TUGAS AKHIR – KI141502

IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS DESTINATION SEQUENCE DISTANCE VECTOR MENGGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA

AHADYN ALIEF ARLINGGA
NRP 5110100079

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

Dosen Pembimbing II
Baskoro Adi Pratomo, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF PICTURE STREAMING ON MESH NETWORK BASED ON DESTINATION SEQUENCE DISTANCE VECTOR ROUTING PROTOCOL USING RASPBERRY PI FOR HIGHWAY MONITORING

AHADYN ALIEF ARLINGGA
NRP 5110100079

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

Supervisor II
Baskoro Adi Pratomo, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2015

LEMBAR PENGESAHAN

IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS DESTINATION SEQUENCE DISTANCE VECTOR MENGGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
AHADYN ALIEF ARLINGGA
NRP : 5110100079

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Royyana Muslim Ijtihadie, S.Kom, M.Kom., PhD ..... (pembimbing 1)
NIP: 197708242006041001

Baskoro Adi Pratomo, S.Kom, M.Kom ..... (pembimbing 2)
NIP: 198702182014041001

SURABAYA
JANUARI 2015

IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS DESTINATION SEQUENCE DISTANCE VECTOR MENGGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA

Nama Mahasiswa	: AHADYN ALIEF ARLINGGA
NRP	: 5110100079
Jurusan	: Teknik Informatika FTIF-ITS
Dosen Pembimbing 1	: Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.
Dosen Pembimbing 2	: Baskoro Adi Pratomo, S.Kom., M.Kom.

Abstrak

Keamanan adalah hal yang penting melihat saat ini jumlah kejahatan yang semakin bertambah di berbagai tempat di Indonesia. Untuk itu diciptakan sebuah perangkat kamera yang berfungsi untuk menangkap gambar baik secara real time atau hanya untuk merekam kegiatan apa saja yang terjadi di tempat tertentu. Closed-circuit Television atau disingkat menjadi CCTV atau kamera pengamatan ini banyak digunakan sebagai alat pemantau keadaan sekitar. CCTV banyak digunakan oleh berbagai instansi-instansi besar seperti perusahaan besar dan pihak kepolisian bahkan tak jarang digunakan untuk keperluan pribadi.

Penerapan CCTV di Indonesia mulai dilakukan oleh pihak kepolisian untuk mengawasi jalan raya dan hasilnya akan dipantau secara langsung dari markas besar kepolisian. Sistem yang dibangun oleh pihak kepolisian rata-rata masih menggunakan sistem terpusat dengan menggunakan menara radio pada base station sebagai media penerima hasil keluaran dari tiap kamera CCTV. CCTV dan base station yang dibangun sendiri masih tergolong mahal dari segi pembangunan, sumber tenaga dan infrastruktur jaringan. Oleh sebab itu diperlukan alternatif lain agar bisa mengurangi biaya yang keluar untuk membangun sistem

keamanan tersebut. Dengan memanfaatkan komputer mini Raspberry Pi yang lebih terjangkau dari segi harga, sumber tenaga yang kecil, dan pembangunan sistemnya yang lebih sederhana diharapkan akan menjadi salah satu solusi dari permasalahan yang ada.

Pengimplementasian picture streaming pada jaringan mesh menggunakan perangkat Raspberry Pi dengan algoritma routing Destination Sequence Distance Vector (DSDV) Protocol merupakan salah satu alternatif untuk permasalahan tersebut. Dengan mengimplementasikan perangkat tersebut akan menawarkan sebuah kemungkinan untuk membangun perangkat pemantau yang hanya tinggal pasang dan langsung melakukan tugasnya.

Kata Kunci: Picture Streaming, Jaringan ad-hoc, Bellman-Ford, Destination Sequence Distance Vector.

IMPLEMENTATION OF PICTURE STREAMING ON MESH NETWORK BASED ON DESTINATION SEQUENCE DISTANCE VECTOR ROUTING PROTOCOL USING RASPBERRY PI FOR HIGHWAY MONITORING

Student's Name	: AHADYN ALIEF ARLINGGA
Student's ID	: 5110100079
Department	: Teknik Informatika FTIF-ITS
First Advisor	: Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.
Second Advisor	: Baskoro Adi Pratomo, S.Kom., M.Kom.

Abstract

Security is the most important thing by seeing the number of criminality which keep increasing in the whole places in Indonesia. Thus a camera is created to capture picture in real time or just for recording what kind of activities which occuring in certain places. Closed-circuit Television also known as CCTV or observations cameras are used by many big agencies like big corporations and Police Departement even used for personal use.

Applying CCTV in Indonesia was started by the Police Departement to watch highway and the result will be monitored from Police Headquarters in real time. The system which is built by the police still uses centered system which uses radio tower at base station as output receiver from each CCTV. CCTV and base station which are built by the police are expensive in terms of development, resources and network infrastructure. Thus we need another alternative to cut the cost for building this security system. By using mini computer Raspberry Pi which is affordable, small resources, and the system development which is simpler will be a solution for the problem.

Picture streaming implementation in mesh network using Raspberry Pi with Destination Sequence Distance Vector (DSDV) Protocol is one of alternatives for that problem. By implementating this device, it will offer a chance to build a monitoring device which only need to set it up and it will do its job.

Keywords: Picture Streaming, Wireless Mesh Network, Babel routing protokol.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul ***"IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS PROTOKOL DESTINATION SEQUENCE DISTANCE VECTOR MENGGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA".***

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ayah, Ibu, dan Kakak saya yang telah memberikan dukungan serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
2. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD. selaku pembimbing I yang telah membantu dan membimbing penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
3. Bapak Baskoro Adi Pratomo, S.Kom., M.Kom. selaku pembimbing II yang telah memberikan motivasi, nasehat, bimbingan dan bantuan yang banyak kepada penulis dalam mengerjakan Tugas Akhir ini.

4. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
5. Ainna Rizkha Dhiany yang selalu memberikan dukungan dan penghilang rasa cemas di kala kesusahan.
6. Ramadhan Wijayanto, Muhtarom Widodo dan Ilmal Alifriansyah selaku teman satu bimbingan yang selalu membantu penulis dalam pengerjaan Tugas Akhir ini.
7. Teman-teman laboratorium *Grid Computing* yang telah berbagi ilmu dan informasi dan motivasi dalam menyelesaikan Tugas Akhir ini.
8. Teman-teman seperjuangan SW 111 yang membantu dan selalu menghibur di kala bingung dalam pengerjaan Tugas Akhir.
9. Teman-teman HMTC yang telah membantu, berbagi ilmu dan motivasi kepada penulis.
10. Teman-teman TC angkatan 2010 yang selalu menjaga kebersamaan, kakak-kakak angkatan yang memberi pengarahan serta adik-adik angkatan 2011 dan 2012 yang membuat penulis untuk selalu belajar.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Januari 2015

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Pada bagian ini akan dijelaskan beberapa hal dasar mengenai tugas akhir ini yang meliputi: latar belakang, tujuan, manfaat permasalahan, batasan permasalahan, metodologi serta sistematika penulisan tugas akhir. Latar belakang berisi mengenai hal-hal yang melatarbelakangi pemilihan judul tugas akhir. Rumusan masalah memuat hal-hal yang harus diselesaikan. Batasan masalah berisi batasan-batasan yang melingkupi pembuatan tugas akhir ini. Tujuan berisi tujuan dari pembuatan tugas akhir ini. Metodologi membahas mengenai metode pengerjaan tugas akhir. Sistematika penulisan membahas sistematika penulisan buku tugas akhir sebagai laporan dari pengerjaan tugas akhir. Penjelasan tentang hal-hal tersebut diharapkan dapat memberikan gambaran umum mengenai permasalahan sehingga penyelesaian masalah dapat dipahami dengan baik.

1.1 Latar Belakang

Keamanan adalah hal yang penting melihat saat ini jumlah kejahatan yang semakin bertambah di berbagai tempat di Indonesia. Untuk itu diciptakan sebuah perangkat kamera yang berfungsi untuk menangkap gambar baik secara real time atau hanya untuk merekam kegiatan apa saja yang terjadi di tempat tertentu. Closed-circuit Television atau disingkat menjadi CCTV atau kamera pengamatan ini banyak digunakan sebagai alat pemantau keadaan sekitar. CCTV banyak digunakan oleh berbagai instansi-instansi besar seperti perusahaan besar dan pihak kepolisian bahkan tak jarang digunakan untuk keperluan pribadi.

Penerapan CCTV di Indonesia mulai dilakukan oleh pihak kepolisian untuk mengawasi jalan raya dan hasilnya akan dipantau secara langsung dari markas besar kepolisian. Sistem yang dibangun oleh pihak kepolisian rata-rata masih menggunakan

sistem terpusat dengan menggunakan menara radio pada base station sebagai media penerima hasil keluaran dari tiap kamera CCTV. CCTV dan base station yang dibangun sendiri masih tergolong mahal dari segi pembangunan, sumber tenaga dan infrastruktur jaringan. Oleh sebab itu diperlukan alternatif lain agar bisa mengurangi biaya yang keluar untuk membangun sistem keamanan tersebut. Dengan memanfaatkan komputer mini Raspberry Pi yang lebih terjangkau dari segi harga, sumber tenaga yang kecil, dan pembangunan sistemnya yang lebih sederhana diharapkan akan menjadi salah satu solusi dari permasalahan yang ada.

Pengimplementasian picture streaming pada jaringan mesh menggunakan perangkat Raspberry Pi dengan algoritma routing Destination Sequence Distance Vector (DSDV) Protokol merupakan salah satu alternatif untuk permasalahan tersebut. Dengan mengimplementasikan perangkat tersebut akan menawarkan sebuah kemungkinan untuk membangun perangkat pemantau plug-and-play, yang berarti penyebaran media ini akan semakin mudah karena cukup ditancapkan pada stopkontak maka perangkat langsung berfungsi. Dengan adanya perangkat ini, diharapkan ke depannya pemantauan jalan raya menjadi semakin optimal karena persebaran perangkat ini.

Jaringan mesh mempunyai kelebihan tersendiri daripada jaringan yang lain karena jaringan tersebut bisa mentransmisikan data dari alat-alat yang berbeda secara bersamaan atau bisa dibilang topologi ini mampu bertahan apabila lalu lintas data sangat tinggi. Bila ada salah satu komponen atau node yang gagal atau mati, komponen lain akan menggantikan komponen tersebut sehingga perpindahan data tidak terpengaruh. Kelebihan yang terakhir adalah ketika topologi mengalami pertambahan atau pengurangan node, topologi tidak akan terpengaruh.

DSDV mempunyai kelebihan antara lain lebih efisien untuk route discovery karena kapanpun rute menentukan tujuan atau destinasi baru, node tujuan tersebut sudah tersimpan sebelumnya. Sehingga latency yang diperlukan untuk melakukan penjelajahan

rute sangat kecil atau bisa dibilang sangat cepat. Selain itu DSDV menjamin tidak terjadinya infinity loops. Shortest Path Problem (SPP) adalah masalah pencarian rute terpendek. Pada umumnya SPP berbentuk graph yang berisi node, edge dan matriks bobot. Semisal terdapat dua node yaitu v_s dan v_t pada graph, masalah pencarian rute terpendek dapat didefinisikan sebagai cara untuk menemukan rute dengan minimum jumlah bobot pada edge dari v_s menuju v_t . Biasanya, v_s disebut node sumber dan v_t disebut node tujuan [1].

1.2 Rumusan Masalah

Rumusan masalah yang akan diangkat dalam tugas akhir ini sebagai berikut:

1 Apakah bisa membangun sistem jaringan mesh menggunakan protokol DSDV menggunakan Raspberry Pi.
2 Apakah bisa membangun sistem CCTV di atas jaringan mesh tersebut.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki batasan sebagai berikut:

1. Untuk membangun perangkat pemantau sederhana menggunakan Raspberry Pi dan modul kamera.
2. Protokol *routing* yang digunakan untuk membangun jaringan *mesh* adalah *Destination Sequence Distance Vector Protokol* pada jaringan *mesh* adalah ukuran *buffer* pada tiap *node* bukan jarak.
3. Tugas Akhir ini mengabaikan aspek keamanan jaringan.
4. Jumlah *node* pemantau berupa Raspberry Pi hanya berjumlah 1 buah.

1.4 Tujuan dan Manfaat

Tugas Akhir ini dibuat dengan tujuan untuk membangun suatu sistem perangkat pemantau di dalam jaringan mesh yang menggunakan algoritma DSDV.

Manfaat dengan dibuatnya perangkat ini, diharapkan mampu memantau keadaan jalan raya dengan pemantauan yang lebih menyebar sehingga mampu untuk mengawasi keadaan lalu lintas maupun sebagai alat pencegahan atau penelurusan terhadap tindak kriminal.

1.5 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1.5.1 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan. Informasi dan studi literatur yang diperlukan adalah implementasi jaringan *ad hoc*, jaringan *mesh* algoritma *Distance Vector*, algoritma *Bellman-Ford*, algoritma rute RIP, protokol rute DSDV, percobaan Raspberrry Pi sekaligus dengan metode penggunaan modul kamera.

1.5.2 Pengumpulan Perangkat Keras

Tahap kedua dari pengerjaan tugas akhir adalah pengumpulan alat-alat untuk melakukan implementasi seperti Raspberry Pi, modul kamera, USB WiFi.

1.5.3 Desain Arsitektur Jaringan

Tahap ketiga setelah semua perangkat keras sudah terkumpul akan dibuat desain arsitektur jaringan agar bisa terlihat jaringan

mesh terbentuk atau tidak beserta membangun perangkat pemantau dengan mengkombinasikan modul kamera dengan Raspberry Pi.

1.5.4 Implementasi Perangkat Lunak

.Untuk implementasi perangkat lunak yang digunakan untuk melakukan *picture streaming* adalah bahasa Python dan Shell Script.

1.5.5 Pengujian dan Evaluasi

Aplikasi akan diuji setelah selesai diimplementasikan menggunakan skenario yang sudah dipersiapkan. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perancanaan. Dengan melakukan pengujian dan evaluasi dimaksudkan juga untuk mengevaluasi jalannya program, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

1.6 Sistematika Penulisan Laporan Tugas Akhir

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan dan manfaat dari pembuatan tugas akhir. Rumusan permasalahan, batasan permasalahan dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar ilmu yang mendukung pembuatan tugas akhir. Dasar ilmu tersebut antara lain algoritma Distance Vector, algoritma Bellman-Ford, algoritma rute RIP, algoritma rute DSDV, algoritma, Raspberry Pi dll.

BAB III. PERANCANGAN PERANGKAT LUNAK

Bab ini berisi penjelasan mengenai desain metode dan implementasi yang digunakan dalam tugas akhir seperti alur kerja pengerjaan sistem, pengambilan, pengiriman dan menampilkan gambar yang diterima.

BAB IV. IMPLEMENTASI

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

BAB V. UJI COBA DAN EVALUASI

Bab ini berisi kesimpulan dan saran-saran tugas akhir. Saran berisi hal-hal yang masih bisa dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan tugas akhir.

BAB VI. PENUTUP

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya

BAB II
TINJAUAN PUSTAKA

Bab ini membahas tentang teori dasar yang menunjang penyusunan tugas akhir. Teori dasar tersebut meliputi algoritma Distance Vector, algoritma rute RIP, algoritma rute DSDV, algoritma, Raspberry Pi dll.

2.1 Distance Vector Routing Protokol

Routing dalam istilah komputasi jaringan adalah kegiatan mencari jalan atau rute dari node pengirim menuju node tujuan. Selama pesan berada dalam satu jaringan atau *subnet*, tiap masalah *routing* akan diatasi oleh teknologi yang cocok untuk jaringan tersebut. Contoh yang dapat kita lihat adalah pada Ethernet dan ARPANET masing-masing mempunyai cara sendiri agar pesan dari node pengirim dalam suatu jaringan bisa sampai ke node tujuan. Maka dari itu pesan harus melewati *gateway* atau gerbang dari suatu subnet yang terhubung ke jaringan. Bila jaringannya tidak saling berdekatan, maka pesan akan melewati jaringan-jaringan lain dan para *gateway* yang menghubungkannya. Ketika pesan sudah masuk ke dalam subnet dari tujuan, maka teknologi dari subnet tersebutlah yang mengurus bagaimana cara pesan tersebut bisa sampai ke node tujuan.

Banyak usaha atau pendekatan yang telah dilakukan untuk mencari rute antar jaringan. Cara-cara yang telah ditemukan tersebut dapat dikategorikan menurut tipe informasi yang diperlukan oleh *gateway* untuk bertukar informasi. Algoritma Distance Vector bergantung pada pertukaran informasi dengan ukuran kecil. Tiap entitas (*gateway* dan host) yang ikut berpatisipasi dalam *routing protokol* diasumsikan untuk menyimpan informasi tentang node-node tujuan dalam suatu sistem. Singkatnya, informasi dari semua entitas yang terhubung dengan suatu jaringan akan disimpulkan oleh satu masukkan yang menjelaskan rute ke semua tujuan di jaringan tersebut. Setiap

database *routing* pada masing-masing entitas mempunyai kolom *gateway* dimana tujuan entitas dari tiap datagram, tidak hanya itu tabel tersebut juga memiliki kolom *metric* atau metrik sebagai tolak ukur jarak atau *cost* yang diperlukan oleh datagram ke entitas lain. Metrik bisa berupa banyak satuan, mulai dari jarak tiap node, *delay time*, biaya uang, ukuran *bandwidth*, jumlah *hop* atau lompatan, dll. Untuk kasus pemilihan metrik ini kita bisa mengambil salah satu atau lebih. Jika menggunakan lebih dari satu parameter untuk metrik, maka metrik tersebut dinamakan *hybrid metric*. Algoritma Distance Vector sendiri mendapatkan namanya berdasarkan fakta bahwa kemungkinan menghitung atau mengkomputasi rute yang optimal bisa terjadi ketika informasi yang ditukar adalah daftar-daftar jarak dari masing-masing entitas. Informasi itu sendiri hanya bisa disebarkan ketika masing-masing entitas saling berdekatan atau bisa dibilang berada pada jaringan yang sama.

Dalam penggunaan database pada Distance Vector Routing Protokol terdapat empat kolom yang harus ada yaitu :

a. Address : alamat IP dari host
b. Gateway : gateway pertama yang harus dilewati untuk mencapai tujuan
c. Interface : bentuk fisik dari jaringan agar mencapati tujuan
d. Metric : angka untuk melihat jarak yang harus ditempuh
e. Timer : untuk melihat kapan pesan ini diperbarui.

Seperti tujuan dari tiap algoritma adalah untuk mencapai suatu hasil yang optimal, algoritma Distance Vector sendiri bertujuan untuk mencari rute paling optimal atau bisa dibilang cepat agar pesan bisa sampai menuju tujuan. Agar lebih mudah kebanyakan implementasi algoritma ini menggunakan jumlah *gateway* atau bisa disebut *hop* yang harus dilalui agar bisa sampai

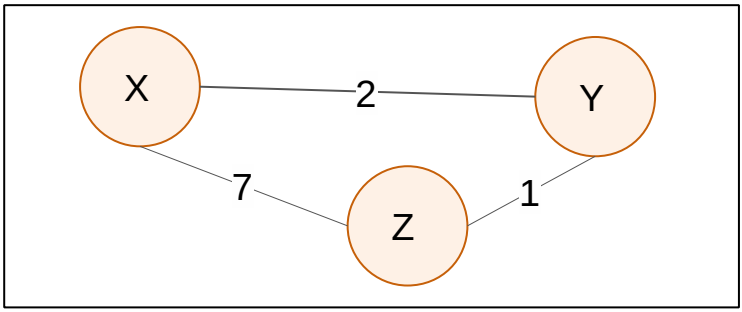
ke tujuan. Tapi untuk keadaan yang lebih rumit, kita bisa menggunakan *hybrid metric* seperti yang dijelaskan sebelumnya.

Distance Vector routing protokol sendiri menggunakam algoritma Bellman-Ford dengan sedikit adaptasi. Algoritma Distance Vector (DV) mempunyai bersifat berulang, asinkronus dan terdistribusi. Berulang berarti proses dalam algoritma ini akan berulang-ulang terus menerus. Asinkronus berarti pesan yang dikirimkan ke node lain tidak perlu adanya konfirmasi lebih lanjut. Terdistribusi karena tiap *node* akan menerima pesan dari node tetangga yang berdekatan, melakukan kalkulasi, lalu hasil kalkulasinya akan didistribusikan kembali ke tiap-tiap tetangganya. Tiap node mempunyai database berupa tabel yang di urus. Tiap baris dalam tabel tersebut berisi kolom node tujuan, node tetangga untuk sampai ke node tujuan (*next hop*). Katakan node X ingin memilih node Y sebagai tujugan dengan melewati node Z yang berdekatan dengan node X atau *next hop*-nya adalah node Z. Maka kalau dinotasikan persamaan akan menjadi :

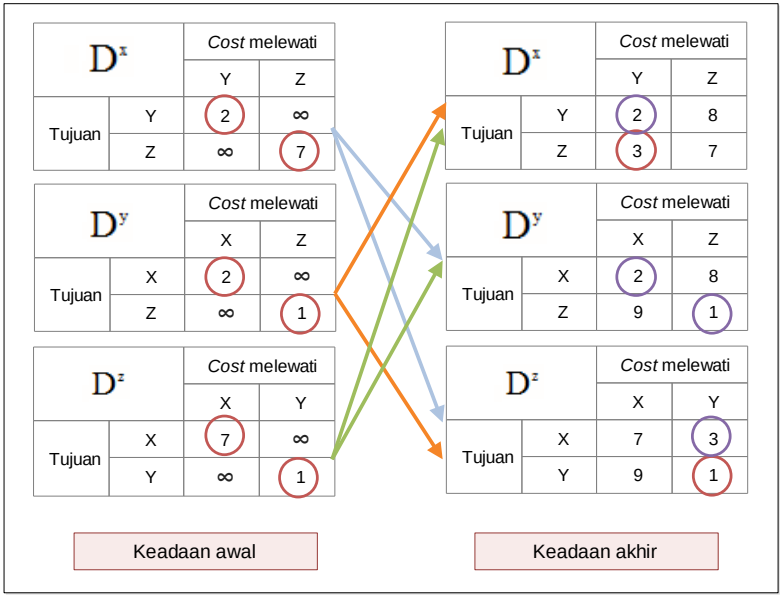
$$D^x(Y,Z) = c(X,Z) + min_w(D^z(Y,w)) \quad \text{.....(1)}$$

Dimana $D^x(Y,Z)$ berarti *distance* atau jarak dari node X menuju node Y melewati node Z itu ditentukan dari $c(X,Z)$ atau *cost* dari node X ke node Z ditambah dengan jarak paling kecil dari node Z menuju node Y $[min_w(D^z(Y,w))]$.

Pada gambar 2.1 adalah contoh skema yang akan dicontohkan dan pada gambar 2.2 adalah penjelasannya. Ada 3 node yaitu X, Y, dan Z dimana masing-masing saling berhubungan secara langsung.



Gambar 2.1 Contoh kasus penentuan jarak terpendek



Gambar 2.2 Penjelasan proses penentuan jarak terpendek

Pada gambar 2.2 diatas menjelaskan tabel *cost* tiap node untuk sampai menuju node tujuan. Pada kolom kiri ada 3 tabel jarak masing-masing node yaitu node X, Y dan Z. Tanda lingkaran pada angka-angka di tabel berarti angka itu adalah *cost* paling optimal. Pada tiga tabel pertama yang berada di sebelah kiri ada

keadaan awal(t0) dimana masing-masing node mencatat *cost* untuk masing-masing tujuannya dimana *next hop* adalah tujuannya langsung sedangkan untuk yang harus melewati node lain akan dianggap tak terhingga atau *infinity* sehingga node tujuan yang menjadi *next hop* langsung akan menjadi *cost* paling optimal yang ditandai dengan lingkaran hitam. Tiga kolom di sebelah kanan menunjukan keadaan setelah t0 yaitu t1 dimana semua node telah saling membagi tabel routing. Setelah itu akan dilakukan kalkulasi untuk mencari tahu apakah ada jalan atau *path* lain yang lebih optimal biaya atau *cost*-nya contohnya pada tabel di node X yang pada awalnya untuk menuju ke node Z memerlukan *cost* sejumlah 7, setelah bertukar informasi dengan node Z ternyata node X memerlukan *cost* lebih kecil dengan melalui node Y terlebih dahulu yaitu 3 [2].

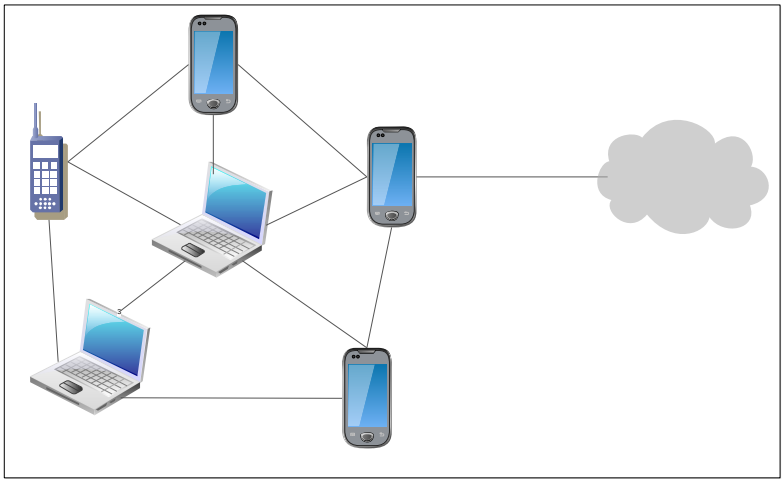
2.2 Jaringan Wireless Mesh

Jaringan *mesh* atau topologi *mesh* atau jaringan jala merupakan salah satu desain jaringan Local Area Network (LAN), topologi dimana semua node atau host dalam jaringan *mesh* saling terhubung satu sama lain. Tidak jauh berbeda dengan jaringan *ad hoc* dimana untuk membangun jaringan ini tidak memerlukan perangkat jaringan terpusat untuk mengorganisir seperti perangkat *router* atau *switch.* Bisa dibilang jaringan *ad hoc* dan jaringan *mesh* merupakan jaringan dengan sistem desentralisasi. Pada jaringan *mesh* semua *node* bisa bertindak sebagai *host* dan juga sebagai *router* untuk meneruskan paket data ke seluruh *node* yang terhubung agar jaringan bisa terbentuk secara terus-menerus.

Pada awalnya jaringan *mesh* digunakan untuk keperluan militer, tapi semenjak tahun 2000 jaringan ini sering digunakan keperluan internet antar rumah via WLAN. Sekarang jaringan *mesh* sering ditemui di peratalan industri, perusahaan, aplikasi bergerak pada kendaraan dan dimana menjadi dasar dari perangkat Zigbee untuk digunakan pada bagian sensornya. Jaringan *mesh* selalu menyediakan solusi untuk menambah jangkauan pengguna

radio melalui jangkauan radio tersebut, dengan perelatan minimum dan jaringan dengan performa tinggi.

Tren yang sedang berkembang sekarang adalah penggunaan jaringan *mesh* tanpa kabel atau *wireless mesh network* dimana jaringan *mesh* saling terhubung dengan menggunakan koneksi nirkabel entah itu menggunakan USB WiFi dongle atau modul perangkat *wireless* yang lain. Kelebihan dari jaringan *mesh* adalah fleksibel dimana ketika terjadi kesalahan seperti rute putus, maka jaringan *mesh* akan berusaha memperbaiki rute dengan cara mengalihkannya ke rute lain dan ketika akan dilakukan perluasan jaringan *mesh* tidak akan terjadi gangguan ke jaringan lainnya dan topologi *mesh* sendiri. Kelemahan dari jaringan *mesh* sendiri adalah dari segi pengadaannya yang berarti membutuhkan banyak sekali perangkat keras jaringan (misalnya : kabel jaringan, perangkat nirkabel, dll) dan untuk membangun topologi ini membutuhkan instalasi dan konfigurasi yang sulit karena topologi ini cukup rumit. Gambar 2.3 di bawah menunjukan salah satu ilustrasi dari jaringan *mesh* nirkabel dengan perangkat laptop dan telepon seluler [3].



Gambar 2.3 Ilustrasi jaringan *mesh*

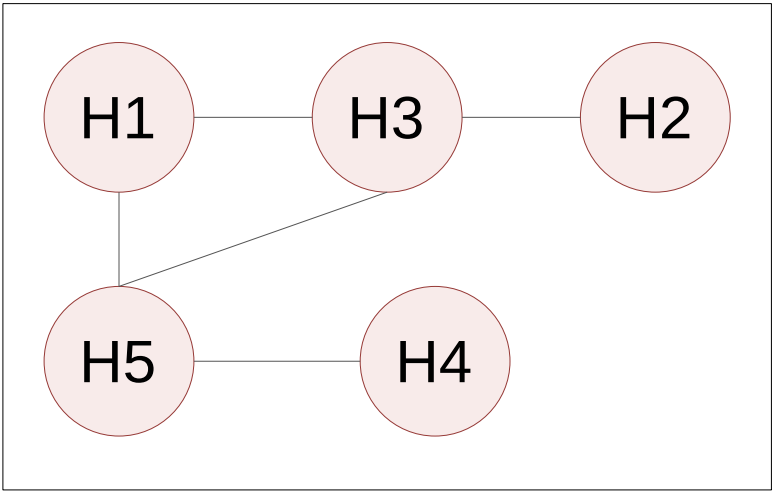
2.3 Destination Sequence Distance Vector Routing Protocol

Pada subbab ini akan dijelaskan lebih mendalam mengenai protokol routing dari algoritma Destination Sequence Distance Vector(DSDV).

2.3.1 Pengertian DSDV

Destination-Sequenced Distance Vector Protokol atau disingkat menjadi DSDV *Protokol* adalah pengembangan dari Routing Information Protokol (RIP) yang termasuk ke dalam *routing protokol* berbasis distance-vector yang dikhususkan untuk jaringan ad hoc. Tapi, RIP masih mempunyai kelemahan yaitu ketika terjadi perubahan topologi, kemungkinan link failures cukup besar. Agar bisa mengoptimalkan convergence dari RIP, triggered updates, split horizon, poison reverse dan mekanisme path hold-down, banyak solusi yang bisa dilakukan. Tapi ketika solusi itu dilakukan muncul masalah-masalah lain. Dengan menambahkan penomoran urutan, node yang bergerak atau yang disebut mobile node bisa membedakan informasi rute lama dan mana yang baru untuk menghindari formasi rute berputar atau routing loops.

Masing-masing router yang menggunakan protokol DSDV mempunyai tabel routing atau rute yang berisi daftar semua tujuan, jumlah metrik, next hop atau node tujuan selanjutnya agar sampai ke *node* tujuan dan nomor urutan yang dibuat oleh node tujuan. Jumlah metrik adalah jumlah lompatan ke node lain yang diperlukan untuk sampai ke *node* tujuan. Contoh jaringan ad hoc bisa dilihat pada Gambar 2.4. Gambar 2.4 menjelaskan ada 5 mobile ad hoc router yang membentuk topologi *mesh*. Sedangkan pada Tabel 2.1 menjelaskan tabel informasi rute pada router H1 dari jaringan mesh pada gambar 2.4.



Gambar 2.4 Contoh skema jaringan *mesh* dengan protokol DSDV

Tabel 2.1 Tabel *routing* dari *node* H1

Tujuan	Lompatan selanjutnya(next hop)	Metrik	No. Urutan
H1	H1	0	T001_H1
H2	H3	2	T001_H1
H3	H3	1	T001_H1
H4	H5	2	T001_H1
H5	H5	1	T001_H1

Setiap terjadi perubahan topologi, node akan melakukan penyebaran informasi rute ke semua node lain dimulai dari node tetangga dengan jumlah metrik paling sedikit yaitu satu untuk kemudian pesan informasi rute tersebut akan diteruskan ke node lain sampai semua node mendapat informasi tersebut dan ketika sebuah node mendapat lebih dari satu informasi rute, maka node yang menerima pesan-pesan informasi tersebut akan mencari informasi rute yang paling optimal dilihat dari jumlah metrik. Setiap router memperbarui rute tabel, maka nomor urutan pada

router tersebut akan bertambah. Nomor urutan pada tabel ini akan digunakan sebagai parameter apakah informasi rute sudah yang paling baru atau belum. Jadi, ketika mendapat informasi tabel baru dari router sebelah, maka kolom nomor urutan dari tabel informasi router yang menerima dan tabel informasi router yang dikirim oleh node tetangga akan dibandingkan mana yang lebih besar.

2.3.2 Permasalahan DSDV

Jika dilihat dari tujuan utama protokol ini tentunya dapat disimpulkan bahwa protokol DSDV berusaha memabangun sebuah jaringan *mesh* tanpa adanya putaran tak hingga atau *infinity looping.* Nyatanya muncul masalah-masalah baru dalam waktu bersamaan yang tidak bisa ditangani oleh protokol DSDV.

2.3.2.1 Damping Fluctuation

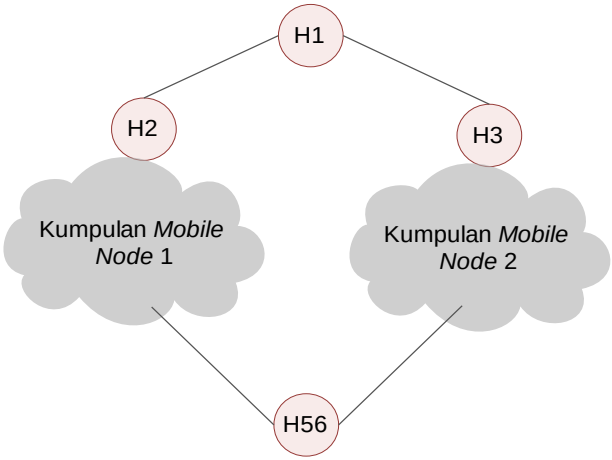
Masalah pertama adalah *Damping Fluctuation. Fluctuation* atau fluktuasi yang dimaksud disini muncul dikarenakan perihal berikut :

1. Rute baru akan selalu dipilih ketika *sequence number*-nya lebih baru dan rute yang lama akan dibuang.
2. Rute dengan *sequence number* yang sama tapi mempunyai *metric* yang lebih kecil akan dipilih dan rute yang sekarang akan dibuang.

Proses *broadcast* informasi *routing* pada *node* bergerak merupakan kejadian asinkronis dimana setelah proses *broadcast* berlangsung *node* bisa melanjutkan tugas yang lain atau sistem kerjanya sama dengan *thread*. Contoh kasus fluktuasi seperti yang akan dijelaskan di bawah pada gambar 2.5 di bawah.

Pada tersebut terdapat *node* H1, H2, H3, kumpulan *mobile node* 1, kumpulan *mobile node* 2 dan H56. Kumpulan *mobile node* adalah berarti ada suatu kumpulan *node* lain dengan jumlah

tertentu. Pada kasus ini H1 menyebarkan tabel *routing*-nya ke node H2 dan H3 yang lalu kemudian oleh node H2 diteruskan menuju ke kumpulan *mobile node* 1 dan H3 diteruskan ke kumpulan *mobile node* 2. Kita asumsikan semua *node* melakukan persebaran tabel *routing* dengan tenggang waktu 15 detik setiap ada perubahan. Lalu untuk menuju *node* H56, *node* H2 mempunyai rute dengan 16 lompatan sedangkan H3 untuk menuju *node* H56 mempunyai rute dengan 12 lompatan. Maka dengan segera node H56 akan menerima paket tersebut dan mengirim kembali informasi rute baru ke *node* H1 padahal *node* dari *node* H3 mempunyai rute yang lebih optimal tapi karena terlambat dating maka yang dipilih adalah rute dari *node* H2. yang dari *node* H2 karena ketidakteraturan yang disebabkan oleh 2 masalah utama yang telah disebutkan di paragraf pertama pada subbab ini.



Gambar 2.5 Contoh kasus rute fluktuatif

Untuk mengatasi hal tersebut, maka diperlukan sebuah mekanisme untuk menahan paket agar tidak segera disebarkan ke *node* tetangga terlebih dahulu. Alasan kenapa diperlukan waktu untuk menunggu karena kemungkinan paket informasi terbaru

akan datang dengan rute optimal. Untuk menghitung berapa lama waktu yang dibutuhkan untuk meunggu aket lain yang akan datang menggunakan *Average Settling Time*. *Average Settling Time* adalah rata-rata jarak waktu yang dibutuhkan oleh sebuah *node* ketika menerima paket informasi dari *node* lain. Rumus untuk mencari *Average Settling Time* sebagai berikut:

$$Average\ Settling\ Time_j = \sum x_j\ T_j\ /\ \sum x_j \text{.....(2)}$$

Average Settling Time untuk setiap tetangga pada suatu *node* berbeda memiliki nilai yang berbeda. x_j merupakan *metric* terhadap *node* j dan T_j merupakan *Last Settling Time* atau jarak waktu informasi terakhir didapat. Contoh tabel Settling Time bisa dilihat pada tabel 2.2 dibawah. Kemudian waktu tunda yang diperlukan bisa menggunakan rumus:

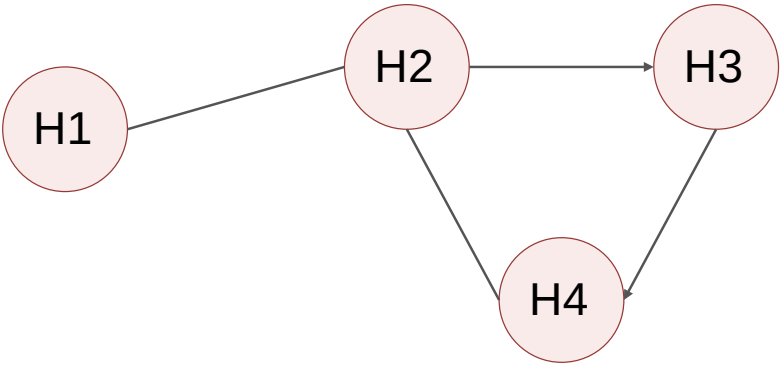
$$Delay_j = Average\ Settling\ Time_j\ X\ 2 \text{ .....(3)}$$

Tabel 2.2 Contoh tabel *Settling Time* pada node H1

Tujuan	Last Settling Time	Average Settling Time
H1	15	13
H2	13	11
H3	13	11
H56	6	3

2.3.2.2 Unidirectional Links

Dalam protokol DSDV, asumsi yang digunakan dalam semua link adalah *bi-directional* atau secara 2 arah. Padahal kenyataanya pada jaringan nirkabel terdapat koneksi asimetris atau *unidirectional.* Berikut adalah contoh kasus Undirectional Links bisa dilihat pada gambar 2.6



Gambar 2.6 Contoh kasus u*nidirectional links*

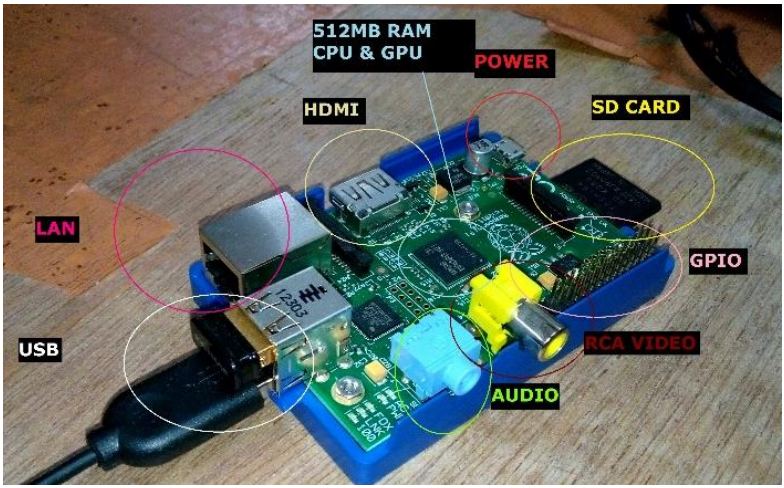
Pada gambar 2.6 terdapat 4 *node* yaitu H1, H2, H3 dan H4. Pada gambar tersebut H1 terhubung secara *bidirectional* dengan H2, H2 terhubung *bidirectional* dengan H4, H2 terhubung *unidirectional* ke arah H3 dan H3 terhubung secara *unidirectional* ke H4. Jika ada kasus dimana H1 mengirim informasi ke H3, maka jalur semestinya adalah melewati H2 lalu H3. Tapi ketika H3 ingin mengembalikan pesan balik ke H1 tidak bisa melewati H2 karena hubungan *unidirectional* yang mengarah ke H3. Maka dari itu H3 segera mengirim pesan tersebut melewati jalan lain yaitu H4 dan karena H4 terhubung secara *bidirectional* dengan H2 maka pesan dikembalikan ke H1 lewat H2 dari H4 [4].

2.4 Raspberry Pi

Raspberry Pi (juga dikenal sebaga RasPi) adalah sebuah komputer mini yang berbentuk *board* mempunyai ukuran seperti kartu kredit yang dikembangkan oleh yayasan Raspberry Pi di Inggris dengan maksud untuk memicu pengajaran ilmu komputer dasar di sekolah-sekolah dengan harga yang sangat terjangkau dan kemampuan yang cukup. Alat ini bisa dikembangkan untuk berbagai macam perangkat-perangkat elektronika yang berguna untuk kehidupan sehari-hari ataupun hanya sekedar untuk melakukan kegiatan komputasi sederhana.

Raspberry Pi ini mampu bekerja layaknya komputer pada umumnya dengan kemampuan untuk menjalankan sistem operasi seperti Raspbian, Pidora, New Out Of Box Software (NOOBS), RaspBMC dan lain-lain. Meskipun mempunyai hampir semua kemampuan yang dimiliki oleh komputer biasa, kemampuan komputasi Raspberry Pi tidak sehebat komputer biasa. Aplikasi-aplikasi open-source bisa dipasang ke dalam komputer mini tersebut seperti LibreOffice, multimedia (audio dan video), web browser atau programming. Dari perusahaan Raspberry Pi sendiri sekarang telah membuat 2 model Raspberry Pi yaitu model A dan model B dimana model B adalah pengembangan dari model A dari segi komponen dan kekuatan spesifikasi lebih hebat. Perbedaan nyata dari model A dan model B adalah model A tidak mempunyai tempat LAN atau port untuk kabel ethernet.

Raspberry Pi dapat menampilkan gambar ke TV/HDTV menggunakan koneksi High Definition Multimedia Interface (HDMI) ataupun televisi standar menggunakan kabel RJ45. Selain itu Raspberry Pi juga meyediakan port serial untuk RCA Video yaitu berupa lubang kuning dan disampingnya ada lubang untuk port audio yang berwarna biru. Selain itu Raspberry Pi model B juga menyediakan 2 slot Universal Serial Bus (USB) yang nanti bisa digunakan untuk keyboard USB, Flash Disk USB, mouse USB, WiFi dongle USB dll. Untuk harddisk dari Raspberry Pi sendiri menggunakan SD Card yang nanti dipasang di bagian bawah board. Adapun General-Purpose Input Output atau GPIO jika kita ingin membuat suatu perangkat yang membutuhkan pin generik pada sirkuit terpadu. Inilah kehebatan Raspberry Pi karena dengan board sederhana dan harga yang terjangkau bisa menampung semua kebutuhan-kebutuhan untuk membangun sebuah alat. Gambar 2.7 merupakan contoh Raspberry Pi tipe B [5].



Gambar 2.7 Contoh Raspberry Pi tipe B

2.5 Sistem Operasi Raspbian

Raspbian adalah salah satu sistem operasi yang bisa digunakan untuk menjalankan perangkat Raspberry Pi.Raspbian merupakan distro turunan dari sistem operasi Linux Debian yang memang dikhususkan dan sudah dilakukan *porting* khusus agar bisa sejalan dengan *board* Raspberry Pi. Saat ini yang digunakan oleh sistem operasi Raspbian adalah Linux Debian versi Wheezy.

Kelebihan dari sistem operasi ini yaitu paket-paket yang disediakan selalu diperbarui oleh pengembang dan paket-paket yang disediakan mudah dipasang di perangkat Raspberry Pi.

Karena kemudahannya dan kebanyakan paket-paket yang disediakan cocok untuk pengerjaan riset Tugas Akhir ini, maka penulis memutuskan untuk menggunakan sistem operasi tersebut pada masing-masing *node* yang terpasang dalam jaringan *mesh.* Sistem operasi Raspbian dengan versi rilis 26 Juli 2013 yang akan digunakan pada riset Tugas Akhir ini [6]

BAB III
PERANCANGAN PERANGKAT LUNAK

Untuk membangun sebuah perangkat lunak yang tepat sesuai sasaran maka diperlukan persiapan yang sangat matang. Maka dari itu perlu dibuat perencanaan teknis mulai dari mikro sampai makro mengenai aplikasi yang akan dibuat. Bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam riset Tugas Akhir ini. Perancangan yang dimaksud meliputi deskripsi umum aplikasi, proses perancangan, alur kerja dan implementasinya.

3.1 Deskripsi Umum Sistem

Sistem yang akan dibangun adalah sistem yang terdiri dari beberapa *node* dari perangkat Raspberry Pi dimana *node-node* tersebut akan membentuk suatu jaringan *mesh. Node-node* tersebut bisa diklasifikasikan sebagai berikut :

1. *Node* pemantau yang berperan sebagai penangkap gambar dengan menggunakan kamera.
2. *Node* penampil yang berperan sebagai penerima hasil akhir dan menampilkan gambar dari hasil tangkapan *node* pemantau.
3. *Node* penghubung berperan sebagai *node* perantara agar gambar dari *node* pemantau bisa sampai ke *node* penampil.

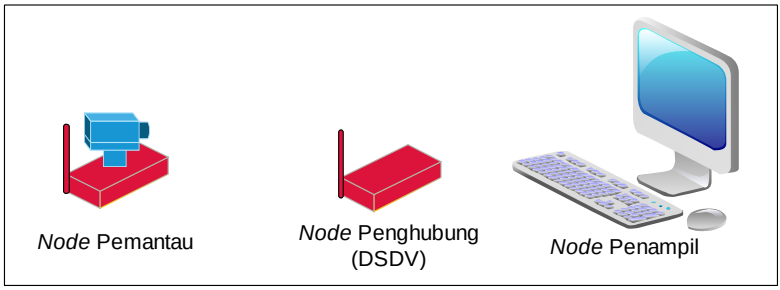
Apapun jenis dari *node* tersebut, masing-masing *node* harus membuat protokol *routing* agar jaringan *mesh* yang dibangun bisa lebih optimal dari segi pembangunan jaringan, keefektifan dan efisiensi jaringan.

Agar bisa menangkap gambar menggunakan Raspberry Pi diperlukan perangkat tambahan yaitu modul kamera yang nantinya akan dipasang pada *node* kamera. Agar bisa mengakses modul

kamera tersebut penulis menggunakan pustaka dari bahasa pemrograman Python yang telah disediakan oleh pihak Raspberry Pi yaitu PiCamera. Dalam pengerjaannya nanti penulis menggunkan *script* Python untuk mengambil gambar dan mengirim dan menerima gambar.

Selain itu, perangkat-perangkat lain yang diperlukan dalam pengerjaan Tugas Akhir ini adalah power adaptor dengan output 5V 1,5A. Untuk membuat jaringan nirkabel sendiri digunakan modul USB WiFi yang kompatibel dengan perangkat mini komputer Raspberry Pi, yaitu salah satunya modul WiFi dengan chipset RTL8188CU. Untuk menangkap citra digunakan perangkat kamera yang dikeluarkan oleh element14 khusus untuk digunakan pada Raspberry Pi.

Gambar 3.1 menjelaskan gambaran *node* pemantau, *node* penghubung dan *node* penampil dari sistem yang akan dibangun oleh penulis.

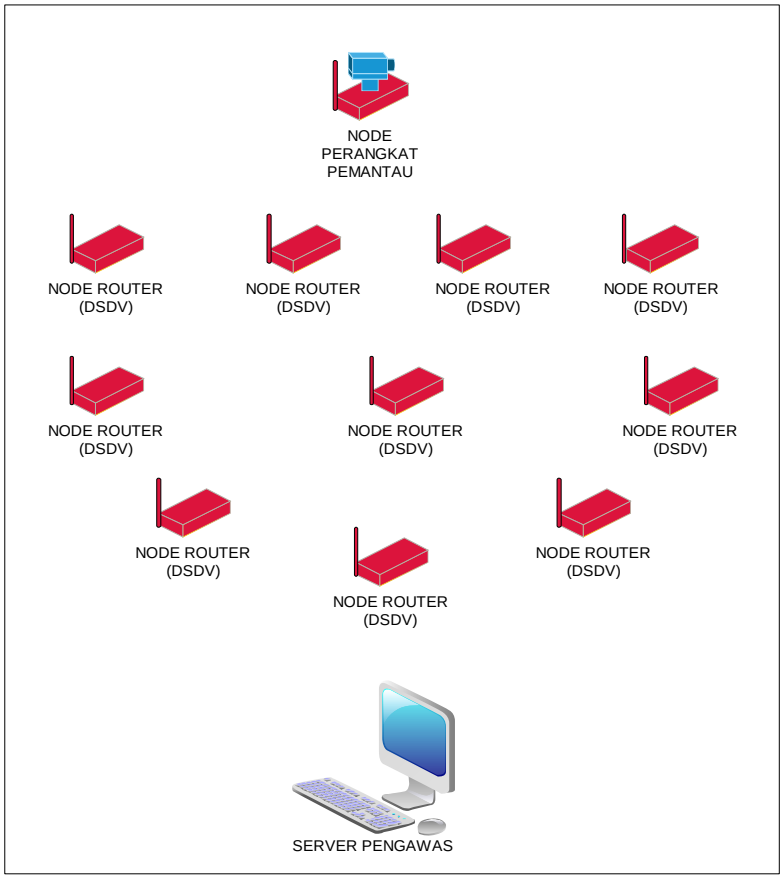


Gambar 3.1 Jenis-jenis *node* dalam jaringan *mesh*

3.2 Arsitektur Umum Sistem

Dalam sistem yang dibangun nanti akan terdapat beberapa *node* berupa Raspberry Pi dan *node* pemantau berupa Raspberry Pi yang telah dipasang kamera dari modul kamera yang disediakan oleh perusahaan Raspberry Pi. Semua *Node* ini akan membentuk suatu jaringan nirkabel *mesh*. Untuk memantau hasil gambar yang

diambil *node* kamera akan digunakan seperangkat komputer yang berfungsi sebagai pusat pemantau.



Gambar 3.2 Arsitektur jaringan *mesh* dengan protokol DSDV yang akan dibangun

Berdasarkan pada Gambar 3.2, alur kerja perangkat dijabarkan sebagai berikut:

1. Masing-masing *node* pertama-tama akan saling menjalankan program yang berjalan secara *daemon* untuk bisa membentuk jaringan *mesh* sesuai dengan *routing protokol* DSDV. melakukan Node pemantau menangkap gambar dari Raspberry Pi yang telah dipasang modul kamera.
2. Setelah jaringan telah terbentuk secara dinamis, maka saatnya *node* pemantau mengirim gambar ke *node* penampil melalui protokol UDP.
3. Sessaat sebelum dikirim melalui protokol UDP, setiap gambar akan diubah ke dalam bentuk paket *string* dengan menggunakan *script* Python. Paket data yang terkirim akan menuju ke *node* penampil sesuai rute dinamis yang telah dibentuk oleh protokol DSDV.
4. *Node* penampil akan menerima paket data *string* yang kemudian akan diubah lagi menjadi gambar dan akan ditampilkan di layar monitor dari *node* penampil menggunakan antarmuka aplikasi.

3.3 Perancangan Perangkat Keras

Perancangan perangkat keras pada penelitian ini menjelaskan tentang perangkat keras apa saja yang akan digunakan untuk mengimplementasikan Tugas Akhir ini. Tugas Akhir ini membutuhkan perangkat keras sebagai berikut:

1. Mini komputer Raspberry Pi model B.
2. 150Mbps *wireless* 802.11b/g/n Edimax nano USB *adapter*.
3. Modul kamera Element14 Raspberry Pi.
4. SDHC *card* Sandisk Ultra 8GB Class 10.
5. *Power* adaptor dengan *output* 5V 1,5A.
6. Kabel LAN.
7. Keyboard USB.
8. Monitor komputer.
9. Kabel HDMI *to* VGA.

Gambar 3.3 Perangkat keras untuk *node* pemantau

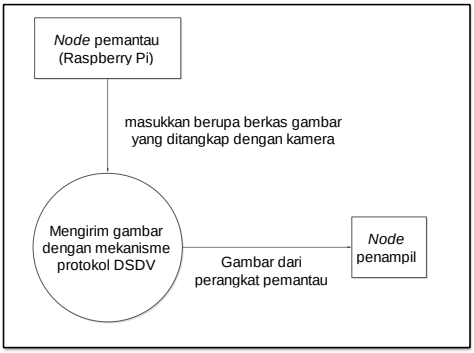
Pada Gambar 3.3 ditunjukkan sejumlah perangkat keras yang dibutuhkan untuk membangun sebuah *node* pemantau sedangkan untuk membangun *node* penghubung tidak dibutuhkan kamera dan untuk *node* penampil sama seperti *node* penghubung hanya menggunakan tambahan monitor untuk menampilkan hasil *streaming* . Semua slot untuk masing-masing perangkat terbut sudah disediakan oleh *board* Raspberry Pi tinggal untuk memasang saja sesuai petunjuk pada gambar 2.7.

Board Raspberry Pi berperan sebagai perangkat utama untuk memasang perangkat lainnya. SDHC *card* digunakan untuk menyimpan data dan sistem operasi pada Raspberry Pi. Nano USB *adapter* nirkabel diperlukan untuk membuat jaringan *ad-hoc*. Pada

perangkat ini penulis gunakan wifi *adapter* yang memiliki chipset RTL8188CU dikarenakan chipset tersebut kompatibel dengan Raspberry Pi. Umumnya perangkat Raspberry Pi model B yang digunakan menggunakan listrik 0,7A – 1,2A. Tapi melihat spesifikasi kebutuhan alat yang akan dibangun akhirnya penulis menggunakan adaptor dengan *output* 5V 1,5A

3.4 Perancangan Diagram Alir Data tingkat 0

Pada diagram alir sistem *picture streaming* pada jaringan *mesh* dengan menggunakan protokol DSDV menggambarkan urutan fungsionalitas sistem secara keseluruhan. Diagram alir sistem dapat dilihat pada Gambar 3.4 di bawah. Sistem diawali dengan input dari kamera yang ada pada Raspberry Pi. Kemudian input dari kamera tersebut diolah dalam bentuk gambar dengan format .jpg dengan resolusi yang ditentukan. Gambar tersebut akan dikirimkan kepada *server* sehingga *server* dapat menampilkan gambar secara *real-time*.



Gambar 3.4 Diagram alir data tingkat 0 dari *Picture Streaming* dari jaringan *mesh* dengan protokol DSDV

3.5 Diagram Alir Aplikasi Sistem

Ada 3 alir proses yang berjalan dalam sistem yang dibuat oleh penulis dan akan dijabarkan disini agar memudahkan

pemahaman proses secara umum atau garis besar. Proses-proses tersebut akan digambarkan dalam bentuk diagram alir. Bagian pertama adalah diagram alir protokol DSDV, diagram alir pada sisi pemantau dan diagram alir pada penampil.

3.5.1 Diagram *Router* Alir protokol DSDV

Dalam diagram alir protokol DSDV terdapat 4 proses yang berjalan secara bersamaan, maka dari itu penulis akan memisah menjadi 4 diagram yang pertama yaitu diagram alir proses untuk mencari *node* tetangga, yang kedua adalah alir proses untuk menunggu(*listen*) pesan baik *advertised table* dari *node* tetangga *node* sekitar, yang ketiga adalah deteksi *Broken Link* dan yang terakhir adalah menyebar pesan *full dump* atau seluruh tabel *routing* yang dimiliki saat itu menjadi *advertised table* secara periodik. Empat proses tersebut akan dijalankan dengan menggunakan *thread* sehingga bisa berjalan secara hampir bersamaan.

Ada sedikit perubahan yang dilakukan dalam penggunaan tabel *routing.* Jika pada jurnal yang diacu menggunakan tambahan kolom "*Time Install*", maka penulis memutuskan untuk tidak menambahkan kolom tersebut karena dianggap tidak ada fungsinya dalam algoritma DSDV. Tujuan dasar dari kolom "*Time Install"* adalah untuk membandingkan rute mana yang paling baru padahal kolom "*Sequence Number*" sudah cukup untuk mengetahui rute mana yang paling baru. Lalu perubahan lain yang dilakukan oleh penulis adalah meniadakan proses penyebaran pesan *incremental.* Pesan *incremental* sama seperti *full dump* akan tetapi yang disebar ke *node* tetangga hanya baris pada tabel *routing* yang mengalami perubahan saja. Pertimbangan penulis menghilangkan proses ini dikarenakan proses penyebaran *full dump* yang cukup cepat sehingga tidak diperlukan.

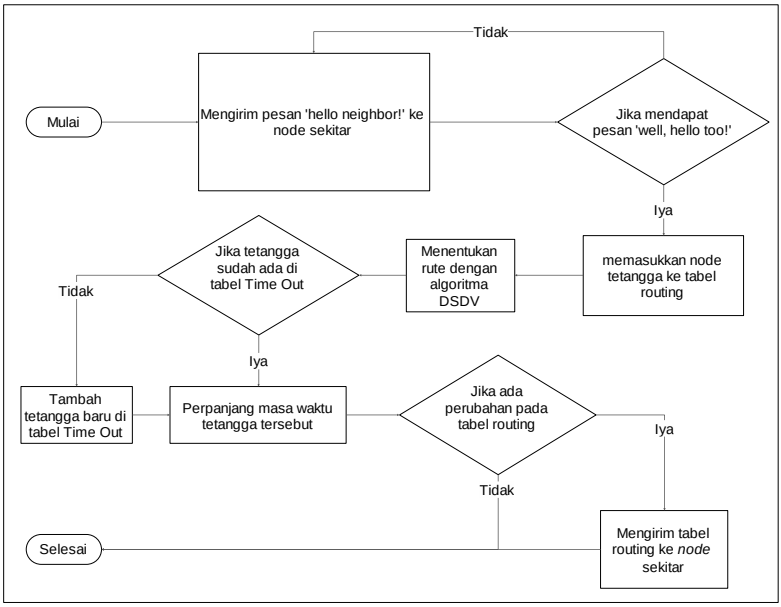
Yang perlu digaris bawahi di sini adalah untuk nilai *sequence number* untuk masing-masing *node* tujuan pada tabel *routing* di setiap yang menentukan adalah masing-masing dari *node* tujuan itu

sendiri. Tiap *node* tujuan akan menyebarkan nilai *sequence number* mereka untuk kemudian akan diserahkan ke *node-node* tetangganya yang nanti oleh *node-node* tersebut akan disebarkan ke seluruh *node* dalam jaringan secara rekursif. Catatan penting di sini adalah untuk setiap dilakukan penambahan *sequence number* pada tiap *node* tujuan, *sequence number* akan ditambah dengan angka genap atau angka *default* yaitu 2 dan ketika salah satu *node* mendeteksi tetangga tujuannya yang hilang maka *node* itu sendiri tersebut yang menambah *sequence number* dengan angka ganjil atau angka *default* yaitu 1.

- ***Thread mencari node tetangga***

Sistem kerja dari protokol routing DSDV harus digunakan oleh masing-masing node untuk kemudian nantinya akan membentuk suatu jaringan mesh. Masing-masing node akan mengirim pesan berisi "hello neighbor!" dengan metode broadcast ke siapapun yang ada di sekitarnya. Kemudian ketika ada node yang menerima pesan "hello neighbor!" akan merespon dengan menjawab "well, hello too!" kemudian hubungan antar tetangga antar 2 node akan terbentuk. Segera setelah menerima pesan "well, hello too", node pengirim pesan akan menyimpan data dari node tetangga baru tersebut kemudian akan dilakukan pemilihan rute baru menggunakan algoritma DSDV dan hasilnya akan disimpan di table routing untuk kemudian table routing tersebut akan dirubah menjadi tabel advertised yang akan dikirim ke node sekitarnya. Setelah itu akan dilakukan pembaruan terhadap tabel Time Out yang nantinya akan digunakan sebagai parameter sebuah tetangga sudah hilang atau tidak (broken link). karena mendapat paket baru maka. Setelah semua proses itu selesai node akan kembali melakukan pengiriman pesan "hello neighbor!" ke sekitarnya lagi karena proses ini harus berlangsung selamanya atau infinity looping karena perangkat ini bersifat dinamis dan bisa berubah kapanpun sesuai keadaan yang ada sekarang missal salah satu tetangga ada yang mati maka semua data akan diperbarui lagi di

masing-masing node seperti yang digambarkan pada Gambar 3.5 dibawah.

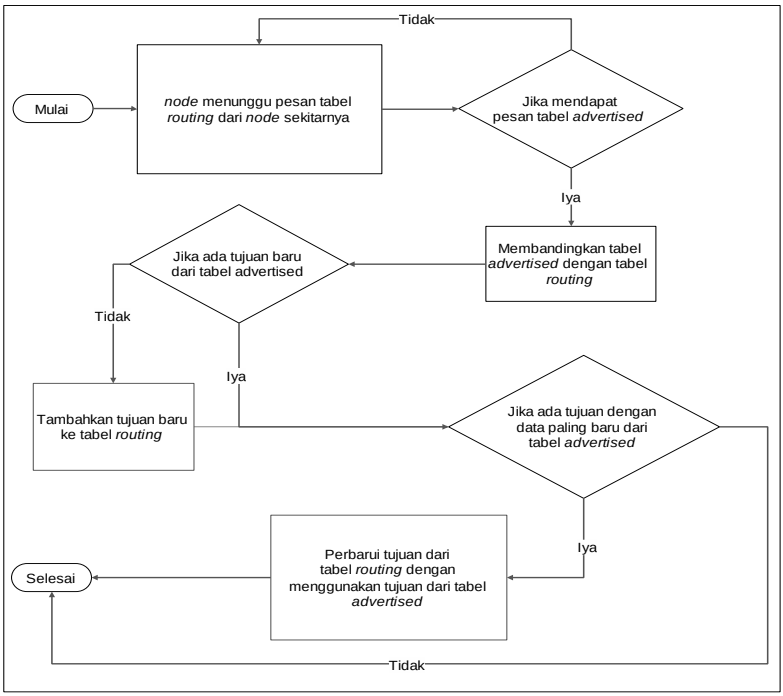


Gambar 3.5 Diagram alir untuk mencari *node* tetangga

- ***Thread untuk membandingkan tabel advertised dengan tabel routing***

Pada gambar 3.6 di bawah dijelaskan skema mengenai proses lain yang berjalan pada protokol *routing* DSDV ketika menerima tabel *advertised* dari node tetangga lain. Masing-masing *node* melakukan *listen* menunggu pesan yang berisi tabel *advertised* dari para *node* tetangga, kemudian setelah itu dilakukan perbandingan dari tabel *routing* yang sudah ada dalam masing-masing *node* dengan tabel *advertised* yang masuk ke dalamnya. Tiap baris yang berisi data identifikasi tiap *node* tujuan lain akan dibandingkan satu sama lain. Ketika terdapat data yang baru salah satu baris dari tabel *advertised* maka baris dengan tujuan yang

sama akan segera diperbarui dengan baris yang ada di tabel *advertised* dan jika ada baris baru dengan *node* tujuan baru dari tabel *advertised* maka akan segera dimasukkan ke tabel *routing.*

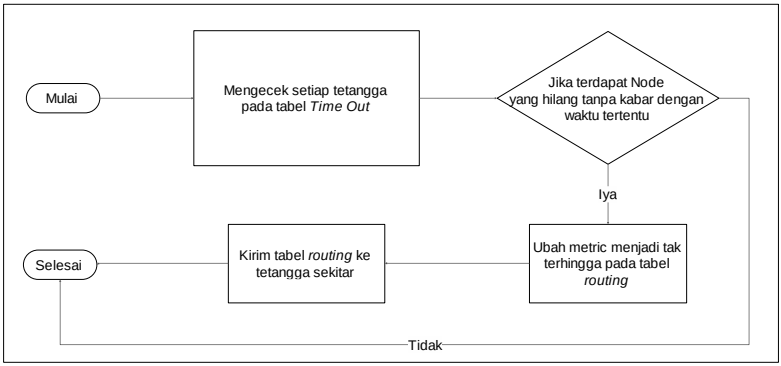


Gambar 3.6 Diagram alir cara membandingkan tabel yang disebar oleh tetangga dengan tabel *routing* oleh tiap *node*

- ***Thread untuk mendeteksi broken link***

Jika 2 proses sebelumnya membahas tentang bagaimana cara mengenali tetangga di sekitarnya, maka proses terakhir adalah menentukan apakah sebuah *node* tetangga sudah putus hubungan atau *broken link.* Proses ini berjalan dengan cara melakukan pengecekan terus menerus pada tabel *Time Out.* Jika pada saat pengecekan ternyata waktu sekarang melebihi waktu *time out* pada

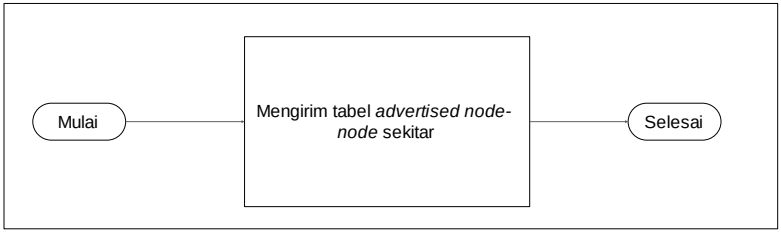
suatu tetangga, maka tetangga tersebut akan dianggap hilang seperti yang akan dijelaskan pada gambar 3.7 dibawah. Jika salah satu *node* tetangga hilang maka *sequence number* dari *node* tujuan tersebut akan ditambah sejumlah nilai ganjil.



Gambar 3.7 Diagram alir proses pengecekan *broken link*

- ***Thread* untuk menyebar pesan *full dump***

Pada proses ini dilakukan proses penyebaran tabel *routing* tiap *node* untuk disebar ke *node-node* sekitar dalam bentuk tabel *advertised.* Proses ini dilakukan secara periodik dengan jarak waktu tertentu, waktu *default* yang digunakan adalah 15 detik. Gambar 3.8 di bawah menjelaskan skema proses pengiriman pesan *full dump* secara periodik.



Gambar 3.8 Diagram alir proses pengiriman pesan *full dump*

3.5.2 Diagram Alir Aplikasi *Node* Pemantau

Pada sisi *node* pemantau terdapat banyak proses yang berjalan secara bersamaan yang akan dijalankan menggunakan *thread* agar bisa berjalan secara bersamaan. *Thread* pertama melakukan proses penangkapan gambar dari modul kamera Raspberry Pi secara terus menerus. *Thread* kedua melakukan proses *listen* menunggu pesan "I'm a viewer" beserta alamat IP dari *node* yang mengirim pesan tersebut. *Thread* ketiga adalah pengiriman gambar yang diambil oleh *thread* pertama ke alamat IP *node-node* penampil.

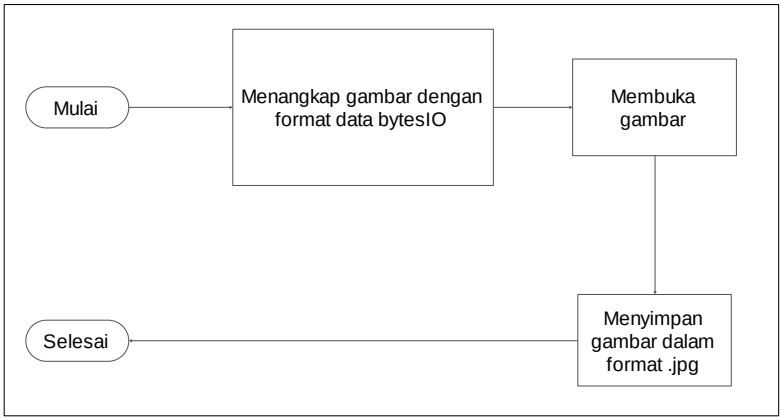
Pada sisi *server*, dalam hal ini adalah *node* pemantau merupakan perangkat mini komputer Raspberry Pi dengan modul kamera. *Node* ini berfungsi untuk memantau keadaan jalan raya dengan cara menangkap gambar secara terus menerus. Setelah gambar ditangkap maka akan diolah untuk disimpan dalam bentuk jpg. Kemudian gambar yang sudah disimpan akan dikonversi ke dalam bentuk *string* agar dapat dikirimkan melalui protokol UDP dalam jaringan *mesh*. Data yang akan dikirim pada klien akan berbentuk objek.

Pada sisi *server* terdapat tiga *thread*, yaitu *thread* untuk menagkap gambar, *thread* untuk mengirim pesan untuk memberitahu alamat IP *server*, *thread* untuk menerima pesan dari *viewer* dan *thread* untuk menangani setiap klien yang terhubung pada *server*. Tiga *thread* tersebut akan dijelaskan lebih dalam pada diagram-diagram alir berikut:

- ***Thread untuk menangkap gamba*r**

Pada Gambar 3.9 di bawah dijelaskan salah satu proses lain menggunakan *Thread* yang berjalan secara bersamaan yaitu proses untuk menangkap gambar yang dilakukan oleh modul kamera dari Raspberry Pi. Pertama modul kamera menangkap gambar dari modul kamera Raspberry Pi dengan hasil format bytesIO lalu dikonversi menjadi berkas dengan format .jpg. Proses ini

berlangsung terus menerus karena akan berlangsung secara *real time* agar *node* penampil bisa menampilkan hasil kiriman gambar dari *node* pemantau secara terus menerus juga. Untuk bisa disinkronkan dengan *thread* pengiriman gambar, *thread* ini membutuhkan *delay* untuk setiap proses yang berhasil dijalankan. Hal ini dilakukan karena untuk mengambil sebuah *frame* dibutuhkan waktu sepersekian detik dan ketika *thread* pengiriman gambar mengirim berkas gambar yang belum sempurna maka *frame* yang dikirim akan *corrupt* atau tidak bisa dibaca.



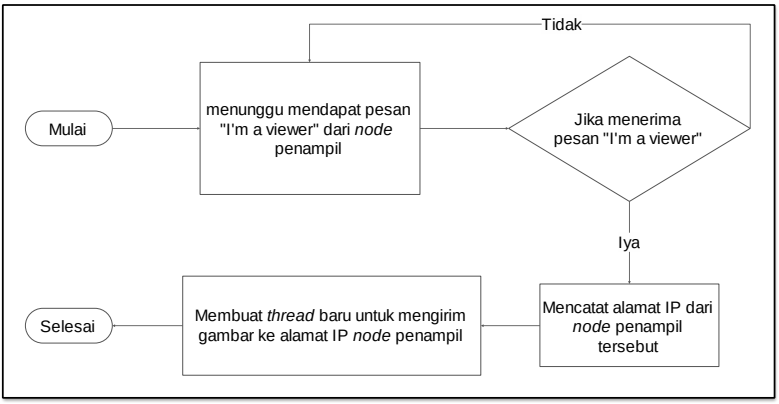
Gambar 3.9 Diagram alir fungsi menangkap gambar

- ***Thread* untuk mengirim gambar ke *node* penampil**

Pada Gambar 3.10 di bawah menjelaskan skema proses bagaimana proses pengiriman gambar ke *node* penampil. Dimulai dari melakukan pekerjaan untuk mendengarkan pesan identifikasi dari *node* penampil sehingga *node* kamera bisa tahu alamat IPV4 dari *node* penampil tersebut. Setelah itu IPV4 dari *node* penampil itu akan dicatat dan kemudian *node* pemantau segera mengirim gambar yang merupakan keluaran dari *thread* penangkap gambar dan dikirim ke alamat IPV4 *node* penampil secara terus menerus.

Agar proses ini bisa berlangsung diperlukan waktu *delay* untuk bisa disinkronkan dengan *thread* penangkapan gambar.

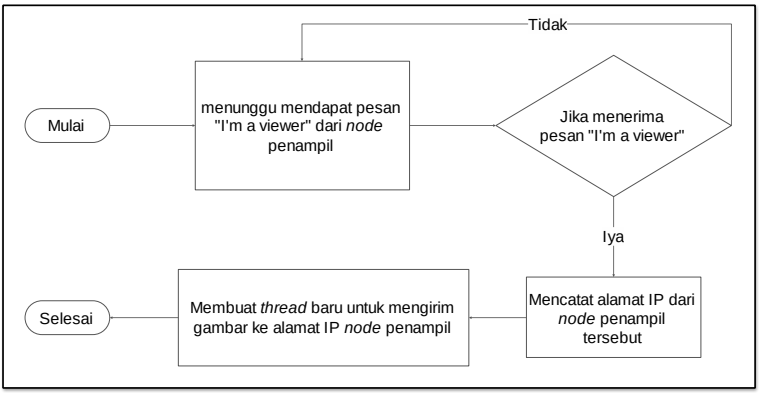
.



Gambar 3.10 Diagram alir mengirim gambar

- ***Thread untuk mengirim gambar***

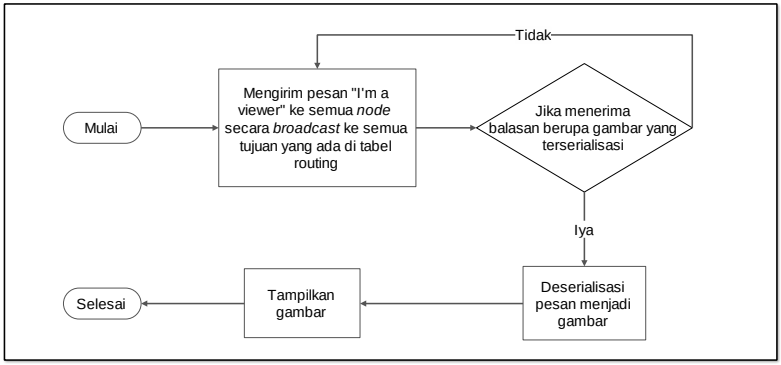
Pada Gambar 3.11 adalah diagram alir *thread* untuk mengirim gambar ke *viewer* yang terhubung. Untuk setiap *viewer* yang terhubung nantinya akan dilayani oleh satu *thread* untuk mengirim gambar secara *realtime*.



Gambar 3.11 Diagram alir proses pengiriman gambar

3.5.3 Diagram Alir Aplikasi *Node* Penampil

Dari sisi *node* penampil terjadi proses pengiriman pesan "I'm a viewer" secara *broadcast* ke semua *node* tetangga. Ini ditujukan agar *node* pemantau bisa tahu *node* mana yang akan dikirimi kumpulan gambar yang sedang dia tangkap. Setelah pesan "I'm a viewer" sampai, *node* pemantau akan segera mengirim pesan dalam bentuk *string* dari gambar yang terserialisasi yang kemudian akan diterima oleh *node* penampil dan akan dilakukan deserialisasi terhadap pesan tersebut agar bisa menjadi berkas dalam bentuk gambar lagi. Setelah selesai maka, antar muka dalam *node* penampil akan menampilkan gambar tersebut secara terus menerus seperti yang dijelaskan pada gambar 3.8 di bawah.

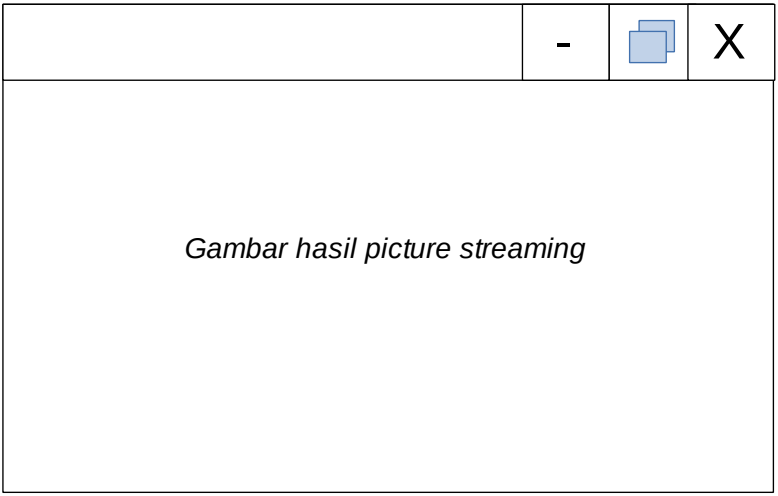


Gambar 3.12 Diagram alir pada *node* penampil

3.6 Rancangan Antar Muka Aplikasi

Agar memudahkan pengguna untuk melihat hasil *picture streaming* yang dilakukan oleh *node* penampil dengan *node* pemantau, penulis membuat antar muka sederhana. Proses penampilan gambar terjadi dengan antar muka terus mendeteksi *folder* yang dikhususkan untuk menerima gambar hasil *stream*. Antar muka akan menampilkan berkas gambar terbaru. Program antar muka itu akan dibuat dengan menggunakan Bahasa Python

dan modul Tkinter. Penulis memilih modul ini dikarenakan kemudahan dan kesederhanaan dalam pemakaiannya.. Pada Gambar 3.12 di bawah merupakan rancangan antar muka untuk menampilkan gambar yang dikirim oleh klien dalam jaringan *wireless mesh*.



Gambar 3.13 Rancang antar muka untuk menampilkan gambar hasil *picture streaming*

BAB IV
IMPLEMENTASI

Bab IV menerangkan segala hal tentang bentuk implementasi dari tugas akhir yang akan dilakukan oleh penulis. Pada bab ini akan dibahas mengenai implementasi yang dilakukan oleh penulis berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung pengembangan sistem sebagai berikut.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan riset jaringan *mesh* berupa *Netbook* dan mini komputer Raspberry Pi. Spesifikasi dari perangkat-perangkat tersebut adalah sebagai berikut:

Tabel 4.1 Spesifikasi perangkat komputasi

Perangkat	Spesifikasi
Netbook Acer Aspire One A110x	Intel® Atom™ Processor N270 512K Cache, 1.60 GHz, 533 MHz FSB
Raspberry Pi model B versi UK	700 MHz ARM1176JZF-S core dan RAM 512 MB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

- Linux Mint 16 “Petra” 32 bit sebagai sistem operasi pada netbook.

- Sistem operasi Raspbian Wheezy dengan tanggal rilis 26 Juli 2013 yang dipasang pada *node* penghubung dan *node* pemantau.
- Bahasa pemrogaman Python untuk mengimplementasikan program pembangun jaringan *mesh* dan membuat aplikasi *picture streaming*.
- PuTTY sebagai aplikasi untuk *remote console* terhadap Raspberry Pi.
- Edrawmax untuk merancang diagram alir data dan ilustrasi jaringan mesh.

4.2 Implementasi Perangkat Keras

Setelah melakukan penelaahan dengan berbagai macam perangkat keras, penulis akhirnya menentukan perangkat yang cocok digunakan dalam pembangunan sistem. Berikut adalah daftar perangkat keras yang akan digunakan:

Tabel 4.2 Spesifikasi perangkat yang dibutuhkan

Perangkat	Jumlah
Netbook Acer Aspire One A110x	1
Raspberry Pi Model B versi UK	5
WiFi dongle Edimax EW-7811un 150Mbps	5
modul kamera Element14 Raspberry Pi	1
SDHC *card* Sandisk Ultra 8GB Class 1	5
power adaptor dengan *output* 5V 1,5A	5
Kabel LAN ketogori 5	1
Keyboard USB	1
Kabel *converter* HDMI *to* VGA	1
Monitor	1

Untuk implementasi dari tiap-tiap perangkat keras diatas dapat dilihat dalam subbab di bawah.

- ***Implementasi Node Pemantau dan Node Penghubung***

Untuk *node* pemantau penulis menggunakan perangkat Raspberry Pi model B versi UK. Masing-masing *node* ini akan dipasang *Power supply* adaptor dengan tegangan *output* 5V 1,5A. Untuk terhubung atau membuat jaringan Raspberry Pi membutuhkan perangkat tambahan yaitu WiFi dongle Edimax EW-7811un 150Mbps *wireless* 802.11b/g/n nano USB. Sedangkan untuk pemantauan dengan menangkap citra dengan kapabilitas 5megapiksel menggunakan modul kamera yang diciptakan oleh element14. Modul ini nanti akan dipasang pada konektor CSI.

Sedangkan untuk *node* penghubung semua perangkat yang digunakan sama seperti *node* pemantau hanya saja tanpa modul kamera dan hanya berfungsi sebagai *router*.

Yang perlu diingat ketika memasang modul kamera pada Raspberry Pi adalah hati-hati karena perangkat ini cukup sensistif. Jika perangkat ini terkena listrik statis maka akan mempercepat kerusakan.

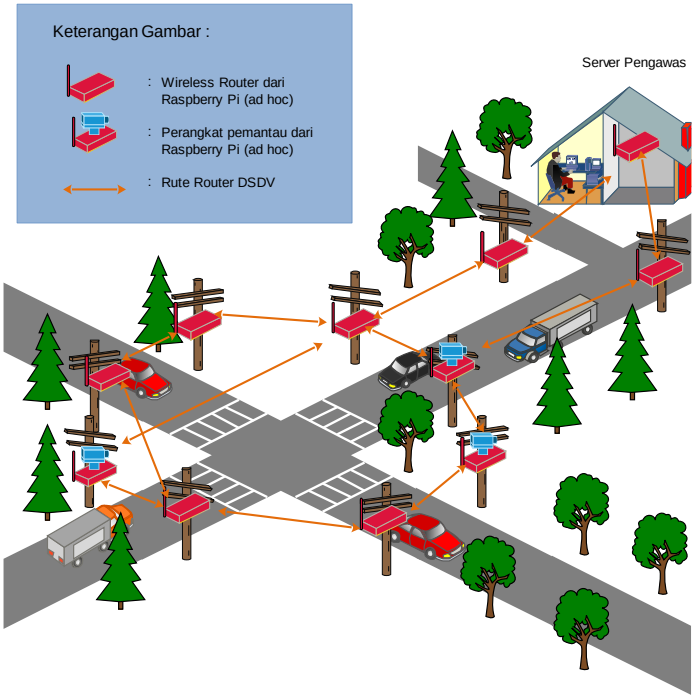
Tabel 4.3 Daftar perangkat yang dibutuhkan untuk *node* pemantau dan penampil

	Node Pemantau	*Node* Penghubung
Perangkat	Raspberry Pi Rev. B versi UK	Raspberry Pi Rev. B versi UK
Power Supply	Power adaptor 5V 1,5A	Power adaptor 5V 1,5A
WiFi adapter	Edimax EW-7811un	Edimax EW-7811un
Modul Kamera	element14 kamera	tidak ada

Pada Gambar 4.1 dibawah merupakan purwarupa dari perangkat pemantau dan penghubung dari Raspberry Pi yang sudah dibungkus dengan case khusus Raspberry Pi untuk perlindungan. Terlihat di *node* pemantau terdapat modul kamera yang berfungsi untuk menangkap gambar sedangkat *node* penghubung tidak ada.

Gambar 4.1 Purwarupa *node* pemantau dan penghubung

Pada Gambar 4.2 dibawah menjelaskan implementasi letak tiap *node* yang akan direncanakan. *Node-node* ini akan diletakkan di tempat yang dekat dengan sumber listrik. *Node* pemantau diletakkan pada tempat-tempat yang strategis untuk melakukan pemantauan. Sedangkan *node* penghubung diletakkan pada area-area yang akan menghubungkan *node* pemantau dan *node* penampil.



Gambar 4.2 Ilustrasi implementasi CCTV di atas jaringan *mesh* pada jalan raya

4.3 Implementasi Perangkat Lunak

Pada subbab ini akan menjelaskan lebih detail tentang implementasi perangkat lunak pada Netbook yang berfungsi sebagai node penampil dan Raspberry Pi sebagai node penghubung dan pemantau.

Di sini juga akan dibahas mengenai pemasangan aplikasi pada sistem operasi Raspbian di node pemantau dan node penampil. Untuk pemasangan sendiri dibutuhkan aplikasi dengan banyak proses yang berjalan secara bersamaan agar aplikasi bisa berjalan dengan semestinya. Untuk mengimplementasikan perangkat lunak pada Rapsberry Pi terdapat beberapa hal penting

yang harus dilakukan, antara lain memasang sistem operasi Raspbian Wheezy, menginisialisasi perangkat kamera pada mini komputer Raspberry Pi, mengkonfigurasi WiFi, memasang modul Python-PiCamera dan memasang Babel daemon dan mengkonfigurasinya. Hal-hal tersebut dapat diketahui lebih lanjut pada lampiran.

4.3.1 Implementasi *Router* protokol DSDV pada Tiap *Node*

Di sini akan dibahas tentang implementasi dari pembangunan rute dari tiap *node* agar tiap *node* bisa membangun jaringan *mesh.* Maka dari itu dibutuhkan *script* khusus yang terdiri dari beberapa *thread* yang mempunyai tugas khusus yang berbeda untuk bisa membangun jaringan *mesh. Script* ini akan dijalankan oleh semua *node* baik itu *node* pemantau, *node* penampil ataupun *node* penghubung. Pada *pseudocode* 4.1 di bawah terdapat 3 *thread* untuk mencari tetangga dan membandingkan tabel. Semua implementasi tersebut akan ditampilkan di bawah:

- ***Implementasi Thread untuk mencari tetangga***

Thread ini adalah mempunyai fungsi penting untuk mencari tetangga dengan nilai metrik 1 seperti yang dijelaskan pada *pseudocode* Gambar 4.3 di bawah. Format pesan yang digunakan untuk mengirim adalah dalam bentuk *list* berisi 2 variabel dimana indeks pertama berisi variable *string* bernilai “hello neighboor!”, lalu variable kedua adalah *list* berisi 2 variabel lagi yaitu, IPV4 saat ini dan *Sequence Number* dari *node* sekarang pada tabel *routing.*

```
# RT = RoutingTable
def lookAround(runEvent):
  string1 = "hello neighboor!"
  string2 = [currentIPV6, sequnce_number_of_this_node]
  data = (string1, string2)

  while runEvent.is_set():
```

```
    multicast(data, IP_Multicast)
    delay()

def listen(runEvent):
  while runEvent.is_set():
    data = data_received_from_ip_multicast()
            if data[1][1] not currentIPV6 then
     if data[0] = "hello neighboor!" then
                        string1 = "well, hello too!"
                        sendBackTo(string1, )
              else if data[0] = "well, hello too!" then
       if data[1] not in RT then
                addRoute(data[1])
                addTimeOutTable(data[1])
                        addSettlingTime(data[1], now)
                multicastAdvertisedTable()
      else if data[1] in RT then
  if data[1][1] >= RT[data[1]['sequence_number']] then
                RT[data[1]]['sequence_number'] = data[1][1]
                if RT[data[1]]['metric'] != 1 then
                 RT[data[1]]['metric'] = 1
                        addSettlingTime(data[1], now)
                updateTimeOutTable(data[1])
```

Gambar 4.3 Implementasi cara mencari *node* tetangga

- ***Thread untuk membandingkan tabel advertised dengan tabel routing***

Thread ini digunakan untuk membandingkan tabel yang diberikan(*advertised table*) dari *node* tetangga. Sejumlah pengecakan dilakukan seperti pengecekan apabila ada rute baru atau rute terbaru atau rute yang lebih optimal akan dilakukan.

Bentuk format pesan *advertised table* yang dikirim oleh *node* tetangga berupa *list of dictionary* atau daftar kamus. Dimana setiap *dictionary* berisi format rute [tujuan, lompatan selanjutnya, metrik, *sequence number*]. Gambar 4.4 di bawah akan menjelaskan lebih

rinci mengenai cara membandingkan tabel *advertised* dengan tabel *routing* yang sedang dimiliki sekarang.

```
def listen(runEvent):
  while runEvent.is_set()
            data = data_received_from_ip_multicast()
    if(data[0] == 'advertised table' and data[2] != currentIPV6)
              addSettlingTime(data[2], now)
      compareTable(data[1], data[2])

def compareTable(advertisedTable, ownerAT):
  for rowAT in advertisedTable:
            if(rowAT['destination'] != rowRT['destination']):
              if(rowAT['destination'] != currentIPV6):
                if rowAT not in RT then
                        addRoute(rowAT)
                        multicastAdvertisedTable()
                        break
                      else then
                        if rowAT in RT then
                          for rowRT in RT
                                  if rowRT = rowAT then
                                    if rowRT['sequence_number'] >=
rowAT['sequence_number'] then
                                rowRT['metric'] = rowAT['metric'] + 1
                                          multicastAdvertisedTable()
                                          break
                                        else
                                          pass
```

Gambar 4.4 Implementasi cara membandingan tabel yang disebar oleh tetangga dengan tabel *routing* oleh tiap *node*

- ***Thread untuk mendeteksi broken link***

Pada Gambar 4.5 di bawah menggambarkan bagaimana cara mencari tetangga yang hilang karena tidak memberi paket

keberadaan sama sekali dengan cara melakukan deteksi terus-menerus terhadap tabel *Time Out*.

```
def timeOutChecker(runEvent):
  while(runEvent.is_set()):
    for rowTOT in timeOutTable
      if now > rowTOT['time_out'] then
        deleteRoute[rowTOT[ipv6]]
      else then
        pass
        delay()
```

Gambar 4.5 Implementasi untuk mendeteksi *broken link*

4.3.2 Implementasi pada *Node* Pemantau

Di subbab ini akan dijelaskan mengenai implementasi perangkat lunak pada *node* pemantau. Terdapat 2 proses yang berjalan dalam perangkat lunak disini yaitu *thread* untuk mengambil gambar dari modul kamera dan manajemen pengiriman gambar ke *node* penampil. Berikut akan dijelaskan lebih terinci pada Gambar 4.6 dan Gambar 4.7 di bawah:

- ***Thread untuk menangkap gambar***

Thread ini berfungsi untuk menangkap gambar dari modul kamera Raspberry Pi yang kemudian akan disimpan hasil gambar tersebut untuk kemudian bisa digunakan oleh *thread* mengirim gambar. *Thread* ini berjalan secara terus menerus. Bentuk implementasinya bisa dilihat pada *pseudocode* di Gambar 4.6 di bawah.

```
camera = picamera.PiCamera()
imgCount = 1
stream = io.BytesIO()
for foo in camera.capture_continuous(stream,'jpeg', use_video_port=True):
  path = "file_" + str(imgCount) + ".jpg"
```

```
  stream.seek(0)
  image = PIL.Image.open(stream)
  image.save(path)
  stream.seek(0)
  stream.truncate()
  imgCount +=1
  if imgCount > 25
    imgCount = 1
```

Gambar 4.6 Implementasi untuk menangkap gambar dari Raspberry Pi

- ***Thread untuk mengirim gambar ke node penampil***

Thread ini berfungsi untuk memberikan gambar ke *node* penampil. Implementasinya bisa dilihat pada *pseudocode* di Gambar 4.7 di bawah.

```
def recInfo(runEvent)
  while runEvent.is_set()
    data = data_received_from_nodes()
                if data[0] = "I'm a viewer!"
                  if data[1] not in listClient:
                              listClient.append(data[1])
      thread.start_new_thread(sender, data[1])
def sender(ipAddr):
  countDown = 60
  start = now()
  while True:
              if now() - start >= countDown:
                listClient.remove(ipAddr)
                          killSignal = "DONE"
                          sendKIllSignal(killSignal,(ipAddr,port))
              imageFile = readImageFile()
              strImage = encodeToBase64encode(imageFile)
  sendImageTo(strImage,ipAddr)
```

Gambar 4.7 Implementasi proses pengiriman gambar ke *node* penampil

4.3.3 Implementasi pada *Node* Penampil

Ada beberapa hal yang harus diperhatikan dalam pengerjaan *node* penampil yaitu pertama pastikan pustaka python-dev, pillow dan libjpeg-dev sudah terpasang di sistem operasi Linux Mint di *Netbook*. Python-dev sebagai bahasa pemrogaman untuk menampilkan hasil dari *picture streaming*, pillow dan libjpeg-dev sebagai modul pengolah gambar.

- ***Implementasi Thread untuk menyebar pesan identifikasi***

Agar node pemantau bisa tahu node mana yang berfungsi sebagai penampil maka, node penampil perlu memberikan informasi ke semua node bahwa dia ada node penampil untuk kemudian setelah dikenalai oleh node Berikut adalah pseudocode dari thread tersebut yang bisa dilihat pada pseudocode 4.8 di bawah.

```
def sendInfo(runEvent):
  message = "I'm a viewer!"
  while runEvent1.is_set():
    rute = getDest()
    for dest in RT:
      sendto(message,(str(ipAddr), port))
    delay()
```

Gambar 4.8 Implementasi untuk menyebar pesan identifikasi

- ***Implementasi Thread penerimaan gambar***

Proses ini adalah proses menerima paket dari *node* pemantau dan menampilkan paket tersebut pada antar muka pengguna. Proses diawali dari input berupa pesan dari *node server* yang memberitahukan alamat IP-nya. Kemudian klien akan membalas pesan tersebut dengan balasan berupa pesan dengan menggunakan UDP jika klien ini adalah *node viewer* atau bukan *node* perantara. Setelah itu *node* penampil akan mendapat pesan berupa gambar

yang berbentuk string yang kemudian akan diubah menjadi berkas gambar lalu berkas tersebut akan dibaca oleh *node* penampil/klien dan akan ditampilkan pada antarmuka sederhana dengan menggunakan modul Tkinter. Pada Kode Sumber 4.9 merupakan implementasi menerima dan menampilkan gambar.

```
def recInfo(runEvent):
  while runEvent1.is_set():
    message, address = sock.recvfrom(65504)
              message = pickle.loads(message)
               if message == 'DONE' then
        exit()
               else then
        path = "file.jpg"
                          imgString = message.decodeToBase64encode(message)
                          writeFileImage(path, imgString)
                          viewFileImageInGUI(path)
```

Gambar 4.9 Implementasi untuk menerima gambar

BAB V
UJI COBA dan EVALUASI

Pada bab ini akan dilakukan pembahasan dari scenario uji coba dimana hasil dari itu semua akan diujui secara mendalam ke dalam bidang fungsionalitas dan performa.

5.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas adalah pengujian sistem yang dibangun yang dilihat dari segi fungsi-fungsi utama dari sistem tersebut. Ini berarti yang diuji dari segi fungsionalitas adalah pengujian terhadap *streaming* gambar beserta pengujian terhadap protokol *routing* DSDV.

Semua program tersebut dijalankan di Raspberry Pi dengan sistem operasi Raspbian dan *Netbook* dengan sistem operasi Linux Mint versi 17 dimana bahasa program yang digunakan adalah bahasa Python. Berikut akan dijelaskan lebih terinci tentang uji coba dari segi fungsionalitas :

1. Protokol DSDV
   Uji coba implementasi protokol DSDV digunakan untuk dilakukan untuk menguji apakah protokol DSDV berjalan dengan semestinya pada setiap *node* yang berjalan pada sistem yang dibangun agar bisa dilihat apakah jaringan *mesh* dengan protocol DSDV benar-benar bisa dibangun dengan optimal atau tidak.
2. Picture Streaming
   Uji coba implementasi *picure streaming* digunakan untuk menguji apakah proses pengiriman gambar pada sudah termasuk optimal atau belum dilihat dari kualitas gambar yang ditampilkan. Terdapat 2 proses yang akan dilihat dalam uji coba ini yaitu pengiriman gambar dan penerimaan gambar tapi hanya kualitas pengiriman gambar saja yang nanti akan dilihat.

5.1.1 Lingkungan Uji Coba

Pada subbab ini akan dijelaskan langkah-langkah dan gambaran mengenai lingkungan yang digunakan sebagai uji coba sistem. Uji coba dilakukan pada lingkungan Teknik Informatika ITS. Tempat yang akan digunakan antara lain:

1. Laboratorium Algoritma dan Pemrograman Teknik Informatika ITS.
2. Area jalan di lantai 3 gedung A Teknik Informatika ITS.
3. Lapangan tempat parkir Teknik Informatika ITS.

5.1.2 Uji Coba protokol DSDV

Pada uji coba DSDV dilakukan pengiriman pesan terus-menerus oleh tiap *node* ini dikarenakan komunikasi harus dibangun agar rute pada jaringan *mesh* bisa terbentuk. Tempat pengujian dilakukan di koridor lantai 3 gedung A Teknik Informatika ITS dan dilakukan pagi hari sekitar pukul 04.00 WIB. Uji coba yang dilakukan dibedakan menjadi beberapa bagian, yang akan dijelaskan pada subbab-subbab di bawah.

5.1.2.1 Uji coba Membangun Jaringan *Mesh* dengan Metrik 1 *Hop*

Pada uji coba ini akan dilakukan pengujian untuk menentukan jarak terjauh dari sebuah *node* untuk menjalin hubungan bertetangga dengan *node* lainnya. Setelah itu akan dilakukan pengukuran kualitas jaringan untuk setiap jarak yang ditentukan. Hal-hal yang akan diukur antara lain adalah *Signal Level, Link Quality* dan *Noise Level*. *Signal Level*(SL) merupakan kualitas sinyal dilambangkan dengan satuan dBm dimana semakin besar nilai dBm suatu sinyal maka sinyal tersebut akan semakin kuat. *Link Quality*(LQ) adalah penilaian keseluruhan dari kualitas

koneksi. *Noise Level*(NL) adalah tingkat gangguan sinyal lain yang menghalangi kualitas sinyal yang dikeluarkan sinyal tersebut.

Dimisalkan node yang digunakan adalah *node* A dan *node* B. Dimana *node* A tersambung dengan monitor untuk bisa menangkap hasil SL, LQ dan NL dan *node* B yang menggunakan sumber listrik berupa *Power Bank* yang *portable* agar bisa dibawa kemanapun dengan leluasa.

Gambar 5.1 Uji coba *node* A dan *node* B untuk mengukur jarak maksimum dalam 1 *hop*

Pada uji coba kali ini, untuk semua nilai *Signal Level, Link Quality* dan *Noise Level,* pengukuran tidak dilakukan menggunakan satuan dBm tapi menggunakan satuan persentase. Sistem pengambilan data dilakukan dengan menangkap 5 kali pembacaan kualitas jaringan kemudian akan dihitung rata-ratanya untuk setiap jarak yang digunakan. Untuk uji coba ini penulis menggunakan jarak dengan kelipatan 4 meter yang nantinya jika menemukan jarak yang membuat *node* A dan B tidak tersambung akan dilakukan penelusuran dengan cara mengukur tiap 1 meter.

Berikut adalah hasil uji coba yang telah dilakukan oleh penulis untuk mengukur jarak maksimal dari 1 *hop* yang bisa dilihat pada Tabel 5.1 di bawah:

Tabel 5.1 Hasil uji coba jarak maksimum antar 1 *hop*

No.	Jarak(m)	Rerata SL(%)	Rerata LQ(%)	Rerata NL(%)	Koneksi
1	0	100	100	0	Terhubung
2	4	60.4	77.4	0	Terhubung
3	8	51.2	86.8	0	Terhubung
4	12	48.6	49.2	0	Terhubung
5	16	47	36	0	Terhubung
6	17	46	47	0	Terhubung
7	18	47.4	49.8	0	Terhubung
8	19	49	69	0	Terhubung
9	20	-	-	-	Putus

Dari tabel 5.1 di atas dapat dilihat bahwa jarak maksimum yang dicapai adalah 19 meter. LQ dan SL berbanding terbalik dengan jarak. Semakin jauh jarak antar *node*, maka kualitas koneksi semakin buruk.

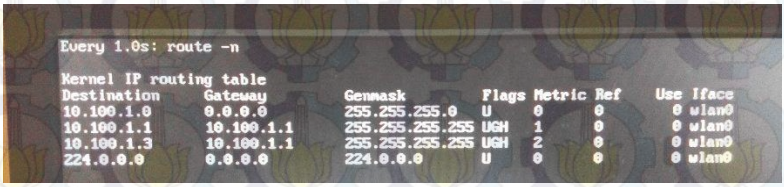
5.1.2.2 Uji coba Membangun Jaringan *Mesh* dengan Metrik Multi-*Hop*

Pada uji coba ini dilakukan untuk menguji seberapa jauh jarak yang diperlukan untuk membangun jaringan *mesh* dengan *hop* lebih dari 1 atau multi-*hop*. Dengan menggunakan referensi dari subbab 5.1.2.1 maka akan dilakukan pengujian dengan menggunakan jarak yang telah diteliti sebelumnya yaitu kelipatan dari 4 meter. Uji coba dilakukan dengan 3 *node* A, B dan C. Skenario penulis adalah *node* A mempuyai metrik 1 *hop* terhadap *node* B dan mempunyai metrik 2 terhadap *node* C dan *node* B mempunyai metrik 1 terhadap *node* C. Uji coba dilakukan di Hasil uji coba bisa dilihat pada tabel 5.2 dibawah:

Tabel 5.2 Hasil uji coba jarak minimal *multihop*

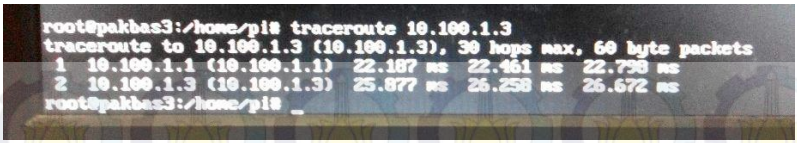
No.	Jarak(m)	Jumlah *Hop*
1	4	1
2	8	1
3	12	1
4	16	1
5	20	1
6	24	1
7	28	2

Pada tabel 5.2 bisa dilihat bahwa terjadi ketidaksamaan data dengan uji coba pada subbab 5.1.2.1. Untuk *node* A ke *node* C diperlukan jarak 28 meter dimana seharusnya pada saat jarak masih 20 – 27 meter masih dianggap metrik 1 *hop* yang seharusnya sudah dianggap 2 *hop*. Gambar 5.2 adalah tabel *routing* kernel hasil uji coba mencari jarak minimum multihop:



Gambar 5.2 Hasil uji coba jaringan mesh dengan metrik multi-*hop*

Pada gambar 5.3 di bawah adalah bukti protokol DSDV berhasil membangun jaringan *mesh* dengan *multi-hop* dengan melakukan pengujian menggunakan *shell script* ‘tracerroute’. Setelah perintah dieksekusi, paket melewati *node* dengan IPV4 10.100.1.1 dulu untuk ke *node* dengan IPV4 10.100.1.3.

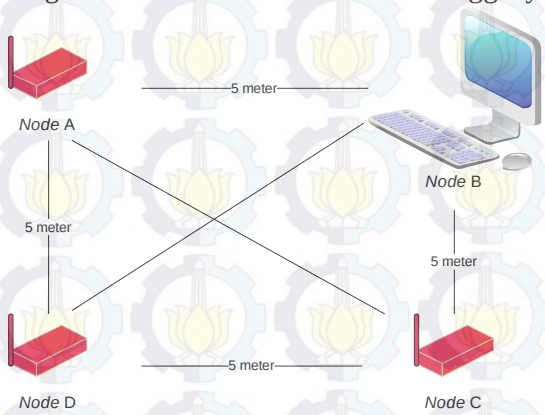


Gambar 5.3 Pembuktian *Multi-Hop* menggunakan *tracerroute*

5.1.2.3 Uji Coba *broken-link* Jaringan *Mesh* pada Protokol DSDV

Pada uji coba kali ini akan dilakukan pengujian terhadap *self healing* terhadap jaringan *mesh* atau penyembuhan jaringan *mesh.* Yang dimaksud dengan penyembuhan adalah cara memperbaiki jaringan *mesh* ketika terjadi *broken Lin.*

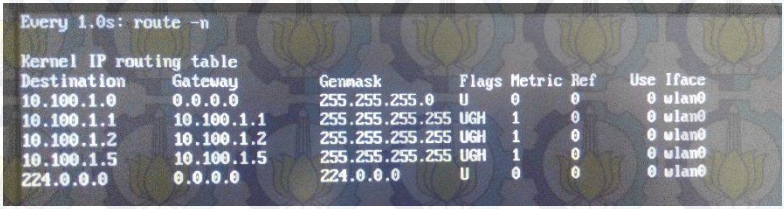
Uji coba dilakukan pada jaringan *mesh* yang telah terbangun kemudian salah satu *node* akan dihilangkan dengan jarak antar *node* 5 meter seperti pada Gambar 5.4. Dari situ akan dilihat bagaimana cara *node* lain menganggapi salah satu tetangganya apabila ada yang hilang. Setelah itu *node* yang hilang tersebut akan dimunculkan lagi dan akan dilihat reaksi *node* tetangganya.



Gambar 5.4 Skema uji coba *self healing* terhadap jaringan *mesh* menggunakan protokol DSDV

Ketika protkol DSDV berjalan, semua *node* saling menangkap sinyal pesan dari seluruh tetangga sehingga

menganggap semua sebagai tetangga dengan metrik 1 *hop*. Berikut adalah tabel *routing* dari salah satu *node*.



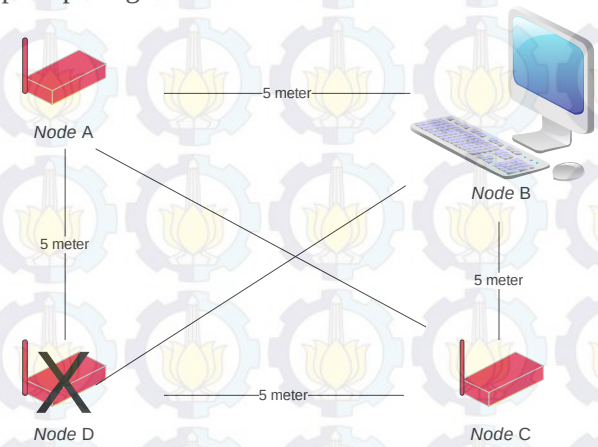
Every 1.0s: route -n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.100.1.0	0.0.0.0	255.255.255.0	U	0	0	0	wlan0
10.100.1.1	10.100.1.1	255.255.255.255	UGH	1	0	0	wlan0
10.100.1.2	10.100.1.2	255.255.255.255	UGH	1	0	0	wlan0
10.100.1.5	10.100.1.5	255.255.255.255	UGH	1	0	0	wlan0
224.0.0.0	0.0.0.0	224.0.0.0	U	0	0	0	wlan0

Gambar 5.5 Tabel rute IPV4 pada salah satu *node* sebelum dihilangkan salah satu *node* tetangganya

Pada gambar 5.5 di atas adalah tabel *routing* dari *node* dengan IPV4 [10.100.1.3]. *Node* tersebut mempunyai tetangga dengan IPV6 [10.100.1.1], [10.100.1.4] dan [10.100.1.5] dengan metrik masing-masing bernilai 1 *hop*. Setelah itu akan dilakukan penghilangan salah satu *node* dengan IPV4 [10.100.1.1] dengan skema seperti pada gambar 5.6 di bawah.



Gambar 5.6 Skema jaringan *mesh* dengan salah satu *node* hilang

Setelah *node* tersebut hilang maka tabel *routing* yang baru akan menghilangkan rute tersebut seperti pada gambar 5.6 di bawah.

Gambar 5.7 tabel rute IPV4 pada salah satu *node* setelah dihilangkan salah satu *node* tetangganya

Setelah itu dilakukan penghidupan kembali terhadap *node* dengan IPV4 [10.100.1.1]. Maka tabel *routing* akan kembali seperti semula sama dengan gambar 5.5.

5.1.3 Uji Coba Picture Streaming

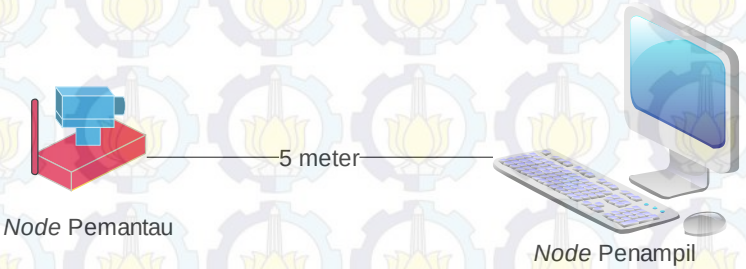
Pada uji coba *picture streaming* terjadi proses pengiriman gambar dari *node* pemantau ke *node* penampil yang berlangsung secara terus menerus. Yang dilihat dalam percobaan *picture streaming* adalah presentase keberhasilan dengan menghitung jumlah *frame* yang dikirim oleh *node* pemantau ke *node* penampil untuk tiap satu menit. Hal yang perlu diperhatikan adalah berapa *frame* yang sampai ke *node* penampil dalam satu detik, *frame* yang *corrupt* atau data rusak tidak dianggap dalam penghitungan fps.

Uji coba dilakukan dengan berbagai skenario. Terdapat dua skenario yang akan diuji coba yaitu uji coba *picture streaming* pada 2 *node* saja dan ada yang lebih dari 2 *node.* Masing-masing uji coba berjalan pada jaringan *mesh* dengan menggunakan protokol rute DSDV.

5.1.3.1 Uji Coba *Picture Streaming* pada One-Hop Distance (2 *Node*)

Tujuan dari uji coba ini adalah untuk menguji fungsionalitas dari *picture streaming* dengan hanya menggunakan 2 *node* saja. Uji coba dilakukan dengan menggunakan 1 *node* penampil berupa *Netbook* dan 1 *node* pemantau berupa Raspberry Pi. Jarak yang

digunakan adalah 5 meter dengan pengukuran dilakukan selama 5 menit dan dilakukan selama 5 kali. Skema yang akan digunakan bisa dilihat pada Gambar 5.8 di bawah.



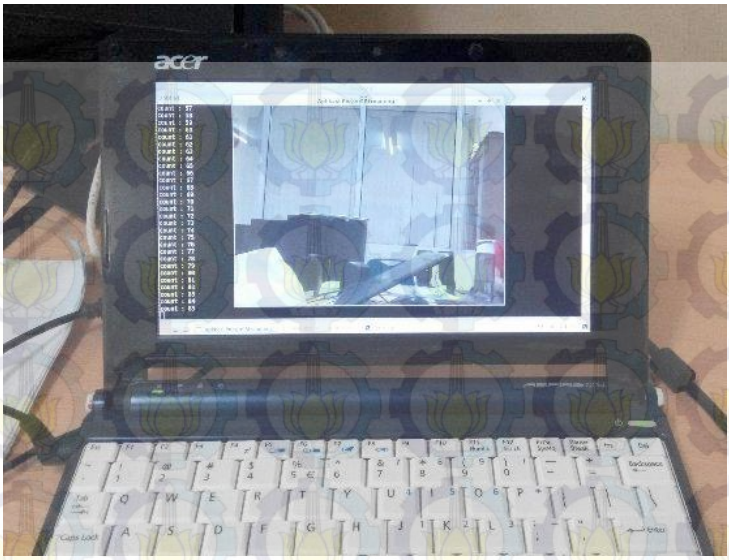
Gambar 5.8 Jaringan mesh dengan 2 node menggunakan protokol routing DSDV

Dari hasil uji coba tersebut maka didapatkan hasil pengujian yang ditulis pada Tabel 5.3 di bawah.

Tabel 5.3 Hasil uji coba *picture streaming* dengan metrik 1 *hop*

No.	FPS
1	3.59
2	3.50
3	3.57
4	3.50
5	3.61

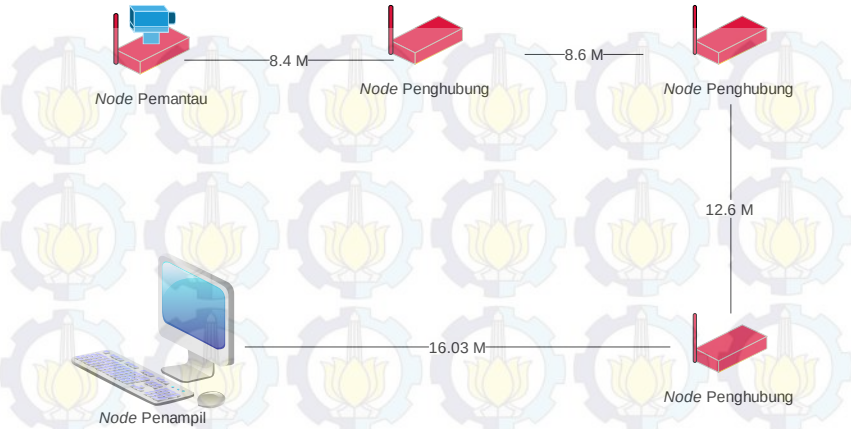
Kolom FPS pada Tabel 5.3 di atas merupakan jumlah gambar yang berhasil dikirimkan ke *node* penerima oleh *node* pemantau selama 1 detik. Dari 5 percobaan didapatkan rata-rata fps senilai 3.55. Uji coba ini membuktikan bahwa *picture streaming* bisa dilakukan dengan lingkungan yang telah dipersiapkan sebelumnya. Hasil uji coba bisa dilihat pada Gambar 5.9 di bawah.



Gambar 5.9 Hasil uji coba *picture streaming* dengan metrik 1 *hop*

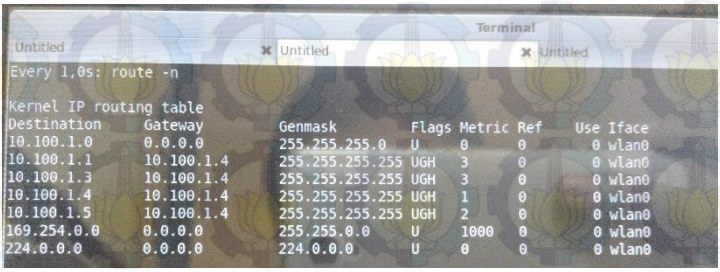
5.1.3.2 Uji Coba *Picture Streaming* pada Jaringan *Mesh* dengan Metrik *Multi-Hop*

Uji coba ini dilakukan untuk membuktikan apakah sistem yang telah dibuat oleh penulis mampu melakukan *picture streaming* di dalam jaringan *mesh* yang mempunyai metrik *multi-hop*. Pada uji coba ini digunakan 5 *node* dengan formasi 1 *node* pemantau, 3 *node* penghubung dan 1 *node* penampil *Netbook*. Inti dari uji coba ini adalah untuk membuktikan apakah *picture streaming* mampu dilakukan pada sejumlah *node* dengan subnet yang berbeda pada jaringan *mesh* yang dibangun dengan menggunakan protokol *routing* DSDV. Uji coba ini dilakukan berdasarkan parameter lingkungan dan hasil uji coba pada subbab 5.1.2.2. Skenario yang akan dibangun akan dijelaskan pada Gambar 5.10 di bawah.



Gambar 5.10 Skema uji coba *picture streaming* pada multi-*hop* pada jaringan *mesh* menggunakan protokol *routing* DSDV

Gambar 5.11 adalah tabel *routing* pada kernel dengan IPV4 yang dihasilkan oleh protokol DSDV yang menghasilkan jaringan *mesh* nirkabel dengan metrik *multi-hop*. Di sini yang menjadi *node* pemantau adalah *node* dengan IPV4 [10.100.1.3], *node* penghubung dengan IPV4 masing-masing yaitu [10.100.1.1], [10.100.1.4] dan [10.10.1.5] dan *node* penampil dengan IPV4 [10.100.1.2]. Dari gambar tersebut bisa dilihat pada kolom *Metric* menunjukan untuk menuju [10.100.1.3] membutuhkan 3 *hop* dengan harus melewati *node* [10.100.1.4] terlebih dahulu.



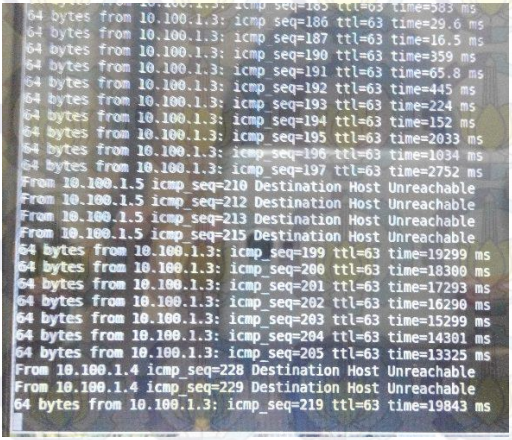
Gambar 5.11 Tabel *routing* kernel pada *node* penampil

Kolom paket terkirim pada Tabel 5.4 di bawah merupakan jumlah gambar yang dikirimkan ke *node* penerima oleh *node* pemantau. Pada uji coba ini hanya dilakukan 3 kali percobaan dikarenakan jaringan yang labil dan sulit untuk membuat skenario jaringan *mesh* dengan metrik *multi-hop*.

Tabel 5.4 Hasil uji coba *picture streaming* pada multi-*hop*

No.	*Frame* per menit
1	1
2	3
3	1

Setelah dilakukan 3 kali uji coba ternyata frame yang diterima sangat sedikit sekali. Hanya 1-3 *frame* yang diterima yang seharusnya diterima adalah sekitar 180 frame dalam 60 detik. Hal ini bisa dibuktikan dengan melihat hasil *ping* yang dilakukan dari [10.100.1.2] ke [10.100.1.3] menunjukan waktu rata-rata yang cukup besar antara 2000-20000 milidetik yang bisa dilihat pada Gambar 5.12 di bawah.

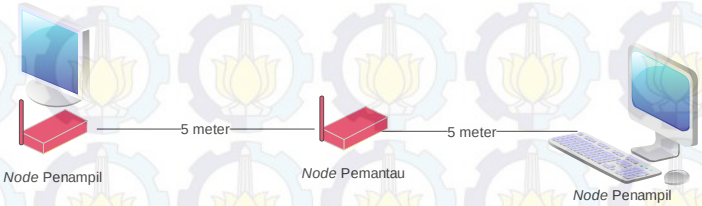


Gambar 5.12 Hasil uji coba tes *ping* dari *node* penampil ke *node* pemantau

Dari hasil pengamatan di atas dapat disimpulkan bahwa untuk melakukan *picture streaming* pada jaringan *mesh* dengan metrik *multi-hop* dengan perangkat-perangkat yang digunakan tidak cocok karena jumlah *frame* yang sampai ke *node* penampil hanya 1-2 saja.

5.1.3.3 Uji Coba P*icture Streaming* Menggunakan 2 *Node* Penampil (M*ultiviewer*)

Uji coba ini dilakukan dengan tujuan untuk membuktikan apakah sistem yang dibangun oleh penulis mampu melakukan pengiriman gambar kepada 2 *node* penampil. Pada uji coba kali ini akan dilakukan dengan menggunakan 1 *node* pemantau Raspberry Pi, 1 *node* pemantau *Netbook* dan 1 *node* penampil Raspberry Pi. Kedua *node* penampil tersebut akan diletakkan pada 2 tempat yang berbeda dengan jarak 5 meter dari *node* pemantau. Skema uji coba ini bisa dilihat pada gambar 5.13 di bawah.



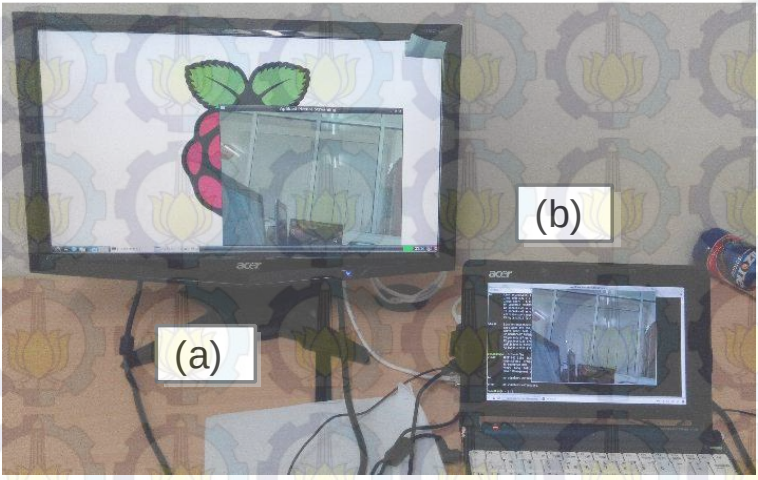
Gambar 5.13 Skema uji coba *picture streaming* menggunakan 2 *node* penampil(*multiviewer*)

Tabel 5.5 Hasil uji coba *picture streaming* pada *multiviewer* dari sisi *node* penampil *Netbook*

No.	FPS	
	Netbook	Raspberry Pi
1	3.52	2.34
2	3.57	2.31
3	3.49	2.21
4	3.55	2.33
5	3.57	2.23

Setelah dilakukan 5 kali uji coba selama 1 menit ternyata ada perbandingan yang cukup besar dari jumlah FPS. Hasil uji coba bisa dilihat pada tabel 5.5 di atas.

Dari hasil uji coba di atas, bisa kita lihat bahwa ternyata *node* penampil berupa Raspberry Pi mampu berfungsi sebagai *node* penampil. Dengan rata-rata nilaiFPS pada Raspberry Pi adalah 2.284 dan pada *notebook* adalah 3.54. Meskipun nilai rata-rata FPS hampir setengah dari pada *node* penampil *Netbook*. Gambar 5.10 di bawah adalah hasil uji coba menggunakan 2 *node* penampil dengan *node* penampil Netbook di sebelah kanan dan *node* penampil Raspberry Pi di sebelah kiri menggunakan layar monitor LCD.



Gambar 5.14 Hasil uji coba *picture streaming multiviewer*. (a) *Node* penampil Raspberry Pi dan (b) *Node* penampil *netbook*

5.1.3.4 Uji Coba Perbandingan *Picture Streaming* pada *Static Routing* dengan Protokol DSDV

Pada uji coba ini akan dilihat perbandingan jumlah FPS pada saat dilakukan *static routing* dengan saat dilakukan protokol

DSDV. Tujuan dilakukan uji coba ini adalah untuk membuktikan apakah protokol *routing* DSDV mempengaruhi proses keberhasilan *picture streaming*. Uji coba dilakukan dengan menggunakan metrik 1 *hop* dengan jarak 5 meter. Uji coba akan dilakukan 5 kali untuk tiap *routing* dan akan dihitung rata-rata FPS untuk nantinya akan dibandingkan mana yang lebih optimal. *Node* Penampil menggunakan *Netbook* dan *node* pemantau menggunakan Raspberry Pi. Hasil uji coba bisa dilihat pada Tabel 5.6 di bawah.

Tabel 5.6 Tabel perbandingan persentase keberhasilan dengan DSDV *routing* dan *static routing*

No.	FPS	
	DSDV Routing	Static *Routing*
1	3.59	3.59
2	3.50	3.49
3	3.57	3.65
4	3.50	3.59
5	3.61	3.57

Dari tabel 5.6 dilihat ternyata rata-rata FPS dengan menggunakan DSDV *routing* adalah 3.55 dan *static routing* adalah 3.58. Maka bisa disimpulkan bahwa *static routing* lebih optimal daripada DSDV *routing* meskipun berbanding tipis.

5.2 Uji Coba Performa

Uji coba ini bertujuan untuk mengetahui seberapa besar pemakaian CPU dan memori pada saat menjalankan fitur-fitur yang ada pada Tugas Akhir ini. Ada 2 uji coba yang dilakukan dalam pengujian uji coba performa ini. Yang pertama adalah uji coba dilakukan menggunakan 1 *node* pemantau Raspberry Pi dan 1 *node* penampil *Netbook* kemudian akan ditaruh berjauhan dan

yang kedua adalah membangun jaringan dengan 2 *node* penghubung dari Raspberry Pi dan *netbook.* Cara menangkap hasil performa dilakukan dengan menambahkan *script* ke dalam proses yang sedang berjalan dimana tiap detik *script* tersebut akan mengambil data performa CPU dan *memory* yang ada.

5.2.1 Uji Coba Penggunaan CPU dan Memory pada Picture Streaming (Singleviewer)

Uji coba ini dilakukan untuk mengukur CPU dan *memory* yang digunakan oleh *node* pemantau dan *node* penampil pada saat terjadi *picture streaming*. Uji coba dilakukan sebanyak 5 kali dengan masing-masing menggunakan durasi waktu 5 menit. Hasil uji coba pada *node* pemantau dapat dilihat pada Tabel 5.7 di bawah. Cara pengambilan data CPU dan *memory* dengan perintah 'top' kemudian diambil program yang berjalan untuk program *picture streaming* tidak termasuk program untuk protokol *routing* DSDV.

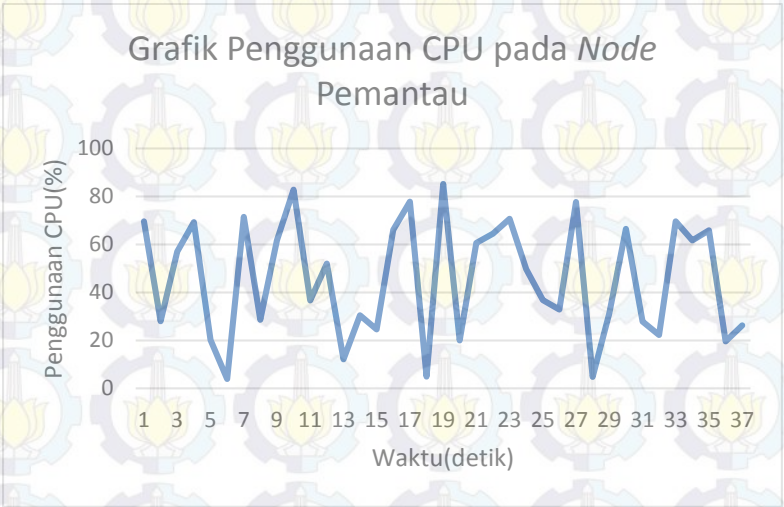
Tabel 5.7 Hasil uji coba penggunaan CPU dan *memory* pada *node* pemantau dan *node* penampil

Node Pemantau			*Node* Penampil	
No.	CPU(%)	*Memory*(%)	CPU(%)	*Memory*(%)
1	48.09	3.10	16.79	1.66
2	48.41	3.10	19.84	1.66
3	47.76	3.10	18.39	1.61
4	45.71	3.10	22.41	1.62
5	42.47	3.10	17.56	1.60

Hasil uji coba bisa dilihat pada Tabel 5.8 di atas. Tabel tersebut menunjukan bahwa rata-rata penggunaan CPU pada *node* pemantau Raspberry Pi memakan 46.49% dan rata-rata *memory* yang digunakan pada *node* pemantau konsisten yaitu 3.10% dari CPU yang dimiliki sedangkan pada *node* penampil rata-rata memakan CPU sebesar 19% dan *memory* sebesar 1.63%. Penyebab perbandingan nilai CPU dan *memory* pada 2 *node* tersebut berbeda

cukup jauh karena spesifikasi perangkat keras Raspberry Pi lebih buruk daripada *notebook* yang digunakan sekarang.

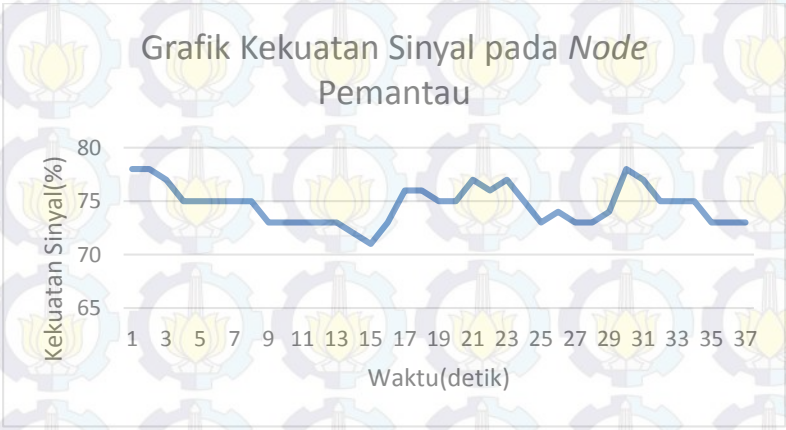
Grafik-grafik yang akan diperlehiatkan setelah ini mengambil dari Tabel 5.8 pada uji coba ke-4 yang dilakukan selama 1 menit. Grafik sangat tidak stabil pada awal *streaming* karena CPU yang digunakan sempat turun tapi setelah itu penggunaan CPU naik lagi dan pola naik turun terus berlangsung selama 1 menit dengan nilai maksimum mencapai sekitar 50%. Pada grafik CPU, *memory* dan kekuatan sinyal yang ditangkap hanya sekitar 37 kali dikarenakan proses untuk melakukan proses '*grep'* memakan waktu cukup lama di Raspberry Pi.



Gambar 5.15 Grafik penggunaan CPU pada *node* pemantau selama 1 menit

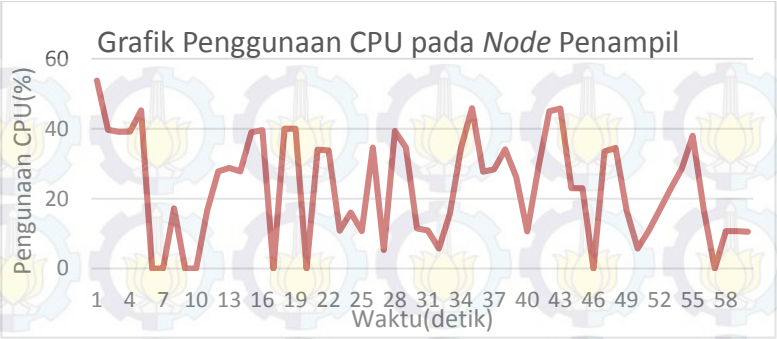
Pada Gambar 5.15 di atas adalah grafik penggunaan CPU pada *node* pemantau selama 1 menit tidak stabil. Sedangkan pada Gambar 5.16 di bawah dapat dilihat grafik penggunaan *memory* pada *node* pemantau. Dari grafik bisa dilihat seiring waktu penggunaan *memory* terus naik dengan keadaan stabil.

Gambar 5.16 Grafik penggunaan *memory* pada *node* pemantau selama 1 menit



Gambar 5.17 Grafik *signal level* pada *node* pemantau

Pada Gambar 5.17 di atas menjelaskan grafik kekuatan sinyal pada *node* pemantau. Grafik menunjukan bahwa sinyal sempet mengalami penurunan tapi setelah itu berangsur naik. Sinyal enderung naik turun pada kisaran nilai 75%.

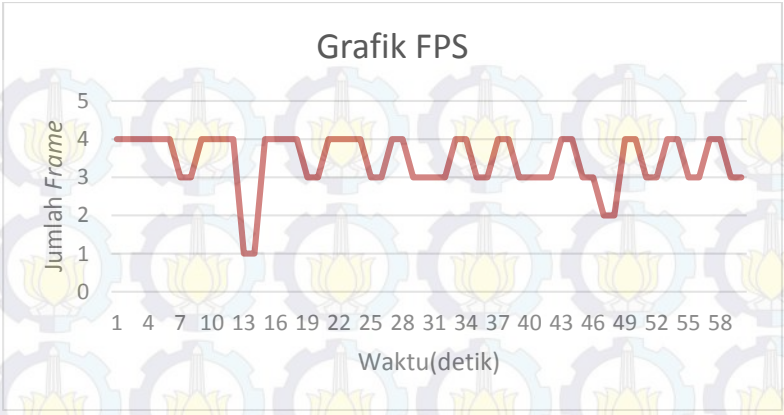


Gambar 5.18 Grafik penggunaan CPU pada *node* penampil selama 1 menit

Pada Gambar 5.18 di atas menggambarkan penggunaan CPU sangat tidak stabil pada *node* penampil. Pada awal *streaming* CPU yang digunakan mencapai 0% yang diperkirakan karena pada saat itu proses penerimaan *frame* sedang tidak bekerja karena belum mendapat *frame* dari *node* pemantau.

Gambar 5.19 Grafik penggunaan *memory* pada *node* penampil selama 1 menit

Pada Gambar 5.19 di atas menunjukan perubahan kebutuhan *memory* oleh *node* penampil. Bila dibandingkan dengan kebutuhan *memory* pada *node* pemantau, *node* pemantau terlihat lebih stabil daripada *node* penampil.



Gambar 5.20 Grafik jumlah *frame* yang diterima untuk tiap detik pada *node* penampil selama 1 menit

Pada Gambar 5.20 di atas menunjukan berapa *frame* yang diterima oleh *node* pemantau untuk tiap detiknya. Dari grafik diatas rata-rata FPS yang diterima adalah 3-4 meskipun pernah sempat mengalami penurunan sampai 1 *frame* per detik.

5.2.2 Uji Coba Penggunaan CPU dan Memory pada Picture Streaming (Multiviewer)

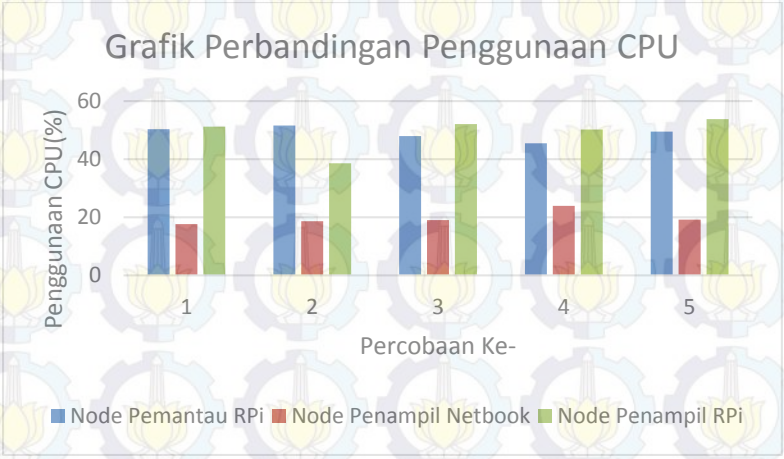
Pada uji coba ini akan menggunakan skenario dengan 1 *node* pemantau Raspberry Pi, 1 *node* penampil Raspberry Pi dan 1 *node* penampil *Netbook*. Kedua *node* penampil diletakkan pada jarak yang sama dari *node* pemantau sesuai uji coba pada subbab 5.1.3.3. Data yang diambil adalah persentase CPU dan *memory* dari tiap *node* ketika melakukan proses *picture streaming.*

Pada Tabel 5.8 di bawah menjelaskan hasil uji coba yang telah dilakukan pada uji coba *picture streaming* menggunakan 2 *node* penampil(*multiviewer*) untuk melihat berapa persen CPU yang dipakai sebuah *node* untuk melaksanakan aplikasi *picture streaming*. Dapat dilihat bahwa penggunaan CPU paling berat dialami oleh *node* pemantau berupa Raspberry Pi, yang kedua adalah *node* penampil Raspberry Pi dan yang ketiga adalah *node*

penampil *Netbook*. Untuk membandingkan lebih jelas bisa dilihat pada grafik di Gambar 5.21 di bawah.

Tabel 5.8 Data perbandingan penggunaan CPU pada masing-masing *node*

Persentase Penggunaan CPU(%)		
Node Pemantau RPi	Node Penampil Netbook	*Node* Penampil RPi
50.32	17.64	51.18
51.59	18.67	38.61
47.96	19.04	52.06
45.46	23.91	50.21
49.46	19.20	53.79



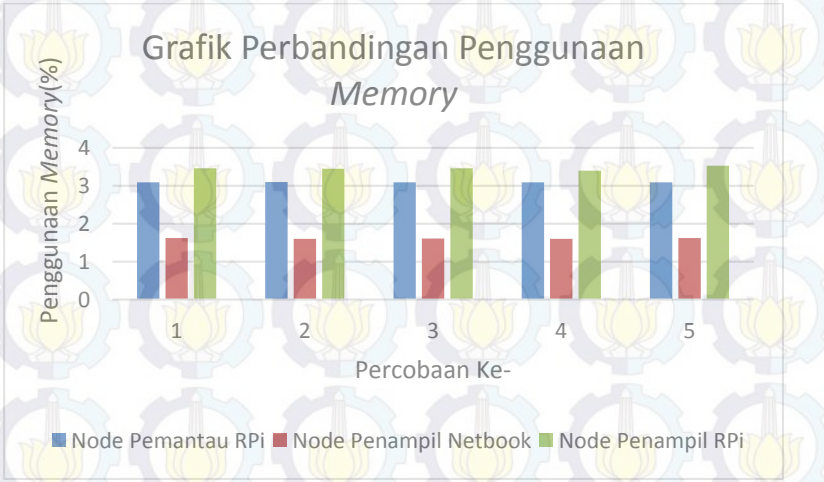
Gambar 5.21 Grafik perbandingan CPU pada *node-node multiviewer picture streaming* di jaringan *mesh*

Pada Tabel 5.9 di bawah menjelaskan hasil uji coba yang telah dilakukan pada uji coba *picture streaming* menggunakan 2 *node* penampil(*multiviewer*) untuk melihat berapa persen *memory*

yang dipakai sebuah *node* untuk melaksanakan aplikasi *picture streaming*. Dapat dilihat bahwa penggunaan CPU paling berat dialami oleh *node* pemantau berupa Raspberry Pi. Untuk membandingkan lebih jelas bisa dilihat pada grafik di Gambar 5.22 di bawah.

Tabel 5.9 Data perbandingan penggunaan *memory* pada masing-masing *node*

Persentase Penggunaan *Memory*(%)		
Node Pemantau RPi	Node Penampil Netbook	*Node* Penampil RPi
3.09	1.62	3.46
3.10	1.60	3.45
3.09	1.61	3.46
3.09	1.60	3.40
3.09	1.62	3.53



Gambar 5.22 Grafik perbandingan *memory* pada *node multiviewer picture streaming*

5.2.3 Uji Coba Penggunaan CPU dan Memory pada Protokol DSDV

Pada uji coba ini akan dilakukan pengukuran terhadap penggunaan CPU dan *memory* yang diperlukan oleh protokol DSDV baik dari *node* berupa Raspberry Pi dan *Netbook.*

Uji coba dilakukan dengan mengambil nilai CPU dan *memory* dari *shell script* 'top' yang dilakukan tiap 1 menit.

Tabel 5.10 Penggunaan CPU untuk protokol *routing* DSDV pada *netbook* (bagian 1)

Netbook		
No.	CPU(%)	Memory(%)
1	0	0.7
2	0	0.7
3	5.8	0.7
4	0	0.7
5	0	0.7
6	0	0.7
7	0	0.7
8	0	0.7
9	0	0.7
10	0	0.7
11	0	0.7
12	0	0.7
13	0	0.7
14	0	0.7
15	0	0.7
16	0	0.7
17	0	0.7

Tabel 5.11 Penggunaan CPU untuk protokol *routing* DSDV pada *netbook* (bagian 2)

No.	CPU(%)	Memory(%)
18	0	0.7
19	0	0.7
20	5.7	0.7
21	0	0.7
22	0	0.7
23	0	0.7
24	0	0.7
25	0	0.7
26	0	0.7
27	0	0.7
28	0	0.7
29	0	0.7
30	0	0.7
31	5.5	0.7
32	0	0.7
33	0	0.7
34	0	0.7
35	0	0.7
36	0	0.7
37	0	0.7
38	10.6	0.7
39	0	0.7
40	0	0.7
41	0	0.7
42	0	0.7

Tabel 5.12 Penggunaan CPU untuk protokol *routing* DSDV pada *netbook* (bagian 3)

No.	CPU(%)	Memory(%)
43	0	0.7
44	0	0.7
45	0	0.7
46	0	0.7
47	0	0.7
48	11.4	0.7
49	0	0.7

Tabel 5.13 Penggunaan CPU untuk protokol *routing* DSDV pada Raspberry Pi (bagian 1)

Raspberry Pi		
No.	CPU(%)	Memory(%)
1	0	1.8
2	0	1.8
3	0	1.8
4	0	1.8
5	0	1.8
6	0	1.8
7	0	1.8
8	0	1.8
9	0	1.8
10	0	1.8
11	0	1.8
12	0	1.8
13	0	1.8

Tabel 5.14 Penggunaan CPU untuk protokol *routing* DSDV pada Raspberry Pi (bagian 2)

No.	CPU(%)	Memory(%)
14	0	1.8
15	0	1.8
16	0	1.8
17	0	1.8
18	0	1.8
19	0	1.8
20	5.2	1.8
21	0	1.8
22	0	1.8
23	5.2	1.8
24	10.3	1.8
25	0	1.8
26	0	1.8
27	0	1.8
28	0	1.8
29	0	1.8
30	0	1.8
31	0	1.8
32	0	1.8
33	5.3	1.8
34	0	1.8
35	10.5	1.8
36	20.8	1.8
37	0	1.8
38	0	1.8
39	0	1.8

Tabel 5.15 Penggunaan CPU untuk protokol *routing* DSDV pada Raspberry Pi (bagian 3)

No.	CPU(%)	Memory(%)
40	0	1.8
41	0	1.8
42	0	1.8
43	0	1.8
44	0	1.8
45	0	1.8

Dari Tabel 5.10 dan Tabel 5.11 di atas merupakan salah satu sampel dari banyak hasil uji coba yang dilakukan tiap menit. Pada sebagian besar uji hampir tidak pernah terjadi perubahan nilai CPU yang berarti, protokol DSDV jarang menggunakannya sehingga nilai yang dihasilkan adalah 0%. Sedangkan yang dicantumkan pada Tabel 5.10-5.12 dan Tabel 5.11-5.15 adalah uji coba yang mengalami paling banyak perubahan pada CPU dan pada penggunaan memory cenderung konstan.

Dari uji coba ini dapat disimpulkan bahwa protokol DSDV tidak membutuhkan sumber daya yang cukup besar karena protokol DSDV yang dibuat baru membutuhkan CPU ketika terjadi pada kejadian tertentu saja.

BAB VI
KESIMPULAN dan SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari sejumlah hasil uji coba yang telah dilakukan terhadap implementasi *picture streaming* pada jaringan *mesh* berbasis protokol DSDV menggunakan Raspberry Pi untuk pemantauan jalan raya dapat diambil kesimpulan sebagai berikut:

1. Untuk membangun jaringan *mesh* dengan protokol DSDV dengan hanya 2 *node*(1 *hop*) membutuhkan jarak maksimal ±19 meter sedangkan untuk membuat *multihop* paling tidak dibutuhkan ±28 meter.
2. Jumlah *frame* pada proses *picture streaming* dengan parameter metrik 1 *hop* dengan jarak 5 meter adalah sebesar 3.55.
3. Tingkat keberhasilan gambar terkirim ke *node* penampil berkurang seiring jumlah *node* yang harus dilalui untuk sampai ke tujuan. Bahkan dengan metrik 3 *hop, frame* yang diterima oleh *node* penampil hanya 1 saja.
4. Raspberry Pi kurang cocok digunakan sebagai *node* penampil pada *multiviewer* karena hampir separuh paket yang dikirimkan oleh *node* pemantau tidak diterima dengan rata-rata FPS senilai 2.28 dan rata-rata FPS pada *notebook* adalah 3.54
5. Protokol *routing* DSDV berjalan sesuai semestinya seperti fitur-fitur *Damping Fluctuation, Unidirectional,* deteksi *broken link* dan *self-healing.*

6.2 Saran

Saran yang diberikan untuk pengembangan picture streaming pada jaringan mesh berbasis protokol DSDV menggunakan Raspberry Pi:

1. Diharapkan untuk riset kedepannya bisa menggunakan *node* lebih banyak agar protokol *routing* DSDV bisa terlihat.
2. Untuk pengembangan lebih lanjut, diharapkan menggunakan kamera dengan kecepatan lebih tinggi sehingga dapat menangkap gambar lebih banyak setiap detiknya.
3. Untuk keamanan data, sebaiknya protokol DSDV ditambahi enkripsi pada pengiriman data.

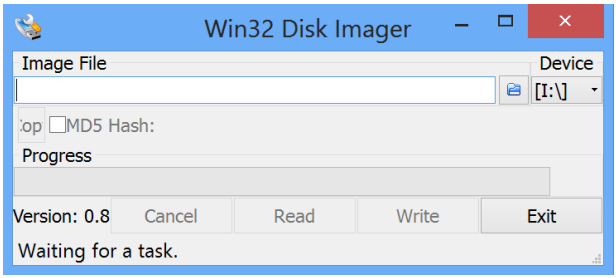
Untuk pengembangan penelitian ini ada baiknya dibuatkan suatu sistem yang *user friendly* sehingga pihak-pihak yang memiliki keperluan dapat menerapkan sistem ini dalam kehidupan mereka.

LAMPIRAN

1. Memasang Sistem Operasi Raspbian Wheezy

Untuk bisa mengimplemntasi protokol DSDV pada Raspberry Pi, yang dilakukan pertama kali adalah memasang sistem operasi terlebih dahulu sebagai dasar sistem untuk menjalankan program. Untuk melakukan instalasi sistem operasi Raspbian Wheezy pada Raspberry Pi akan dijelaskan melalui langkah-langkah sebagai berikut:

1. Siapkan sebuah SDHC card dengan ukuran *storage* minimal 4GB.
2. Unduh *image* sistem operasi Raspberry Pi, yaitu Raspbian Wheezy dengan tanggal rilis 26 juli 2013 pada situs http://www.raspbian.org/.
3. Ekstrak file yang telah diunduh, sehingga terdapat *file* dengan format .img.
4. Unduh aplikasi Win32 Disk Imager untuk melakukan instalasi Raspbian Wheezy pada SDHC *card*.
5. Ekstrak aplikasi Win32 Disk Imager.



Gambar A.1 Ilustrasi Win32 Disk Imager

6. Pada Gambar A.1 di atas ditunjukkan tampilan aplikasi Win32 Disk Imager, pilih *drive* SD *card* yang ingin di-*format* dan sorot berkas *image* dari sistem operasi Raspbian Wheezy.

7. Klik *Write* dan tunggun sampai prosesnya selesai dan SD *card* siap digunakan untuk menjalankan sistem operasi Raspberry Pi.

2. Inisialisasi Modul Kamera

Modul kamera Raspberry Pi adalah elemen penting dalam pembuatan sistem ini maka hal ini sangat diperlukan. Untuk membangun, pertama pasang modul kamera pada port Raspberry Pi yang telah disediakan untuk modul kamera, kemudian untuk menginisialisasi:

- Masukkan perintah berikut pada CLI

```
$raspistill –o gambar.jpg
```

- Kemudian jika kamera sudah terpasang dengan benar, maka akan ada *preview* pada layar monitor sebelum gambar ditangkap oleh kamera. Gambar A.2 merupakan hasil tangkapan kamera mini komputer Raspberry Pi.

Gambar A.2 Gambar sampel contoh hasil tangkapan modul kamera Raspberry Pi

3. Mengkonfigurasi Wifi

Untuk bisa berkomunikasi dengan perangkat lain maka perlu dibangun sebuah jaringan pada setiap perangkat. Untuk mengkonfigurasi jaringan pada saat *startup*, maka yang harus dikonfigurasi adalah:

```
$ sudo nano /etc/network/interfaces
```

Pada nano text editor, akan dilihat konfigurasi awal seperti pada Gambar A.3 di bawah:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
```

Gambar A.3 Konfigurasi awal di /etc/network/interfaces

Pada Gambar A.3 di atas merupakan konfigurasi dasar untuk menghubungkan ke kabel Ethernet, sehingga perlu ditambahkan konfigurasi untuk wireless. Khususnya untuk membangun jaringan *ad-hoc*. Berikut adalah konfigurasi yang perlu ditambahkan bisa dilihat pada gambar A.4 di bawah.

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wireless-channel 1
wireless-essid MESH
wireless-mode ad-hoc
iface default inet dhcp
```

Gambar A.4 Konfigurasi tambahan untuk mengatur jaringan *ad-hoc* pada /etc/network/interfaces

Setelah itu masukkan perintah berikut untuk me-*restart* konfigurasi jaringan.

```
$ /etc/init.d/networking restart
```

4. Instalasi PiCamera

Picamera merupakan pustaka khusus yang diciptakan untuk bisa memanfaatkan perangkat modul kamera Raspberry Pi dari dalam *script* Python. Python yang dimaksud adalah Python bersi 2.7 dan Python 3.2. Untuk bisa memasang pustaka ini, banyak pustaka lain yang harus dipasang terlebih dahulu. Berikut adalah daftar paket dan cara memasang pustaka yang harus dipasang terlebih dahulu:

a. python-setuptools

Python-setuptools adalah modul dari bahasa Python yang mempunyai fungsi dasar untuk bisa menginstall modul Python yang lain.

```
$ sudo apt-get install python-setuptools
```

b. python-pip

Python pip adalah modul dari Python untuk digunakan memasang modul-modul python dengan mudah. Pip berfungsi layaknya seperti apt-get.

Perintah pasang :

```
$ sudo apt-get install python-pip
```

c. picamera

Setelah modul PiCamera sudah terpasang lakukan ujicoba dengan mengetikkan perintah berikut:

```
$ sudo pip install picamera
```

d. pillow

Modul pillow adalah modul untuk pengolahan gambar yang kompatibel dengan Python. Ketikkan perintah tersebut untuk memasang pillow.

```
$ sudo pip install pillow
```

e. libjpeg-dev

libjpeg-dev sebagai decoder jpeg dengan mengetikkan perintah berikut:

```
$ sudo apt-get install libjpeg-dev
```

DAFTAR PUSTAKA

[1] X. Lu and M. Camitz, "Finding the shortest paths by node combination," *Applied Mathematics and Computation 217*, pp. 6401-6408, 2011.

[2] Electrical & Computer Engineering / Computer Science University of Delaware, "University of Delaware," [Online]. Available: http://www.eecis.udel.edu/~sunshine/courses/F05/CIS450/class17.pdf. [Accessed 1 Maret 2014].

[3] airberry, "http://airberry.com/," [Online]. Available: http://airberry.com/downloads/airberry_Whitepaper_EN_02_Wireless_Mesh.pdf. [Accessed 27 Januari 2015].

[4] Guoyou He, "Destination-Sequenced Distance Vector (DSDV) Protocol," [Online]. Available: www.netlab.tkk.fi/opetus/s38030/k02/Papers/03-Guoyou.pdf. [Accessed 1 Februari 2014].

[5] RaspberryPi, "Raspberry Pi: FAQs," [Online]. Available: http://www.raspberrypi.org/faqs#introWhatIs. [Accessed 24 Februari 2014].

[6] R. P. Foundation, "Raspbian," Raspberry Pi Foundation, [Online]. Available: http://www.raspbian.org/RaspbianFAQ. [Accessed 1 Agustus 2013].

BIODATA PENULIS

AHADYN ALIEF ARLINGGA, lahir di Gresik, pada tanggal 1 November 1992. Penulis menempuh pendidikan mulai dari SDN Pekuncen I II III Pasuruan(1998-2004), SMPN 2 Pasuruan (2004-2007), SMAN 1 Pasuruan (2007-2010) dan S1 Teknik Informatika ITS (2010-2014).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen Dalam Negeri dan sebagai Kepala Departemen Dalam Negeri Himpunan Mahasiswa Teknik Computer-Informatika ITS. Selain itu penulis juga menjadi salah satu administrator pada laboratorium Grid Computing Laboratory di Teknik Informatika ITS untuk mendalami tentang jaringan dan pemrograman paralel. Selama kuliah di teknik informatika ITS penulis mengambil bidang minat Komputasi Berbasis Jaringan (KBJ). Komunikasi dengan penulis melalui : ahadyn.arlingga[at]gmail.com.