IMPLEMENTASI DAN ANALISIS *QOS WIFI* MENGGUNAKAN *EMBEDDED SYSTEM*

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mecapai gelar Sarjana Komputer

Angie Pramudita Adhitama

0910683016

KEMENTRIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

IMPLEMENTASI DAN ANLISIS QOS
MENGGUNAKAN EMBEDDED SYSTEM

Disusun Oleh :

Angie Pramudita Adhitama
0910683016

Skripsi ini telah disetujui oleh dosen pembimbing
pada tanggal 17 Desember 2013

Dosen Pembimbing I	Dosen Pembimbing II
Eko Setiawan, ST.,M.Eng.	Aryo Pinandito, ST.,M.MT.
NIK 87061006110256	NIK. 83051916110374

LEMBAR PENGESAHAN

IMPLEMENTASI DAN ANLISIS QOS
MENGGUNAKAN EMBEDDED SYSTEM

Disusun oleh :

Angie Pramudita Adhitama
NIM. 0910683016

Skripsi ini diuji dan dinyatakan lulus pada
tanggal 2 Januari 2014

Penguji I

Wijaya Kurniawan, S.T., M.T.
NIP. 82012516110481

Penguji II

Barlian Henryranu P. S.T., M.T.
NIP. 82102406110254

Penguji III

Gembong Edhi Setyawan, S.T., M.T.
NIK. 76120116110373

Mengetahui,
Ketua Program Studi Informatika/Ilmu Komputer

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

PERNYATAAN ORISINILITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh pihak lain untuk mendapatkan karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebut dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (S-1) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan pasal 70).

Malang, 7 Oktober 2013

Angie Pramudita Adhitama
0910683016

KATA PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi Allah SWT karena atas rahmat dan hidayahNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "Implementasi da Analisis *QoS* Menggunakan *Embedded System*". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan – bantuan baik lahir maupun batin selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Ibunda Komsiati, Ayahanda Edy Susanto, Adik Reynold dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti – hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
4. Bapak Eko Setiawan, ST.,M.Eng. dan Bapak Aryo Pinandito, ST.,M.MT. selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
5. Bapak Arief Andy Soebroto, ST., M.Kom. selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.

6. Barlian Henryranu Prasetio, ST., MT., Sabriansyah Rizqika Akbar, ST., M.Eng. yang telah banyak membantu dan memberi penulis inspirasi dalam penulisan skripsi ini.
7. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
8. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini
9. Septya Dewi Cindrawati yang selalu memberikan semangat dan dukungan kepada penulis.
10. Nizamudin Ghonim, Hafid Rozaq , Wildan Ryan, Dhimas Sawung, Himawan Aditya, Riezky Ananda, Hoyi Ndadak Aji, Alfian Pratama, Hendra, Mas Ijal, Bagas Dhimas, Dhimas Rosiawan, Mikko, Evarda Syuman dan Iqbal selaku teman – teman di kontrakan bahagia tidak bebas banjir, yang selalu setia memberikan semangat dan kritikan selama penulisan skripsi ini.
11. Ryan Nanda, Zulfikar, Wibi, Magdalena, Suhe, Ichsan, dan teman-teman maupun kakak – kakak seperjuangan sebagai asisten jarkom, yang selalu setia memberikan semangat dan dukungan pada penulis.
12. Sahabat - sahabatku Angkatan 2009 Teknik Informatika, terimakasih atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya.
13. Semua sahabatku di SMA atas dukungan dan semangat yang selalu diberikan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya.
14. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga skripsi ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 16 Desember 2013

Penulis

ABSTRAK

Angie Pramudita Adhitama. 2013. : Implementasi dan Analisis QoS WiFi Menggunakan Embedded System
Dosen Pembimbing : Eko Setiawan, ST.,M.Eng dan Aryo Pinandito, ST.,M.MT.

Dewasa ini, pengukuran QoS jaringan nirkabel biasa dilakukan dengan menggunakan notebook PC atau telepon genggam. Pengukuran QoS jaringan komputer nirkabel yang memiliki lebih dari lima akses poin relatif sulit untuk dianalisis. Penggunaan notebook PC atau telepon genggam Android kurang efisien untuk melakukan pengukuran QoS jaringan tersebut karena koneksi jaringan nirkabel pada perangkat yang digunakan untuk melakukan analisis tersebut sering diputus dan disambungkan secara manual. Pada penelitian ini, Raspberry Pi sebagai salah satu sistem embedded mampu digunakan untuk mengukur QoS jaringan nirkabel secara sekuensial dengan menggunakan Shell Script. Nilai throughput dan latency jaringan nirkabel yang dianalisis dengan menggunakan Raspberry Pi menghasilkan standar deviasi latency sebesar 0,222 dan throughput sebesar 0,234.

Kata Kunci : *Embedded System, latency, throughput, QoS WiFi.*

ABSTRACT

Angie Pramudita Adhitama. 2013. : ***Implement and analyze QoS WiFi Using Embedded System*****.**

Advisor **: Eko Setiawan, ST.,M.Eng dan Aryo Pinandito, ST.,M.MT.**

Nowadays, wireless network QoS measurement is usually conducted using notebook PC or mobile phone. The measurement process of wireless network, which has more than five access points, is difficult to be analyzed. The usage of notebook PC or mobile phone has to be connected and disconnected manually to measurement QoS wireless network. Duringmeasurements process, Raspberry Pi as an embedded system is capable to measurement wireless network sequentially using Shell Script. Wireless network Throughput and latency values of wireless network are analyzed using the Raspberry Pi generates a standard deviation of 0.222 latency and throughput of 0.234.

Keywords : Embedded System, latency, throughput, QoS WiFi.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi yang berkembang saat ini memberikan dampak yang sangat signifikan di dalam berbagai aspek, terutama di bidang telekomunikasi. Berbagai layanan yang berbasiskan *Internet Protocol* (*IP*) juga ikut mengalami dampaknya dengan adanya standart – standart yang terus berkembang di dalam dunia *network* (Gunawan ,2008). Untuk mendapatkan layanan *IP* yang baik dibutuhkan analisis dan pengujian. *IP* merupakan bagian jaringan *network*, salah satu metode untuk menganalisis kualitas jaringan *network* adalah dengan melakukan analisis pada *Quality of Service* (*QoS)* Jaringan.

Analisis jaringan menggunakan *QoS* khususnya adalah *latency* dan *throughput* mampu memberikan analisis jaringan yang baik, aspek ini yang sering digunakan di dalam analisis jaringan. *QoS* didefinisikan sebagai sebuah mekanisme atau cara yang memungkinkan layanan dapat beroperasi sesuai dengan karakteristiknya masing masing dalam jaringan *IP* (Gani, 2010). Pemberian prioritas kepada aplikasi-aplikasi yang kritis pada jaringan merupakan hal yang penting (Djatikusuma, 2008). Penelitian ini diharapkan mampu memberikan solusi kepada penggunaanya agar memudahkan di dalam melakukan analisis suatu jaringan *Wireless Fidelity* (*WiFi)* yang memiliki *access point* sangat banyak.

Pengujian dan analisis yang ada saat ini masih menggunakan *personal computer (PC)* dan laptop. Pengujian aringan memiliki bermacam macam metode salah satunya pengujian dan analisis menggunakan Wireshark yang *diinstal* di dalam *PC* (Rosnelly, 2011). Metode ini masih memiliki kendala di dalam ruang lingkup kerja serta memiliki keterbatasan mobilitas penggunaan. Permasalahan tersebut dapat diselesaikan dengan menggunakan alat yang bersifat *portable*. Solusi yang ada saat ini adalah menggunakan *handphone* dengan sistem operasi Android. Namun *handphone* dengan sistem operasi Android bukan merupakan alat yang dikhusukan untuk melakukan pengujian jaringan, sehingga sumber daya, *RAM* dan *proccesor* tidak hanya digunakan untuk aplikasi pengujian jaringan, namun juga digunakan untuk aplikasi lainya. Berbeda dengan *embedded system*,

embedded system merupakan komputer yang dikhusukan untuk melakukan pekerjaan yang spesifik (Istiyanto, 2013).

Penulis memberikan solusi penggunaan alat yang dikhusukan untuk melakukan pengujian jaringan dengan menggunakan *embedded system* yang berupa Raspberry Pi dengan sistem operasi Raspbian Wheezy. Sistem yang dirancang akan memiliki kelebihan didalam mobilitas karena memiliki ukuran *device* yang kecil serta sumber daya minimal selain itu sistem memiliki kemampuan konektifitas yang bersifat sekuensial.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka dapat diambil perumusan masalah sebagai berikut :

1. Bagaimana merancang dan membuat suatu sistem *Portable* yang dapat menganalisis Jaringan komputer.
2. Bagaimana melakukan pengukuran *throughput* dan *latency* dari beberapa *access point WiFi* dalam satu area.
3. Bagaimana mengukur kualitas performansi sistem dalam analisis.

1.3 Batasan Masalah

Batasan masalah di dalam penelitian ini adalah:

1. Jaringan yang dianalisis berada pada area PTIIK Universitas Brawijaya.
2. Jaringan yang dianalisis adalah jaringan (*Wireless Fidelity*) *WiFi* yang tidak membutuhkan *password* untuk melakukan konektifitas pada *access point*nya.
3. Analisis dilakukan pada jaringan (*Wireless Fidelity*)*WiFi* yang memiliki (*Service Set Identifier*) *SSID* berbeda.
4. Konektifitas yang dilakukan bergantian pada *access point yang ada.*
5. Pengujian dilakukan pada jaringan (*Wireless Fidelity*)*WiFi* yang memiliki *Dynamic Host Control Protocol* (*DHCP) Server*.
6. Pengukuran *QoS* tiap *access point* dibatasi oleh waktu tertentu.
7. *QoS* yang diukur adalah *throughput* dan *latency*

8. Bahasa yang digunakan didalam pemrograman adalah bahasa Shell Script dan Python.

1.4 Tujuan Penelitian

Tujuan dilakukannya penelitian ini adalah membangun sistem pengukur *QoS* dengan menggunakan *embedded system* Raspberry Pi. Selanjutnya akan dimiliki alat *portable* yang mampu melakukan pengukuran *QoS* pada jaringan *WiFi* yang memiliki *access point* lebih dari satu, hal ini akan memberikan kemudahan saat melakukan *maintenance* atau menganalisis kualitas jaringan *WiFi*. Dengan penelitian diharapkan pengguna mampu melakukan analisis kualitas jaringan dengan data yang telah diperoleh dari sistem ini.

1.5 Manfaat

Manfaat bagi pengguna yang nantinya dapat diambil dari penelitian ini adalah menyediakan alat *portable* yang dapat digunakan untuk melakukan penghitungan *throughput* dan pengukuran *latency* jaringan *WiFi*. Dengan menggunakan sistem ini dapat memudahkan analisis jaringan *WiFi* di area tertentu.

1.6 Sistematika Penulisan

Sistematika penulisan ditujukan untuk memberikan gambaran dan uraian dari pembuatan laporan skripsi secara garis besar yang meliputi beberapa bab, sebagai berikut :

BAB I Pendahuluan

Bab ini berisi tentang latar belakang yang menjadi alasan pemilihan judul, perumusan masalah, pembatasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II Kajian Pustaka dan Dasar Teori

Bab ini menguraikan tentang konsep-konsep dan teori-teori yang berhubungan dengan permasalahan yang dirumuskan sebagai acuan dalam membahas permasalahan yang dihadapi di dalam menganalisis *QoS WiFi* dengan menggunakan *embedded system* Raspberry Pi

BAB III Metode Penelitian

Bab ini menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis serta pengambilan kesimpulan dan saran.

.

BAB IV Perancangan

Bab ini menguraikan tentang sistematika pembuatan alat pengukur QoS serta proses di dalam penelitian yang sudah dilaksanakan. Menguraikan permasalahan permasalahan yang ditemui di dalam penelitian.

BAB V Implementasi

Bab ini membahas implementasi, penggunaan dan perancangan sistem pengukur *QoS embedded system* Raspberry Pi yang telah dibuat.

BAB VI Pengujian dan Analisis

Bab ini memuat hasil pengujian dan analisis terhadap perangkat lunak yang telah direalisasikan dan diimplementasikan.

BAB VII Penutup

Bab ini memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Pada Bab II berisikan tinjauan pustaka membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai analisis dan *implementasi QoS jaringan WiFi dengan menggunakan embedded system*. Pada penelitian ini, dasar teori yang diperlukan berdasarkan latar belakang dan rumusan masalah adalah: konsep dasar *embedded system* ,konsep dasar Raspberry Pi,konsep dasar Raspbian Wheezy , konsep dasar *access point*, konsep dasar Analisis *QoS*, rumus rumus analisis *QoS* yang diterapkan pada bahasa pemrograman Python, penjelasan bahasa pemrograman Shell Script maupun Python, dan referensi referensi yang digunakan di dalam membuat *source code* program.

2.1 Kajian Pustaka

Pada penelitian yang sudah ada berjudul '*Comprehensive QoS analysis of enhanced distributed channel access in wireless local area networks'* (Woodward, 2012) menjelaskan tentang analisis *QoS*. Di dalam penelitian ini membahas rumus – rumus yang digunakan di dalam melakukan analisis *QoS*. Analisis *QoS* memiliki point – point parameter tertentu di dalam melakukan analisis tergantung aspek yang akan dianalisis. *QoS* disini dapat diuraikan di dalam beberapa point yakni *throughput*, *latency*, dan *Jitter*. Analisis pada ketiga aspek tersebut memiliki parameter parameter dan rumus yang berbeda. Pada penelitian ini diuraikan pembahasan penggunaan rumus dan langkah langkah penghitungan *QoS*.

latency jaringan menentukan kualitas jaringan, karena semakin besar *latency* jaringan akan berpengaruh pada kecepatan paket yang diterima oleh pengguna jaringan. Semakin besar *latency* jaringan akan mengakibatkan keterbatasan penggunaan jaringan, jaringan yang memiliki *latency* tinggi sangat tidak cocok untuk melakukan konektifitas langsung seperti *video call*, telepon *Internet*, dan *streaming TV online*.

Throuhput berpengaruh pada kecepatan *download* di dalam jaringan internet. Semakin besar *throughput* jaringan akan semakin cepat *download* yang dilakukan. Cara penghitungan *throughput* ini sebenarnya hanya tergantung dari besar ukuran *file* yang di *download* dan waktu untuk *download file* tersebut.

Pada penelitian tentang *embedded system* yang berjudul '*Embedded system Network Analyzer* Pada Jaringan *LAN*' menyebutkan bahwa dibutuhkan kondisi jaringan komputer yang baik agar mampu membuat komunikasi antar perangkat tersebut berjalan dengan optimal. Untuk mengukur kualitas jaringan komputer terdapat beberapa parameter yang dapat diukur antara lain *packet loss, troughput, latency* dan *jitter*. (Denny, 2013).

Pada penelitian ini menggunakan *Embedded system* Raspberry Pi dan fokus melakukan analisis hanya pada *WiFi*, berbeda dengan penelitian yang sebelumnya oleh Denny yang menggunakan Beagle Board dan fokus melakukan analisis pada jaringan (*Local Area Network*) *LAN*. Raspberry Pi digunakan karena memiliki ketersediaan port USB untuk *WiFi* dongle selain itu fokus penelitian *WiFi* dilakukan untuk memberikan solusi sesuai judul yang diambil.

Dalam penelitian ini, Raspberry Pi dibangun menjadi sistem yang mampu melakukan analisis *latency* dan *throughput* pada seluruh *WiFi* yang berada di area sistem.

.

2.2 *Embedded system*

Embedded system adalah *device* yang memiliki suatu fungsi khusus yang digunakan untuk membangun sistem yang lebih besar dan kompleks. *embedded system* mempunyai fungsi yang spesifik (Istiyanto, 2013). Sehingga *embedded system* memiliki ukuran yang lebih kecil dibandingkan dengan (*Personal Computer*)*PC* maupun *device* konvensional. Menurut Jazi Eko Istiyanto, Ph.D Salah satu *dosen Elektronika* dan *Instrumentasi* di *FMIPA* UGM menyatakan bahwa *embedded system* adalah *"A special purpose computer built into a larger device"*. Jika diartikan menjadi "Sebuah komputer yang khusus dibangun untuk sebuah perangkat yang lebih besar".

Embedded system mempunyai karakteristik sebagai sistem yang lebih besar dimana *embedded system* ini menjadi pendukung dari sistem yang besar,

bisa juga menjadi pengatur sistem – sistem kecil yang lain pada sistem yang besar. Selain itu *embedded system* mempunyai fungsi yang spesifik dimana perancangan *hardware* dan *software* dikhususkan untuk aplikasi yang spesifik sesuai dengan kebutuhan.

Salah satu *embedded system* yang mudah diimplementasikan dan dirancang adalah Raspberry Pi. Raspberry Pi adalah sebuah komputer berukuran kartu kredit yang dapat dihubungkan ke *TV* dan *keyboard*. Raspberry PI adalah *PC* kecil yang dapat digunakan untuk banyak hal selayaknya *PC desktop*, seperti *spreadsheet*, pengolah kata dan permainan. Raspberry Pi juga dapat digunakan untuk memainkan *video* dengan definisi tinggi (Raspberrypi, 2013).

Raspberry Pi memiliki dua model yakni model A dan B, model A memiliki 256MB *RAM*, 1 port USB dan tanpa Ethernet sedangkan pada tipe B memiliki 2 port USB dan memiliki Ethernet (Raspberrypi, 2013). Model B ditunjukan pada Gambar 2.1.

Gambar 2.1 Raspberry Pi
Sumber : (Raspberrypi, 2013)

Adapun spesifikasi Raspberry Pi type B dijelaskan pada Tabel 2.1.

Tabel 2.1 Spesifikasi Raspberry Pi type B

Spesifikasi Raspberry Type B	
Chip	Broadcom BCM2835 SoC full HD multimedia application processor
CPU	700 MHz Low Power ARM1176JZ-F Application Processor

GPU	Dual Core VideoCore IV® Multimedia Co-Processor
Memory	512MB SDRAM
Ethernet	Onboard 10/100 ethernet RJ45 jack
USB 2.0	Dual USB connector
Video Output	HDMI (rev 1.3 &1.4) Coposite RCA (PAL and NTSC)
Audio Output	3.5 Jack, HDMI
Media Penyimpanan	SD. MMC, SDIO Card slot
Sistem Operasi	Linux
Dimensi	8,6cm X 5,4cm X 1,7cm

Sumber: (Raspberrypi, 2013)

2.3 Raspbian Wheezy

Raspbian Wheezy adalah sistem operasi bebas berbasis Debian dioptimalkan untuk perangkat keras Raspberry Pi. Sebuah operasi sistem adalah seperangkat program dasar dan *utilitas* yang membuat Raspberry Pi Anda berjalan. Namun, Raspbian menyediakan lebih dari *OS* murni. Raspbian dibuat lebih dari 35.000 paket, *Software pre-compiled* dibundel dalam format yang bagus untuk memudahkan instalasi pada Raspberry Pi.

Raspbian dioptimalkan dan dikembangkan untuk kinerja terbaik pada Raspberry Pi, selesai pada Juni 2012. Namun, Raspbian Wheezy masih dalam pengembangan aktif dengan penekanan pada peningkatan stabilitas dan kinerja paket Debian sebanyak mungkin (Raspberrypi, 2013).

2.4 Wireless *Access Point*

Wireless acces point merupakan titik akses jaringan *WiFi* komputer. *Wireless Access point* merupakan perangkat keras yang memungkinkan perangkat *wireless* lain seperti laptop, ponsel untuk terhubung ke jaringan kabel menggunakan *WiFi*, *bluetooth* atau perangkat standar lainnya. *Wireless Access point* umumnya dihubungkan ke router melalui jaringan kabel saat ini kebanyakan telah terintegrasi dengan *router* dan dapat digunakan untuk saling mengirim data

antar perangkat *wireless* seperti laptop, *printer* yang memiliki *WiFi* dan perangkat kabel pada jaringan (Vicky, 2012).

Access point berfungsi sebagai pengatur lalu lintas data, sehingga memungkinkan banyak *Client* dapat saling terhubung melalui jaringan. Penelitian ini menggunakan *access point* sebagai bahan analisis.

2.5 *QoS (Quality of Service)*

QoS menunjukkan kemampuan sebuah jaringan untuk menyediakan layanan yang lebih baik lagi bagi layanan trafik yang melewatinya. *QoS* merupakan sebuah system arsitektur *end to end* dan bukan merupakan sebuah *feature* yang dimiliki oleh jaringan (Kurnia, 2004). *QoS* "*the collective effect of service performance which determines the degree of satisfaction of a user of the service*" International Telecommunication Union (Tiphon, 1999). Berdasarkan beberapa definisi diatas, dapat disimpulkan *QoS* adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan *bandwith*, mengatasi *jitter* dan *latency*. Parameter *QoS* adalah *latency, Jitter, packet loss, Throughput, MOS, echo cancellation* dan *PDD.*

Di dalam *QoS* memiliki beberapa parameter – parameter yang spesifik yang berfungsi untuk melakukan perhitungan atas kebutuhan analisis yang dibutuhkan, diantaranya yaitu :

2.5.1 *Throughput*

Throughput merupakan kecepatan transfer data efektif, yang diukur dalam bps dan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut (Huang, 2006).

$$\textit{Throughput} = \frac{\text{Paket data diterima}}{\text{Lama pengamatan}} \qquad (2\text{-}1)$$

2.5.2 *Packet Loss*

Packet Loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan. Menurut (Joesman, 2008) Nilai *packet loss* sesuai dengan versi *(Telecommunications and Internet Protocol Harmonization Over Networks)* dapat dilihat pada Tabel 2.2 sebagai berikut:

Table 2.2 Kategori Degradasi

Kategori Degadrasi	Packet Loss	Indeks
Sangat Bagus	0%	4
Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

Sumber : (Tiphon, 1999)

$$\textit{Packet Loss} = \frac{\text{(Paket data terkirim – Paket data diterima)}}{\text{Paket data yang dikirim}} \quad (2\text{-}2)$$

Persamaan 2-2 adalah rumus yang digunakan untuk mendapatkan nilai *Paket Loss*, hasil yang didapatkan merupakan persentase perbandingan data yang diterima dengan data yang tidak diterima.

2.5.3 *Delay (Latency)*

Delay adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama. Menurut versi *Cisco - Understanding Delay in Packet Voice Network* (Anonymous, 2008) besarnya *delay* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Kategori *Delay*

Kategori *Delay*	Besar *Delay*
Sangat Bagus	<150ms
Bagus dan masih bisa ditoleransi	150 s/d 400ms
Tidak bisa ditoleransi pada kebanyakan topologi jaringan	>450ms

Sumber : (Anonymous, 2008)

$$\text{Delay rata – rata} = \frac{\text{Total } delay}{\text{Total paket yang diterima}} \quad (2\text{-}3)$$

Delay rata – rata merupakan *delay* yang diperoleh dari akumulasi *delay* yang dibagi dengan banyaknya indeks *delay*.

2.5.4 *Jitter* atau Variasi Kedatangan Paket Hilang

Jitter atau variasi kedatangan paket hilang diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket di akhir perjalanan *Jitter*. *Jitter* lazimnya disebut variasi *delay* ,berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan. Terdapat empat kategori penurunan performansi jaringan berdasarkan nilai *peak Jitter* (Yanto, 2012) dijelaskan pada Tabel 2.5 nilai *Jitter* dapat dihitung dengan menggunakan persamaan 2-4. Jaringan yang baik adalah jaringan yang memiliki *Jitter* kurang dari 75ms.

Tabel 2.4 Kategori Degradasi *Jitter*

Kategori Degradasi	Peak *Jitter*	Indeks
Sangat Bagus	0ms	4
Bagus	0 s/d 75 ms	3
Sedang	75 s/d 125 ms	2
Jelek	125 s/d 225 ms	1

Sumber : (Tiphon, 1999)

$$\text{Jitter} = \frac{\text{Total variasi}}{\text{Total paket yang diterima}} \quad (2\text{-}4)$$

$$\text{Total variasi} = \text{Delay} - \text{Rata rata } delay \quad (2\text{-}5)$$

2.6 Referensi *Source Code*

Referensi *source code* digunakan peneliti sebagai bahan rujukan mealakukan edit pada program yang sudah ada, adapun beberapa sumber *Source*

code yang digunakan oleh peneliti adalah kode program untuk *latency*, *throughput*, dan konektifitas.

2.6.1 Python Ping.py

Python Ping.py di dalam sistem merupakan program utama di dalam melakukan penghitungan *throughput* dan pengukuran *latency*. Di dalam program ini terdapat algoritma untuk mendapatkan nilai *throughput* dan *latency*. Referensi dari coding ping dapat dilihat pada Gambar 2.2.

```
import os, sys, socket, struct, select, time
ICMP_ECHO_REQUEST = 8
def checksum(source_string):
    sum = 0
    countTo = (len(source_string)/2)*2
    count = 0
    while count<countTo:
        thisVal    =    ord(source_string[count    +    1])*256    +
ord(source_string[count])
        sum = sum + thisVal
        sum = sum & 0xffffffff
        count = count + 2
    if countTo<len(source_string):
        sum = sum + ord(source_string[len(source_string) - 1])
        sum = sum & 0xffffffff
    sum = (sum >> 16)  +  (sum & 0xffff)
    sum = sum + (sum >> 16)
    answer = ~sum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
     return answer
 # Menerima ping dari Socket
def receive_one_ping(my_socket, ID, timeout):
    timeLeft = timeout

    while True:
        startedSelect = time.time()
        whatReady = select.select([my_socket], [], [], timeLeft)
```

```
        howLongInSelect = (time.time() - startedSelect)
        if whatReady[0] == []: # Timeout
            return
        timeReceived = time.time()
        recPacket, addr = my_socket.recvfrom(1024)
        icmpHeader = recPacket[20:28]
        type, code, checksum, packetID, sequence = struct.unpack(
            "bbHHh", icmpHeader
        )
        if packetID == ID:
            bytesInDouble = struct.calcsize("d")
            timeSent = struct.unpack("d", recPacket[28:28 +
bytesInDouble])[0]
            return timeReceived - timeSent
         timeLeft = timeLeft - howLongInSelect
        if timeLeft <= 0:
            return
def verbose_ping(dest_addr, timeout = 2, count = 4):
    for i in xrange(count):
        print "ping %s..." % dest_addr,
        try:
            Delay  =  do_one(dest_addr, timeout)
        except socket.gaierror, e:
            print "failed. (socket error: '%s')" % e[1]
            break
        if Delay  ==  None:
            print "failed. (timeout within %ssec.)" % timeout
        else:
            Delay  =  Delay * 1000
            print "get ping in %0.4fms" % Delay
    print
if __name__ == '__main__':
    verbose_ping("www.ub.ac.id")
```

Gambar 2.2 Source code latency
sumber : (Notaras, 2013)

2.6.2 Shell Script Untuk Konektifitas Pada Jaringan *WiFi*

Shell Script konektifitas jaringan *WiFi* merupakan *source code* yang digunakan untuk melakukan konektifitas pada jaringan *WiFi* yang berada di dalam area sistem. Prinsip utama *source code* Shell Script ini ialah memasukan hasil search *WiFi* pada *array* dan melakukan konektifitas secara berulang pada nama (*Service Set Identifier*) *SSID* yang tersimpan di dalam *array*. Shell Script tersebut diambil dari referesi pastebin seperti yang dijelaskan pada Gambar 2.3.

```
#!/bin/bash
# Memilih konektifitas atau mebuat konektifitas baru
function connect (){
printf "Connecting using file $*\n"
# Capture incoming argument
SSID=$*
kill $(ps aux | grep -E '[w]pa_supplicant.*\'$INT'' | awk '{print
$2}') 2>/dev/null | xargs
dhclient $INT -r
ifconfig $INT down
# Melakukan konfigurasi interface WiFi
iwconfig $INT mode managed SSID "$SSID"
printf "Configured interface $INT; SSID is $SSID\n"
ifconfig $INT up
printf "Interface $INT is up\n"
wpa_supplicant   -B   -Dwext   -i$INT   -c/etc/wpa_supplicant.conf
2>/dev/null | xargs
printf "wpa_supplicant running, sleeping for 15...\n"
# Menunggu konektifitas sebelum mendapatkan ip
sleep 15
printf "Running dhclient\n"
dhclient $INT
# Menampilkan IP yang sudah didapat
ifconfig $INT | grep inet
exit
}
function clean_slate (){
```

```
}
clean_slate
read_saved

exit
```

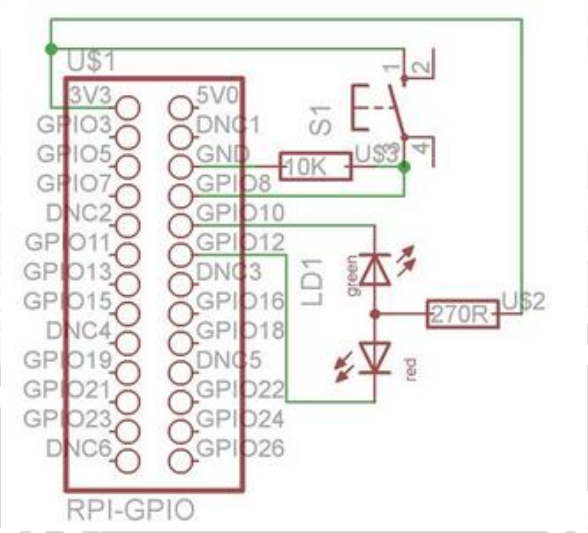
Gambar 2.3 *Source code* konektifitas Wi-fi
Sumber : (Pastebin, 2013)

2.7 GPIO (*General Purpose Input Output*) Raspberry

GPIO Raspberry Pi adalah pin *generic* pada *chip* yang dapat dikontrol (diprogram) melalui perangkat lunak baik di konfigurasi sebagai pin *input* maupun pin *output*. Raspberry Pi GPIO memiliki 26 pin dengan ukuran 2,54 mm. konektor GPIO memiliki fitur-fitur diantaranya:

a. Pin antarmuka I2C yang memungkinkan untuk menghubungkan modul *hardware* dengan hanya dua pin *control.*
b. SPI antarmuka, memiliki konsep mirip dengan I2C tetapi dengan standar yang berbeda.
c. Serial Rx dan Tx, pin untuk berkomunikasi dengan perangkat serial.
d. Pin PWM (*Pulse Width Modulation*) untuk *control* daya.
e. Pin PPM (*Pulse Position Modulation*) untuk mengendalikan motor servo.

Tegangan yang disediakan GND, 3.3V dan 5V, semua pin GPIO dapat digunaka baik sebagai digital *input* atau output. Pin yang berlabel SCL dan SDA dapat digunakan untuk I2C. Pin yang berlabel MOSI, MISO dan SCKL dapat digunakan untuk menghubungkan ke perangkat SPI kecepatan tinggi. Semua pin memiliki tingkat *log*ika 3.3V sehingga tingkat output 0-3.3V dan *input* tidak boleh lebih tinggi dari 3.3V (Hakim, 2013). GPIO dalam Penelitian ini digunakan untuk menjalankan program dengan *push botton*. Rangakaian dasar led dan *push botton* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Rangkaian GPIO
Sumber : (Mullins, 2012)

2.8 Standar Deviasi

Standar Deviasi digunakan didalam pengujian akurasi dari *embedded system*, dilakukan dengan membandingkan antara hasil pengukuran *embedded system* dengan PC dan Smartphone Android. (Lipi, 2010) Persamaan standar deviasi dapat dilihat pada uraian dibawah ini.

$$d_1 = x_1 - \bar{x} \tag{2-6}$$

d_1 = penyimpangan terhadap nilai rata – rata

x_1 = pembacaan data

$\bar{x}$ = rata - rata

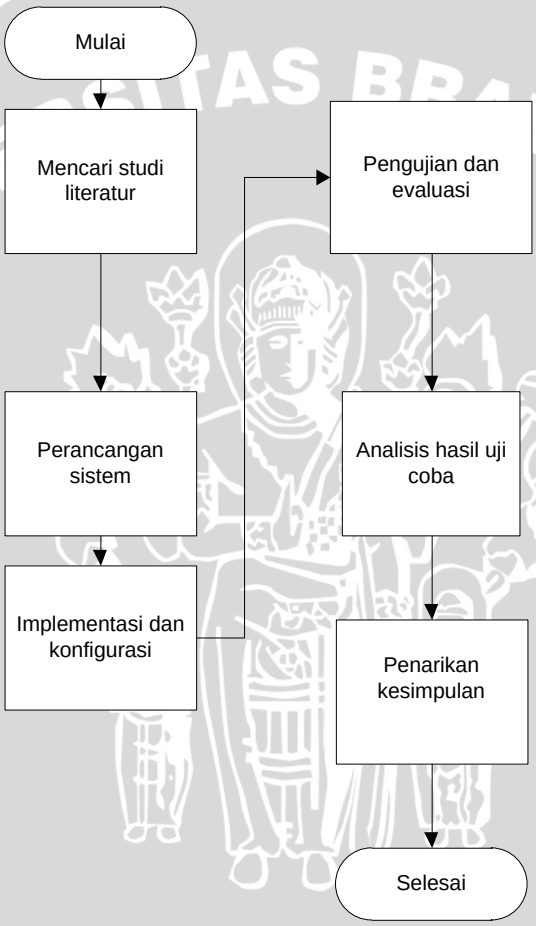
$$\sigma = \sqrt{\frac{d_1^2 + d_2^2 + d_3^2 + ... + d_n^2}{n}} = \sqrt{\frac{\sum d_t^2}{n}} \tag{2-7}$$

σ = standar deviasi

N = jumlah data total

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan tentang metode penelitian yang akan digunakan dalam tahapan-tahapan penelitian dan penyusunan skripsi. Metode Penelitian yang akan dilakukan pada skripsi ini secara umum dapat dilihat dari diagram alir pada Gambar 3.1 sebagai berikut:



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi Literatur digunakan untuk menambah studi pustaka dan pengetahuan yang dilakukan untuk mengerjakan penulisan laporan dan penelitian. Studi Literatur dilaksanakan dengan cara mengumpulkan teori dan pustaka yang berkaitan dengan tugas akhir ini meliputi :

1. *embedded system*
   a. Raspberry Pi
   b. Raspbian Wheezy
2. Bahasa pemrograman
   a. *Shell Script*
   b. Python
3. *Access point*
   a. *Signal Access point*
   b. Kualitas *Access point*
4. Jaringan komputer
   a. Analisis *QoS*
   b. *Throughput*
   c. *latency*

3.2 Perancangan Sistem

Perancangan arsitektur sistem adalah tahap mulai merancang suatu sistem yang mampu memenuhi semua kebutuhan fungsional aplikasi dalam tugas akhir ini. Perancangan sistem yang akan dibuat berdasarkan hasil study literatur tentang analisis *QoS* pada *embedded system*.

Perancangan sistem memberikan kemudahan dalam proses pembuatan sistem yang akan diimplementasikan untuk penelitian. Perancangan sistem meliputi analisis kebutuhan perangkat lunak dan perangkat keras. Perancangan sistem mempunyai tujuan sebagai bahan acuan dalam tahap implementasi sistem.

3.3 Implementasi dan Konfigurasi

Dalam tahap ini, dilakukan implementasi berdasarkan studi literatur dan perancangan sistem. Setelah spesifikasi sistem telah ditentukan maka implementasi yang pertama kali dilakukan adalah melakukan instalasi sistem operasi Raspbian Whezzy pada *embedded system* Raspberry Pi. Langkah selanjutnya adalah membuat *software* berbasis *Shell Script* yang mampu melakukan konektifitas secara otomatis dengan *WiFi* yang berada di area Raspberry Pi tersebut. Selain itu mengintegrasikan *Shell Script* agar dapat

melakukan *login* pada jaringan internal Universitas Brawijaya. Seperti yang sudah diketahui, agar dapat terhubung dengan jaringan internal UB diperlukan autentifikasi. Setelah permasalahan konektifitas selesai dilanjutkan dengan pembuatan program berbasis Python yang digunakan sebagai program inti untuk pengukuran *QoS*. Implementasi selanjutnya adalah mengubah konektifitas yang bersifat manual menjadi otomatis dan melakukan *looping* sejumlah *access point* yang ada. Setelah semua tahapan implementasi selesai maka akan didapat sebuah *embedded system* yang dapat mengukur *QoS* khususnya *throughput* dan *latency*.

3.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba *testing* konektifitas *timeout* dan penghitungan *QoS*. Pengujian konektifitas dimaksudkan agar memperoleh *embedded system* yang optimal dan mampu melakukan pengukuran pada semua *access point* yang ada. Pengujian ini fokus menganalisis masalah yang timbul di dalam proses konektifitas dengan *access point*.

Penelitian ini hanya melakukan pengukuran latency dan penghitungan throughput karena parameter ini merupakan parameter yang sering digunakan didalam menentukan kualitas jaringan internet. Latency akan berpengaruh kepada throughput jaringan sebab dengan latency kecil akan didapatkan throughput yang besar. Dengan uraian ini dapat disimpulkan apabila suatu jaringan memiliki latency yang kecil dan throughput yang besar maka jaringan tersebut merupakan jaringan yang memiliki kualitas yang baik. Kasus yang sering ditemui didalam jaringan internal UB adalah latency sangat berpengaruh kepada kesetabilan dan kualitas jaringan. Dengan througputh yang dihitung user dapat mengetahui kecepatan download jaringan.

Tahap pengujian selanjutnya adalah melakukan perbandingan hasil anaisa *QoS* dengan menggunakan Wireshark dan *software* analisis *QoS* yang dimiliki oleh *handphone* dengan sistem operasi Android. Di dalam analisis ini memfokuskan kepada presisi dari penghitungan yang didiapat dengan *embedded system*. Melakukan perbandingan hasil yang diperoleh dari ketiga metode tersebut. serta menguraikan permasalahan yang timbul di dalam melakukan analisis maupun pengujian ini.

Simulasi pengujian dilakukan dengan membandingan hasil penghitungan yang diperoleh *embedded system* dengan Wireshark maupun *software* analisis *QoS* yang dimiliki Android. Di dalam skenario pengujian terdapat kriteria pengujian, diantaranya:

a. Waktu yang diperlukan untuk melakukan konektifitas dengan *access point.*
b. Waktu yang diperlukan untuk melakukan analisis *QoS* pada satu *access point*.
c. Tingkat presisi yang didapat dari *embedded system* yang telah dibandingan dengan Wireshark maupun *handphone* Android.

3.5 Analisis Hasil Uji Coba

Pada tahap ini dilakukan analisis uji coba terhadap sistem yang telah dibuat. jika terjadi kekurangan dan kesalahan terhadap *embedded system* untuk mengukur *QoS* maka diperlukan analisis yang menyeluruh yang memungkinkan memberikan solusi dari kesalahan *embedded system*. Dalam tahap ini juga membuat suatu kesimpulan hasil uji coba yang sistematis yang dapat menjadi pedoman pengembangan selanjutnya.

3.6 Kesimpulan Saran

Kesimpulan didapatkan setelah melakukan perancangan, implementasi, pengujian dan analisis kepada sistem. Kesimpulan disusun berdasarkan hasil pengujian dan analisis sistem yang telah dibuat. Isi dari kesimpulan diharapkan dapat menjadi acuan pada penelitian lain untuk mengembangkan *embedded system analisis QoS.* di dalam penulisan akhir terdapat saran yang bertujuan untuk memberikan kemudahan kepada peneliti selanjutnya, apabila akan meneruskan penelitian ini.

BAB IV
PERANCANGAN

4.1 Perancangan Perangkat Keras

Perancangan sistem analisis *QoS embedded system* dilakukan setelah melakukan spesifikasi tentang kebutuhan sistem yang dibutuhkan. Penghitungan *QoS* jaringan *WiFi* dilakukan dengan menggunakan studi litelatur dan rumus untuk menghitung *trougputh* maupun *latency* jaringan *WiFi.*

Penelitian dilakukan pada jaringan *wireless LAN* yang terdiri dari *router wireless* yang memiliki *DHCP server* agar alat ini dapat memperoleh IP. Selain itu alat ini tidak membatasi jumlah dari *access point* yang akan diuji selama *access point* tersebut bersifat *open network*. *Embedded system* melakukan pengujian secara *looping* dan mengukur QoS *WiFi* yang bersifat *open* sehingga meminimalkan interaksi *user* dengan *embedded system* tersebut. Karena apabila mengukur QoS *WiFi* yang *secure* masih membutuhkan *username* maupun *password* di dalam autentifikasi awal. Dalam perancangan perangkat keras ini akan dibahas analisis kebutuhan perangkat keras dan topologi jaringan yang digunakan.

4.1.1 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras digunakan untuk menganalisis dan menyiapkan kebutuhan akan perangkat keras yang digunakan untuk membangun sistem pengukur *QoS* dengan menggunakan *embedded system*. Di dalam tahapan ini peneliti akan menganalisis kebutuhan – kebutuhan yang diperlukan di dalam membangun sistem pengukur *QoS*. Hal ini dilakukan untuk mempermudah di dalam pembuatan *embedded system* pengukur *QoS.* Setelah melakukan pemilihan *embedded system* pada tahap sebelumnya telah ditentukan oleh peneliti menggunakan Raspberry Pi. Adapun kebutuhan Perangkat keras yang dibutuhkan di dalam penelitian, dapat dilihat pada tabel 4.1 berikut:

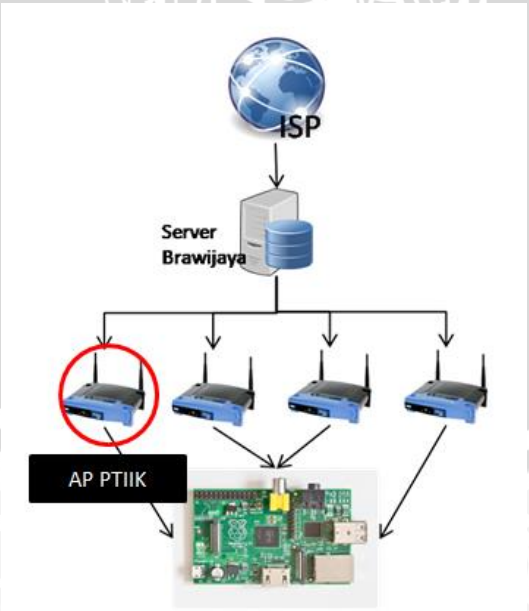
Tabel 4.1 Kebutuhan Perangkat Keras

Embedded system	Alat pengujian 1	Alat pengujian 2	Peralatan Jaringan
Raspberry Pi Tipe B *Processor* ARM	Processor *intel Core i3* CPU	Lenovo A3000	Kabel UTP Cat 5
512 MB SDRAM	2GB RAM	1GB RAM	Konektor RJ – 45
> 4GB SD Card	> 256 GB hardisk drive	> 2GB	Nano *WiFi* Adapter

Penggunaan Raspberry Pi didasarkan atas kebutuhan akan *memory* dimana Raspberry Pi type B ini memiliki *memory RAM* yang cukup besar yakni 512 MB, memori ini cukup untuk digunakan di dalam melakukan pengukuran *QoS*. Selain masalah *memory* penggunaan Raspberry Pi ini karena kebutuhan akan *Ethernet* maupun *port* USB yang digunakan untuk *WiFi Dongle* maupun kabel *LAN* sebagai penghubung dengan jaringan.

4.1.2 Topologi Jaringan

Topologi jaringan merupakan gambaran umum struktur jaringan yang dianlisis. Topologi jaringan yang akan dianalisis dapat dilihat pada Gambar 4.1.



Gambar 4.1 Topologi Jaringan

Keterangan Gambar :

a. ISP : Merupakan penyedia layanan Internet.

b. *Server* Brawijaya : *Server* yang dimiliki jaringan internal brawijaya.

c. AP : *Access point* yang akan dianalisis

d. Raspberry Pi : Alat untuk melakukan pengukuran *latency* dan penghitungan *throughput access point.*

Pada Gambar 4.1 topologi dapat dilihat Raspberry Pi melakukan konektifitas kepada semua *access point* dalam jangkauan sistem yang bersifat *open.* Raspberry Pi ini bertugas melakukan pengukuran *latency dan* penghitungan *throughput* pada *access point* tersebut. Tanda merah menunjukan *access point* yang pengukuran dalam waktu tertentu, pengukuran yang dilakukan dalam *access point* selanjutnya akan berganti pada *access point* yang lainnya di dalam jangkauan sistem.

4.2 Perancangan Perangkat lunak

Program yang dibuat untuk melakukan penghitungan dan pengukuran *QoS* ini merupakan program yang berbasis Shell Script dan Python. Kedua program ini merupakan pemrograman yang berjalan di Raspbian Wheezy, sebab program ini merupakan pemrograman yang sudah ada di dalam Raspbian Wheezy sehingga lebih mudah di dalam pengembangan program. Didalam perancangan perangkat lunak akan dijelasakan analisis kebutuhan perangkat lunak dan diagram alir program.

4.2.1 AnalisisKebutuhan Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses implementasi sistem analisis *QoS* dengan *embedded system* dijelaskan pada Tabel 4.2.

Tabel 4.2 Spesifikasi perangkat

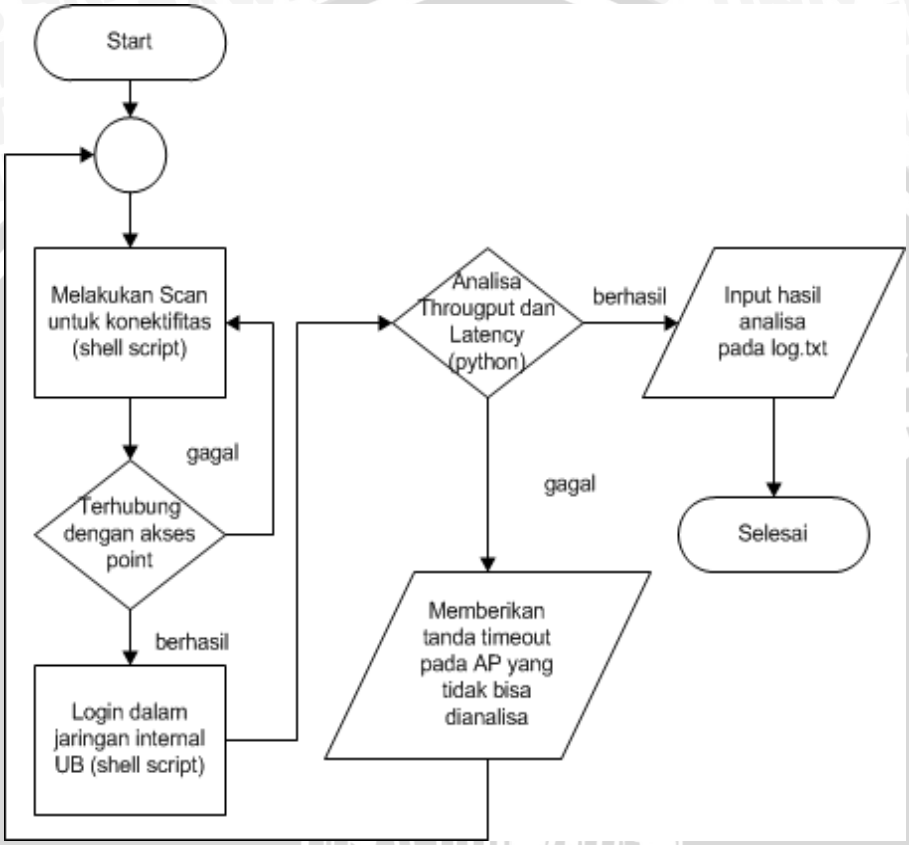
Raspberry Pi	
Operating System	Linux Raspbian Wheezy
Programming Language	Shell Script, Python
Asus A43S	
Operating System	Microsoft Windows 8 64-bit
Programing Language	Putty, WinScp, Python, connectify
Lenovo A3000	
Operating System	Android jellybean 4.2
Programing Language	*WiFi* analyzer,Voiptester,Net ping, ping- test

Tabel 4.2 merupakan perangkat lunak yang digunakan di dalam membangun sistem. Penjelasan Tabel 4.2 dapat dilihat pada uraian dibawah ini:

a. Linux Raspbian Wheezy merupakan operasi sistem Raspberry Pi
b. Shell Script dan Python merupakan bahasa pemrograman yang digunakan untuk membangun sistem pengukur.
c. Microsoft Windows 8 64-bit adalah operasi sistem PC
d. Putty digunakan sebagai SSH untuk melakukan *remote* pada Raspberry Pi saat konfigurasi
e. WinScp adalah *software* yang digunakan untuk mentransfer *file* melalui jaringan untuk Raspberry Pi, agar memudahkan pertukaran file.
f. Connectify adalah *software* yang digunakan pada operasi sistem Windows agar PC dapat memberikan IP pada *device* Raspberry Pi.
g. Android Jellybean 4.2 merupakan operasi sistem *hanphone* Android.
h. *WiFi* analyzer digunakan sebagai pengukur kekuatan signal *WiFi* dengan menggunakan *handphone* Android.
i. Net ping adalah *software handphone* Android yang digunakan sebagai penghitung *latency*.

4.2.2 **Diagram alir program**

Pembuatan struktur program merupakan hal penting di dalam arsistektur perangkat lunak adapun *flowcart* dari program yang dibuat seperti pada Gambar 4.2.



Gambar 4.2 Diagram Alir Program

Gambar 4.2 merupakan Gambaran umum alir program sistem analisis *latency* dan *throughput*, penjelasan Gambar 4.2 diuraikan dibawah ini

a. Melakukan *scan* untuk konektifitas merupakan *source code* yang dibangun pada bahasa pemrograman Shell Script, fungsi dari *source code* ini adalah untuk melakukan *scan* selanjutnya menyimpan hasil scan *WiFi* pada *array* dan melakukan konektifitas pada *access point* yang berada di dalam *array*.

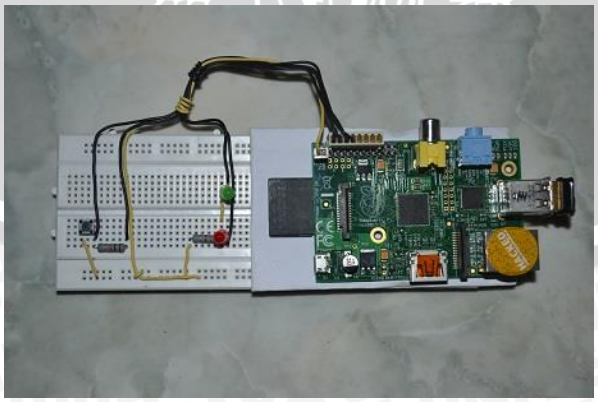
b. Pada seleksi kondisi yang berada di proses kedua eksekusi program akan dilanjutkan apabila sistem mendapat *IP*.
c. Setelah sistem mendapatkan *IP* proses selanjutnya adalah melakukan *login* (*Network Authentication System) NAS* Brawijaya.
d. penghitungan *throughput* dan pengukuran *latency* merupakan analisis yang dilakukan oleh sistem. Di dalam analisis ini memiliki seleksi kondisi, apabila analisis berhasil maka akan dilanjutkan ke proses selanjutnya apabila tidak sistem akan memberikan tanda pada *accsess point* yang dianalisis.
e. *Input* hasil pengukuran dan penghitungan pada *logWiFi*.txt adalah proses akhir dari sistem, di dalam proses ini hasil analisis dicatat.

BAB V
IMPLEMENTASI

Pada bab implementasi dibahas mengenai langkah-langkah dalam implementasi dan konfigurasi analisis *QoS* dengan menggunakan *embedded system*. Langkah – langkah dalam implementasi mengacu pada perancangan yang telah dijelaskan pada bab sebelumnya. Implementasi meliputi instalasi dan konfigurasi *embedded system,* implementasi Shell Script, implementasi Python, implementasi GPIO dalam Raspberry.

5.1 Instalasi dan Konfigurasi *Embedded System*

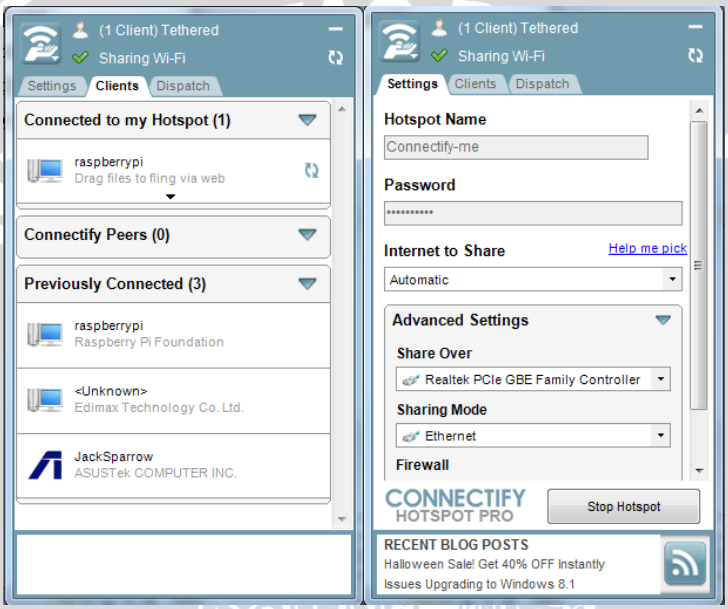
Perangkat keras yang digunakan di dalam membuat alat pengukur *QoS* ini adalah Raspberry Pi. Operasi sistem yang digunakan adalah Raspbian Wheezy , penggunaan operasi sistem ini karena Raspbian Wheezy merupakan operasi sistem yang dikembangkan khusus untuk Raspberry Pi. Instalasi Raspbian Wheezy merupakan langkah pertama untuk memperoleh sistem *embedded system,* dengan instalasi ini akan didapat *embedded system* yang siap dikonfigurasi menjadi *embedded system* pengukur *QoS.* Didalam instalasi dan konfigurasi *embedded system* menjelaskan diantaranya : Penggunaan Connectify, Penggunaan SSH (Putty), penggunaan WinScp, implementasi Shell Script, implementasi Python, dan implementasi (GPIO) dalam Raspberry PI.



Gambar 5.1 Instalasi Raspberry Pi.

5.1.1 Penggunaan Connectify

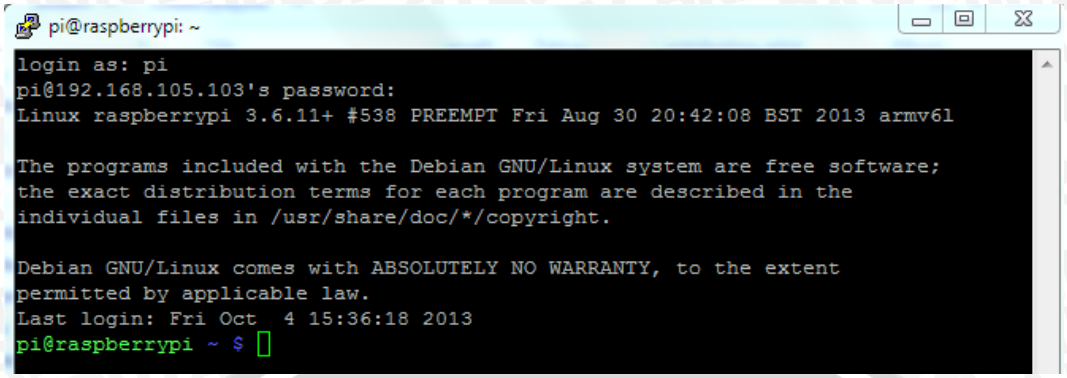
Penggunaan connectify karena Raspberry Pi membutuhkan *access point*, konfigurasi langsung dari *network option* di windows sering tidak *support* di dalam membuat *server* yang akan digunakan Raspberry Pi. Penggunaan connectify ini membantu di dalam menghubungkan Raspberry Pi dengan *server* melalui Ethernet. Dengan terhubungnya *server* dengan *client* akan dapat melakukan SSH.



Gambar 5.2 Connectify

5.1.2 Penggunaan SSH (Putty)

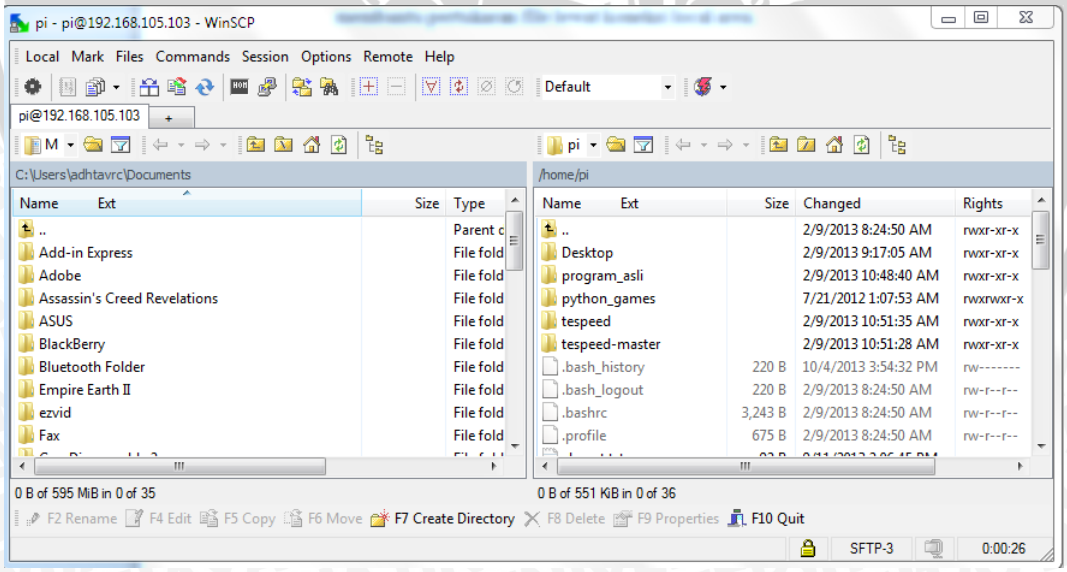
Di dalam melakukan konfigurasi Raspberry Pi maupun membuat program Shell Script yang berada di dalam *embedded system* dibutuhkan bantuan *software* untuk melakukan SSH, *software* tersebut adalah Putty. *Software* ini merupakan *software* yang berjalan di operasi sistem Windows, fungsinya adalah sebagai perantara antara Windows dengan Raspberry Pi. *Interface* Putty dapat dilihat pada Gambar 5.3.



Gambar 5.3 Putty

5.1.3 Penggunaan WinSCP

Penggunaan WinSCP dilakukan untuk memudahkan di dalam memindahkan *file* maupun memanajemen *file* yang berada di dalam *storage* Raspberry Pi. Jumlah *port* yang dimiliki Raspberry Pi hanya dua, *port* tersebut digunakan untuk keyboard dan *WiFi* dongle sehingga tidak ada lagi *port* yang tersisa. Penelitian ini membutuhkan manajemen *file*. yang berada di dalam Raspberry Pi sehingga digunakan WinSCP ini sebagai *software* yang dapat membantu pertukaran *file* lewat koneksi *local area*.



Gambar 5.4 WinSCP.

5.2 Implementasi Shell Script

Dalam membuat *embedded system* pengukur *QoS WiFi* ini dibutuhkan Shell Script yang mampu memanajemen konektifitas dan alur sistem. Beberapa masalah yang diselesaikan dengan menggunakan Shell Script diantarantanya konektifitas dengan *access point, login* jaringan Internal UB, dan pemanggilan seluruh program yang ada.

5.2.1 Shell Script *WiFi* Connect

Shell Script *WiFi* Connect digunakan untuk melakukan konektifitas pada *access point* yang ada. Sehingga tidak dibutuhkan lagi *interface* yang berbasis GUI. Berikut ini adalah beberapa potongan Shell Script *WiFi*.

```
function connect (){
printf "Connecting using file $*\n"

SSID=$*
kill $(ps aux | grep -E '[w]pa_supplicant.*\'$INT'' | awk
'{print $2}') 2>/dev/null | xargs

dhclient $INT -r
ifconfig $INT down

iwconfig $INT mode managed SSID "$SSID"
printf "Configured interface $INT; SSID is $SSID\n"
ifconfig $INT up

printf "Interface $INT is up\n"
wpa_supplicant -B -Dwext -i$INT -c/etc/wpa_supplicant.conf
2>/dev/null | xargs


printf "wpa_supplicant running, sleeping for 15...\n"
sleep 15
printf "Running dhclient\n"
dhclient $INT
```

```
        # Show current ip for interface
        ifconfig $INT | grep inet
        exit
}
```

Gambar 5.5 *Source code WiFi* connect

5.2.2 Shell Script *Login NAS* UB

Shell Script *Login* berfungsi untuk *login* pada jaringan internal UB, seperti yang sudah diketahui untuk dapat mengakses jaringan UB diperlukan autentifikasi pada nas.ub.ac.id. Program ini memudakan *login* tanpa harus memasuki aplikasi browser. Shell Script *login* dapat dilihat pada Gambar 5.6.

```
#!/bin/bash
cat /home/pi/login.txt | nc NAS.ub.ac.id 80;
```

Gambar 5.6 *Source code* Shell Script *login*

Shell Script diatas merupakan *source code* untuk menjalankan *code* txt yang berisi *account* maupun *password NAS*, uraian di dalam txt dapat dilihat pada Gambar 5.7.

```
POST /webAuth/ HTTP/1.1
Host: NAS.ub.ac.id
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:13.0)
Gecko/20120313 Firefox/13.0a1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://NAS.ub.ac.id/webAuth/
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
username=0910683016&password=aa11223344&pwd=aa11223344&secret=true
```

Gambar 5.7 *Login*.txt

5.2.3 Shell Script *Loop* konektifitas

Shell Script *loop* konektifitas berfungsi untuk melakukan konektifitas secara langsung pada sejumlah *WiFi* yang tersimpan di dalam *array search WiFi*. *source code* Shell Script *loop* dapat dilihat pada Gambar 5.8.

```
eval LIST=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
'/SSID/{print $2}') )
eval KEY=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
'/key/{print $2}') )
eval MAC=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
'/Address/{print $2 $3 $4 $5 $6 $7}') )

for (( num=0; num <= ${#LIST[@]}-1; num++ ))
do
 echo "Hasil :--------------\n"
if [ "${KEY[num]}" == "off" ]; then
 conf_create $num
 ./login.sh
 ./ping.py
 else
 echo "Access point ${LIST[num]} dengan MAC Address
${MAC[num]}" >> logWiFi.txt
 echo "Adalah Access SECURE, maka tidak dilakukan pengujian
lebih lanjut" >> logWiFi.txt
 echo "------------------------------------------------------------
--------" >> logWiFi.txt
fi
done
```

Gambar 5.8 *Source code Loop* Konektifitas

5.3 Implementasi Python

Python digunakan sebagai program inti di dalam sistem pengukur *QoS* ini. Program Python digunakan sebagai pengukur *latency* maupun *throughput* dari jaringan *WiFi*. Berikut *source code* Python untuk menghitung *latency* maupun *throughput* jaringan *WiFi*. *Source code throughput* dapat dilihat pada Gambar 5.9.

```
waktu= (time.time()-t0)
troughput=(437/waktu)

print output
print('\n 437 KB file terdownload dalam %s second ' %waktu)
print(' trougputh 437 KB file = %f KB/s \n' % troughput)

fd = open("logWiFi.txt",'a+')
fd.seek(0,2)
fd.write('\n%s' %output)
```

Gambar 5.9 *Source code Throughput*

5.4 Implementasi GPIO dalam Raspberry

Implementasi (*General Purpose Input Output*) GPIO dalam penelitian ini digunakan sebagai *push botton* yang berfungsi melakukan eksekusi program. Dengan menggunakan GPIO tidak lagi dibutuhkan keyboard sebagai *input* untuk melakukan eksekusi program. Dengan digunakan GPIO memungkinkan menambah nilai *portable* dari sistem pengukur *QoS* dengan menggunakan *embedded system* ini. *Source code* untuk GPIO dapat dilihat pada Gambar 5.10.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(8, GPIO.IN)
GPIO.output(10, True)
GPIO.output(12, True)
SwitchState = 0
LEDState = 0
while 1:
    if GPIO.input(8):
        if LEDState == 0:
            GPIO.output(10, False)
            GPIO.output(12, True)
```

```
            os.system("/home/pi/koneksinew.sh")
              LEDState = 1
            GPIO.output(10, True)
              GPIO.output(12, False)
          elif LEDState == 1:
            GPIO.output(10, True)
              GPIO.output(12, True)
              LEDState = 0
              time.sleep(1)
```

Gambar 5.10 *Source code* GPIO

5.5 *Log File* hasil pengukuran

Log file hasil pengukuran berisi hasil dari pengujian yang dilakukan oleh *embedded system*, di dalam *log file* terdapat informasi informasi yang dapat digunakan sebagai dasar analisis jaringan. Di dalam *log file* ini terdapat informasi besarnya *throughput*, *MAC address access point*, *SSID*, kekuatan *signal access point*, *noise level*, dan *link quality*.

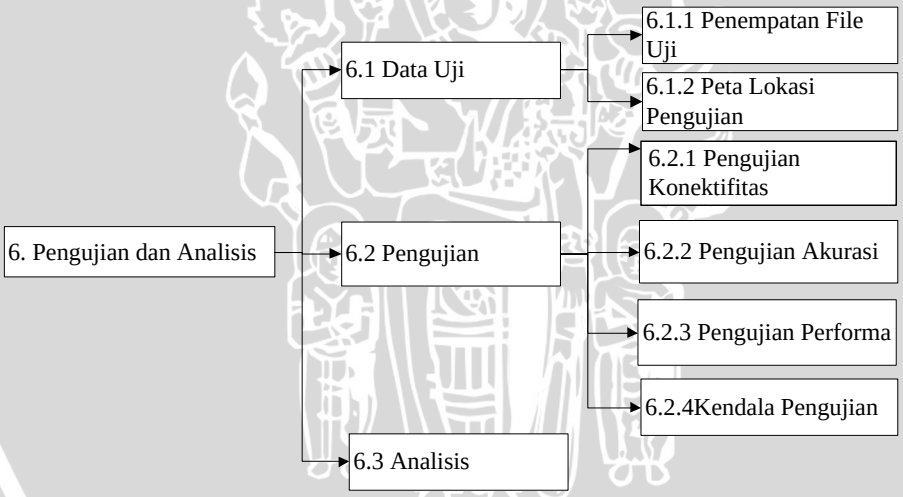
```
------------------------------------------------------------

wlan0     IEEE 802.11bgn  SSID:"PTIIK"  Nickname:"<WIFI@REALTEK>"
          Mode:Managed     Frequency:2.437  GHz     Access  point:
00:02:6F:87:53:10
          Bit Rate:72 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link   Quality=97/100     Signal   level=63/100     Noise
level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

```
ping 90.704918ms
ping 96.349001ms
ping 91.239929ms
ping 89.757204ms

133.2578125  KB file terdownload dalam 1.923503 second
trougputh 133.2578125 KB file = 69.278716 KB/s

------------------------------------------------------------
```

Gambar 5.11 *Log* hasil pengukuran

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap sistem analisis *latency dan throughput WiFi embedded* Raspbery Pi. Proses pengujian dilakukan melalui tiga tahapan yaitu pengujian konektifitas, akurasi, dan pengujian performa aplikasi. Pada pengujian akurasi digunakan teknik pengujian *Black Box (Black Box Testing)*. Pada pengujian performa dilakukan dengan menghitung waktu eksekusi analisis *access point* yang berada pada jangkauan sistem. Proses analisis dilakukan dengan menilai sejauh mana sistem membantu di dalam melakukan pengukuran QoS jaringan, yaitu membandingkan pengukuran QoS jaringan sebelum dan sesudah penggunaan sistem pengukuran *latency* dan penghitungan *throughput embedded* Raspberry Pi.



Gambar 6.1 Pohon Pengujian dan Analisis

6.1 Data Uji

Data uji merupakan uraian agar pembaca mengetahui informasi dasar pengujian. Dengan informasi dasar pengujian yang diketahui oleh pembaca, membantu didalam memahami prosedur pengujian. Data uji berisi lokasi pengujian dan penempatan *file* pengujian.

6.1.1 Penempatan File Uji

Di dalam melakukan penghitungan *throughput* dibutuhkan *file* yang dapat di *download* sebagai *file* uji. *File* yang akan digunakan sebagai bahan pengujian harus diketahui besar ukuranya, untuk memudahkan di dalam melakukan penghitungan *throughput*. Selain itu *file* yang digunakan sebagai bahan pengujian ditempatkan pada jaringan internal yang akan dianalisis, apabila *file* tersebut ditempatkan pada luar jaringan internal akan banyak faktor yang berpengaruh dalam hasil penghitungan. Detail *file* pengujian dapat dilihat pada uraian ini :

Lokasi *File* : blog.ub.ac.id

Link : http://blog.ub.ac.id /adhtavrc/files /2013/09/AngiePramudita-Adhitama_0910683016_kelas-F.docx

Ukuran *File* : 437 KB

6.1.2 Peta Lokasi Pengujian

Peta lokasi pengujian merupakan gambar titik lokasi yang digunakan sebagai tempat pengujian. Dengan menggunakan peta lokasi pengujian membuat pengujian sistem lebih sistematis, lokasi pengujian dapat dilihat pada Gambar 6.2.

Gambar 6.2 Peta lokasi pengujian
Sumber (PTIIK, 2013)

6.2 Pengujian

Proses pengujian dilakukan melalui tiga tahapan yaitu pengujian konektifitas, akurasi, dan pengujian performa sistem pengukuran *QoS WiFi embedded* Raspberry Pi pada *access point* di area Laboratorium Sistem Komputer dan Robotika.

6.2.1 Pengujian Konektifitas

Pengujian konektifitas digunakan untuk mengetahui apakah sistem yang dibangun sudah dapat melakukan koneksi pada jaringan *access point* yang berada di dalam area sistem tersebut. Pengujian ini berfokus pada penghitungan waktu yang dibutuhkan untuk melakukan hubungan pada jaringan *WiFi*, baik jaringan yang memiliki *traffic* lenggang maupun *traffic* padat. Selain melakukan penghitungan waktu koneksi, diperlukan juga pengujian untuk memastikan *embedded system* telah terhubung pada jaringan internal di PTIIK Universitas Brawijaya, sehingga diperlukan pengujian untuk memastikan *login* pada *NAS (Network Authentication System)* Brawijaya berhasil. Pada skripsi ini pengujian dilakukan pada jaringan *access point* yang berada di Laboratorium Sistem Komputer dan Robotika. Pada Tabel 6.1 dijelaskan kasus pengujian untuk *loop* konektifitas.

Tabel 6.1 Kasus uji untuk pengujian *loop* konektifitas

Nama Kasus Uji	Kasus pengujian *loop* konektifitas
Objek Uji	Shell Script *loop* system
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat melakukan hubungan konektifitas sejumlah *access point* yang terekam didalm *log array* scaning
Prosedur Uji	1. Penguji menjalankan sistem *embedded system* 2. Menjalankan Shell Script untuk melakukan *loop* konektifitas jaringan *WiFi* 3. Menghitung jumlah *WiFi* yang direkam di

	dalam *array* 4. Melakukan pembandingan jumlah *loop* dengan jumlah *array* yang terekam di dalam scaning *WiFi*
Hasil yang diharapkan	Sistem dapat melukan perulangan untuk melakukan konektifitas pada jaringan *WiFi* yang berada di area tersebut sesuai jumlah *access point* yang ada

Pengujian konektifitas juga mencatat waktu yang dibutuhkan sistem sampai mendapatkan *IP*. Waktu yang dibutuhkan sistem sampai mendapat *IP* bervariasi, dari hasil pengamatan diperoleh variasi rata – rata waktu yang dibutuhkan sistem untuk mendapatkan *IP* dapat dilihat pada Tabel 6.2.

Tabel 6.2 Rata – rata waktu untuk memperoleh IP

No	*SSID*	Tipe keamanan	Waktu rata rata eksekusi
1	PTIIK	*Open*	20 detik
2	(*SSID* Kosong)	*Open*	1 menit 17 detik
3	HMIF	*Secure*	23 detik
4	Kebidanan	*Open*	26 detik

Lama eksekusi oleh sistem untuk melakukan konektifitas tergantung dari kondisi jaringan, apabila jaringan padat maka eksekusi oleh sistem akan semakin lama. Dengan data yang diperoleh untuk waktu memperoleh *IP* dapat melakukan analisis apakah sistem yang dibuat sudah memenuhi performa yang diharapkan. Untuk melakukan pengujian pada area gedung PTIIK UB diperlukan *login NAS* agar sistem dapat terhubung dengan jaringan tersebut, sehingga *login NAS* pada jaringan UB harus dianalisis agar sistem mampu bekerja optimal. Kasus pengujian *login NAS* pada jaringan UB dapat dilihat pada Tabel 6.3.

Tabel 6.3Kasus pengujian *Login* jaringan internal UB

Nama Kasus Uji	Kasus Uji *Login* jaringan internal UB
Objek Uji	Shell Script *Log*in
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat terhubung dengan jaringan internal PTIIK Universitas Brawijaya.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system* 2. Menjalankan Shell Script untuk *login* jaringan internal PTIIK UB. 3. Melakukan ping pada server Universitas Brawijaya memastikan *embedded system* telah terhubung dengan jaringan internal
Hasil yang diharapkan	Sistem dapat melukan *login* pada *NAS*.ub.ac.id sehingga sistem memiliki akses untuk melakukan konektifitas dengan internet.

Hasil Pengujian konektifitas merupakan hasil yang diperoleh dari pengujian yang dilakukan sesuai prosedur yang sudah ditulis pada Tabel 6.3. Hasil dari pengujian dituliskan dalam Tabel 6.4 dibawah ini.

Tabel 6.4 Hasil pengujian konektifitas

No.	**Nama Kasus Uji**	**Hasil yang Diharapkan**	**Hasil yang Didapatkan**
1	Hasil Pengujian *loop* Konektifias	Sistem dapat melakukan *loop* untuk konektifitas sejumlah list *WiFi* yang terekam di dalam *log scaning*	Sistem mampu melakukan konektifitas sejumlah list *WiFi* yang tersimpan di dalam *log*

2	Kasus pengujian *login* jaringan internal Universitas Brawijaya	Sistem dapat melakukan *login* pada jaringan internal Universitas Brawijaya	Sistem dapat terhubung pada jaringan internal Universitas Brawijaya khusunya PTIIK

Proses analisis terhadap hasil pengujian konektifitas dilakukan dengan melihat waktu yang dibutuhkan oleh sistem untuk melakukan eksekusi *access point*. Analisis pengujian konektifitas juga menganalisis waktu rata – rata yang dibutuhkan perulangan untuk mengukur QoS semua *access point* yang berada pada jangkauan *embedded* Raspberry Pi. Pengujian konektifitas juga menguji sistem dapat melakukan *login* pada jaringan internal PTIIK UB. Berdasarkan hal tersebut maka dapat diambil kesimpulan sebagai berikut :

a. Sistem dapat melakukan konektifitas pada semua jaringan *WiFi* yang berada di dalam area sistem
b. Sistem memiliki waktu relatif singkat yaitu 20 - 77 detik saat terhubung pada jaringan internal di PTIIK Universitas Brawijaya.

6.2.2 Pengujian Akurasi

Pengujian akurasi dilakukan untuk mengukur akurasi *latency* dan akurasi *throughput*. Pengujian akurasi digunakan untuk mengetahui apakah sistem yang dibangun sudah menghasilkan hasil pengukuran sesuai dengan aplikasi pengukuran yang sudah ada. Pembandingan hasil pengukuran akan dibandingkan dengan pengujian *QoS WiFi* dengan menggunakan aplikasi Wireshark maupun dengan menggunakan aplikasi pengukuran jaringan yang dimiliki oleh *smartphone* Android. Parameter – parameter yang diukur digunakan sebagai acuan di dalam melakukan pengujian akurasi ini. Pengujian akurasi ini menggunakan metode pembandingan, karena hasil pengukuran menggunakan *embedded* Raspberry Pi ini akan dibandingkan dengan hasil pengukuran yang dihasilkan aplikasi Wireshark dan aplikasi pengukuran jaringan *WiFi* pada *smartphone* Android. Pada skripsi ini dilakukan penghitungan akurasi sistem analisis *QoS WiFi embedded* Raspberry Pi pada gedung PTIIK UB.

Kasus pengujian akurasi *latency* merupakan prosedur yang dilakukan di dalam melakukan pengujian hasil akurasi *latency* yang dihasilkan sistem. Prosedur pengujian akurasi *latency* dapat dilihat dalam Tabel 6.5 dibawah ini.

Tabel 6.5 Kasus uji untuk pengujian Akurasi *latency*

Nama Kasus Uji	Kasus Uji akurasi *latency*
Objek Uji	*Latency*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan akurasi dalam melakukan melakukan pengujian *latency*.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system* 2. Penguji melakukan pengukuran *latency* dengan *embedded system* dengan menjalankan program ping.py 3. Penguji menjalankan pengukuran *latency* pada Wireshark maupun *smartphone* Android 4. Membandingkan hasil pengujian *latency* dengan menggunakan *embedded system* dengan Wireshark maupun aplikasi analisis jaringan pada *smartphone* Android.
Hasil yang diharapkan	pengukuran *latency* yang dihasilkan tidak memiliki perbedaan yang sangat jauh dengan aplikasi Wireshark maupun aplikasi Android sehingga bisa dipastikan bahwa pengujian dengan *embedded system Raspberry Pi* ini valid.

Dari pengujian yang dilakukan sesuai prosedur pengujian akurasi *latency* diperoleh data rata – rata pengujian selama tiga hari pada tanggal 22 november, 7 desember, dan 8 desember yang dapat dilihat pada Tabel 6.6 dibawah ini :

Tabel 6.6 Hasil Pengujian *latency*

Waktu Pengujian	*Raspberry Pi*	Asus A43s	Lenovo A3000
08.00	27.73333333 ms	26.4 ms	26.23333333 ms
09.00	32.79 ms	31.73333333 ms	32.80666667 ms
11.00	37.29166667 ms	35.96666667 ms	36.55 ms
12.00	39.16666667 ms	38.82333333 ms	37.25833333 ms
14.00	31.089 ms	32 ms	32.25833333 ms
Rata – Rata	33.61413333 ms	32.*98466667* ms	33.02133333 ms

Standar deviasi Asus A43s dengan Raspberry.

$$\sigma = \sqrt{\frac{\sum d_t^2}{n}}$$

$$\sigma = \sqrt{\frac{0{,}314733^2}{2}} = 0{,}222$$

Standar deviasi Asus A43s dengan Lenovo A300.

$$\sigma = \sqrt{\frac{\sum d_t^2}{n}}$$

$$\sigma = \sqrt{\frac{0{,}036667^2}{2}} = 0{,}025$$

Pengujian akurasi latency menghasilkan nilai standar deviasi Raspberry dengan Asus A43s sebesar 0,222 dan standar deviasi latency Asus A43s dengan Lenovo A3000 sebesar 0,025.

Kasus pengujian akurasi *throughput* merupakan prosedur yang dilakukan untuk pengujian agar mendapatkan data sebagai hasil pengujian akurasi *throughput*. Adapun prosedur pengujiannya dapat dilihat dalam Tabel 6.7.

Tabel 6.7 Kasus uji untuk pengujian Akurasi *Throughput*

Nama Kasus Uji	Kasus Uji akurasi *throughput*
Objek Uji	*Throughput*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan akurasi dalam melakukan melakukan pengujian *throughputh*.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system*. 2. Penguji melakukan penghitungan *throughput*h dengan *embedded system* dengan menjalankan program ping.py. 3. Penguji menjalankan penghitung *throughput*h pada Wireshark maupun *smartphone* Android. 4. Membandingkan hasil pengujian *throughput* dengan menggunakan *embedded system* dengan Wireshark maupun aplikasi analisis jaringan pada *smartphone* Android.
Hasil yang diharapkan	Pengukuran *throughput* yang dihasilkan tidak memiliki perbedaan yang sangat jauh dengan Wireshark maupun aplikasi pada Android, sehingga bisa dipastikan bahwa pengujian dengan *embedded system Raspberry Pi* ini valid.

Pengujian yang dilakukan pada area gedung Laboratorium Sistem Komputer dan Robotika PTIIK UB dengan *device* Asus A43s didapatkan hasil 561.34 KB/s. hasil ini diperoleh dari persamaan 2-1 dimana *file* yang di *download* memiliki besar 437 KB dan waktu yang dibutuhkan untuk *download file* tersebut adalah 0.779314054 *second.* Sedangkan pengujian yang dilakukan dengan Lenovo A3000 didapatkan hasil 560.7495436 KB/s. hasil 560.7495436 KB/s merupakan data yang diperoleh dari hasil pengujian, mengacu pada **Lampiran 10**.

Tabel 6.8 adalah hasil pengujian *throughput* yang dilakukan selam 3 hari. Tabel 6.8 berisi rata – rata pengujian yang dilakukan pada tanggal 22 November, 7 Desember, dan 8 Desember.

Tabel 6.8 Pengujian Akurasi *throughput*

Waktu Uji	*Raspberry Pi*	Asus A43s	Lenovo A3000
08.00	561.5547787 KB/s	567.3324 KB/s	563.9847243 KB/s
09.00	560.8781843 KB/s	562.523602 KB/s	559.3390687 KB/s
11.00	458.3507477 KB/s	463.0777556 KB/s	513.9634333 KB/s
12.00	483.7997143 KB/s	483.2328974 KB/s	461.6931979 KB/s
14.00	400.086092 KB/s	385.1817019 KB/s	366.7568849 KB/s
Rata – Rata	492.9339034 KB/s	492.2696714 KB/s	493.1474618 KB/s

Standar deviasi Asus A43s dengan Raspberry

$$\sigma = \sqrt{\frac{\sum d_t^2}{n}}$$

$$\sigma = \sqrt{\frac{0{,}332116^2}{2}} = 0{,}234$$

Standar deviasi Asus A43s dengan Lenovo A3000

$$\sigma = \sqrt{\frac{\sum d_t^2}{n}}$$

$$\sigma = \sqrt{\frac{0{,}87779^2}{2}} = 0{,}620$$

Dari Tabel 6.8 didapatkan standar deviasi akurasi throughput Asus A43s dengan Raspberry maupun Asus A43s dengan Lenovo A3000 . Standar deviasi akurasi dihitung dengan Persamaan 2-7. dengan menggunakan persamaan 2-7 diperoleh nilai standar deviasi akurasi pengukuran antara Asus A43s dengan Raspberry sebesar 0,234 sedangkan untuk asus A43s dengan Lenovo A3000 sebesar 0,620. Hasil Pengujian akurasi merupakan hasil yang didapatkan dari

prosedur pengujian pada kasus pengujian akurasi yang sudah ditulis pada bab sebelumnya. Hasil pengujian akurasi dapat dilihat pada Tabel 6.9 dibawah ini.

Tabel 6.9 Hasil pengujian akurasi

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Kasus pengujian akurasi *latency*	Pengukuran *latency* yang dihasilkan tidak memiliki perbedaan yang sangat jauh dengan wireshark maupun aplikasi Android	*Latency* yang didapatkan memiliki selisih perbedaan yang sedikit dimana Raspberry Pi memiliki rata – rata *latency* yang lebih besar dibanding Asus A43s dan Lenovo A3000
2	Kasus pengujian akurasi *throughput*	Pengukuran *throughput* yang dihasilkan tidak memiliki perbedaan yang sangat jauh dengan Asus A43s maupun Lenovo A3000s	Waktu yang diperlukan untuk menguji *throughput* yang didapat dari hasil *download* tidak memiliki perbedaan yang jauh

Dari pengujian akurasi yang dilakukan dengan membandingkan hasil penghitungan antara *embedded system* dengan PC, menghasilkan standar deviasi 0,222. Analisis hasil dapat dilihat pada uraian dibawah ini :

a. Hasil *latency* yang didapatkan tergantung dari waktu melakukan akses jaringan maupun banyaknya *user* yang melakukan konektifitas pada *access point* yang diukur.
b. *Throughput* yang dihasilkan sistem *embedded* dengan asus A43s memiliki standar deviasi sebesar 0,234.
c. *Latency* yang dihasilkan sistem *embedded* dengan Asus A43s memiliki standar deviasi 0,222.

6.2.3 Pengujian Performa

Pengujian performa digunakan untuk mengetahui apakah sistem yang dibangun dapat melakukan pengukuran QoS *WiFi* dengan cepat. Di dalam pengujian performa dilakukan penghitungan waktu eksekusi, pengukuran QoS pada *access point* yang berada pada jangkauan *embedded system* Raspberry Pi dan melakukan pengujian pada sistem untuk mengetahui sistem mampu melakukan seleksi kondisi pada tipe keamanan jaringan *WiFi*. Kasus pengujian performa waktu eksekusi dapat dilihat pada Tabel 6.10.

Tabel 6.10 Kasus uji untuk pengujian performa waktu eksekusi

Nama Kasus Uji	Kasus Uji waktu konektifitas dan eksekusi
Objek Uji	Waktu konektifitas dan eksekusi
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui waktu yang diperlukan sistem melakukan konektifitas pada jaringan pada kondisi – kondisi yang berbeda.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system* 2. Menjalankan aplikasi analisis dan melakukan penghitungan waktu manual secara bersamaan. 3. Mencatat waktu yang dibutuhkan setiap melakukan satu konektifitas pad *access point*
Hasil yang diharapkan	Dapat mengetahui waktu yang dibutuhkan untuk melakukan konektifitas pada *access point* berbagai kondisi.

Pengujian performa waktu eksekusi menghasilkan data lama waktu yang dibutuhkan sistem untuk melakukan pengukuran *latency* maupun penghitungan *throughput* jaringan *WiFi*. Dalam tabel 6.11 dijelaskan waktu yang diperlukan untuk melakukan pengukuran QoS jaringan *WiFi*. Pada pengukuran ini pengambilan data dilakukan di dalam area Laboratorium Sistem Komputer dan Robotika.

.

Tabel 6.11 Hasil pengujian waktu eksekusi pengukuran.

Indeks *Loop*	***SSID Access point***	**Tipe *access point***	**Waktu eksekusi *access point***
1	PTIIK	*Open*	00:00:23.05
2	PTIIK	*Open*	00:00:19.72
5	PTIIK	*Open*	00:00:18.44
6	(*SSID* kosong)	*Open*	00:01:17.25
7	PTIIK	*Open*	00:00:17.97
8	Kebidanan	*Open*	00:00:33.40
	Rata - rata		00:00:30.80

Pengujian performa seleksi kondisi *WiFi* berfungsi untuk mengetahui apakah sistem sudah dapat membedakan antara *WiFi* yang bersifat *open* atau *WiFi* yang bersifat *secure*, karena di dalam skripsi ini hanya fokus pada jaringan yang tidak memiliki keamanan maka dalam pengujian ini diharapkan hanya *access point* atau *WiFi* yang bersifat *open* saja yang akan dieksekusi lebih lanjut. Prosedur pengujian performa seleksi kondisi dapat dilihat pada Tabel 6.12.

Tabel 6.12 Kasus pengujian performa seleksi kondisi *WiFi*

Nama Kasus Uji	Pengujian performa seleksi kondisi *WiFi*
Objek Uji	Seleksi kondisi *WiFi*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat melakukan seleksi pada tipe keamanan jaringan.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system*. 2. Penguji melakukan analisis pada *logWiFi*.txt
Hasil yang diharapkan	Analisis dalam *LogWiFi*.txt menghasilkan dua kesimpulan yaitu pemberian tanda untuk jaringan yang bersifat *secure* maupun *open*.

Di dalam beberapa studi kasus tentang *access point*, memungkinkan di dalam satu area tertentu terdapat dua atau lebih *access point* yang memiliki *SSID*

yang sama. Sehingga di dalam melakukan konektifitas pada *SSID* tersebut, *device* secara otomatis memilih *signal* yang terkuat pada *SSID* yang sama. Pengujian performa konektifitas *WiFi* dilakukan agar didapat kesimpulan mengenai sistem pada *embedded system* Raspberry Pi ini, mampu atau tidak di dalam melakukan konektifitas pada *SSID* yang sama namun tidak melakukan konektifitas pada *MAC address* yang sama. Kasus pengujian *loop* dapat dilihat pada Tabel 6.13.

Tabel 6.13 Kasus pengujian performa *loop* konektifitas *WiFi*

Nama Kasus Uji	Kasus pengujian performa *loop* konektifitas *WiFi*
Objek Uji	*Loop* konektifitas
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui apakah sistem mampu melakukan konektifitas pada *SSID* yang sama tanpa memperhatikan kekuatan signalnya.
Prosedur Uji	1. Penguji menjalankan sistem *embedded system*. 2. Menjalankan program pengukur *QoS WiFi* 3. Melakukan analisi pada program yang *running*
Hasil yang diharapkan	*SSID* yang sama namun memiliki *MAC address* berbeda akan terhubung dengan *embedded system*

Hasil pengujian performa yang telah dilakukan sesuai prosedur yang sudah ditulis di dalam kasus pengujian menghasilkan data yang dapat dilihat pada Tabel 6.14 dibawah ini.

Tabel 6.14 Hasil pengujian performa

No.	**Nama Kasus Uji**	**Hasil yang Diharapkan**	**Hasil yang Didapatkan**
1	Kasus Uji waktu konektifitas	Pengujian dilakukan untuk mengetahui waktu yang diperlukan sistem	Dapat mengetahui waktu yang dibutuhkan untuk melakukan konektifitas

		melakukan konektifitas pada jaringan pada kondisi – kondisi yang berbeda.	pada *access point* berbagai kondisi.
2	Pengujian performa seleksi kondisi *WiFi*	Sistem dapat membedakan jenis keamanan *access point* dan hanya melakukan eksekusi lebih lanjut pada jaringan yang *open*	Sistem mampu membedakan jenis keamanan jaringan dan hanya akan melakukan eksekusi pada jaringan yang memiliki tipe keamanan *open*
3	Kasus pengujian performa loop konektifitas WiFi	Sistem mampu melakukan konektifitas pada *SSID* yang sama tetapi memiliki *MAC address* yang berbeda.	Sistem tidak mampu melakuakan konektifitas pada *SSID* yang sama dengan *MAC address* berbeda.
4	Pengujian Performa waktu eksekusi	Sistem mampu melakukan konektifitas tidak lebih dari 3 menit setiap 1 *access point.*	Rata – rata dibutuhkan waktu 30 detik dalam melakukan konektifitas 1 *access point.*

Proses analisis terhadap Hasil pengujian performa dilakukan dengan menghitung waktu yang dibutuhkan sistem untuk melakukan eksekusi pengukuran *QoS WiFi*. Dengan analisis performa ini diperoleh sejumlah kesimpulan sistem yang telah dibuat. Kesimpulan dari hasil pengujian performa diantaranya:

a. Sistem dapat melakukan seleksi kondisi dimana hanya *access point* yang bersifat *open* saja yang akan dilakukan pengukuran dan penghitungan pada tahap berikutnya.
b. Waktu eksekusi pada *access point* yang tidak memiliki kejelasan baik masalah *SSID* maupun tipe keamanan lebih lama dibandingkan dengan *access point* yang memiliki tipe keamanan dan *SSID* yang jelas.

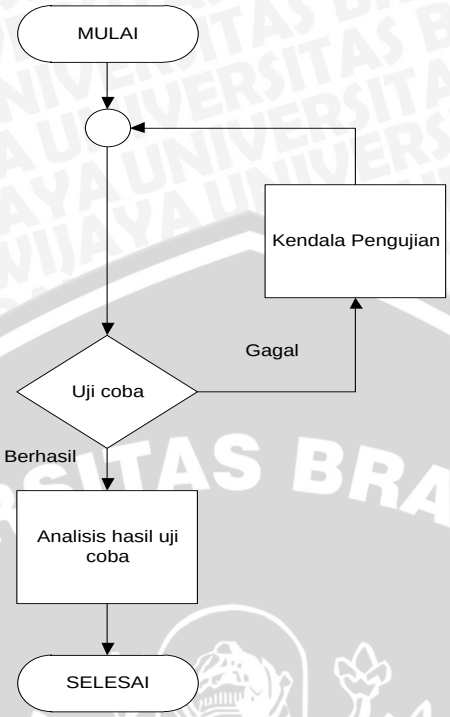
c. Dengan dihasilkannya beberapa data oleh sistem ini diantaranya kekuatan *signal*, *link quality* dan *nosise* dapat membuat pemetaan *access point* yang efisien di dalam melakukan perancangan jaringan *WiFi*.

6.2.4 Kendala Pengujian

Di dalam melakukan pengujian ditemui berbagai macam kendala pada sistem analisis*QoS WiFi* dengan *Embdded System*. kendala yang paling sering dijumpai ialah untuk melakukan konektifitas pada jaringan *WiFi* yang berada di area PTIIK UB, padatnya *traffic* jaringan dan banyaknya pengguna jaringan *WiFi* di area PTIIK menyebabkan sistem sulit mendapatkan *DHCP* atau sistem menjadi mati ketika akan mendapatkan *DHCP*. Selain sulitnya mendapat IP di dalam proses *DHCP*, terdapat kendala di dalam konektifitas yang disebabkan banyaknya *SSID* yang sama di area PTIIK UB dan *access point* yang tidak memiliki nama *SSID*.

6.3 Analisis

Analisis bertujuan untuk menganalisa data hasil pengujian hingga menghasilkan kesimpulan-kesimpulan. Analisis ini dilakukan berdasarkan hasil pengujian yang telah dilakukan terhadap sistem dan mengacu pada dasar teori. Proses analisis yang dilakukan antara lain adalah analisis konektifitas, analisis akurasi, analisis performa. Proses analisis dapat dilihat pada Gambar 6.3.



Gambar 6.3 Diagram analisis

Berdasarkan hasil pengujian konektifitas, implementasi perancangan sistem dapat dikatakan layak. *Access point* yang terdapat di area sistem dapat tercatat dalam *array*. Selain itu, sistem mampu melakukan seleksi kondisi pada jaringan yang bersifat *open* maupun *secure* dan hanya melakukan konektifitas pada jaringan yang bersifat *open.*

Mengacu pada hasil pengujian akurasi, sistem dapat dikatakan layak dan berhasil. Hasil pengukuran *latency* dan penghitungan *throughput* yang dilakukan oleh sistem memiliki standar deviasi kecil dengan Asus 43s throughput sebesar 0,234 dan latency sebesar 0,222. Dari kesimpulan ini implementasi sistem dikatakan layak.

Berdasarkan hasil pengujian performa, kualitas sistem dapat dikatakan cukup layak. Hal ini didukung dengan keterbatasan sistem yang tidak mampu melakukan konektifitas menyeluruh pada *access point* yang memiliki *SSID* sama tetapi *MAC address* berbeda. Dalam kondisi ini sistem akan melakukan *loop* pada *SSID* yang memiliki kekuatan signal terbesar, sehingga terdapat kendala pengujian apabila dilakukan pada area yang memiliki duplikasi nama *SSID*.

Secara keseluruhan, sistem dapat dikatakan layak. Sistem mampu melakukan pengujian tidak terbatas pada jumlah *access point* yang berada di dalam area sistem.

BAB VII
PENUTUP

7.1 Kesimpulan

Dari hasil penelitian, didapatkan hasil sebagai berikut.

1. Untuk membuat sistem analisis jaringan komputer *portable* diperlukan Raspberry Pi dan *WiFi* dongle Edimax tipe nano sebagai perangkat keras serta Shell Script serta Python sebagai perangkat lunak.
2. Pengukuran *throughput* dan *latency* dapat dilakukan dengan cara melakukan pengukuran secara bergantian pada seluruh *access point* yang berada dalam 1 area.
3. Kualitas performansi sistem dapat diukur dengan pengujian konektifitas,tingkat akurasi dan waktu eksekusi.
4. Dari hasil pengujian yang dilakukan didapatkan hasil konektifitas sistem mampu mendapat *IP* dan *login* pada jaringan UB. Untuk akurasi didapatkan standar deviasi akurasi latency Raspberry dengan asus A43s sebesar 0,222 dan standar deviasi *throughput* 0,234. Sedangkan untuk performa sistem memiliki kekurangan didalam melakukan konektifitas pada jaringan *WiFi* yang memiliki nama *SSID* yang sama pada satu area sistem.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem ini antara lain:

1. Untuk penelitian lebih lanjut disarankan sistem analisis *QoS* ini dapat dikembangkan dengan parameter yang lebih banyak diantaranya *packet loss, jitter,* dan *rate download.*
2. Untuk penelitian lebih lanjut disarankan penggunaan parameter *SSID* dikombinasikan dengan *Mac Address device,* agar tidak terjadi *loop* berulang pada *SSID* yang memiliki nama sama.
3. Untuk penelitian selanjutnya sistem dapat dikembangkan sebagai alat pemetaan penentuan posisi efektif untuk *access point,* Sebab sistem ini mampu malakukan pengukuran kekuatan *signal* dan *link quality WiFi*

DAFTAR PUSTAKA

Anonymous. 2008. " Understanding Delay in Packet Voice Network". Cisco.

Alexander, L., Song,Yeong-tae., Krishnan, Mahesh., Mathur, Vijita., Ahn, Jin., Shyamasundar,Vijay. 2005. "Throughput Measurement for UDP Traffic in an IEEE 802.11g WLAN".Towson : Towson University.

Djatikusuma, Edin S. 2008. "Analisis Weighted Fair Queueing *QoS* Pada Jaringan Padat". Palembang: STMIK MDP.

Fatoni, 2011. "Analisis Kualitas Layanan Jaringan Intranet (Studi Kasus : Universitas Bina Darma)". Palembang : Universitas Bina Darma.

Gani, Taufik Abdul., Rahmad., Afdhal. 2010. "Aplikasi Pengaruh Quality Of Service (Qos) Video Conference Pada Trafik H.323 Dengan Menggunakan Metode Differentiated Service (Diffserv)". Universitas Syiah Kuala.

Gunawan, Arif Hamdani. 2008. "Quality of Service dalam Data Komunikasi ,dunia telekomunikasi". diakses pada 6 oktober 2013, http://telecommunicationforall.blogspot.com/2008/05/quality-service.html

Hakim, Malik Abdillah Ibnul., Putra, Yeffry Handoko. 2013. "Pemanfaatan Mini Pc Raspberry Pi Sebagai Pengontrol Jarak Jauh Berbasis Web Pada Rumah". Bandung: Jurusan Teknik Komputer Unikom.

Herianto, Denny. 2013. "Embedded system Network Analyzer pada Jaringan Lan" Malang : PTIIK, Universitas Brawijaya.

Huang, Chih-Wei., Chindapol, A., Ritcey, J.A., Hwang, "Link Layer Packet Loss Classification for Link Adaptation in WLAN". 2006. 40th Annual Conference on Information Sciences and Systems.

Jonathan, Antony Pradana. 2011. "Network Traffic Management, Quality of Service (QoS), Congestion Control dan Frame Relay". , Jakarta: Tugas Akhir Universitas Gunadarma.

Lipi. 2010. "Jurnal Elektronika". Jakarta : Lembaga Ilmu Pengetahuan Indonesia.

Mullins, Robert. 2012. "Raspberry Pi ". University of Cambridge, diakses 9desember 2013 http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/two.html

Ningsih, Yuli Kurnia., Susila, Tjandra., Ismet, Rizky Febrian. 2004. "analisis quality of service (qos) pada simulasi jaringan multiprotocol label switching virtual private network (mpls vpn)". JETri.

Notaras, George. 2009. "Python Implementation of the ping command". gloaded journal, diakses pada 16 September 2013
http://www.g-loaded.eu/2009/10/30/Python-ping/

Pastebin. 2013. "Shell Script for connect WiFi connect.sh" pastebin.

PTIIK. 2013 "PTIIK Maps". Diakses pada 11 desember 2013
http://ptiik.ub.ac.id/apps/info/map

Raspbian . 2013. "Spesifikasi dan setting operasi sistem raspberry Pi"
diakses pada 23 juli 2013,
http://www. raspbian.org

Rosnelly, Rika., Pulungan, Reza. 2011. "Membandingkan AnalisisTrafic Data Pada Jaringan Komputer anatar Wireshark dan NMAP". FMIPA UGM.
Istiyanto, Jazi Eko. 2013. " Embedded systems : Hardware or Software?". FMIPA UGM.

Tiphon. 1999. "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) General aspects of Quality of Service (QoS)". TIPHON.

Woodwart, Mike E., Hu, Jia., Min, Geyong., Jia, Weijia. 2012. "Comprehensive *QoS* analysis of enhanced distributed channel access in wireless local area networks". SciVerse ScienceDirect.

Yanto. 2012. "Analisis *QoS* (Quality Of Service) Pada Jaringan Internet (Studi Kasus: Fakultas Teknik Universitas Tanjungpura)". Universitas Tanjungpura.

Lampiran 1 Source code GPIO

```
#!/usr/bin/python
# Import the required modules.

import RPi.GPIO as GPIO
import os
import time

# Set the numbering sequence of the pins, then set pins ten and
twelve to output, and pin eight to input.

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(8, GPIO.IN)

# Turn both of the LEDs off.
GPIO.output(10, True)
GPIO.output(12, True)

# The SwitchState variable is 1 if the button is pressed, and 0
otherwise. LEDState is 0 when off, 1 when red, and 2 when green.
SwitchState = 0
LEDState = 0

while 1:
    if GPIO.input(8):
# When the LED is off, keep the green LED off, turn the red one
on, then change the state of the LED to reflect that it is red,
and wait one second.

        if LEDState == 0:
            GPIO.output(10, False)
            GPIO.output(12, True)
          os.system("/home/pi/koneksinew.sh")
            LEDState = 1
```

```
37.             GPIO.output(10, True)
38.               GPIO.output(12, False)
39.               #time.sleep(1)
40.
41.   # When the LED is red, turn the green LED on, turn the red one
42.   off, then change the state of the LED to reflect that it is
43.   green, and wait one second.
44.           elif LEDState == 1:
45.             GPIO.output(10, True)
46.               GPIO.output(12, True)
47.               LEDState = 0
48.               time.sleep(1)
```

Lampiran 2 Source code konektifitas (koneksinew.sh)

```
#!/bin/bash

function conf_create (){

if [ -z "${LIST[0]}" ]; then
printf "No available wifi connection using wlan0\n"
exit
fi

# Memilih dari hasil SCAN

a=$*

ITEM="${LIST[a]}"
KUN="${KEY[a]}"
ifconfig wlan0 | grep inet


FILENAME=$ITEM".wpa"


cat "$FILENAME">/etc/wpa_supplicant.conf | xargs
printf "Creating new configuration using $ITEM\n"


printf " \n parameter KUN===== $KUN\n"
printf " parameter ITE===== $ITEM\n"
connect $ITEM
}

function connect (){
printf "Connecting using file $*\n"

# menangkap hasil fungsi sebelumnya

ESSID=$*
```

```
37.
38.   # melepas ip dan mematikan wlan
39.
40.   kill $(ps aux | grep -E '[w]pa_supplicant.*\'wlan0'' | awk
41.   '{print $2}') 2>/dev/null | xargs
42.   dhclient wlan0 -r
43.   ifconfig wlan0 down
44.
45.
46.   # membuat hubungan baru dengan interface
47.
48.   iwconfig wlan0 mode managed essid "$ESSID"
49.   printf "Configured interface wlan0; ESSID is $ESSID\n"
50.   ifconfig wlan0 up
51.   printf "Interface wlan0 is up\n"
52.   wpa_supplicant -B -Dwext -iwlan0 -c/etc/wpa_supplicant.conf
53.   2>/dev/null | xargs
54.   printf "wpa_supplicant running, sleeping for 15...\n"
55.
56.
57.   # menunggu koneksi sebelum mendapatkan IP
58.
59.   sleep 15
60.   printf "Running dhclient\n"
61.   dhclient wlan0
62.
63.
64.   # menampilkan IP device
65.
66.   ifconfig wlan0 | grep inet
67.   echo $KUN
68.
69.   }
70.
71.
72.   eval LIST=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
73.   '/ESSID/{print $2}') )
74.   eval KEY=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
```

```
'/key/{print $2}') )
eval MAC=( $(iwlist wlan0 scan 2>/dev/null | awk -F":"
'/Address/{print $2 $3 $4 $5 $6 $7}') )

for (( num=0; num <= ${#LIST[@]}-1; num++ ))
do
 echo "Hasil :--------------\n"
if [ "${KEY[num]}" == "off" ]; then
 conf_create $num


 #hilangan coment untuk jaringan UB....
 ./login.sh

 ./ping.py
 else
 echo "Access Point ${LIST[num]} dengan  MAC Address
${MAC[num]}" >> logWifi.txt
 echo "Adalah Access SECURE, maka tidak dilakukan pengujian
lebih lanjut" >> logWifi.txt
 echo "------------------------------------------------------
-----------" >> logWifi.txt

fi
done
```

Lampiran 3 Source code Login nas (login.sh)

```
1.  #!/bin/bash
2.
3.  cat /home/pi/login.txt | nc nas.ub.ac.id 80;
```

Lampiran 4 Login.txt

```
1.  POST /webAuth/ HTTP/1.1
2.  Host: nas.ub.ac.id
3.  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:13.0) Gecko/20120313
4.  Firefox/13.0a1
5.  Accept:
6.
7.  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
8.
9.  Accept-Language: en-us,en;q=0.5
10. Accept-Encoding: gzip, deflate
11. Connection: keep-alive
12. Referer: http://nas.ub.ac.id/webAuth/
13. Content-Type: application/x-www-form-urlencoded
14. Content-Length: 62
15.
16. username=0910683016&password=aa11223344&pwd=aa11223344&secret=true
17.
18.
```

Lampiran 5 Source code analisis latency dan throughput (Ping.py)

```
#!/usr/bin/env python

import os, sys, socket, struct, select, time
import time
import urllib
import urllib2
import sys
import itertools as it
import subprocess

test = subprocess.Popen(["iwconfig"],stdout=subprocess.PIPE)
output = test.communicate()[0]

url='http://blog.ub.ac.id/adhtavrc/files/2013/09/Angie-
Pramudita-Adhitama_0910683016_kelas-F.docx'


t0=time.time()
source = urllib.urlopen(url).read()
file = open('file.docx','w')
file.write(source)
file.close()
waktu= (time.time()-t0)
troughput=(437/waktu)

print output
print('\n 437 KB file terdownload dalam %s second ' %waktu)
print(' trougputh 437 KB file = %f KB/s \n' % troughput)

fd = open("logWifi.txt",'a+')
fd.seek(0,2)
fd.write('\n%s' %output)


# From /usr/include/linux/icmp.h; your milage may vary.
ICMP_ECHO_REQUEST = 8 # Seems to be the same on Solaris.
```

```
36.
37. print '\n'
38.
39. def checksum(source_string):
40.     """
41.     I'm not too confident that this is right but testing seems
42.     to suggest that it gives the same answers as in_cksum in
43. ping.c
44.     """
45.     sum = 0
46.     countTo = (len(source_string)/2)*2
47.     count = 0
48.     while count<countTo:
49.         thisVal = ord(source_string[count + 1])*256 +
50. ord(source_string[count])
51.         sum = sum + thisVal
52.         sum = sum & 0xffffffff # Necessary?
53.         count = count + 2
54.
55.     if countTo<len(source_string):
56.         sum = sum + ord(source_string[len(source_string) - 1])
57.         sum = sum & 0xffffffff # Necessary?
58.
59.     sum = (sum >> 16)  +  (sum & 0xffff)
60.     sum = sum + (sum >> 16)
61.     answer = ~sum
62.     answer = answer & 0xffff
63.
64.     # Swap bytes. Bugger me if I know why.
65.     answer = answer >> 8 | (answer << 8 & 0xff00)
66.
67.     return answer
68.
69.
70. def receive_one_ping(my_socket, ID, timeout):
71.     """
72.     receive the ping from the socket.
73.     """
```

```
74.     timeLeft = timeout
75.     while True:
76.         startedSelect = time.time()
77.         whatReady = select.select([my_socket], [], [], timeLeft)
78.         howLongInSelect = (time.time() - startedSelect)
79.         if whatReady[0] == []: # Timeout
80.             return
81.
82.         timeReceived = time.time()
83.         recPacket, addr = my_socket.recvfrom(1024)
84.         icmpHeader = recPacket[20:28]
85.         type, code, checksum, packetID, sequence =
86. struct.unpack(
87.             "bbHHh", icmpHeader
88.         )
89.         if packetID == ID:
90.             bytesInDouble = struct.calcsize("d")
91.             timeSent = struct.unpack("d", recPacket[28:28 +
92. bytesInDouble])[0]
93.             return timeReceived - timeSent
94.
95.         timeLeft = timeLeft - howLongInSelect
96.         if timeLeft <= 0:
97.             return
98.
99.
100. def send_one_ping(my_socket, dest_addr, ID):
101.     """
102.     Send one ping to the given >dest_addr<.
103.     """
104.     dest_addr  =  socket.gethostbyname(dest_addr)
105.
106.     # Header is type (8), code (8), checksum (16), id (16),
107. sequence (16)
108.     my_checksum = 0
109.
110.     # Make a dummy heder with a 0 checksum.
111.     header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
```

```
112. my_checksum, ID, 1)
113.     bytesInDouble = struct.calcsize("d")
114.     data = (192 - bytesInDouble) * "Q"
115.     data = struct.pack("d", time.time()) + data
116.
117.     # Calculate the checksum on the data and the dummy header.
118.     my_checksum = checksum(header + data)
119.
120.     # Now that we have the right checksum, we put that in. It's
121. just easier
122.     # to make up a new header than to stuff it into the dummy.
123.     header = struct.pack(
124.         "bbHHh", ICMP_ECHO_REQUEST, 0,
125. socket.htons(my_checksum), ID, 1
126.     )
127.     packet = header + data
128.     my_socket.sendto(packet, (dest_addr, 1)) # Don't know about
129. the 1
130.
131.
132. def do_one(dest_addr, timeout):
133.     """
134.     Returns either the delay (in seconds) or none on timeout.
135.     """
136.     icmp = socket.getprotobyname("icmp")
137.     try:
138.         my_socket = socket.socket(socket.AF_INET,
139. socket.SOCK_RAW, icmp)
140.     except socket.error, (errno, msg):
141.         if errno == 1:
142.             # Operation not permitted
143.             msg = msg + (
144.                 " - Note that ICMP messages can only be sent
145. from processes"
146.                 " running as root."
147.             )
148.             raise socket.error(msg)
149.         raise # raise the original error
```

```
150.
151.     my_ID = os.getpid() & 0xFFFF
152.
153.     send_one_ping(my_socket, dest_addr, my_ID)
154.     delay = receive_one_ping(my_socket, my_ID, timeout)
155.
156.     my_socket.close()
157.     return delay
158.
159.
160. def verbose_ping(dest_addr, timeout = 2, count = 4):
161.     """
162.     Send >count< ping to >dest_addr< with the given >timeout<
163. and display
164.     the result.
165.     """
166.     for i in xrange(count):
167.         print "ping %s..." % dest_addr,
168.         try:
169.             delay  =  do_one(dest_addr, timeout)
170.         except socket.gaierror, e:
171.             print "failed. (socket error: '%s')" % e[1]
172.             break
173.
174.         if delay  ==  None:
175.             print "failed. (timeout within %ssec.)" % timeout
176.         else:
177.             delay  =  delay * 1000
178.             print "get ping in %0.4fms" % delay
179.             fd.write(' ping %fms \n' % delay)
180.     print
181.
182.
183. if __name__ == '__main__':
184.     verbose_ping("www.ub.ac.id")
185.
186. fd.write(' \n 437  KB file terdownload dalam %f second ' %waktu
187. +'\n')
```

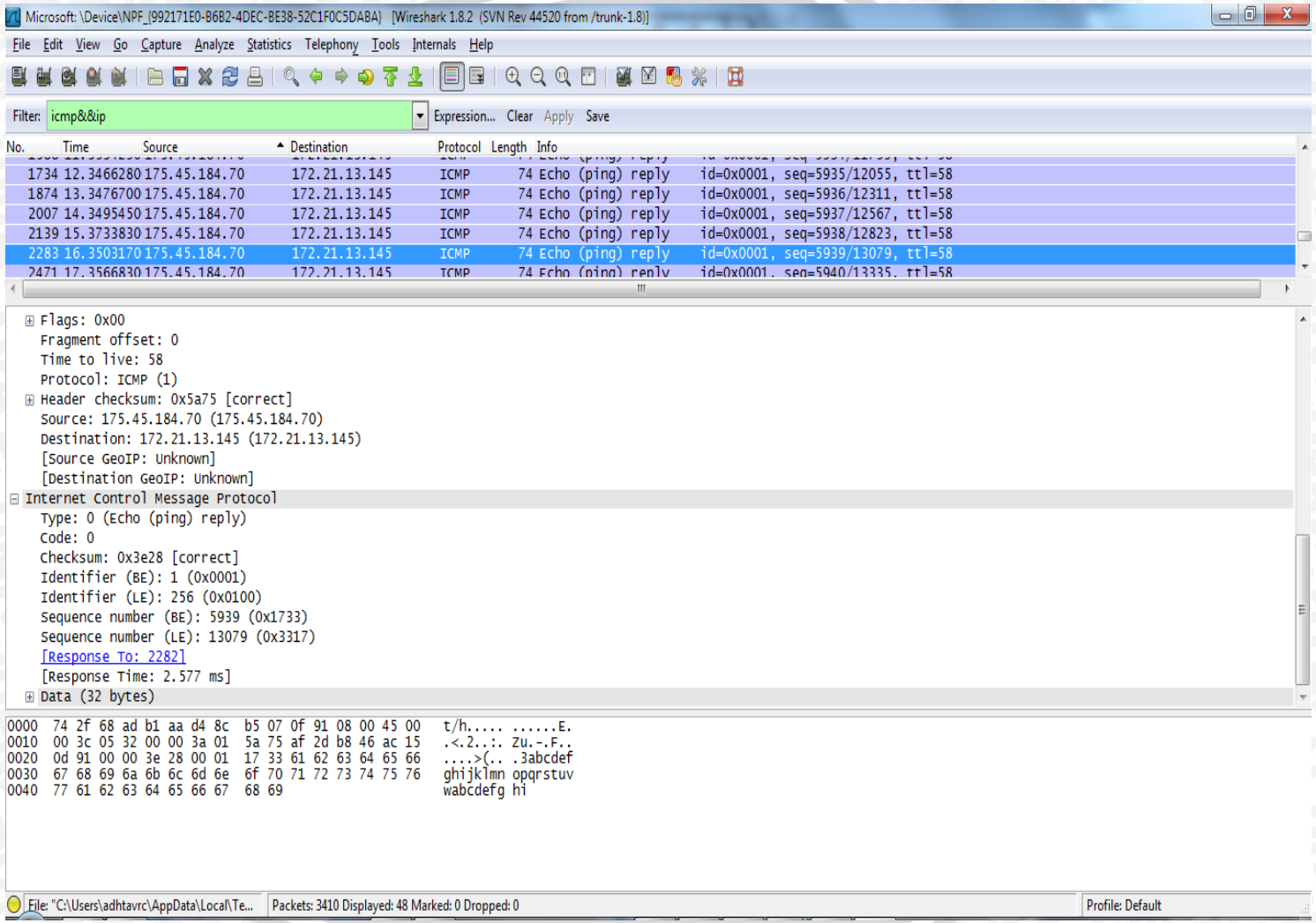
```
188.  fd.write(' trougputh  437 KB file = %f KB/s \n' % troughput
189.  +'\n')
190.  fd.write(' ----------------------------------------------------
191.  ----------------------------------\n')
192.
193.  fd.close()
194.
```

Lampiran 6 LogWifi.txt

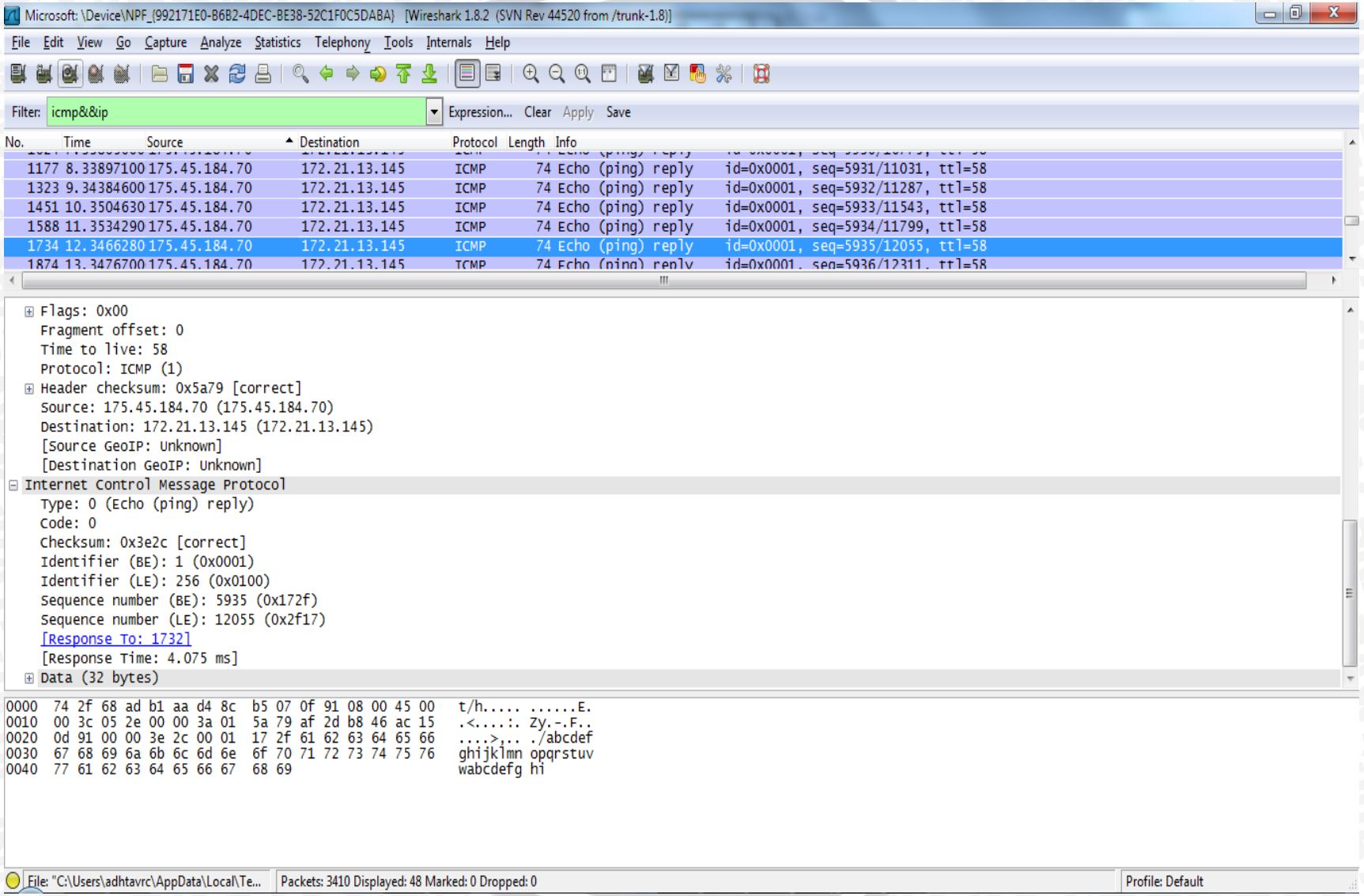
```
wlan0     IEEE 802.11bgn  ESSID:"PTIIK"
Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.437 GHz  Access Point:
00:02:6F:87:53:10
          Bit Rate:72 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=97/100  Signal level=63/100  Noise
level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
frag:0
          Tx excessive retries:0  Invalid misc:0   Missed
beacon:0

 ping 90.704918ms
 ping 96.349001ms
 ping 91.239929ms
 ping 89.757204ms

 133.2578125  KB file terdownload dalam 1.923503 second
 trougputh 133.2578125 KB file = 69.278716 KB/s

 ------------------------------------------------------------
------------------------------------------------------

wlan0     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Auto  Frequency=2.412 GHz  Access Point: Not-
Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
frag:0
          Tx excessive retries:0  Invalid misc:0   Missed
beacon:0

 ping 60.501099ms
 ping 59.821129ms
 ping 60.449123ms
 ping 60.162067ms

 133.2578125  KB file terdownload dalam 2.201003 second
 trougputh 133.2578125 KB file = 60.544128 KB/s

 ------------------------------------------------------------
```

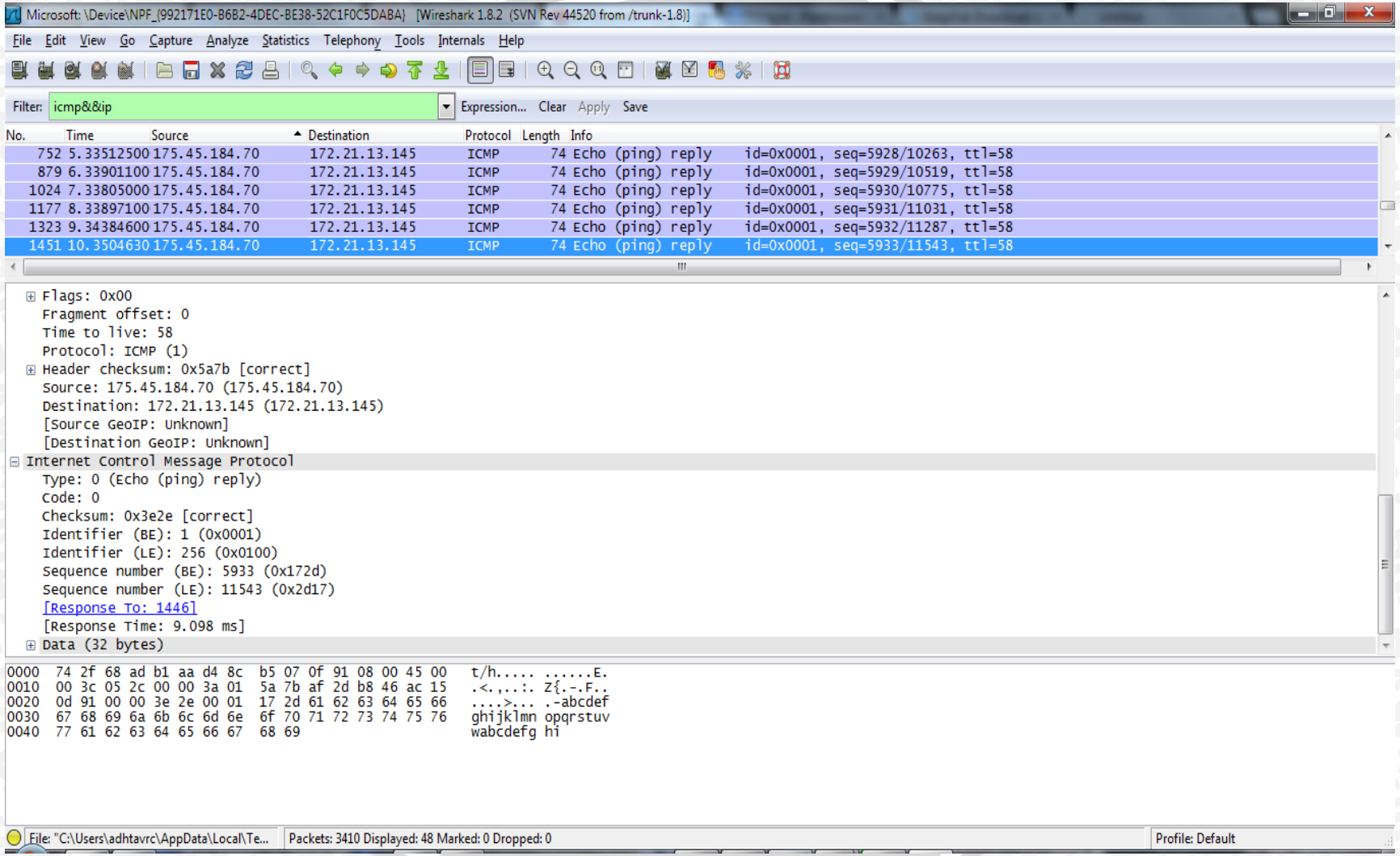
```
-------------------------------------------------------

wlan0     IEEE 802.11bgn  ESSID:"PTIIK"
Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.437 GHz  Access Point:
00:02:6F:87:57:90
          Bit Rate:150 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=100/100  Signal level=97/100  Noise
level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
frag:0
          Tx excessive retries:0  Invalid misc:0   Missed
beacon:0

 ping 48.410892ms
 ping 41.843891ms
 ping 46.281099ms
 ping 45.426846ms

 133.2578125  KB file terdownload dalam 0.510846 second
 trougputh 133.2578125 KB file = 260.857042 KB/s

-------------------------------------------------------------
-------------------------------------------------------
Access Point koneksi2 dengan  MAC Address 162F68ADB1AA
Adalah Access SECURE, maka tidak dilakukan pengujian lebih
lanjut
-------------------------------------------------------------
-----

wlan0     IEEE 802.11bgn  ESSID:"smartfren_kontrakan"
Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.412 GHz  Access Point:
8C:44:35:02:3B:F0
          Bit Rate:150 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=99/100  Signal level=48/100  Noise
level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
frag:0
          Tx excessive retries:0  Invalid misc:0   Missed
beacon:0

 ping 132.688046ms
 ping 339.698076ms
 ping 122.383833ms
 ping 454.823017ms
```

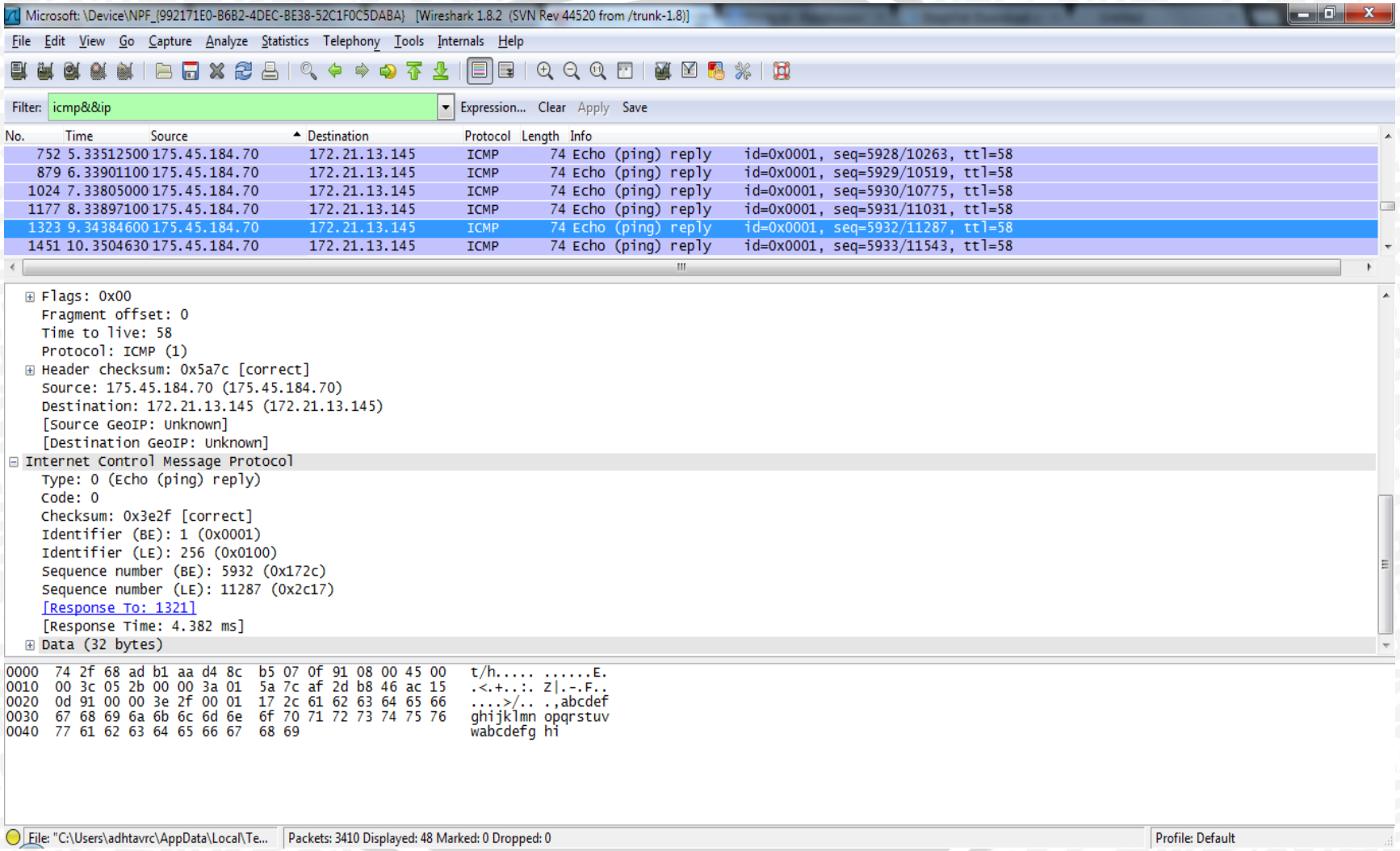
```
 437  KB file terdownload dalam 6.946461 second
 trougputh  437 KB file = 62.909732 KB/s

------------------------------------------------------------
------------------------------------------------------
Access Point koneksi2 dengan  MAC Address 162F68ADB1AA
Adalah Access SECURE, maka tidak dilakukan pengujian lebih
lanjut
------------------------------------------------------------
-----

wlan0     IEEE 802.11bgn  ESSID:"smartfren_kontrakan"
Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.412 GHz  Access Point:
8C:44:35:02:3B:F0
          Bit Rate:150 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=77/100  Signal level=45/100  Noise
level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
frag:0
          Tx excessive retries:0  Invalid misc:0   Missed
beacon:0

 ping 283.501148ms
 ping 353.197098ms
 ping 105.228901ms
 ping 123.942137ms

 437  KB file terdownload dalam 4.580335 second
 trougputh  437 KB file = 95.407871 KB/s

------------------------------------------------------------
------------------------------------------------------
Access Point AP Dosen dengan  MAC Address 162F68ADB1AA
Adalah Access SECURE, maka tidak dilakukan pengujian lebih
lanjut
------------------------------------------------------------
------------------------------------------------------
Access Point koneksi2 dengan  MAC Address 162F68ADB1AA
Adalah Access SECURE, maka tidak dilakukan pengujian lebih
lanjut
------------------------------------------------------------
-----
```

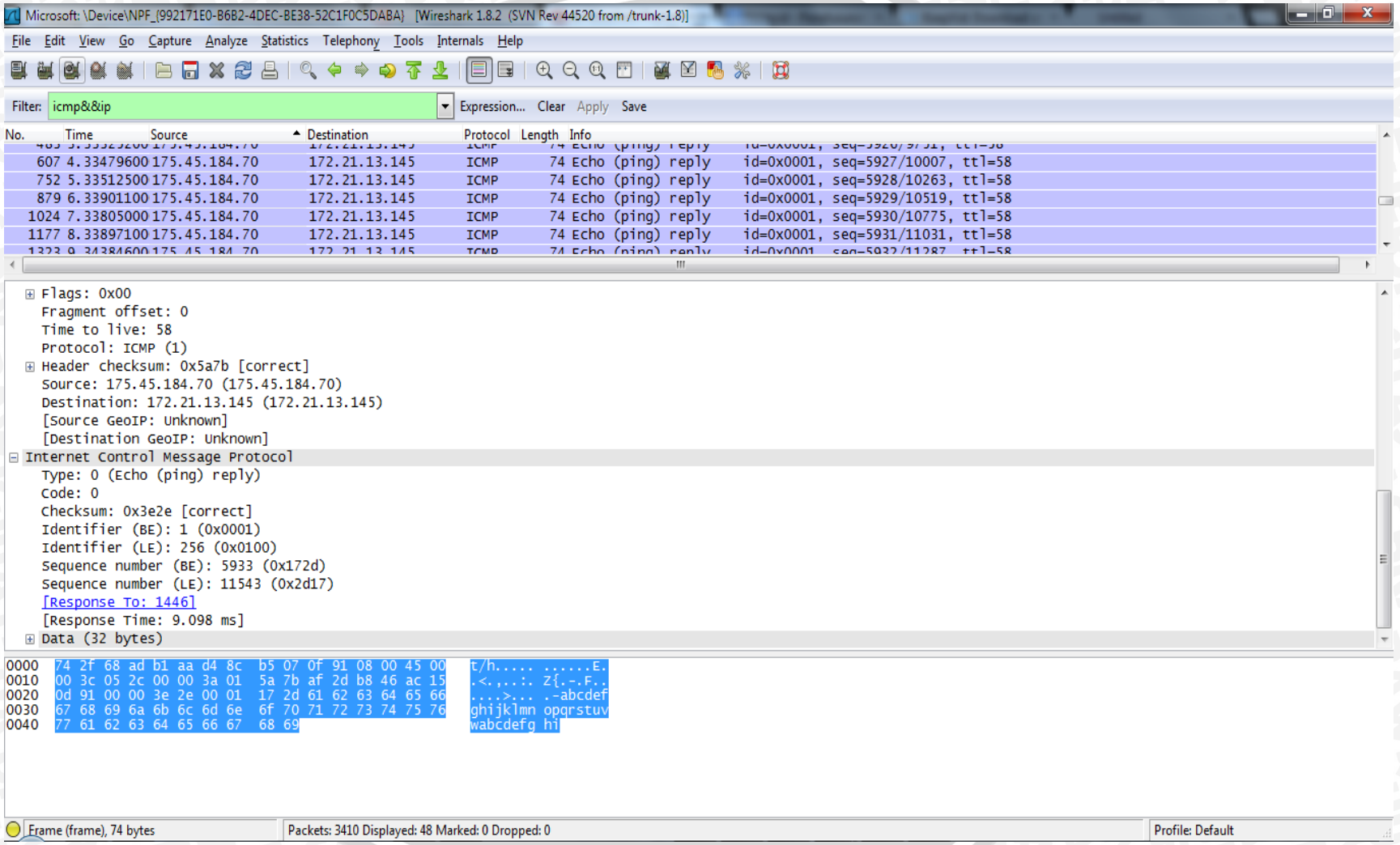


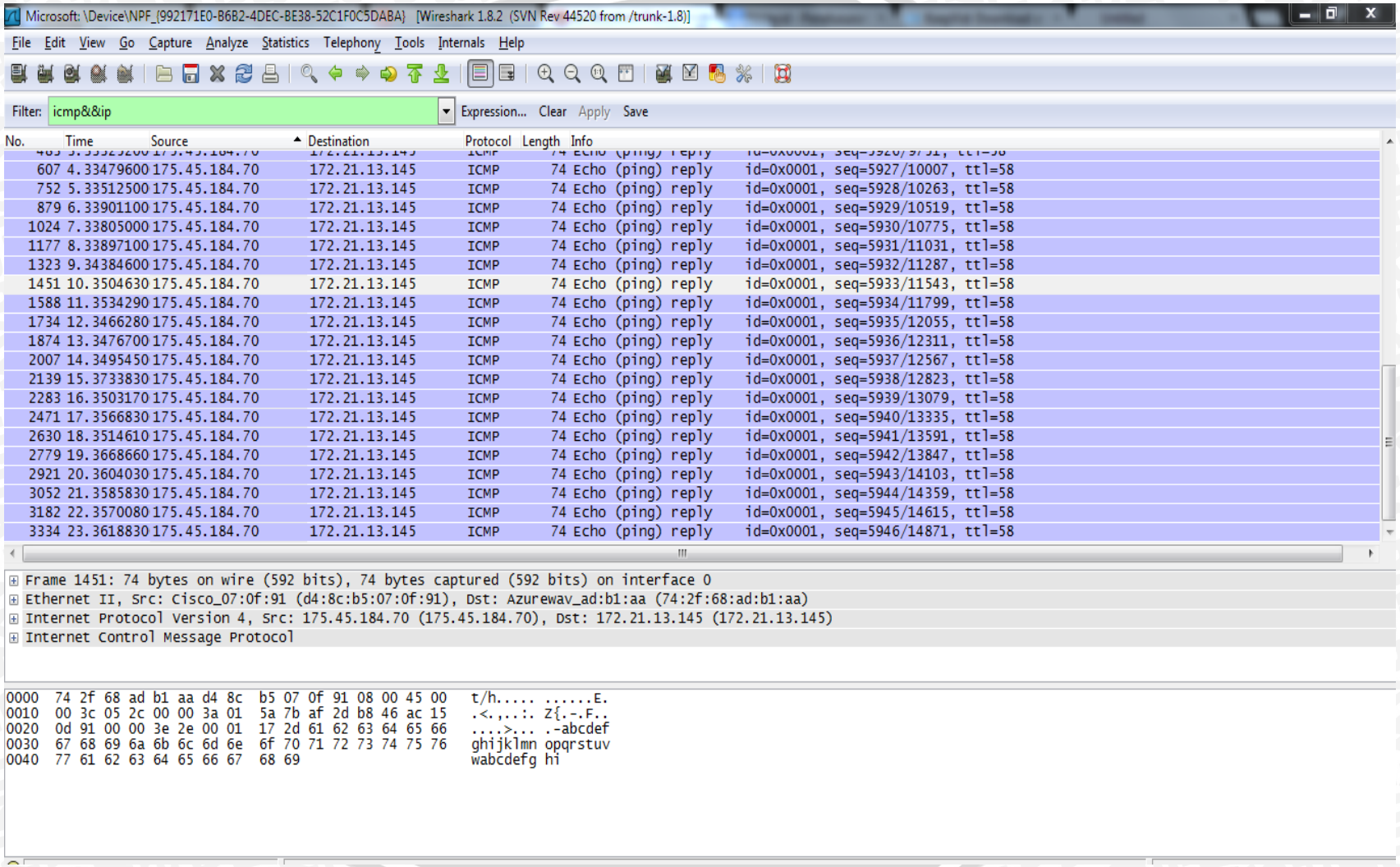
Lampiran 7 Screen shoot pengujian Wireshark

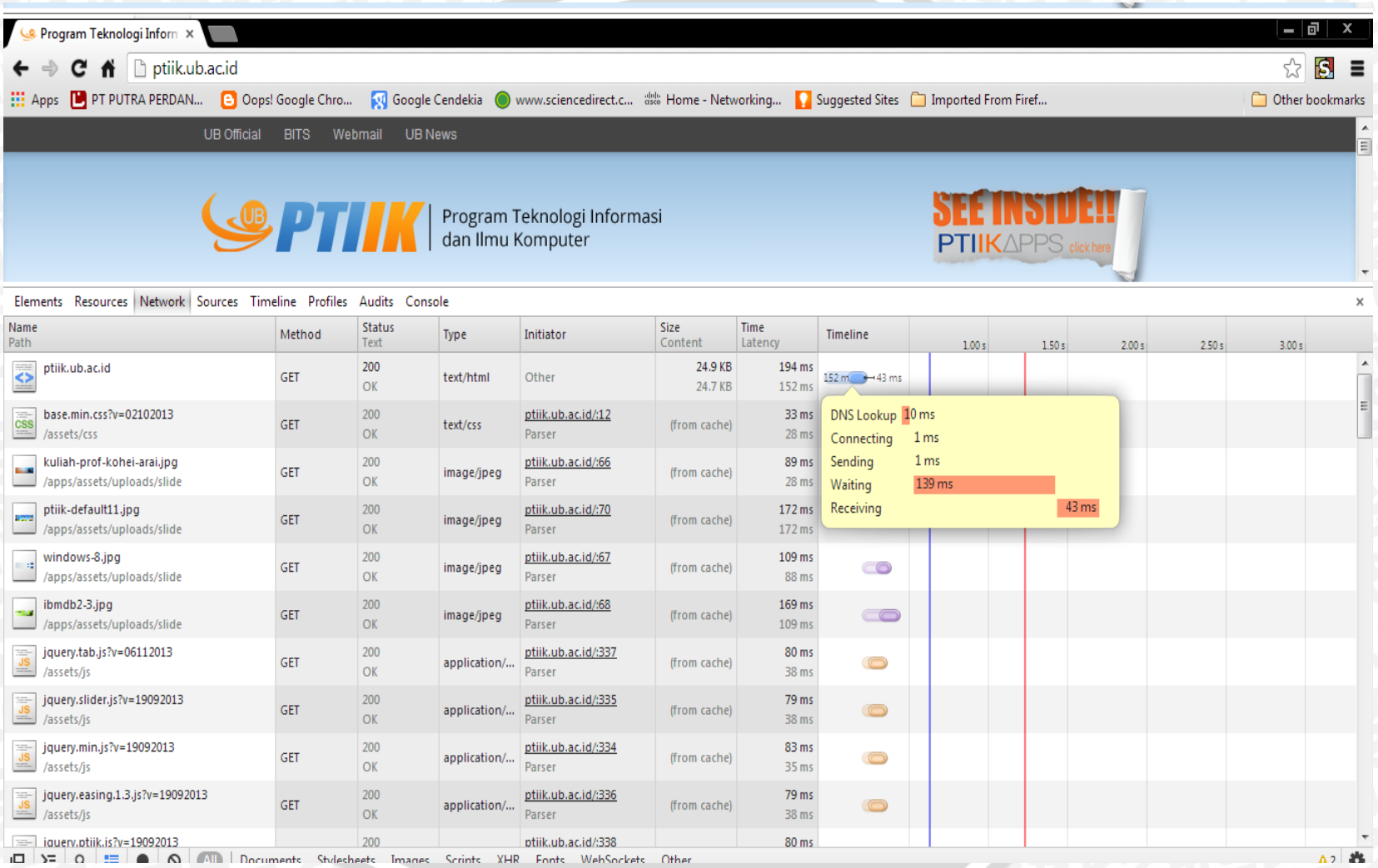




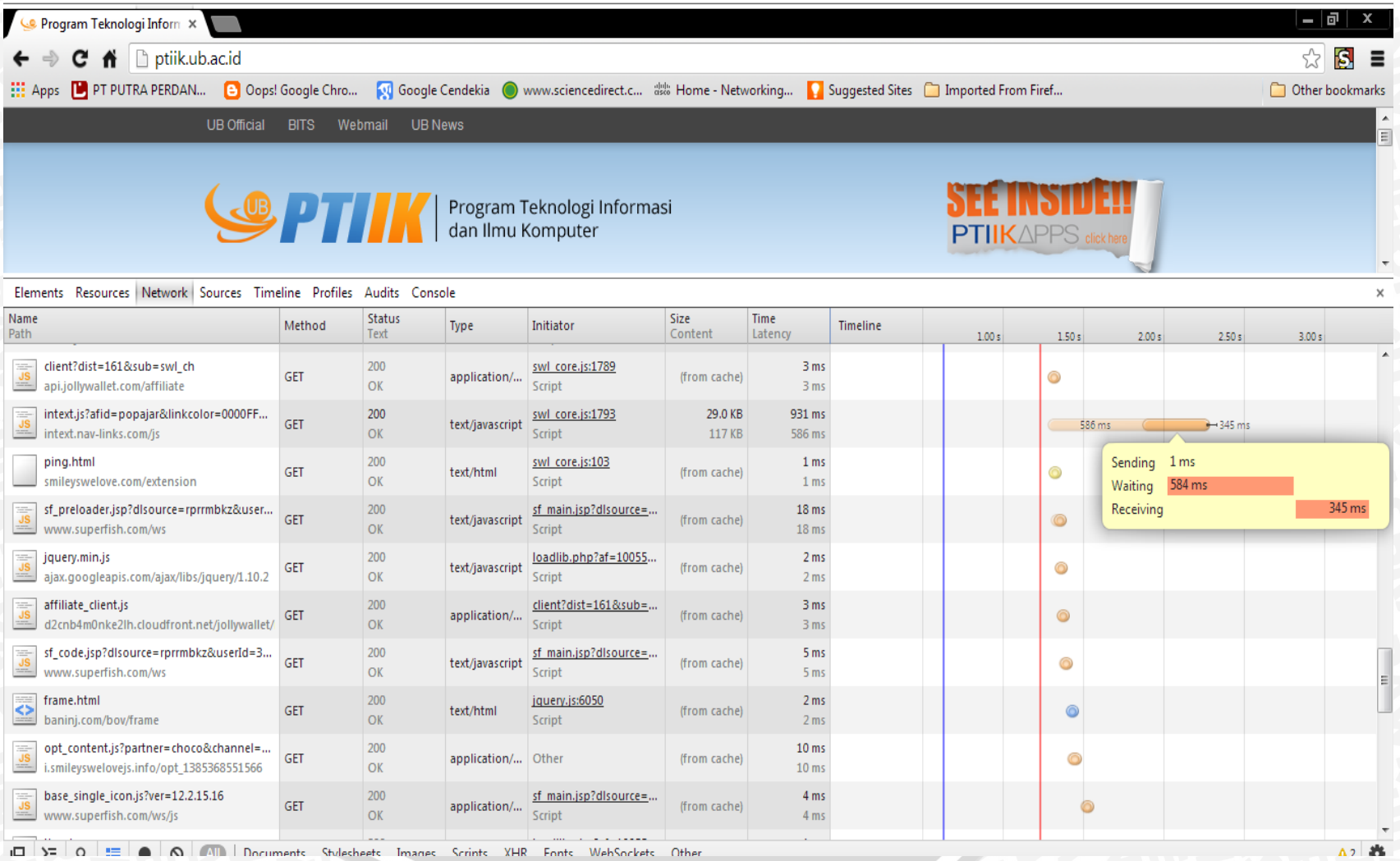


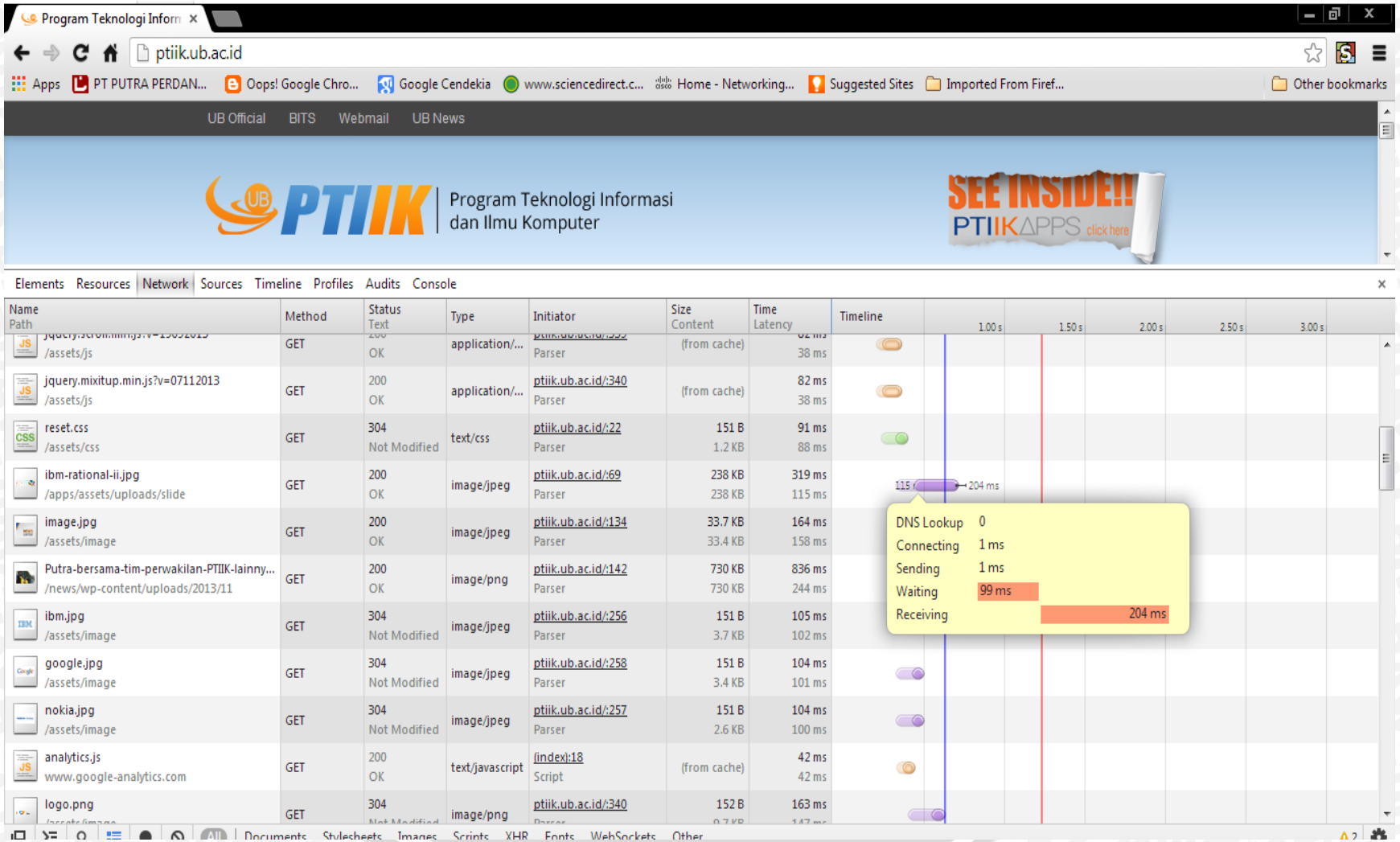


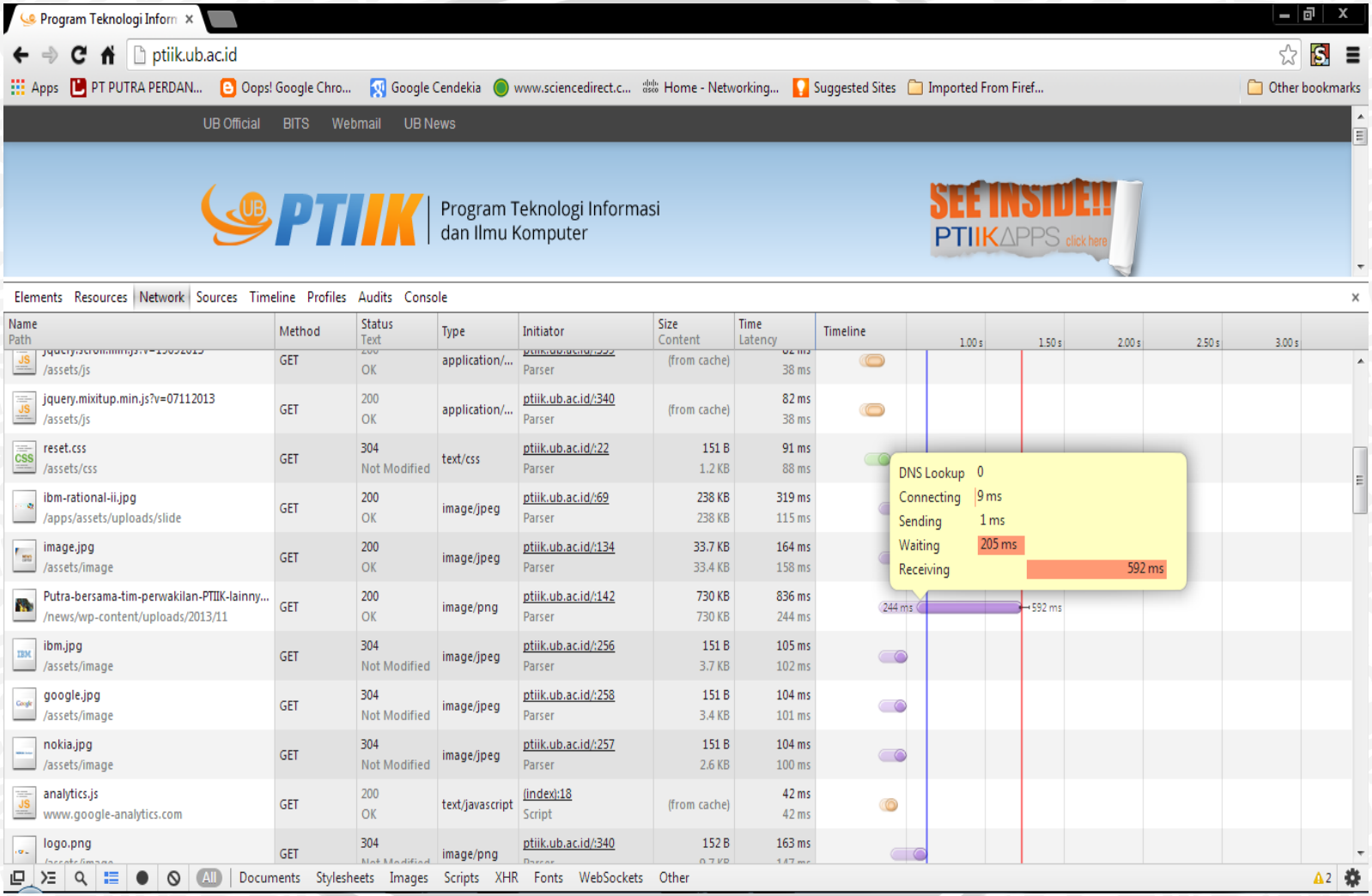




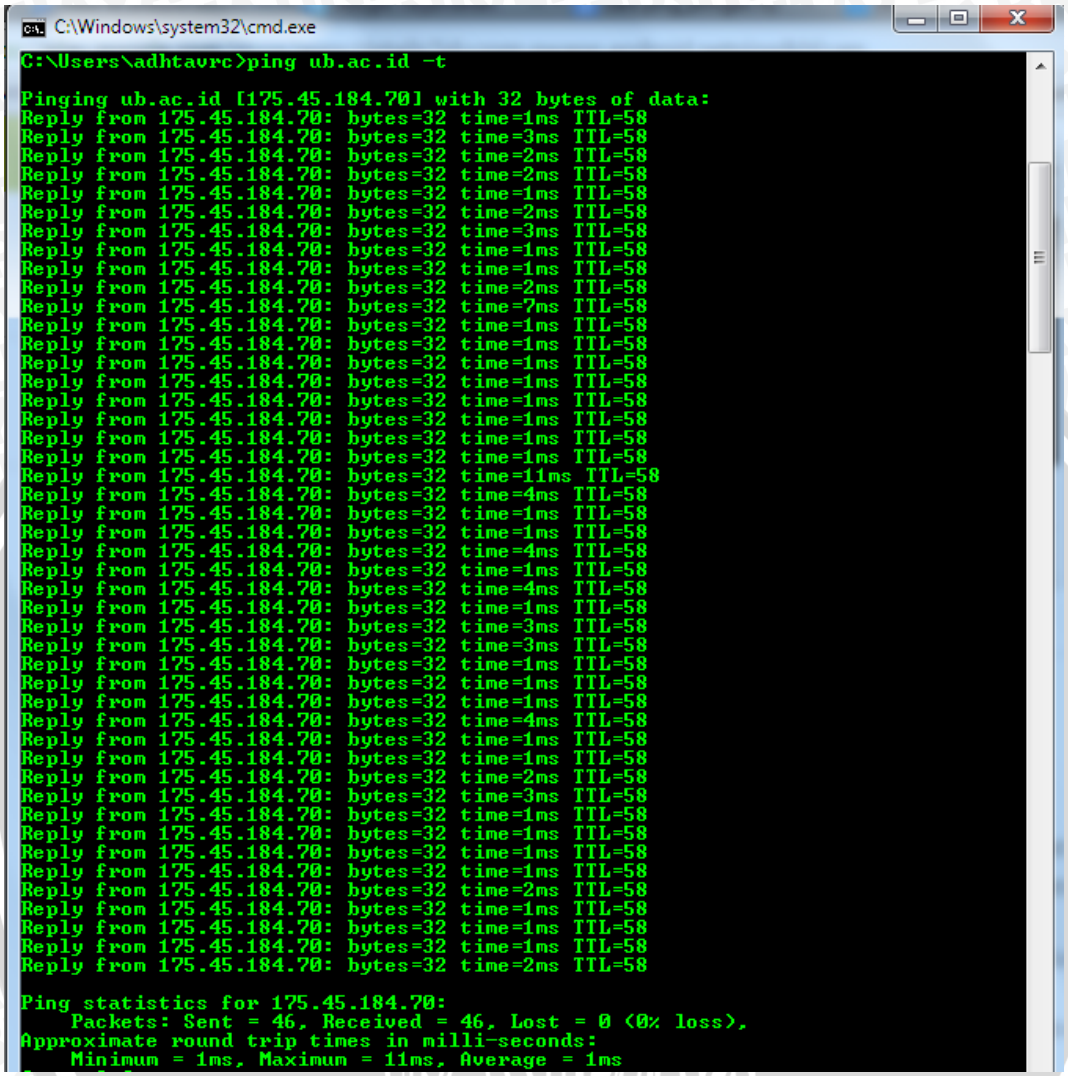
Lampiran 8 Screen Shoot Chrome







Lampiran 9 latency Asus A43s



Lampiran 10 Data Pengujian Throughput

Waktu Uji	Raspberry Pi	Asus A43s	Lenovo A3000
8	550.5644	545.4232	556.224
9	453.21	452.122	440.345
11	375.228	378.211	390.2547
12	350.72	349.748	347.2144
14	380.9116	352.2845528	337.2113
Rata - rata	422.1268	415.5577506	414.24988

Waktu Uji	*Raspberry Pi*	Asus A43s	Lenovo A3000
8	533.975936	545.25	530.5061728
9	654.104553	652.238806	652.238806
11	574.066243	581.6666667	575
12	689.624743	672.3076923	690.6507937
14	351.911676	352.2845528	350.4193548
Rata - rata	560.7366302	560.7495436	559.7630255

Waktu Uji	Raspberry Pi	Asus A43s	Lenovo A3000
8	600.124	611.324	605.224
9	575.32	583.21	585.4334
11	425.758	429.3556	576.6356
12	411.0544	427.643	347.2144
14	467.435	450.976	412.64
Rata - rata	495.93828	500.50172	505.42948

Waktu Uji	*Raspberry Pi*	Asus A43s	Lenovo A3000
8	561.5547787	567.3324	563.9847243
9	560.8781843	562.523602	559.3390687
11	458.3507477	463.0777556	513.9634333
12	483.7997143	483.2328974	461.6931979
14	400.086092	385.1817019	366.7568849
Rata - rata	492.9339034	492.2696714	493.1474618

Lampiran 11 Data Pengujian Latency

Waktu	Raspberry	Asus	android
8	12.4	12.2	10.1
9	36.7	34.1	37.21
11	59.2	62.1	58.2
12	80.1	81.2	76.1
14	50.2	55.1	56.6
Rata - rata	47.72	48.94	47.642

Waktu	Raspberry	Asus A43s	android
8	63.7	60.2	61.4
9	46	45	44.9
11	36.175	30.7	34.55
12	17.1	15.6	15.575
14	23.367	22	21.175
Rata – rata	37.2684	34.7	35.52

Waktu	Raspberry	Asus	android
8	7.1	6.8	7.2
9	15.67	16.1	16.31
11	16.5	15.1	16.9
12	20.3	19.67	20.1
14	19.7	18.9	19
Rata - rata	15.854	15.314	15.902

Waktu	Raspberry	Asus A43s	android
8	27.73333333	26.4	26.23333333
9	32.79	31.73333333	32.80666667
11	37.29166667	35.96666667	36.55
12	39.16666667	38.82333333	37.25833333
14	31.089	32	32.25833333
Rata - rata	33.61413333	32.98466667	33.02133333

Tabel merah merupakan rata – rata pengujian *latency*