BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan proses pengimplementasian sistem yang telah dibuat. Penerapan ini sesuai dengan perancangan yang telah dibuat pada bab metodologi penelitian dan perancangan.

4.1 Lingkungan pengujian sistem

Rancangan lingkungan pengujian sistem terdiri dari aplikasi yang mendukung simulasi terhadap pengujian performa dari setiap protokol *file sharing*. Komponen yang digunakan untuk melakukan simulasi terhadap kondisi lingkungan pengujian adalah sebagai berikut:

- VMware Workstation. VMware Workstation adalah aplikasi yang berfungsi untuk melakukan virtualisasi komputer terhadap sistem operasi dan komunikasi antar komputer yang dibuat. Dalam penelitian ini aplikasi VMware Workstation digunakan untuk membuat *virtual machine* yang nantinya digunakan untuk menjalankan program *file sharing* sesuai dengan protokol yang telah ditentukan dalam penelitian ini. *Virtual machine* yang dibuat menggunakan sistem operasi Ubuntu 14.04 LTS yang didalamnya akan dilakukan instalasi *Java Runtime Environment* dan aplikasi yang digunakan untuk melakukan *file sharing*. Untuk dapat berkomunikasi dengan *peer* lain melalui aplikasi *file sharing* yang digunakan sesuai dengan topik penelitian, mode *network adapter* yang digunakan adalah *Bridged*. Mode ini digunakan agar *virtual machine* tersambung langsung dengan jaringan FILKOM.
- Java Runtime Environment. Java Runtime Environtment (JRE) adalah sebuah aplikasi yang dapat membuat lingkungan *java* sehingga program atau aplikasi berbasis bahasa pemrograman *java* dapat dijalankan.
- Wireshark. Wireshark adalah aplikasi berbasi *open-source* yang berfungsi untuk melakukan proses analisis secara detail terhadap paket-paket data yang berjalan dan proses pertukaran data yang berlangusng di dalam berbagai macam jaringan komputer. Dalam penelitian ini, aplikasi Wireshark digunakan untuk melakukan analisis terhadap trafik yang berjalan ketika proses pertukaran data pada aplikasi *file sharing* sedang berlangsung. Parameter yang digunakan untuk melakukan analisis terhadap performansi aplikasi *file sharing* adalah berdasarkan protokol transport yang digunakan yaitu TCP.

Langkah pertama dalam implementasi lingkungan pengujian sistem diawali dengan installasi dan konfigurasi empat *Virtual Machine* (VM) menggunakan sistem operasi Ubuntu 14.04 LTS pada VMware Workstation dengan mode jaringan *Bridged* pada masing-masing VM. Setelah VM terinstall, dilakukan konfigurasi IP *address* pada setiap VM dan komputer yang digunakan.

Selanjutnya melakukan proses installasi JRE pada setiap VM dan komputer yang digunakan sebagai *peer* untuk melakukan *file sharing*. Setelah konfigurasi pada seluruh VM dan komputer berhasil diimplementasikan, selanjutnya yaitu melakukan installasi aplikasi Gnucleus dan myjxta pada setiap VM dan komputer yang akan digunakan sebagai *peer*.

Untuk melakukan analisis terhadap performansi kedua aplikasi *file sharing*, berikutnya dilakukan installasi aplikasi Wireshark pada *peer* yang berperan sebagai *Edge/Leaf peer* dan *receiver*.

4.1.1 Konfigurasi *VMware Workstation*

VMware Workstation digunakan untuk mengimplementasikan lingkungan jaringan komputer untuk pengujian aplikasi *file sharing*. VMware Workstation yang digunakan adalah versi 11.0.0. Berikut langkah-langkah implementasi dan konfigurasi yang dilakukan pada aplikasi VMware:

1. Membuat satu VM pada setiap komputer yang digunakan untuk melakukan pengujian menggunakan sistem operasi Ubuntu 14.04.
2. Setiap VM menggunakan alokasi memori sebesar 1 GB dan alokasi *hard disk* sebesar 20 GB untuk menunjang proses yang berjalan pada VM
3. Konfigurasi *Network Adapter* menggunakan mode *Bridged*.
4. Melakukan proses installasi sistem operasi pada masing-masing VM sesuai dengan sistem operasi yang telah ditentukan.
5. Konfigurasi IP *address* secara statis pada seluruh VM berdasarkan perancangan sebelumnya sebagai berikut.

Lab Kecerdasan Visual		Lab Game	
RDV/Ultra peer (PC)	10.34.17.244	RDV/Ultra peer (PC)	10.34.9.103
Edge/Leaf peer (PC)	10.34.17.197	Edge/Leaf peer (PC)	10.34.9.104
Edge/Leaf peer (VM)	10.34.17.200	Edge/Leaf peer (VM)	10.34.9.102
Edge/Leaf peer (VM)	10.34.17.210	Edge/Leaf peer (VM)	10.34.9.100

Tabel 4.1 IP *address* pada setiap komputer dan VM

Tabel 4.1 menunjukkan distribusi alamat IP terhadap komputer dan VM yang digunakan pada pengujian sesuai dengan perancangan yang ditunjukkan pada Gambar 3.2. Pada setiap lokasi komputer dan VM yang digunakan untuk pengujian terdapat masing-masing satu *peer* yang berperan sebagai *RDV* dan *Ultrapeer* yaitu komputer dengan alamat IP 10.34.17.244 pada Lab kecerdasan buatan dan 10.34.9.103 pada Lab game. Sedangkan untuk *peer* yang berperan sebagai *edge* atau *leaf* berjumlah tiga yaitu dengan masing-masing komputer dan VM yang memiliki alamat IP 10.34.17.197, 10.34.17.200, dan 10.34.17.210 pada Lab kecerdasan buatan dan alamat IP 10.34.9.104, 10.34.9.102, dan 10.34.9.100 pada Lab game.

Setelah konfigurasi selesai, installasi aplikasi *file sharing* dapat dilakukan pada setiap VM untuk menjalankan mekanisme *file sharing*. Selanjutnya dilakukan

konfigurasi pada setiap aplikasi *file sharing* dan menyiapkan *dummy file* yang diperlukan untuk melakukan mekanisme *file sharing*. Penggunaan *dummy file* pada pengujian aplikasi *file sharing* ini bertujuan untuk memudahkan dalam penentuan besaran *file* yang akan dipakai karena besaran *file* pada *dummy file* jumlahnya tetap.

4.1.2 Installasi *Java Runtime Environment*

Java Runtime Environment (JRE) digunakan agar aplikasi *file sharing* dapat berjalan dan dapat digunakan. JRE diperlukan karena aplikasi *myjxta* dan *Gnucleus* berbasis bahasa pemrograman *java* sehingga aplikasi dapat dijalankan karena JRE dapat membuat sebuah *virtual environment* yang dibutuhkan oleh aplikasi berbasis *java*. Penelitian ini menggunakan JRE versi 6.0.17.

Installasi JRE dilakukan pada setiap komputer dan VM yang digunakan dalam pengujian *file sharing*. Prosedur installasi yang dilakukan berbeda antara komputer dan VM yang menggunakan sistem operasi Windows 7 Ultimate 64bit dan Ubuntu LTS 14.04.

Pada komputer yang menggunakan sistem operasi Windows 7 Ultimate 64bit, installasi dilakukan dengan mengeksekusi *file* installasi jre-6u17-windows64.exe yang didapatkan dari mengunduh *file* tersebut pada halaman web http://www.oracle.com/technetwork/java/javase/downloads/index.html. Setelah eksekusi *file* dilakukan, proses installasi akan berjalan dan installasi JRE telah selesai diimplementasikan sehingga dapat digunakan.

```
1  $ sudo apt-get update
2  $ java -version
3  $ sudo apt-get install default-jre
```

Kode 4.1 Perintah installasi JRE pada *terminal*

Sedangkan pada VM yang menggunakan sistem operasi Ubuntu LTS 14.04, prosedur installasi dimulai dengan melakukan beberapa perintah pada *terminal* yang ditunjukkan pada Kode 4.1. Perintah pertama bertujuan untuk memperbarui daftar paket didalam *repository* yang terdaftar. Perintah kedua dilakukan untuk memeriksa apakah JRE telah terinstall sebelumnya. Jika JRE belum tersedia pada sistem, selanjutnya perintah ketiga dijalankan yang berfungsi untuk melakukan installasi terhadap JRE kepada sistem dengan cara mengunduh paket-paket yang diperlukan dari *repository* yang selanjutnya akan dilakukan installasi terhadap sistem.

4.1.3 Implementasi Wireshark

Wireshark digunakan sebagai aplikasi monitoring terhadap performansi proses *file sharing* yang dilakukan oleh aplikasi *file sharing*. Wireshark diimplementasikan pada *Edge/Leaf peer* yang mengunduh *file* yang dibagikan oleh *Edge/Leaf peer* lainnya (*seeder*) untuk melakukan monitor terhadap trafik yang masuk dari *seeder* melalui *network interface* yang digunakan dalam komunikasi dengan *peer* yang lainnya. Versi Wireshark yang digunakan dalam

penelitian ini yaitu versi 2.0.4. Wireshark akan melakukan monitoring terhadap performansi dari aplikasi *file sharing* yang diujikan sesuai dengan protokol transport yang digunakan pada masing-masing aplikasi dalam melakukan mekanisme *file sharing*.

Hasil monitoring oleh Wireshark selanjutnya dilakukan *filter* untuk menemukan nilai dari parameter yang telah ditentukan sebelumnya yaitu rata-rata *delay* dan *troughput*. Nilai *throughput* dan rata-rata *delay* tidak diperoleh langsung dari wireshark, melainkan diperoleh dari perhitungan lebih lanjut terhadap nilai-nilai yang ditampilkan dan dihitung menggunakan rumusan tertentu.

Untuk memperoleh *throughput*, dilakukan perhitungan terhadap nilai pada jumlah data yang dikirimkan dibagi dengan waktu pengiriman data yang ditunjukkan pada aplikasi wireshark berupa nilai pada *packets* dibagi dengan *time span*. Satuan yang digunakan adalah Mb/s.

Sedangkan untuk memperoleh rata-rata *delay*, nilai yang digunakan adalah nilai yang ditunjukkan oleh *time since reference of first frame* pada detail paket. Nilai tersebut selanjutnya dihitung rata-ratanya sehingga didapatkan rata-rata *delay* dalam satuan detik(*second*).

4.2 Implementasi aplikasi *file sharing*

Langkah ini menjelaskan mengenai implementasi aplikasi *file sharing* yang akan diujicobakan dan dianalisis performansinya terhadap lingkungan pengujian sistem jaringan komputer yang telah diimplementasikan pada langkah sebelumnya. Aplikasi *file sharing* yang digunakan dalam penelitian ini yaitu myjxta dan Gnucleus.

4.2.1 Implementasi myjxta

Implementasi aplikasi myjxta dilakukan pada setiap komputer dan VM yang digunakan untuk pengujian *file sharing*. Implementasi dilakukan dengan melakukan installasi aplikasi terhadap komputer dan VM yang telah dipersiapkan. Prosedur installasi aplikasi yang sama dilakukan pada komputer dan VM yang menggunakan sistem operasi Windows dan Ubuntu.

Prosedur installasi aplikasi dimulai dengan melakukan unduhan terhadap *soure code* program yang tersedia pada halaman web https://sourceforge.net/projects/myjxta/?source=directory berupa *file* berformat .rar. Selanjutnya dilakukan ekstraksi terhadap *file* tersebut sehingga didapatkan *source code* dari aplikasi myjxta. Proses selanjutnya adalah melakukan *compile* terhadap *source code* tersebut pada sebuah *compiler* yaitu Netbeans IDE.

```
compile:
Copying 1 file to E:\New folder (5)\myjxta\build
Copy libraries to E:\New folder (5)\myjxta\dist\lib.
Building jar: E:\New folder (5)\myjxta\dist\myjxta.jar
To run this application from the command line without Ant, try:
C:\Program Files\Java\jdk1.6.0_17/bin/java -jar "E:\New folder (5)\myjxta\dist\myjxta.jar"
jar:
BUILD SUCCESSFUL (total time: 4 seconds)
```

Gambar 4.1 *Output Build Project*

Compile pada Netbeans IDE dilakukan dengan melakukan klik pada *Build Project* yang terdapat pada menu *Run*. *Compiler* selanjutnya akan meng-*compile source code* yang digunakan dan pada Gambar 4.1 diatas menunjukkan bahwa proses *compile* telah berhasil dilakukan dan aplikasi myjxta dapat dijalankan.

Implementasi aplikasi myjxta dilakukan untuk selanjutnya dilakukan pengujian aplikasi *file sharing* sesuai dengan perancangan yang telah dibuat sebelumnya sehingga dapat diperoleh nilai *throughput* dan *delay* pada masing-masing aplikasi *file sharing*. Kedua nilai tersebut selanjutnya dilakukan analisis dan perbandingan untuk mengetahui performansi dari setiap aplikasi *file sharing* yang diujikan.

4.2.2 Implementasi Gnucleus

Implementasi aplikasi Gnucleus dilakukan pada setiap komputer dan VM yang digunakan untuk pengujian *file sharing*. Implementasi dilakukan dengan melakukan installasi aplikasi sehingga dapat dijalankan. Komputer dan VM yang digunakan untuk melakukan pengujian menggunakan sistem operasi yang berbeda yaitu Windows 7 Ultimate 64bit dan Ubuntu LTS 14.04 sehingga prosedur dalam melakukan installasi berbeda.

Installasi pada sistem operasi Windows 7 Ultimate 64bit dimulai dengan melakukan pengunduhan terhadap *file* installasi aplikasi Gnucleus. Setelah *file* didapatkan selanjutnya dilakukan klik dua kali pada *file* Gnucleus.exe dan installasi akan berjalan. Setelah proses installasi aplikasi Gnucleus selesai dijalankan maka aplikasi Gnucleus siap untuk digunakan.

Prosedur installasi aplikasi Gnucleus pada sistem operasi Ubuntu LTS 14.04 berbeda dengan prosedur installasi pada sistem operasi Windows 7 Ultimate 64bit. Installasi dimulai dengan melakukan installasi aplikasi Wine agar *file* installasi Gnucleus dapat dijalankan pada sistem operasi Ubuntu LTS 14.04. Setelah aplikasi Wine terinstall selanjutnya *file* installasi Gnucleus dapat dijalankan.

```
1  $ sudo dpkg --add-architecture i386
2  $ sudo add-apt-repository ppa:wine/wine-builds
3  $ sudo apt-get update
4  $ sudo apt-get install --install-recommends winehq-devel
```

Kode 4.2 Perintah untuk melakukan installasi Wine

```
1   $ sudo wine Gnucleus_2.2.0.0_Setup.exe
```

Kode 4.3 Perintah untuk melakukan installasi Gnucleus melalui Wine

Installasi aplikasi Wine diperlukan untuk mengeksekusi *file* yang mempunyai *file system* yang berbeda dengan *file system* Ubuntu. Sehingga untuk menjalankan program ber-*file system* untuk sistem operasi Windows diperlukan aplikasi Wine dalam lingkungan sistem operasi Ubuntu.

Kode 4.1 merupakan prosedur yang diperlukan untuk melakukan installasi Wine pada Ubuntu yang dilakukan melalui *terminal*. Baris pertama pada kode tersebut berfungsi untuk mengaktifkan arsitektur 32 bit pada Ubuntu. Perintah selanjutnya berfungsi untuk menambahkan *repository* agar dapat dilakukan installasi aplikasi Wine dengan mengambil data yang ada pada *repository*. Selanjutnya pada baris ketiga berfungsi untuk memperbarui daftar paket dari *repository* yang terdaftar. Baris terakhir merupakan perintah untuk menjalankan proses installasi aplikasi Wine.

Sedangkan Kode 4.2 merupakan perintah yang dijalankan setelah proses installasi aplikasi Wine selesai. Perintah tersebut menjalankan aplikasi Wine untuk mengeksekusi *file* installasi aplikasi Gnucleus. Setelah proses installasi selesai selanjutnya aplikasi Gnucleus dapat dijalankan pada sistem operasi Ubuntu LTS 14.04.

Implementasi aplikasi Gnucleus dilakukan dengan tujuan agar dapat dilakukan pengujian aplikasi *file sharing* sesuai dengan perancangan yang telah dibuat sebelumnya sehingga dapat diperoleh nilai *throughput* dan *delay* pada masing-masing aplikasi *file sharing*. Kedua nilai tersebut selanjutnya dilakukan analisis dan perbandingan untuk mengetahui performansi dari setiap aplikasi *file sharing* yang diujikan.

4.3 Pengujian

Prosedur pengujian dilakukan untuk mengetahui performansi dari masing-masing aplikasi *file sharing* dalam distribusi konten data yang dilakukan oleh aplikasi *file sharing* sesuai dengan skema-skema yang telah dideskripsikan pada Bab 3. Pengujian aplikasi dilakukan sebanyak empat kali sesuai dengan jumlah *file* yang diujikan pada setiap skema dan dilakukan perhitungan rata-rata untuk mendapatkan nilai performansi yang akurat. Parameter yang digunakan untuk mengetahui performansi dari setiap aplikasi yaitu atau *throughput* dan rata-rata *delay* pada sisi *peer* yang berperan sebagai *receiver* dengan menggunakan bantuan aplikasi Wireshark. Prosedur pengujian terdiri dari pengujian aplikasi *file* sharing dan konfigurasi Wireshark. Sebelum dilakukan pengujian terhadap aplikasi, terlebih dahulu dilakukan persiapan *dummy file* yang diunduh dari http://download.thinkbroadband.com/ dengan ukuran 50MB, 100MB, 200MB, dan 512MB pada sisi *seeder*. *Dummy file* selanjutnya akan digunakan sebagai sampel ukuran konten data dan dimasukkan kedalam satu direktori yang selanjutnya didaftarkan sebagai direktori yang di *share* oleh aplikasi *file sharing*.

4.3.1 Pengujian Aplikasi *File Sharing*

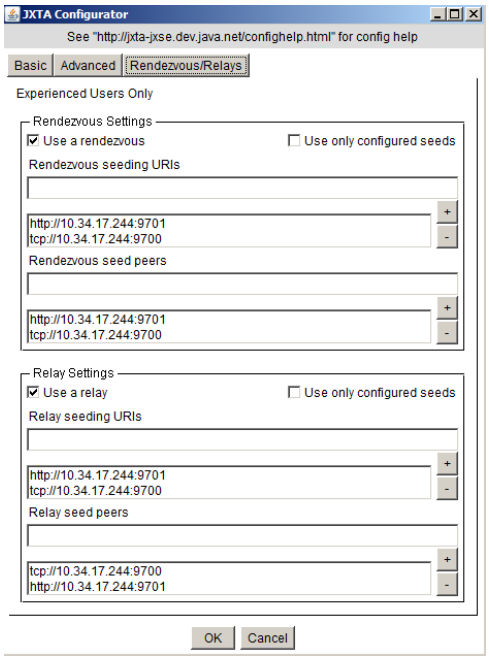
Pengujian masing-masing aplikasi *file sharing* dilakukan secara berulang-ulang sesuai dengan skenario yang telah dibuat sebelumnya. Pada langkah ini, akan dijelaskan mulai dari cara menjalankan aplikasi sampai dengan melakukan distribusi konten data pada lingkungan pengujian. Masing-masing aplikasi memiliki prosedur pengoperasian yang berbeda saat akan melakukan *file sharing*.

4.3.1.1 Pengujian Aplikasi myjxta

Pengujian aplikasi myjxta bertujuan untuk mengetahui performansi aplikasi saat melakukan distribusi konten data yang dihasilkan dari beberapa parameter yang telah ditentukan. Selanjutnya hasil pengujian akan dilakukan analisis dan perbandingan.

Tujuan dari pengujian aplikasi myjxta adalah untuk menjalankan skenario pengujian yang telah dibuat dan disesuaikan dengan perancangan sebelumnya sehingga dapat diperoleh nilai parameter yang telah ditentukan yaitu *trhoughput* dan *delay*. Nilai tersebut akan dijadikan parameter untuk dilakukan analisa dan perbandingan sehingga dapat diketahui faktor yang mempengaruhi perbedaan performansi yang dihasilkan oleh aplikasi *file sharing* yang diujikan.

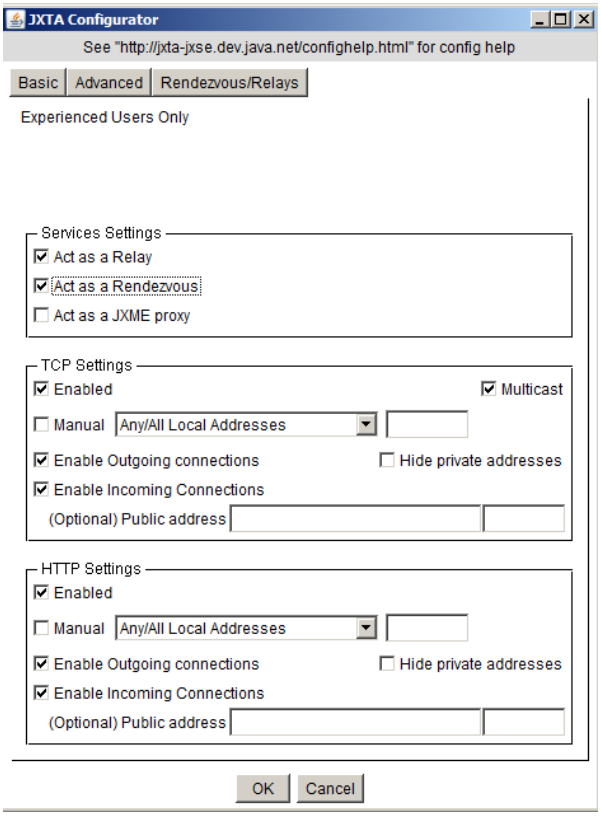
Pengujian aplikasi myjxta dilakukan dimulai dengan melakukan konfigurasi pada aplikasi sesuai dengan perancangan yang telah dibuat. Konfigurasi pada aplikasi myjxta terbagi menjadi 2 macam, yaitu sebagai *rdv/relay* dan sebagai *edge*.



Gambar 4.2 Konfigurasi *peer* yang bertindak sebagai *edge*

Gambar 4.2 menunjukkan konfigurasi *peer* pada aplikasi myjxta yang akan berperan sebagai *edge*. Konfigurasi dimulai dengan memilih *checkbox Use a*

rendezvous sehingga kolom pada konfigurasi dapat diisikan. Kolom konfigurasi tersebut diberikan input berupa alamat IP *peer* yang bertindak sebagai *rdv/relay* sehingga *edge* dapat terhubung dengan *rdv/relay*.



Gambar 4.3 Konfigurasi *peer* yang bertindak sebagai *rdv/relay*

Konfigurasi berikutnya dilakukan pada komputer yang akan berperan sebagai *rdv/relay*. Pada Gambar 4.3 ditunjukkan jendela konfigurasi pada *peer* yang akan berperan sebagai *rdv/relay*. Konfigurasi dimulai dengan memilih *checkbox Act as a Relay* dan *Act as a Rendezvous* untuk memberikan peran kepada *peer* sebagai *rdv/relay*. Selanjutnya dilakukan klik pada *checkbox Enabled* yang terdapat pada *TCP Settings* dan *HTTP Settings* yang berfungsi untuk mengaktifkan protokol TCP dan HTTP yang dibutuhkan oleh aplikasi. Pada *checkbox Enable Outgoing Connection* dan *Enable Incoming Connection* yang terdapat pada menu *TCP Settings* dan *HTTP Settings* juga dilakukan klik sehingga fungsinya dapat berjalan yaitu memperbolehkan adanya koneksi yang masuk dan keluar pada masing-masing protokol yang digunakan.

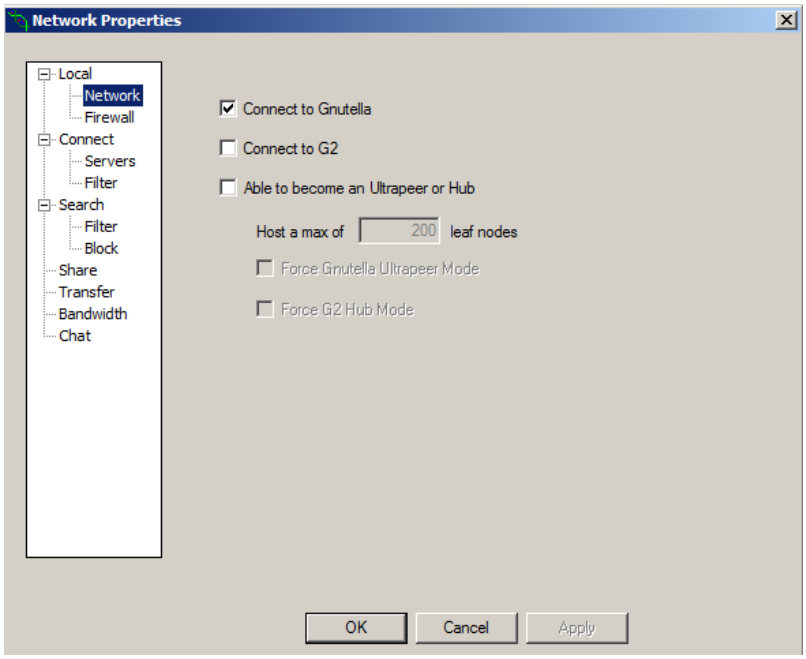
Setelah konfigurasi dilakukan dan setiap *peer* terhubung pada jaringan, dilakukan penentuan *file* yang akan diuji. Selanjutnya proses *file sharing* dapat dimulai antar *peer* sesuai dengan perancangan pengujian yang telah dibuat.

4.3.1.2 Pengujian aplikasi Gnucleus

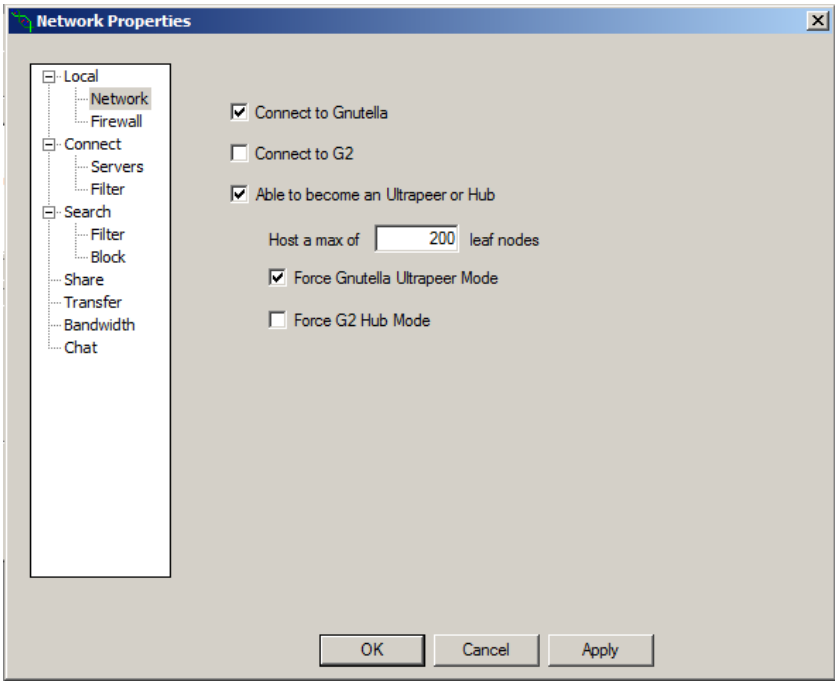
Pengujian aplikasi Gnucleus bertujuan untuk mengatahui performansi aplikasi dalam melakukan proses *file sharing* yang dihasilkan dari beberapa parameter yang telah ditentukan pada tahap perancangan. Setelah didapatkan hasil, akan dilakukan perbandingan.

Pengujian aplikasi Gnucleus adalah bertujuan untuk menjalankan skenario pengujian yang telah dibuat dan disesuaikan dengan perancangan sebelumnya sehingga dapat diperoleh nilai parameter yang telah ditentukan yaitu *trhoughput* dan *delay*. Nilai tersebut akan dijadikan parameter untuk dilakukan analisa dan perbandingan sehingga dapat diketahui faktor yang mempengaruhi perbedaan performansi yang dihasilkan oleh aplikasi *file sharing* yang diujikan.

Pengujian aplikasi Gnucleus dilakukan dimulai dengan melakukan konfigurasi pada aplikasi disetiap *peer* yang ditentukan dan disesuaikan dengan perancangan yang telah dibuat. Konfigurasi pada aplikasi Gnucleus dibedakan menjadi dua, yaitu sebagai *leafpeer* dan sebagai *ultrapeer.*



Gambar 4.4 Konfigurasi gnucleus sebagai *leafpeer*



Gambar 4.5 Konfigurasi gnucleus sebagai *ultrapeer*

Konfigurasi aplikasi Gnucleus dimulai dengan melakukan pemilihan terhadap *checkbox Connect to Gnutella* yang berfungsi agar aplikasi hanya terhubung pada jaringan Gnutella seperti yang ditunjukkan pada Gambar 4.4 dan 4.5. Selanjutnya adalah penentuan peran *peer* sebagai *leafpeer* atau *ultrapeer.* Prosedur yang dilakukan adalah memilih pada *checkbox Able to become an Ultrapeer or Hub* dan *Force Gnutella Ultrapeer Mode* sehingga *peer* dapat berperan sebagai *ultrapeer* dan *peer* akan berperan sebagai *leafpeer* jika *checkbox* tidak dipilih.

Gambar 4.6 *Leaf peer* terhubung dengan *Ultrapeer*

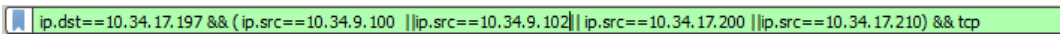
Setelah konfigurasi dilakukan, *peer* akan bergabung dengan jaringan yang terbentuk dan melakukan pencarian terhadap *peer* lain seperti yang ditunjukkan pada Gambar 4.6. Selanjutnya ditentukan *file* yang akan diuji dan dilakukan pencarian *file* oleh *peer* lain. Setelah *file* ditemukan, proses *file sharing* dapat dilakukan antar *peer* yang terhubung dan memiliki *file* yang sama.

4.3.2 Konfigurasi Wireshark

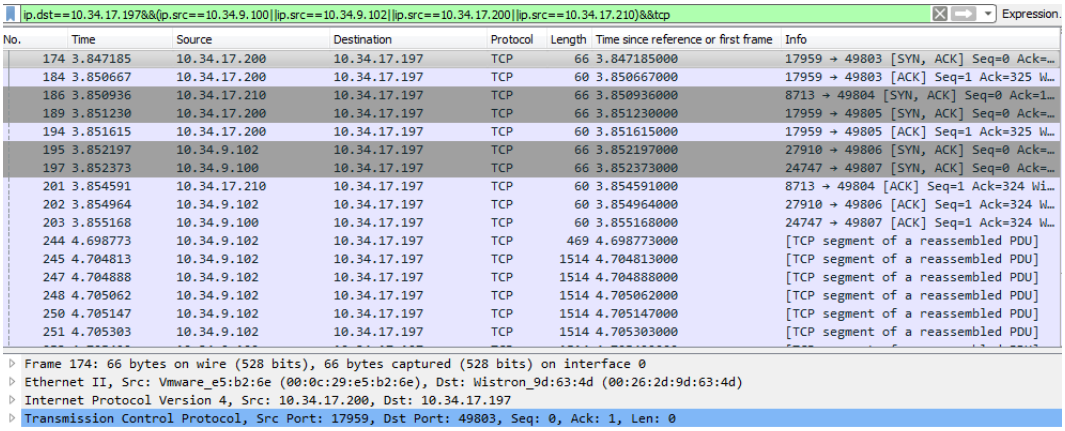
Wireshark digunakan untuk melakukan monitoring terhadap proses distribusi konten data yang berlangsung antara *seeder* dan *receiver*. Selain itu Wireshark juga digunakan untuk melakukan analisis terhadap performansi dari masing-masing aplikasi *file sharing* yang digunakan berdasarkan hasil monitoring yang dilakukan.

Wireshark dijalankan pada komputer *peer* yang berperan sebagai *Edge/leaf peer* pada sisi *receiver* atau penerima *file*. Hal itu dilakukan agar dapat menentukan *peer* mana saja yang melakukan pengiriman *file* saat *filter* dilakukan sehingga dapat terlihat bahwa *file* yang diterima oleh *receiver* adalah *file* yang dikirimkan oleh *seeder* mana saja.

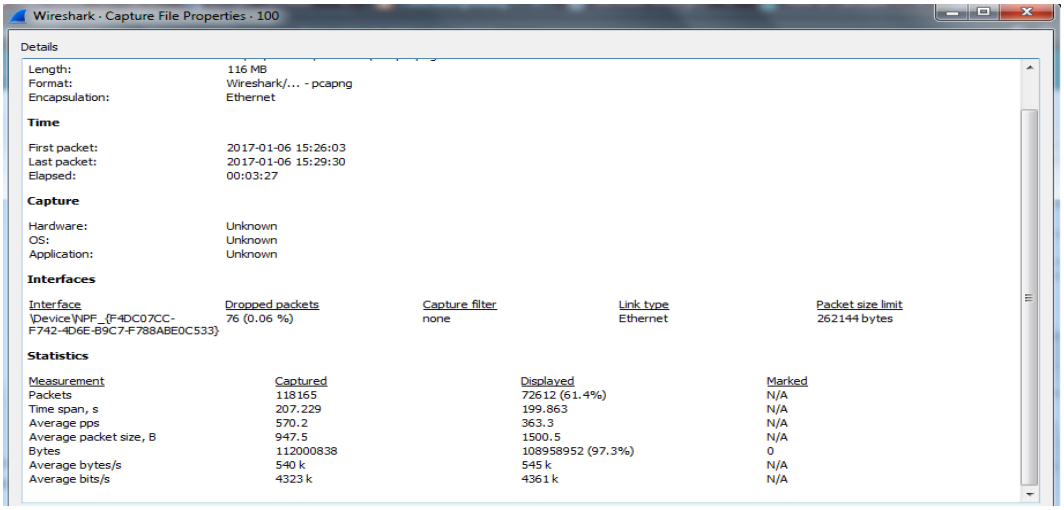
Konfigurasi wireshark dilakukan pada penelitian ini bertujuan untuk melakukan *capture* terhadap *traffic data* yang terjadi saat pengujian aplikasi *file sharing* berlangsung. Dari konfigurasi yang dilakukan dapat diperoleh nilai-nilai yang dibutuhkan dengan melakukan perhitungan untuk memperoleh nilai-nilai lain yang digunakan sebagai parameter. Nilai yang diperoleh dari konfigurasi wireshark yaitu jumlah paket dan *time span* untuk memperoleh nilai *throughput* dan nilai yang ditunjukkan oleh *time since reference of first frame* untuk memperoleh rata-rata *delay*.



Gambar 4.7 Contoh *filter* yang diterapkan pada wireshark



Gambar 4.8 Hasil *filter* yang telah diterapkan



Gambar 4.9 Statistik dari hasil *capture*

Untuk memperoleh hasil analisis yang akurat, diperlukan untuk melakukan *filter* pada aplikasi wireshark. *Filter* yang dimasukkan disesuaikan dengan alamat *ip* tujuan, alamat *ip* sumber dan protokol yang digunakan saat proses *file sharing* berlangsung. Setelah hasil ditampilkan sesuai dengan *filter* yang diterapkan selanjutnya dihitung dan disesuaikan dengan parameter yang telah ditentukan.

Setelah *filter* diterapkan, maka akan didapatkan hasil *traffic data* saat mekanisme *file sharing* berlangsung. Pada Gambar 4.8 ditunjukkan hasil saat *filter* diterapkan dan didalamnya terdapat sejumlah informasi yang terkait dengan *traffic* yang sedang berjalan, diantaranya yaitu :

- *Time*: Menampilkan waktu saat paket tertangkap
- *Source*: Menampilkan alamat IP sumber dari paket
- *Destination*: Menampilkan alamat IP tujuan dari paket
- *Protocol*: Menampilkan protokol yang digunakan paket
- *Length*: Menampilkan panjang paket
- *Time since reference or first frame*: Menampilkan jeda waktu pada saat paket dikirimkan
- *Info*: Menampilkan informasi dari paket

Pada Gambar 4.9 ditunjukkan statistik dari hasil *capture* terhadap *trafic* yang telah dilakukan *filter* yang berisi informasi – informasi pada saat mekanisme *file sharing* sedang berlangsung. Informasi yang diberikan yaitu jumlah paket, waktu yang dibutuhkan, rata – rata paket dalam satu detik, rata – rata ukuran paket, ukuran paket, dan kecepatan *transfer rate*.

Dari informasi – informasi yang ditampilkan oleh hasil *capture* dan telah dilakukan *filter*, nilai *throughput* dapat diperoleh dari nilai yang ditampilkan pada Gambar 4.9 yaitu pada statistik yang disediakan oleh Wireshark. Perhitungan nilai *throughput* dapat dilakukan dengan melakukan perhitungan nilai jumlah data yang dikirim dibagi dengan waktu pengiriman data. Nilai *througput* didapatkan dari nilai yang ditunjukkan pada statistik yaitu pada *Average bits/s* bernilai 4361 k yang berarti memiliki nilai *throughput* sebesar 4,361 Mb/s.Nilai itu didapatkan dengan melakukan perhitungan yaitu jumlah paket data dalam satuan bit dibagi dengan waktu yang dibutuhkan yaitu 871671616 bits/199.863 s. Sedangkan nilai rata – rata *delay* diperoleh dari perhitungan total delay dibagi dengan total paket yang diterima. Pada aplikasi Wireshark telah ditunjukkan dengan menampilkan dan akan didapatkan rata – rata dari nilai – nilai yang ditunjukkan pada kolom *Time since reference or first frame* dari paket pertama sampai paket terakhir yang ditangkap dan dilakukan *filter*.