SISTEM KOMUNIKASI ALTERNATIF PADA DAERAH BENCANA MENGGUNAKAN INFRASTRUKTUR MOBILE AD-HOC NETWORK

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

ARIF HIDAYATULLAH

NIM. 0910683021

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

SISTEM KOMUNIKASI ALTERNATIF PADA DAERAH BENCANA MENGGUNAKAN INFRASTRUKTUR MOBILE AD-HOC NETWORK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

ARIF HIDAYATULLAH

NIM. 0910683021

Skripsi ini telah disetujui dosen pembimbing pada

tanggal 24 Januari 2014

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T., M.Eng.	**Eko Setiawan, S.T., M.Eng.**
NIK. 820809 06 1 1 0084	**NIK. 870610 06 1 1 0256**

LEMBAR PENGESAHAN

SISTEM KOMUNIKASI ALTERNATIF PADA DAERAH BENCANA MENGGUNAKAN INFRASTRUKTUR MOBILE AD-HOC NETWORK

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

ARIF HIDAYATULLAH

NIM. 0910683021

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 24 Januari 2014

Penguji I	Penguji II
Adharul Muttaqin, ST., MT	**Achmad Basuki, ST., MMG., Ph.D**
NIP. 19760121 200501 1 001	**NIP. 19741118 200312 1 002**

Penguji III

Barlian Henryranu Prasetio, ST., MT.

NIK. 82102406110254

Mengetahui

Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 20 Desember 2013

Mahasiswa,

Arif Hidayatullah
NIM.0910683

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat rahmat dan hidayah-Nya, penulis dapat menyelesaikan penyusunan skripsi ini. Tak lupa penulis haturkan shalawat serta salam kepada Rasulullah Nabi Muhammad SAW, keluarga, sahabat-sahabat, dan para pengikutnya, semoga rahmat Allah SWT selalu tercurah untuk kita semua. Aamiin.

Skripsi ini disusun dengan judul "**SISTEM KOMUNIKASI ALTERNATIF PADA DAERAH BENCANA MENGGUNAKAN INFRASTRUKTUR MOBILE AD-HOC NETWORK**". Penulisan skripsi ini disusun untuk memenuhi sebagian syarat menjadi Sarjana Komputer. Dalam penyusunan, skripsi ini telah banyak mendapat bantuan dari berbagai pihak. Ucapan terima kasih penulis sampaikan kepada:

1. Bapak Abdullah Maksum , Ibu Roichah, Luluk Hamidah & iwan, Imam Tazi & saudah, Masruroh & zaenal, Fatkhur Rahman & naning, selaku Bapak, Ibu dan saudara-saudaraku tercinta atas segenap do'a, dukungan dan kasih sayang yang telah diberikan.
2. Wilda, Halfi, Rafi, Albi, Nabil, Irfan, Akmal, Alfa,yafie, Ana, Lia, Riska selaku keponakanku yang ku sayangi.
3. KH. Marzuki Mustamar, KH. Murtadho Amin, KH. Aziz Husein, KH.Khotib, & KH. Ahmad yang telah turut mendoakan penulis.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. dan Bapak Eko Setiawan, S.T., M.Eng. selaku dosen pembimbing yang telah membimbing penulis dalam pengerjaan dan penulisan skripsi ini.
5. Bapak Ir. Sutrisno, M.T. selaku Ketua Program Teknologi Informasi dan Ilmu Komputer dan selaku dosen penasihat akademik penulis.
6. Bapak Drs. Marji, M.T. selaku Ketua Program Studi Informatika/Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer.
7. Bapak Barlian Henryranu Prasetyo, ST., MT.. selaku Kepala Laboratorium Sistem Komputer dan Robotika Program Teknologi Informasi dan Ilmu Komputer.

8. Mas didit dan mas sony yang selalu membantu & menyediakan alat-alat skripsi untuk penulis.
9. Segenap dosen Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segenap ilmu pengetahuan dan perhatian yang diberikan Segenap staff dan pegawai Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segala bantuan yang bersifat administratif.
10. Nadhirotul Auliya atas cinta, kasih sayang, semangat dan perhatian yang luar biasa kepada penulis.
11. Saudara-saudara TIM Futsal TIF-C 09 yang telah berjuang mengeluarkan keringat bersama dalam setiap pertandingan.
12. Bintang Karunia Akbar, Raviqul Haidir Francasmara, Ichsan Wahyudi, Galih Julianto Purnomo, Dani Prasnanto, S.Kom., Reviangga Dika Satyatama, S.Kom., Adien Faishol Habib, S.Kom., Ardy Purnama Putra, Rico Maulana,Aldim Irfani Fikri, Yusuf Oktofani, S.Kom., Arga Suwastika Priwawan Putra S.Kom, Thiery rahman aziz, Darmawan Lahru, Zahrul, Doni atas semua bantuan dan persahabatan yang tak ternilai selama penulis menjalani kuliah.
13. Mohammad Halimi, Yoga Pradana Pamungkas, Antarangga Bhawika Manggala, Delis Sukmawati, Riski James, S.Kom., Sawung, Alan, Nizam yang berjuang bersama di Lab.Siskom dalam menjalani skripsi.
14. Saudara-saudara Kamar Sunan Kalijaga (Shohib, yani, Gus faiz, saiful, gamblis, fiton, ghozi, saiful) & keluarga besar PP. Sabilurrosyad yang memberikan tempat untuk belajar agama & banyak hal.
15. Riski prima, Norman Firdaus, yafie , Imam yang selalu mengingatkan tentang mimpi untuk sukses bersama.
16. Para Alumni OSNEMA yang selalu memberikan semangat dan dukungan sebagai sahabat yang sama-sama berjuang dalam suka dan duka.
17. Seluruh teman-teman TIF C 09 & UB 09 tercinta atas dukungan dan kebersamaannya dari awal perkuliahan sampai akhir.
18. Teman-teman Program Studi Informatika/Ilmu Komputer angkatan 2009 tercinta yang selalu memberikan semangat, dorongan, dan bantuan pikiran.
19. Serta semua pihak yang telah membantu dan memberikan pengalaman berharga bagi penulis selama penulis menjalani masa perkuliahan.

Akhirnya atas segala bantuan semua pihak semoga mendapat balasan yang setimpal dari Allah SWT. Harapan penulis semoga skripsi ini dapat memberikan manfaat pada semua pihak, terutama kepada penulis dan para pengembang yang nantinya akan mengembangkan penelitian dari penulis.

Malang, 20 Desember 2013

Penulis

ABSTRAK

Arif Hidayatullah. 2013. : Sistem Komunikasi Alternatif Pada Daerah Bencana Menggunakan Infrastruktur Mobile Ad-Hoc Network. Skripsi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Sabriansyah Rizqika Akbar, S.T., M.Eng. dan Eko Setiawan, S.T., M.Eng.

Indonesia memiliki tingkat potensi bencana alam yang tinggi yang dapat menyebabkan kerusakan infrastruktur *telekomunikasi*. Di satu sisi, komunikasi pada saat terjadi bencana sangat penting agar penanggulangan bencana dapat berjalan dengan baik.

Solusi untuk menangani masalah diatas adalah dengan menggunakan teknologi *Mobile Ad-Hoc Network (Manet)* untuk membangun sistem komunikasi. Sistem komunikasi diharapkan dapat digunakan pada saat infrastruktur utama mengalami kerusakan sehingga bisa digunakan sebagai alternatif komunikasi.

Salah satu sistem komunikasi adalah model pengiriman pesan dengan menggunakan aplikasi *chat Client-Server*. Aplikasi *chat Client-Server* digunakan pada tiap node dalam *Manet* untuk berkomunikasi dengan *node manet* lain dengan menggunakan pesan berbasis teks.

Dari penelitian yang dilakukan diperoleh hasil bahwa teknologi *manet* cocok digunakan untuk sistem komunikasi pada daerah bencana, karena telah dilakukan uji konektifitasnya yang dapat melakukan self configure dan self healing ketika berada di *signal wifi* yang telah terpasang di tiap *node manet*, meskipun setiap *node* dalam *manet* bersifat *mobile* yang menyebabkan sering terjadinya perubahan topologi.

Kata Kunci : Telekomunikasi, *Mobile Ad-hoc Network* (*Manet*), *Chat Client-Server, signal wifi, node, Manet.*

ABSTRACT

Hidayatullah, Arif. 2013. : Alternative Communication System in Disaster Areas Using Mobile Ad-Hoc Network Infrastructure. Thesis of Informatics/Computer Science Study Program, Information Technology and Computer Science Program, University of Brawijaya. Adviser : Sabriansyah Rizqika Akbar, S.T., M.Eng. dan Eko Setiawan, S.T., M.Eng.

Indonesia has a high potential for natural disasters that can cause damage to the telecommunications infrastructure. On the one hand, communication in the event of a disaster is very important that disaster prevention can work well.

Solution for handling the above problem is by using Mobile Ad - Hoc Network (Manet) technology to build communication systems . The communication system is expected to be used during major infrastructure is damaged, so it can not be used .

One of the communication system is message delivery model using the chat client - server application. Chat client - server application is used at each Manet node to communicate with other Manet nodes by using a text -based message.

From the research done is obtained the result that Manet technology is suitable for communication systems in disaster areas, because it has been done the automated connectivity test when in a wifi signal that has been installed on each Manet node, though each Manet is mobile that cause frequent topology changes .

Keyword : Telecomunication, Mobile Ad-hoc Network (Manet), Chat client-server, wifi signal, Manet node.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia memiliki tingkat potensi bencana alam yang tinggi, adanya bencana yang dapat menyebabkan kerusakan infrastruktur telekomunikasi, sehingga penggunaan telekomunikasi tidak bisa digunakan. Disatu sisi, komunikasi pada saat terjadi bencana sangat penting agar penanganan bencana dapat ditanggulangi dengan baik.

Karena permasalahan tersebut diperlukan infrastruktur jaringan komunikasi alternatif yang diharapkan dapat digunakan pada saat infrastruktur telekomunikasi utama mengalami masalah yaitu menggunakan teknologi *Mobile ad hoc network (Manet)*, *Manet* merupakan tipe jaringan khusus yang mampu melibatkan banyak orang atau peralatan komunikasi tanpa ketergantungan terhadap suatu infrastruktur. Dengan memanfaatkan perangkat *Raspberry Pi* sebagai *node* dalam *Manet*, *Manet* dapat dibangun sebagai jaringan yang *mobile*.

Setiap node dalam *Manet* dapat bebas bergerak secara acak dan berpindah-pindah, dengan demikian topologi jaringan *nirkabel* dapat berubah cepat dan tak terduga. Dengan memanfaatkan antar muka *nirkabel* untuk menghubungkan antara satu node dengan node lainnya, untuk mengimplementasikan *Manet* pada daerah bencana dibutuhkan aplikasi pengiriman pesan berbasis teks untuk sistem komunikasi pada jaringan *Manet*.

Salah satu model pengiriman pesan adalah dengan menggunakan komunikasi *Chat*. Aplikasi *Chat* ini digunakan pada tiap node untuk berkomunikasi dengan node lain menggunakan pesan teks.

Atas dasar uraian diatas, penulisan laporan skripsi ini akan membahas mengenai SISTEM KOMUNIKASI ALTERNATIF PADA DAERAH BENCANA MENGGUNAKAN INFRASTRUKTUR *MOBILE AD-HOC*

NETWORK (MANET). Diharapkan dapat memberikan acuan untuk pemanfaatan teknologi *Manet* untuk pengembangan lebih baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan, maka dirumuskan masalah sebagai berikut:

1. Bagaimana merancang sistem komunikasi *chat* dengan memanfaatkan *Manet*.
2. Bagaimana pengaruh *node* terhadap perubahan topologi yang dinamis.
3. Bagaimana program *chat* melakukan komunikasi melalui infrastruktur *Manet*.

1.3 Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dikemukakan, penelitian ini mempunyai batasan-batasan masalah sebagai berikut:

1. Lingkungan sistem operasi *Arch Linux ARM* yang khusus digunakan untuk hardware *Raspberry Pi*.
2. Diimplementasikan menggunakan jaringan *wireless dongle Edimax EW-7811Un.*
3. Menggunakan empat *node Raspberry Pi* sebagai simulasi perpindahan *node.*
4. Penelitian ini tidak menguji interval waktu protokol routing untuk melakukan konfigurasi secara mandiri.
5. Penelitian ini mengabaikan aspek keamanan jaringan.
6. Aplikasi *Chat* menggunakan bahasa *Python*.

1.4 Tujuan

Tujuan dari skripsi ini adalah membuat sistem komunikasi menggunakan aplikasi *Chat* pada *Manet* yang dibangun menggunakan *Raspberry pi* untuk komunikasi jaringan yang mempunyai topologi dinamis.

1.5 Manfaat

Dengan memanfaatkan teknologi *Manet* menggunakan *Raspberry Pi* untuk sistem komunikasi alternatif pada daerah bencana. Penulisan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut:

- Bagi Penulis
  1. Sebagai media untuk mengimplementasikan ilmu pengetahuan teknologi khususnya di bidang perancangan jaringan dan komunikasi jaringan.
  2. Mendapatkan pengetahuan dan wawasan teknologi Manet untuk komunikasi.
- Bagi pembaca
  1. Mendapatkan wawasan tentang teknologi Manet.
  2. Mendapatkan wawasan terkait sistem komunikasi jaringan menggunakan pemrogaman *socket*.
  3. Dengan adanya penerapan teknologi Manet ini, dapat memudahkan proses komunikasi ketika terjadi suatu bencana.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I : Pendahuluan

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

BAB II : KAJIAN PUSTAKA DAN DASAR TEORI

Menguraikan kajian pustaka dan dasar teori yang mendasari teknologi *manet* dan sistem komunikasi aplikasi *Chat* menggunakan bahasa pemrogaman *Python*.

BAB III : Metode Penelitian dan Perancangan

Menguraikan tentang metode dan langkah kerja yang terdiri dari studi literatur, analisis kebutuhan perangkat keras dan perangkat lunak, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan.

BAB IV : Implementasi

Menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

BAB V : Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VI : Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Dasar atau acuan yang berupa teori-teori atau temuan-temuan melalui hasil berbagai penelitian sebelumnya merupakan hal yang sangat perlu dan dapat dijadikan sebagai *data* pendukung. Salah satu *data* pendukung yang menurut peneliti perlu untuk dijadikan bagian tersendiri adalah penelitian terdahulu yang relevan dengan permasalah yang sedang dibahas dalam penelitian ini. Dalam hal ini, fokus pada penelitian terdahulu yang dijadikan acuan adalah terkait dengan masalah tentang penerapan sistem komunikasi menggunakan teknologi *Manet*. Oleh karena itu, peneliti melakukan langkah kajian terhadap beberapa hasil penelitian berupa skripsi.

Berdasarkan hasil-hasil penelitian yang telah dilakukan menyiratkan bahwa *Manet* adalah teknologi yang cocok digunakan sebagai alternatif untuk membuat infrastruktur telekomunikasi yang dinamis pada daerah bencana.

Tabel 2.1 Kajian Pustaka

NO	Tahun	Judul	Hasil / Temuan	Peneliti	Variabel yang terkait
1	2007	Analisis Kinerja *Voice Over Internet Protocol* pada *Manet*	Memberikan hasil dan gambaran tentang Voip layak dan berhasil dilewatkan pada *Manet*	Tofan Teguh Perkasa	- *Manet* - Analisa Sistem Komunikasi
2	2012	*Vanet* untuk Solusi Komunikasi *Data* di	Teknologi Vanet tepat diterapkan di kawasan bali karena tidak dapat	I Komang Ari Mogi	- *Vanet (Vehicle Ad Hoc Network)* - Komunikasi

		Kawasan Pariwisata Bali	dibangunnya infrastruktur komunikasi karena hukum adat yang membatasi pembangunan di daerah kawasan pariwisata.		*Data*
3	2008	Pengiriman Pesan Secara Berantai Pada Daerah Bencana Terisolasi Menggunakan Teknologi *Manet*	Memanfaatkan Bluetooth yang ada di HP untuk pengiriman pesan berantai menggunakan teknologi *Manet*	Tri Hadiah Muliawati	- *Manet* - Pengiriman Pesan

Dari beberapa contoh hasil penelitian diatas, maka dapat digambarkan beberapa persamaan dan perbedaannya. Persamaan penelitian ini dengan hasil-hasil penelitian sebelumnya adalah pada salah variabel yang digunakan dalam membahas pokok permasalahan, yaitu variable *Manet* dan komunikasi.

Sedangkan, perbedaan antara penelitian ini dengan hasil-hasil penelitian yang ada pada tabel 2.1 adalah pada kaitan pembahasan variabel *Manet* itu sendiri. Pada penelitian ini peneliti menggunakan *routing* protokol *Babel* untuk membangun jaringan *Manet* dan untuk komunikasinya menggunakan aplikasi *Chat*. Sementara itu, pada penelitian di tabel 2.1 menjelaskan *Manet* dengan menggunakan perangkat dan protokol yang berbeda dengan penelitian ini.

2.2 Mobile ad-hoc network

Manet terbentuk dari sekumpulan *node* yang menggunakan antarmuka *nirkabel* untuk melakukan komunikasi antara satu *node* dengan *node* yang lainnya. *Manet* banyak digunakan pada sistem militer, operasi penanganan bencana, layanan darurat, penelitian dan mahasiswa. [NYN-13]

Manet menjadi subjek yang sangat popular untuk penelitian karena adanya laptop dan teknologi *wireless* yang banyak digunakan pada pertengahan akhir tahun 1990-an. Penerapan *Manet* terutama di bidang militer, pada situasi darurat (bencana) yang tidak didukung oleh infrastruktur komunikasi dan penggunaan seperti *sharing file* atau perangkat keras antar beberapa *node* di rumah maupun di perkantoran. Jaringan ini dipilih karena penerapannya yang cenderung lebih murah, cepat, mudah dikonfigurasi dibandingkan jaringan infrastruktur *wireless* maupun jaringan kabel. Dalam *Manet*, setiap *node* tidak hanya berperan sebagai pengirim atau penerima saja, tetapi juga akan memiliki kemampuan layaknya *router* yang meneruskan pesan antar *node* di sekitarnya. Untuk itu dibutuhkan *routing protocol* untuk membantu tiap-tiap *node* melakukannya. *Routing protocol* untuk *Manet* tentunya berbeda dengan *routing protocol* yang biasa diimplementasikan pada jaringan kabel. Hal ini disebabkan sifat *Manet* yang dinamis dan cenderung memiliki topologi yang berubah-ubah, berbeda dengan jaringan kabel yang cenderung tetap.

2.2 *Wireless LAN*

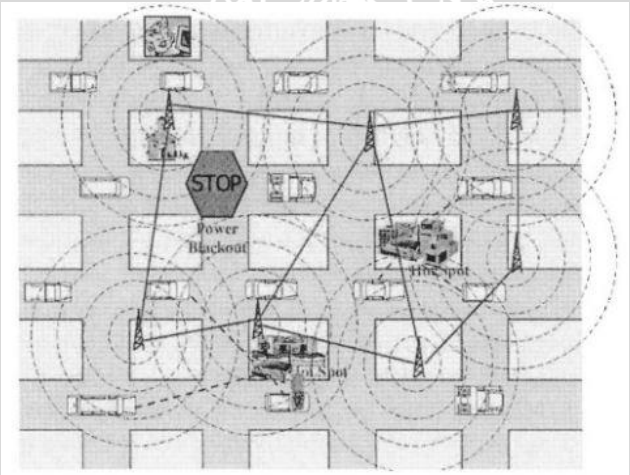
Teknologi *wireless* dapat diartikan teknologi tanpa kabel. Teknologi *wireless* menggunakan udara sebagai media perantara untuk melakukan pertukaran *data*. Teknologi *wireless* tidak hanya diterapkan pada dunia komputer saja tetapi juga pada bidang telekomunikasi. [PAB-10]

2.3 Topologi Jaringan Ad-Hoc

Topologi jaringan *ad-hoc* terdiri dari beberapa *mobile node* yang dapat saling berkomunikasi secara *peer-to-peer* tanpa menggunakan infrastruktur seperti *access point*. Setiap *mobile node* memiliki *wireless network interface* dan saling

berkomunikasi dengan memanfaatkan media *radio*. Contoh *node* pada jaringan *ad-hoc* adalah laptop komputer dan *Personal Digital Assistant (PDA)* yang dapat berkomunikasi secara langsung satu dengan lainnya. *Node-node* pada konfigurasi jaringan *ad hoc* dapat bergerak dengan bebas atau diam pada posisinya. Pada gambar 2.1 diperlihatkan konfigurasi jaringan *ad-hoc* dengan tiga *node* yang dapat berkomunikasi secara langsung.

Setiap *node* pada konfigurasi *ad-hoc* memiliki *coverage* area tertentu. Setiap *node* dapat saling bertukar *data* apabila masih berada di dalam *coverage* area atau dapat pula menggunakan *node* lain untuk mem-*forward data* menuju *node* tujuan seperti yang ditunjukkan pada 2.1. Sehingga dapat dikatakan bahwa setiap *node* pada konfigurasi *ad-hoc* dapat berperan sebagai *host* dan sebagai *router* yang dapat mengarahkan *data* menuju *node* tujuan.



Gambar 2.1 Jaringan Ad-Hoc
Sumber : [PSH-05]

2.4 Protokol *routing* pada *Manet*

Protokol *routing* pada jaringan *ad-hoc* menjadi suatu permasalahan yang menantang untuk diteliti semenjak sebuah *node* bisa bergerak secara bebas (acak). Pada ad-hoc ada dua tipe protokol *routing*, yaitu :

1. Proaktif: *Babel, Destination Sequenced Distance Vector (DSDV), Cluster Switch Gateway Routing (CSGR), Wireless Routing protocol (WRP), Optimized Linkstate (OLSR).*
2. Reaktif: *Dynamic Source Routing (DSR), Ad-hoc On-demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA), Associativy Based Routing (ABR), Signal Stability Routing (SSR).*

Ada juga pendekatan pada *protocol routing hybrid* yang mengkombinasikan antara kedua tipe protokol *routing*, proaktif dan reaktif, contohnya *Zone Routing Protocol (ZRP).* [STA-04]

2.5 Protokol routing babel

Babel adalah *distance vector routing protocol* untuk protokol jaringan *packet-switched internet* yang dirancang untuk menjadi kuat dan efisien di kedua jaringan *mesh nirkabel* dan jaringan kabel, terinspirasi oleh *DSDV*, yang dirancang untuk menjadi kuat dan efisien baik dalam jaringan yang menggunakan berbasis *prefix routing* dan dalam jaringan menggunakan *flat routing* (jaringan *mesh*), fitur utama yang membuat *Babel* cocok untuk jaringan tidak stabil adalah bahwa tidak seperti *routing* protokol *RIP*, hal itu tidak menyebabkan *routing loop* dan *blackhole* selama *reconvergence*. Bahkan setelah peristiwa mobilitas terdeteksi, jaringan *Babel* biasanya tetap *loop-free*.

Babel cepat *reconvergen* ke konfigurasi yang melindungi *loop* free dan keterhubungan terhadap jaringan, tetapi belum tentu optimal, dalam banyak kasus, operasi ini tidak memerlukan pertukaran paket sama sekali, dan dalam kasus terburuk mengambil sejumlah bursa paket yang sebanding dengan diameter jaringan. *Babel* kemudian perlahan-lahan menyatu, dalam waktu pada skala menit, ke konfigurasi yang optimal. Lebih tepatnya, *Babel* memiliki sifat sebagai berikut:

1. ketika setiap awalan berasal oleh paling banyak satu *router*, *Babel* tidak pernah menderita *routing loop.*
2. ketika awalan berasal oleh beberapa *router*, *Babel* sesekali dapat membuat *routing* lingkaran transien untuk awalan ini, lingkaran ini menghilang dalam waktu yang proporsional dengan diameternya, dan tidak akan

pernah lagi *router* berpartisipasi dalam *routing loop* untuk awalan yang sama.

3. setiap *routing* yang mengalami *blackhole* yang mungkin muncul setelah peristiwa *mobilitas* diperbaiki dalam waktu paling sebanding dengan diameter jaringan.

Babel memiliki ketentuan untuk kualitas estimasi link dan metrik yang cukup sewenang-wenang. Ketika dikonfigurasi sesuai, *Babel* dapat menerapkan *path routing* terpendek, atau mungkin menggunakan berbasis *metrik* pada statistik *packet* loss. *Node Babel* akan berhasil mendirikan sebuah asosiasi bahkan ketika mereka dikonfigurasi dengan parameter yang berbeda. Sebagai contoh, sebuah *mobile node* yang rendah pada baterai dapat memilih untuk menggunakan konstanta waktu yang lebih besar (*Hello message* dan interval pembaruan, dll) dari sebuah *node* yang memiliki akses ke listrik di dinding. Sebaliknya, simpul yang mendeteksi tingkat *mobilitas* yang tinggi dapat memilih untuk menggunakan konstanta waktu yang lebih kecil. Kemampuan untuk membangun jaringan heterogen seperti membuat *Babel* khususnya disesuaikan dengan lingkungan *nirkabel*. *Babel* adalah protokol *routing* hybrid, dalam arti bahwa hal itu dapat membawa *rute* untuk beberapa protokol lapisan jaringan (*IPv4* dan *IPv6*). [JCH-11]

2.5.1 *Transmisi* pesan dan penerimaan pesan

Pertukaran pesan pada *Babel*, satu atau lebih pesan *Babel* yang ditambahkan untuk membentuk paket *Babel*, yang dikirim dalam *datagram UDP* tunggal. Alamat sumber dari paket *Babel* selalu alamat *unicast link-local*. Paket *Babel* dapat dikirimkan ke alamat *multicast link-local* atau ke (*link-local*) alamat *unicast*. Dalam operasi normal, seorang pembicara *Babel* mengirimkan paket *multicast* dan *unicast* baik dengan tetangganya. Dengan pengecualian pesan *Hello* dan *acknowledgment*, semua pesan *Babel* dapat dikirim ke alamat baik *unicast* atau *multicast*, dan semantik mereka tidak tergantung pada apakah tujuan adalah alamat *unicast* atau *multicast*. Oleh karena itu, seorang pembicara *Babel* tidak perlu menentukan alamat tujuan dari sebuah paket yang diterima untuk menafsirkannya.

Sebuah jumlah *jitter* diterapkan pada pesan yang dikirim oleh *Babel*, keluar pesan yang *buffered*, dan harus dikirim dengan penundaan acak kecil. Hal ini dilakukan untuk dua tujuan menghindari sinkronisasi beberapa pembicara *Babel* melalui jaringan (*jitter*), dan memungkinkan untuk *agregasi* beberapa pesan ke dalam satu paket. *Delay* yang tepat dan jumlah *jitter* diterapkan pada pesan tergantung pada apakah pesan sangat mendesak atau tidak. Pesan *acknowledment* harus dikirim sebelum batas waktu yang ditentukan dalam permintaan yang sesuai. Kelas tertentu pesan pembaruan yang ditentukan harus dikirim pada waktu yang tepat. Kelas permintaan tertentu dan memperbarui pesan harus dikirim pada waktu yang tepat. [JCH-11]

2.5.2 Proses *Route Babel*

Tabel *rute* di indeks tiga kali lipat dari bentuk (*prefix, plen, neighbour*), dan setiap entri tabel *rute* berisi *data* sebagai berikut:

1. *Advertised prefix (prefix, plen).*
2. *Neighbour* yang di *advertised* pada *rute* ini.
3. *Metrik* dengan *rute* ini di *advertised* oleh tetangga, yang dikenal sebagai referensi *metrik rute* itu, atau *0xFFFF* (tak terhingga) untuk *rute* yang baru ditarik.
4. Nomor urutan dengan *rute* yang telah di *advertised*.
5. alamat hop berikutnya dari *rute* ini.
6. Sebuah *flag* yang menunjukkan apakah *rute* ini dipilih, yaitu apakah saat ini sedang digunakan untuk meneruskan dan yang di *advertised*.
7. Ada satu waktu terkait dengan setiap entri tabel *rute*, *rute* berakhirnya *timer*. Hal ini dijalankan dan *reset*. [JCH-11]

2.5.3 Pemeliharaan Tabel *Routing*

Secara konseptual, *update Babel* adalah *quintuple* (*prefix, plen, Router-Id , seqno, metric*), di mana *(prefix, plen)* adalah *prefix* untuk *rute* yang *advertised*, *Router-Id* adalah nomor *router* yang berasal dari *update route*, *seqno* adalah nomor *advertised* yang berurutan, tanpa pengurangan *(modulo 2 ^ 16) integer* yang didefinisikan oleh *router* berasal, dan mengumumkan *metrik*. Sebelum

diterima, pembaruan tersebut diperiksa terhadap kondisi kelayakan, suatu kondisi yang memastikan bahwa *rute* tidak membuat *routing loop (dual)*. Jika kondisi kelayakan tidak benar, pembaruan diabaikan atau diperlakukan sebagai pencabutan, tergantung pada beberapa kondisi lain. Jika kondisi kelayakan benar, maka *update* tidak mungkin menyebabkan *routing loop* , dan *update* diterima. Sebelum *advertising rute*, *update* tabel *node Babel* sumber dengan informasi yang akan diperlukan untuk mengevaluasi kondisi yang terjadi. [JCH-11]

2.5.4 Perhitungan metrik

Metrik Sebuah *rute* yang dihitung dari referensi metrik-metrik tetangga yang di *advertised*, dan biaya link penyebaran tetangga. Sama seperti perhitungan link, perhitungan metrik dianggap sebagai masalah kebijakan local, sejauh *Babel* yang bersangkutan, fungsi M (c, m) yang digunakan untuk komputasi metrik dari biaya neighbour dan referensi *rute* ini metrik harus hanya memenuhi persyaratan sebagai berikut:

1. jika c adalah tidak terbatas, maka M (c, m) tidak terbatas.
2. M adalah tidak berubah: M (c, m)> m.
3. Metrik harus memenuhi kondisi berikut:
   M adalah *isotonik*: jika m <= m' maka M (c, m) <= M (c, m').

Perhatikan bahwa sementara *monotonicity* yang ketat sangat penting untuk integritas jaringan (*loop routing* yang persisten dapat muncul jika tidak benar), isotonisitas tidak, jika tidak benar, *Babel* masih akan konvergen ke tabel *routing* lokal secara optimal, tetapi tidak berarti mencapai optimum *global* (pada kenyataannya, seperti *optimum global* bahkan mungkin tidak ada). [JCH-11]

2.5.5 *Route Selection*

Pemilihan *rute* adalah proses dimana satu *rute* untuk awalan yang diberikan dipilih untuk digunakan untuk paket forwarding dan akan *readvertised* ke tetangga sebuah *node*. *Babel* ini dirancang untuk memungkinkan kebijakan pemilihan *rute* yang fleksibel. Sejauh kebenaran protokol yang bersangkutan, kebijakan pemilihan *rute* hanya harus memenuhi sifat-sifat sebagai berikut:

1. *rute* dengan metrik yang tak terbatas tidak pernah dipilih.
2. *rute* yang tidak layak tidak pernah dipilih.

Bagaimanapun, *Babel* tidak secara alami menjamin stabilitas *routing*, dan mengkonfigurasi kebijakan pemilihan *rute* bertentangan pada *router* yang berbeda dapat menyebabkan persisten *osilasi rute*. Mendefinisikan kebijakan pemilihan *rute* yang baik untuk *Babel* merupakan masalah penelitian yang masih terbuka. Pemilihan *rute* dapat mempertimbangkan beberapa kriteria yang saling bertentangan, yaitu:

1. *rute* dengan metrik kecil sebaiknya diutamakan dibanding *rute* dengan metrik besar.
2. perubahan *Router-Id* harus dihindari.
3. *rute* melalui tetangga stabil sebaiknya diutamakan dibanding *rute* melalui yang tidak stabil.
4. *rute* yang stabil sebaiknya diutamakan dibanding yang tidak stabil.
5. beralih *hop* berikutnya harus dihindari.

Sebuah strategi sederhana adalah untuk memilih *rute* layak dengan metrik terkecil, dengan sejumlah kecil *hysteresis* diterapkan untuk menghindari beralih *Router-Id*. Setelah prosedur pemilihan *rute* dijalankan, *update* dipicu dan permintaan dikirim. [JCH-11]

2.5.6 *Sending Updates*

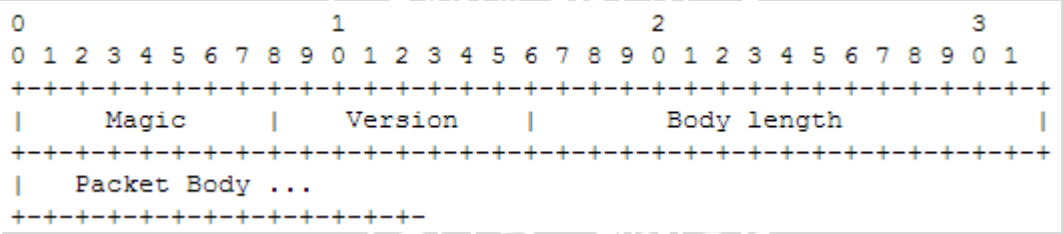
Babel menyebarkan dengan tetangganya set dari *rute* yang dipilih. Biasanya, hal ini dilakukan dengan mengirimkan satu atau lebih paket *multicast* berisi pesan *Update* pada semua *interface* yang terhubung, namun pada teknologi *multicast link* di mana secara signifikan lebih mahal daripada *unicast*, *node* mungkin memilih untuk mengirim beberapa salinan pembaruan dalam paket *unicast* ketika jumlah tetangga kecil. Selain itu, dalam rangka untuk memastikan bahwa setiap *blackhole* yang andal dibersihkan secara tepat waktu, *node Babel*

mengirimkan retraksi (*update* dengan metrik tak terbatas) untuk setiap prefiks baru ditarik.

Jika *update* untuk *rute* disuntikkan ke dalam domain *Babel* oleh *node* lokal (misalnya alamat antarmuka lokal, awalan jaringan langsung terpasang, atau menyebarkan dari protokol *routing* yang berbeda), *Router-Id* diatur ke *id* lokal, metrik diatur untuk beberapa nilai terbatas sewenang-wenang (biasanya 0), dan *seqno* diatur ke nomor urut *router* lokal. Jika pembaruan untuk *rute* pembelajaran dari *Babel* speaker yang lain, *Router-Id* dan nomor urut yang disalin dari entri tabel *routing*, dan metrik dihitung. [JCH-11]

2.5.7 *Packet Format*

Sebuah paket *header Babel* terdiri dari empat oktet, diikuti oleh urutan pesan *Babel*.



Gambar 2.2 *packet format*

Sumber: [JCH-11]

Fields :

Magic Berubah-ubah tetapi hati-hati memilih nilai 42 (desimal), paket dengan oktet pertama berbeda dari 42 harus diabaikan.

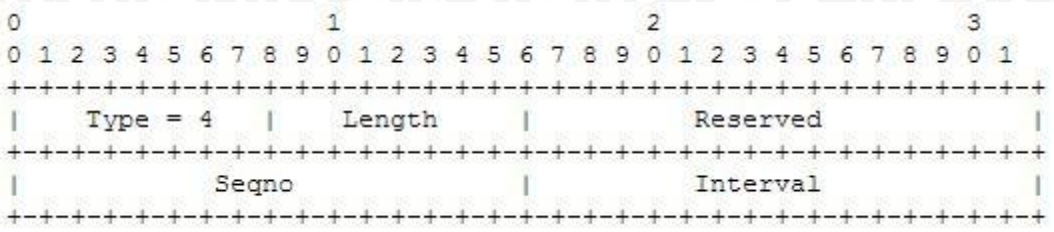
Version Dokumen ini menetapkan versi 2 dari protokol *Babel*. Paket dengan oktet kedua berbeda dari 2 harusdiabaikan.

Body Length Panjang dalam header octet mengikuti header paket.

Body *body packet* adalah sebuah urutan pesan.

Setiap *data* yang mengikuti *body* harus diabaikan. [JCH-11]

2.5.8 *Hello Message*



Gambar 2.3 Pesan *Hello*

Sumber: [JCH-11]

TLV ini digunakan untuk penemuan tetangga dan untuk menentukan biaya metrik link ini.

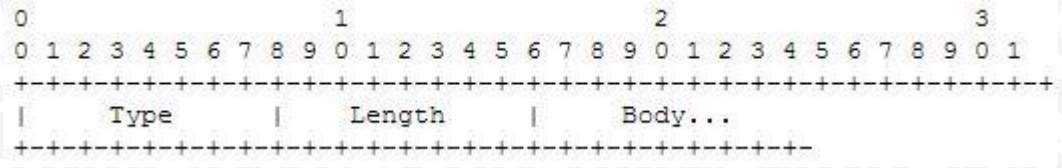
Fields :

Type	Di set 4 untuk mengindikasikan sebuah pesan *Hello* TLV.
Length	Panjang *body*, jenis ekslusive dan panjang *field*.
Reserved	*Field* ini dikirim sebagai 0, dan harus diabaikan pada penerimaan.
Seqno	Nilai dari pengirim *node* hallo seqno untuk *interface* ini.
Interval	Batas atas, dinyatakan dalam centiseconds, pada waktu setelah *node* pengirim akan mengirim *Hello* TLV baru. Ini tidak harus 0.

Karena ada seqno tunggal yang membalas untuk semua *Hello* dikirim oleh *node* yang diberikan melalui antarmuka yang diberikan, TLV ini HARUS dikirim ke tujuan multicast. Untuk menghindari diskontinuitas besar dalam kualitas link, beberapa TLV *Hello* tidak harus akan dikirim dalam paket yang sama. [JCH-11]

2.5.9 Message Format

Dengan pengecualian pada *Pad1*, semua pesan harus mengikuti struktur berikut :



Gambar 2.4 *Message Format*

Sumber: [JCH-11]

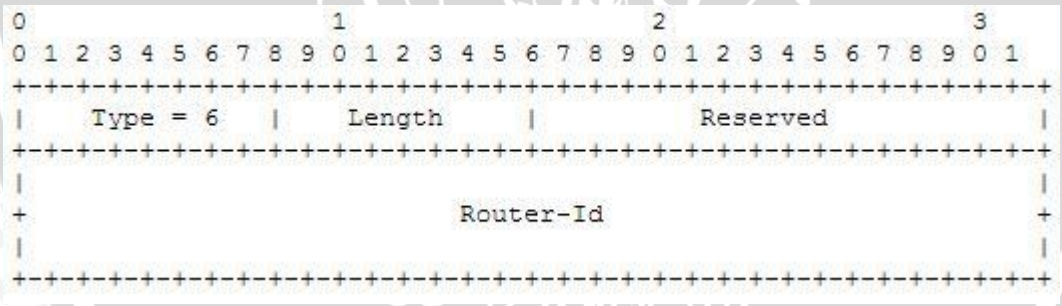
Fields:

Type Kolom ini menentukan jenis pesan.

Length Panjang *body*, *type* eksklusif dan panjang *field*. Jika *body* lebih panjang dari panjang yang diharapkan dari suatu jenis pesan, *data* tambahan harus diam-diam diabaikan.

Body *Body* pesan, penafsiran yang tergantung pada jenis pesan. Jenis pesan yang tidak diketahui harus diabaikan. [JCH-11]

2.5.10 Router ID



Gambar 2.5 *Router* ID

Sumber: [JCH-11]

Sebuah pesan *Router-Id* menetapkan *Router-Id* yang tersirat oleh *update* pesan berikutnya.

Fields:

Type Mengatur sampai 6 untuk menunjukkan pesan *Router-Id* .

Length Panjang *body*, eksklusif *type* dan Panjang *field*.

Reserved Bidang ini dikirim sebagai 0, dan harus diabaikan pada penerimaan.

Router-Id *Field* ini berisi *Router-Id* untuk *rute* disebarkan di dalam *update* pesan berikutnya. [JCH-11]

2.6 Chating

Chatting dalam bahasa Indonesia berarti ngobrol atau berbicara dua arah antara satu atau beberapa orang. Di dalam dunia komputer, *Chatting* berarti berbicara dengan orang lain dengan menggunakan komputer. Suara yang dihasilkan, biasanya digantikan dengan teks yang diketik. Namun dengan berkembangnya *multimedia* dengan komputer, *Chatting* tidak hanya dengan menggunakan teks, tapi bisa juga dengan menggunakan suara dan *video*. Lebih sempit lagi, pengertian *Chatting* di dalam dunia *internet*. Di sini pengertian *Chatting* adalah menggunakan *internet* untuk berbicara dengan orang lain.

Menurut jumlah orang yang berbicara, *Chatting* dapat dibagi menjadi dua, yaitu group *Chat* dan *Private Chat*. Group *Chat* adalah *Chatting* yang melibatkan lebih dari dua orang. Biasanya orang-orang ini berkumpul di dalam suatu *Chat room* atau *channel*, tempat di mana mereka bisa berinteraksi. Apabila satu orang mengirimkan suatu pesan, maka seluruh orang yang berada di *Chat room* atau *channel* tersebut bisa membacanya. Sedangkan *Private Chat* tidak demikian. *Private Chat* hanya melibatkan dua orang. Jadi hanya ada orang tertentu yang dapat membaca pesan kita. [MFH-10]

2.7 Pemrograman *Socket* dengan *TCP* di *Python*

Pada aplikasi berbasis jaringan, proses berjalan pada dua mesin yang berbeda yang berkomunikasi satu sama lain melalui sebuah *socket*. Dianalogikan bahwa proses adalah sebuah rumah, dan *socket* merupakan sebuah Pintu dari rumah. Dan seorang pengembang perangkat lunak hanya memiliki kontrol yang sangat sedikit dalam sisi *layer transport*. Sisi *Server* harus selalu siap dalam

melayani permintaan klien, dan proses pada klien akan melakukan inisiasi koneksi *TCP* ke *Server*. Ketika klien membuat soket, klien harus melakukan spesifikasi alamat dari sebuah proses yang dijalankan pada sisi *Server* dengan memberikan nama atau alamat IP dan juga nomor *port* pada sisi *Server*.

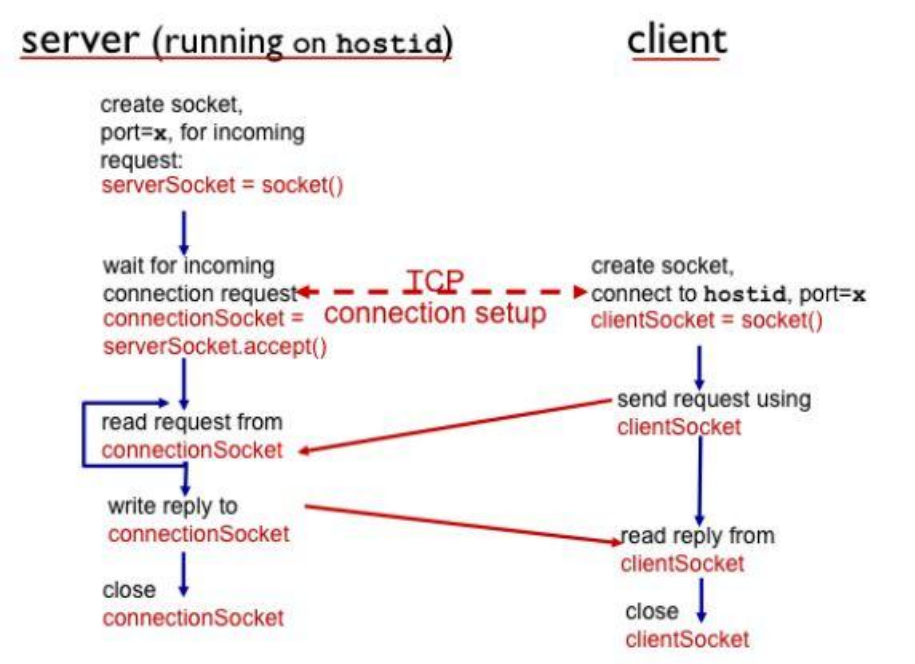
Ketika proses *three way handshaking* terjadi, klien akan melakukan proses "mengetuk pintu" pada *Server* proses *(welcome socket*). Ketika *Server* mendengar ketukan pintu tersebut, *Server* akan membuat *socket* baru (*connection socket*) yang didedikasikan untuk melayani klien. Pada pemrograman jaringan komputer, terdapat istilah *stream* yang merupakan sebuah urutan karakter yang mengalir menuju atau keluar pada sebuah proses. *Input Stream* merupakan sumber input proses seperti *keyboard.*

Perlu menjadi perhatian bahwa ketika sebuah proses dibuat pada masing-masing *end-system* terdapat dua buah *socket* yang diciptakan disisi *Server* yaitu *welcoming socket* yang digunakan untuk melakukan proses *listen dan three way handshake* pada sisi klien, dan *connection socket* yang digunakan untuk saling mempertukarkan pesan.

Langkah-langkah ketika membuat sebuah program dengan *socket* ditunjukkan pada gambar 2.6, langkah-langkahnya adalah sebagai berikut:

1. Sisi *Server* membuat welcoming *socket* pada *port* kesekian (bukan pada *wellknown port*) , yang akan menunggu setiap permintaan yang nantinya akan diberikan oleh *Client.*
2. *Server* menunggu permintaan dari klien, ketika sisi klien membuat *socket* dan menghubungi *Server* (bisa melalui alamat *IP* maupun *hostname*) melalui *port* yang telah diberikan, *Server* akan membuat *connection socket* yang akhirnya menghubungkan *TCP connection* antara klien dengan *Server*.
3. Klien mengirimkan request menggunakan *Client socket*, dan *Server* membaca permintaan klien pada *connection socket*.
4. *Server* membalas permintaan klien melalui *connection socket*, dan sisi klien menerimanya melalui *Client socket*.

5. *Server* menutup *connection socket* dan *Client* menutup *Client socket*.



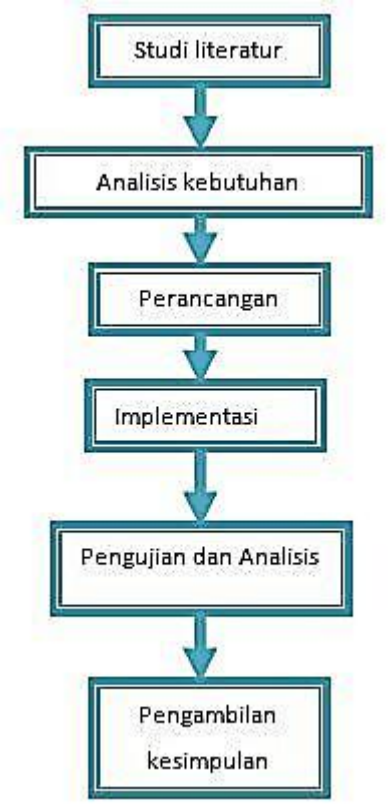
Gambar 2.6 pemrogaman *socket*

Sumber: [KKR-00]

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Bab yang menjelaskan langkah-langkah yang akan ditempuh dalam penyusunan skripsi, yaitu perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.



Gambar 3.1 *Flowchart* Metode Penelitian

3.2 Studi Literatur

Studi literatur dilakukan untuk mempelajari dan memahami konsep yang terkait dengan SISTEM KOMUNIKASI ALTERNATIF BENCANA MENGGUNAKAN TEKNOLOGI *MANET*. Studi literatur yang dilakukan adalah

mengenai karakteristik, parameter, serta teori pendukung lain yang menunjang dalam penulisan skripsi ini. Teori pendukung tersebut di dapat dari buku, jurnal, paper dan internet.

3.3 Penyusunan Dasar Teori

Penyusunan dasar teori dilakukan setelah mendapatkan referensi yang tepat untuk mendukung penulisan penelitian ini. Teori-teori pendukung tersebut meliputi:

1. Kajian pustaka
2. *Mobile Ad-Hoc Network*
3. *Wireless LAN*
4. *Topologi jaringan ad-hoc*
5. *Protokol routing pada Manet*
6. Protokol *Routing Babel*
    a. *Transmisi* pesan dan penerimaan pesan
    b. Proses *Route Babel*
    c. Pemeliharaan Tabel *Routing*
    d. Perhitungan *metric*
    e. Route Selection
    f. Sending Updates
    g. Packet Format
    h. Hello Message
    i. Message Format
    j. Router ID
7. *Chatting*
8. Pemrograman *Socket* dengan *TCP* di *Python*

3.4 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk membahas bagaimana sistem harus menyelesaikan masalah pada sistem yang akan dibangun. Analisis kebutuhan

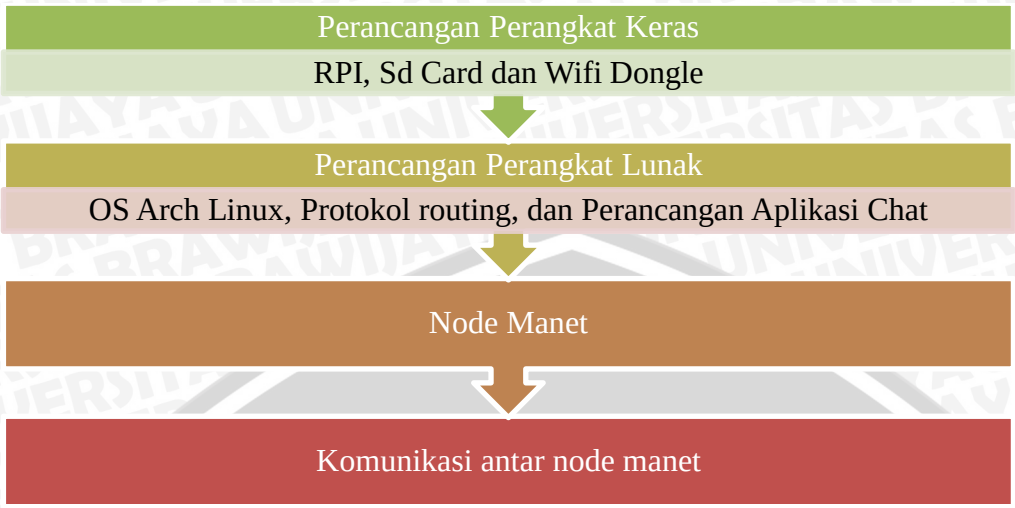
dilakukan dengan mengidentifikasi kebutuhan sistem. Analisis kebutuhan tersebut meliputi, kebutuhan perangkat keras dan kebutuhan perangkat lunak.

Gambar 3.2 Diagram kebutuhan sistem

Kebutuhan sistem yang diperlukan dalam penelitian ini adalah *Raspberry Pi*, *SD card* untuk storage system, dan *wifi* sebagai perangkat nirkabel untuk media komunikasi menggunakan *signal radio*, sistem operasi yang digunakan *operating system arch linux ARM* yang khusus digunakan untuk *Raspberry Pi*, di dalam sistem operasi *arch linux* ini protokol *routing babel* di *install* untuk membangun infrastruktur *Manet*. Aplikasi *Chat* digunakan untuk komunikasi pengiriman pesan berbasis teks.

3.5 Perancangan Sistem

Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan yang meliputi kebutuhan perangkat keras dan kebutuhan perangkat lunak. Setelah kebutuhan sistem terpenuhi selanjutnya adalah perancangan perangkat keras dan perancangan perangkat lunak. Seperti yang ditunjukkan dalam gambar 3.3.

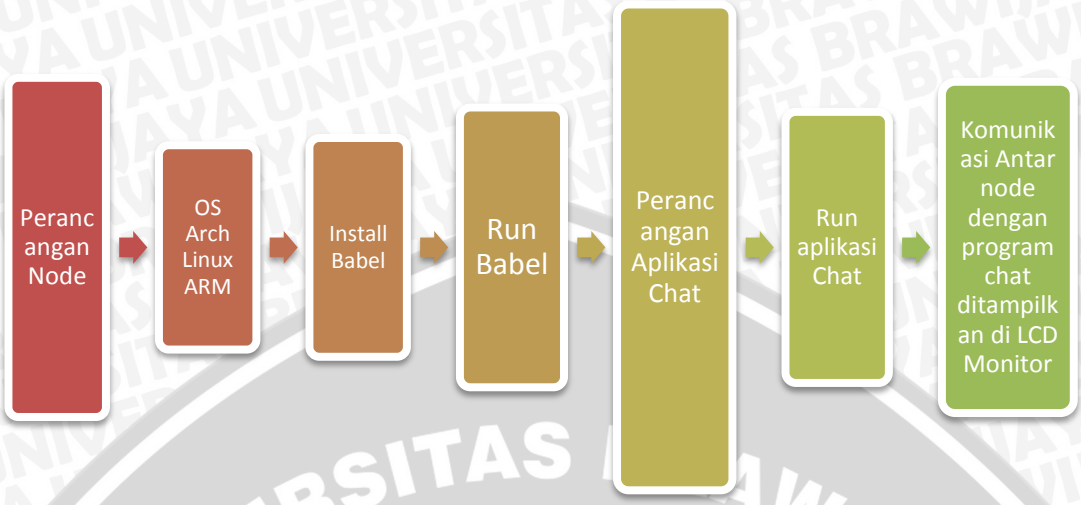


Gambar 3.3 Alur perancangan Sistem secara umum

Pada gambar 3.3 dijelaskan mengenai perancangan sistem secara keseluruhan, dimulai dari merancang perangkat keras sistem, yang terdiri dari *Raspberry Pi, SD Card* sebagai *storage* sistem dan *Wifi Dongle edimax*. Setelah merancang perangkat keras selesai, kemudian merancang perangkat lunak yang terdiri dari sistem operasi Arch linux, perancangan protokol *routing* yang menggunakan protokol *routing babel*, sedangkan perancangan aplikasi *chat* menggunakan aplikasi *Client-Server chat* untuk mengirim pesan teks antar node. Dari hasil merancang perangkat keras dan perangkat lunak, menghasilkan node *manet* yang bisa berfungsi sebagai *transmitter, receiver dan router*.

3.6 Implementasi

Implementasi merupakan penjelasan mulai dari perancangan program sampai hasil akhir yang di tampilkan dalam LCD Monitor. Seperti yang di gambarkan dalam gambar 3.4.



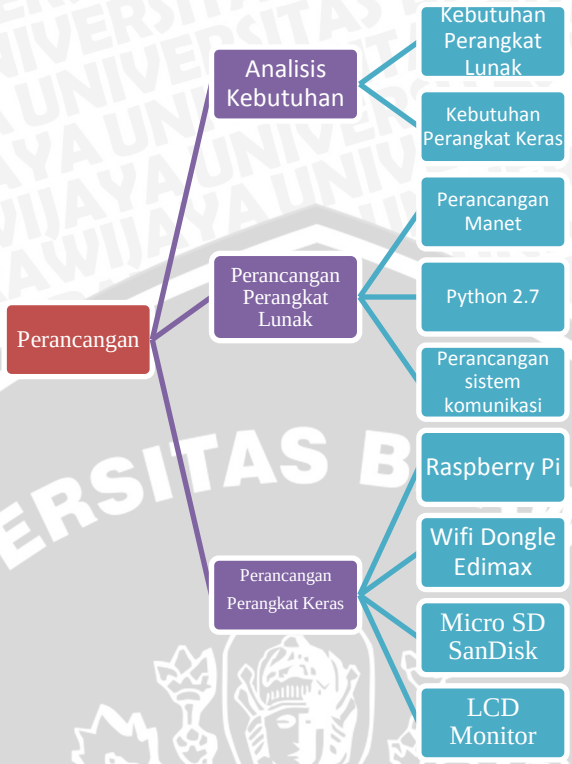
Gambar 3.4 Gambaran implementasi secara umum.

Implementasi dilakukan dengan mengacu pada perancangan sistem.Langkah implementasi meliputi:

1. Perancangan *node (Raspberry Pi, SD Card* dan *Wifi dongle).*
2. Tanamkan sistem operasi *arch linux ARM* pada *SD Card*.
3. *Install protocol routing babel.*
4. Jalankan *protocol routing babel* untuk membangun *Manet.*
5. Perancangan aplikasi *Chat* untuk komunikasi antar *node* berbasis teks.
6. Jalankan aplikasi *Chat.*
7. Lakukan pengiriman pesan teks antar *node* menggunakan aplikasi *Chat* yang akan ditampilkan pada *LCD Monitor*.

3.7 Perancangan

Perancangan jaringan *manet* dengan menggunakan *routing protokol babel* untuk sistem komunikasi. Perancangan jaringan dan sistem dilakukan sesuai dengan subbab 3.4 yaitu tahap analisis kebutuhan sistem, yang meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Gambaran umum perancangan akan dijelaskan dalam pohon di bawah ini.



Gambar 3.5 Pohon perancangan

3.8 Analisis Kebutuhan perangkat keras / lunak

Analisis kebutuhan bertujuan untuk mengidentifikasi kebutuhan yang terlibat dan diperlukan untuk membangun sebuah jaringan *Manet*. Analisis kebutuhan juga dilakukan untuk mengetahui kondisi lapangan, sehingga dapat diketahui implementasi yang nantinya diterapkan dalam simulasi baik desain sistem maupun perangkat yang akan digunakan.

3.8.1 Analisis Kebutuhan Perangkat Keras

a. *Raspberry Pi* digunakan sebagai *node Manet*, yang terinstal protokol *routing babel* dan aplikasi *Chat*.
b. *Wifi dongle edimax EW-7811Un* untuk menghubungkan dengan perangkat *nirkabel* lainnya atau *node Manet* lain.
c. *Micro SD SanDisk 8 GB* sebagai tempat penyimpanan *data* dan *operating system Arch linux* pada *Raspberry Pi*.

d. *LCD Monitor* digunakan untuk menamPilkan komunikasi pengiriman pesan teks antar *node Manet*.

3.8.2 Analisis Kebutuhan Perangkat Lunak

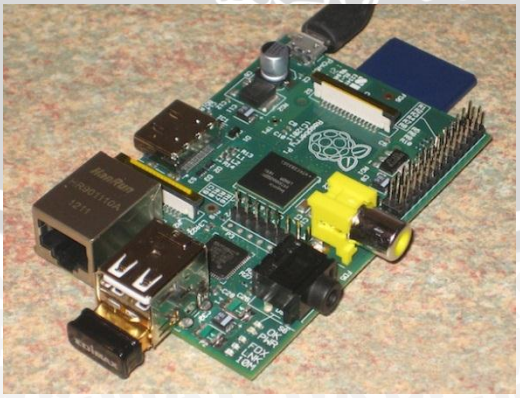
a. Protokol *routing babel* digunakan untuk membuat jaringan *Manet.*
b. *Python 2.7* untuk *compiler* aplikasi *Chat.*
c. Aplikasi *Chat* untuk sistem komunikasi antar *node* dengan menggunakan *Chat* lewat terminal *linux.*

3.9 Perancangan Perangkat Keras/Lunak

Pada penelitian ini perancangan menjelaskan tentang perancangan perangkat keras dan perangkat lunak yang akan diimplementasikan berdasarkan analisa kebutuhan sistem. Terdapat empat bagian yaitu perancangan *node,* menanamkan *OS arch linux ARM* pada *SD Card,* menginstal protokol *routing babel,* dan perancangan aplikasi *Chat.*

3.9.1 Perancangan *node*

Seperti yang sudah disebutkan di dalam analisis kebutuhan perangkat keras, pada perancangan perangkat keras terdapat *Raspberry Pi*, *Micro SD SanDisk* dan *Wifi dongle edimax EW-7811Un*. Mikro *SD* berfungsi sebagai *storage Raspberry* untuk menyimpan sistem operasi dan penyimpanan data. *Node Raspberry* dipasang *wifi* sebagai perangkat *nirkabel* untuk membangun jaringan *Manet* yang berfungsi untuk menghubungkan dengan node lainnya.



Gambar 3.6 Raspberry Pi

3.9.2 OS *Arch linux* ARM

Setelah pemasangan perangkat keras selesai, langkah selanjutnya adalah menginstal sistem operasi untuk *Raspberry pi*, sistem operasi yang digunakan adalah sistem operasi *Arch linux*. *Arch linux* merupakan distribusi yang mengadaptasi prinsip *keep-it-simple-stupid* dan tidak berbasis distribusi manapun. Sangat ringan dan cepat karena sistem operasi dalam keadaan kosong, dari sinilah dapat membangun dan memasang paket-paket sesuai dengan kebutuhan.

3.9.3 Protokol *Routing Babel*

Setelah *Raspberry pi* sudah di *install operating system Arch linux ARM*, langkah selanjutnya adalah menginstall protokol *routing babel*. Proses *installing* protokol *routing babel* ini membutuhkan paket-paket yang harus di *install* terlebih dahulu yang akan dijelaskan dalam bab perancangan, Protokol routing babel adalah protokol *routing* yang termasuk jenis *distance vector routing* protokol untuk membuat jaringan *Manet*, kelebihan lain dari protokol routing ini adalah dapat di pakai pada *IPv4* maupun *IPv6 (Dualstack)*. Selain itu, protokol *routing babel* cocok digunakan pada *embedeed system Raspberry pi* karena hanya membutuhkan persyaratan *CPU* dan memori yang kecil.

Tabel 3.1 Algoritma protokol routing babel

Algoritma protokol *routing babel*
- Deteksi semua link yang terhubung di dalam *node* - Catat semua informasi link ke dalam table *routing* - Tabel *routing* yang dimiliki masing-masing *router* akan berisi alamat jaringan yang terhubung langsung dengan *router* tersebut. - Secara periodik masing-masing *router* saling bertukar informasi sehingga isi tabel *routing* terisi lengkap (*converged)* - Jika terjadi perubahan topologi jaringan, maka *router* akan segera meng*update* informasi *routing*. - Proses *update* tiap-tiap *router* dilakukan secara bertahap. - Jika letak *router* jauh, maka dalam proses penerimaan informasi tentang

perubahan jaringan akan lama pada suatu lokasi.

- Beritahukan Setiap link yang terhubung di dalam *node* ke *node* tetangga

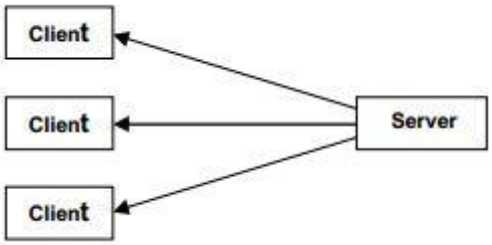
3.9.4 Aplikasi *Chat*

Koneksi antar *node* yang dibentuk untuk aplikasi *Chat* adalah koneksi dari *socket* ke *socket*. *Socket* adalah dua buah nilai yang mengidentifikasi setiap *endpoint* sebuah alamat *IP* dan sebuah nomor *port* . Analogikan *socket* sebagai Pintu, untuk bisa berkomunikasi maka Pintu kedua *node* harus terbuka. *Client*-yang dalam hal ini harus lebih proaktif untuk membuka koneksi terlebih dahulu, *Client* harus "mengetuk Pintu" dahulu pada *Server*. *Server* memonitor "tiap ketukan" atau permohonan koneksi *Client*. Setelah *Server* menerima atau "membuka Pintu", maka sebuah koneksi baru bisa terbentuk sehingga baik *Client* maupun *Server* sudah bisa berkomunikasi dan saling bertukar data.

Konsep untuk memahami pembentukan koneksi *Client-Server* dapat diterangkan sebagai berikut:

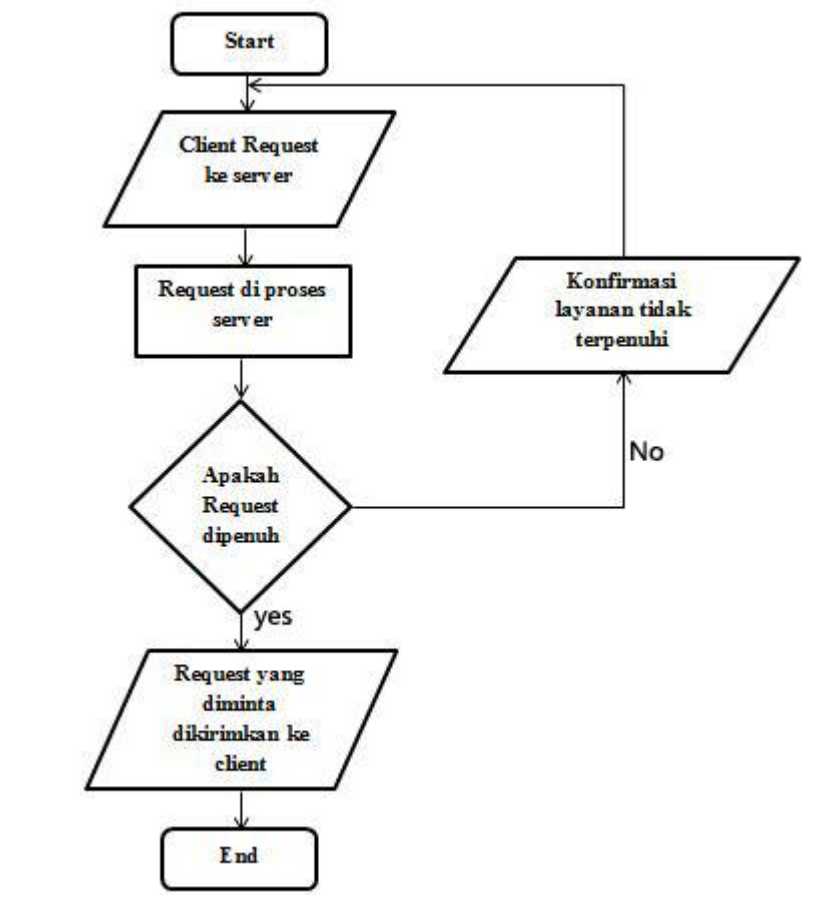
1. *Server* membuat sebuah *socket* dengan menggunakan karakter unik (dengan penentuan alamat IP dan nomor port), yang dapat diidentifikasi dan ditemukan oleh *Client*, pada saat ini *Server* telah memasuki kondisi listening. Kondisi listening adalah keadaan dimana *Server* dalam kondisi siap untuk menerima permintaan servis dari *Client*.
2. *Client* membuat *socket*, mencari alamat *socket Server* dan kemudian menyambungkannya untuk menginisialisasi sebuah komunikasi.
3. Setelah inisialisasi dilakukan maka *Client* dan *Server* sudah bisa saling mengirimkan *data* dan menerima *data*.

Berikut ini adalah gambar mengenai hubungan antara *Server* dan beberapa *Client*.



Gambar 3.7 Hubungan *Server* dengan beberapa *Client*

Diagram alir komunikasi *chat* menunjukkan aktivitas antara *Server* dan *Client*. *Client* memulai permintaan layanan (*request*), *Server* menanggapi *request* tersebut (*response*), selanjutnya permintaan layanan *Client* diproses oleh *Server*. *Server* memperhatikan apakah *request Client* dapat dipenuhi atau tidak. Jika terpenuhi, *Server* akan kembali mengirimkan kembali hasil permintaan tersebut ke *Client*. Namun, apabila *request Client* tersebut tidak dipenuhi, *Server* akan mengkonfirmasikan *Client* bahwa permintaan layanan saat ini belum bisa dipenuhi. Selanjutnya, proses *Client–Server* kembali ke awal yaitu *Client* memulai permintaan layanan ke *Server*. Aktivitas *Client-Server* ini diperlihatkan pada gambar 3.8.

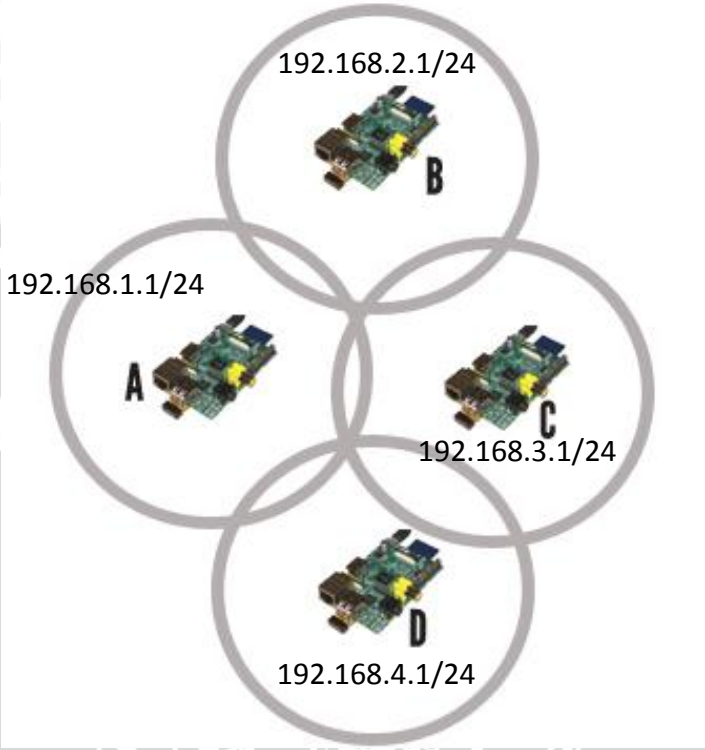


Gambar 3.8 Flowchart Aplikasi chat

3.10 Skenario penelitian

Skenario penelitian ini terdiri dari 4 *node Raspberry Pi* yang setiap *node* telah dipasang *wifi dongle edimax* dengan sistem operasi *arch linux*, tiap *node* terkoneksi dengan *node* lainnya dalam jangkauan *transmisi* tertentu dengan *coverage signal* 9 meter yang merupakan jarak maksimal untuk pengiriman data dari signal wifi edimax yang digunakan [SIN-13]. Untuk komunikasi diluar jangkauan, suatu *node* membutuhkan bantuan *node* yang lain yang bertindak sebagai *bridge* sehingga *node* dalam *Manet* bisa bertindak sebagai *terminal* dan *router*. *Node* bergerak bebas dan secara otomatis terkoneksi ketika mencapai signal range *wifi* masing-masing *node*. Ketika semua *node* sudah terkoneksi satu sama lain, lakukan 4 skenario perubahan topologi untuk mengetahui efektifitas dari *Manet*, lakukan komunikasi dengan menjalankan aplikasi *Chat* untuk saling

mengirim pesan teks, dari keempat *node* tersebut harus ada salah satu *node* yang menjadi *Server Chat* untuk melayani *Client*.



Gambar 3.9 Rancangan Arsitektur

3.11 Pengujian dan Analisis

Pengujian sistem komunikasi pada penelitian ini dilakukan untuk menunjukkan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi kebutuhan. Serta menguji kehandalan jaringan *Manet* ketika nanti akan di implementasikan menggunakan *Raspberry Pi* pada daerah bencana. Adapun pengujian yang dilakukan adalah sebagai berikut:

3.11.1 Pengujian *Self configure*

Pengujian yang pertama adalah melakukan pengujian untuk mengetahui kemampuan dari jaringan *Manet* dalam melakukan *self configure*. *Self configure* adalah kemampuan untuk mengkonfigurasi secara mandiri dari *node Manet* untuk bergabung dengan jaringan *Manet* yang sudah ada sebelumnya sehingga dapat

meneruskan paket-paket *data* yang akan dikirimkan dapat melalui *node* yang baru bergabung tersebut. Terdapat 4 skenario letak *node* dalam pengujian ini.

3.11.2 Pengujian *self healing*

Pengujian yang kedua adalah melakukan pengujian untuk mengetahui kemampuan dari jaringan *Manet* dalam melakukan *self healing*. *Self healing* adalah kemampuan dari *node Manet* untuk mencari jalur *routing* yang baru apabila pada jalur yang dilaluinya mengalami kerusakan atau hilangnya *node* dalam jaringan. Gangguan pada jalur ini dapat terjadi karena jalur yang dilewati bermasalah. Intinya adalah jika terjadi kegagalan dalam mengirimkan paket *data*, *node Manet* akan mencari *rute* alternatif untuk meneruskan paket yang akan dikirimkan.

3.11.3 Pengujian komunikasi antar *node* menggunakan aplikasi *Chat*

Pengujian yang ketiga adalah melakukan pengujian pada aplikasi *Chat* untuk komunikasi pengiriman pesan berbasis teks antar *node*. Dari empat *node* yang ada salah satunya akan menjadi *Server*, sedangkan *node* yang lain akan menjadi *Client* yang saling berkomunikasi melalui *Server*.

3.12 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi *manet* dan aplikasi *chat* yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak/keras. Implementasi terdiri atas penjelasan spesifikasi sistem, konfigurasi jaringan, prosedur implementasi, dan implementasi fisik.

4.1. Spesifikasi Lingkungan Sistem

Implementasi *Manet* dan aplikasi *chat* dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

4.1.1. Spesifikasi Lingkungan Perangkat Keras

Perangkat keras yang dibutuhkan dalam implementasi *Manet* dan pembuatan aplikasi *chat* pada penelitian sistem komunikasi alternatif bencana menggunakan teknologi *Manet* menggunakan 3 perangkat keras yaitu *Raspberry* Pi, *SD Card*, dan *Wifi Dongle edimax*.

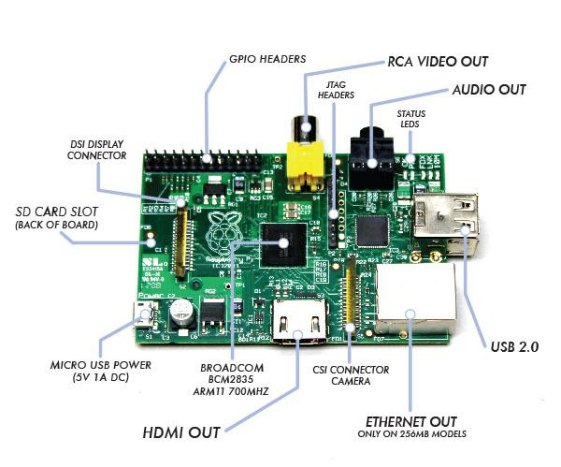
4.1.2 *Raspberry* Pi

Raspberry pi adalah *Singe Board Computer* seukuran kartu kredit yang dikembangkan oleh *Raspberry Pi Foundation*. Berikut adalah tabel penjelasan tentang spesifikasi hardware *Raspberry*.

Tabel 4.1. Spesifikasi *Raspberry pi*

Raspberry pi	
Chip	*Broadcom BCM2835 SoC full HD multimedia Applications processor*
CPU	*700 MHz Low Power ARM1176JZ-F Applications Processor*

GPU	*Dual Core VideoCore IV® Multimedia Co-Processor*
Memory	*512MB SDRAM*
Ethernet	*onboard 10/100 Ethernet RJ45 jack*
USB 2.0	*Dual USB Connector*
Video Output	*HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)*
Audio Output	*3.5mm jack, HDMI*
Onboard Storage	*SD, MMC, SDIO card slot*
Operating System	*Linux*
Dimensions	*8.6cm x 5.4cm x 1.7cm*



Gambar 4.1 Raspberry Pi

4.1.3 SD Card

SD Card merupakan Storage yang digunakan dalam penyimpanan *data* pada *Raspberry*, berikut adalah table penjelasan tentang spesifikasi hardware *SD Card*.

Tabel 4.2. Spesifikasi *SD Card*

Micro SD SanDisk 8 GB	
Kapasitas Penyimpanan	*8 GB*
Kecepatan	*Up to 30MB/second – Class 10*
Ukuran	*6.7 x 4.5 x 1.1 inches ; 2.4 ounces*
Type produk	*Flash memory card*

Gambar 4.2 Micro SD SanDisk 8 GB

4.1.4 *Wifi*

EW-7811Un adalah *nano USB adapter nirkabel* yang menawarkan jangkauan maksimum dan kecepatan. Meskipun ukurannya kecil, *adaptor USB* ini memiliki kecepatan yang tinggi ketika terhubung dengan perangkat *nirkabel 802.11n* lainnya, yang mana kecepatannya 3 kali lebih cepat daripada koneksi *11g* biasanya.

Tabel 4.3. Spesifikasi *Wifi*

Wifi dongle edimax EW-7811Un	
Standar	*IEEE802.11b, 802.11g, 802.11n*
Hardware Interface	*1 USB 1.0/2.0 Tipe A, Antena internal*
Frequensi band	*2,4000 ~ 2.4835GHz*

Data rate	- *11b: 1/2/5.5/11Mbps* - *11g: 6/9/12/24/36/48/54Mbps* - *11n (20MHz): MCS0-7 (sampai 72Mbps)* - *11n (40MHz): MCS0-7 (sampai dengan 150Mbps)*
KELEMBABAN & SUHU	*Operasi: 10 ~ 90% (Non Kondensasi)* *Penyimpanan: Max. 95% (Non Kondensasi)* *Operasi: 32 ~ 104 ° F (0 ~ 40 ° C)* *Penyimpanan: -4 ~ 140 ° F (-20 ~ 60 ° C)*
OUTPUT DAYA	*11b: 17 ± 1.5dbm* *11g: b: 15 ± 1.5dbm* *11n: 14 ± 1.5dbm*
PERSYARATAN SISTEM	*Windows XP/Vista/7* *Linux & Mac OS*

Sumber : Perancangan

Gambar 4.3 Wifi dongle edimax EW-7811Un

4.2 Spesifikasi Lingkungan Perangkat Lunak

Perangkat lunak yang dibutuhkan dalam implementasi *Manet* dan pembuatan aplikasi *chat* pada penelitian sistem komunikasi alternative bencana menggunakan teknologi *Manet* dijelaskan pada table 4.4.

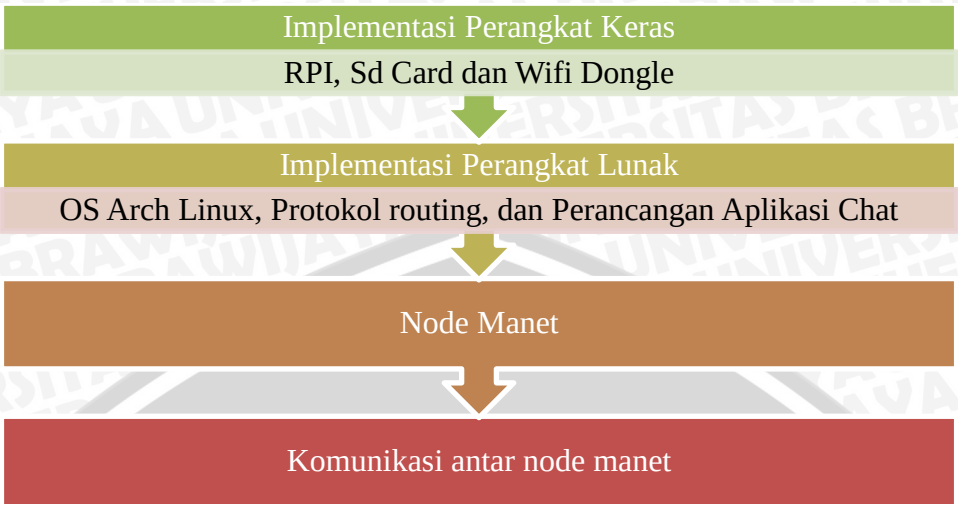
Tabel 4.4. Spesifikasi lingkungan perangkat lunak komputer

Operating system	*Arch linux ARM 13-06-2012*
Protokol *Routing*	Protokol *routing Babel*
Programming language	*Python*
Aplikasi	Aplikasi *Chat Client-Server*

Sumber : Implementasi

4.3 Implementasi Sistem

Pada gambar 4.4 dijelaskan mengenai perancangan sistem secara keseluruhan, dimulai dari implementasi perangkat keras sistem, yang terdiri dari *Raspberry pi, SD Card* sebagai *storage sistem* dan *Wifi Dongle edimax* sebagai perangkat *nirkabel* untuk komunikasi melalui media gelombang *radio, SD Card* dan *Wifi Dongle edimax* dipasang pada slot yang sudah disediakan pada *Raspberry pi*. Setelah merancang perangkat keras selesai, kemudian implementasi perangkat lunak yang terdiri dari sistem operasi, protokol *routing*, dan aplikasi *chat*. Sistem operasi yang digunakan adalah sistem operasi *Arch linux*, sistem operasi ini disimpan pada *SD Card*. Kemudian, menginstall protokol *routing* pada sistem operasi *Arch Linux*, protokol *routing* yang digunakan adalah protokol *routing babel*. Dengan adanya protokol *routing* ini menjadikan *node* bisa menggunakan *routing* secara dinamis. Setelah itu, implementasi aplikasi *chat* menggunakan aplikasi *Client-Server chat* untuk mengirim pesan teks antar *node*, pemrogaman aplikasi *chat* menggunakan bahasa pemrogaman *socket python* yang dijalankan pada sistem operasi *Arch Linux*. Dari hasil merancang perangkat keras dan perangkat lunak, menghasilkan *node Manet* yang bisa berfungsi sebagai *transmitter, receiver dan router*. Setelah *node Manet* terbentuk, dan membuat jaringan yang terdiri dari *node-node Manet*. Tiap *node* tersebut saling berkomunikasi menggunakan aplikasi *chat* yang tiap *node* bisa menggunakannya untuk saling mengirim pesan teks antar *node*.



Gambar 4.4 Implementasi Sistem

4.3.1 Instalasi *OS Arch linux* pada *SD Card*

1.*Download sistem operasi Raspberry Pi dari http://archlinuxARM.org/os/ArchLinuxARM-rpi-latest.zip.*

2. *Unzip file* yang di *Download*

a) Klik kanan *file* dan Pilih *"Extract all".*

b) Ikuti instruksi selanjutnya, kemudian terdapat *file.img*

3. *Download software Win32DiskImager*.

a) *Download Win32DiskImager.zip* dari : *https://launchpad.net/win32-Image writer/+Download*

b) *Unzip file* tersebut.

c) Akan terdapat folder baru dengan nama *Win32DiskImager-binary*

4. Menulis *Arch linux ARM* pada *SD Card*

a) Pasang *SD Card* pada komputer.

b) Jalankan dengan mengklik dua kalo *file Win32DiskImager.exe*. Akan terlihat seperti ini:

Gambar 4.4 *Software Win32 Disk Imager*

c) Jika *SD Card* (*Device*) tidak ditemukan secara otomatis, kemudian klik drop down box dan memilihnya.

d) Pada *Image file* box, Pilih *file arch linux* yang telah di *Download*.

e) Klik *write*

f) Setelah beberapa menit progress penuh, *SD Card* sudah dapat digunakan pada *Raspberry pi*.

g) Tancapkan *SD Card* pada *Raspberry*, dan nyalakan *Raspberry pi* untuk booting pertama kali pada sistem operasi *Arch linux ARM*, untuk *username* dan *password = root*.

4.3.2 Setting Wifi

Setelah instalasi sistem operasi *Arch linux ARM* pada *SD Card*, perlu untuk mendapatkan koneksi *internet* karena dibutuhkan paket-paket yang harus di *Download* di *internet*. Untuk melakukan proses ini harus mengatur *wifi dongle edimax* terlebih dahulu. Berikut ini cara *setting wifi* untuk berkoneksi dengan *internet*:

Setelah *Booting Up, login* ke *console* menggunakan *username* dan *password = root*. Untuk mengetahui apakah *Device wifi dongle* cocok dengan *arch linux*, ketikkan perintah berikut :

```
[root@alarmpi ~]# lsusb
```

Ketikkan perintah di bawah ini untuk memastikan *Wireless interface* telah dibuat.

```
[root@ alarmpi ~]# ip link
```

System akan menjawab *wlan0* link telah terbuat

```
5: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DORMANT qlen 1000
```

Membuat *interface up* dengan *ip link set up*, untuk contohnya *interface wlan0 :*

```
[root@alarmpi ~]# ip link set wlan0 up
```

Setelah itu, *scan network* yang tersedia untuk koneksi internet.

```
[root@ alarmpi ~]# iwlist wlan0 scan

Cell 01 - Address: XX:XX:XX:XX:XX:XX

          ESSID:"your_wifi_name"

          Protocol:IEEE 802.11bgn

          Mode:Master

          Frequency:X.XXX GHz (Channel X)

          Encryption key:on

          Bit Rates:XX Mb/s

Extra:wpa_ie=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

          IE: WPA Version 1

            Group Cipher : TKIP
```

Pairwise Ciphers (2) : CCMP TKIP

Authentication Suites (1) : PSK

Extra:rsn_ie=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X

IE: IEEE 802.11i/WPA2 *Version* 1

Group Cipher : TKIP

Pairwise Ciphers (2) : CCMP TKIP

Authentication Suites (1) : PSK

IE:Unknown:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Quality=100/100 Signal level=100/100

Untuk pengaksesan ke koneksi internet yang tersedia kita perlu mengetahui *username* dan *passwordnya,* pada *file wpa _suplicant* kita harus memasukkan nama *wifi* dan *password* sesuai dengan konfigurasi *wifi* yang tersedia.

```
[root@ alarmpi ~]# wpa_passphrase your_wifi_name "your_wifi_password" > /etc/wpa_supplicant.conf
```

Jika cara di atas sudah benar, gunakan perintah di bawah ini untuk koneksi ke jaringan.

```
[root@ alarmpi ~]# wpa_supplicant -iwlan0 -c /etc/wpa_supplicant.conf & dhcpcd wlan0
```

Supaya ketika awal *booting system* langsung koneksi ke jaringan, *copy file wireless-wpa* ke */etc/netctl/*

```
[root@ alarmpi ~]# cp /etc/netctl/examples/wireless-wpa /etc/netctl/
```

Edit username dan *key* pada *file /etc/netctl/*, gunakan perintah

```
Nano /etc/netctl/
```

Start netctl untuk menghidupkan settingan *wifi* dengan menggunakan perintah

```
[root@ alarmpi ~]# netctl start pro*file*
```

Enabel koneksi ketika setiap memulai *system* dengan menggunakan perintah.

```
[root@ alarmpi ~]# netctl enable pro*file*
```

4.3.3 Instalasi *Babel*

Untuk menginstall paket-paket *babel*, pertama kali yang dilakukan adalah *update* paket-paket *os arch linux* terlebih dahulu. Ketikkan perintah :

```
[root@ alarmpi ~]# pacman -S haveged && haveged -w 1024 && pacman-key --init && pkill haveged && pacman -Rs haveged
```

Kemudian *install* paket-paket berikut:

```
pacman -S jshon git base-devel fakeroot

pacman -S dialog *wireless*_tools netcfg

cd /tmp/ && wget http://aur.arch*linux*.org/packages/pa/packer/PKGBUILD

makepkg --asroot PKGBUILD

pacman -U packer*

packer -Syu && packer -S wpa_auto newlan
```

Install protokol *routing Babel* dengan mengetikkan perintah:

```
packer -S *babel*d
```

4.3.4 Konfigurasi protokol *routing Babel*

Protokol *routing babel* pada masing-masing *node* menggunakan program *shell script* untuk berjalan secara otomatis setiap kali *node* di jalankan. Berikut ini adalah penjelasan konfigurasi protokol *routing babel*.

Matikan koneksi *wireless* supaya *wifi card* tidak langsung menyala saat *boot* dengan mengetikkan perintah:

```
netctl stop name.service
```

Jalankan *wifi* dongle dengan mengetikkan perintah

```
ip link set wlan0 up
```

Kemudian buat konfigurasi jaringan dengan perintah

```
iwconfig wlan0 mode ad-hoc channel 11 essid "mesh"
```

Pengalamatan IP address secara manual

```
ip addr add 192.168.x.x dev wlan0
```

Kemudian Jalankan *babel* dengan perintah

```
babeld -D wlan0
```

4.3.5 Pengalamatan *Node*

Pada pengujian ini terdapat 4 *node* yang yang ditempatkan pada posisi tertentu agar sesuai dengan topologi jaringan yang dikehendaki, Setiap *node* memiliki alamat *IP Address* yang berbeda sebagai berikut:

a. *Node* dengan alamat *IP Address* 192.168.1.3 *Subnet Mask* 255.255.255.0
b. *Node* dengan alamat *IP Address* 192.168.1.4 *Subnet Mask* 255.255.255.0
c. *Node* dengan alamat *IP Address* 192.168.1.5 *Subnet Mask* 255.255.255.0
d. *Node* dengan alamat *IP Address* 192.168.1.6 *Subnet Mask* 255.255.255.0

4.3.6 Shell script

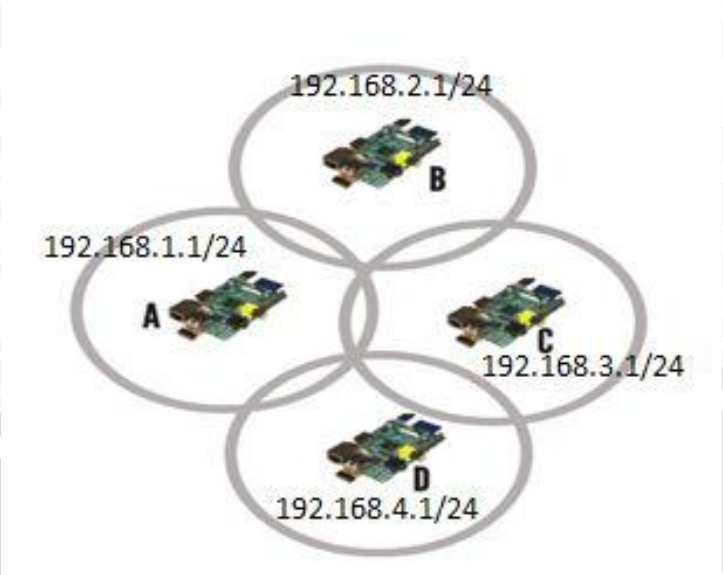
Program *shell script* dijalankan di *background* sistem operasi *arch linux*, yaitu mengkonfigurasi *ad-hoc* terlebih dahulu untuk membuat jaringan *ad-hoc*, kemudian menjalankan program *babel.sh* dan program *ping.sh* setelah 60 detik. diset setelah 60 detik karena *babel* membutuhkan waktu selama 20 detik untuk mengetahui informasi-informasi *node Manet* tetangga.

```
if ! $connected; then
   createAdHocNetwork
fi
sleep 60 && /bin/sh /home/babel.sh && /bin/sh /home/ping.sh
exit 0
```

4.4 Implementasi Fisik

Dalam rangka membangun implementasi fisik merupakan hal yang cukup penting untuk membentuk *Manet*, karena penentuan letak tiap *node* diletakkan di beberapa posisi yang berbeda dalam jarak tertentu, pemilihan lokasi juga memperhatikan banyak faktor antara lain : tempat untuk menempatkan *node* harus memiliki sumber listrik, selain itu penempatan *node* harus terlindung dari pengaruh cuaca seperti hujan karena dapat menyebabkan kerusakan pada *node*. Terdapat 4 skenario dalam impelementasi ini.

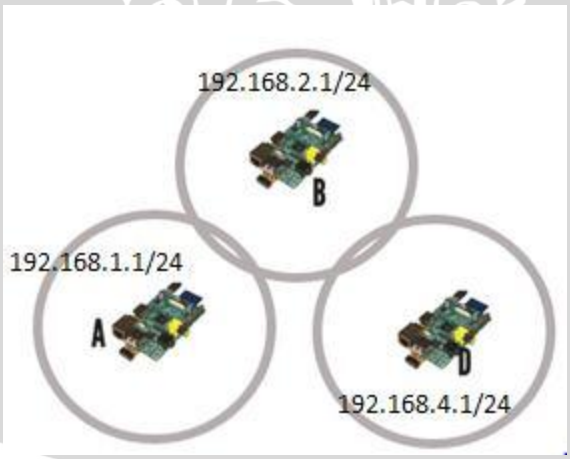
Skenario 1:



Gambar 4.6 letak *node-node* pada skenario 1

Terdapat 4 *node* dalam skenario ini, *Node* A, B, C dan D. Semua *node* terhubung dengan signal *wifi node* lainnya. Jarak antar *node* dengan *node* lainnya 1 meter.

Skenario 2:

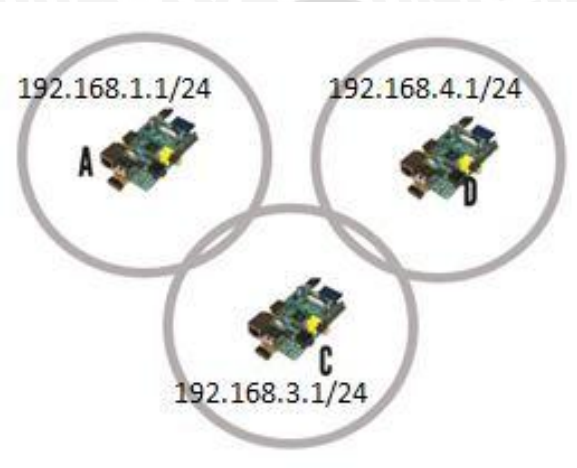


Gambar 4.7 Letak *node-node* pada skenario 2

Terdapat 3 *node* dalam skenario impementasi ini, *Node* A, B, dan D. Signal *node* A terhubung dengan *node* B, *node* B terhubung dengan *node* D.

Ketika *node* A berkomunikas dengan *node* D harus melewati *node* B terlebih dahulu. Jarak antar *node* dengan *node* lainnya adalah 9 meter.

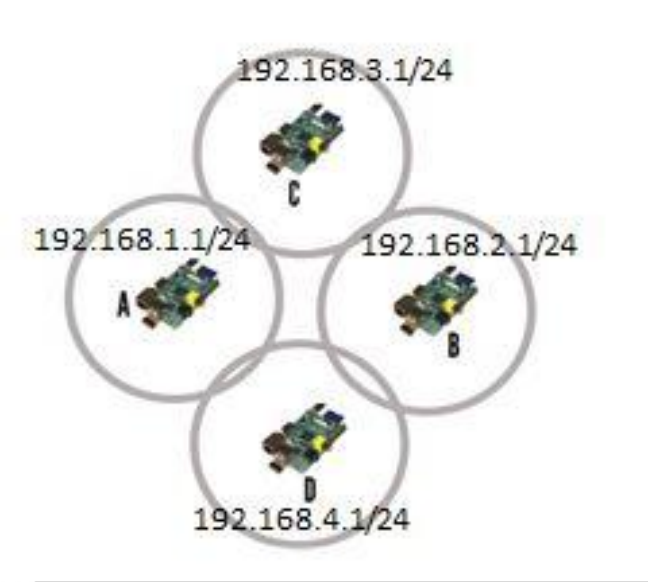
Skenario 3:



Gambar 4.8 Letak *node-node* pada skenario 3

Terdapat 3 *node* dalam skenario impementasi ini, *Node* A, C, dan D. Signal *node* A terhubung langsung dengan *node* C, *node* C terhubung dengan *node* D. Ketika *node* A berkomunikas dengan *node* D harus melewati *node* C sebagai gateway terlebih dahulu. Jarak antar *node* dengan *node* lainnya adalah 9 meter.

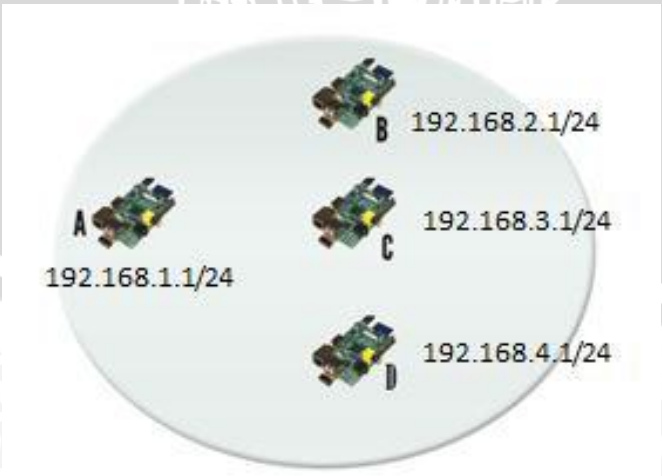
Skenario 4:



Gambar 4.9 Letak *node-node* pada skenario 4

Terdapat 4 *node* dalam skenario implementasi ini, *Node* A, B, C dan D. Signal *node* A terhubung dengan *node* C dan *node* D, tapi tidak terhubung langsung dengan signal *node* B. Ketika *node* A berkomunikasi dengan *node* B harus memilih salah satu jalur yaitu melewati *node* C sebagai gateway atau melewati *node* D, dalam hal ini yang dipilih adalah *node* yang memiliki nilai op count yang lebih kecil.

4.10 Skenario implementasi aplikasi *chat*.



Gambar 4.10 Letak node pada implementasi *chat*

Komunikasi yang terbentuk dari aplikasi *chat* adalah koneksi dari *socket* ke *socket*. *Socket* adalah dua buah nilai yang telah di tentukan untuk mengidentifikasikan setiap *node* sebuah alamat IP dan nomor *port* . Analogikan *socket* sebagai Pintu, untuk berkomunikasi maka Pintu kedua *node* harus terbuka. *Client*-lah yang dalam hal ini harus lebih proaktif untuk membuka koneksi terlebih dahulu, *Client* harus "mengetuk Pintu" dahulu pada *Server*. *Server* memonitor setiap ketukan atau permohonan koneksi dari *Client*, setiap koneksi memiliki ID yang unik. Setelah *Server* menerima koneksi, maka sebuah koneksi baru bisa terbentuk sehingga baik *Client* maupun *Server* sudah bisa berkomunikasi. Hubungan koneksi antar *node* dalam implementasi ini ada dua macam, yaitu broadcast *message* dan *Private message*. Pada skenario implementasi aplikasi *chat Client Server*, terdapat 4 *node* yaitu *node* A, B, C dan D. *Node* A bertindak sebagai *Server*, sedangkan *node* B, C dan *node* D sebagai *Client*.

4.11 Kendala dan Solusi

Dalam penelitian ini ada beberapa kendala yang dihadapi, yaitu masalah konektivitas secara otomatis pada *routing* protokol *babel*, protokol *routing babel* harus menggunakan *IP* dinamis supaya bisa terkoneksi secara otomatis, untuk menggunakan IP dinamis dibutuhkan protokol *AHCP (Ad-Hoc Configuration Protocol)* yang berfungsi untuk memberikan *IP* secara dinamis pada jaringan *Manet*, sedangkan aplikasi *chat Client Server* membutuhkan *IP* statik untuk berkoneksi ke *Server* untuk melakukan komunikasi antar *node* berbasis teks.

Untuk mensiasati kendala yang dihadapi tersebut. Dalam penelitian ini dibuat program ping menggunakan *shell script*. Program *shell script* berfungsi untuk menyimpan konfigurasi *ad-hoc*, menjalankan protokol *routing babel* dan program *ping*. Protokol *routing babel* dan program *ping* berjalan diset 10 detik setelah *startup* dan program berjalan di *background*, antar *node* melakukan *ping* terus menerus dengan tujuan alamat *IP node Server*, ketika *ping unreachable* selama proses 60 kali *ping* saat melakukan *ping* ke *Server*, maka *node* akan

melakukan *restart*, setelah *restart node* akan terkoneksi kembali ke *Server* dan node lainnya yang terjangkau dengan *signal range wireless*.

BAB V
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan pembahasan tentang sistem komunikasi pada *Manet*. Proses pengujian dilakukan melalui 2 macam pengujian yaitu pengujian *Manet* dan pengujian aplikasi *chat*. Pengujian *Manet* digunakan untuk mengetahui apakah tiap *node* bisa berkoneksi secara otomatis menggunakan protokol *routing babel*, pada pengujian *Manet* ini dibagi lagi menjadi dua pengujian yaitu *self configure* dan *self healing*. Pengujian aplikasi *chat Client Server* digunakan untuk menguji komunikasi pengiriman pesan berbasis teks antar *node,* Dari hasil pengujian tersebut, diharapkan penelitian ini bisa diterapkan di daerah bencana yang sebenarnya.

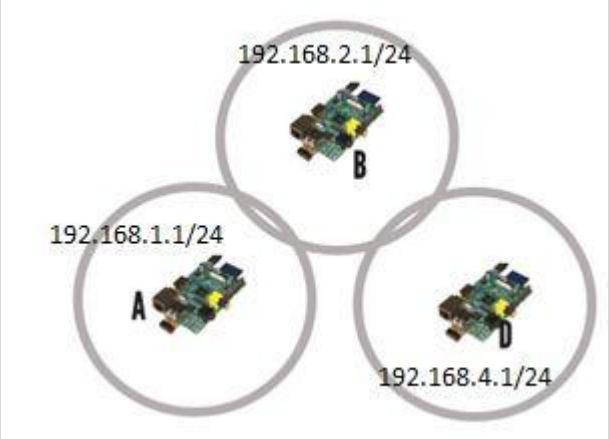
Proses pembahasan bertujuan untuk mendapatkan kesimpulan dari hasil pengujian implementasi dan pengujian sistem komunikasi pada *Manet*. Proses pembahasan mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Pembahasan dilakukan terhadap hasil sesuai pengujian di setiap tahap pengujian. Proses pembahasan yang dilakukan meliputi pembahasan hasil pengujian *Manet* dan aplikasi *chat*.

5.1 Pengujian dan analisa

Pengujian dilakukan untuk menguji jaringan *Manet* dan aplikasi *chat* untuk sistem komunikasi yang nantinya akan diterapkan pada daerah bencana ketika sistem komunikasi lain tidak bisa digunakan, dalam pengujian-pengujian yang dilakukan membuat scenario semua komunikasi tidak dapat digunakan seperti terkena dampak bencana yang menyebabkan semua infrastruktur komunikasi mengalami kerusakan. Skenario pengujian *Manet* yang dilakukan adalah untuk melihat kemampuan dari masing-masing *node Raspberry* dari jaringan *Manet* dalam melakukan *self configure* dan *Self healing*, sedangkan untuk menguji aplikasi *chat* adalah dengan melakukan pengiriman pesan antar *node Client* untuk berkomunikasi.

5.1.1 Pengujian *Self configure*

Pengujian pertama yang dilakukan pada penelitian ini adalah untuk melihat kemampuan dari jaringan *Manet* dalam melakukan *self configure* menggunakan protokol *routing Babel* . *Self configure* adalah kemampuan *node Manet* melakukan *routing* dengan menggunakan protokol *routing* untuk membangun koneksi dengan *node-node Manet* lain yang sudah ada sebelumnya sehingga dapat meneruskan paket-paket *data* yang akan dikirimkan dapat melalui *router*/*node* yang baru bergabung tersebut. Pengujian *self configure* ini penting karena ketika terjadi bencana, *node Manet* bergerak tak beraturan dengan jangkauan yang berubah-ubah, sehingga bisa mencapai suatu keadaan dimana antar *node* tidak menjangkau *signal wireless* satu sama lain, sehingga dibutuhkan *node* baru sebagai jembatan penghubung komunikasinya. Skenario pengujian ini dilakukan dengan menggunakan 3 buah *node* yang diletakkan seperti gambar 5.1

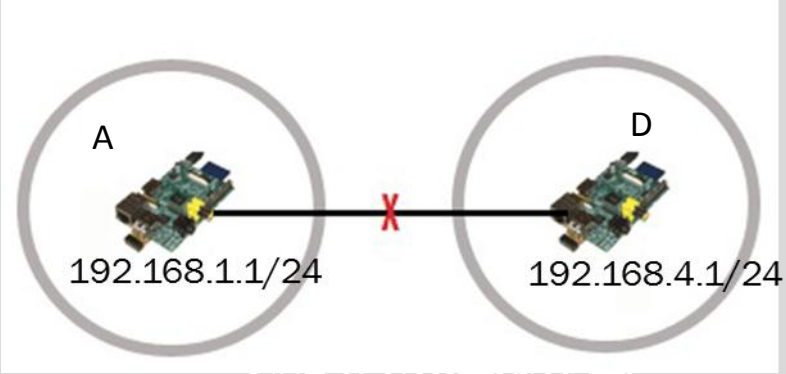


Gambar 5.1 letak *node-node* pada pengujian *self configure*

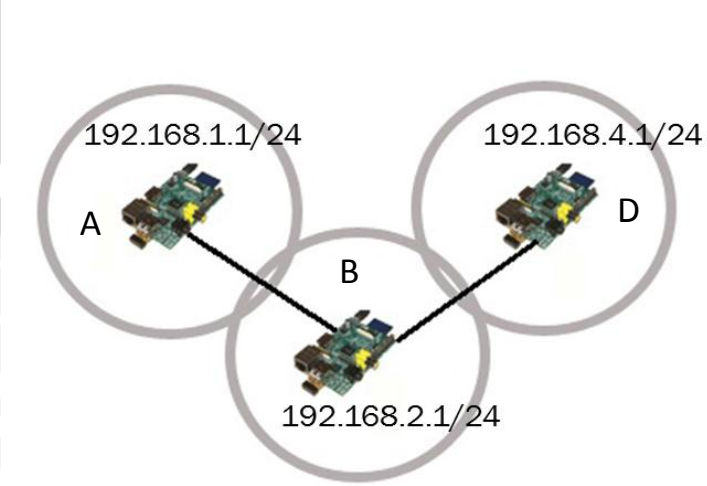
Node yang akan digunakan adalah *node* A, B, dan C. Cara yang dilakukan adalah *node* A melakukan Ping terhadap *node* C karena *node* B belum diaktifkan sehingga jaringan belum terbentuk maka pada *node* A akan muncul tamPilan *request time out.* Setelah *node* B diaktifkan maka dalam beberapa saat jaringan akan terbentuk dan sistem multihop dapat berjalan sehingga *node* A dapat berkomunikasi dengan *node* C melalui *node* B. dan akan muncul pada *node* A yang melakukan Ping terhadap *node* C adalah *reply from IP node C.*

Langkah uji cobanya adalah:

1. Letakkan *Node* A, diluar jangkauan *node* C.
2. Jalankan *babel* pada *node* A dan *node* C.
3. Kondisi awal *node* B dalam keadaan tidak aktif *(off)*
4. Kemudian aktifkan *node* B sehingga *node* A bisa terhubung dengan *node* C melalui *node* B.
5. Lakukan perintah *route* untuk mengetahui apakah semua *node* sudah terdaftar pada table *routing*.
6. Lakukan *Ping* ke semua *node* untuk mengecek apakah sudah terhubung dengan semua *node* yang terdaftar di tabel *routing*.
7. Kemampuan *node* C untuk bergabung dengan jaringan yang sudah ada inilah yang dikatakan sebagai kemampuan *self configure* dan hubungan yang terjadi menggunakan sistem *multihop.*

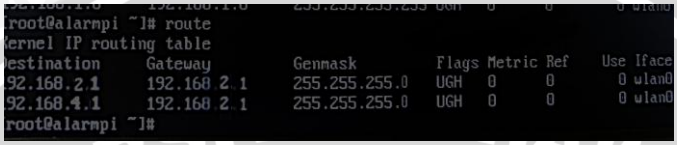


Gambar 5.2 *Node* A tidak terkoneksi dengan *Node* C



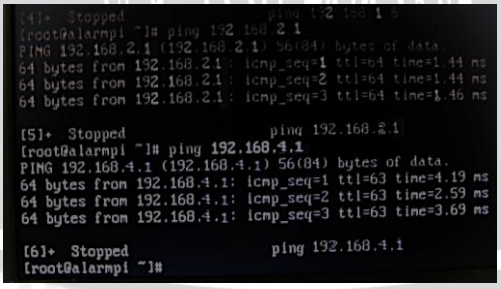
Gambar 5.3 *Node* A terkoneksi dengan *node* C lewat *Node* B

Pada gambar 5.2 *node* A tidak bisa melakukan komunikasi dengan *node* C, karena jalur yang dilalui belum terbentuk hal ini disebabkan oleh keterbatasan jarak antara *node* A dan *node* C yang berjauhan tidak mencapai signal masing-masing. Setelah *node* B ditempatkan seperti gambar 5.3, maka jalur antara *node* A dan *node* C telah terbentuk, keterbatasan jarak antara *node* A dan *node* C dapat dijembatani oleh *node* B yang ditempatkan diantara *node* A dan *node* C.



Gambar 5.4 *Routing* table *node* A pengujian self configure

Perintah *route* adalah untuk melakukan *routing* pada jaringan. *Routing* adalah proses penerusan paket dari suatu *node* supaya dapat sampai ke tujuan dari satu lokasi ke lokasi lain, tujuan utama *routing* adalah *router-router* tidak mempelajari jalur-jalur terhubung secara langsung dengannya, tetapi mengatur bagaimana meneruskan paket *data* ke jalur yang terhubung tidak langsung. *Routing* harus mampu mengatasi perubahan topologi jaringan serta lalu lintas jalur tanpa pembatalan proses pada host. Informasi sumber dan tujuan disimpan pada tabel *routing*. Hasil output *route* menunjukkan bahwa *node* A terhubung dengan *node* B dan *node* C, tetapi untuk menuju *node* C harus melewati *node* B sebagai gateway.



Gambar 5.5 Ping *node* A

Ping adalah perintah yang biasa digunakan untuk menguji koneksi jaringan dengan cara mengirimkan paket *data* kepada *host* dan menghitung lamanya waktu yang dibutuhkan untuk proses pengiriman *data* tersebut. Ping

adalah *Packet Internet or Inter-Network Group*. Perintah Ping menggunakan pengiriman dengan *packet Internet Protocol (IP)*, yang biasa dikenal dengan protokol *ICMP (Internet Control Message Protocol*) dengan mengirimkan *packet echo request datagram*. Setiap paket yang dikirimkan menunggu jawaban dari alamat tujuan *(destination)*. Hasil *output Ping* berisikan lamanya waktu jawaban dari alamat tujuan.

5.1.2 Pengujian *Self healing*

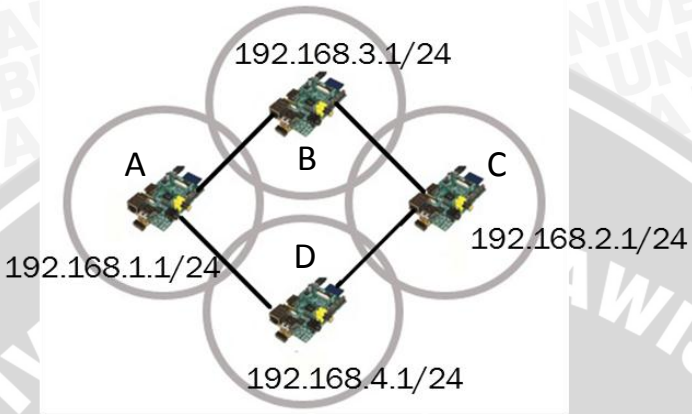
Skenario kedua yang dilakukan pada penelitian ini adalah untuk melihat kemampuan dari jaringan *Manet* dalam melakukan *self healing* dengan menggunakan protokol *routing Babel*. *Self healing* adalah kemampuan dari *Manet* mencari jalur *routing* yang baru apabila pada jalur yang dilaluinya terjadi kerusakan. Gangguan pada jalur ini dapat terjadi karena jalur yang dilewati bermasalah. Intinya adalah jika terjadi kegagalan dalam mengirimkan paket *data*, *node* dapat mencari jalur alternatif untuk meneruskan paket yang akan dikirimkan. Pengujian ini penting karena, ketika terjadi bencana *node* bisa mengalami kerusakan sewaktu-waktu, sehingga *node* harus mempunyai kemampuan untuk mencari jalur alternatif sebagai pengganti jalur yang rusak untuk meneruskan paket *data* yang dikirimkan.

Skenario yang dilakukan untuk pengujian *self healing* dalam jaringan *Manet* adalah dengan menggunakan 4 buah *node Manet* yang telah diposisikan seperti pada gambar 5.6 dimana semua *node* berada dalam posisi aktif sehingga setiap *node* dapat melakukan Ping terhadap *node* lainnya.

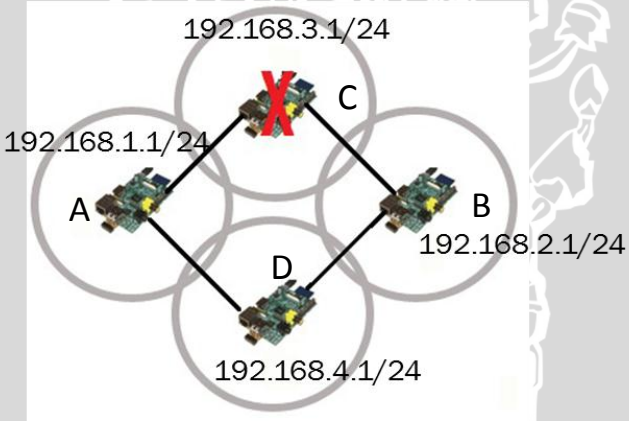
Langkah-langkah yang dilakukan pada pengujian self-healing adalah:

1. Lakukan *Ping node* A terhadap *node* C hingga mendapatkan balasan dari *node* C berupa reply from *node* C.
2. Lihat jalur yang digunakan oleh *node* A untuk berkomunikasi dengan *node* C, misal jalur yang digunakan adalah melalui:

Node A – *node* B – *node* C seperti mematikan power pada *router* B dan perintah Ping masih terus dilakukan hingga terbentuk jalur baru seperti pada gambar 5.7 *node* A-*node* D- *node* C.



Gambar 5.6 pengujian *self healing*

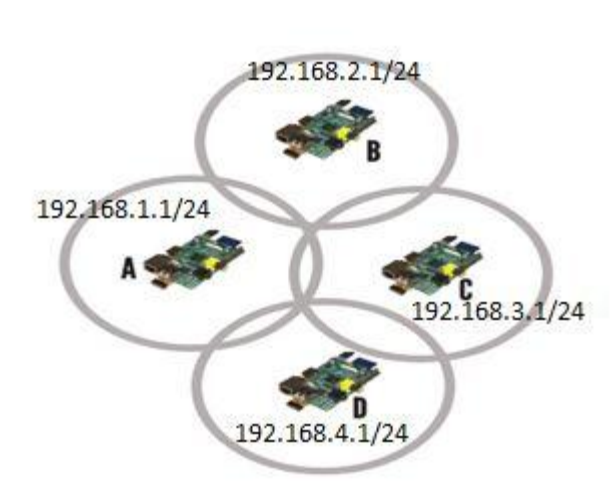


Gambar 5.7 Terjadi kerusakan pada *node* B

Pada gambar 5.6 Jalur awal yang dibentuk untuk komunikasi antara *node* A dan *node* C adalah melalui *router* B. Untuk menguji kemampuan *self healing* maka *node* B di non-aktifkan, sehingga *rute* baru terbentuk untuk komunikasi antara *node* A dan *node* C yaitu melalui jalur *node* D.

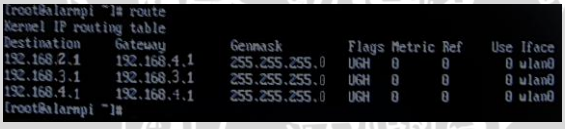
5.1.3 Pengujian Letak *Node Manet*

Skenario 1:

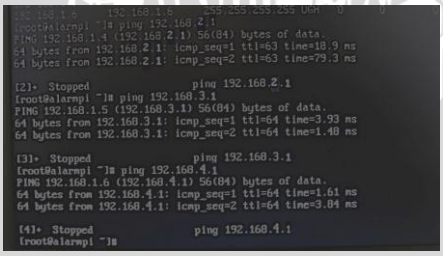


Gambar 5.8 letak *node-node* pada skenario 1

Pengujian *Manet* dilakukan pada penelitian ini dengan menempatkan 4 *node Manet* dalam skenario ini, yaitu *node* A, B, C dan D. Semua *node* terhubung dengan signal *wifi node* lainnya. Jarak antar *node* dengan *node* lainnya adalah 1 meter sehingga semua signal *node* saling mengenai.



Gambar 5.9 *Routing* table *node* A pada skenario 1

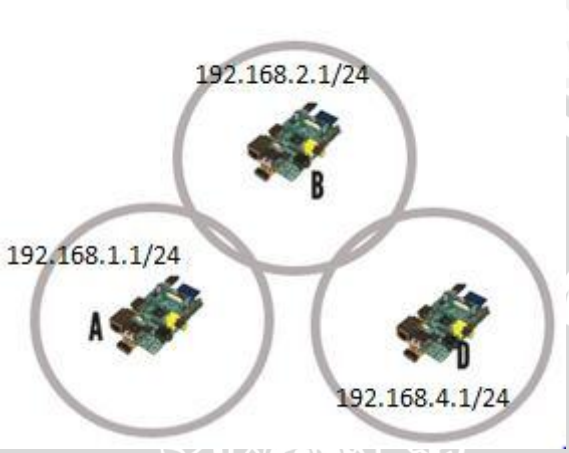


Gambar 5.10 *Routing* Ping *node* A pada skenario 1

Pada gambar 5.9 Menjelaskan hasil *route* dari *node* A yang menunjukkan bahwa telah terkoneksi dengan *node* B, C, dan D secara langsung karena informasi gateway adalah miliknya sendiri, informasi *route* tersebut disimpan di tabel *routing*. Kemudian dilakukan Ping terhadap masing-masing *node* untuk

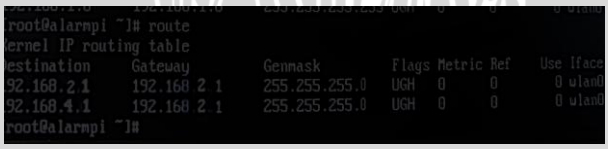
menguji koneksi jaringan, dari hasil tersebut diketahui bahwa ketika dilakukan Ping semua terkoneksi dengan baik yang ditunjukkan dari gambar 5.10

Skenario 2:

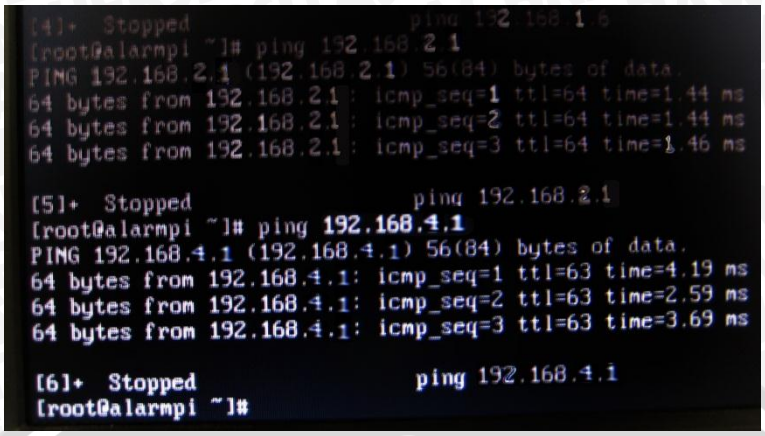


Gambar 5.11 Letak *node-node* pada skenario 2

Pengujian *Manet* dilakukan pada penelitian ini dengan menempatkan 3 *node Manet* dalam skenario ini, yaitu *node* A, B, dan D. *Node* A terhubung dengan signal *wifi node* B tapi tidak terhubung langsung dengan *node* D, *node* B terhubung dengan *node* A dan *node* D sehingga ketika *node* A berkomunikasi dengan *node* D harus melewati *node* D.



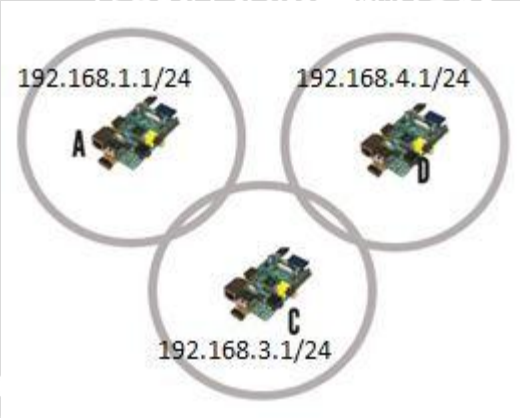
Gambar 5.12 *Routing* table *node* A pada skenario 2



Gambar 5.13 Ping *node* A pada skenario 2

Pada gambar 5.12 Menjelaskan hasil *route* dari *node* A yang menunjukkan bahwa telah terkoneksi dengan *node* B, dan D. Signal *node* A tidak terhubung langsung dengan signal *node* D sehingga informasi gateway dari *node* D adalah IP dari *node* B. Informasi *route* tersebut disimpan di tabel *routing*. Kemudian dilakukan Ping terhadap masing-masing *node* untuk menguji koneksi jaringan, dari hasil tersebut diketahui bahwa ketika dilakukan Ping semua terkoneksi dengan baik yang ditunjukkan dari gambar 5.13

Skenario 3:

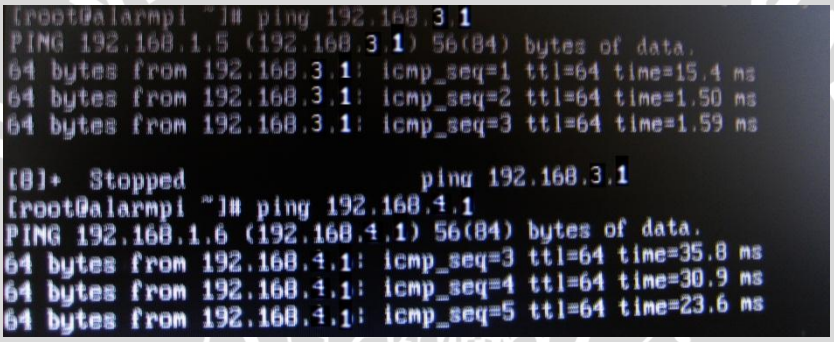


Gambar 5.14 Letak *node-node* pada skenario 3

Pengujian *Manet* dilakukan pada penelitian ini dengan menempatkan 3 *node Manet* dalam skenario ini, yaitu *node* A, C, dan D. *Node* A terhubung dengan signal *wifi node* C taPi tidak terhubung langsung dengan *node* D, *node* C

terhubung dengan *node* A dan *node* D sehingga ketika *node* A berkomunikasi dengan *node* D harus melewati *node* C.

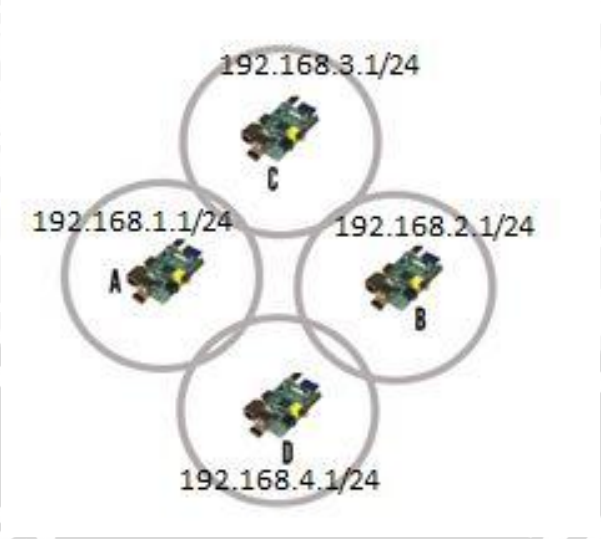
Gambar 5.15 *Routing* table *node* A pada skenario 3



Gambar 5.16 Ping *node* A pada skenario 3

Pada gambar 5.15 Menjelaskan hasil *route* dari *node* A yang menunjukkan bahwa telah terkoneksi dengan *node* C, dan D. Signal *node* A tidak terhubung langsung dengan signal *node* D sehingga informasi gateway dari *node* D adalah IP dari *node* C. Informasi *route* tersebut disimpan di tabel *routing*. Kemudian dilakukan Ping terhadap masing-masing *node* untuk menguji koneksi jaringan, dari hasil tersebut diketahui bahwa ketika dilakukan Ping semua terkoneksi dengan baik yang ditunjukkan dari gambar 5.16

Skenario 4:



Gambar 5.17 Letak *node-node* pada skenario 4

Pengujian *Manet* dilakukan pada penelitian ini dengan menempatkan 4 *node Manet* dalam skenario ini, yaitu *node* A, B, C dan D. Signal *node* A terhubung dengan *node* C dan *node* D taPi tidak terhubung langsung dengan *node* B, untuk berkomunikasi dengan *node* B harus melewati salah satu dari *node* C atau *node* D.

```
[root@alarmpi ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.1     192.168.4.1     255.255.255.0   UGH   0      0        0 wlan0
192.168.3.1     192.168.3.1     255.255.255.0   UGH   0      0        0 wlan0
192.168.4.1     192.168.4.1     255.255.255.0   UGH   0      0        0 wlan0
[root@alarmpi ~]#
```

Gambar 5.18 *Routing* table *node* A pada skenario 4

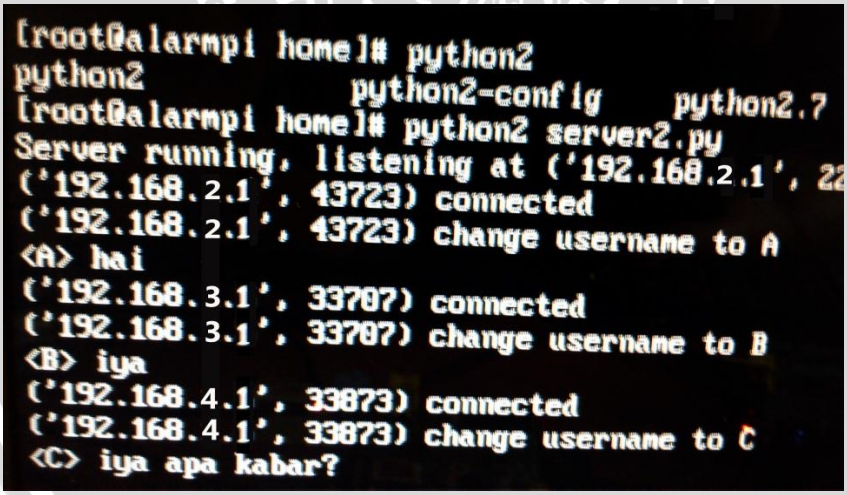
```
192.168.1.6     192.168.1.6     255.255.255.255 UGH   0      0        0
[root@alarmpi ~]# ping 192.168.2.1
PING 192.168.1.4 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=63 time=18.9 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=63 time=79.3 ms

[2]+  Stopped                 ping 192.168.2.1
[root@alarmpi ~]# ping 192.168.3.1
PING 192.168.1.5 (192.168.3.1) 56(84) bytes of data.
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=3.93 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=64 time=1.48 ms

[3]+  Stopped                 ping 192.168.3.1
[root@alarmpi ~]# ping 192.168.4.1
PING 192.168.1.6 (192.168.4.1) 56(84) bytes of data.
64 bytes from 192.168.4.1: icmp_seq=1 ttl=64 time=1.61 ms
64 bytes from 192.168.4.1: icmp_seq=2 ttl=64 time=3.84 ms

[4]+  Stopped                 ping 192.168.4.1
[root@alarmpi ~]#
```

Gambar 5.19 Ping *node* A pada skenario 4

Pada gambar 5.18 Menjelaskan hasil *route* dari *node* A yang menunjukkan bahwa telah terkoneksi dengan *node* B, C, dan D, dari hasil *route* tersebut terlihat bahwa untuk menuju destination *node* B harus melewati gateway *node* D terlebih dahulu, informasi *route* tersebut disimpan di tabel *routing*. Kemudian dilakukan Ping terhadap masing-masing *node* untuk menguji koneksi jaringan, dari hasil tersebut diketahui bahwa ketika dilakukan Ping semua terkoneksi dengan baik yang ditunjukkan dari gambar 5.19.

5.1.4 Pengujian Aplikasi *chat*

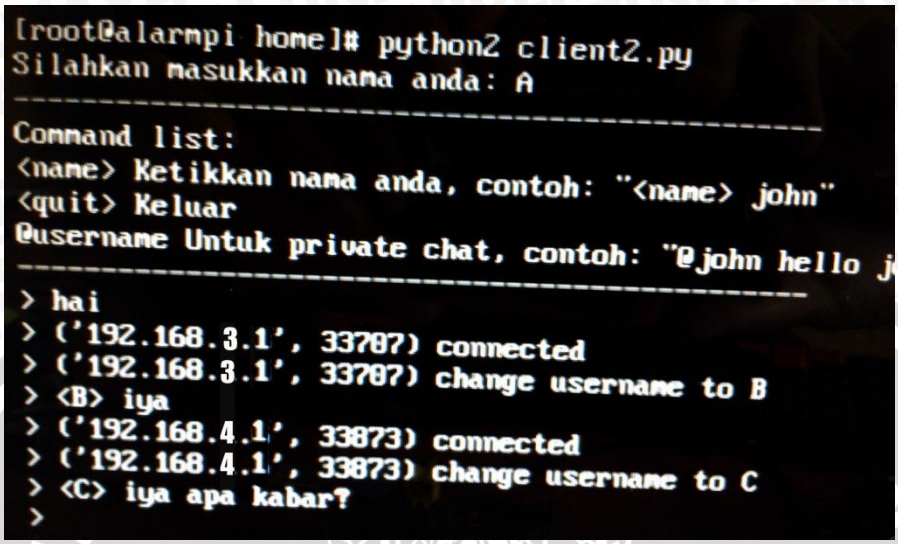
Pengujian aplikasi *chat* adalah menguji komunikasi antar *node Manet* dengan saling mengirimkan pesan, komunikasi ini dibutuhkan ketika sistem komunikasi lain mengalami kerusakan akibat terkena dampak bencana alam. Pengujian dilakukan pada *Client* dan *Server*. Pengujian *Client* meliputi pengujian komunikasi antar *Client*, sedangkan untuk *Server* pengujian dilakukan untuk mengetahui komunikasi antar *Client* yang terkoneksi pada *Server*.



Gambar 5.20 *Server*

Server berfungsi untuk memberikan layanan kepada *Client*, *Server* menerima koneksi dari *Client* yang menghubunginya untuk melakukan komunikasi dengan *Client* lainnya. *Server* berjalan pada IP 192.168.1.3 dengan menggunakan *port* 22226. Setiap ada komunikasi pesan dan *Client* yang baru

berkoneksi dengan *Server* akan ditampilkan di *Server* karena semua komunikasi antar *Client* melewati *Server*.



Gambar 5.21 *Client*

Untuk melakukan pengiriman pesan masing-masing *Client* harus sudah terkoneksi dengan *Server*, setelah terkoneksi dengan *Server*, langkah pertama adalah mengetikkan nama user, kemudian ketikkan pesan untuk mengirim pesan ke *Client* lainnya, pesan yang dikirim akan melalui *Server* terlebih dahulu kemudian oleh *Server* dikirimkan ke semua *Client* yang terhubung dengan *Server* kecuali si pengirim. Pada gambar 5.21 *Client* menggunakan nama user A untuk berkomunikasi dengan *Client* lainnya, sedangkan *Client* lain dengan nama user B dan user C. setelah user A mengetik pesan "hai", user B baru melakukan koneksi ke *Server* dan mengirimkan pesan "iya", kemudian user C baru berkoneksi dengan *Server*, dan mengirim pesan "iya apa kabar". Terdapat juga menu untuk keluar dengan mengetikkan perintah <quit> dan pengiriman pesan secara privat dengan mengetikkan perintah @nama isi pesan.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, Implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian *self configure* menggunakan protokol *routing Babel*, node dapat mengkonfigurasi secara mandiri sehingga bisa langsung terkoneksi dengan *node* lainnya.
2. Berdasarkan hasil pengujian *self healing* menggunakan protokol routing Babel, node harus melakukan reboot terlebih dahulu supaya bisa meneruskan paket dengan jalur routing yang baru.
3. Berdasarkan hasil pengujian letak *node Manet* menggunakan protokol *routing Babel* menunjukkan bahwa *node* dapat berkoneksi secara mandiri sesaat setelah *startup* sistem operasi.
4. Pengujian aplikasi *chat Client Server* menggunakan metode *peer to peer* untuk melakukan komunikasi antar *node* dengan cara *broadcast* pesan ke semua *Client* kecuali *Client* pengirim.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan penelitian ini antara lain :

1. Untuk pengembangan lebih lanjut, diharapkan terdapat *LCD* dan *keyboard mini* yang terpasang pada *Raspberry Pi* sehingga untuk menerapkan teknologi *Manet* menggunakan *Raspberry Pi* bisa semakin *mobile*.
2. Untuk pengembangan lebih lanjut, diharapkan menambahkan aspek *security* pada *Manet.*
3. Untuk pengembangan lebih lanjut, bisa menggunakan *IP* dinamis menggunakan *AHCP (Ad-Hoc Configuration Protocol).*

DAFTAR PUSTAKA

.

[PAB-10] Baskoro, Pranata Ari. 2010. "Jaringan Wi-Fi 802.11n di Area Asrama Mahasiswa ITS Surabaya". Skripsi: Fakultas Teknik Elektro Institut Teknologi Sepuluh November Surabaya.

[FFO-10] Fajar, Firman. 2010. "Perencanaan Coverage Indor Wireless Local Area Network (WLAN) di Hotel Graha Petrokimia Gresik". Skripsi: Fakultas Teknik Elektro Universitas Brawijaya.

[EZZ-11] Zamidra, Efvy. 2011. "Panduan Lengkap Membuat Jaringan Wireless". Jakarta: Elex Media Komputindo.

[RWI-08] Windarti, Rini. 2008. "Perbandingan Media Transmisi Wireless dan Satelite". TA : Teknik Informatika Universitas Sriwijaya.

[KWM-11] Wibowo, Kristiano. 2011. "Menghitung Link Budget untuk Koneksi Radio WLAN Menggunakan Radio Mobile". Naskah Publikasi: Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta.

[NYN -13] Nguyen, cole, herberg. 2013. "Network management of mobile ad hoc network (*MANET*)" : Architechture, Use case, and Aplicability". Draft-nguyen-*MANET*-management-00: Naval research Laboratory.

[JCH-11] Juliusz Chrobocze. 2011, "The Babel Routing Protocol". Request for Comments: 6126, University of Paris, France.

[MFH-10] Muhammad Fauzi Haryadi. 2010. "Analisa dan perancangan aplikasi chatting berbasis web menggunakan flash CS3". Sekolah tinggi manajemen informatika dan computer AMIKOM, Yogyakarta.

[SFS-07] Siagian, Frans. 2007. "Perancangan Komunikasi *Client Server* dan

sistem database". Skripsi: Departemen Teknik Elektro Universitas Sumatra Utara.

[PSH-05] Prasant, Srikanth. 2005. "Ad hoc networks technologies and protocols" University of California.

[STA-04] Staub. 2004. "Performance Comparison of MANET Routing Protocols in Ad-Hoc and Hybrid Network" Computer Science Project, University of Berne, Switzerland.

[KKR-00] James F. Kurose, Keith W. Ross, 2000. "Computer Network Top Down Approach Featuring the Internet", Order the preliminary edition for spring 2000.

[TTP-07] Tofan Teguh Perkasa. 2007. " Analisis Kinerja Voice Over Internet Protocol pada Manet". Skripsi: Teknik Informatika Universitas Kristen Duta Wacana.

[IKW-12] I Komang, Waskitho. 2012. "Vanet untuk Solusi Komunikasi Data di Kawasan Pariwisata Bali". Naskah Publikasi: Jurusan Teknik Informatika Universitas Teknologi Sepuluh Nopember, Surabaya.

[TIY-08] Tri, Isbat, Yuliana. 2008. "Pengiriman Pesan Secara Berantai Pada Daerah Bencana Terisolasi Menggunakan Teknologi Manet". TA: Jurusan Elektronika PENS-ITS, Surabaya.

[SIN-13] Singh. (2013, 03 Maret). Edimax EW-7811Un Wireless 802.11b/g/n Nano USB Adapter Review. Diperoleh 15 Januari 2014, dari http://www.erodov.com/forums/edimax-ew-7811un-wireless-802-11b-g-n-nano-usb-adapter-review/62464.html.