PERBANDINGAN KINERJA PROTOKOL
RFCOMM DAN OBEX UNTUK
IMPLEMENTASI *PSEUDO VIDEO STREAMING* PADA *SMARTPHONE*
MELALUI BLUETOOTH

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 7 Januari 2008

Febri Kurniawan Widyantoro
NPM. 04 03 03 039 X

PENGESAHAN

Skripsi dengan judul :

PERBANDINGAN KINERJA PROTOKOL RFCOMM DAN OBEX UNTUK IMPLEMENTASI *PSEUDO VIDEO STREAMING* PADA *SMARTPHONE* MELALUI BLUETOOTH

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia. Skripsi ini telah diujikan pada sidang ujian skripsi pada tanggal 4 Januari 2008 dan dinyatakan memenuhi syarat/sah sebagai skripsi pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, 7 Januari 2008

Dosen Pembimbing

Dr-Ing. Kalamullah Ramli, M.Eng.

NIP. 132 092 429

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada :

Dr-Ing. Kalamullah Ramli, M. Eng

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Febri Kurniawan Widyantoro
NPM 04 03 03 039 X
Departemen Teknik Elektro

Dosen Pembimbing
Dr-Ing. Kalamullah Ramli, M. Eng.

PERBANDINGAN KINERJA PROTOKOL RFCOMM DAN OBEX UNTUK IMPLEMENTASI *PSEUDO VIDEO STREAMING* PADA *SMARTPHONE* MELALUI BLUETOOTH

ABSTRAK

Bluetooth merupakan teknologi nirkabel yang menjanjikan bagi peralatan elektronik dan divais *mobile*. Teknologi ini beroperasi pada frekuensi 2,4 GHz dan berdaerah jangkauan hingga 100 meter. Dibandingkan dengan teknologi nirkabel lainnya, seperti 802.11 dan IrDA, Bluetooth mempunyai keunggulan dalam efisensi, biaya dan konsumsi daya yang lebih kecil, dan juga dapat menyediakan layanan *ad hoc*. Kecepatan *data rate* hingga 732 Kbps memungkinkan tak hanya transmisi web dan email, tetapi juga konferensi video dan *streaming* siarang langsung. Namun, risiko kendala interferensi, keterbatasan jangkauan, dan kehilangan data menjadi ancaman tersendiri. Belum banyak penelitian mengenai *video streaming* melalui Bluetooth yang telah dilakukan, hal itu menjadi sebuah tantangan menarik untuk mengembangkannya lebih lanjut.

Skripsi ini memfokuskan pada perbandingan kinerja implementasi *pseudo video streaming* menggunakan RFCOMM dan OBEX Bluetooth protokol pada sisi *client*, khususnya *user interface* pada *smartphone.* Layanan ini diberikan dari *server* ke *client (smartphone)*. Aplikasi didesain dan dibangun menggunakan *platform* J2ME dan *Netbeans Mobility* 5.5.1 sebagai *Integrated Development Environment* (IDE).

Hasil kinerja menunjukan protokol RFCOMM lebih baik daripada protokol Obex diukur berdasarkan waktu pencarian divais bluetooth lainnya. Sedangkan untuk transfer file kinerja Obex lebih baik ketimbang RFCOMM.

Kata Kunci : *video streaming*, bluetooth, J2ME, *smartphone*

Febri Kurniawan Widyantoro
NPM 04 03 03 039 X
Electrical Engineering Department

Counselor
Dr. Ing.Kalamullah Ramli, M. Eng.

PERFORMANCE FOR THE IMPLEMENTATION OF PESUDO VIDEO STREAMING APPLICATION ON SMARTPHONE USING BLUETOOTH CONNECTION

ABSTRACT

Bluetooth is the promising wireless technology for electronic and mobile devices. It operates on 2,4 GHz frequency and up to 100 meter range area. In comparison with other wireless technology, such as 802.11 and IrDA, Bluetooth has advantage in efieciency, lower cost and power consumption, and also can provide *ad hoc* services. Speed data rate up to 732 kbps allow not only transmission of the web and e-mail, but also audio/video conferencing dan streaming of live shows. But, the problem's risk of interference, limitation range area, and data loss be a threat apart. There are not many research yet have been done about video streaming over Bluetooth, and for the further this is an opportunity and challenge for the researcher to develop it.

This final project focuses on the comparison of pseudo video streaming implementation using OBEX and RFCOMM Bluetooth protocol on client side, especially user interface at smartphone. This service is given by computer server to client (smartphone). Application is designed dan developed using Java 2 Micro Edition (J2ME) platform and Netbeans Mobility 5.5.1 as Integrated Development Environment (IDE).

The performance show that the RFCOMM is better than Obex on service discovery. But for transfer file data the Obex is much better that RFCOMM.

Keywords : video streaming, bluetooth, J2ME, smartphone.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

3GPP	3rd Generation Partnership Project
ACL	Asynchronous ConnectionLess
API	Application Programming Interface
Codec	Compression/decompression
GUI	Graphical User Interface
HCI	Host Controller Interface
IDE	Integrated Development Environment
ISM	Industrial, Scientific and Medical
J2ME	Java 2nd Micro Edition
J2SE	Java 2nd Standard Edition
JABWT	Java APIs For Bluetooth Wireless Technology
L2CAP	Logical Link and Control Adaptation Protocol
MMAPI	Mobile Media API
MPEG	Motion Picture Expert Group
SCO	Synchronous Connection Oriented
QoS	Quality of Service

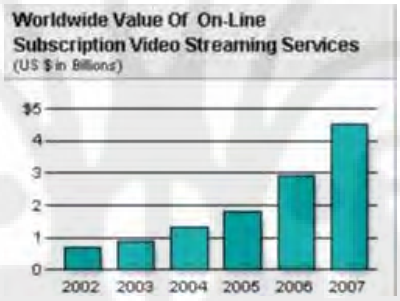
DAFTAR ISTILAH

Bluetooth	Teknologi wireless yang beroperasi pada frekuensi 2,4 GHz
Client	Divais bergerak yang dilayani oleh *server* dan dioperasikan oleh pengguna (*smartphone*, PDA)
Server	Komputer pusat yang mengatur sistem
Smartphone	Divais telepon yang dilengkapi sistem operasi terbuka (Symbian, Linux, Pocket PC, Palm) dan memori yang cukup besar sehingga mampu menjalankan beragam fungsi
Video Streaming	Video yang dijalankan secara terus menerus

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Layanan informasi multimedia, terutama video, semakin populer, baik di kalangan dunia pendidikan, hiburan, periklanan, maupun ekonomi dan bisnis. Hal ini dikarenakan media audio visual ini bersifat lebih baik dan mudah dalam menyampaikan informasi. Salah satu bentuk bentuknya yaitu layanan *video streaming* yang cukup potensial, dimana jumlah penggunanya terus meningkat setiap tahun. Hal ini sesuai dengan hasil analisa nilai pendapatan dari pengguna *video streaming* oleh In-Stat/MDR, sebuah lembaga peneliti dan analis di Amerika Serikat, pada gambar 1.1.



Gambar 1.1 Analisa nilai pendapatan dari pengguna layanan *video stream* [15]

Layanan *video streaming* merupakan layanan akses video yang dapat dijalankan tanpa harus men-*download* file secara utuh terlebih dahulu, sebagai hasil aliran transmisi data video yang telah mengalami mekanisme paketisasi, fragmentasi, dan kompresi untuk memenuhi tuntutan efisiensi beban proses.

Layanan *pseudo video streaming* melalui teknologi Bluetooth adalah sebuah alternatif layanan yang menarik untuk dieksplorasi dan dikembangkan. Bluetooth merupakan standar teknologi komunikasi *wireless mobile* untuk area lokal yang menawarkan fitur rendah biaya dan konsumsi daya. Teknologi yang didesain berukuran mini ini beroperasi pada frekuensi 2,4 GHz dan telah banyak tersedia di sebagian besar perangkat *mobile*, seperti telepon selular, *smartphone*, PDA (*Personal Digital Assistant*), *tablet* PC, *notebook*, *headset*, serta mampu

menyediakan koneksi data dan suara secara *real time*. Penggunaan protokol RCOMM dan OBEX dapat menghasilkan performa yang berbeda.

1.2 PERUMUSAN MASALAH

Implementasi aplikasi *pseudo video streaming* pada *smartphone* sebagai sisi *client* melalui teknologi koneksi Bluetooth adalah sebuah alternatif penyelenggaraan layanan *pseudo video streaming* yang menarik. Implementasi diujicobakan menggunakan protokol RFCOMM dan OBEX untuk membandingkan kinerja antara keduanya.

1.3 TUJUAN PENELITIAN

Skripsi ini ditujukan untuk merancang dan mengimplementasikan aplikasi *video streaming* melalui teknologi Bluetooth dengan basis J2ME pada *smartphone*. Adapun, tujuan khusus penulisan adalah perbandingan kinerja implementasi *user interface* sisi *smartphone* sistem layanan *pseudo video streaming over Bluetooth* pada protokol RFCOMM dan OBEX.

1.4 BATASAN MASALAH

Permasalahan yang dibahas pada skripsi ini terbatas pada perbandingan kinerja aplikasi *pseudo video streaming* menggunakan protokol RFCOMM dan OBEX Bluetooth yang diimplementasikan menggunakan J2ME.

Implementasi *video streaming* yang dilakukan mempunyai sumber *pseudo streaming server* berupa komputer, dan tujuan *streaming client* berupa *smartphone.* Aplikasi *user interface* pada *smartphone* yang dibuat dapat memainkan video tanpa dapat melakukan proses *editing* maupun *rewind*, adapun video tersebut telah direkam sebelumnya. Format video yang dapat diakses yaitu .3gpp. *Server* dimaksudkan hanya dapat menangani sebuah *client.*

1.5 METODOLOGI PENELITIAN

Metodologi penelitian yang digunakan yaitu :

a) Studi literatur

Mempelajari informasi dari jurnal, buku, dan artikel-artikel yang berkaitan dengan aplikasi yang dibuat.

b) Perancangan dan pembangunan aplikasi *video streaming* berkoneksi *Bluetooth* pada sisi *client*, dalam hal ini *smartphone*, menggunakan metode berorientasi objek. Tahapan yang tercakup didalamnya adalah analisa dan identifikasi kebutuhan serta perancangan sistem, dilanjutkan dengan pengimplementasiannya pada bahasa pemrograman J2ME.
c) Pengujian kinerja protokol Bluetooth RFCOMM dan OBEX untuk implementasi aplikasi.
d) Penarikan kesimpulan.

1.6 SISTEMATIKA PENULISAN

Penulisan skripsi ini dibagi menjadi 5 (lima) bab, dengan sistematika penulisan sebagai berikut :

BAB 1 PENDAHULUAN

Membahas mengenai latar belakang penulisan, perumusan masalah, tujuan penelitian, pembatasan masalah, metodologi pembahasan, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Berisi tentang konsep *video streaming*, koneksi Bluetooth, dan J2ME sebagai *tool* yang digunakan dalam pembuatan aplikasi.

BAB 3 PERANCANGAN DAN ANALISA SISTEM KOMUNIKASI APLIKASI PADA SISI *CLIENT*

Menjelaskan tentang analisa perancangan dan kebutuhan sistem dengan menggunakan metode berorientasi objek.

BAB 4 IMPLEMENTASI dan ANALISA

Membahas tentang implementasi perancangan aplikasi sisi *client*, dan analisa hasil uji kinerja sistem pada koneksi protokol Bluetooth RFCOMM dan OBEX.

BAB 5 KESIMPULAN

Berisi mengenai kesimpulan atas pembahasan yang telah dilakukan dalam penulisan skripsi ini.

BAB II

LANDASAN TEORI

2.1 *VIDEO STREAMING*

2.1.1 Definisi

Streaming dapat didefinisikan sebagai metode pengiriman data secara terus menerus dan kontinu, dengan disertai proses kompresi terhadap file audio dan video agar lebih efisien. Metode ini dilakukan dari *server* ke *client* setelah mengalami paketisasi, yakni data pada file *streaming* dibagi menjadi beberapa paket kecil. Paket-paket tersebut ditransfer melalui sebuah aliran informasi ke pengguna akhir sehingga memungkinkan *decode* dan *playback* video dilakukan oleh penerima tanpa harus menunggu seluruh video terkirim.

Layanan *video streaming* dapat diklasifikasi menjadi dua metode, yaitu *on-demand* dan *live*. Metode *live video streaming* menyelenggarakan layanan yang disiarkan secara *broadcast* pada saat itu juga, seperti pada acara radio dan televisi. Sedangkan, yang termasuk dalam kategori *on-demand* adalah akses musik dan video berdasarkan permintaan.

Permasalahan yang dihadapi oleh *video streaming* tradisional umumnya adalah *bandwidth, delay, packet loss,* keterbatasan radius jangkauan, dan interferensi dengan divais lainnya maupun sumber luar. Beberapa masalah dasar dalam *video streaming* adalah [1]:

- *Bandwidth*

  Jika pengirim mengirimkan data lebih cepat dibandingkan ketersediaan *bandwdith* yang ada, maka akan terjadi *congestion* (kemacetan) pada jaringan, *packet loss*, dan buruknya kualitas video. Tetapi, apabila pengiriman paket data dilakukan lebih lambat daripada *bandwidth* yang tersedia, maka kualitas video yang diterima pengguna kurang optimal. Salah satu cara mengatasinya adalah melakukan estimasi ketersedian *bandwidth* yang dimiliki terhadap *bit rate* video yang akan ditransmisikan.

- *Delay Jitter*

Delay jitter merupakan permasalahan akibat terlambat datangnya *frame* sehingga menyulitkan rekonstruksi video yang diterima. Penerima idealnya harus menerima, *decode*, dan menampilkan *frame* dengan *rate* yang konstan.

- *Loss Rate*

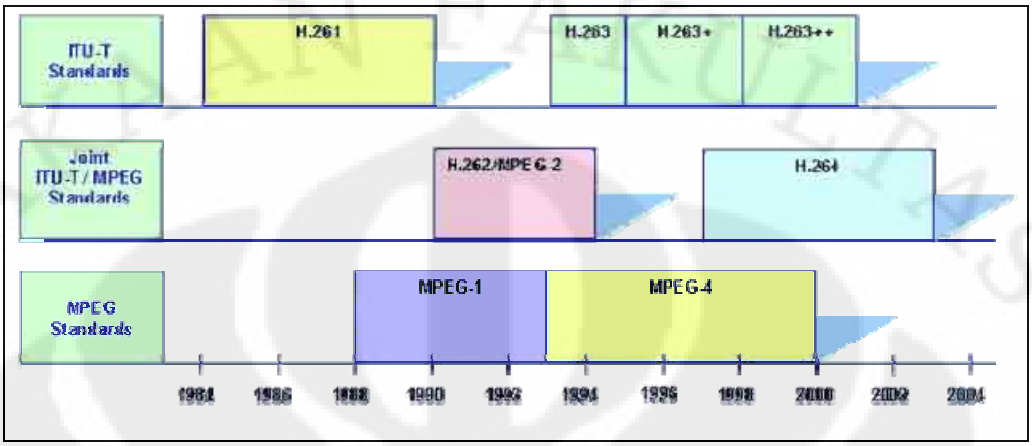
  Pada jaringan nirkabel, *loss rate* dapat disebabkan oleh *bit error*. *Loss rate* dapat menimbulkan kerusakan kualitas video hasil rekonstruksi. Untuk mengatasinya, sistem *video streaming* dapat didesain dengan fasilitas pengendalian *error rate*..

2.1.2 *Encode / Decode*

Proses *encoding/decode* video dalam konteks *streaming* meliputi proses pengkodean dan kompresi/dekompresi (*compression/decompression*). Proses *compression/decompression,* atau yang dikenal sebagai singkatan : CODEC, merupakan suatu metode atau algoritma yang terdapat pada sebuah *streaming player* yang berfungsi untuk melakukan proses pengkompresan dan pendekompresan pada file media *streaming* [2]. Metode kompresi dilakukan antara lain dengan menghilangkan informasi video yang berlebihan, rusak, dan atau kurang penting sebelum ditransmisikan sehingga file yang dihasilkan berukuran lebih kecil dan efisien. Hal ini dilatarbelakangi oleh keterbatasan *bandwidth* yang tersedia pada layanan *streaming*, khususnya pada Bluetooth yang hanya mencapai 732 Kbps.

Data video terbentuk dari tiga dimensi, yaitu dua dimensi spatial (horisontal dan vertikal), dan satu dimensi waktu. Proses kompresi data video dapat dilakukan terhadap aspek *redundancy spatial* (warna dalam *frame* atau *still image*), *redundancy temporal* (perubahan antar *frame*), serta audio yang dimilikinya. Kompresi terhadap *redundancy spatial* atau *intraframe* hampir serupa dengan teknik kompresi *lossy color reduction* pada *image*, dimana penghilangan beberapa warna dilakukan dengan memanfaatkan fakta bahwa mata manusia tidak terlalu sensitif terhadap perbedaan warna dibandingkan dengan *brightness* (intensitas cahaya). Sedangkan, kompresi *redundancy temporal* atau *interframe*, dilakukan dengan mengirimkan dan melakukan *encode* terhadap *frame* yang berubah saja, namun data yang sama masih disimpan. Perkembangan teknik

kompresi video yang cukup popular ditunjukkan pada gambar 2.1, beberapa diantaranya adalah H.261, H.263, MPEG-1, MPEG-2, dan MPEG-4.



Gambar 2.1 Perkembangan standar kompresi video

2.1.2.1 *H.263 dan H.261*

H.261 dan H.263 merupakan standar kompresi video yang dipublikasikan oleh *International Telecommunication Union* (ITU) yang umum digunakan dalam transmisi melalui saluran ber-*bandwidth* kecil. Standar kompresi video H.263 dipublikasikan oleh ITU-T pada tahun 1995/1996. Algoritma pengkodean standar ini didasarkan pada rekomendasi H.261, namun telah dilakukan beberapa perbaikan kesalahan dan peningkatan. Standar ini awalnya diperuntukkan bagi aplikasi konferensi video maupun *video telephony* pada saluran berkecepatan (*bitrate*) rendah, yakni mulai dari 20-30 Kbps ke atas. Pada kompensasi gerakan H.263 digunakan ketelitian setengah piksel, sedangkan pada H.261 digunakan ketelitian piksel penuh.

Untuk mengurangi *bandwidth* yang dibutuhkan dalam pengiriman data, *frame* video yang telah dikirimkan sebelumnya dikurangi dengan *frame* yang dikirimkan saat ini sehingga hanya perbedaan atau sisanya saja yang harus di-*encode* dan dikirimkan. Sedangkan, bagian *frame* video yang tidak berubah (misalnya : latar belakang) tidak di-*encode*. Disamping itu, upaya lainnya adalah estimasi dan kompensasi gerakan. Estimasi dilakukan pada daerah dari *frame* sebelumnya yang bergerak di *frame* selanjutnya. Modul estimasi gerakan akan membandingkan setiap 16x16 piksel blok (macroblock) di *frame* selanjutnya dengan daerah sekelilingnya di *frame* sebelumnya guna mencari kesesuaian.

Daerah yang sesuai akan dimasukkan ke posisi *macroblock* oleh modul kompensator gerakan. *Macroblock* gerakan terkompensasi akan dikurangkan dari *macroblock* dari *frame* selanjutnya. Jika estimasi gerakan dan proses kompensasi tersebut efisien, maka sisa *macroblock* yang ada akan berisi informasi yang sangat sedikit.

Beberapa penelitian mengajukan H.263 sebagai standar video untuk *streaming* melalui Bluetooth. H.263 memungkinkan penggunaan *bandwidth* mencapai *full-motion video* (30 fps) pada kecepatan serendah 128 Kbps, dimana mulanya dibangun untuk *stream* video berkualitas rendah pada *bandwidth* 20 hingga 62 Kbps. Namun, H.263 tidak mendukung beberapa fitur efisiensi kompresi dan kesalahan kanal yang *robust* dibandingkan MPEG-4.

2.1.2.2 *MPEG*

MPEG dikembangkan oleh *Motion Pictures Expert Group* untuk standar transmisi audio/video. MPEG-4 versi 1 dipublikasikan pada Oktober 1998, sedangkan versi 2 dipublikasikan pada Desember 1999. Standar ini sangat cocok untuk aplikasi dengan *bandwidth* dan *bit rate* rendah, yakni 4,8 hingga 64 Kb/detik dengan *bit rate* video : 5 Kb/s s/d 10 Mb/s dan *bit rate* audio : 2 Kb/s s/d 64 Kb/s. Sehingga, sesuai untuk *streaming* audio/video dalam jaringan. Banyak penelitian mengajukan MPEG-4 sebagai video *codec* untuk *streaming* pada Bluetooth yang sampai saat ini *bandwidth*-nya hanya mencapai 732 Kbps. MPEG-4 kemudian digabungkan ke dalam spesifikasi 3GPP (*3rd Generation Partnership Project*) untuk multimedia *mobile*.

MPEG-4 mempergunakan vektor *motion* diantara *frame* untuk meng-*encode redundancy temporal*, dan DCT (*Discrete Cosine Transform*) untuk meng-*encode redundancy spatial.* Sejumlah informasi yang dikodekan pada P-*frame* dan B-*frame* dikurangi karena terdapat diferensiasi pengkodean, dimana I-*frame* lebih utama. Berdasarkan pembahasan di atas, maka dapat komparasi dapat dinyatakan secara sederhana dalam tabel 2.1 berikut.

Tabel 2.1 Perbandingan standar format kompresi

Format kompresi	Waktu publikasi	*Bandwidth*	Resolusi	Aplikasi
H.261	1988-1990	384 k – 2 M	176 x 144	Konferensi video,

			352 x 288	ISDN, *delay* rendah
H.263	1995	28,8 k – 768 k	128 x 96 720 x 480	Konferensi video
MPEG-1	1993	400 k – 1,5 M	352 x 288	CD-ROM
MPEG-4	1998	28,8 k – 500k	176 x 144 352 x 288	*Fix web* dan *mobile web*

2.1.2.3 3GP

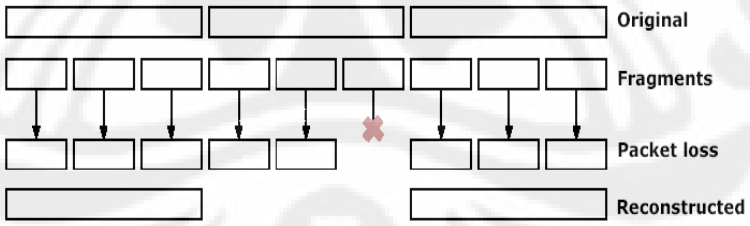
3GP merupakan versi sederhana dari format MPEG-4 *Part* 14 yang didefinisikan oleh 3GPP (*Thrid Generation Partnership Project*) untuk digunakan pada telepon *mobile* 3G. Standar ini digolongkan dua jenis, yaitu :

1. 3GPP untuk basis GSM, yang akan memiliki ekstesni file .3gp
2. 3GPP2 untuk basis CDMA, yang akan memiliki ekstensi file .3gp2

Keduanya didasari oleh video standar MPEG-4 dan H.263 serta standar audio AAC (*Advanced Audio Encoding*) atau AMR (*Adaptive Multi Rate*).

2.1.2.4 Fragmentasi

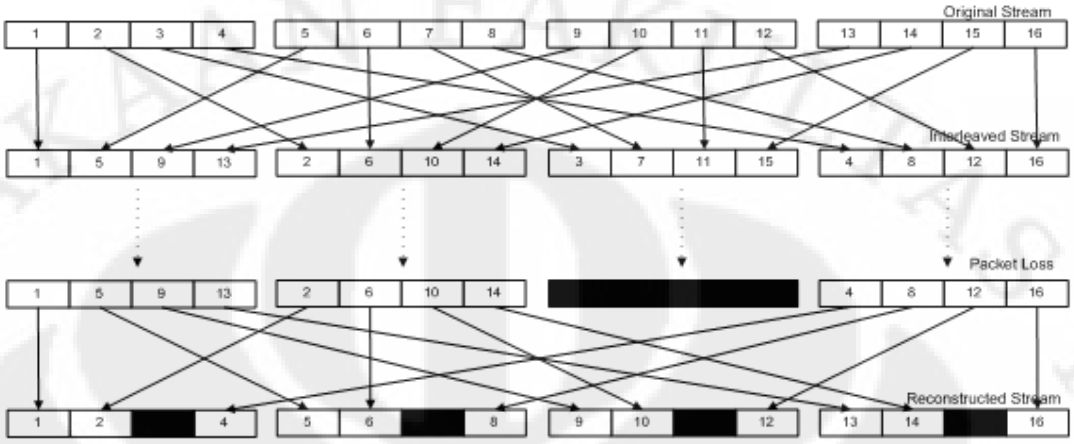
Untuk melakukan pengiriman secara *streaming*, data video terlebih dahulu dilakukan kompresi dan fragmentasi, yakni membagi paket-paket besar menjadi beberapa *slice* yang lebih kecil untuk kemudian dikirimkan melalui jaringan. Hal itu dimaksudkan agar paket dapat di*transmisi*kan secara efisien melalui jaringan yang memiliki keterbatasan *bandwidth*. Ilustrasi proses tampak pada gambar 2.2.



Gambar 2.2 Proses Fragmentasi

Pada proses fragmentasi, jika salah satu *slice* dalam satu paket mengalami kerusakan, maka paket tersebut akan dibuang sehingga akan dianggap sebagai *packet loss*. Untuk mengatasi masalah tersebut, maka dilakukan modifikasi fragmentasi guna memaksimalkan penanganan terhadap *packet loss. Original stream* dibagi menjadi beberapa *chunk* (potongan) yang lebih kecil, kemudian *chunk* tersebut di-*interleaved* (diselipkan) pada paket-paket yang terpisah,

sehingga jika terjadi *packet loss*, masih terdapat *chunk* lain yang dapat digunakan untuk rekonstruksi data. Hal tersebut diilustrasikan seperti gambar 2.3 berikut.



Gambar 2.3 Fragmentasi modifikasi

2.2 BLUETOOTH

2.2.1 Definisi

Bluetooth merupakan spesifikasi standar terbuka berbasis frekuensi radio untuk konektivitas data dan suara jarak pendek (*short-range*). Standar ini dirancang untuk menggantikan peranan koneksi kabel dan menjadi sistem jaringan nirkabel secara cepat, mudah, dan murah, serta mendukung kompabilitas semua kelas peralatan *portable*. Bluetooth juga didisain untuk beroperasi pada lingkungan dengan frekuensi radio yang ribut (*noisy radio frequency environment*). Teknologi ini beroperasi pada frekuensi bebas lisensi 2,4 GHz dalam ISM (*Industrial, Scientific, Medical*). serta mampu menyediakan koneksi layanan komunikasi data dan suara secara *real time* antara *host* yang dimilikinya dengan jangkauan layanan hingga 100 meter. Dibandingkan teknologi nirkabel lainnya, seperti 802.11 dan IrDA, teknologi ini menawarkan fitur *wireless mobile* yang lebih efisien, baik dari sisi biaya maupun konsumsi daya, serta telah mampu menyediakan layanan *video streaming* dan *ad hoc*.

2.2.2 Kelas Divais

Divais Bluetooth dapat dikategorikan menjadi tiga kelas seperti tercantum pada tabel 2.2 berikut :

Tabel 2.2 Kategori kelas divais Bluetooth

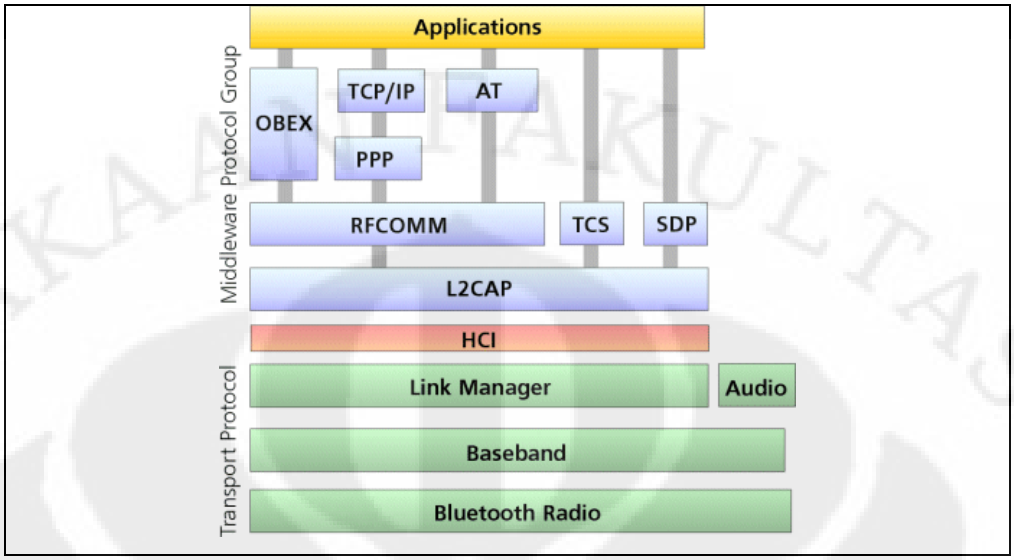
Kelas	Kekuatan Pancar		Jangkauan
	(Watt)	(dBm)	
Kelas 3	≤ 1 mW	≤ 0 dBm	10 m (ruang terbuka), 5 s.d. 7m (ruang tertutup)
Kelas 2 (*Output Control Optional*)	0,25 s.d. 2,5 mW	-6 s.d. 4 dBm	30 m
Kelas 1 (*Output Control* 2,5 mW s.d. 100 mW)	1 s.d. 100 mW	0 s.d. 20 dBm	100 m *Long-Range Bluetooth*

2.2.3 Protokol *Stack*

Protocol stack Bluetooth meliputi perangkat keras dan perangkat lunak yang merupakan bagian dari sistem suatu divais agar dapat mendukung implemetasi Bluetooth. Arsitektur utamanya terdiri dari *physical layer*, *link layer*, dan *application layer*.

Protokol *stack* Bluetooth secara garis besar dibagi menjadi dua bagian, yaitu Bluetooth *controller*, yang diimplementasikan pada perangkat keras, dan Bluetooth *host*, yang berhubungan dengan aplikasi dan layanan-layanan. Antara perangkat keras (*hardware*) dan perangkat lunak (*software*) dipisahkan oleh *layer* HCI. *Layer* tersebut juga mengimplementasikan *hardware* bagian bawah *layer* dan mengimplementasikan *software* pada bagian atasnya.

Protokol *physical-level* meliputi frekuensi radio (RF) dan *baseband* atau *Link Controller*. Protokol *link-level* terdiri dari *Link Manager Protokol* (LMP) dengan suatu lapisan adaptasi berupa *Logical Link Control and Adaptation Layer Protocol* (L2CAP) yang memungkinkan protokol *layer* atas untuk saling berhubungan dengan lapisan yang lebih rendah.



Gambar 2.4 *Protocol Stack* Bluetooth [6]

Komponen atau *layer* penyusun *protocol stack* Bluetooth, seperti yang telah diilustrasikan pada gambar 2.4, adalah :

1. *Applications*, profil yang mengendalikan pembangun dalam menentukan penggunaan protokol *stack* oleh sebuah aplikasi
2. *Telephony Control System* (TCS), *layer* yang menyediakan layanan *telephony*, serta kontrol panggilan suara dan data.
3. *Service Discovery Protocol* (SDP), berfungsi menemukan layanan dan karakteristiknya yang tersedia dari divais bluetooth *remote*
4. WAP dan OBEX, menyediakan *interface* pada bagian layer yang lebih tinggi pada protokol komunikasi yang lain. OBEX merupakan protokol independen, sehingga dapat digunakan API-nya secara terpisah dari API Bluetooth. Profil aplikasi yang dikembangkannya adalah sinkronisasi, transfer file, dan objek *push* untuk dukungan *business card.*
5. RFCOMM, *layer* yang menyediakan koneksi dan transfer melalui Bluetooth seperti pada emulasi standar serial (COM) *port*.
6. *Logical Link Control and Adaptation Layer Protocol* (L2CAP), *layer* ini menangani *segmentation and reassembly* (SAR) paket, multipleksi data dari *layer* yang lebih tinggi, dan merubah antara ukuran paket yang berbeda.
7. *Host Control Interface (HCI), layer* yang menangani komunikasi antara *host* (protokol *upper-layer*) dan modul Bluetooth (protokol *lower-layer*)

8. *Link Manager Protocol (LMP), layer* ini melakukan kontrol dan konfigurasi *link* kepada divais lain. Bersama dengan *baseband* menetapkan koneksi untuk jaringan itu.
9. *Baseband* dan *Link Controller, Layer* yang mengontrol akses dan pengiriman/penerimaan paket data yang melewati *bluetooth radio*.
10. Bluetooth Radio, merupakan *layer* fisik yang melakukan modulasi dan demodulasi data dari transmisi sinyal digital dari udara. *Layer* ini menggunakan FHSS

Layanan yang disediakan *Baseband* dan *Link Controller* adalah : prosedur *inquiry* dalam menemukan dan identifikasi divais, fungsi autentikasi (pengesahan) dan generasi kunci enkripsi, fungsi sinkronisasi *clock* dan pembangunan *link* divais pada frekuensi *hopping* yang tepat, fungsi koreksi kesalahan, *flow control*, dan pengaturan mode *power-saving*. *Baseband* memproses sinyal digital dari *layer* bawahnya. *Link Controller* mendukung *Link Manager.*

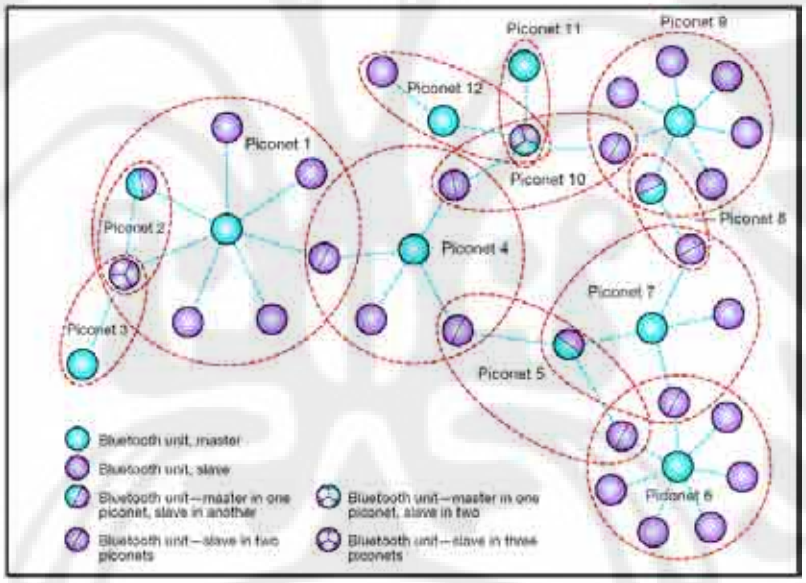
Link Manager Protokol (LMP) mengangkut paket melalui *baseband link* berdasarkan mekanisme *time-slot-oriented*, yaitu pembatasan ukuran paket LMP menurut slot waktu tunggal. Format PDU (*Protokol data Unit*) sederhana menggunakan dua bidang yang meliputi *Opcode*, yang mengidentifikasi urutan dan jenis dari paket, dan *Content Field* berisi informasi aplikasi spesifik. Urutan protokol LMP serupa dengan pola pertukaran informasi *request-response* pada arsitektur *client-server*. Secara umum, respon tunggal PDU dikirim setelah diterimanya *request* yang asli. Satu set pesan *request* dapat disiarkan kepada semua peserta suatu jaringan, yang berdampak satu permintaan dapat menimbulkan beberapa tanggapan.

Logical Link Control and Adaptation Layer Protocol (L2CAP) memungkinkan transmisi data antar *upper-layer* dan *lower-layer*, serta mendukung pihak ketiga protokol *upper-layer* seperti TCP/IP. L2CAP menyediakan manajemen grup dengan memetakan protokol *upper-layer* ke dalam jaringan.

2.2.4 Mekanisme kerja

Bluetooth menyediakan tiga jenis koneksi jaringan, yakni: *point-to-point*, *piconet* atau *point-to-multipoint*, dan *scatternet*. Unit kecil kumpulan beberapa divais Bluetooth disebut *piconet*. *Piconet* terdiri dari sebuah divais *master*, dan *slave* aktif dengan jumlah maksimum tujuh buah. Unit master adalah unit yang memulai transmisi, sedangkan unit *slave* adalah unit yang menjawabnya.

Kumpulan beberapa *piconet* yang saling terhubung atau berpotongan dan memungkinkan untuk saling berkomunikasi disebut *scatternet*. Unit manapun di dalam *piconet* dapat berperan sebagai *master/slave* bagi suatu *piconet* berdasar pembagian waktu *multiplexing*, dengan batasan hanya bertindak selaku master hanya untuk satu *piconet* secara bersamaan. Gambar 2.5 menunjukkan sebuah master *piconet* yang menjadi *slave* untuk *piconet* yang lain dalam *scatternet*.



Gambar 2.5 *Piconet* dan *Scatternet* Bluetooth [5]

Divais Bluetooth mempunyai empat *basic states*, yaitu: *master* (pengontrol *piconet*), *active slave*, *parked slave*, dan *standby*. *Parked slave* secara logika masih bagian dari *piconet* tetapi *low power*. Pada kondisi *standby*, konsumsi daya divais dapat berkurang sekitar 98%.

Inquiry adalah proses mempelajari divais Bluetooth lainnya yang berada dalam jangkauan. Selama proses tersebut, master melakukan *broadcast page command* terus-menerus dengan menggunakan *inquiry* ID kepada setiap divais *standby* dalam jangkauannya. Dengan melakukan persetujuan, divais akan

merespon dengan sebuah standar paket FHSS yang menyediakan BT ID yang unik. Setelah, proses *inquiry*, dilanjutkan dengan proses *paging*, yakni pembangunan hubungan antar divais. *Master* akan mengirim sebuah paket FHS sebagai respon dari balasan *slave* sebelumnya dan menetapkan sebagai *Active Member Address* pada *piconet*.

Bluetooth menggunakan metode *fast acknowledgement* dan mekanisme *Frequency Hopping Spread Spectrum* (FHSS) untuk membuat *link* menjadi kuat. Mekanisme FHSS memperkecil terjadinya interferensi dari divais lainnya dengan cara *hopping* ke frekuensi baru setelah mengirim atau menerima paket sebanyak 1600 kali per detik. Bluetooth membagi frekuensi band 2.4 GHz menjadi 79 kanal frekuensi pada 1 MHz (dari 2.402 sampai 2.480 GHz). Jumlah perangkat yang terhubung maskimal sejumlah 256 unit, dimana jumlah maksimal perangkat yang aktif secara bersamaan dalam sebuah *piconet* adalah tujuh buah.

2.2.5 Bluetooth Link

Bluetooth mengkategorikan dua tipe *link* yang merupakan bagian dari spesifikasi *baseband*, yaitu :

2.2.5.1 *Synchronous Connection Oriented* (SCO) *link*

Synchronous Connection Oriented (SCO) merupakan koneksi *point-to-point circuit switched* simetris dengan *bandwidth* hingga 64Kbps. *Link* SCO diperuntukkan bagi transmisi audio dengan penyediaan jaminan QoS *streaming* bidireksional. Untuk QOS, dibutuhkan alokasi waktu cadangan untuk *transmisi* data saat mengkonfigurasi *link*. Hal itu dikarenakan bahwa kerusakan dan kesalahan paket tidak dikirim kembali. Semua link SCO beroperasi pada 64 kbps. Sebuah *master device* dapat memiliki hingga tiga *link* SCO pada waktu yang bersamaan, semua pada *slave* yang sama atau *slave* yang berbeda.

2.2.5.2 *Asynchronous ConnectionLess* (ACL) *link*

Asynchronous ConnectionLess (ACL) *link* digunakan untuk transmisi paket data. ACL mendukung komunikasi *point-to-multipoint* hingga 732 Kbps *downlink* dan 128Kbps *uplink*, serta koneksi *broadcast* dan media *streaming*. *Link* ini menyediakan pengiriman *error-free* pada data yang berarti kerusakan atau kesalahan paket akan dikirim kembali. Sebuah divais master dapat memiliki

sebuah nomor *link* ACL ke nomor divais yang berbeda, tetapi hanya satu *link* ACL yang dapat eksis diantara dua divais. Transfer data biasanya dipercayakan pada *layer Logical Link Control and Adaptation Protocol* (L2CAP) dan RFCOMM pada Bluetooth *stack*. Bluetooth dapat menggunakan link yang manapun dan dapat merubah link selama transmisi, walaupun suatu link ACL harus dibentuk sebelum suatu link SCO dapat digunakan.

2.2.6 Keunggulan

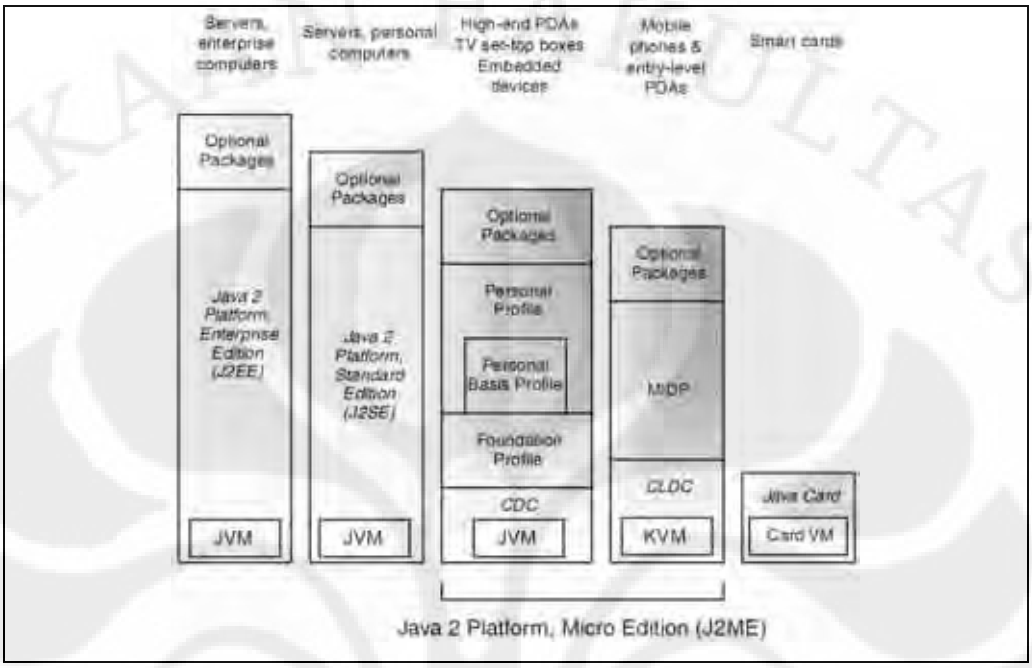
Bluetooth mampu menyelenggarakan jaringan *ad-hoc* dengan mengembangkan koneksi *peer-to-peer* nirkabel tanpa memerlukan *base station* atau terminal. Sehingga, memungkinkan semua alat untuk berkomunikasi secara spontan.

Bluetooth mempunyai konsumsi daya dan biaya yang efisien. Divais dapat berpindah ke mode *lower-power* saat tidak aktif dalam jaringan (*standby*). Bluetooth radio mengkonsumsi kurang dari 3 persen energi dari yang biasa dikonsumsi telepon *mobile*.

Bluetooth menyediakan dua buah mekanisme koreksi kesalahan yang meliputi *Forward Error Correction* (FEC) dan *Automatic Repeat reQuest* (ARQ). Mekanisme FEC diperuntukkan pada lalu lintas audio yang memprioritaskan ketepatan waktu daripada ketelitian, yakni tidak dapat ditolerirnya faktor keterlambatan transfer audio. Sedangkan, mekanisme ARQ digunakan pada transfer dan aplikasi data. Sedangkan, untuk keamanan, Bluetooth memiliki teknik autentikasi yang berfungsi untuk memverifikasi identitas alat yang mengirimkan informasi, dan enkripsi yang digunakan untuk memastikan integritas dari data

2.3 J2ME

2.3.1 Arsitektur J2ME



Gambar 2.6 Perbandingan Arsitektur J2ME [10]

Arsitektur J2ME berbeda dengan J2SE dan J2EE. Perbandingannya turut ditentukan oleh jenis perangkat yang digunakan, seperti yang telah diilustrasikan pada gambar 2.6 di atas. Desain yang minim dan ringan dilakukan pada J2ME dengan menghilangkan beberapa komponen, seperti Java.awt yang dibutuhkan untuk pengembangan GUI (*Graphical Interface Unit*), java.sql yang berfungsi untuk kebutuhan koneksi dengan *database* melalui *driver* JDBC (*Java Database Conectivity*), dan *Included Java library*, sehingga dapat dimanfaatkan oleh divais berukuran mini, dantidak semua *library* J2SE dapat dipakai. Namun, J2ME tetap mempunyai semua karakterisitik Java dan memiliki *library* khusus yang tidak dimiliki J2SE.

Perbedaan spesifikasi divais atau perangkat turut menentukan arsitektur J2ME ke dalam dua kategori dengan arsitektur yang sedikit berbeda.Hal ini tampak pada gambar 2.6. KVM (*Kauai Virtual Machine* atau sering disebut sebagai *Kilo Virtual Machine*) merupakan paket JVM yang dirancang untuk perangkat yang kecil, namun tetap mendukung sebagian fitur JVM. Beberapa perbedaan KVM pada CLDC dengan CDC adalah tidak didukungnya operasi

floating-point dan finalisasi objek, terdapat pembatasan *error handling*, serta tidak adanya *Java Native Interface* (JNI) dan refleksi.

2.3.1.1 *Konfigurasi J2ME*

Konfigurasi merupakan *library* Java minimum yang mengimplementasikan fitur atau kapabilitas standar berbagai perangkat. J2ME memiliki dua jenis konfigurasi, yaitu :

1. CDC (*Connected Device Configuration*), konfigurasi yang digunakan pada perangkat dengan ukuran memori dan *bandwidth* jaringan yang cukup besar, dan dapat mengimplementasikan seluruh fitur J2SE. Contohnya : TV *mobile*, Internet TV, *communicator*, PDA *high*-end, dll.
2. CLDC (*Connected Limited Device Configuration*), konfigurasi yang digunakan pada perangkat yang lebih kecil, dengan memori, koneksi, prosesor, dan *interface* yang berukuran kecil. Contohnya : telepon selular, *pager* dua arah, PDA, dll.

2.3.1.2 *Profil J2ME*

Profil J2ME merupakan *layer* tambahan atau perluasan di atas konfigurasi J2ME, dan menyediakan Java API untuk kelas spesifik bagi divais [10]. Beberapa tipe profil yang dapat dipergunakan adalah :

1) *Foundation Profile*

   Profil pertama dari ketiga profil CDC ini digunakan untuk konfigurasi CDC, sebagai pondasi untuk membentuk profil lainnya, serta menyediakan implementasi kemampuan jaringan tanpa *user interface*. Disamping itu, profil ini dapat dikombinasikan dengan *Personal Profile* dan *Personal Basis Profile* apabila dibutuhkan *user interface*.

2) *Personal Profile*

   Profil ini merupakan hasil perluasan dari *Foundation Profile* yang mendefinisikan ulang *Personal Java* sebagai profil yang dapat digunakan dalam J2ME. Profil ini ditujukan bagi divais yang membutuhkan GUI secara maksimal dan dukungan untuk Internet, dengan kemampuannya menjalankan

applet berbasiskan *web* untuk *desktop environment. Library* untuk *Abstract Window Toolkit* (AWT) dan ketelitian *web* turut disediakan olehnya.

3) *Personal Basis Profile*

Profil bagian dari *Personal Profile* ini menawarkan *environment* berbasis jaringan bagi alat-alat yang terkoneksi ke jaringan yang mendukung GUI yang terbatas atau membutuhkan *Graphical interface* spesial.

4) *Information Module Profile (IMP)*

Profil IMP dibangun berdasarkan *profile* MIDP 1.0. Kombinasi IMP dan CLDC menyediakan Java *application environment* untuk alat dengan res*ource* yang terbatas dan divais jaringan *embedded*, sehingga memudahkan pengembang menggunakan IMP meski divais tidak kaya akan *Graphical user interface.*

5) *Mobile Information Device Profile (MIDP)*

Profile ini menyediakan fungsi-fungsi inti untuk *mobile application*, seperti *user interface*, *network* connectivity, *local data storage* dan *application lifecycle management.* Selain sebagai referensi implementasi pada *mobile phone* dan *pager*, terdapat juga implementasi untuk Palm OS, yang dikenal dengan nama MIDP for Palm OS.

Perkembangan MIDP saat ini menghasilkan dua versi MIDP yaitu MIDP 1.0 dan MIDP 2.0 dengan perbandingan spesifikasi seperti tercantum pada tabel 2.3.

Tabel 2.3 Perbandingan spseifikasi MIDP 1.0 dan MIDP 2.0 [9]

Spesifikasi	MIDP 1.0	MIDP 2.0
Display	96 x 54	96 x 54
Kedalaman *display*	1 bit	1 bit
Bentuk pixel	Mendekati 1 : 1	Mendekati 1 : 1
Input	*Keyboard* dan *touch screen*	*Keyboard* dan *touch screen*
Memori	**1.** 128 Kb *non-volatile* memori **2.** 8 Kb *non-volatile* memori data *persistence* yang dibuat oleh aplikasi **3.** 32 Kb memori *volatile* untuk JRE	**1.** 256 Kb *non-volatile* memori **2.** 8 Kb *non-volatile* memori data *persistence* yang dibuat oleh aplikasi **3.** 128 Kb memori *volatile* untuk JRE
Jaringan	Dua arah, nirkabel	Dua arah, nirkabel
Multimedia	Tidak memiliki kemampuan	Memiliki kemampuan

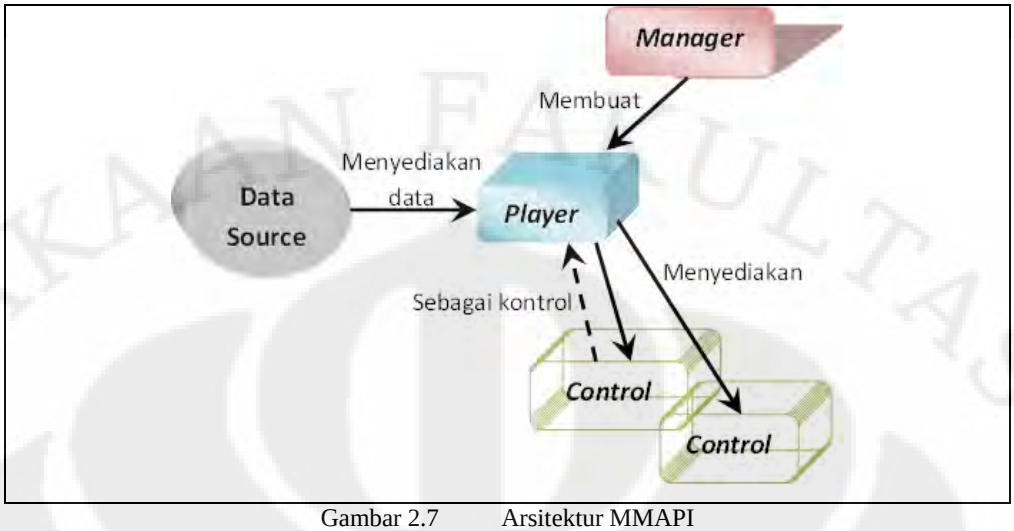
	multimedia (suara dan video)	multimedia (suara dan video)
Library	Tidak sebanyak MIDP 2.0	Banyak

2.3.1.3 *Mobile Media API (MMAPI)*

Mobile Media - Application Programming Interface (MMAPI) merupakan *package* opsional MIDP J2ME yang dirancang untuk memfasilitasi dan mendukung aplikasi multimedia pada perangkat *mobile*. MMAPI didefinisikan oleh *Java Community Process* (JCP) melalui standar spesifikasi JSR-135.

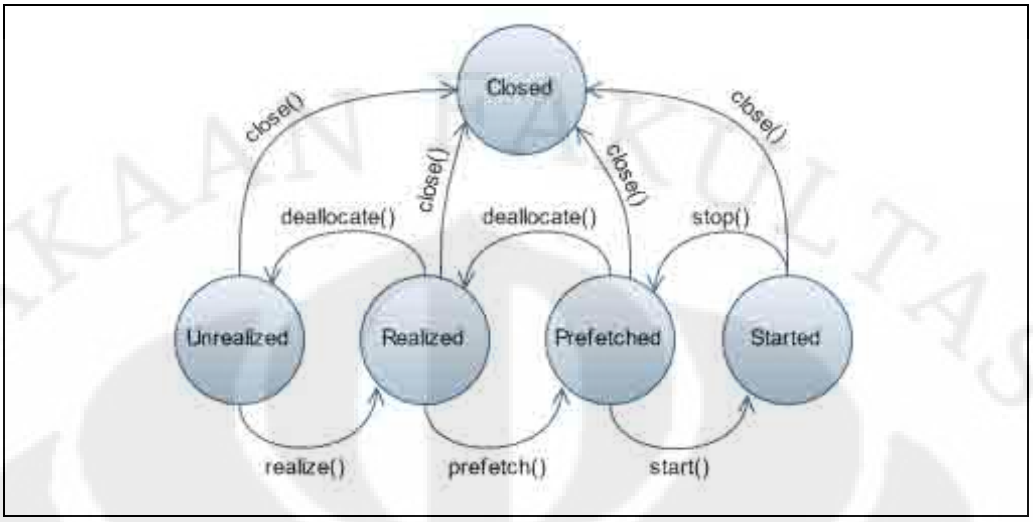
MMAPI menyediakan kemampuan multimedia yang fleksibel dan memiliki *interface* yang sederhana. Standar ini mendukung fitur untuk memainkan audio dan video, merekam sinyal gambar, audio dan video ke dalam file, akses ke vibrator telepon selular, dan fitur-fitur lainnya. Namun, meski suatu peralatan mempunyai dukungan MMAPI tidak berarti telah mendukung semua fitur yang tercakup melainkan tergantung pada spesifikasi yang dimiliki, misalnya divais yang mempunyai kemampuan perekaman audio belum tentu memiliki kemampuan perekaman video.

Dukungan aplikasi multimedia ditangani oleh empat buah komponen utama, yaitu: *Data source*, *Player*, *Control*, dan *Manager*. *Data source* berfungsi mendapatkan data dari sumber yang tersedia, yakni dapat berasal dari jaringan (HTTP, URL), perekaman RMS, koneksi *streaming*, dll. *Manager* merupakan kelas yang menangani metode dan detail penggenerasian (pembuatan) *Player* dan akses.aturan pendukungnya. Sedangkan, *Player* merupakan sebuah *interface*. Aplikasi dapat menghendaki mengenkapsulasi perilaku untuk mengendalikan *playback*, *capture*, dan yang lainnya melalui *interface* objek *Control* setelah *player* terbentuk. *Player* tidak terlalu mempermasalahkan asal dan bagaimana cara mendapatkan data, melainkan hanya perlu membaca, memproses, menampilkan, dan memainkan *playback* media pada divais keluaran. Sebuah *Player* dapat memiliki lebih dari satu buah *Control*. Hubungan media *framework* tersebut seperti terlihat pada arsitektur MMAPI pada gambar 2.7.



Gambar 2.7 Arsitektur MMAPI

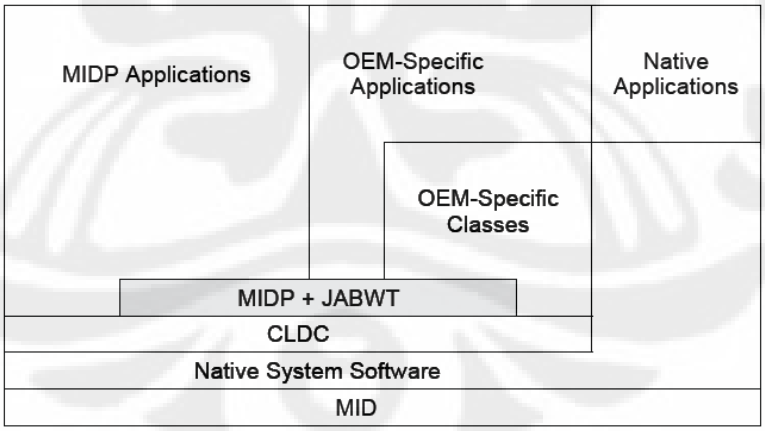
Objek *Player* dapat berada diantara lima status (*state*) yang meliputi : *UNREALIZED, REALIZED, PREFETCHED, STARTED,* dan *CLOSED*. *Player* pertama kali terbentuk berada pada status *unrealized,* yakni keadaan saat *Player* telah diinstansiasi pada *heap memory* namun *Player* belum mengalokasikan sumber datanya (*resource).* Ketika berubah dari *unrealized* menjadi *realized*, *player* melakukan komunikasi untuk melokasikan dan mendapatkan sumber data yang dikehendaki. Pada umumnya, *Player* bergerak dari status *unrealized* ke *realized*, kemudian *prefetched* lalu *started*. Status *prefetched* terjadi setelah *Player* menjalankan proses *start-up* dan persiapan kontrol. Metode Player.start() akan memulai dijalankannya *Player* sehingga status berubah menjadi *started,* dan akan berhenti ketika mencapai *end of media* atau ketika metode *stop* dipanggil. Status *closed* menunjukkan tertutupnya *player* setelah melepas semua sumber yang terikat. Hubungan antara kelima status tersebut diilustrasikan pada gambar 2.8.



Gambar 2.8 *Media Player State Diagram* MMAPI

2.3.1.4 *Java APIs For Bluetooth Wireless Technology (JABWT)*

Java APIs for Bluetooth Wireless Technology (JABWT) merupakan *package* opsional J2ME yang didefinisikan pada standar JSR-82 oleh *Java Community Process* (JSR-82) untuk mendukung pengembangan Bluetooth pada divais *mobile*. *Package* ini beroperasi di atas CLDC, dan dimaksudkan untuk memberi kemampuan pada profil, seperti MIDP. Gambar 2.9 berikut merupakan ilustrasi keberadaan JABWT pada platform J2ME.

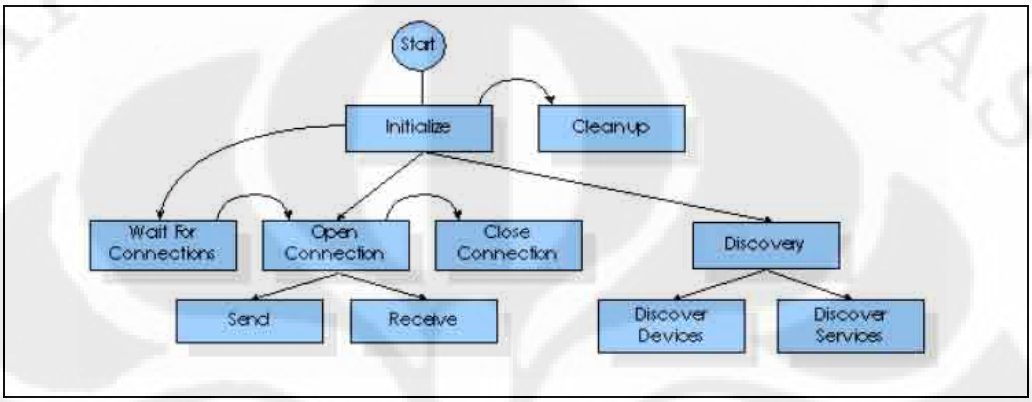


Gambar 2.9 JABWT pada J2ME

JABWT mendefinisikan dua *package* yaitu :

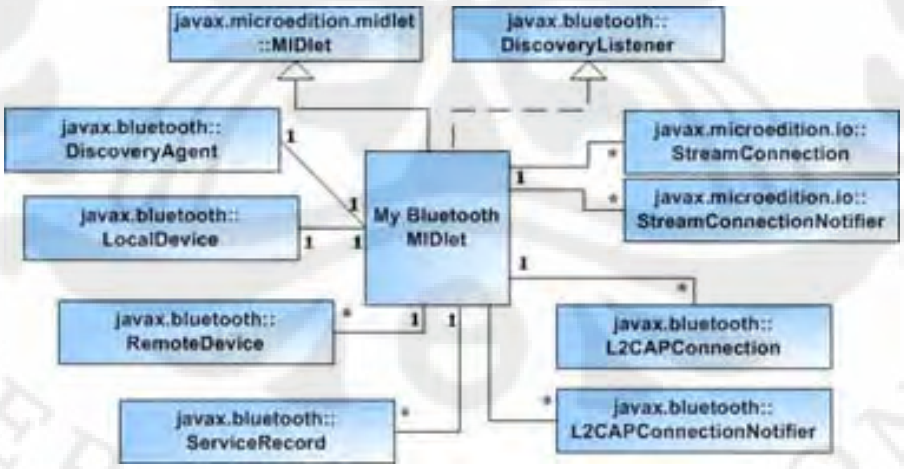
1. javax.bluetooth, berfungsi menyediakan *core bluetooth* API
2. javax.obex, berfungsi menyediakan *Object Exchange* (OBEX) API pada tabel 2.4

Penggunaan JABWT meliputi tahapan operasi inisialisasi *stack* Bluetooth, pencarian divais atau layanan di sekitarnya, menunggu koneksi, membuka dan menutup koneksi, dan perform I/O. Berikut merupakan gambar ilustrasi perbedaan operasi bluetooth yang dapat digunakan. Gambar 2.10 menunjukkan hubungan yang terjalin pada penggunaan JABWT.



Gambar 2.10 JABWT pada J2ME

Protokol OBEX umumnya digunakan oleh teknologi inframerah untuk melakukan transmisi objek, jauh sebelun teknologi Bluetooth dikenalkan. Para desainer dari Java Bluetooth memutuskan untuk tidak memasukannya ke dalam Bluetooth. JSR-82 mengandung dua paket independen `javax.bluetooth` (13 kelas dan interface untuk komunikasi dengan protocol Bluetooth), dan `javax.obex` (8 kelas untuk mengirimkan objek antar piranti.



Gambar 2.11 Kelas dan Interface dalam JABWT

Tabel 2.4 Kelas dalam javax.obex

Kelas	Deskripsi
Authenticator	Interface yang menyediakan cara respon

	authentifikasi
ClientSession	Interface ini menyediakan metode *request* OBEX
HeaderSet	Interface yang menjelaskan metode mendapatkan dan memberi nilai OBEX *header*.
Operation	*Interface* yang menyediakan cara memanipulasi operasi OBEX GET dan PUT
SessionNotifier	*Interface* yang menjelaskan pemberitahuan mengenai koneksi pada sisi server dalam koneksi OBEX.
PasswordAuthentication	Kelas yang bertanggung jawab untuk kombinasi nama pengguna dan sandi.
ReponseCodes	Kelas ini mengandung beberapa kode respon yang benar, yang dikirmkan *server* ke *client*.
ServerRequestHandler	Kelas yang menjelaskan *event listener* yang memberikan respon pada OBEX *request* untuk *server*

Tabel 2.5 Kelas dalam javax.bluetooth

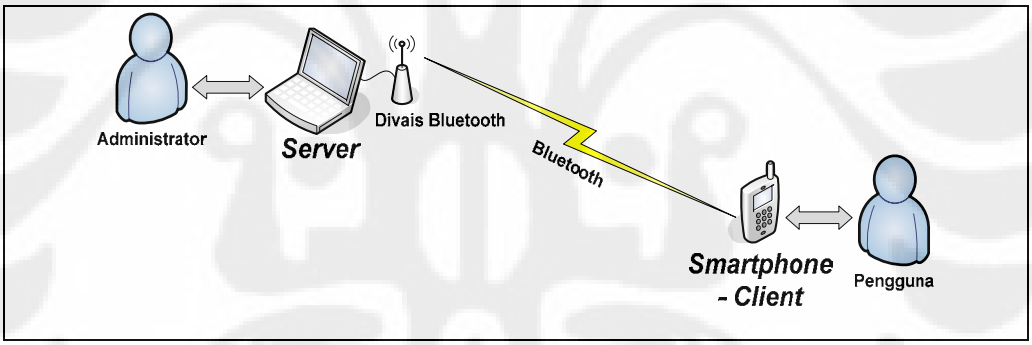
Kelas	**Deskripsi**
DiscoveryListener	*Interface* yang menyediakan sebuah aplikasi agar dapat menerima *device discovery* dan *service discovery events*
L2CAPConnection	*Interface* yang mewakili *connection oriented* melalui L2CAP
L2CAPConnectionNotifier	*Interface* yang menyediakan pemberitahuan untuk koneksi L2CAP
ServiceRecord	*Interface* yang menjelaskan karakteristik servis Bluetooth
DataElement	Kelas yang mendefinisikan berbagai macam tipe atribut dari servis Bluetooth
DeviceClass	Kelas yang mewakili perekaman *Class of Device* (CoD) yang didefinisikan di bluetooth
DiscoveryAgent	Kelas yang menyediakan metode untuk *DeviceDiscovery* dan *ServiceDiscovery*.
LocalDevice	Kelas yang mereprenstasikan piranti lokal Bluetooth.
RemoteDevice	Kelas yang mereprensentasikan piranti *remote* dari Bluetooth.
UUID	Kelas yang mendefiniskan bilangan universal yang unik.
BluetoothConnectionException	*Exception* yang di lempar ketika terjadi koneksi Bluetooth (L2CAP,RFCOMM,OBEX) ketika bermasalah
BluetoothStateException	*Exception* yang dilempar ketika permintaan yang dibuat kepada sistem Bluetooth tidak didukung.
ServiceRegistrationException	*Exception* yang dilempar saat gagal menambahkan *service record* pada SDDB (*Service Discovery Database*) lokal

BAB III
PERANCANGAN DAN ANALISA SISTEM KOMUNIKASI APLIKASI PADA SISI *CLIENT*

3.1 ARSITEKTUR SISTEM

Layanan hanya ditujukan kepada pengguna yang berada pada lingkungan sekitar dengan radius jangkauan hingga 50 meter. Hal ini disebabkan oleh keterbatasan daya pancar yang dimiliki oleh koneksi Bluetooth.

Arsitektur sistem *video streaming over Bluetooth* terdiri dari dua komponen utama, yaitu *smartphone* dan komputer *server*. Komunikasi antara keduanya diselenggarakan menggunakan koneksi serial Bluetooth. Gambar 3.1 menampilkan skema arsitektur sistem yang dimaksud.



Gambar 3.1 Arsitektur *video streaming* berkoneksi *Bluetooth*

Dalam arsitektur sistem ini, *smartphone* berfungsi sebagai *client* dengan *user interface* yang menghubungkan interaksi pengguna dengan sistem. Melalui komponen ini, seorang pengguna dapat melakukan identifikasi dan koneksi divais, pencarian layanan, pembacaan daftar video yang ditawarkan, pemilihan permintaan video yang dikehendaki kepada server, dan pemutaraan video secara *streaming*. Sedangakn, s*erver* berupa komputer berfungsi sebagai tempat penyimpanan data video, inisialisasi dan koneksi divais, penyiaran daftar video yang ditawarkan, dan proses *streaming* video ke *smartphone* (*client*) milik pengguna.

3.2 SKENARIO UMUM

3.2.1 Aplikasi Interaktif

Pada sebuah sistem, calon pengguna yang hendak menikmati layanan akses video diharuskan mengaktifkan *user interface* aplikasi pada *smartphone* yang dimilikinya dan koneksi Bluetooth *smartphone* terlebih dahulu. Setelah melakukan inisialisasi, identifikasi, dan membangun koneksi dengan *server* yang dituju, pengguna akan menunggu daftar video yang tersedia oleh pihak *server.*

Layanan disediakan berupa akses terhadap video yang telah tersimpan di *server*. Administrator terlebih dahulu menentukan beberapa judul video yang hendak ditawarkan kepada pengguna layanan. Daftar judul tersebut kemudian dipublikasikan kepada pengguna sebagai *client* yang berada di sekitar jangkauannya melalui koneksi Bluetooth. Administrator mempunyai wewenang dalam menentukan file video apa saja yang akan ditawarkan untuk dijalankan.

Daftar yang telah diperoleh dari *server* dapat dipilih melalui *user interface* yang ada pada *smartphone*. Permintaan akan judul terpilih akan diinformasikan kepada *server* melalui koneksi Bluetooth. Hal itu akan direspon *server* sehingga permintaan video dapat dinikmati pada *smartphone* pengguna setelah diproses dan diakses secara *streaming* dari *server* ke *client.* Selama proses *streaming*, *client* dapat melakukan *pause* (berhenti sejenak) yang kemudian dapat di-*play* kembali, atau *stop.*

3.2.2 Aplikasi Non Interaktif

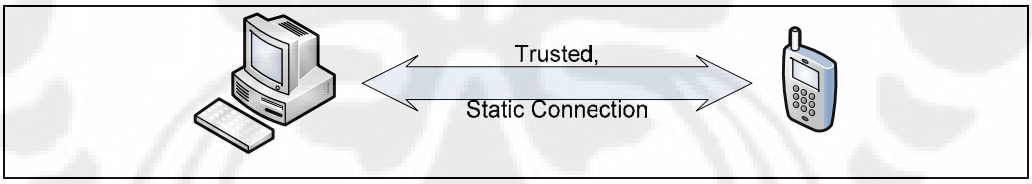
Pengguna sebagai *client* menunggu ketersediaan akses dari *server* Pengiriman video ditentukan hanya oleh administrator pada sisi *server*, sedangkan sisi *client* bersifat pasif. Setelah video diterima, *interface* pada *smartphone* dapat memainkannya seperti pada tipe interaktif di atas.

3.3 ANALISA SISTEM

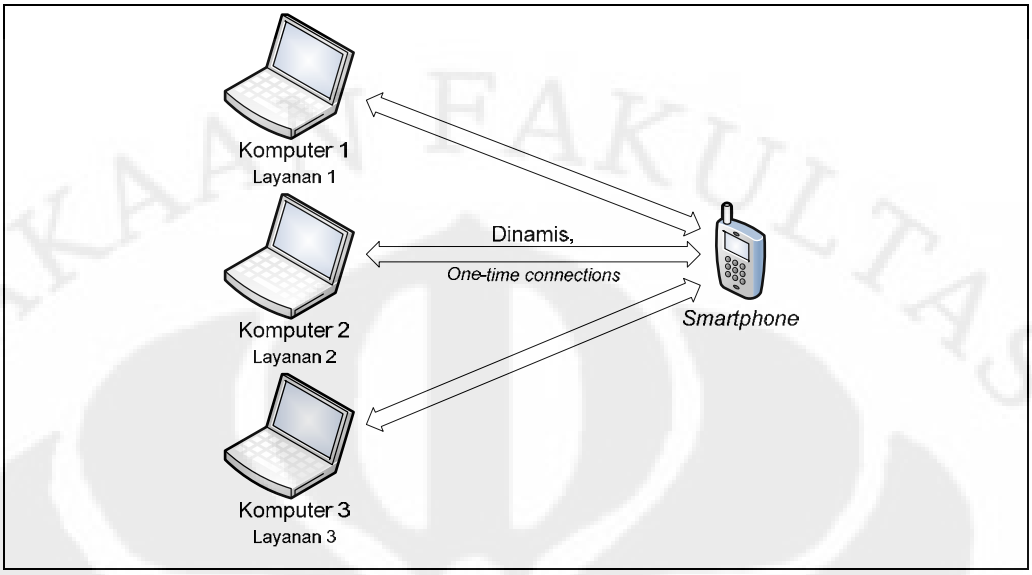
3.3.1 Analisa Koneksi

Koneksi Bluetooth pada suatu sistem dapat digolongkan menjadi dua tipe, yaitu koneksi dinamis dan statik. Pemanfaatan salah satu dari tipe tersebut dilakukan berdasarkan pertimbangan terhadap karakteristik yang dimiliki setiap

tipenya. Pertama, koneksi dinamis melakukan pencarian divais berdasarkan layanan yang disediakan yakni ketepatan layanan lebih diutamakan daripada divais mana yang terkoneksi, dimana tahapan yang dilakukan adalah mekanisme saling menemukan koneksi, *pairing,* kemudian proses transfer, dan setelah itu koneksi pun selesai. Koneksi ini diilustrasikan pada gambar 3.3. Sedangkan, koneksi statis, seperti yang ditunjukkan oleh gambar 3.2, dipergunakan untuk melakukan jenis komunikasi yang intensif atau frekuensinya berulang diantara dua divais yang sudah saling kenal. Tipe koneksi statis dibutuhkan oleh pengguna ketika ingin melakukan sinkronisasi data, kalender, pengiriman file, dan instalasi aplikasi. Kedua, sinkronisasi dan pencarian koneksi tipe dinamis berjalan secara otomatis dimana penggunanya hanya mendapat info bahwa datanya telah sinkron, serta autentifikasi tidak perlu dilakukan oleh kedua pihak. Sedangkan, tipe statis umumnya hanya akan melakukan *pairing* satu kali, dalam hal ini antara *smartphone* dan komputer *server*, dan perlu mengetikan *passkey* yang sama untuk kedua divais. Namun, otorisasi dapat dihilangkan pada pembuatan divais berstatus *trusted* sehingga koneksi berikutnya akan langsung diterima secara otomatis tanpa perlu autentifikasi. Proses autentifikasi dan autorisasi hanya dapat di set aktif ataupun tidak pada MIDlet *client* atau *smartphone*.



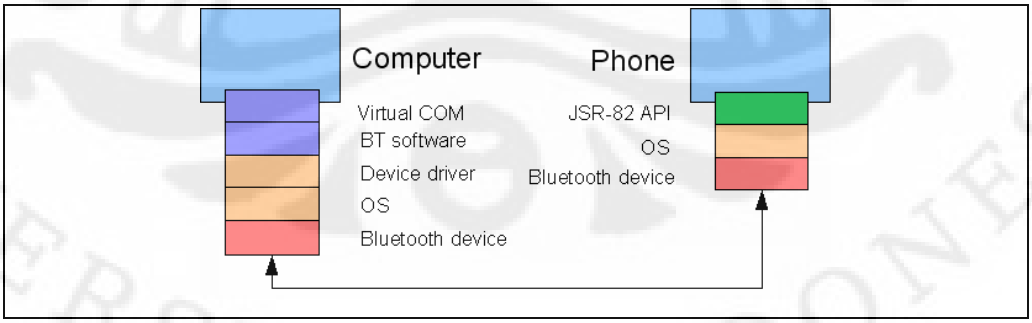
Gambar 3.2 Koneksi Bluetooth statik



Gambar 3.3 Koneksi Bluetooth dinamis

Berdasarkan analisa pertimbangan tersebut di atas, maka koneksi yang akan dipergunakan pada penyusunan skripsi ini koneksi statis. Hal ini dikarenakan kriteria yang dibutuhkan adalah komunikasi dilakukan secara intensif hingga proses aplikasi dan transmisi file video selesai diselenggarakan (tidak segera putus setelah transfer data dilakukan), pencarian *server* yang dituju lebih diutamakan daripada bentuk layanan, komunikasi pada penulisan ini hanya berupa *point-to-point* berupa satu *server* dan satu *client,* pengguna harus dapat mengendalikan tahapan proses yang dilakukan, dan kedua pihak (*server* dan *client)* dapat ditetapkan dalam status *trusted.*

3.3.2 Analisa Kebutuhan Sistem



Gambar 3.4 Spesifikasi kebutuhan minimal sistem

3.3.2.1 *Perangkat Lunak*

Perangkat lunak yang dibutuhkan pada sisi *client* yang berupa *smartphone* adalah :

1. Sistem Operasi : dibutuhkan untuk mendukung operasioanl *Bluetooth stack* dan memungkinkan dilakukan instalansi perangkat lunak *mobile.* Dalam skripsi ini dipergunakan sistem operasi Symbian yang dapat lebih mudah ditemui pada *smartphone* dibandingkan sistem operasi lainnya.
2. *Driver* divais Bluetooth **:** diperlukan untuk melakukan komunikasi menggunakan Bluetooth versi 2.0. yang telah didapatkan secara terintegrasi dalam *smartphone* yang mendukung.
3. JSR-82 API : telah terkandung pada *smartphone* yang mendukung untuk melangsungkan koneksi Bluetooth antara J2SE dan J2ME pada *server* dan *client.*
4. J2ME (*Java2 Micro Edition)* : Java Profil MIDP versi 2.0 dan MMAPI dibutuhkan untuk menjalankan aplikasi multimedia, serta CLDC versi 1.0 sebagai standar konfigurasi Bluetooth *stack.*
5. IDE (*Integrated Development Environment*) : Netbeans 5.5.1 dan Netbeans_mobility 5.5.1. berfungsi sebagai perangkat untuk membuat sistem *client* dengan menggunakan bahasa pemrograman Java
6. *Java Development Kit* (JDK) dan *Java Runtime Environment* (JRE) : WTK (*Wireless Tool Kit*) J2ME (*Java2 Micro Edition)* versi 2.2 sebagai perangkat yang harus terintegrasi dengan IDE untuk membuat dan menjalankan sistem *client* dengan menggunakan bahasa pemrograman Java pada perangkat *mobile.*

Perangkat lunak yang dibutuhkan pada sisi *server* adalah sebagai berikut :

1. Sistem Operasi : pada skripsi ini digunakan Windows XP *Service Pack* 2
2. *Driver* divais Bluetooth **:** diperlukan untuk melakukan komunikasi dengan divais Bluetooth pada komponen perangkat lunak level rendah yang umumnya diimplementasikan pada Bluetooth *stack. Driver* diperoleh bersamaan dengan divais Bluetoothnya, disamping itu beberapa divais juga telah mendapat dukungan *driver* secara langsung yang telah disediakan oleh Windows XP. Bluetooth *stack* yang ada yaitu BlueSoleil, Widcomm (Broadcom), Winsock (Microsoft), dan Mac OS X.
3. IDE (*Integrated Development Environment*) : Netbeans 5.5.1.
4. JDK dan JRE : JDK 1.5 yang harus terintegarsi dengan IDE.

5. JSR-82 API : dibutuhkan BlueCove.jar sebagai *library* API (*Application Peripheral Interface*). BlueCove.jar bekerja pada JVM Bluetooth yang menghubungkan dengan *layer* pada Bluetooth Stack.

3.3.2.2 *Perangkat Keras*

Perangkat keras yang dibutuhkan adalah sebagai berikut :

1. *Server* : komputer atau *Personal Computer* sebagai media penyimpanan file video dan pusat penyedia layanan.
2. *Client* : *smartphone* yang telah memenuhi spesfikasi yang dibutuhkan, yakni memiliki fitur Bluetooth dengan versi 2.0, sistem operasi Symbian, dukungan minimal profil Java MIDP versi 2.0, CLDC 1.0, dan MMAPI, serta mendukung untuk menjalankan format video. Untuk perancangan dan implementasi penelitian dalam penyusunan skripsi ini, dipergunakan *smartphone* produksi Sony Ericsson tipe W950i. Uji coba juga dilakukan terhadap telepon selular biasa, yakni W810i pada gambar 3.5 dan P800.

Gambar 3.5 *Smartphone* SonyEricsson tipe W950i

3. Divais Bluetooth : merupakan perangkat Bluetooth yang ditanamkan pada komputer *server*, berupa Bluetooth Dongle. Adapun, divais yang dipergunakan dalam penelitian tampak pada gambar 3.6 yaitu adalah Billionton, JAHT, dan produk Blutetooh tanpa merek khusus.

Gambar 3.6 Divais Bluetooth Dongle

3.3.3 Analisa Input dan Output

3.3.3.1 *Input Sistem*

Input pada sisi *client* adalah pilihan *server* dari daftar divais yang dihasilkan dari identifikasinisialisasi dan pencarian koneksi Bluetooth, serta daftar pilihan *server* dan file video yang diperoleh dari *server.*

Input pada sisi *server* adalah pesan proses koneksi dan pilihan layanan video yang diminta oleh pengguna.

3.3.3.2 *Output Sistem*

Output pada *client* berupa tampilan *video streaming* berformat .3gp yang diminta pengguna dan informasinya, serta transmisi informasi dan permintaan kepada *server* berupa pilihan pengguna melalui *user interface* dari *smartphone.*

Output pada server adalah pesan komunikasi berupa publikasi daftar video yang ditawarkan, dan layanan *video streaming* yang ditransmisikan kepada *client* (*smartphone*).

3.3.4 Analisa Pelaku

Secara umum, sistem terdiri dari dua pelaku interkasi, yaitu :

- *Client*, pihak yang hendak menggunakan *smartphone* unutk menikmati layanan aplikasi *video streaming.*
- *Administrator* sistem, pihak yang bertugas mengatur dan mengendalikan sistem pada *server*

3.4 PERANCANGAN

Tahap perancangan dilakukan dengan menggunakan metode berorientasi objek disertai pemanfaatan *Unified Modeling Language* (UML) yang berfungsi membantu melakukan dokumentasi, visualisasi, spesifikasi, dan konstruksi komponen-komponen sistem perangkat lunak.

3.4.1 Aplikasi pada koneksi RFCOMM

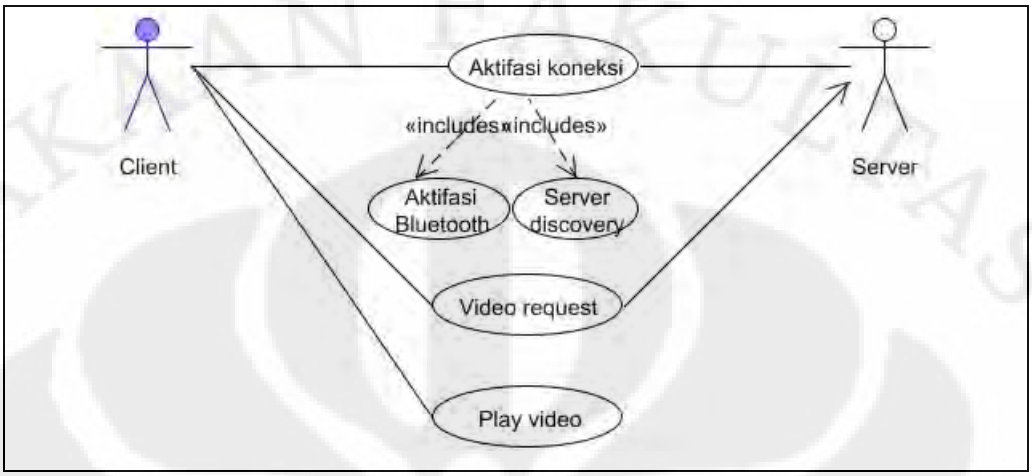
Skenario *use case* pada perancangan ini adalah :

Use Case :	Koneksi	
Aktor :	*Client* dan *Server*	
Deskripsi :	Sistem melakukan koneksi dengan pengaktifan koneksi Bluetooth oleh setiap aktor, dilanjutkan publikasi video oleh *server*, dan pemilihan video oleh *client*. *Server* pada PC dan *client* pada smartphone.	
Skenario :	Aktor	Koneksi Sistem
		1.Menyediakan pilihan menu pembuka
	2.Aktifasi koneksi *server*	
		3.Informasi video yang bisa ditawarkan aksesnya
	4.Publikasikan video yang ditawarkan *server*	
	5.Aktifasi koneksi *client*	
		6.Informasi video yang telah ditawarkan *server* dan dapat diakses oleh *client*
	7.Pemilihan video oleh *client*	

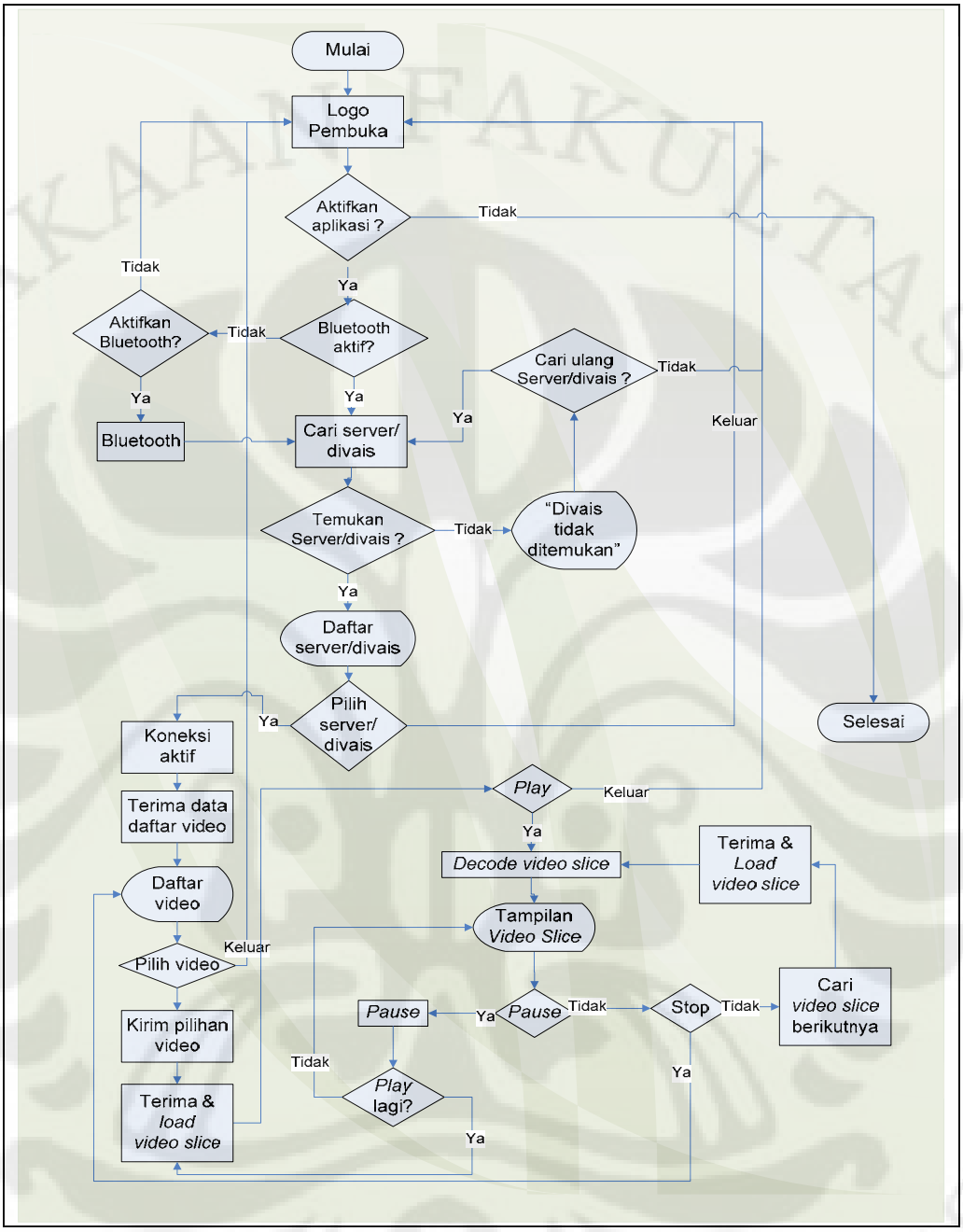
Use Case :	Permintaan (*request)*	
Aktor :	*Client*	
Deskripsi :	Sistem menyediakan info dan akses video yang dapat dimainkan bagi aktor setelah terjalin koneksi	
Skenario :	Aktor	Koneksi Sistem
		1.Daftar video yang dapat diakses
	2.Pemilihan video *client*	
		3.Koneksi dengan *server*
		4.Info bahwa video siap diakses
	5.*Play* video	
		6.Pengiriman video dari *server* dan *decode*
	7.Stop atau *pause*	

Proses inisialisasi merupakan tahapan awal yang harus dilakukan pada semua aplikasi yang menggunakan Bluetooth. Sehingga, dapat diasumsikan bahwa tahap inisialisasi dilakukan oleh kedua pihak (*server* dan *client*) setiap kali aplikasi dijalankan. Pada sisi *client*, tahapan berikutnya meliputi proses pencarian divais/*server* dan penjalinan koneksi serta penggunaan layanan yang disediakan. Sedangkan, tahapan pada sisi *server* meliputi penawaran atau publikasi layanan, penantian serta penanganan terhadap *client* hingga selesainya pemberian layanan. Diagram *use case* pada gambar 3.7 memberikan informasi mengenai perilaku atau

aksi yang diorganisasi ke dalam kumpulan transaksi atau interaksi sistem. Adapun, diagram alir aplikasi tampak seperti gambar 3.8.



Gambar 3.7 Diagram *Use Case* sisi *client* pada RFCOMM

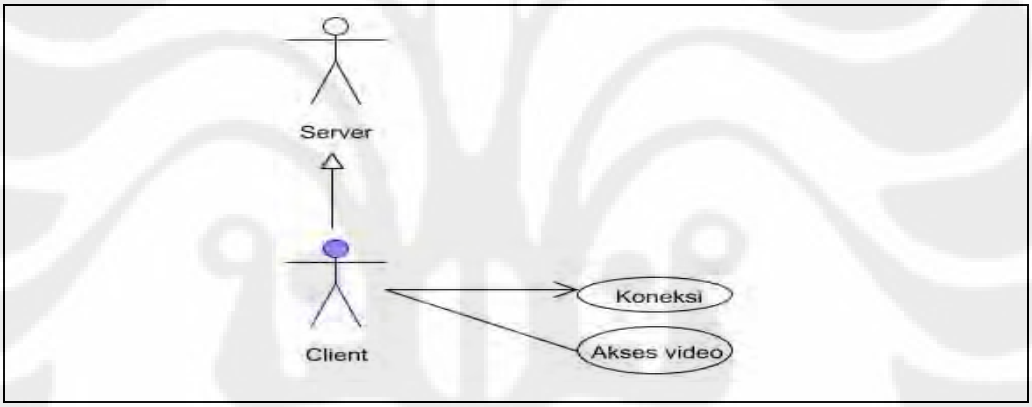


Gambar 3.8 Diagram alir aplikasi sisi *client* pada RFCOMM

3.4.2 Aplikasi pada koneksi OBEX

Pemanfaatan koneksi OBEX mempunyai keterbatasan dalam melakukan interaksi *server* dan *client*, yakni hanya bersifat *half-duplex*. Oleh karena itu, unsur interaksi antara keduanya dalam perancangan harus dikurangi. Diagram *use case* aplikasi ditunjukkan pada gambar 3.9.

***UseCase* : Kirim File**	
Penjelasan	Obex *server* mengirim file kepada divais yang dipilih oleh pengguna
Prioriitas	Penting
Frekuensi Penggunaan	Sering
Actor	Obex *server* berupa sebuah PC
Skenario	1. Jalankan aplikasi OBEX *Server* 2. Cari divais Bluetooth 3. List dari divais Bluetooth muncul 4. Pilih divais 5. Pilih file yang mau dikirim 6. Kirim file 7. Muncul pesan “file telah sukses terkirim”



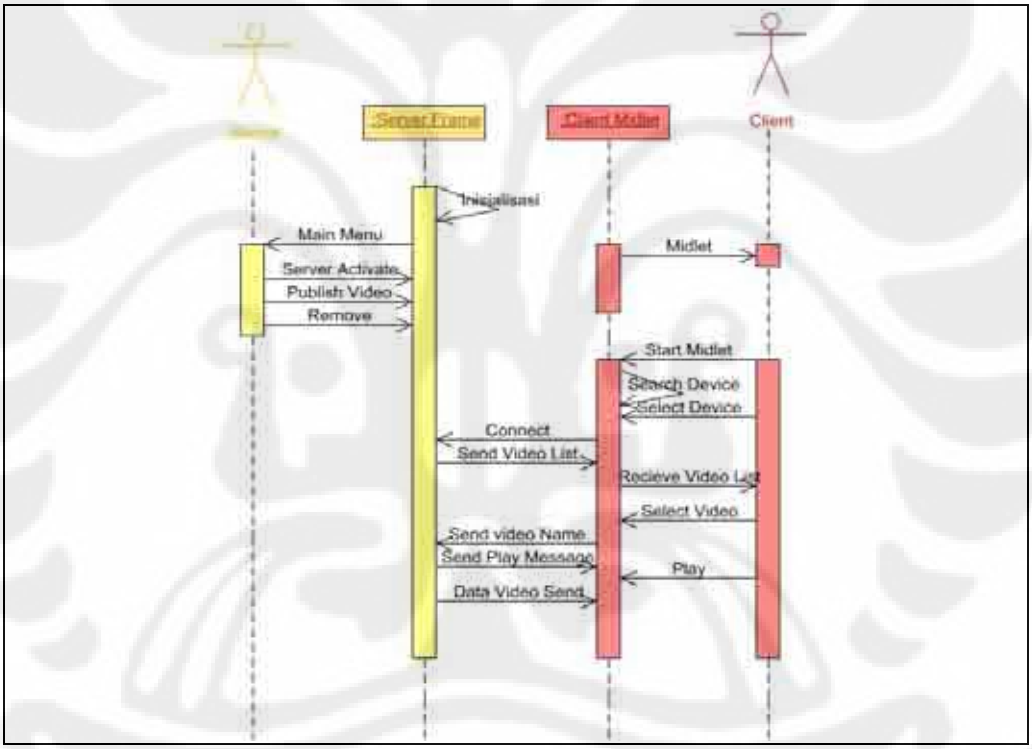
Gambar 3.9 Diagram *Use Case* sisi *client* pada OBEX

BAB IV
IMPLEMENTASI DAN ANALISA
USER INTERFACE APLIKASI

4.1 PENERAPAN PADA PROTOKOL KOMUNIKASI RFCOMM

4.1.1 Perancangan

Skenario lebih rinci beserta interaksi dan urutan aliran pesan antar objek dan pengguna pada sisi *client* berdasarkan waktu dapat dilihat melalui diagram *sequence* pada gambar 4.1.



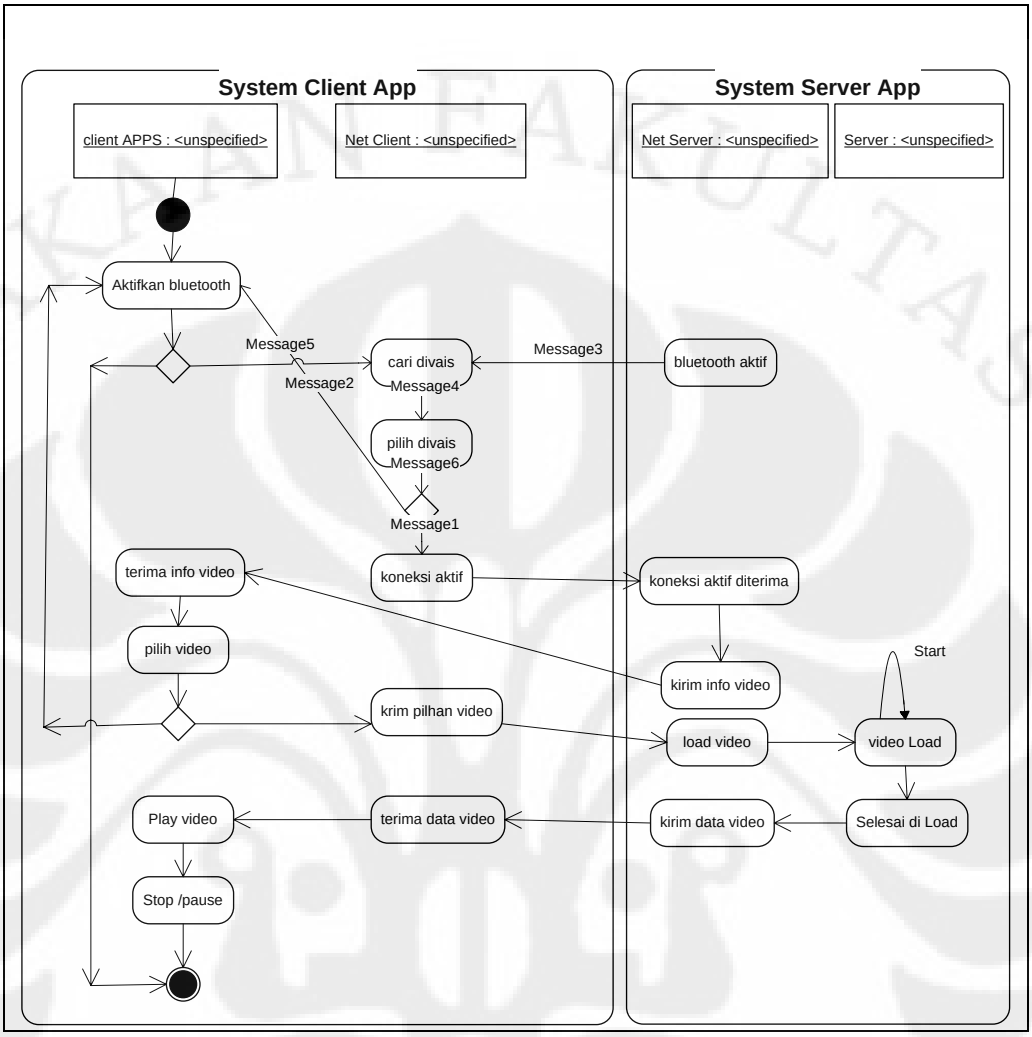
Gambar 4.1 Diagram *Sequence* pada RFCOMM

Pertama kali, sisi *client* diawali dengan melakukan aktifasi aplikasi dan Bluetooth serta dilanjutkan dengan pendeteksian divais berkoneksi Bluetooth aktif disekitarnya. Sedangkan, *server* memulai aplikasi dengan melakukan inisialisasi dan aktifasi *server* dan divais Bluetooth. Kemudian, *server* melalui administrator sebagai operatornya melakukan publikasi terhadap semua video yang dipilih untuk ditawarkan kepada *client* agar dapat diakses secara *pseudo streaming*. Setelah dilakukan pemilihan dan pembangunan koneksi terhadap *server* yang

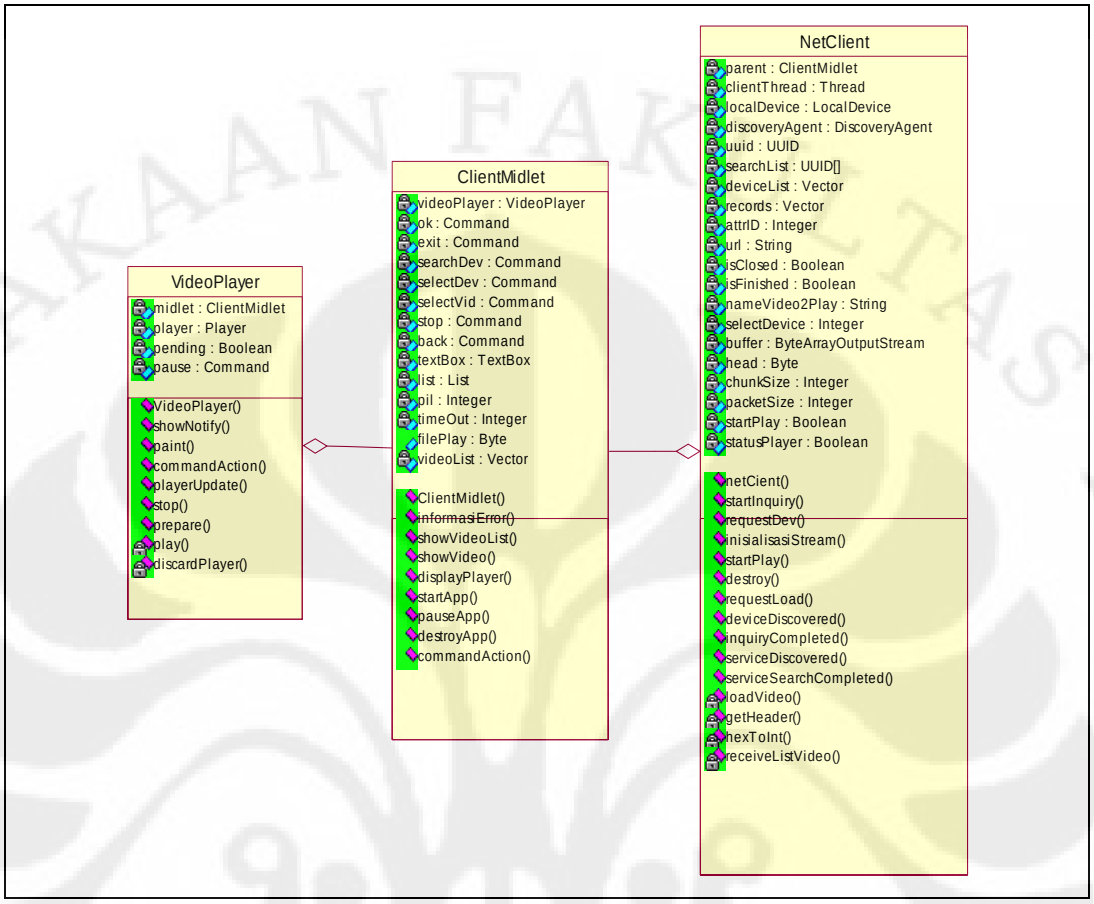
dituju, *client* akan mendapatkan daftar video yang ditawarkan administrator *server*.

Sebagai respon atas permintaan *client*, *server* akan melakukan *encoding* terhadap video. Pesan kesediaan *server* diberikan kepada *client* untuk mempersilakan kehendak pengguna dalam memulai dimainkannya video. Pilihan "*play*" ditindaklanjuti dengan proses *decode* oleh *client* terhadap paket-paket data video yang telah di transmisikan melalui Bluetooth dimana hasilnya akan ditampilkan secara *stream*. Proses pengiriman paket data video dilakukan terus menerus sampai paket-paket tersebut habis (video selesai di*streaming*kan). Namun, selama proses *streaming*, client dapat melakukan *pause* (berhenti sejenak) yang kemudian dapat di-*play* kembali, atau *stop.*

Alur proses kontrol aktivitas atau fungsi dari setiap metode dalam sistem aplikasi *client* dapat ditunjukkan oleh diagram aktivitas pada gambar 4.2. Adapun, kumpulan kelas, *user interface*, dan kerja sama serta hubungan antar satu kelas dengan lainnya kelas diilustrasikan oleh diagram kelas pada gambar 4.3. Aplikasi dibangun melalui tiga kelas, yakni NetClient, ClientMidlet, dan VideoPlayer. NetClient mempunyai fungsi proses koneksi antara *server* dan *client*, sedangkan VideoPlayer berfungsi sebagai dekoder dan pemutar video.



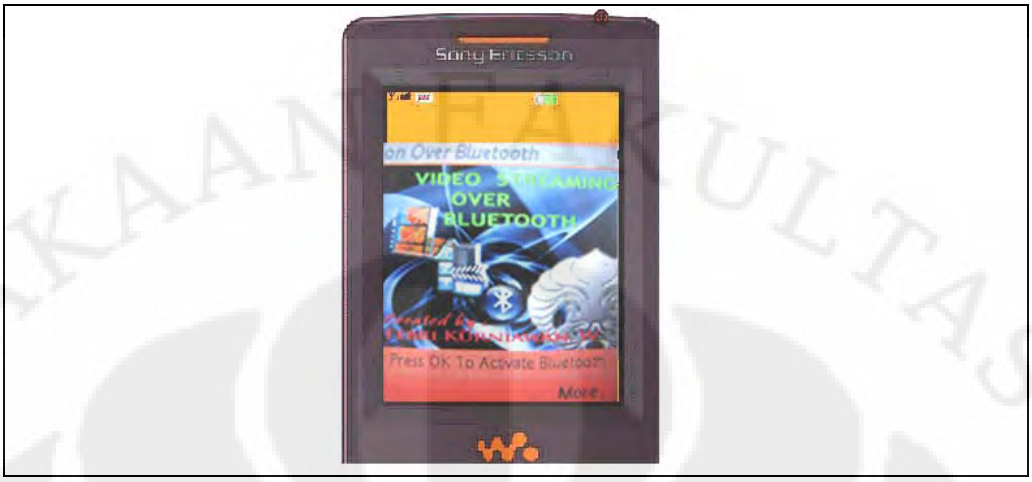
Gambar 4.2 Diagram aktivitas pada *client* pada RFCOMM



Gambar 4.3 *Class diagram* pada *client* pada RFCOMM

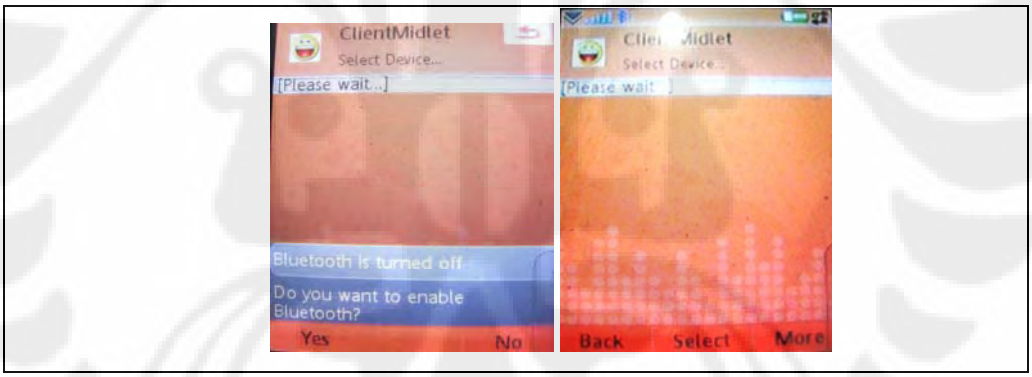
4.1.2 Implementasi dan Pengujian

Untuk dapat mengimplemetasikan aplikasi pada sisi *client*, terlebih dahulu dilakukan instalasi file berekstensi .jar pada *smartphone* yang mendukung dan memenuhi kriteria seperti yang telah disebutkan pada bab sebelumnya. Tampilan pembuka aplikasi akan meminta pengguna (*client*) untuk mengaktifkan Bluetooth sebelum aplikasi dijalankan, seperti tampak pada gambar 4.4. Pada *server*, untuk menjalankan aplikasi dibutuhkan instalasi *library* atau paket tambahan berupa paket BlueCove-20050514.jar yang ditambahkan pada *classpath* serta file "intelbth.dll" dari paket tersebut yang diletakkan pada C:\WINDOWS\SYSTEM32. Implementasi selanjutnya secara umum menggunakan Winsock (Microsoft) *Bluetooth Stack*.



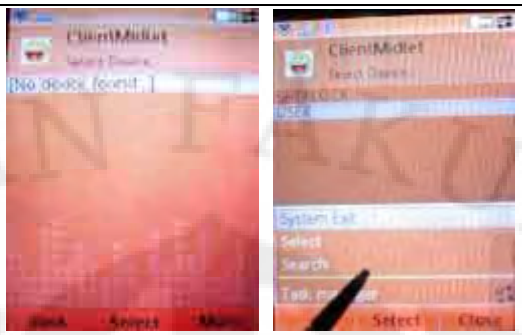
Gambar 4.4 Tampilan pembuka aplikasi pada *client* (*smartphone*)

Aplikasi diawali dengan opsi untuk mengaktifkan Bluetooth pada *smartphone* jika terdeteksi bahwa komponen tersebut belum aktif, seperti ditunjukkan pada gambar 4.5. Jika telah berstatus aktif, aplikasi secara otomatis akan melakukan pencarian divais (*device discovery*) berkoneksi Bluetooth di sekitarnya.



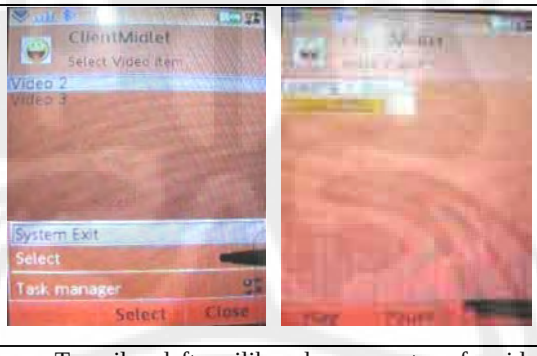
Gambar 4.5 Tampilan aktifasi Bluetooth dan pencarian divais pada *client*

Jika tidak ada divais yang ditemukan, maka *client* hanya dapat melakukan pencarian kembali atau keluar dari aplikasi tanpa melanjutkan. Nama divais yang telah teridentifikasi akan ditampilkan pada layar. Pada sesi ini, gambar 4.6, *client* dapat memilih salah satu divais atau *server* yang memberikan layanan video yang dikehendaki. Oleh karena itu, *client* diharuskan mengetahui terlebih dahulu nama *server* yang dimaksud.



Gambar 4.6 Tampilan opsi hasil pencarian divais pada *client*

Permintaan *client* akan direspon oleh *server* yang dipilih dalam bentuk pembangunan koneksi dan pengiriman daftar video yang ditawarkan. Proses ini, seperti halnya proses pencarian divais, berlangsung dalam waktu tertentu. Setelah daftar video ditampilkan, *client* dapat memilih salah satu video yang ada. Hal ini berlanjut dengan opsi *client* untuk melakukan instruksi "*play*" saat layar menampilkan "`Tekan play untuk memainkan video`", serta berjalannya proses transfer video sebagaimana tampak pada gambar 4.7. Video yang diterima akan di-*decode* agar dapat ditampilkan, dimana saat video dijalankan tersedia opsi "*pause*" dan "*exit*" untuk menghentikannya.



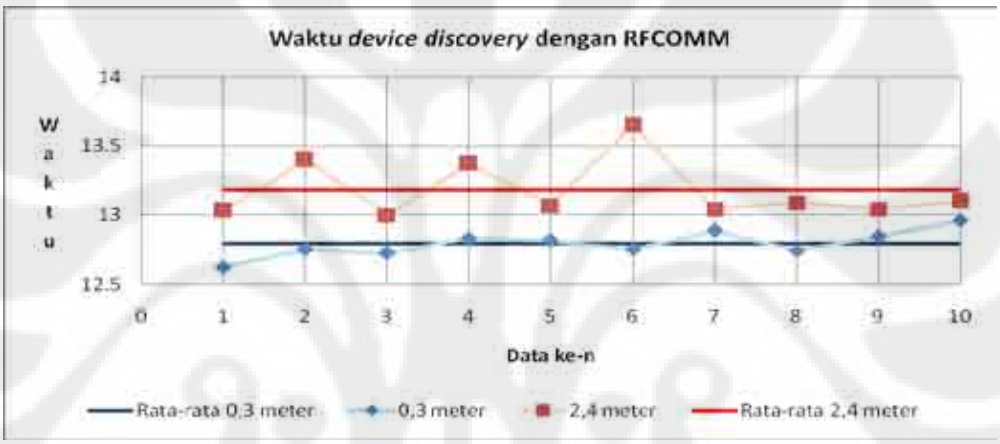
Gambar 4.7 Tampilan daftar pilihan dan proses transfer video pada *client*

Koneksi berlangsung setelah *client* dapat mengidentifikasi *server* memperoleh layanan. Pengujian dilakukan terhadap waktu yang dibutuhkan aplikasi untuk menemukan *server* pada jarak 0,3 dan 2,4 meter. Hasil pengujian yang diperoleh tercantum pada tabel 4.1 berikut.

Tabel 4.1 Waktu *device discovery* terhadap *server* menggunakan RFCOMM

Data ke - n	Waktu *device discovery* (detik)	
	pada jarak 0,3 m	pada jarak 2,4 m
1	12.63	13.03
2	12.76	13.4

3	12.73	13
4	12.84	13.37
5	12.82	13.07
6	12.76	13.65
7	12.89	13.04
8	12.75	13.09
9	12.85	13.04
10	12.96	13.1
Total	127.99	131.79
waktu rata-rata	12.799	13.179



Gambar 4.8 Grafik waktu *device discovery* menggunakan RFCOMM

Pengukuran tersebut dilakukan sebanyak sepuluh kali pada jarak 0,3 meter dan 2,4 meter. Pada gambar 4.8, tampak bahwa waktu yang dibutuhkan *client* untuk *device discovery* pada jarak 0,3 meter mempunyai rata-rata 12,799 detik, sedangkan pada jarak 2,4 meter rata-rata waktu yang dibutuhkan adalah 13,179 detik. Jika dibandingkan dengan waktu minimum pada *error-free environment* pada spesifikasi Bluetooth untuk *device discovery* yang bernilai 10,24 detik, nilai rata-rata waktu yang diperoleh pada pengukuran ini lebih lambat. Hal ini dipengaruhi oleh faktor mobilitas divais, resiko interferensi pada lingkungan yang *noise*, dan *multipath fading*. *Multipath fading* dapat terjadi akibat penerimaan daya yang lemah dan fluktuatif dengan disertai superposisi ataupun interferensi gelombang.

Berdasarkan perbandingan waktu rata-rata yang diperoleh, jarak antara *server* dan *client* dapat mempengaruhi waktu yang dibutuhkan dalam proses pencarian dan pembangunan koneksi *server*, walaupun nilainya tidak terlalu

signifikan, yakni hanya 0.38 detik. Namun, jika jarak antara *server* dan *client* semakin jauh maka waktu yang dibutuhkan dalam *device discovery* dan kemungkinan terjadinya *packet loss* akan semakin meningkat.

Kondisi s*erver* ditemukan tetapi tidak dapat memberikan layanan kepada divais dapat terjadi karena beberapa hal, antara lain aplikasi pada *server* belum aktif, terdapat divais lain dengan nama yang sama, dan s*erver* sibuk.

Implementasi aplikasi *video streaming* kurang didukung oleh penggunaan koneksi RFCOMM Bluetooth. Hal ini dikarenakan faktor ketidakstabilan RFCOMM dalam melakukan transfer data serta terdapatnya resiko kerusakan atau *corrupt* terhadap bit-bit pada file video yang ditransfer. Hal tersebut mengakibatkan proses *encode* terhadap file video yang cacat itu terhambat dan tidak dapat dijalankan dengan normal. Ketidakberhasilan proses pembacaan terhadap file video yang cacat pada penelitian yang dilakukan ditunjukkan berupa tampilan *white screen* atau *green screen,* dan kalimat peringatan pada Symbian OS *client* :

```
MediaException.
SymbianOS error  = -5. Not Supported VGDU.
handleBlockingOperation2.
```

Pesan `MediaException` menunjukkan bahwa file media tersebut tidak dapat dibaca atau *encode*. `SymbianOS error  = -5` merupakan pesan dari Symbian OS yang berarti operasi yang diminta tidak didukung akibat ketidakstabilan koneksi dan transmisi.

Penelusuran permasalahan dilakukan dengan tahapan :

1. Pengujian terhadap file video asli pada *server*
2. Pengujian terhadap file video hasil paketisasi dan segmentasi pada *server* sebelum dikirimkan
3. Pengujian terhadap file video asli dan hasil segmentasi pada *client* dengan metode transfer melalui kabel serial
4. Penelusuran algoritma rancang bangun aplikasi
5. Pengujian terhadap file video pada *client* menggunakan *video player* yang telah dibuat

6. Pengujian terhadap file video pada *client* menggunakan *video player* yang telah dibuat dengan metode transfer melalui koneksi Bluetooth OBEX
7. Pengujian analisa paket melalui upaya pembuatan *debugging* pada *streaming server*

Penelusuran dan pengujian dalam penelitian untuk mengetahui kendala yang ada memberikan hasil bahwa hingga tahapan pengujian ke enam berjalan dan berfungsi dengan normal. Format video yang diuji telah didukung secara teknis oleh *client.* Penelusuran algoritma, pengujian file video dan *video player* yang sama dengan kasus di atas menggunakan koneksi OBEX Bluetooth tidak menunjukkan adanya masalah. Pada *debugging* di *streaming server,* gambar 4.9, tampak bahwa *console* mengirimkan data sebanyak 7kB dan data di-*slice* ke dalam ByteArrayOutputStream dengan besar *slice* sebesar 1024b. dan setiap file data yang berhasil dikirim maka *client* memberikan pesan "Client Request FW" yang menandakan bahwa data telah diterima client. Cuplikan sebagian hasil *debugging* yang tampak yaitu :

```
Besar data : 7
Sending data : baos *** 6547

Waiting request client :
Client request : FW
55555
File ditemukan
Nilai p6657
Nilai pjg6
Besar data : 7
Sending data : baos *** 6661

Waiting request client :
.
.
```

Terlihat bahwa ukuran file video yang diterima dan dilaporkan oleh *client* nilainya selalu berubah-ubah, dari 6547 byte menjadi 6661 byte. Hal ini menunjukkan bahwa telah terdapat bagian file yang hilang atau bertambah selama proses berlangsung. Jika video akan ditampilkan, maka akan tampak pesan peringatan atau *green screen* seperti pada pembahasan di atas sebelumnya.

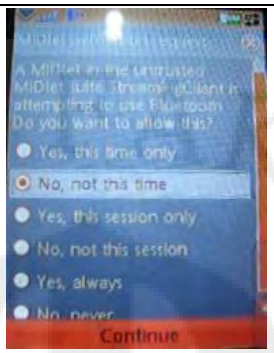
Oleh sebab itu, dapat diketahui atau disimpulkan bahwa telah terjadi file *corrupt* pada video yang diterima oleh *client* melalui koneksi RFCOMM Bluetooth sehingga tidak berhasil dijalankan. *Layer* RFCOMM tidaklah didukung oleh fitur akses pengaturan QoS. Selain itu, ditemukan adanya pernyataan dari

pihak produsen bahwa *smartphone* yang digunakan dalam penelitian ini tidaklah mendukung fasilitas *progressive download*.

Gambar 4.9 *Console debugging video streaming server*

Upaya proses *encode* terhadap aliran kiriman file video dalam bentuk potongan atau paket-paket kecil yang terus menerus akan memunculkan pesan peringatan dari Symbian OS berupa : `Kern-Exec -3`

Respon Symbian OS tersebut merupakan pesan bahwa operasi dibatalkan sehingga segera menutup aplikasi yang bersangkutan dengan pertimbangan sekuritas perangkat. Fleksibilitas pembuatan dan pengaksesan aplikasi pada perangkat *smartphone* berhadapan dengan sistem keamanan perangkat yang protektif, diantaranya melalui mekanisme konfirmasi berulang kali kepada pengguna pada gambar 4.10. Hal ini menyebabkan keterbatasan dalam hal akses dan pengembangan aplikasi terhadapnya.



Gambar 4.10 Tampilan contoh mekanisme sistem keamanan pada *smartphone*

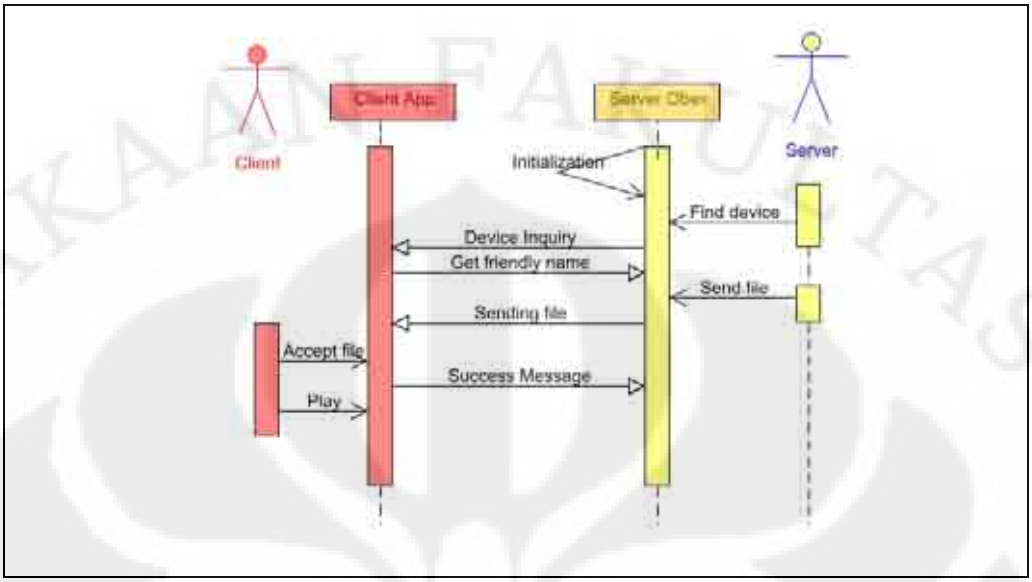
4.2 PENERAPAN PADA PROTOKOL KOMUNIKASI OBEX

4.2.1 Perancangan

Aplikasi menggunakan protokol komunikasi OBEX Bluetooth dirancang untuk melakukan transfer video dari *server* ke *client* sesuai kehendak administrator (sisi *server*), tanpa adanya kesempatan bagi *client* untuk memilih video yang diinginkan seperti halnya aplikasi pada penggunaan RFCOMM sebelumnya. Hal ini disebabkan karena tipe koneksi OBEX tidak memungkinkan interaksi dua arah seperti pada RFCOMM. Protokol transfer ini diperuntukkan bagi proses sinkronisasi, transfer file, dan *object push* dengan model *client-server*.

Diagram kelas aplikasi sisi *client* meliputi kelas FileMidle sebagai fungsi koneksi file, dan VideoPlayer sebagai dekoder dan pemutar video, sebagaimana ditunjukkan pada gambar 4.11.

Gambar 4.11 Diagram kelas *client*



Gambar 4.12 Diagram *sequence* pada OBEX

4.2.2 Implementasi dan Pengujian

Terlebih dahulu perlu dilakukan instalasi file berekstensi .jar pada *smartphone* yang mendukung. Instalasi *library* atau paket tambahan berupa paket BlueCove-20050514.jar yang ditambahkan pada *classpath* serta file "intelbth.dll" dari paket tersebut yang diletakkan pada C:\WINDOWS\SYSTEM32 harus dilakukan pada *server* agar dapat menjalankan aplikasi. Implementasi secara umum masih menggunakan Winsock (Microsoft) *Bluetooth Stack.*

Diawali dengan aktifasi Bluetooth, *client* menunggu adanya kiriman video dari administrator sisi *server* sebagai pihak yang menentukan kapan dan siapa *client* yang akan mendapatkan kiriman video. Setelah *server* menentukan file video dan melakukan instruksi pengiriman kepada *client,* video akan terkirimkan secara langsung. Pada konfigurasi status *server* pada *client* sebagai *trust device/server,* maka proses permintaan persetujuan atau autentikasi tidak lagi diperlukan. Proses pengiriman file video termonitor dalam tampilan layar *client* meliputi nama file dan ukurannya yang telah terkirimkan seperti tampak pada gambar 4.12. Video yang diterima akan tersimpan sementara pada memori *buffer* yang akan hilang atau terhapus setelah tidak dipergunakan.

Gambar 4.13 Proses penerimaan video dari *server*

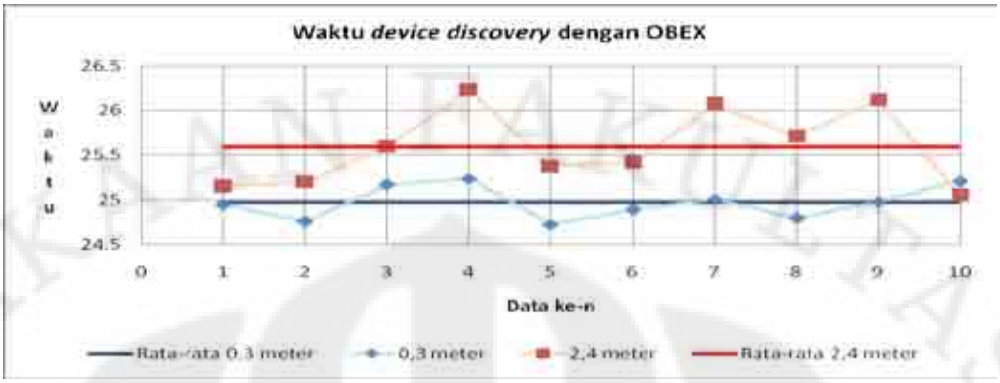
Pengujian dilakukan yaitu :

1. Waktu *device discovery* terhadap perubahan waktu pada penggunaan OBEX
2. Kecepatan dan waktu yang dibutuhkan terhadap ukuran file saat transfer menggunakan OBEX

Waktu yang dibutuhkan aplikasi *server* untuk menemukan *client* pada perubahan jarak tertentu yang diukur sejumlah sepuluh kali menghasilkan data pada tabel 4.2.

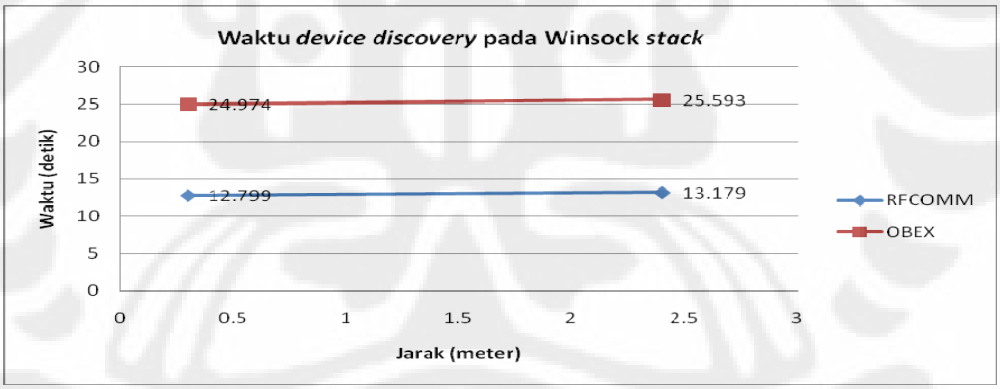
Tabel 4.2 Waktu *device discovery* menggunakan OBEX

Data ke - n	Waktu *device discovery* (detik)	
	pada jarak 0,3 m	pada jarak 2,4 m
1	24.95	25.15
2	24.76	25.2
3	25.17	25.6
4	25.24	26.23
5	24.73	25.38
6	24.89	25.42
7	25.01	26.07
8	24.8	25.71
9	24.98	26.12
10	25.21	25.05
Total	249.74	255.93
waktu rata-rata	24.974	25.593



Gambar 4.14 Grafik waktu *device discovery* menggunakan OBEX

Pada gambar 4.13, tampak bahwa waktu yang dibutuhkan *client* untuk *device discovery* pada jarak 0,3 meter mempunyai rata-rata 24.974 detik, sedangkan pada jarak 2,4 meter rata-rata waktu yang dibutuhkan adalah 25.593 detik. Rata-rata waktu yang dibutuhkan sedikit bertambah pada seiring pertambahan jarak antar divais. Sedangkan, tampak bahwa nilai rata-rata waktu *device discovery* yang diperoleh pada pengukuran dengan OBEX lebih lambat dibandingkan dengan waktu *device discovery* pada RFCOMM. Hal ini dapat ditunjukkan pada grafik gambar 4.15.



Gambar 4.15 Grafik perbandingan waktu *device discovery*

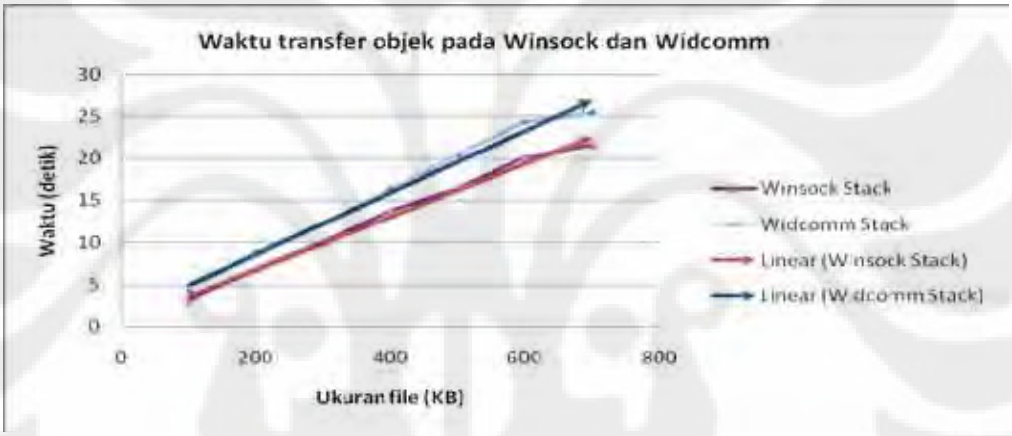
4.3 PENERAPAN WINSOCK DAN WIDCOMM

Penggunaan sistem operasi Microsoft Windows XP *Service Pack* 2 telah mengandung Bluetooth *stack* Microsoft yang disebut juga Winsock. Namun, koneksi yang didukungnya terbatas. Bluetooth *stack* lainnya yang ada yaitu Widcomm (Broadcom), Blue Soleil, dan Mac OS X. Pada penelitian ini, dilakukan pengukuran terhadap kecepatan transfer file video melalui koneksi

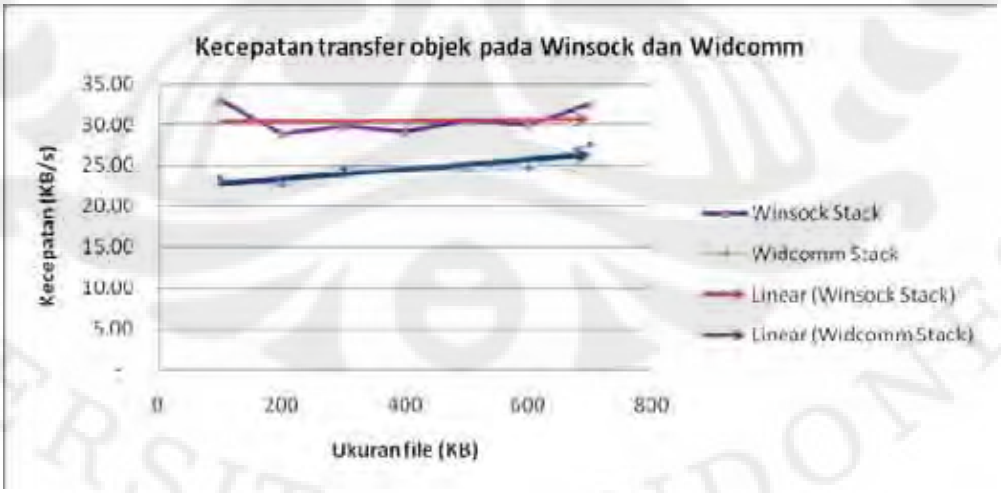
OBEX menggunakan *stack* Winsock dan Widcomm yang menghasilkan data pada tabel 4.3.

Tabel 4.3 Perbandingan kecepatan transfer Winsock dan Widcomm

Data ke-n	Ukuran file (KB)	Lama pengiriman (s)		Kecepatan transfer (KB/s)	
		Pada Winsock stack	Pada Widcomm stack	Pada Winsock stack	Pada Widcomm stack
1	100	3.04	4.28	32.89	23.36
2	200	6.92	8.86	28.90	22.57
3	300	10.07	12.21	29.79	24.57
4	400	13.74	16.32	29.11	24.51
5	500	16.41	20.36	30.47	24.56
6	600	20.04	24.32	29.94	24.67
7	700	21.56	25.42	32.47	27.54



Gambar 4.16 Grafik waktu transfer pada Winsock dan Widcomm



Gambar 4.17 Grafik kecepatan transfer pada Winsock dan Widcomm

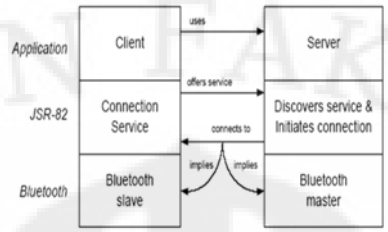
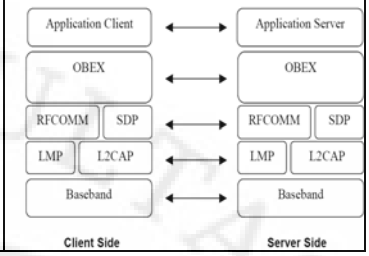
Berdasarkan data yang diperoleh, dapat dihasilkan grafik pada gambar 4.15 dan 4.16. Melalui grafik, dapat diketahui bahwa kecepatan transfer objek

akan semakin menurun jika jumlah data yang ditransfer semakin besar. Hal ini disebabkan oleh faktor kecepatan tulis pada memori flash yang rendah dengan pengisian *buffer* dan prosesor yang berkemampuan rendah pada perangkat penerima. Kecepatan transfer pada Winsock lebih besar daripada Widcomm karena dipengaruhi kecepatan akses *stack* pada sistem operasi yang telah terintegrasi pada *server* dan kompatibilitas divais yang lebih baik.

Berdasarkan pengujian dan spesifikasi teknis yang dimiliki, maka perbandingan antara RFCOMM dan OBEX dapat dituliskan pada tabel 4.4.

Tabel 4.4 Perbandingan RFCOMM dan OBEX

	RFCOMM	**OBEX**
Waktu *Device Discovery* (terhadap parameter standar *free-error* = 10,24 detik)	Lebih mendekati standar, lebih cepat (rata-rata = 12,989 detik)	Lebih jauh dari standard, lebih lambat (rata-rata = 25,284 detik), karena posisi *layer* lebih tinggi dan butuh lebih banyak enkapsulasi
Pengaruh jarak terhadap waktu *device discovery*	Ya	Ya
Pengaruh tipe Bluetooth *stack* (Winsock & Widcomm) terhadap waktu *device discovery*	Ya	Ya
Reliabilitas proses transfer Objek/data	Kurang stabil, berdampak file *corrupt* atau *packet loss*	Lebih stabil, file utuh (lengkap)
Metode Koneksi	Stream / binary (DataInputStream, DataOutputStream). ByteArrayOutputStream mempunyai batas maksimum 80b	Object (*get, put)*
Modifikasi *header* file	Tidak bisa	Bisa (nama, panjang, deskripsi)
Profil Bluetooth	LAN *Acces Protocol*	*Object Push, File Transfer, Synchronization.*
Aplikasi yg cocok	*Bluetooth Chat* (pesan dengan atribut string)	*Bluetooth File Transfer* (objek berupa file)
Paket dalam JSR-API82	Javax.bluetooth	Javax.obex
URL koneksi	`btspp:// localhost:" + uuid;authenticate=true;authorize=true;encrypt=true`	`"btgoep://localhost:" + uuid +;authenticate=false;master=false;encrypt=false";`
OSI *Layer*	Transport	Sesion
Kompleksitas implementasi	Tinggi	Menengah
Bagian Termodifikasi	Perangkat Lunak	Perangkat lunak

Dukungan point-to-multipoint	Ya	Ya
Diagram Blok		

BAB V
KESIMPULAN

1. Jarak antara *server* dan *client* yang semakin jauh maka berdampak pada pertambahan waktu *device discovery* dan meningkatnya resiko terjadi *packet loss*.
2. Waktu *client* untuk *device discovery* pada koneksi RFCOMM mempunyai rata-rata 12,989 detik, sedangkan pada koneksi OBEX adalah 25,284 detik. Waktu rata-rata keduanya lebih lambat dibandingkan waktu minimum pada *error-free environment* pada spesifikasi Bluetooth yang bernilai 10,24 detik. Hal ini dapat disebabkan paket data *inquiry* belum diterima oleh divais bluetooth akibat pengaruh faktor mobilitas divais, resiko interferensi lingkungan yang *noise, multipath fading*, dll.
3. Aplikasi *video streaming* kurang didukung oleh koneksi RFCOMM karena ketidakstabilannya dalam mentransfer data sehingga timbul resiko kerusakan atau *corrupt* terhadap bit-bit file video yang ditransfer yang berdampak pada gagalnya proses *encode*. *Layer* RFCOMM tidak didukung oleh fitur akses pengaturan QoS.
4. Sistem keamanan perangkat yang protektif menyebabkan keterbatasan fleksibilitas pembuatan dan pengaksesan aplikasi pada perangkat *smartphone.*
5. Kecepatan transfer objek semakin menurun jika jumlah data yang ditransfer semakin besar. Hal ini disebabkan oleh faktor kecepatan tulis memori flash yang rendah dan prosesor berkemampuan rendah pada perangkat penerima. Kecepatan transfer pada Winsock lebih besar daripada Widcomm karena dipengaruhi kecepatan akses *stack* pada sistem operasi yang telah terintegrasi pada *server* dan kompatibilitas divais yang lebih baik.
6. Protokol OBEX lebih baik dalam proses transfer file daripada RFCOMM karena memiliki profil GEOP (General Object Push), *object push* dan sinkronisasi. Namun, OBEX bukan merupakan protocol komunikasi sehingga hanya dapat melakukan transfer secara *half duplex*.

DAFTAR ACUAN

[1] John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee, "Video Streaming : Concepts, Algorithms, and Systems". Mobile and Media Systems Laboratory, Hewlett-Packard Paolo Alto, USA. Diakses 20 Oktober 2007 dari Hewlett-Packard Labs.
www.hpl.hp.com/techreports/2002/HPL-2002-260.pdf

[2] Donny B.U., "Streaming: Membuat File Besar Serasa Kecil", 2002. Diakses 10 Oktober 2007.
www.free.vlsm.org/v17/com/ictwatch/paper/paper018.htm

[3] Onno W. Purbo, "Kebutuhan Infrastruktur Untuk Video Conference". Diakses 30 Oktober 2007 dari situs Dr. Onno W Purbo.
www.onno.vlsm.org/v11/onno-ind-3/network/kebutuhan-infrastruktur-untuk-**video**-conference-3-2003.rtf

[4] ITU-T. *Line Transmission of Non-Telephone Signals.* ITU-T Recommendation H.261 (03/93). 1994. Diakses 22 November 2007 dari situs ITU-T.
www.itu.int/rec/T-REC-H.261-199303-I/en

[5] Tesi di Laurea, "Confronto tra due protocolli per reti Wireless: IEEE 802.11 e Bluetooth", 2001. Diakses 30 November 2007
http://fly.isti.cnr.it/didattica/tesi/Tallarico/tesi

[6] Keven Boyett, "Protocol Tests Lay Groundwork for Bluetooth Success", Tektronix, 2001. Diakses 30 November 2007
http://archive.e.valuationengineering.com/archive/articles/0801blue.htm

[8] Sun Microsystems, "JDK 5.0 Documentation". Diakses 9 November 2007 dari Sun Microsystem.
http://java.sun.com/j2se/1.5.0/docs/index.html

[9] Salahudin, M dan Rosa A.S, *Pemrograman J2ME*. (Bandung : Informatika, 2006).

[10] Martin de Jode, *Programming Java 2 Micro Edition on Symbian OS*, A developer's guide to MIDP 2.0. (Inggris: John Wiley & Sons Ltd, 2004), hal 4.

[13] Wang Xiaohang, "Video Streaming over Bluetooth: A Survey". Institute for Infocomm Research (I2R), Singapura. Diakses 20 Agustus 2007 dari CiteSeer.IST
http://www.citeseer.ist.psu.edu/708770.html

DAFTAR PUSTAKA

Apostolopoulos, John G., Wai-tian Tan, Susie J. Wee, "Video Streaming : Concepts, Algorithms, and Systems". Mobile and Media Systems Laboratory, Hewlett-Packard Paolo Alto, USA. Diakses 20 Oktober 2007 dari Hewlett-Packard Labs. www.hpl.hp.com/techreports/2002/HPL-2002-260.pdf

Austerberry, David. *The Technology of Video and Audio Streaming* (Inggris : Elsevier, 2005).

Boyett, Keven, "Protocol Tests Lay Groundwork for Bluetooth Success", Tektronix, 2001. Diakses 30 November 2007 http://archive.e.valuationengineering.com/archive/articles/0801blue.htm

de Jod, Martin. *Programming Java 2 Micro Edition on Symbian OS.* A developer's guide to MIDP 2.0. (Inggris: John Wiley & Sons Ltd, 2004).

Demetriades, Gregory C. *Streaming Media*, Building and Implementing a Complete Streaming System. (USA : Wiley Publishing Inc., 2003)

Hopkins, Bruce, Ranjith Antony. *Bluetooth for Java*. (USA : Apress, 2003).

ITU-T. *Line Transmission of Non-Telephone Signals.* ITU-T Recommendation H.261 (03/93). 1994. Diakses 22 November 2007 dari situs ITU-T. www.itu.int/rec/T-REC-H.261-199303-I/en

Kapoor, R., Matteo Cesana, Mario Gerla , "Link Layer Support for Streaming MPEG Video over Wireless Links", International Conference on Computer Communications and Networks (ICCCN 2003), 2003. Diakses 20 Oktober 2007 dari UCLA Computer Science Department.

Klingsheim, Andre N. "J2ME Bluetooth Programming." Tesis, Program Pasca Sarjana Departemen Informatika Universitas Bergensis, Norwegia, 2004.

M, Salahudin, dan Rosa A.S. *Pemrograman J2ME*. (Bandung : Informatika, 2006).

Purbo, Onno W. "Kebutuhan Infrastruktur Untuk Video Conference". Diakses 30 Oktober 2007 dari situs Dr. Onno W Purbo. www.onno.vlsm.org/v11/onno-ind-3/network/kebutuhan-infrastruktur-untuk-**video**-conference-3-2003.rtf

Raharjo, B., Imam Haryanto, Arif Haryono. *Tuntunan Pemrograman Java untuk Handphone.* (Bandung : Informatika, 2007).

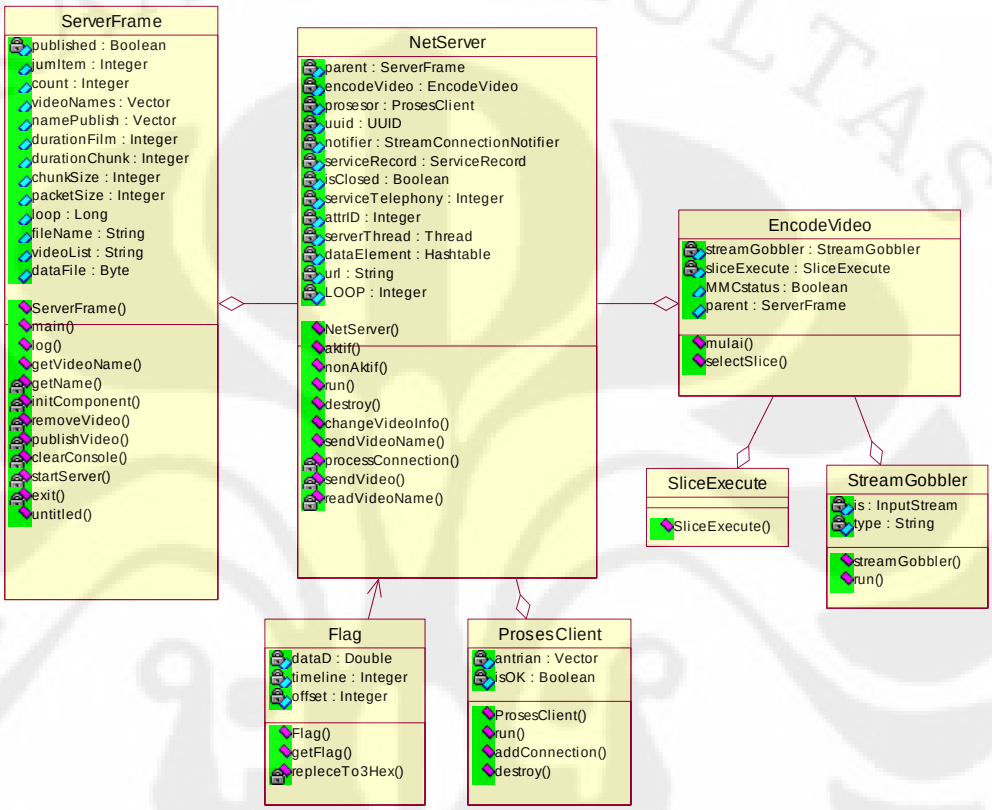
Sun Microsystems, “JDK 5.0 Documentation”. Diakses 9 November 2007 dari Sun Microsystem.
http://java.sun.com/j2se/1.5.0/docs/index.html

Sony Ericsson. Music and Video in Sony Ericsson phones. November 2007.

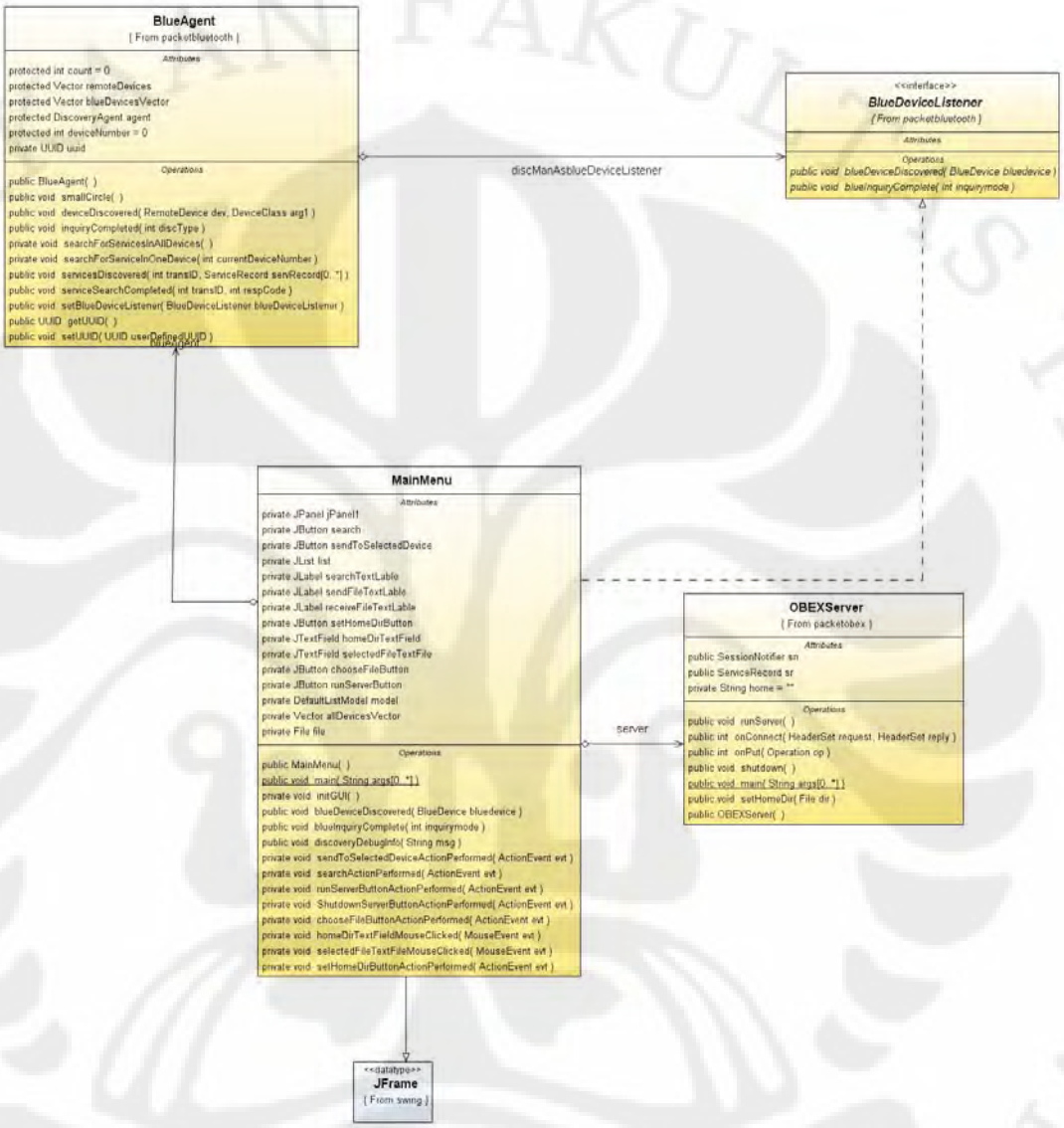
Xiaohang, Wang “Video Streaming over Bluetooth: A Survey”. Institute for Infocomm Research (I2R), Singapura. Diakses 20 Agustus 2007 dari CiteSeer.IST
http://www.citeseer.ist.psu.edu/708770.html

LAMPIRAN

Lampiran 1 *Class diagram* sisi *server* pada RFCOMM



Lampiran 2 *Class diagram* sisi *server* pada OBEX



Lampiran 3 Ketidukungan *smartphone* yang diuji terhadap *progressive download*

requirements within this area. The relevant 3GPP specification is TS 26.234, "Transparent end-to-end packet switched streaming service (PSS)". The PSS includes media codecs for video, still images, bitmap graphics, text, audio, and speech.

Downloaded media

Whereas streamed media is aiming at making media available in real-time, downloaded media allows the user to bring and enjoy the media wherever. The media is stored locally on the mobile phone, in the built-in memory or on a memory card. Downloaded media has also a much wider range than streamed media, since it could be still pictures, themes, ringtones, as well as video or audio.

Transfer method selection is more flexible for downloading than for streaming. Downloaded files could be transferred to the phone via all the different connectivity modes explained earlier this document. This gives the user greater possibilities to choose the preferred or most suited transfer method. There are cases though when this is limited by the media provider.

Note: The K600, K750, V600, W700, W800, W810, Z520 and Z525 series of phones support the WTP protocol, which limits the size of downloaded files to ~300 kB. This limitation can be avoided by setting an Internet profile using the HTTP protocol instead, in which case the size of downloaded files is limited only by available user memory in the phone. All other phones in this document do not support the WTP protocol and use HTTP by default, therefore allowing unlimited download file sizes. However, gateways and proxy servers may also limit the maximum size of downloaded files, due to operator or service provider settings.

Downloaded files often have a need to be copyright protected and might have other copyright issues. Sony Ericsson has a DRM (Digital Rights Management) solution defined by the OMA (Open Mobile Alliance), described in the DRM and Sony Ericsson section of this document.

Progressive download

Note: The K600, K750, M600, P1, P990, V600, W700, W800, W810, W950, W960, Z520 and Z525 series of phones do **not** support progressive download.