PERANCANGAN SISTEM PERTUKARAN INFORMASI DI PEDESAAN BERBASIS DELAY TOLERANT NETWORK MENGGUNAKAN RASPBERRY PI

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

EVAN HARDYANTO PRAKASITA

0910680013

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan Rahmat serta HidayahNYA kepada penulis berupa nikmat kesehatan, keselamatan, dan nikmat ilmu sehingga penulis dapat menyelesaikan skripsi ini.

Dalam skripsi ini, penulis mengambil judul "**Perancangan Sistem Pertukaran Informasi Di Pedesaan Berbasis Delay Tolerant Network menggunakan Raspberry PI.**" Penelitian ini membahas mengenai membuat sebuah infrastruktur yang murah dan yang bisa mengatasi permasalahan pertukaran informasi di pedesaan.

Dalam menyelesaikan skripsi ini, penulis banyak mendapat bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih kepada beberapa pihak yang telah berperan dalam proses penyusunan skripsi ini kepada:

1. Orang Tua penulis, yang tidak ada hentinya memberikan do'a, dukungan, dan motivasi.
2. Bapak Eko Sakti P, S.Kom., M.Kom. dan Bapak Kasyful Amron, ST., M.Sc. selaku Dosen Pembimbing yang telah memberikan banyak bimbingan dan arahan dalam proses penyusunan skripsi ini.
3. Beberapah pihak yang tidak bisa penulis sebutkan satu-persatu yang membantu dalam proses pembuatan skripsi ini mulai awal hingga selesai.

Penulis menyadari bahwa untuk memajukan skripsi di masa yang akan datang, penulis membutuhkan dan mengharapkan saran dan masukan dari pihak lain. Sehingga semangat dalam memberikan ilmu yang bermanfaat pada masyarakat akan tetap terjaga. Semoga, hasil dari penelitian ini dapat bermanfaat bagi masyarakat. Amin.

Malang, 2013

Penulis

ABSTRACT

Evan Hardyanto Prakasita, 2013. Designing Rural Information Exchange System Based Delay Tolerant Network Using The Raspberry Pi
Advisor: Eko Sakti Pramukantoro, S.Kom., M.Kom. and Kasyful Amron, ST., M.Sc.

Rural is a place that is still lacking or rare occurrence of the exchange of information, both from urban to rural nor from the Internet directly. Rural geographic conditions, power, and cost become an obstacle for the exchange of information in remote rural. The problem can be solved with inexpensive infrastructure with additional use of Delay Tolerant Network (DTN) protocol. Over time, the Raspberry Pi was created that can work like a desktop PC, but the price, size, and power used are all small. IBR-DTN is a software that provides with advantages DTN protocol that uses slightly resource. With so, IBR-DTN implanted in Raspberry will be tested the connectivity and size of files that are sent to the same device. On testing, IBR-DTN software can work automatically when there is a connection and transmission of files on the network that are often disconnected. However, when the default RAM used in the Raspberry Pi is still 512 MB + 100 MB swap file, the file may be sent only at 228 MB. Because at the time of extraction in the package at Raspberry Pi destination, need RAM memory resource equal to the file being sent. So with the addition of the swap file on the Raspberry Pi to 2 GB or more, then the file transfer can be 2 GB or more.

***Keywords**: Rural, IBR-DTN, Raspberry Pi*

ABSTRAKSI

Evan Hardyanto Prakasita, 2013. Perancangan Sistem Pertukaran Informasi Di Pedesaan Berbasis Delay Tolerant Network Menggunakan Raspberry Pi. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing: Eko Sakti Pramukantoro, S.Kom., M.Kom. dan Kasyful Amron, ST., M.Sc.

Pedesaan merupakan tempat yang masih kurang atau jarang terjadinya pertukaran informasi, baik dari kota ke desa maupun dari internet langsung. Kondisi geografis pedesaan, listrik, maupun biaya yang menjadi penghambat terjadinya pertukaran informasi di pedesaan yang terpencil. Masalah tersebut dapat diatasi dengan infrastruktur murah dengan tambahan penggunaan Delay Tolerant Network (DTN) protocol. Seiring berjalannya waktu, diciptakan Raspberry Pi yang mampu bekerja seperti PC Desktop, akan tetapi harga, ukuran, dan daya listrik yang digunakan semuanya kecil. IBR-DTN adalah sebuah perangkat lunak yang menyediakan protokol DTN dengan keunggulannya yaitu menggunakan resource yang sedikit. Dengan begitu, IBR-DTN yang ditanamkan pada Raspberry akan dilakukan pengujian konektivitas dan besarnya *file* yang dikirim terhadap perangkat yang sama. Pada pengujian, perangkat lunak IBR-DTN dapat bekerja secara otomatis apabila ada koneksi dan pengiriman *file* pada jaringan yang sering terputus. Akan tetapi, apabila RAM yang digunakan pada default Raspberry Pi masih 512 MB + 100 MB swap *file*, maka *file* yang boleh dikirim hanya sebesar 228 MB. Karena pada saat paket di ekstraksi di Raspberry Pi tujuan, membutuhkan resource memory RAM sama dengan besarnya *file* yang dikirim. Sehingga dengan penambahan swap *file* pada Raspberry Pi sebesar 2 GB atau lebih, maka pengiriman *file* bisa mencapai 2 GB atau lebih.

Kata kunci: Pedesaan, IBR-DTN, Raspberry Pi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Swap memory : biasa disebut virtual memory digunakan sebagai memory tambahan.

RAM : Random Access Memory, sebagai penyimpanan sementara komputer.

Bundle : Kumpulan dari *file* yang dikirim melalui protokol DTN pada saat yang bersamaan.

USB Drive : Merupakan alat penyimpanan data yang memiliki alat penghubung USB.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pedesaan merupakan tempat yang masih kurang atau jarang terjadinya pertukaran informasi, baik dari kota ke desa maupun dari internet langsung. Kondisi geografis pedesaan, listrik, maupun biaya yang menjadi penghambat terjadinya pertukaran informasi di pedesaan yang terpencil.

Dengan memanfaatkan arsitektur DTN, komunikasi dapat terjamin di sebuah lingkungan dengan konektivitas terbatas dan putus-putus, delay yang besar, dan tingginya *bit error rate* [HUS-11]. Fasilitas infrastruktur yang pada saat ini dapat digunakan adalah penggunaan Raspberry PI [THE-12], dengan maksud agar hemat energi, biaya, dan ukuran. Untuk pendukungnya menggunakan sebuah perankat lunak DTN [SCH-11] untuk mengirim dan menerima data menggunakan protokol DTN.

Pada penelitian sebelumnya yang dikerjakan oleh Emir Husni [HUS-11], yaitu membutuhkan sebuah kereta untuk media pembawa perangkat kerasnya sehingga yang terjangkau hanya daerah di sekitar rel kereta api dan kereta api tidak setiap saat ada. Maka perlu suatu solusi baru untuk mencapai ke daerah pedesaan yang belum terjangkau, yaitu membuat sebuah infrastruktur yang murah, kecil, hemat energi dan bekerja pada konektivitas terbatas. Berdasar dari latar belakang tersebut, diambilah judul "Perancangan Sistem Pertukaran Informasi Di Pedesaan Berbasis Delay Tolerant Network Menggunakan Rapsberry Pi". Hasil dari perancangan infrastruktur ini diharapkan bisa melakukan pertukaran berbagai jenis *file* (berbagai macam informasi) pada konektivitas yang terbatas dengan menggunakan perangkat lunak dan perangkat keras yang berbiaya murah, berukuran kecil, dan hemat energi yang sesuai dengan kondisi wilayah di pedesaan.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka dapat dirumuskan permasalahan pada tugas akhir ini yaitu sebagai berikut:

1. Bagaimana merancang infrastruktur yang murah, berukuran kecil, hemat energi, dan dapat bekerja pada konektivitas yang terbatas.
2. Mengimplementasi infrastruktur dengan melakukan pertukaran informasi dengan berbagai macam jenis informasi (*file*).
3. Mengimplementasi infrastruktur dengan melakukan pertukaran *file* pada jaringan yang putus-sambung.

1.3 Batasan Masalah

Agar permasalahan lebih terfokus, maka tugas akhir ini dibatasi dalam hal:

1. Perancagan dan implementasi DTN dengan menggunakan perangkat lunak IBR-DTN dan perangkat keras Raspberry PI
2. Perancangan dan implementasi DTN hanya digunakan untuk bertukar infromasi.

1.4 Tujuan

Tujuan penulisan tugas akhir ini adalah untuk merancang infrastruktur berbasis DTN di pedesaan sebagai media pertukaran informasi.

1.5 Manfaat

Manfaat yang bisa didapat dari penelitian ini adalah:

1. Memberikan informasi yang dibutuhkan pengguna dengan memanfaatkan infrastruktur berbasis DTN ini.
2. Dapat tetap saling bertukar informasi walaupun di tempat yang memiliki koneksivitas terbatas.

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

Bab I Pendahuluan

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematia penulisan

Bab II Kajian Pustaka dan Dasar Teori

Berisi tentang kajian pustaka dan dasar teori secara luas mengenai perangkat lunak maupun perangkat keras, serta informasi yang diperlukan untuk Perancangan Sistem Pertukaran Informasi di Pedesaan Berbasis Delay Tolerant Network Menggunakan Raspberry Pi.

Bab III Metode Penelitian dan Perancangan

Bab ini menjelaskan tentang langkah-langkah dalam penelitian dan merancang Infrastruktur di Pedesaan Sebagai Media Pertukaran Informasi.

Bab IV Implementasi

Bab ini berisi tentang implementasi infrastruktur yang dibangun, meliputi perancangan infrastruktur dan penanaman DTN pada perangkat keras.

Bab V Pengujian dan Analisis

Bab ini berisi tentang pengujian langsung pada DTN yang ditanamkan pada perangkat keras Raspberry PI dan melakukan analisa infrastruktur yang sudah dibuat pada Raspberry PI yang sudah ditanamkan protokol IBR-DTN.

Bab VI Kesimpulan dan Saran

Pada Bab ini berisi kesimpulan yang diambil berdasarkan analisa hal-hal penting, meliputi keunikan, kelebihan atau kekurangan, serta saran-saran untuk penyempurnaan infrastruktur yang dibuat.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Pedesaan

Menurut kamus besar bahasa indonesia (bahasa.kemdiknas.go.id) menyatakan pedesaan merupakan daerah permukiman penduduk yang dipengaruhi oleh kondisi tanah, iklim, dan air sebagai syarat penting bagi terwujudnnya pola kehidupan agraris penduduk tempat itu.

Dengan berkembangnya jaringan internet, informasi dapat dikirim ke tempat yang jauh dalam beberapa detik saja. Informasi dapat dipertukarkan antara komunitas di perkotaan, maupun antar komunitas di perkotaan, maupun antar komunitas di pedesaan. Akan tetapi akan mengalami sedikit kendala apabila jaringan internet di deploy atau disebarkan ke daerah pedesaan. Infrastruktur di daerah pedesaan cenderung kurang kondusif untuk pentebaran jaringan internet. Hal tersebut tidak terlepas dari kontur tanah yang naik dan turun, dan rumah-rumah penduduk yang berpencar. Selain itu juga kemampuan finansial penduduk yang terbatas [SAN-11].

2.2 Delay Tolerant Network (DTN)

Saat ini internet secara luas beroperasi pada TCP/IP, sehingga menjamin satu set standar protokol di mana jutaan perangkat di seluruh dunia beroperasi. Namun, protokol yang banyak digunakan melalui Internet mungkin tidak berlaku untuk semua jenis jaringan, terutama mereka yang beroperasi di bawah kendala tingginya *delay* dan *losses*. Contoh jaringan tersebut yaitu jaringan nirkabel *terestrial:* jaringan tersebut dapat menghubungkan perangkat mobile wireless, atau perangkat di daerah tanpa banyak infrastruktur untuk mendukung konektivitas berkelanjutan. Contohnya termasuk jaringan di pedesaan/daerah terpencil, jaringan kendaraan, dll. Jaringan tersebut membutuhkan protokol yang mempertimbangkan kebutuhan komunikasi jaringan tersebut, termasuk konektivitas link, delay, *data rate* asimetri, pengalamatan, mekanisme reliabilitas, kualitas layanan, dll.

Delay Tolerant Network (DTN) adalah jaringan overlay untuk keperluan umum yang beroperasi di atas berbagai jaringan regional, termasuk Internet. DTN memungkinkan jaringan regional dengan berbagai karakteristik delay untuk beroperasi dengan menyediakan mekanisme untuk menerjemahkan antara parameter jaringan masing-masing. Oleh karena itu, protokol yang mendasari dan teknologi untuk jaringan ini daerah mungkin berbeda jauh, tetapi fleksibilitas dari arsitektur DTN memungkinkan mereka untuk terhubung satu sama lain [FAL-03].

Delay Toleran Network (DTN) adalah arsitektur jaringan *end-to-end* yang menyediakan komunikasi pada lingkungan yang tidak mendukung terjadinya komunikasi. Pada lingkungan tersebut, sering terjadinya konektivitas yang putus-putus, besarnya *delay*, dan tingginya kegagalan. Untuk memberikan layanan pada lingkungan tersebut, Bundle Protocol berada pada *aplication layer* lalu membentuk *overlay* jaringan untuk melakukan *store-and-forward* [SCO-07].

Terdapat beberapa faktor kenapa Delay Tolerant Network digunakan dan menjadi memotivasi untuk dieksplorasi menurut RFC4838 [CER-07]:

> Protokol Internet yang ada saat ini tidak dapat bekerja dengan baik pada beberapa lingkungan, karena beberapa asumsi dasar:
>
> - Jalur *end-to-end* antara sumber harus ada selama durasi sesi komunikasi
> - (Untuk komunikasi yang handal) bahwa mentransmisi ulang berdasarkan *feedback* yang tepat waktu dan stabil dari penerima data merupakan cara yang efektif untuk memperbaiki kesalahan.
> - E*nd-to-end loss* relatif kecil.
> - Bahwa semua router dan *end station* mendukung protokol TCP/IP.
> - Bahwa aplikasi tidak perlu khawatir tentang kinerja komunikasi.
> - Mekanisme keamanan berbasis *end point* yang memadai untuk memenuhi keamanan yang paling dikhawatirkan.

- *Packet switching* adalah abstraksi yang paling tepat untuk interoperabilitas dan kinerja.
- Memilih satu rute antara pengirim dan penerima cukup untuk mencapai kinerja komunikasi yang diterima.

Arsitektur DTN dipahami untuk tidak mempermasalahkan pada sebagian besar asumsi tersebut, yang semuanya didasarkan pada sejumlah prinsip desain yang dirangkum berikut ini:

- Menggunakan *variabel-length messages* (bukan *stream* atau paket yang terbatas ukuran) sebagai abstraksi komunikasi untuk membantu meningkatkan kemampuan jaringan untuk membuat pilihan penjadwalan/jalur keputusan yang baik bila memungkinkan.
- Menggunakan sintaks penamaan yang mendukung berbagai penamaan dan *addressing conventions* untuk meningkatkan interoperabilitas.
- Menggunakan penyimpanan antar jaringan untuk mendukung operasi *store-adn-forward* melalui beberapa jalur, dan lebih berpotensi rentang waktu lama (misalnya, untuk mendukung operasi pada lingkungan di mana banyak dan/atau tidak adanya jalur *end-to-end* yang mungkin pernah ada); tidak membutuhkan keandalan *end-to-end*.
- Menyediakan mekanisme keamanan yang melindungi infrastruktur dari penggunaan yang tidak sah dengan membatalkan *traffic* secepat mungkin.
- Menyediakan layanan yang *coarse-grained*, pilihan pengiriman, dan cara untuk mengatur *lifetime* yang berguna pada data untuk memungkinkan jaringan untuk memberikan data yang lebih baik dalam melayani kebutuhan aplikasi.

Penggunaan lapisan bundel yang diarahkan tidak hanya oleh prinsip-prinsip desain itu sendiri, tetapi juga oleh beberapa prinsip desain aplikasi:

- Aplikasi harus meminimalkan jumlah pertukaran round-trip.
- Aplikasi harus mengatasi *restart* setelah terjadi kegagalan ketika transaksi pada jaringan masih tertunda.

Aplikasi harus menginformasikan ke jaringan dari masa yang berguna dan kepentingan data yang relatif akan dikirimkan.

2.2.1 Bundle Protocol

Bundle Protocol adalah sebuah protokol Delay Tolerant Network (DTN) yand didesain untuk komunikasi jaringan yang tidak stabil. Sehingga membentuk sebuah kelompok blok data kedalam *bundle* dan mengirimkan *bundle* tersebut dengan menggunakan teknik *store-and forward.*

Bundle Protocol menghubungkan beberapa subnet menjadi satu jaringan. Mereka menyediakan layanan yang akan menahan lalu mengirim ulang dan juga menyimpan paket untuk waktu yang lama. Sinyal transmiter menjamin pengiriman paket. Sehingga, dapat dengan mudah mengatasi masalah konektivitas internet seperti *bandwidth delay* dan sambungan yang terputus.

Bundle itu sendiri merupakan data unit protokol dari *bundle protocol* DTN. Setiap *bundle* terdiri dari dua rangkaian atau lebih dari "blok" data protokol, yang melayani berbagai keperluan. Beberapa contoh dari *bundle* yang sama (unit yang sama dari DTN protokol data) mungkin ada secara bersamaan di berbagai bagian jaringan yang lain -- mungkin dalam representasi yang berbeda -- dalam memori lokal untuk satu atau lebih node bundel dan/atau transit diantara *nodes*. Dalam konteks pengoperasian *bundle node*, sebuah *bundle* adalah contoh beberapa *bundle* dalam jaringan yang ada di memori lokal node tersebut [SCO-07].

Pada *bundle protocol* terdapat *Self-Delimiting Numeric Value* (SDNV) yang penggunaannya menjadi kunci desain dari *bundle protocol*. Skema pengkodean SDNV mengadaptasi dari Abstract Syntax Notation One Basic Encoding Rules dalam nilai identifier objek. Sebuah SDNV merupakan nilai numerik yang dikodekan pada oktet N, setiap akhir dari kebanyakan bit (MSB) diset menjadi nol; MSB pada setiap oktet yang lain pada SDNV harus di set ke 1. Nilai yang dikodekan kedalam SDNV adalah bilangan biner yang diperoleh dengan menggabungkan menjadi satu bit dari 7 bit yang paling signifikan dari setiap oktet dari SDNV.

Tabel 2.1 merupakan *Bundle Block* utama yang menampilkan dasar dari *bundle protocol*.

Tabel 2.1 Primary Bundle Block

Version	Proc. Flags
Block Length	
Destination scheme offset (*)	Destination SSP offset (*)
Source scheme offset (*)	Source SSP offset (*)
Report scheme offset (*)	Report-to SSP offset (*)
Custodian scheme offset (*)	Custodian SSP offset (*)
Creation Timestamp time (*)	
Creation Timestamp sequence number (*)	
Lifetime (*)	
Dictionary length (*)	
Dictionary byte array (variable)	
[Fragment offset (*)]	
[Total application data unit length (*)]	

Sumber : [SCO-07]

Lifetime: Kolom lifetime adalah SDNV yang menunjukkan waktu di mana payload bundel tidak akan lagi berguna, yang dikodekan sebagai jumlah pada detik telah melewati waktu penciptaan. Sebuah *bundle* akan habis masanya pada

saat waktu saat ini lebih besar dari waktu penciptaan bundel ditambah lifetime sebagaimana yang ditentukan dalam *primary block bundle*, maka *node bundle* tidak perlu lagi mempertahankan atau meneruskan *bundle*, *bundle* dapat dihapus dari jaringan. *Bundle expiration* dapat terjadi pada setiap titik dalam pengolahan *bundle*. Ketika bundel habis masanya, agen *bundle protocol* harus menghapus *bundle* dengan alasan "*lifetime expired*". Dalam melakukan penghapusan *bundle* terdapat langkah-langkahnya yaitu:

- Langkah 1: Jika penyimpanan terkendala "*Custody accepted*" saat ini mencegah berkas ini agar tidak dibuang, dan tujuan bundel adalah endpoint tunggal, maka:
  - *Custody* node dilepaskan.
  - Sebuah laporan status penghapusan *bundle* mengutip alasan tersebut untuk penghapusan harus dihasilkan, maka laporan bundel ini ditujukan ke endpoint ID.

  Sebaliknya, jika "*request reporting of bundle deletion*" *flag* dalam kolom laporan status permintaan *bundle* ini diatur ke 1, maka laporan status penghapusan *bundle* mengutip alasan tersebut untuk penghapusan harus dihasilkan, maka laporan bundel ini ditujukan ke endpoint ID.
- Langkah 2: Semua kendala penyimpanan bundel itu harus dihilangkan.

Version: Sebuah field 1-byte yang menunjukkan versi dari protokol bundel yang dibangun blok ini. Dokumen ini menjelaskan versi 0x06 dari protokol bundel.

Bundle Processing Control Flag: Bidang Pengendalian Flags Pengolahan Bundle merupakan SDNV yang berisi pengolahan flag kontrol bundel.

Block Length: Bidang Blok Panjang merupakan SDNV yang berisi Panjang agregat dari semua bidang yang tersisa dari blok.

Destination Scheme Offset: The Destination Skema bidang Offset berisi offset dalam array byte kamus nama skema titik akhir ID tujuan bundel ini, yaitu, titik akhir yang berisi node (s) di mana bundel yang harus diberikan.

Destination SSP Offset: The Destination SSP bidang Offset berisi offset dalam array byte kamus bagian skema-spesifik dari endpoint ID tujuan bundel itu.

Source Skema Offset: Sumber Skema bidang Offset berisi offset dalam array byte kamus nama skema titik akhir ID sumber nominal bundel ini, yaitu, titik akhir nominal berisi node yang bundel itu awalnya dikirim.

Source SSP Offset: Sumber SSP bidang Offset berisi offset dalam array byte kamus bagian skema-spesifik dari endpoint ID sumber nominal bundel itu.

Report-to Offset: Laporan-to Skema bidang Offset berisi offset dalam array byte kamus nama skema ID dari titik akhir yang statusnya laporan yang berkaitan dengan forwarding dan pengiriman berkas ini harus ditransmisikan.

Report-to SSP Offset: Laporan-ke lapangan SSP Offset berisi offset dalam array byte kamus bagian skema-spesifik dari ID dari titik akhir yang statusnya laporan yang berkaitan dengan forwarding dan pengiriman berkas ini harus ditransmisikan .

Custodian Scheme Offset: The "saat kustodian endpoint ID" dari blok bundel primer mengidentifikasi titik akhir yang keanggotaannya meliputi node yang paling baru diterima tahanan bundel pada forwarding bundel ini. Kustodian Skema bidang Offset berisi offset dalam array byte kamus nama skema arus kustodian endpoint ID.

Custodian SSP Offset: The Kustodian SSP bidang Offset berisi offset dalam array byte kamus khusus skema arus kustodian endpoint ID.

Creation Timestamp: Penciptaan timestamp adalah sepasang SDNVs itu, dengan sumber endpoint ID dan (jika bundel adalah fragmen) fragmen offset dan panjang payload, berfungsi untuk mengidentifikasi bundel. The SDNV pertama timestamp adalah waktu pembuatan thebundle, sementara yang kedua adalah

penciptaan urutan nomor timestamp bundel itu. Waktu pembuatan bundel adalah waktu - dinyatakan dalam detik sejak awal tahun 2000, di Coordinated Universal Time (UTC) skala [UTC] - di mana permintaan transmisi diterima yang mengakibatkan penciptaan bundel. Urutan count adalah nilai terbaru (pada waktu di mana permintaan itu transmisi diterima) dari monoton meningkat kontra bilangan bulat positif yang dikelola oleh agen protokol bundel node sumber yang dapat diatur ulang ke nol setiap kali kemajuan zaman saat ini dengan satu detik. Sebuah sumber Bundle Protokol Agen tidak harus membuat dua bundel yang berbeda dengan sumber yang sama endpoint ID dan bundel penciptaan timestamp. Kombinasi sumber endpoint ID dan bundel penciptaan timestamp karena berfungsi untuk mengidentifikasi permintaan transmisi tunggal, memungkinkan untuk diakui oleh aplikasi penerima (menyediakan sumber endpoint ID tidak "dtn: none").

Dictionary Length: Bidang Panjang Kamus merupakan SDNV yang berisi panjang dari array byte kamus.

Dictionary: Bidang Kamus adalah sebuah array byte dibentuk dengan menggabungkan nama diakhiri null-skema dan SSPS semua endpoint ID direferensikan oleh setiap bidang dalam Blok Primer bersama, berpotensi, ID endpoint lain yang dirujuk oleh ladang di lain TBD DTN blok protokol . Panjangnya diberikan oleh nilai field Panjang Kamus.

Fragment Offset: Jika Bundle Pengolahan Pengendalian Flags blok ini primer menunjukkan bahwa bundel adalah fragmen, maka bidang Offset Fragmen adalah SDNV menunjukkan offset dari awal aplikasi data unit asli di mana byte terdiri dari payload ini bundel berada. Jika tidak, maka bidang Offset Fragmen dihilangkan dari blok tersebut.

Total Application Data Unit Length: Jika Bundle Pengolahan Pengendalian Flags blok ini primer menunjukkan bahwa bundel adalah fragmen, maka Total Application Data Unit Length adalah SDNV menunjukkan panjang total dari aplikasi data unit asli yang payload bundel ini adalah bagian. Jika tidak, maka Total Application Data Unit Length dihilangkan dari blok tersebut.

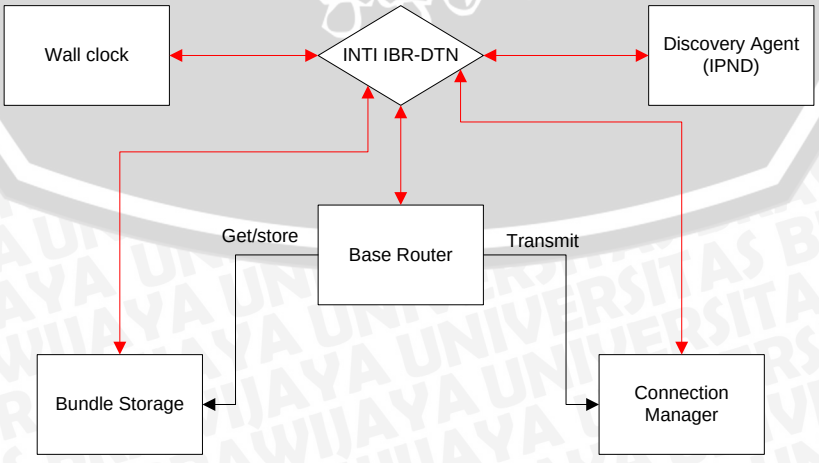
2.3 IBR-DTN

Protokol yang digunakan dalam DTN adalah *Bundle Protocol* [SCO-07]. Akan tetapi, implementasi yang dilakukan pada Bundle Protokol DTN2, tidak cocok untuk penggunaan pada *embedded system. Embedded system* tidak memiliki banyak aplikasi yang banyak, terbatas oleh biaya dan energi yang ckup besar. Tujuan diciptakannya IBR-DTN adalah menjadi *Bundle Protocol* yang kompatibel dan dirancang untuk berjalan pada *embedded system* dan juga dikhususkan dibangun pada sistem operasi Linux yang ditanam pada *embedded system* [SEB-11].

Berbagai macam jenis *file* bisa dikirim melalui *Bundle Protocol* ini, seperti *file* yang berukuran besar maupun kecil dan juga *file* yang memiliki ekstensi yang unik maupun tidak ada ekstensi-nya tetap bisa dikirim pada saat terjadinya putus sambung pada jaringan. Akan tetapi, hanya pertukaran *file* yang dilakukan secara real time tidak bisa dilakukan ketika pada jaringan dengan konektivitas yang putus sambung, misalnya pertukaran informasi dengan VoIP, Chatting, Video Streaming, dan Video Call.

2.3.1 Arsitektur

Gambar 2.1 menunjukkan keseluruhan arsitektur dari IBR-DTN. Sebagaimana IBR-DTN ditargetkan untuk *embedded systems*, sehingga *system library*-nya juga didesain untuk *embedded systems*.



Gambar 2.1 Arsitektur IBR-DTN

Sumber: Kajian pustaka dan dasar teori

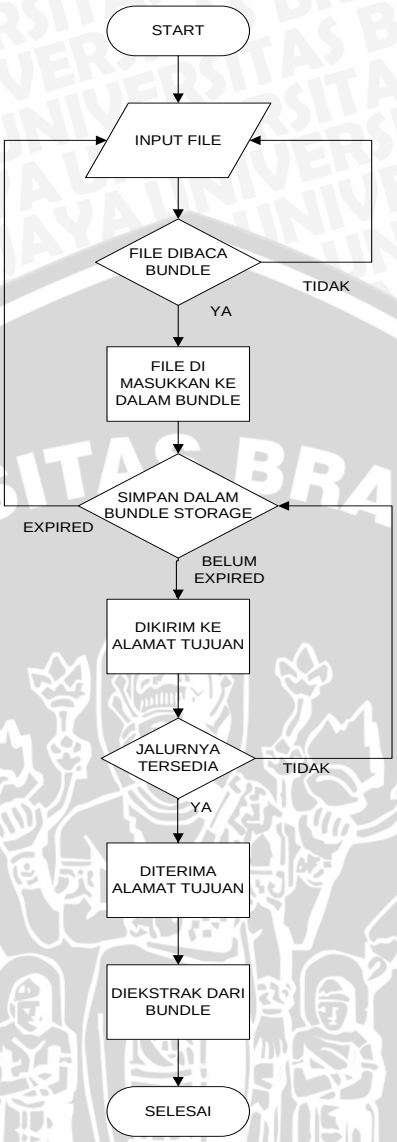
Seperti yang terlihat pada gambar 2.1, IBR-DTN terdiri dari tiga modul utama yang memberikan fungsi dasar layanan modul inti DTN yang semuanya diatur oleh inti dari IBR-DTN:

a. **Connection Manager**: Connection Manager merupakan modul pada IBR-DTN yang menyediakan konektivitas antar DTN atau yang bisa disebut lapisan konvergensi. Untuk setiap konektivitas, memiliki *interface* yang berbeda-beda sesuai dengan jenis lapisan konvergensinya (TCP, UDP, dan HTTP).

b. **Bundle Storage**: Sebagai DTN yang menggunakan paradigma menyimpan dan meneruskan, sebuah node DTN harus mempunyai kemampuan untuk menyimpan *bundle* sementara untuk jangka waktu yang tidak ditentukan. Sebuah modul dari Bundle Storage menyediakan sebuah interface untuk menyimpan dan mengirim bundle dengan parameter yang berbeda, contohnya id unik bundle atau tujuannya.

c. **Base Router**: Base Router merupakan modul yang mengatur pengiriman *bundle* ke DTN tujuan. Jika ada *bundle* yang akan dikirim ke infrastruktur tujuan, maka inti IBR-DTN menyuruh Base Router untuk mengambil *bundle* yang sudah ada di Bundle Storage, lalu meminta lapisan konvergensi (TCP, UDP, atau HTTP) yang sesuai kepada Connection Manager untuk bisa dikirim ke tujuan [SEB-11].

Melihat penjelasan diatas, didapatkan diagram alir untuk pengiriman sebuah *file*. Sehingga disimpulkan dengan gambar berikut:



Gambar 2.2 Alur pengiriman *file*

Sumber: Kajian pustaka dan dasar teori

2.3.2 Kinerja Pada Embedded Device

Evaluasi yang dilakukan oleh michael doering dkk [DOE-08] terhadap IBR-DTN secara kuantitas dan kualitas untuk menganalisa kemampuan dan efisiensinya. Tabel 2.1 menunjukkan konsumsi *resource* memori dan *storage* dari IBR-DTN. Bukan hanya kecilnya aplikasi tersebut, tetapi juga rendahnya konsumsi memori, membuat IBR-DTN cocok untuk berjalan pada *embedded device* ataupun pada *device* yang mempunyai *resource* terbatas.

Tabel 2.2 Pemakaian RAM oleh IBR-DTN

	DTN2	IBR-DTN
Size of daemon aplication	21,9 MB	114 kB
Size of attached libraries	7,1 MB	605 kB
RAM usage (swappable VZS)	41,7 MB	3,4 MB
RAM usage (non-swappable VZS)	4,7 MB	1,7 MB

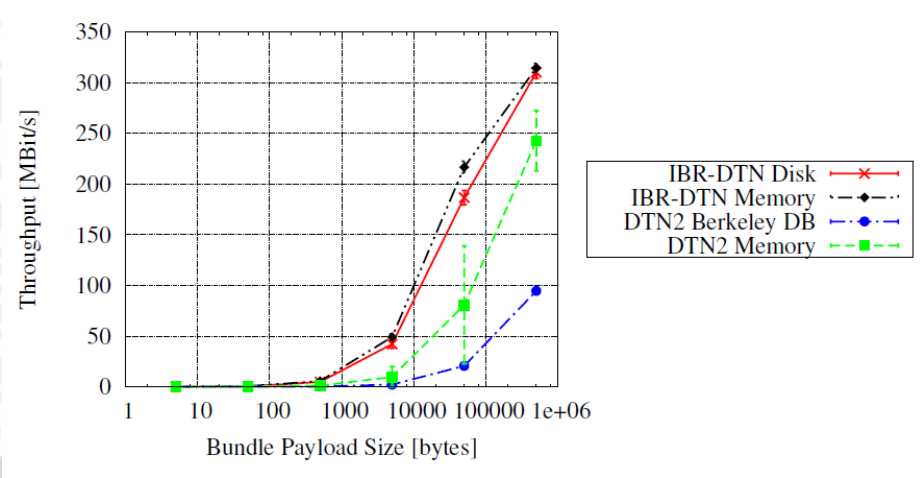
Sumber: [DOE-08]

Untuk mengetahui hasil penggunaan IBR-DTN pada embedded device [SEB-11], dilakukan tes pada OpenWRT dengan platform RouterStation Pro (RSPro). Untuk setiap tes dilakukan 10 percobaan dengan 100 *bundles* terdiri dari 1 MByte. *SD Card* digunakan sebagai penyimpanan *disk*. The RSPro terhubung menggunakan Gbit LAN Port. Dengan menggunakan konfigurasi ini kinerja iPerf puncak antara dua papan RSpro menggunakan TCP adalah 251 Mbit/s. Data *throughput* total IBR-DTN antara dua RouterStation Pro adalah 33,85 Mbit/s (± 5,44 Mbit/s). *Throughput* dari dan ke PC adalah dengan 45.36 Mbit/s (± 12.10 Mbit/s) sedikit lebih cepat. Kecepatan penyimpanan yang digunakan dapat membatasi *throughput*, karena semua berkas harus simpan ke *SD Card*. Hasil penelitian menunjukkan, bahwa akan ada kemungkinan untuk menyerap 54 Mbit/s 802.11g WiFi Link (net data rate 22,5 Mbit/s).

2.3.3 Kinerja Jaringan IBR-DTN dengan DTN2

Dalam penelitian yang dilakukan sebelumnya [SEB-11] untuk mengukur kinerja jaringan pada IBR-DTN dan DTN2 yang dilakkukan sebagai berikut: Dua node terhubung melalui Ethernet, tetapi konektivitas diblokir. Pada node pengirim digunakan dtnsend untuk menyuntikkan 1000 bundel yang ditujukan ke node penerima. Setelah berkas telah diterima oleh daemon pengirim kita membuka koneksi dan mengukur waktu sampai semua berkas daemon pengirim telah tiba di penerima. Ini indikasi yang bagus dari kinerja jaringan dari sebuah daemon dengan pendekatan ini menghilangkan pengaruh API dan kinerja *storage write* pada pengirim. Namun tes ini masih termasuk waktu yang dihabiskan dalam fungsi *storage retrieval* pengirim dan fungsi *storage writing* pengirim. Pada IBR-DTN menggunakan iptables untuk penanganan sambungan, untuk DTN2

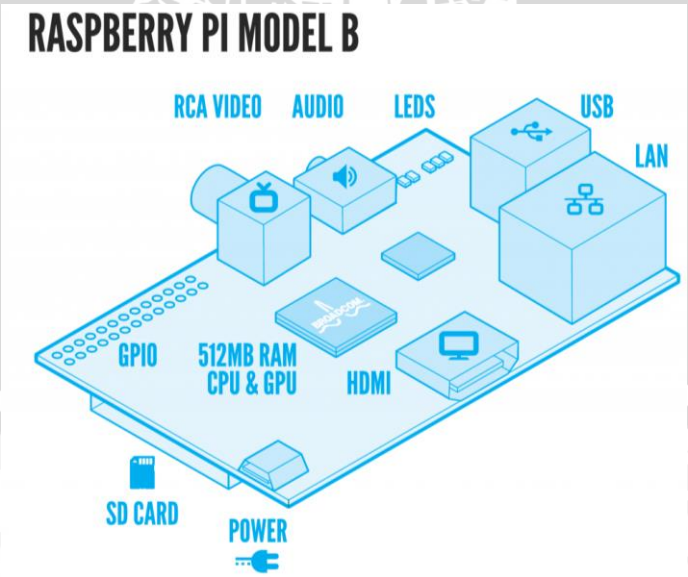
menggunakan builtin TCP API untuk menambah atau menghapus rute yang sesuai dengan cepat. Eksperimen ini diulangi dengan berbagai ukuran *payload bundle* mulai dari 5 byte untuk 500 kBytes. Tes dilakukan tidak berjalan secara paralel: untuk memastikan bahwa semua *bundle* pada percobaan sebelumnya telah diterima sebelum mulai mengirim *bundle* baru. Dalam penelitian ini, menggunakan lapisan konvergensi TCP. Untuk setiap ukuran *bundle payload* dan setian jenis penyimpanan dilakukan 10 percobaan. Daemon IBR-DTN berjalan terus menerus dan belum berulang antara percobaan berikutnya. Daemon DTN2 harus direstart pada setiap ukuran *bundle* yang berbeda, karena saat pengamatan, kinerjanya yang terus memburuk untuk masing-masing percobaan ketika menggunakan penyimpanan Memori. Efek ini memanifestasikan DTN2 sebagai standar deviasi yang tinggi dalam tes memori DTN2, hasil dari tes untuk setiap ukuran *bundle* yang diulang 10 kali. Hasil penelitian ini dapat dilihat pada gambar 2.3 dengan skala logaritmik sumbu x. Aritmatika mean dan deviasi standar untuk setiap ukuran *payload*. Pertama dapat dilihat bahwa bahkan dengan ukuran *bundle* 500 kByte tidak DTN2 atau IBR-DTN mampu memaksimalkan penggunaan link (480 MBit). IBR-DTN mencapai kecepatan rata-rata ~ 310 MBit sementara DTN2 maksimal hanya di ~ 245 MBit untuk penyimpanan berbasis memori dan secara signifikan lebih lambat (~ 98 MBit) untuk penyimpanan yang lebih berbasis disk yang relevan praksis. Hal ini menjadi masalah utama bagi DTN, di mana itu adalah tujuan untuk mentransfer data sebanyak mungkin selama kontak yang terjadi secara singkat. Untuk IBR-DTN alasannya adalah, bahwa standar TCP dan parameter Protokol Bundle disesuaikan untuk link nirkabel dengan bandwidth yang lebih kecil. Itu dianggap hal yang sama berlaku untuk DTN2. Dalam semua *testcases* IBR-DTN melebihi DTN2 baik di memori dan kasus *disk-based storage*. Bahkan juga, *disk-based storage* IBR-DTN melebihi *disk-based storage* DTN2 untuk semua ukuran *bundle* yang diuji.



Gambar 2.3 Kinerja keluaran
Sumber: [SEB-11]

2.4 Raspberry PI

The Raspberry Pi adalah sebuah komputer berukuran kartu kredit yang dapat dihubungkan ke TV dan keyboard. Ini adalah PC kecil yang dapat digunakan untuk banyak hal selayaknya PC desktop, seperti spreadsheet, pengolah kata dan permainan. Raspberry Pi juga memainkan video dengan definisi tinggi [RAS-13].



Gambar 2.3 Raspberry Pi
Sumber: [RAS-13]

2.4.1 Spesifikasi

Dari melihat gambar 2.3, sudah dapat dibuktikan bahwa ukurannya sebesar kartu kredit. Raspberry Pi berukuran 85,60mm x 56mm x 21mm, dengan tambahan dari SC card dan penghubung yang ada di pinggiran dari papan tersebut dengan berat 45g. *System on Chip* (SoC) yang digunakan adalah Broadcom BCM2835. System tersebut memiliki ARM1176JZFS, dengan *floating point*, berkerja pada 700Mhz, dan sebuah Videocore 4 GPU. (*Graphic Procesing Unit*) GPU tersebut mampu memutar berkualitas BluRay, menggunakan H.264 pada 40MBit/s. GPU tersebut mempunyai fast 3D core yang diakses menggunakan suplai dari OpenGL ES2.0 dan OpenVG *libraries*. Pada Raspberry Pi model B memiliki RAM 512 MB dengan 2 USB port dan sebuah ethernet port 10/100 Ethernet RJ45. Untuk booting, dibutuhkan sebuah *SD Card* atau sebuah Hard Drive yang di sambungkan melalui USB port. Terdapat port suara 3,5mm jack dan HDMI, power yang dibutuhkan yaitu 700 mA ~1200 mA (3,5W) [RAS-13].

2.5 Sistem Operasi Debian

Sistem operasi Debian merupakan salah satu sistem operasi yang menggunakan kernel Linux atau kernel Linux FreeBSD. Linux merupakan sistem operasi yang *free* (bebas), sehingga banyak pengguna linux yang melakukan experiment dengan maksud agar bisa membuat sistem operasi yang lebih baik lagi.

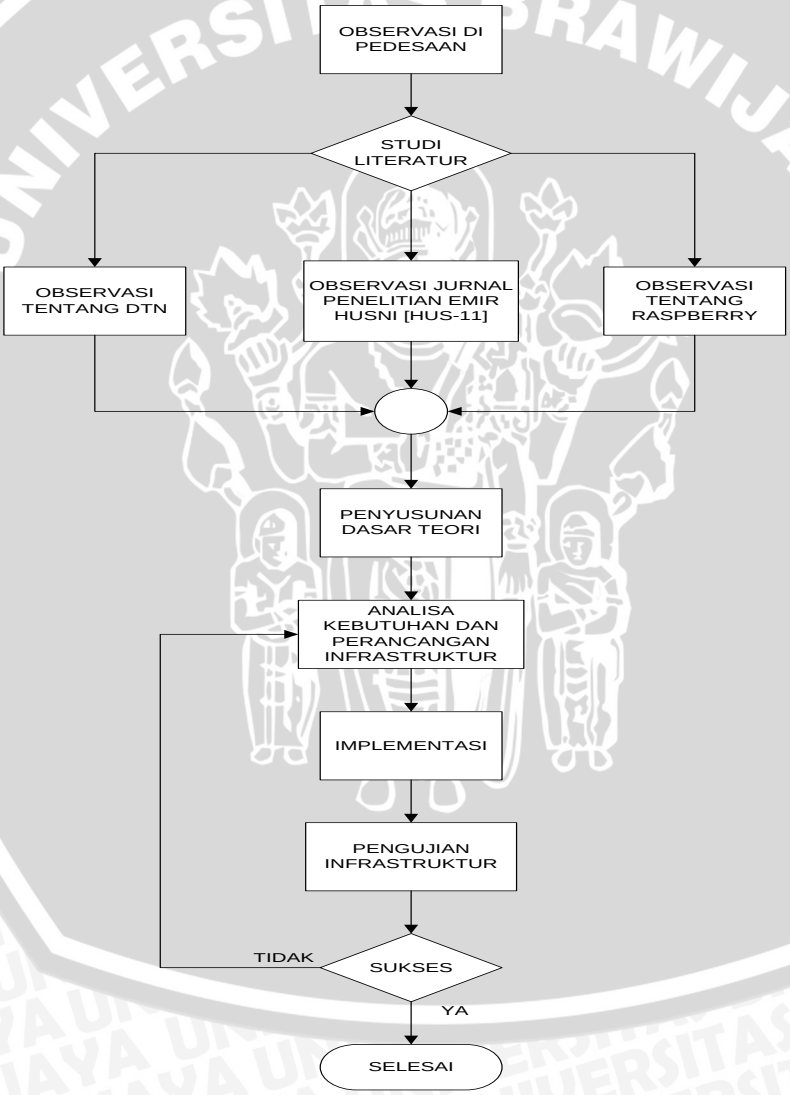
Tentu saja, hal yang orang inginkan adalah aplikasi perangkat lunak yaitu program untuk membantu mereka mendapatkan apa yang mereka ingin lakukan, seperti dari mengedit dokumen untuk menjalankan bisnis, bermain game, menulis ataupun untuk membuat perangkat lunak lagi. Debian datang dengan lebih dari 37500 paket (*software precompiled* yang terbungkus dalam format yang bagus untuk kemudahan instalasi di mesin anda) yang semuanya gratis.

Pada dasarnya, ini sedikit seperti sebuah menara. Dasar adalah kernel, di atas itu semua alat dasar, selanjutnya adalah semua perangkat lunak yang dijalankan pada komputer dan di bagian atas menara adalah Debian [DEB-13].

BAB III
METODE PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Pada bagian ini dijelaskan mengenai metode penelitian yang dilakukan dalam penelitian perancangan sistem pertukaran informasi di pedesaan berbasis Delay Tolerant Network menggunakan Raspberry Pi.



Gambar 3.1 Diagram alir metode penelitian

Sumber: Metode penelitian dan perancangan

3.1.1 Observasi

Dengan mengadakan penelitian dan menganalisa secara langsung terhadap kondisi masyarakat di pedesaan, sehingga dapat dilihat kebutuhan infrastruktur yang dirancang, dimana observasi ini meliputi pengamatan terhadap perangkat keras, perangkat lunak, dan sebagainya. Observasi dimulai dengan melihat permasalahan yang ada dipedesaan, lalu membaca jurnal penelitian milik Emir Husni pada tahun 2011 dengan judul "Delay Tolerant Network Based Internet Services for Remote Areas Using Train Systems" [HUS-11]. Didapatkan sebuah kelemahan yaitu penggunaan *resource* komputer, daya listrik dan ditambah biaya penggunaan atau pengoperasian yang cukup besar, yang bisa dianggap cukup memberatkan masyarakat di pedesaan yang rata-rata memiliki daya listrik terbatas dan biaya yang terbatas.

3.1.2 Studi Literatur

Studi literatur digunakan untuk menunjang dasar teori yang akan digunakan. Teori-teori pendukung tersebut meliputi:

a. Pedesaan
b. Delay Tolerant Network
c. IBR-DTN
d. Raspberry PI
e. Sistem Operasi Debian
f. Jurnal Penelelitian Emir Husni yang berjudul "Delay Tolerant Network Based Internet Services for Remote Areas Using Train Systems"

3.1.3 Perancangan

Dalam perancangan diawali dengan melakukan analisis kebutuhan. Analisa kebutuhan dilakukan dengan mengidentifikasi dan menentukan kebutuhan apa saja yang diperlukan untuk merancang infrastruktur di pedesaan sebagai

media pertukaran informasi. Setelah menentukan jenis kebutuhan yang diperlukan, tahap selanjutnya adalah melakukan perancangan infrastruktur berbasis Delay Tolerant Network yang akan dilakukan pada dua perangkat Raspberry Pi yang masing-masing terhubung melalui jaringan adhoc WiFi yang disediakan oleh Access Point atau laptop. Perangcangan dilakukan setelah mengetahui bahwa dipedesaan masih banyak yang belum banyak melakukan pertukaran informasi. Hasil dari observasi dipedesaan dan mempelajari penelitian yang sudah dilakukan sebelumnya, akan dijadikan acuan untuk melakukan implementasi.

3.1.4 Implementasi

Implementasi infrastruktur dilakukan dengan pengacu pada perancangan infrastruktur. Implementasi infrastruktur dapat dimulai dengan menentukan spesifikasi sistem yang akan digunakan sebagai mengetahui perangkat keras dan perangkat lunak apa saja yang akan digunakan. Tahap selanjutnya adalah melakukan beberapa konfigurasi dari perangkat lunak IBR-DTN. Setelah IBR-DTN siap digunakan, tahapan berikutnya adalah pembuatan swap memory yang akan ditanamkan pada USB Flash Drive yang akan digunakan sebagai tambahan memori pada Raspberry Pi dan konfigurasi fstab untuk membuat swap memory dan USB Flash Drive bisa digunakan saat booting, dan terakhir melakukan konfigurasi untuk WiFi *receiver* agar bisa terkoneksi ke jaringan sercara otomatis saat booting dilakukan.

3.1.5 Pengujian dan analisis

Pengujian infrastruktur dilakukan untuk mengetahui apakah kinerja dan performa infratruktur telah memenuhi spesifikasi kebutuhan yang melandasinya. Pada tahapan ini dilakukan pengujian terhadap pertukaran informasi melalui protokol DTN dengan skenario yang berbeda-beda, yaitu membandingan pertukaran data tanpa menggunakan protokol DTN dengan pertukaran data dengan menggunakan protokol DTN yang kedua-duanya dikondisi jaringan yang putus-sambung. Dengan begitu mana yang lebih berguna pada jaringan yang putus-sambung.

3.1.6 Pengambilan kesimpulan dan saran

Kesimpulan didasarkan atas pengujian dan analisis yang dilakukan di dalam proses penelitian. Isi dari kesimpulan diharapkan dapat menjadi acuan untuk pemakaian protokol jaringan yang sesuai dengan kebutuhan serta konfigurasi DTN pada Raspberry Pi. Tahap terakhir dari penulisan adalah saran yang berisi hal-hal yang diperlukan dalam rangka pengembangan topik skripsi selanjutnya maupun perbaikan yang harus dilakukan sesuai dengan kesimpulan.

3.2 Analisis Kebutuhan

Dengan melakukan analisis kebutuhan terhadap infrastruktur yang akan dirancang, sehingga dapat diketahui kondisi atau kemampuan yang harus dimiliki infrastruktur untuk memenuhi apa yang diinginkan atau diisyaratkan pemakai.

3.2.1 Analisis Lingkungan

Penelitian ini diimplementasi di situasi jaringan yang sering putus sambung, biasanya pada pedesaan. Dimana Raspberry Pi penyebar informasi akan selalu terhubung ke WiFi, sedangkan Raspberry Pi yang terletak di pedesaan yang biasanya sering terganggu konektivitasnya, bisa terjadi karena mati listrik atau sambungan ke wifi terputus karena kondisi lingkungan pedesaan tersebut. Kondisi seperti ini yang membutuhkan DTN untuk melakukan pengiriman data.

3.2.2 Analisis Infrastruktur

Sesuai dengan latar belakang yang sudah disebutkan diatas, maka infrastruktur yang harus dibangun harus mempunyai biaya murah, berukuran kecil, hemat energi, dengan tambahan sebuah *software* yang dapat membuat infrastruktur tersebut bekerja pada kondisi konektivitas yang terbatas.

3.2.3 Analisis Fungsional dan non-Fungsional

Secara fungsi, ada dua jenis kebutuhan untuk membangun infrastruktur:

1. Kebutuhan fungsional

Kebutuhan fungsional adalah kebutuhan utama yang harus ada agar infrastruktur ini dapat terwujud, berikut kebutuhan fungsional yang dimiliki sistem, yaitu:

a. Infrastruktur mampu melakukan pertukaran informasi antar device.

b. Infrastruktur mampu melanjutkan pertukaran informasi antar device walaupun sempat terjadi putusnya koneksi antar device.

2. Kebutuhan non-fungsional

Kebutuhan non-fungsional adalah sesuatu yang diperlukan oleh infrastruktur ini agar dapat berjalan dengan optimal. Berikut kebutuhan non-fungsional, yaitu:

a. Availbility:

Infrastruktur ini dapat diakses selama 24 jam sehari, 7 hari seminggu.

b. Usability:

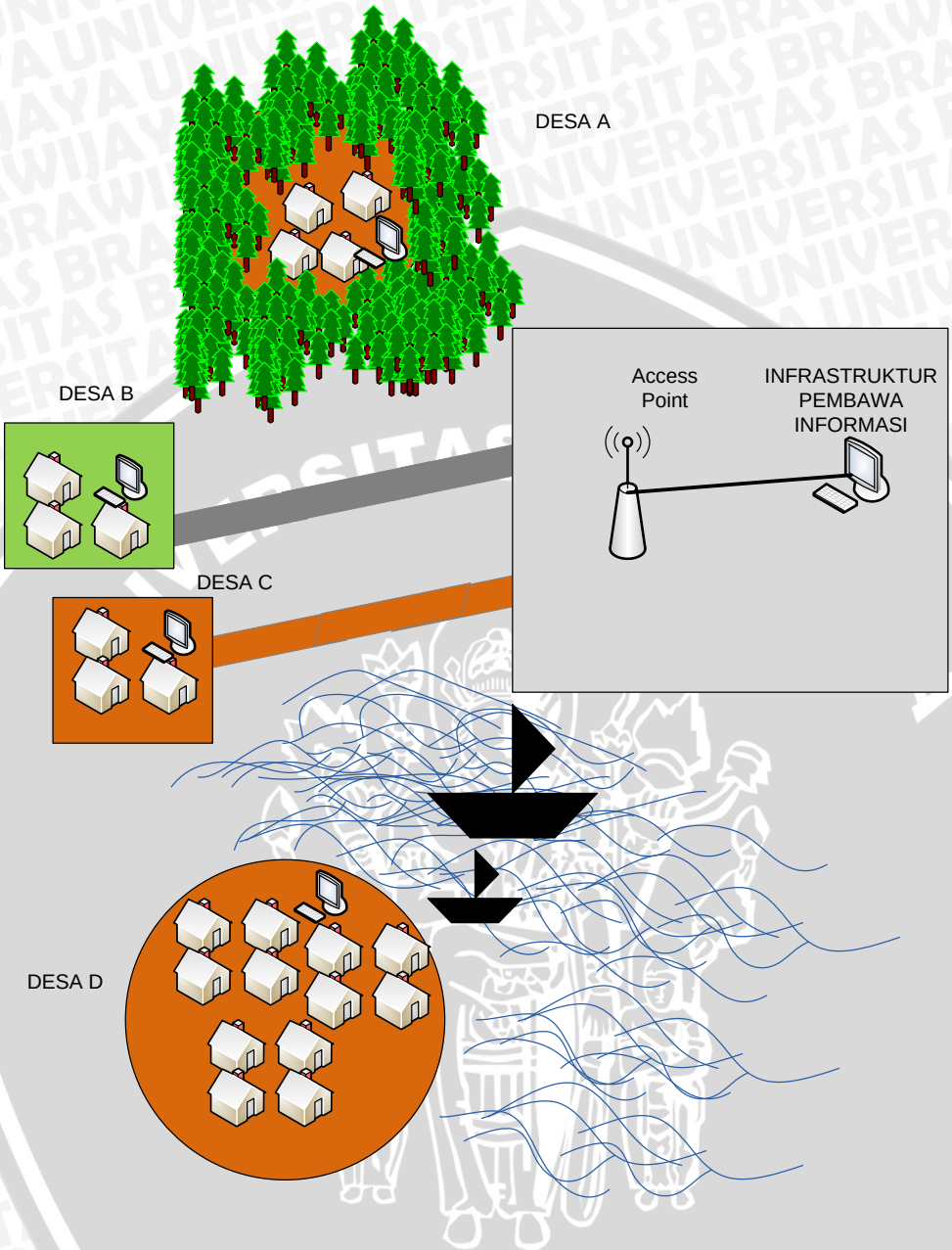
Infrastruktur ini dapat digunakan dengan mudah dan sesuai dengan kebutuhan masyarakan yang membutuhkan pertukaran informasi serta mudah untuk dikembangkan lebih lanjut.

3.3 Perancangan Infrastruktur

Perancangan infrastruktur merupakan proses merancang atau mendesain sebuah infrastruktur yang isinya adalah kebutuhan serta prosedur teknis infrastruktur yang akan dibuat. Perancangan infrastruktur dimulai dari site plan desa, desain jaringan, perancangan skema, dan perancangan sistem.

a. Perancangan site plan desa

Pada gambar 3.2 menggambarkan bahwa ada beberapa jenis lingkungan pedesaan yang melingkupinya, seperti hutan ataupun lautan.



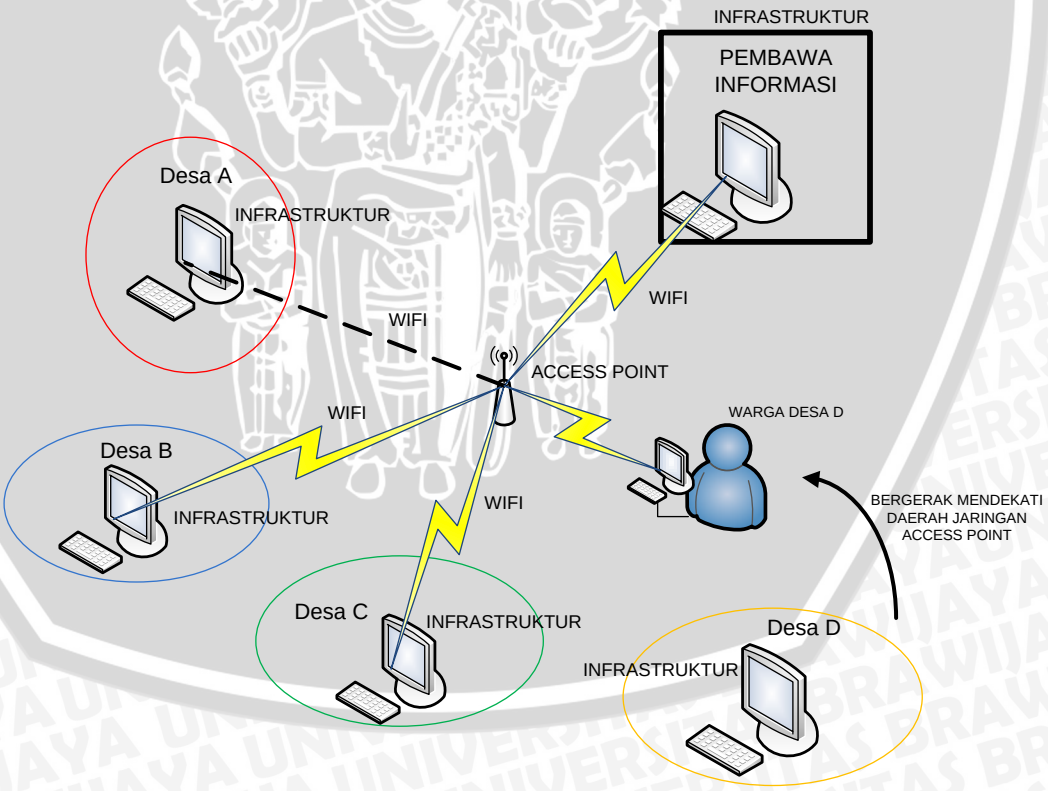
Gambar 3.2 Site Plan Desa
Sumber : Perancangan

- Pada Desa A, desa tersebut lingkungannya didaerah pegunungan atau bisa dikatakan daerah yang tertutupi oleh hutan. Jarak dari desa ke *Access Point* tidak terlalu jauh, akan tetapi jaringan konektivitas terganggu karena adanya pepohonan di daerah lingkungan tersebut, sehingga terjadinya konektivitas yang sering terjadi putus-sambung dan delay yang cukup tinggi.

- Pada desa B dan C, daerah lingkungan desa tersebut sangat terbuka dan mudah menjangkau jaringan *Access Point* yang terhubung langsung ke infrastruktur pembawa informasi
- Pada desa D, daerah lingkungannya terkendala dengan jarak dari desa ke *Access Point* dan juga daerah tersebut dikelilingi oleh laut. Akan tetapi, banyak warganya yang bisa mendekat ke daerah terjngkaunya ke jaringan WiFi dengan menggunakan kapal ataupun perahu.

b. Perancangan jaringan

Seluruh desa memiliki infrastruktur yang akan dibuat dan seluruhnya tersambung ke jaringan *Access Point* pembawa informasi dengan menggunakan WiFi sebagai media komunikasinya.



Gambar 3.3 Desain Jaringan
Sumber: Perancangan

Dengan menggunakan media WiFi, biaya dalam penerapan infrastruktur ini bisa lebih murah dan infrastruktur bisa digunakan secara mobile apabila jaringan WiFi tersebut terputus.

Pada Desa A desain jaringan yang digunakan yaitu dengan konektivitas yang putus sambung terhadap *Access Point* yang terhubung melalui media WiFi.

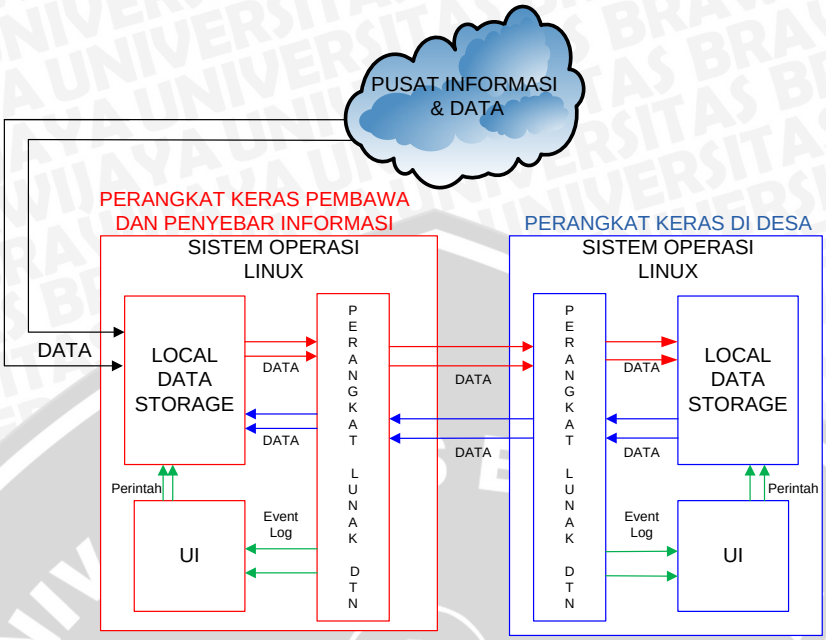
Pada Desa B dan desa C, direncanakan sebuah jaringan dengan kondisi yang sangat stabil terkoneksi dengan *Access Point* yang terhubung melalui media WiFi.

Pada Desa D, koneksi jaringan antara infrastruktur yang berada di Desa D terhadap *Access Point* terputus. Dengan infrastruktur tersebut, warga Desa D dapat membawa infrastruktur tersebut untuk mendekati ke daerah *Access Point* agar infrastruktur yang dibawanya dapat terhubung kembali ke *Access Point* melalui media WiFi.

c. Perancangan arsitektur sistem

Arsitektur sistem dari jaringan berbasis DTN sebagai media pertukaran informasi di pedesaan dengan komponen utama sebagai berikut:

- Pusat Informasi dan Data, merupakan sumber data atau informasi yang akan di simpan di perangkat keras pembawa dan penyebar data dan informasi.
- Local Data Storage, digunakan untuk melakukan pemrosesan dan mengatur data atau informasi yang akan dikirim ataupun disimpan.
- User Interface (UI), digunakan sebagai tampilan default dari sistem operasi yang membantu user untuk mengakses data atau informasi yang dibutuhkan user.
- Perangkat Lunak DTN, sebagai inti dari perangkat lunak untuk memproses seluruh transmisi data menggunakan protokol DTN.



Gambar 3.4 Arsitektur Sistem

Sumber: Metode penelitian dan perancangan

d. Desain penggunaan sistem

Desain langkah-langkah penggunaan sistem, dimaksudkan agar pengguna bisa mudah memahami cara penggunaan sistem ini seperti yang di gambarkan pada lampiran 1.

1. Hal pertama yang harus dilakukan adalah mempersiapkan minimal 2 device (device pengirim dan device penerima) yang akan digunakan pada sistem ini dan memastikan apakah kedua device sudah hidup juga sudah terkoneksi ke Access Point melalui media WiFi.

2. Kedua device harus sudah terinstal *bundle protocol* DTN, jika sudah maka aktifkan *bundle protocol* pada kedua device agar keduanya bisa saling terhubung dan saling mengetahui satu sama lain.

3. Aktifkan *tools bundle protocol* yang digunakan untuk membuat folder yang akan digunakan sebagai tempat pengiriman *file* (folder *outbox*) pada *device* pengirim.

4. Aktifkan *tools bundle protocol* yang digunakan untuk membuat folder yang akan digunakan sebagai tempat penerima *file* (folder *inbox*) pada *device* penerima.

5. Tentukan *file* yang akan dikirm dari *device* pengirim ke *device* penerima, besar *file* mempengaruhi lama pengiriman dan nama *file* tidak boleh ada spasi dan karakter simbol. Lalu *file* di taruh pada folder *outbox* yang sudah di-*setting* menjadi tempat pengiriman *file*. Maka dalam beberapa saat *file* akan menghilang dari folder tersebut yang dikarenakan *file* harus dimasukkan kedalam *bundle storage* sebelum melakukan pengiriman melalui jaringan.

6. Apabila saat pengiriman *file* terjadi terputusnya sambungan antar dua device, maka *file* yang dikirim akan tetap ada pada bundle storage dan menunggu perintah selanjutnya dari *bundle protocol*.

7. *Bundle protocol* akan tetap menyimpan *file* tersebut pada *bundle storage* untuk lifetime tertentu yang sudah ditentukan pada saat awal pengiriman *file*. *Bundle protocol* akan menghapus *file* yang *lifetime*-nya sudah *expired* karena *lifetime* yang lebih sedikit saat menunggu koneksinya kembali terhubung. Tetapi apabila lifetime sudah diatur selama tak hingga, maka *file* tersebut akan menunggu sampai kapanpun hingga koneksi kembali terhubung kembali.

8. Saat sudah terhubung kembali, *bundle protocol* akan mem-*push file* yang berada pada *bundle storage* sehinigga dikirim kembali secara otomatis tanpa perlu lagi mengirim ulang *file* yang sama.

9. Setelah beberapa waktu, *file* yang dikirim sudah sampai pada *device* penerima lalu di ekstrak dari *bundle* lalu disimpan ke folder inbox yang sudah ditentukan sebelumnya.

BAB IV
IMPLEMENTASI

4.1 Spesifikasi Infrastruktur

Spesifikasi perangkat keras dan perangkat lunak yang akan digunakan dalam perancangan sistem pertukaran informasi di pedesaan berbasis Delay Tolerant Network menggunakan Raspberry Pi dijelaskan pada tabel berikut.

Tabel 4.1 Spesifikasi Infrastruktur

Nama Perangkat Keras	Raspberry Pi
Prosesor	ARM1176JZFS 700 Mhz
Memori (RAM)	512MB
Storage	SDHC Card 4GB
Interface	Ethernet LAN, Wireless LAN
Eksternal Storage	USB Drive (Flash Disk)
Konsumsi Listrik	700mA – 1200mA (3,5 Watt)
Sistem Operasi	Linux Debian “wheezy”
Perangkat Lunak DTN	IBR-DTN

Sumber: Implementasi

4.2 Implementasi Infrastruktur

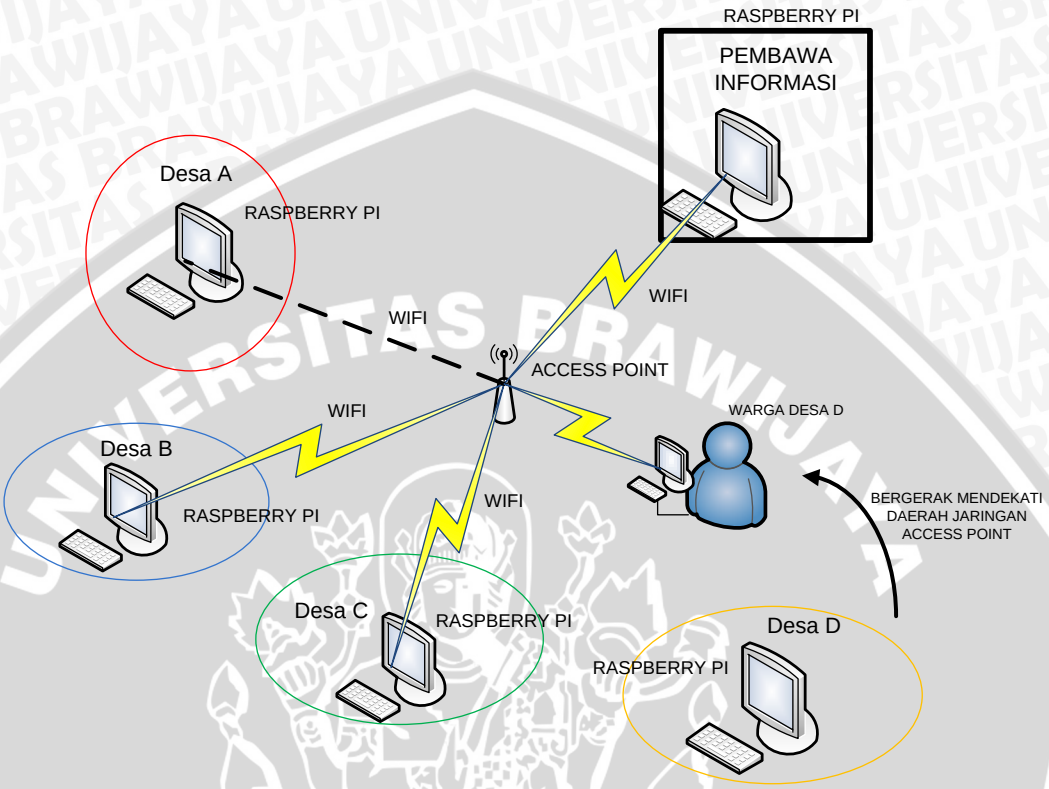
Setelah mendapatkan hasil dari analisis kebutuhan, perancangan, dan mengetahui spesifikasi infrastruktur, tahapan penelitian selanjutnya adalah dengan melakukan implementasi pada perangkat keras dan perangkat lunak yang digunakan sebagai infrastruktur ini.

4.2.1 Konfigurasi Perangkat Keras

Raspberry Pi digunakan sebagai pernagkat keras yang menggantikan peran PC Desktop. Selain itu, Raspberry Pi dipilih karena beberapa alasan yaitu:

- Raspberry Pi menggunakan daya listrik hanya 3,5 Watt dan harga yang murah.

- Raspberry Pi memiliki ukuran sebesar kartu kredit, sehingga apabila diperlukan pertukaran informasi secara *mobile* pun bisa dilakukan.



Gambar 4.1 Konfigurasi Perangkat Keras

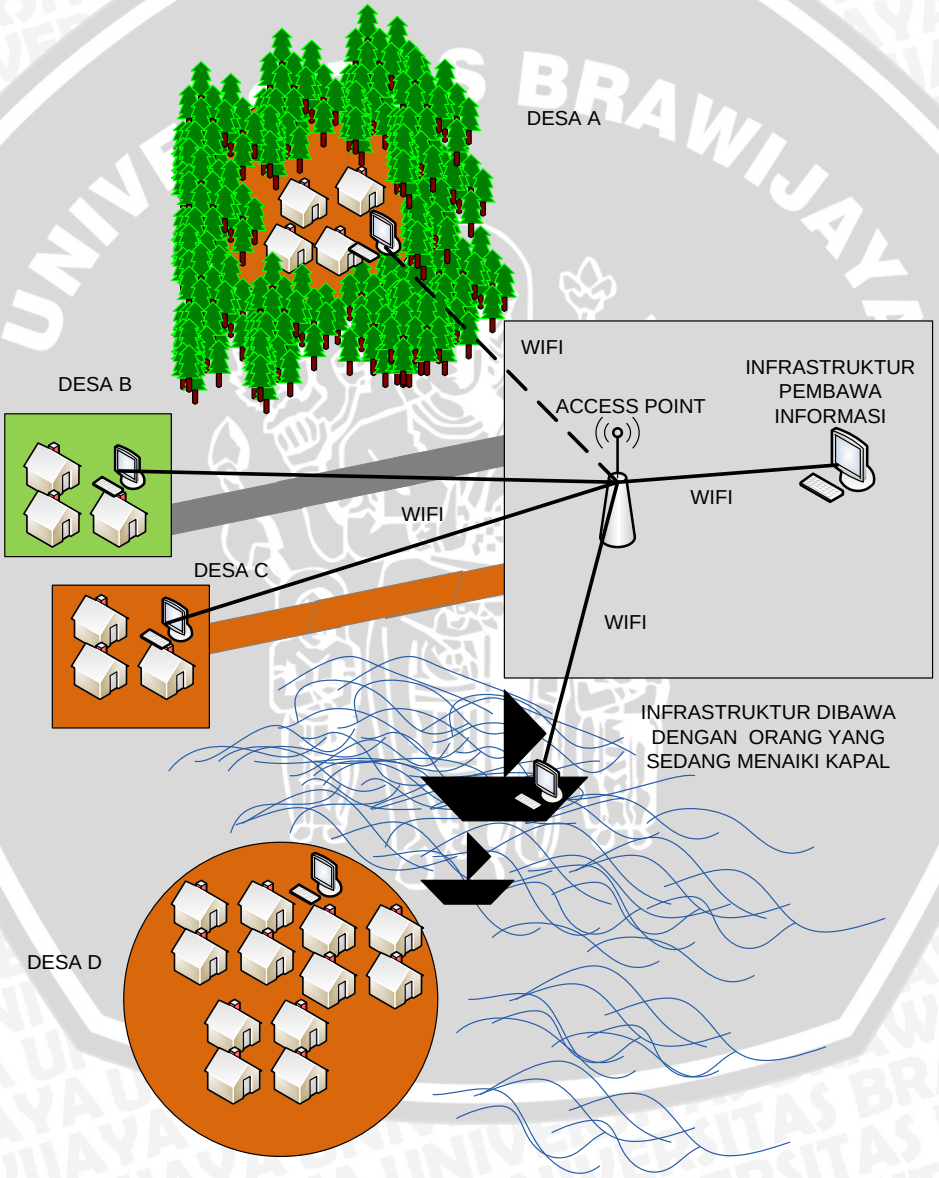
Sumber: Implementasi

WiFi digunakan sebagai jaringan *intermediate* antar Raspberry Pi di desa dengan Raspberry Pi pembawa informasi. Ada beberapa alasan mengapa dipilihnya WiFi, yaitu:

- Jaringan WiFi dapat dipasangkan secara fleksibel dan dapat mengurangi masalah yang disebabkan koneksi melalui kabel
- Dalam membangun WiFi, hanya mengeluarkan biaya pada awal pemasangan dan untuk biaya pemeliharaan berkala juga tidak semahal jika menggunakan satelit atau layanan komunikasi selular, jadi

pengguna di pedesaan bisa mendapatkan keuntungan dari layanan dengan harga yang tidak mahal.

Implementasi perangkat keras yang digunakan, harus sesuai dengan perancangan pada skema infratuktur pada gambar 3.3. Setiap desa memiliki situasi lingkungan yang berbeda-beda, sehingga dalam penerapannya dan penggunaan infrastruktur setiap desa berbeda juga.



Gambar 4.2 Skema pada site plan
Sumber : Implementasi

Skenario pada gambar 4.2 terdapat berbagai kondisi yang dijelaskan dibawah ini:

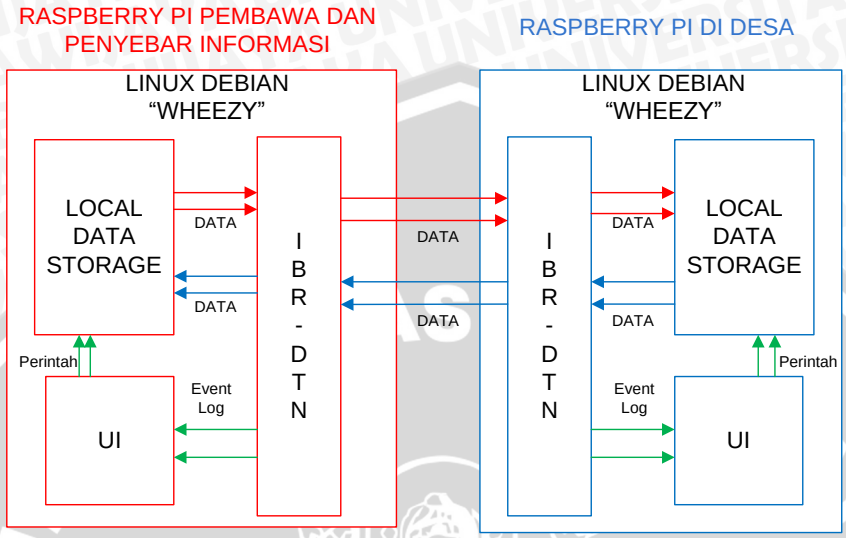
1. Desa A, konektivitas infrastruktur dari desa A ke WiFi ataupun kondisi kelistrikan di desa A sering kali putus sambung. Sehingga, dengan adanya infrastruktur yang dibuat ini, warga desa A dapat merasakan manfaat dari infrastruktur ini dan dapat melakukan pertukaran informasi tanpa khawatir *file* yang dikirim dikembalikan lagi ataupun hilang.
2. Desa B dan Desa C, infrastruktur dari desa B atau desa C dapat terkoneksi secara stabil dan bisa dapat bertukar informasi setiap saat melalui media WiFi yang terhubung ke *Access Point*.
3. Desa D, koneksi dari desa D ke WiFi terputus dan Infrastruktur tidak dapat terhubung kembali sama sekali meskipun sebelumnya sedang melakukan pertukaran informasi. Akan tetapi, *file* yang sudah dikirim disimpan kedalam antrian yang akan dikirim secara otomatis apabila terhubung ke jaringan kembali.
4. Karena di Desa D tidak dapat memungkinkan terjadinya koneksi melalui WiFi. Oleh karena itu, Infrastruktur yang berada di Desa D dibawa oleh warga Desa D dengan mendekat ke daerah yang terjangkau oleh Access Point dengan menaiki kapal atau perahu, sehingga kegiatan pertukaran informasi yang tertunda bisa dilanjutkan kembali tanpa khawatir *file* yang sudah dikirim sebelumnya hilang.

4.2.2 Konfigurasi Perangkat Lunak

Sistem Operasi Linux Debian Wheezy digunakan sebagai sistem operasi pada infrastruktur ini. Ada beberapa alasan mengapa sistem operasi ini yang digunakan:

- Sistem Operasi Linux Debian Wheezy memiliki versi yang khusus untuk dijalankan pada Raspberry Pi

- Sistem Operasi ini tidak berbayar, sehingga sistem operasi ini cocok sekali untuk warga desa yang memiliki dana terbatas



Gambar 4.3 Konfigurasi Sistem Operasi dan Perangkat Lunak

Sumber: Implementasi

IBR-DTN digunakan sebagai perangkat lunak yang menjalankan protokol DTN pada infrastruktur ini. IBR-DTN ini dipilih karena alasan berikut:

- IBR-DTN menggunakan resource RAM pada Raspberry Pi yang sedikit, sehingga cocok sekali jika ditanamkan pada Raspberry Pi ini yang hanya memiliki RAM sebesar 512 Mb.
- IBR-DTN dapat berjalan pada sistem operasi Linux Debian Wheezy.

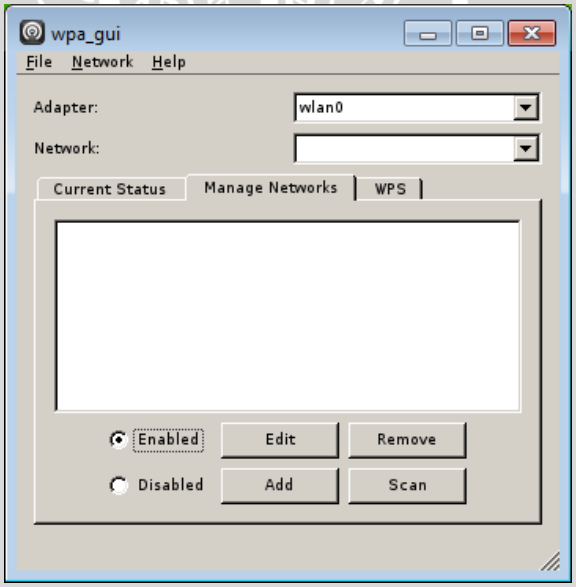
4.3 Instalasi dan Konfigurasi Raspberry Pi

Proses ini dilakukan dengan pemasangan Sistem Operasi Debian Wheezy pada Raspberry Pi dan penyetelannya dapat dilihat pada lampiran 2. Sebelumnya, harus diperhatikan dahulu jenis sistem operasi yang akan di gunakan, karena Raspberry Pi menggunakan prosesor berjenis ARM. Sistem Operasi tersebut karena yang khusus untuk Raspberry Pi, maka biasa disebut dengan nama Raspbian.

Pada instalasi, harus dipastikan bahwa yang paling sistem bisa masuk ke bagian *Desktop* dan lalu bisa melakukan *remote*. *Desktop* penting sekali bagi user yang ingin menggunakan sebagai media pengiriman informasi, karena *desktop* memberikan tampilan yang *user friendly* supaya dapat digunakan dengan mudah.

4.4 Instalasi dan Konfigurasi WiFi Receiver

Supaya bisa terhubung dengan jaringan WiFi yang sudah disiapkan dengan nama "kesini" tanpa *authentication* atau *open*, diperlukan sebuah USB WiFi *receiver* untuk dapat terhubung ke WiFi dengan nama "kesini". Secara default, pada raspberry pi yang menggunakan sistem operasi Debian Wheezy, tidak perlu menginstalasi paket yang lain-lain jika hanya digunakan sebagai receiver. Pada desktop Raspberry Pi, terdapat aplikasi WPA Supplicant yang dapat digunakan sebagai pengatur jaringan yang akan digunakan.



Gambar 4.4 WPA Supplicant gui

Sumber: Implementasi

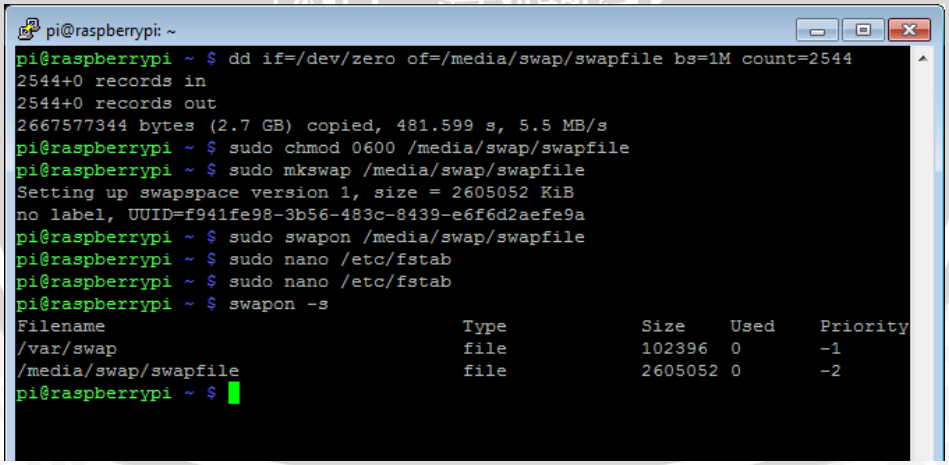
Supaya dapat terhubung dengan jaringan yaitu dengan melakukan konfigurasi pada tampilan WPA yang sudah disediakan oleh Debian Wheezy. Setelah selesai melakukan konfigurasi, diperlukan suatu skript untuk melakukan koneksi otomatis ke jaringan tersebut. *File* rc.local [lapiran1] merupakan *file* pada

Debian Wheezy yang digunakan untuk menjalankan suatu perintah secara otomatis pada saat Raspberry Pi booting.

4.5 Instalasi dan Konfigurasi Swap Memory di USB Flash Drive

Untuk menambah kapasitas memory yang akan digunakan pada Raspberry Pi, diperlukan swap memory yang cukup besar. Penggunaan swap memory bisa pada *SD Card* atau pada USB Flash Drive dengan kapasitas yang besar. Pada penelitian ini, menggunakan USB Flash Drive dengan kapasitas 8Gb sebagai media penyimpanan swap memory dan bundle storage dari IBR-DTN.

Biasanya sebuah Drive yang ditancapkan pada salah satu *port usb* akan terdeksi pada /dev/sda*. Lalu langkah selanjutnya adalah dengan membuat *file* yang akan disimpan pada USB Flash Drive yang sudah terpasang dengan menggunakan perintah dd. Setelah selesai pembuatan *file*, dilakukan chmod agar *file* hanya user root yang membaca dan mengubah isi dari *file* tersebut. Tahap selanjutnya adalah dengan membuat *file* yang sudah dibuat menjadi *file* swap dengan perintah mkswap, lalu dilakukan pengenalan sistem terhadap *file* swap dengan perintah swapon.

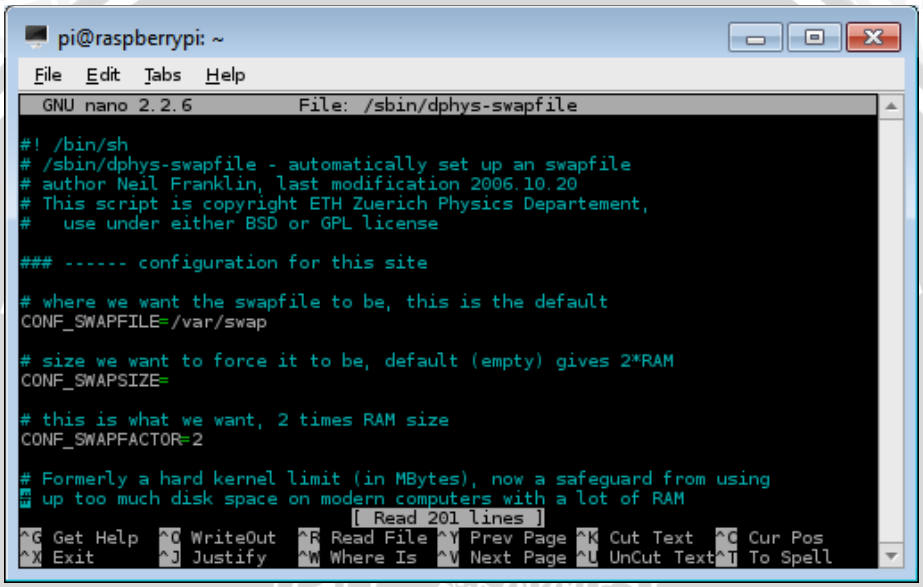


Gambar 4.5 Konfigurasi Swap *File* tambahan

Sumber: Implementasi

Secara default, Debian Wheezy yang berjalan pada Raspberry Pi sudah menyediakan konfigurasi *file* swap yang sudah terpasang dan berjalan secara

otomatis pada saat booting dilakukan. Agar *file* swap yang sudah dibuat bisa digunakan secara otomatis, dilakukan penyetelan pada *file* konfigurasi dphys-swap*file* yang terdapat pada /sbin/dphys-swap*file*. Dan terakhir yaitu dengan mengubah *file* /etc/dphys-swap*file* dengan mengubah nilai dari swap*file* sesuai dengan *file* swap yang sudah dibuat. Akan tetapi penggunaan melalui konfigurasi ini, sebaiknya *file* yang digunakan berada pada *SD Card*, karena *USB Drive*r tidak ter-*mount* secara otomatis.



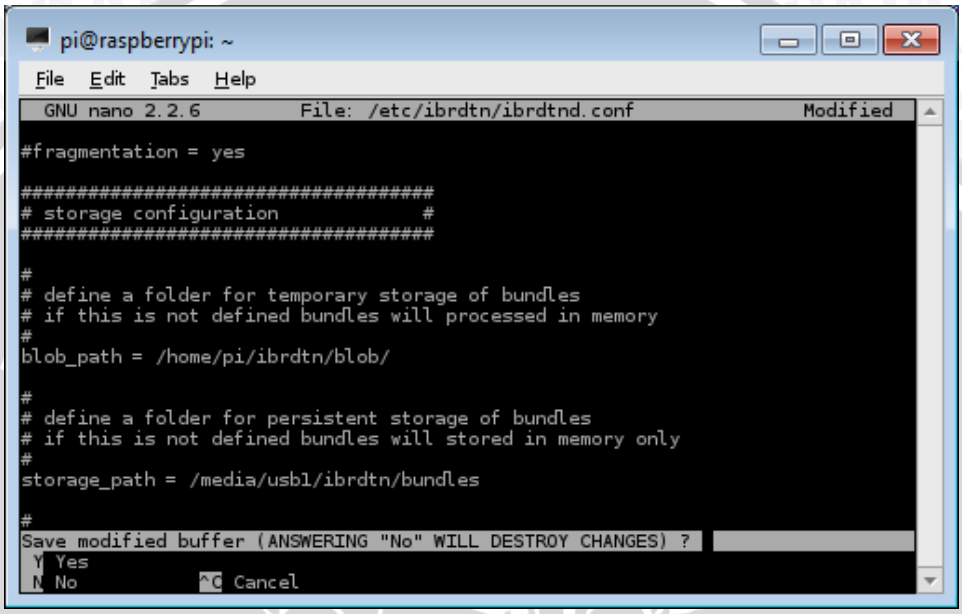
Gambar 4.6 Konfigurasi dphys-swap*file*

Sumber: Implementasi

4.6 Instalasi dan Konfigurasi IBR-DTN

Sebelum melakukan instalali IBR-DTN, yang harus dilakukan adalah menambah list dari repository yang terletak di /etc/apt/sources.list. Jika menambahkan list repository yang "*unstable*", maka IBR-DTN yang akan dipasang masih belum stabil atau masih dalam tahap percobaan. Sebelum melakukan update, harus meminta gpg.key yang digunakan untuk memperbolehkan melakukan update ke repository tersebut. Lalu baru bisa dilakukan update pada sistem operasi.

Alur langkah isntalasi IBR-DTN bisa diliat pada lampiran 3. Proses instalasi IBR-DTN harus dilakukan sesuai dengan alur pada lampiran 3 karena akan mempengaruhi keberhasilan proses instalasi IBR-DTN. Untuk mensukseskan instalasi IBR-DTN, yang harus dilakukan adalah menginstal paket ibrdtnd dan ibrdtn-tools. Jika salah satu paket tidak terpasang, maka IBR-DTN tidak dapat digunakan.



Gambar 4.7 Konfigurasi IBR-DTN

Sumber: Implementasi

Konfigurasi yang akan digunakan adalah konfigurasi untuk menggunakan persistent storage atau memory RAM yang bisa diatur sesuai dengan kebutuhan. Sesuai dengan kebutuhan penelitian, maka penggunan persisten storage diperlukan karena memory RAM pada Raspberry tersebut hanya 512 Mb, sedangkan untuk mengirim *file* bisa saja mencapai *gigabyte*. Lalu penyetelan tcp socket yang digunakan yaitu menjadi "*any*" dan mematikan dht yang secara default terhubung ke internet.

Tahap selanjutnya yaitu memastikan IBR-DTN bisa berjalan pada infrastruktur ini dengan menggunakan terminal yang dapat dilihat pada gambar 4.8. Pada setiap ada hal yang dilakukan oleh IBR-DTN, maka akan ditampilkan

pada event log yang akan muncul setelah, mengaktifkan IBR-DTN seperti pada lampiran 2. Fungsi dari event log itu sendiri ada lah apa bila ada koneksi yang terhubung, ada device lain yang ter-*discovery*, lalu *file* tidak bisa terkirim, dan semua kejadian yang ada pada IBR-DTN dapat terlihat pada event log tersebut.

BAB V
PENGUJIAN DAN ANALISIS

Pada bab ini berisi hasil pengujian dan analisis terhadap sistem infrastruktur jaringan DTN yang telah diimplementasikan. Yaitu pengujian dan analisis untuk mengetahui apakah infrastruktur yang telah dirancang dan diimplementasikan telah bekerja sesuai dengan tujuan.

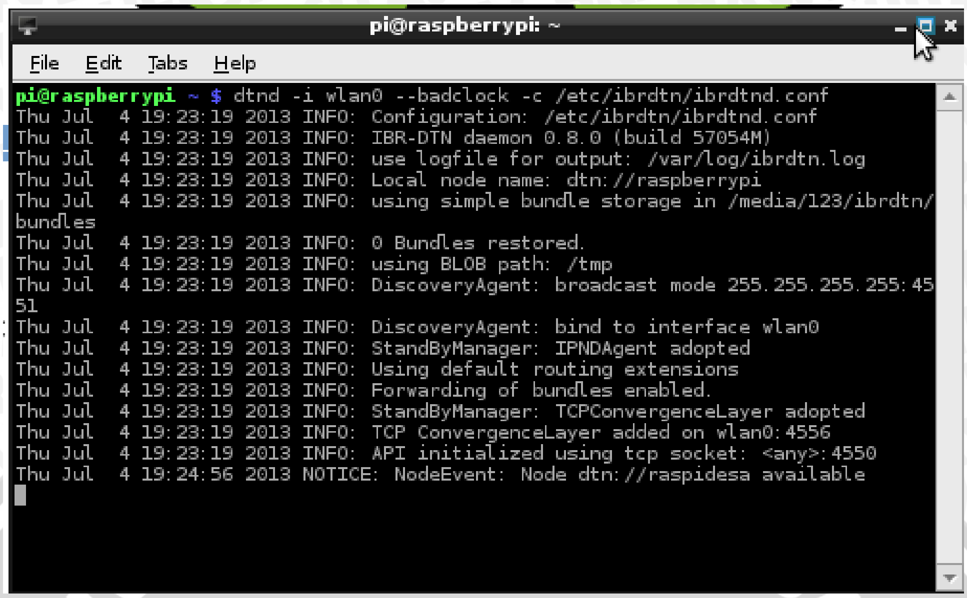
Proses pengujian dilakukan dengan tiga tahap, yaitu pengujian konetivitas, pengujian besar pengiriman *file*, dan pengujian pengiriman *file* dengan konektivitas terbatas. Pada pengujian konektivitas, dilakukan percobaan dengan cara mematikan daya listrik pada salah satu Raspberry Pi yang kemudian dinyalakan kembali serta mengaktifkan IBR-DTN pada Raspberry Pi tersebut. Proses pengujian selanjutnya yaitu dengan percobaan bersarnya *file* yang akan dikirim ke Raspberry Pi tujuan dengan maksud agar mengetahui maksimal besar *file* yang boleh kirim.

Pengujian terakhir yaitu pengiriman *file* dengan kondisi Raspberry Pi tujuan sempat mati sementara, dengan maksud agar pengiriman *file* bisa terus berlanjut atau tidak. Semua pengujian bisa dilakukan secara bersamaan, sehingga bisa mempersingkat waktu pengujian.

5.1 Pengujian Konektivitas

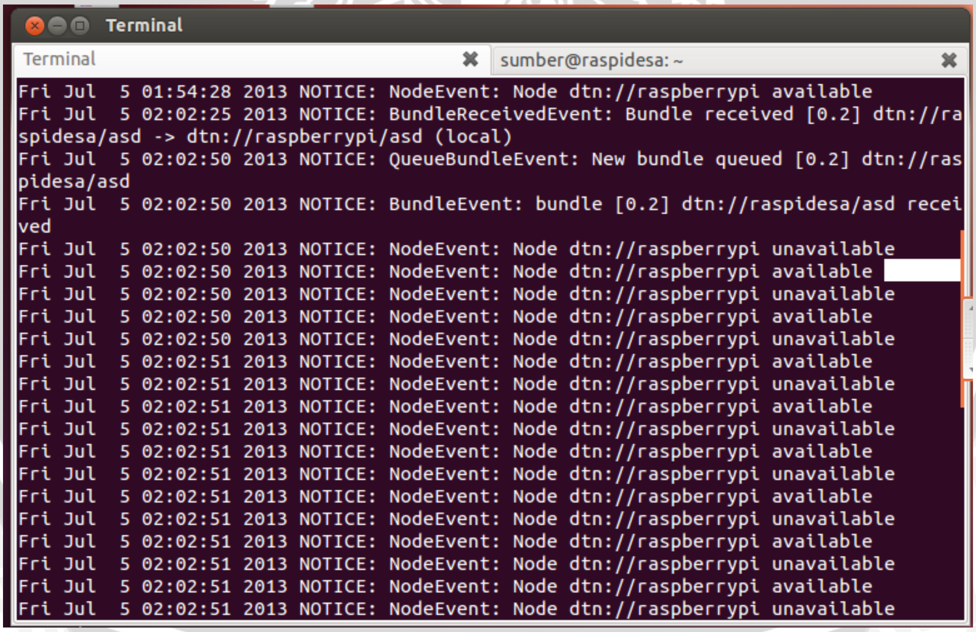
Pengujian dilakukan dengan cara mematikan konetivitas pada salah satu Raspberry Pi lalu dinyalakan kembali. Sebelumnya sudah harus dipastikan apakah setiap device sudah saling mengetahui jika DTN-nya sudah aktif.

Lalu setelah saling mengetahui, maka salah satu *device* akan dimatikan secara paksa yaitu dengan memutuskan daya listrik. Maka device yang masih hidup atau device pengirim data dapat mengetahui bahwa Raspberry Pi yang tadi terhubung menjadi mati.



Gambar 5.1 Log Event pada raspberrpi

Sumber: Pengujian dan Analisis



Gambar 5.2 Log event pada raspidesa

Sumber: Pengujian dan Analisis

Pada gambar 5.2, memberitahukan bahwa *storage* yang digunakan untuk media penyimpanan sementara bundle yaitu tidak menggunakan memory atau

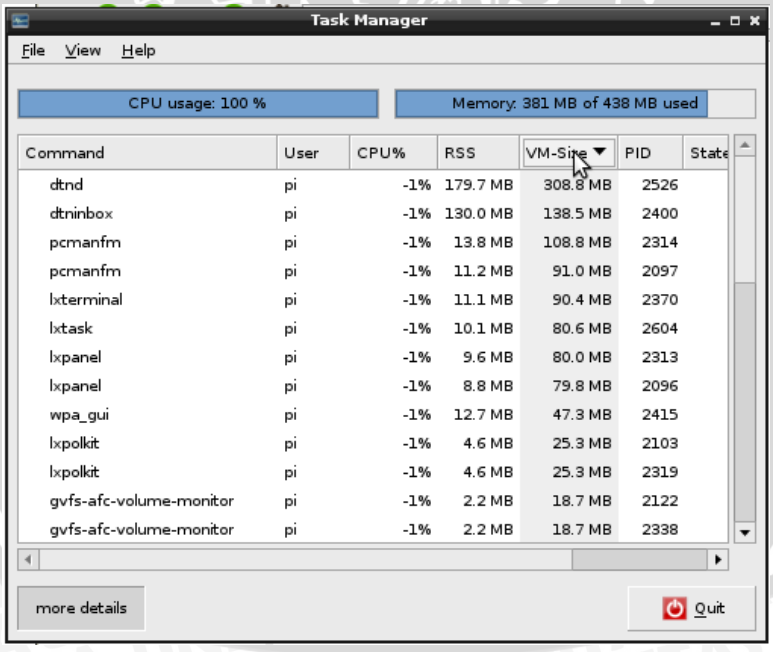
sudah bisa menggunakan *persistent storage* yang memang sudah disediakan oleh software IBR-DTN itu sendri dan bisa diatur dimana letak penyimpanannya.

5.2 Pengujian Besaran *File* yang dikirim

Pengujian ini dilakukan dengan menggunakan besarnya *file* yang akan dikirim dengan melihat penggunaan RAM pada Raspberry Pi tujuan pengiriman, baik dengan menggunakan RAM default maupun dengan penambahan swap *file* sebagai swap memory.

5.2.1 Pengiriman dengan memory RAM *default*

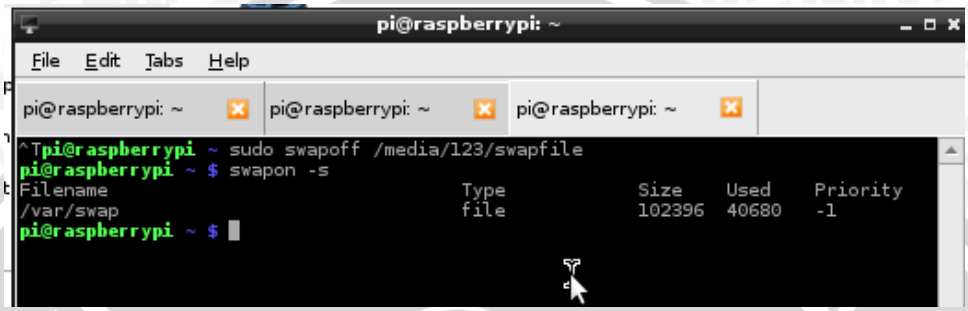
Memory RAM bawaan dari Raspberry Pi dengan penggunaan sistem operasi Debian Wheezy yaitu sebesar 438 MB dengan tambahan 100 MB dari *swap memory*. Dengan melihat memory sebesar itu, maka dilakukan pengiriman *file* sebesar 246 MB dan menggunakan mode memory only pada untuk penyimpanan *bundle*-nya.



Gambar 5.3 Kekurangan Memory

Sumber: Pengujian dan Analisis

Pada gambar 5.3, dapat diketahui bahwa maksimal pengiriman jika yang dipakai sebagai penyimpanan *bundle* adalah memory RAM default, maka yang terjadi seperti diatas, yaitu resource yang kurang. Sehingga, oleh perangkat lunak IBR-DTN tersebut langsung di hapus jika tidak sesuai dengan besarnya paket yang datang.



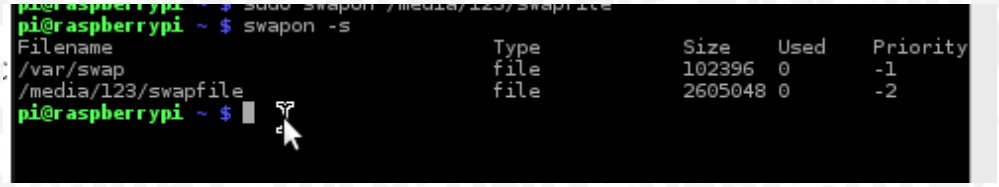
Gambar 5.4 Default Swap Memory pada Raspberry

Sumber: Pengujian dan Analisis

Gambar 5.4 dapat menjelaskan bahwa memory swap default yang ada pada Raspberry Pi dengan sistem operasi Debian Wheezy yaitu hanya sebesar 100MB dan itu pun tidak cukup untuk memberi tambahan *memory* apabila ada *file* besar yang akan dikirim. Supaya agar pengiriman dapat berjalan dengan lancar dan tidak melebihi kapasitas memory (RAM) default, maka besar *file* yang dikirim maksimal sebesar 220MB. Akan tetapi, penggunaan RAM tersebut belum termasuk dengan penambahan berbagai macam aplikasi pada infrastruktur ini. Sehingga apabila masih meggunakan memory RAM default, ada kemungkinan jika maksimal pengiriman pun akan berkurang sesuai dengan berarnya memory RAM yang tersedia.

5.2.2 Pengiriman dengan tambahan *Swap Memory*

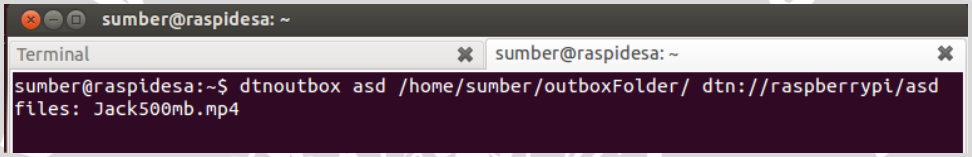
Tahap ini adalah melakukan pengujian dengan menggunakan *swap memory*. Swap memory ini berguna untuk menambahkan jumlah memory yang tersedia pada Raspberry Pi.



Gambar 5.5 Swap*file* sebagai memory tambahan pada raspberrypi

Sumber: Pengujian dan Analisis

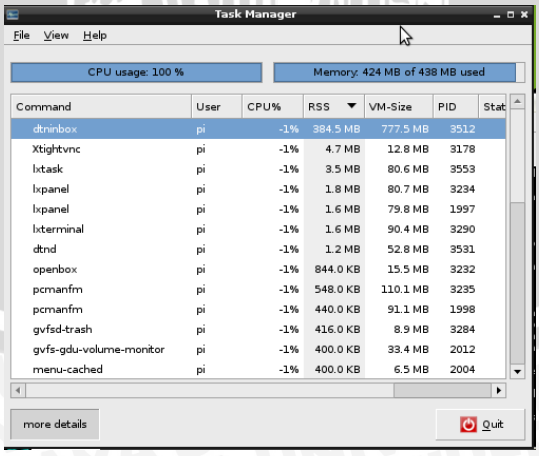
Swap Memory yang akan digunakan adalah sebesar 2GB yang sudah dibuat menjadi *file* pada /media/123/swap*file* (pada USB Storage). Percobaan selanjutnya adalah dengan melakukan pengiriman *file* diatas 500 MB.



Gambar 5.6 *File* sebesar 500mb dikirim ke raspberrypi

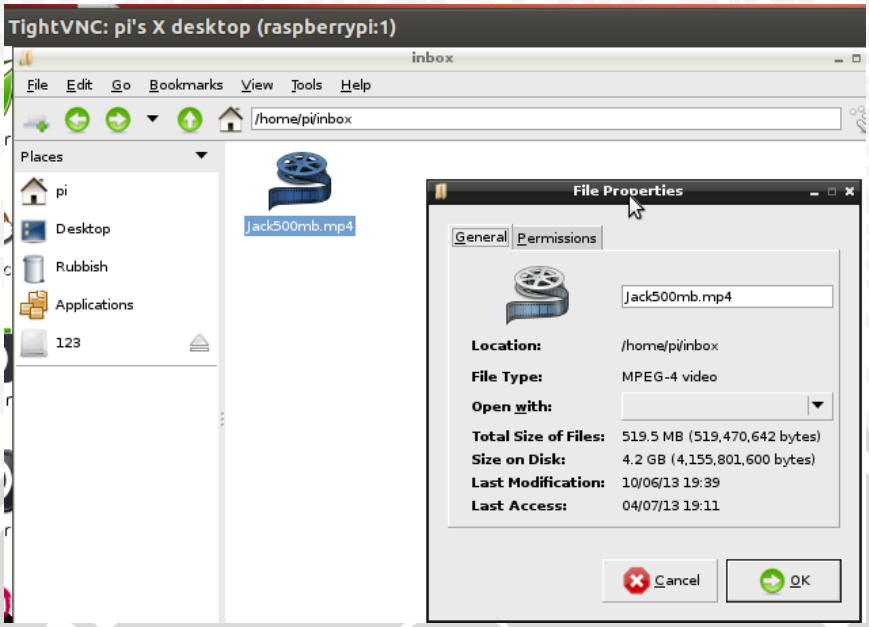
Sumber: Pengujian dan Analisis

Walaupun RAM yang ada pada raspberrypi hampir semuanya terpakai, akan tetapi pengekstrakan *file* masih tetap berlanjut karena sudah dibantu dengan adanya tambahan memory yang sudah dibuat pada *USB Drive* sebesar 2 GB.



Gambar 5.7 Penggunaan resource pada raspberrypi

Sumber: Pengujian dan Analisis



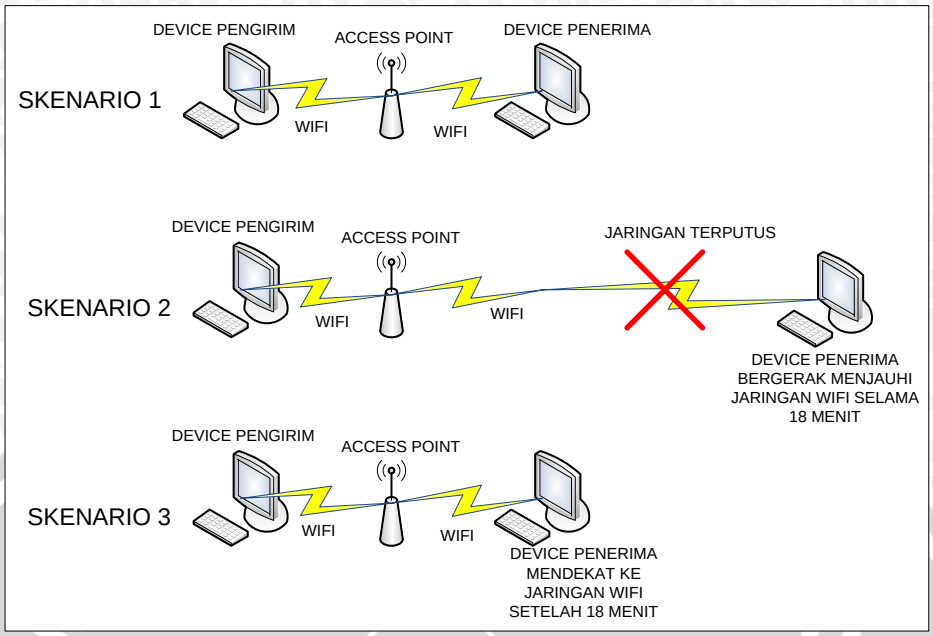
Gambar 5.8 *File* sebesar 500MB lebih sampai pada raspberrypi

Sumber: Pengujian dan Analisis

Dari gambar 5.8, dapat disimpulkan bahwa besar pengiriman *file* tergantung sekali pada memory RAM Raspberry Pi tujuan perngirman *file.* Apabila RAM yang digunakan pada infrastruktur tersebut mencapai 2 GB, maka pengiriman *file* bisa mencapai 2 GB. Sehingga, besar *file* yang dipertukarkan pada infrastruktur ini tergantung pada besarnya memory RAM yang digunakan.

5.3 Pengujian Pengiriman *File* dengan Koneksivitas Putus-Sambung

Pengujian terakhir yaitu pengujian pengiriman sebuah *file* dari *device* pengirim ke device penerima dalam kondisi konektivitas yang pustus-sambung. Untuk melakukan pengujian tersebut, dilakukan sesuai dengan skenario pada gambar 5.9. Pada saat melakukan pengiriman file, *device* penerima dijauhkan dari jangkauan *WiFi Access Point* selama 15 menit, sehingga koneksi antar kedua device tersebut terputus. Lalu didekatkan kembali agar dapat terhubung kembali dan dapat melajutkan pengiriman *file* yang sempat tertunda



Gambar 5.9 Skenario pengujian

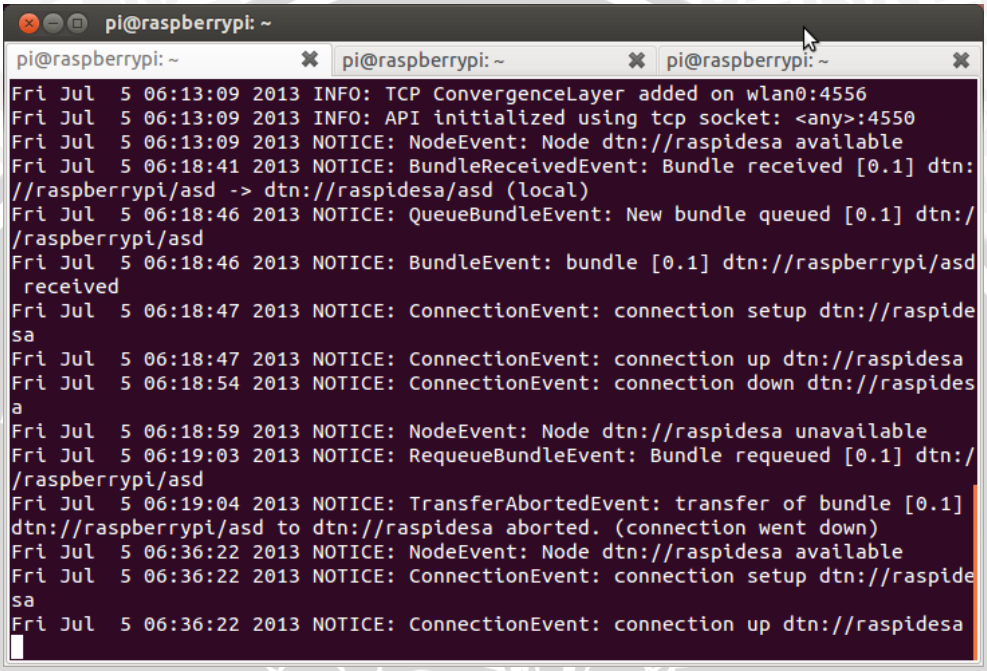
Sumber : Pengujian dan Analisa

Pada skenario 1, terdapat dua *device* (*device* pengirim dan *device* penerima) yang digunakan sebagai alat pengujian yang saling kerkoneksi satu sama lain menggunakan jaringan WiFi pada satu *Access Point* yang sama. Setiap device menjalankan *bundle protocol*, sehingga setiap device mengetahui bahwa ada device yang bisa berhubungan dengan menggunakan *bundle protocol* tersebut. Pada sesi ini, device pengirim mengirimkan sebuah data ke device penerima dalam keadaan jaringan masih normal pada menit-menit awal.

Pada skenario 2, pengujian selanjutnya yaitu dengan membawa device penerima untuk menjauhi jaringan WiFi Access Point yang membuat koneksi antar kedua device terputus dan pengiriman file menjadi tertunda. Sehingga file yang saat itu sedang dikirim untuk segera dibatalkan lalu masih tetap tersimpan pada bundle storage device pengirim dan menunggu untuk adanya sambungan kembali ke device penerima.

Pada skenario 3, pengujian yang terakhir yaitu dengan mendekatkan kembali device penerima ke jaringan WiFi Access Point yang terhubung ke

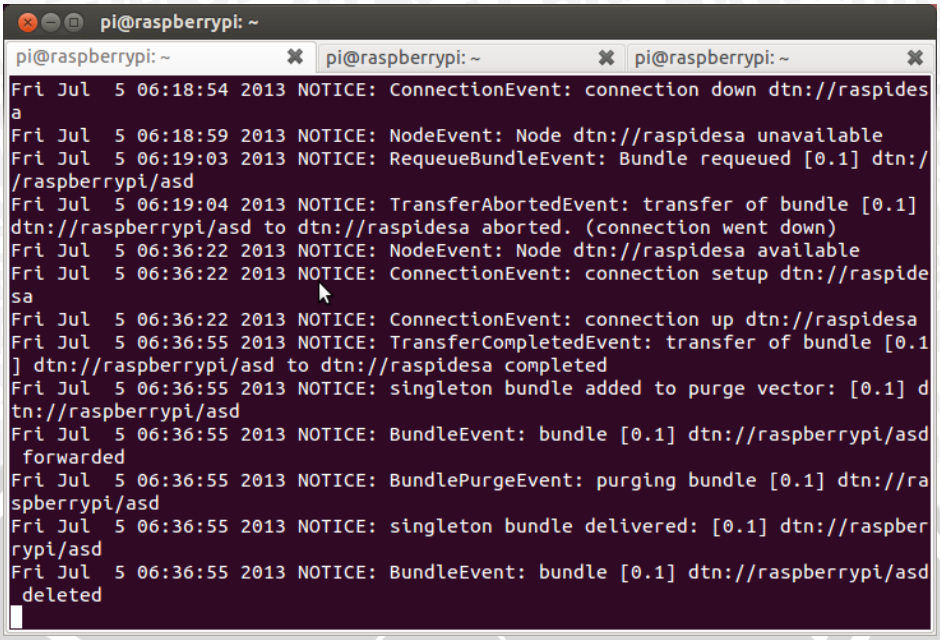
device pengirim. Lalu dilakukan pengamatan terhadap event log yang ditampilkan oleh bundle protocol. Secara otomatis, pengiriman file yang tadi tertunda, bisa dilanjutkan kembali tanpa harus mengirimkan kembali file yang sama ke device tujuan. Pada device tujuan file akan tersedia pada folder yang sudah dipersiapkan sebagai tempat menyimpan file yang datang.



Gambar 5.10 Log event pada raspberrypi

Sumber: Pengujian dan Analisis

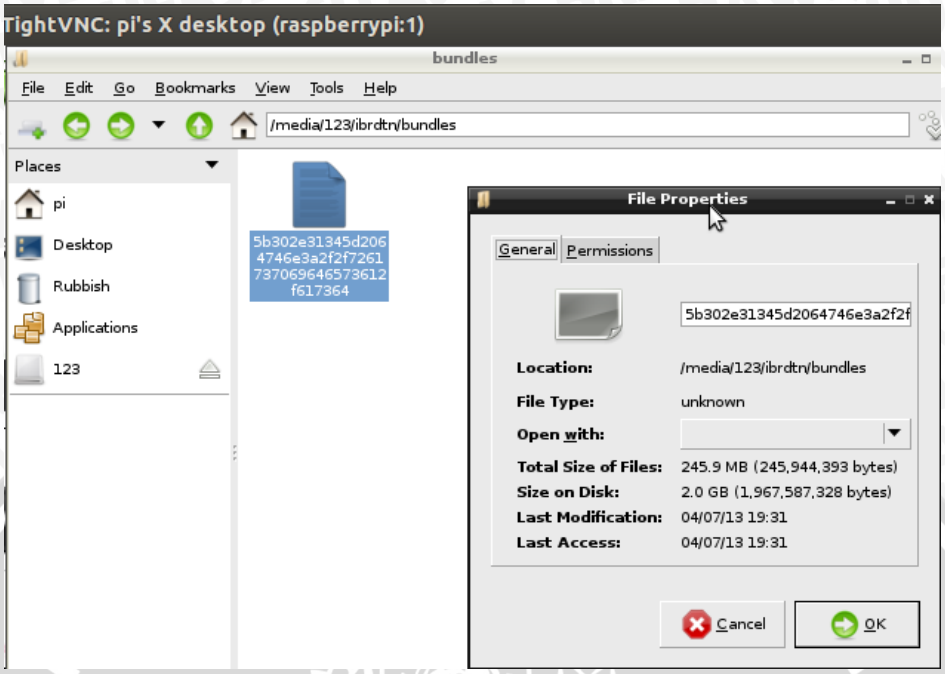
Pada gambar 5.10 diketahui bahwa device pengirim (raspberrypi) mengetahui bahwa ada device penerima (raspidesa) yang terhubung ke satu jaringan yang sama pada menit 13:09. Sesuai dengan skenario 1 pada gambar 5.9, dilakukan pengiriman sebuah file yang ditujukan kepada device penerima dengan dimasukkan kedalam foleder yang disediakan untuk mengirim file pada menit 18:41, yang selanjutnya file disimpan pada *bundle storage* pada menit 18:46. Setelah mengetahui bahwa file tersebut merupakan antrian yang pertama, maka file tersebut segera dikirim ke device penerima pada menit 18:47.



Gambar 5.11 Event Log pada raspberrypi

Sumber: Pengujian dan Analisis

Pada gambar 5.11 menjelaskan bahwa saat pengiriman *file* ke raspberrypi (*device* penerima) terputus pada menit 18:54 sesuai dengan skenario yang kedua pada gambar 5.9, lalu *bundle protocol* mengetahui bahwa koneksi ke *device* penerima sedang *down* maka pengiriman dibatalkan dan akan disimpan ke antrian dalam *bundle storage* dengan *lifetime* tertentu. Setelah 18 menit kemudiaan saat koneksi ke *device* penerima kembali terhubung pada menit 36:22 sesuai dengan skenario ketiga pada gambar 5.9, maka *bundle protocol* akan secara otomatis kembali mengirimkan *file* yang sempat dibatalkan akibat putusnya sambungan. Pada menit 36:55, disebutkan bahwa *file* sudah sampai pada *device* penerima, sehingga *bundle* yang masih ada pada *bundle storage* segera dihapus.



Gambar 5.12 *File Bundle* yang disimpan pada *USB Drive*

Sumber: Pengujian dan Analisis

File yang akan dikirim ke *device* penerima dimasukkan ke dalam *bundle storage* yang pada awal konfigurasi disetel menjadi eksternal *storage* yang terlihat pada gambar 5.11. Setiap *file* yang ditersimpan pada *bundle storage*, memiliki lifetime tertentu pada konfigurasi saat awal pengiriman *file*. Apabila lifetime *file* tersebut sudah habis, maka pada saat itu juga bundle akan dihapus. Sehingga dapat diketahui bahwa infrastruktur ini dapat melakukan pertukaran informasi, meskipun saat pengiriman *file* sempat terjadi putusnya komunikasi antar device. Lamanya *file* agar tidak dihapus oleh *bundle protocol* dalam menunggu koneksi kembali terjadi, tergantung pada saat penyetelan manual konfigurasi pada saat awal pengiriman *file* yang bisa dilihat pada lampiran 4 mengenai konfigurasi dari IBR-DTN.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perangcangan, implementasi, dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Infrastruktur dapat melakukan pertukaran informasi pada konektivitas yang terbatas/putus-sambung.

2. Infrastruktur ini dapat melanjutkan pengiriman *file* secara otomatis pada saat koneksi terputus secara tiba-tiba ketika sedang melakukan pengiriman sebuah *file*. Apabila terputus secara total, maka infrastruktur tersebut bisa bergerak secara mobile (bisa dibawa oleh orang, kendaraan darat, kendaraan laut, ataupun kendaraan udara) untuk mendekat ke Access Point agar dapat terhubung kembali ke koneksi yang sempat terputus lalu melanjutkan pertukaran informasi yang sempat tertunda.

3. Lifetime pada antrian *bundle* dapat diatur secara manual, sehingga dapat diatur memiliki waktu yang terbatas, maka pada waktu tertentu antrian bundle akan terhapus ketika sambungan tidak ada ke device tujuan terus menerus. Jika, diatur tak terhingga maka antrian bundle tidak ada yang akan terhapus sampai kapanpun.

4. Dengan menggunankan RAM default (512MB + 100 MB swap memory) dan penyimpanan *bundle* pada *memory, file* yang bisa terkirim hanya sekitar 220 MB. Sedangkan dengan menggunakan memory swap tambahan yang bisa diatur sesuai keinginan, sehingga dapat melakukan pengiriman *file* sebesar *memory* RAM yang digunakan pada infrastruktur ini.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan infrastruktur ini antara lain:

1. Komponen-komponen perangkat keras dan perankat lunak yang digunakan dalam membangun infrastruktur agar tidak terjadi permasalah pada saat pembangunan maupung pengembangan infrastruktur.
2. Untuk pengembangan lebih lanjut, infrastruktur ini dapat dimodifikasi dan dikembangkan dengan merubah isi dari perangkat lunak agar bisa digunakan lebih mudah bagi pengguna.
3. Untuk pengembangan lebih lanjut, penggunaan USB Drive bisa diganti dengan SD Card yang mempunyai kapasitas besar dan kecepatan tinggi ataupun dengan menggunakan Hard Drive external yang memiliki kapasitas besar dan kecepatan tinggi.
4. Penggunaan RAM pada Raspberry Pi untuk menjalankan IBR-DTN, sebaiknya menggunakan memory swap tambahan untuk dapat bekerja lebih baik dalam melakukan pertukaran informasi.

DAFTAR PUSTAKA

[CER-07] Cerv, V., Burleigh, S., Hooke, A., Torgerson, L., Durst. R., Scott, K., Fall, K. dan Weiss, H. 2007, "*Delay Tolerant Networking Achitecture*". Diakses melalui http://tools.ietf.org/html/rfc4838. Tanggal akses: 24-06-2013

[DEB-13] Debian Project, 2013. Diakses melalui http://www.debian.org/intro/about. Tanggal akses: 25-06-2013

[FAL-03] Fall Kevin. 2003, "*A Delay-Tolerant Network Architecture For Challenged Internets*", *Intel Research*, Berkeley.

[HUS-11] Husni, Emir. 2011, "Delay Tolerant Network based internet services for remote areas using train systems," *17th IEEE International Conference on Networks*, hal. 47-52

[KEM-13] Kemdiknas. 2013, "*Kamus Besar Bahasa Indonesia*". Diakses melalui http://bahasa.kemdiknas.go.id/kbbi/index.php. Tanggal akses: 25-06-2013

[RAS-13] Raspberry PI Foundation, 2013. Diakses melalui http://www.raspberrypi.org/faqs. Tanggal akses: 25-06-2013

[SAN-11] Santosa Iwan, Armein Z. R. Langi, Yoanes Bandung. 2011, "*Perancangan Web Base Instant Messenger to SMS Gataway Untuk Mendukung Komunikasi Pada Jaringan Pedesaan*", PPTIK, ITB, 2011

[SCH-11] Schildt, S., Morgenroth, J., Pöttner, Wolf-Bastian dan Wolf, L. 2011, "*IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation*", *Electronic Communicatinos of the EASST*, Vol.37, hal.1-11.

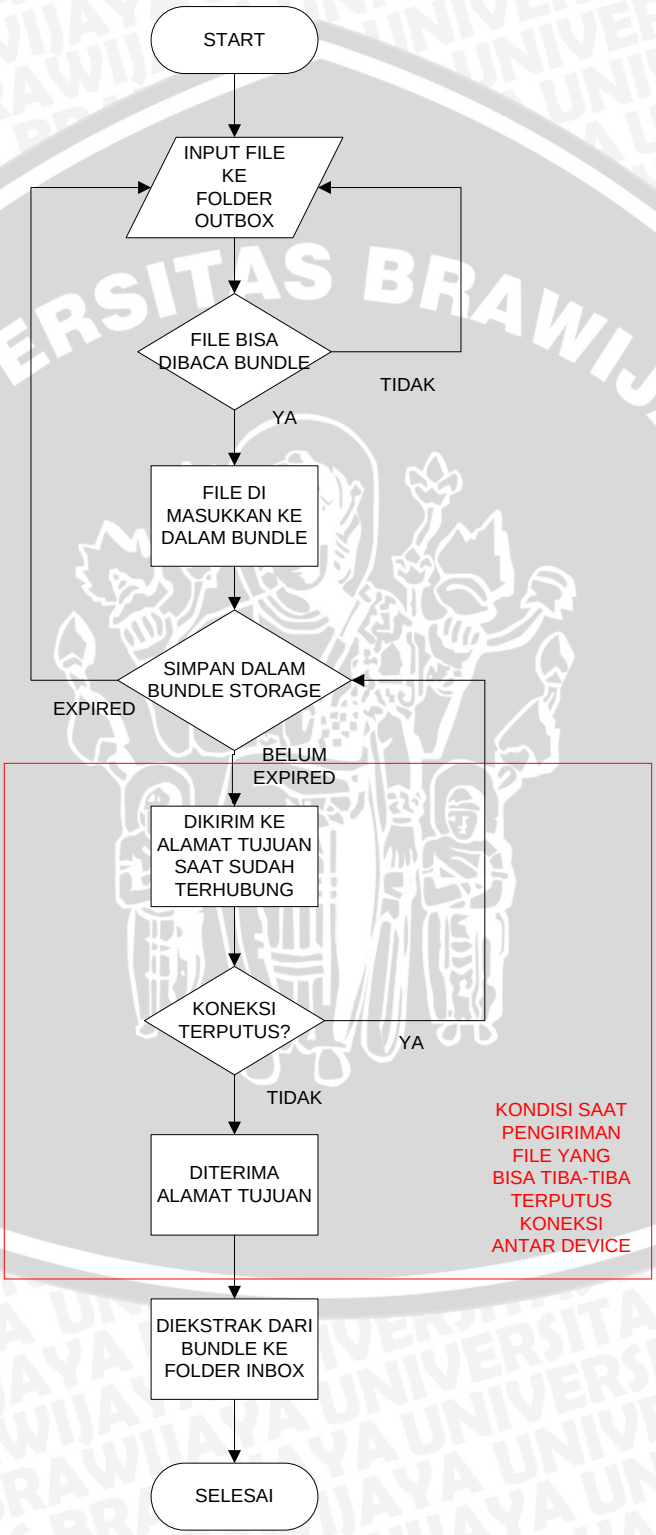
[SCO-07] Scott, K. dan Burleigh, K. 2007, "*Bundle Protocol Specification*". Diakses melalui http://tools.ietf.org/html/rfc5050. tanggal akses: 24-06-2013

[SDOE-08] Doering, M., Lahde, S., Morgenroth, J. dan Wolf, L. 2008, "*IBR-DTN: an efficient implementation for embedded systems*", *Proceeding of the Third Workshop on Challenged Networks*, CHANTS, San Francisco, California, USA, hal. 117-120.

[THE-12] TheMagPI. 2012, "*issue 01*". Diakses melalui http://www.themagpi.com. Tanggal akses: 24-06-2013

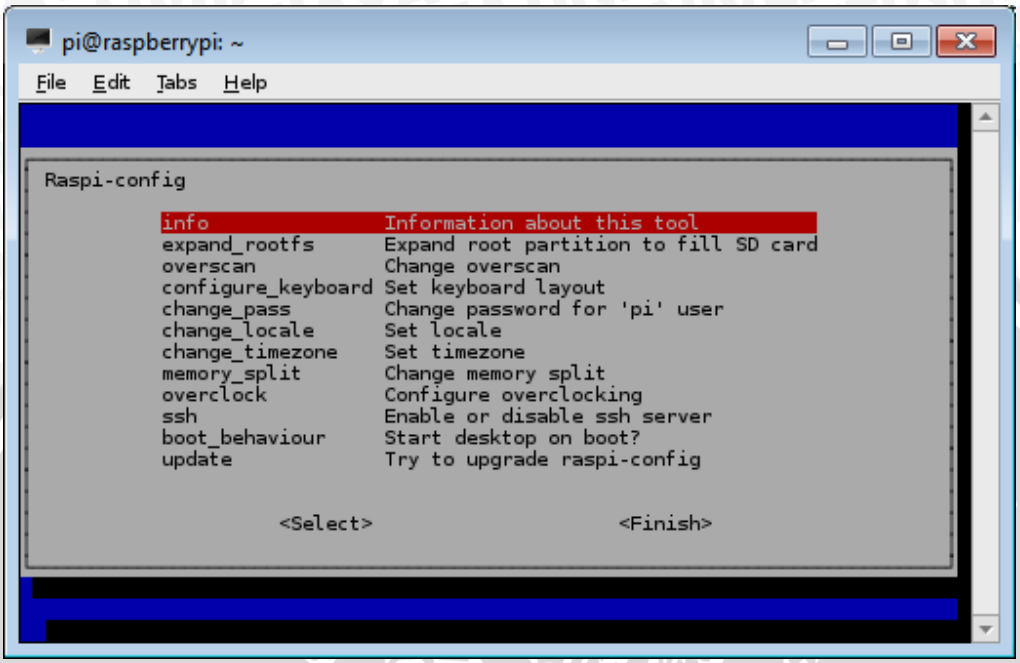
LAMPIRAN – LAMPIRAN

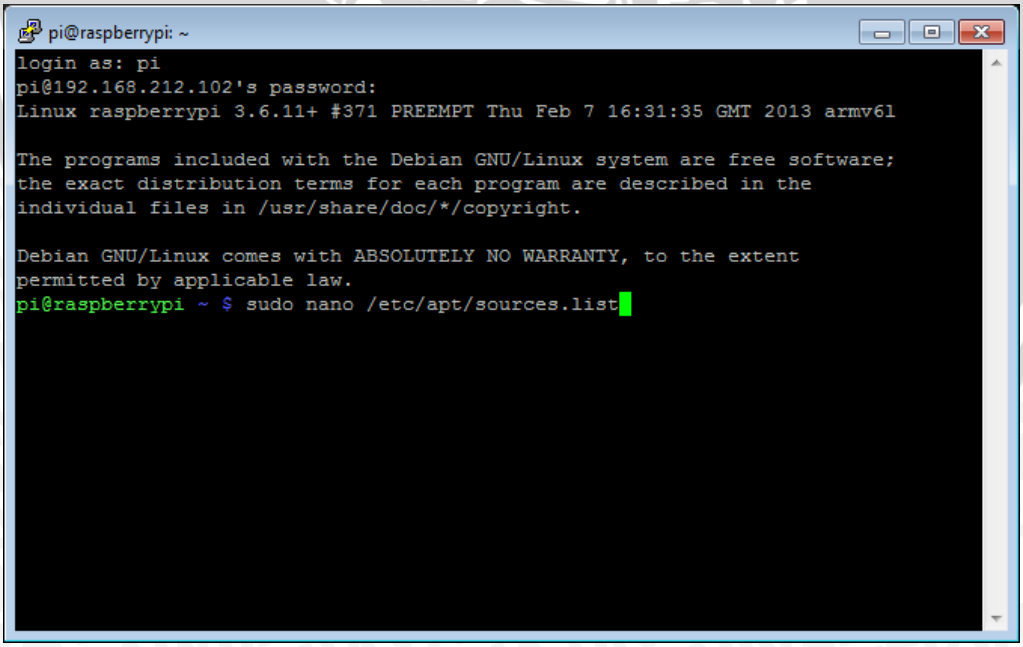
Lampiran 1. Diagram Alur penggunaan sistem

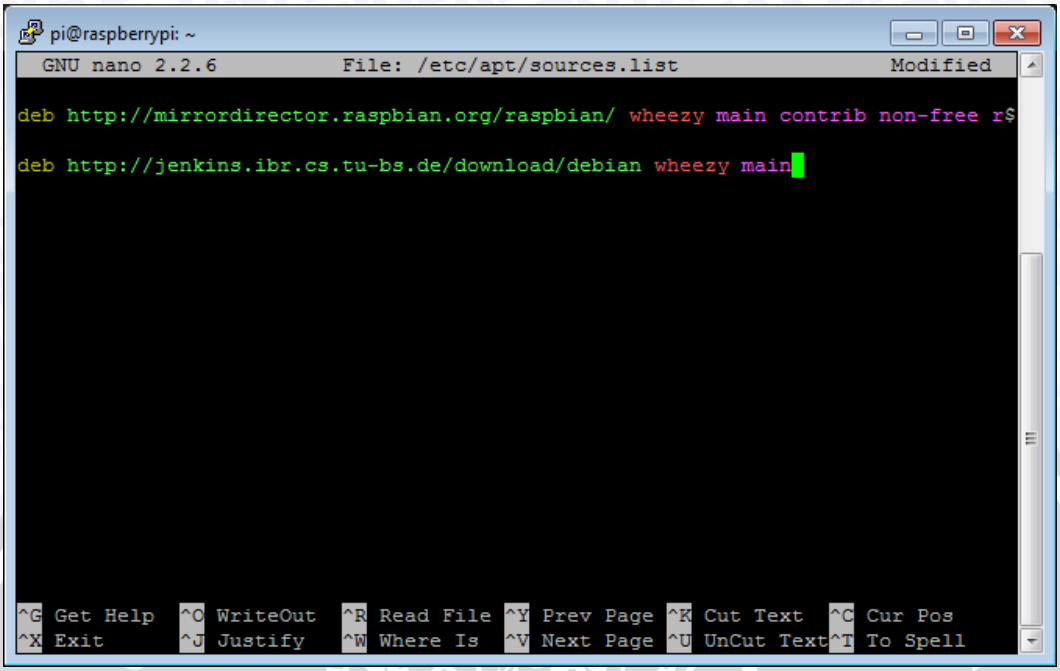


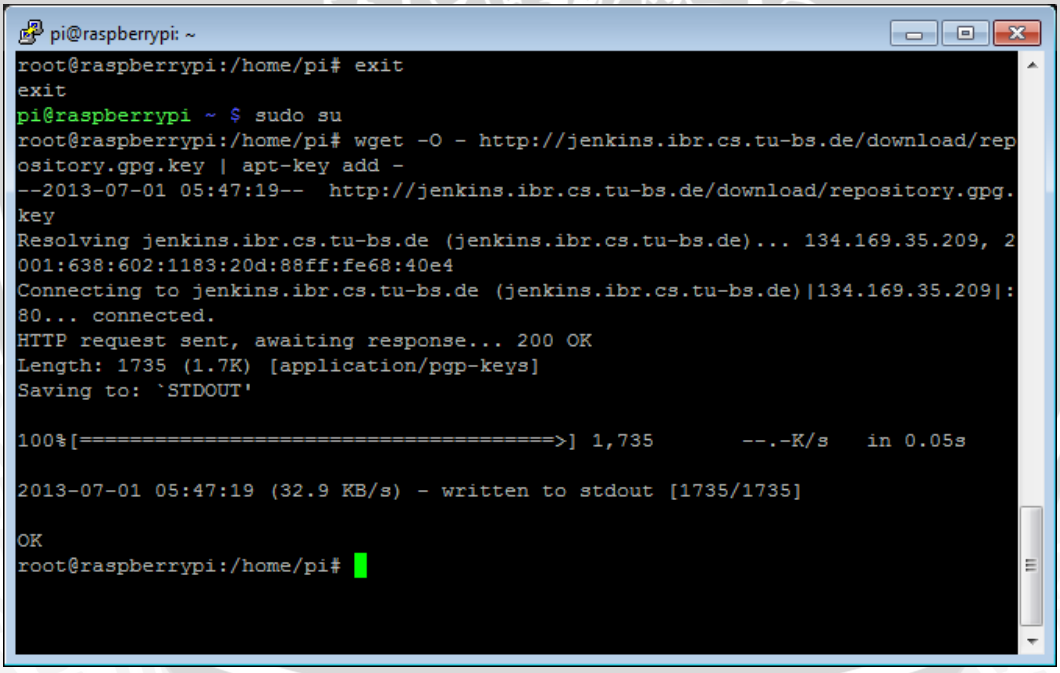
Lampiran 2. Instalasi Infrastruktur

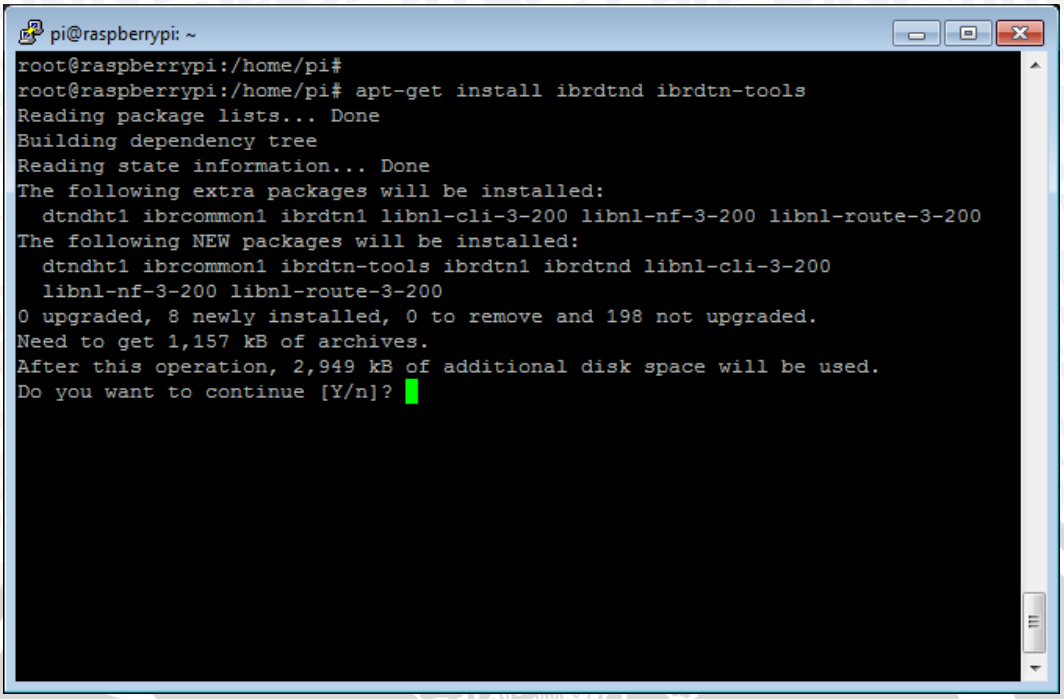
Pada Lampiran ini menampilkan gambar instalasi infrastruktur dimulai dari instalasi sistem operasi sampai semuanya terpasang

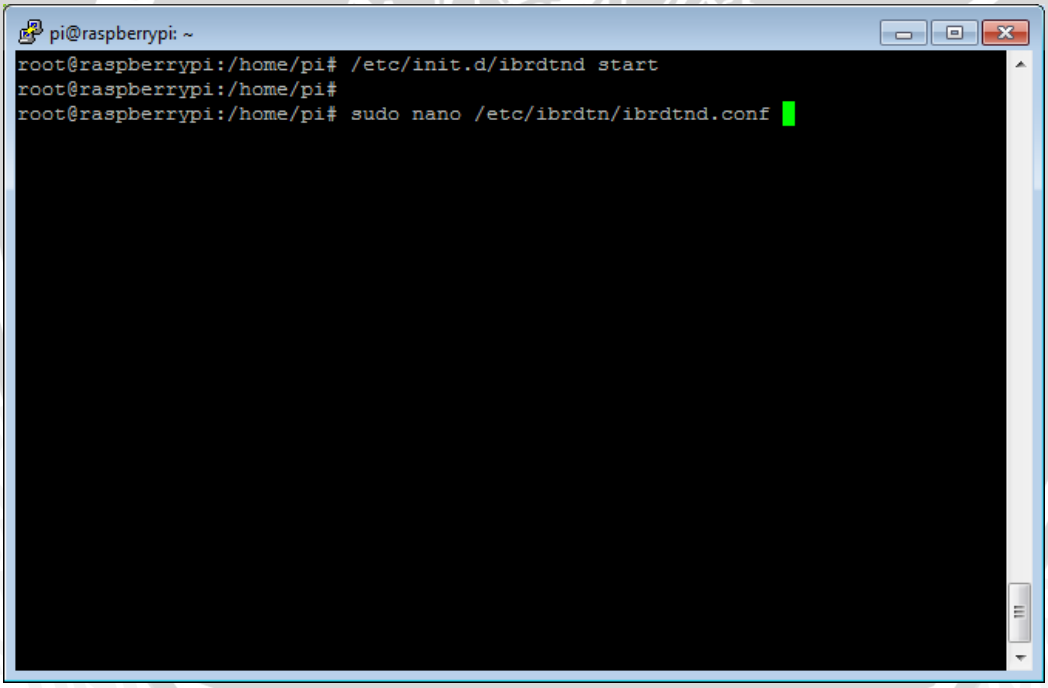


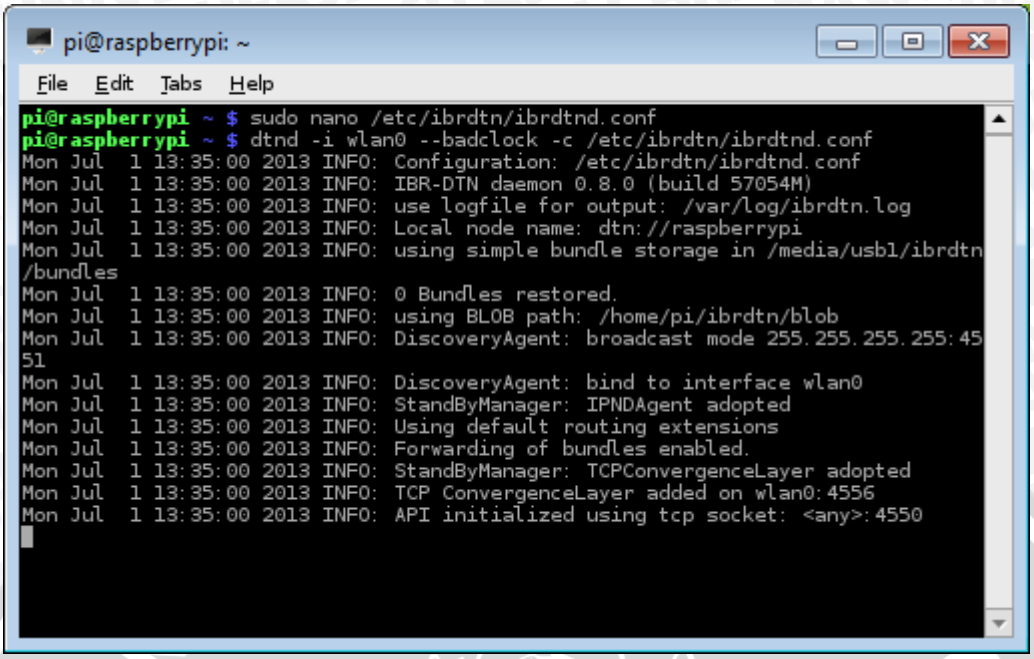


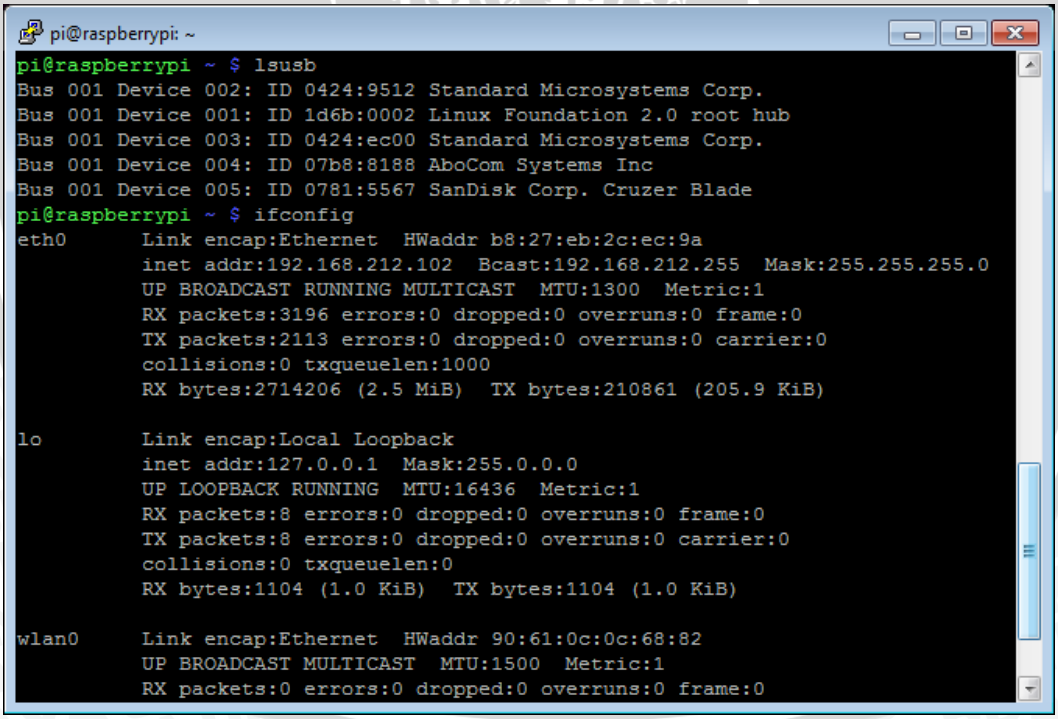

pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/apt/sources.list Modified
deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free r$
deb http://jenkins.ibr.cs.tu-bs.de/download/debian wheezy main
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

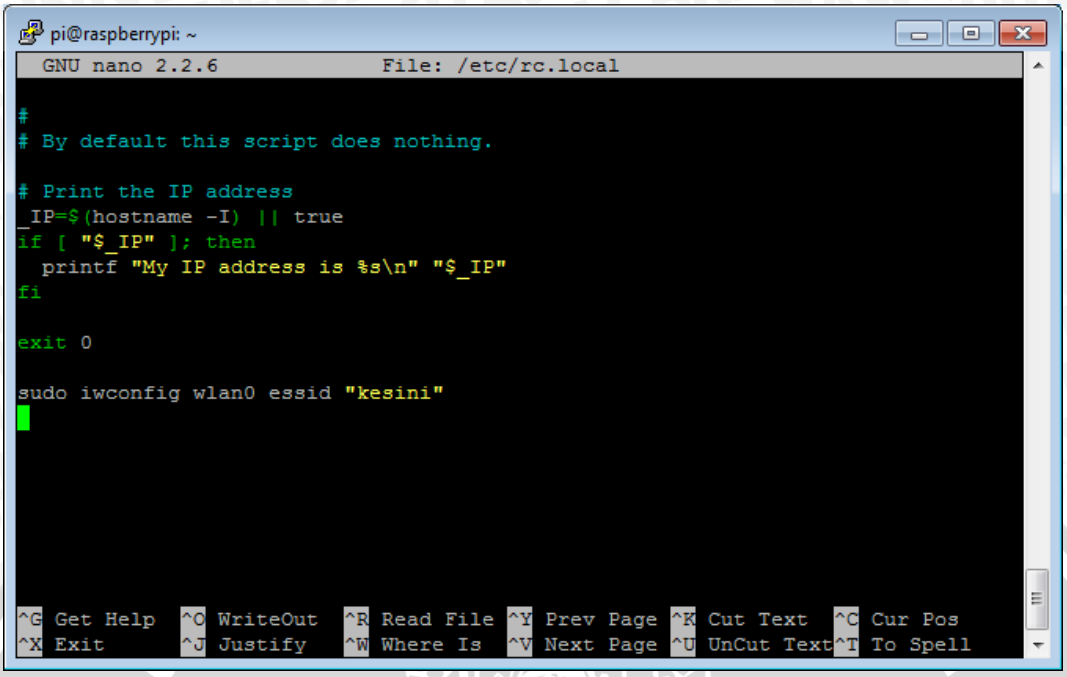

pi@raspberrypi: ~
root@raspberrypi:/home/pi# exit
exit
pi@raspberrypi ~ $ sudo su
root@raspberrypi:/home/pi# wget -O - http://jenkins.ibr.cs.tu-bs.de/download/rep
ository.gpg.key | apt-key add -
--2013-07-01 05:47:19-- http://jenkins.ibr.cs.tu-bs.de/download/repository.gpg.
key
Resolving jenkins.ibr.cs.tu-bs.de (jenkins.ibr.cs.tu-bs.de)... 134.169.35.209, 2
001:638:602:1183:20d:88ff:fe68:40e4
Connecting to jenkins.ibr.cs.tu-bs.de (jenkins.ibr.cs.tu-bs.de)|134.169.35.209|:
80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1735 (1.7K) [application/pgp-keys]
Saving to: `STDOUT'
100%[======================================>] 1,735 --.-K/s in 0.05s
2013-07-01 05:47:19 (32.9 KB/s) - written to stdout [1735/1735]
OK
root@raspberrypi:/home/pi#

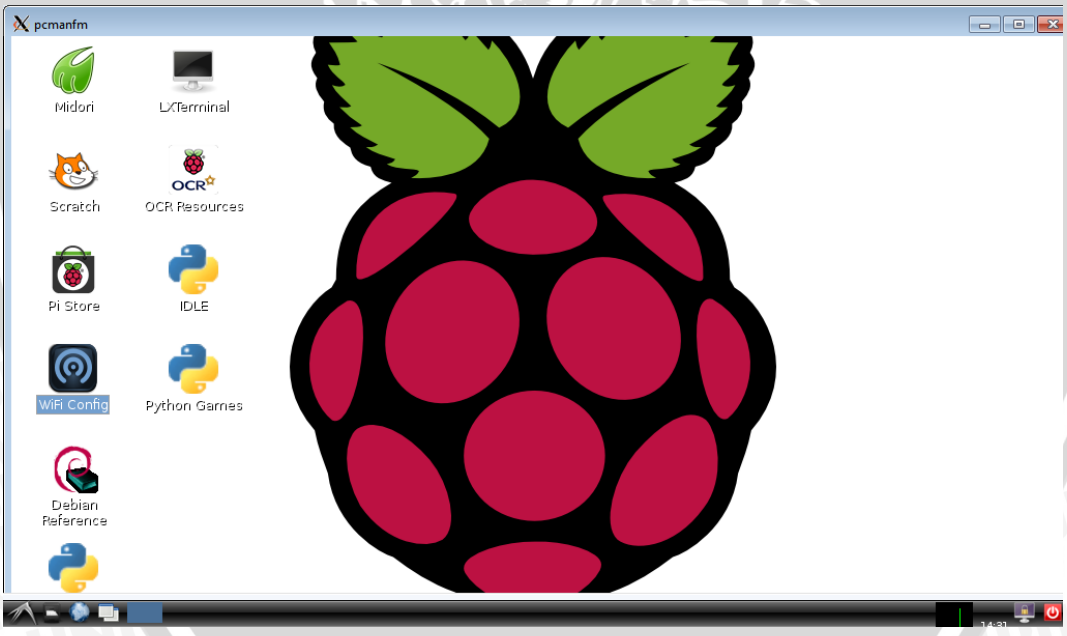

pi@raspberrypi: ~
root@raspberrypi:/home/pi#
root@raspberrypi:/home/pi# apt-get install ibrdtnd ibrdtn-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
dtndht1 ibrcommon1 ibrdtn1 libnl-cli-3-200 libnl-nf-3-200 libnl-route-3-200
The following NEW packages will be installed:
dtndht1 ibrcommon1 ibrdtn-tools ibrdtn1 ibrdtnd libnl-cli-3-200
libnl-nf-3-200 libnl-route-3-200
0 upgraded, 8 newly installed, 0 to remove and 198 not upgraded.
Need to get 1,157 kB of archives.
After this operation, 2,949 kB of additional disk space will be used.
Do you want to continue [Y/n]?

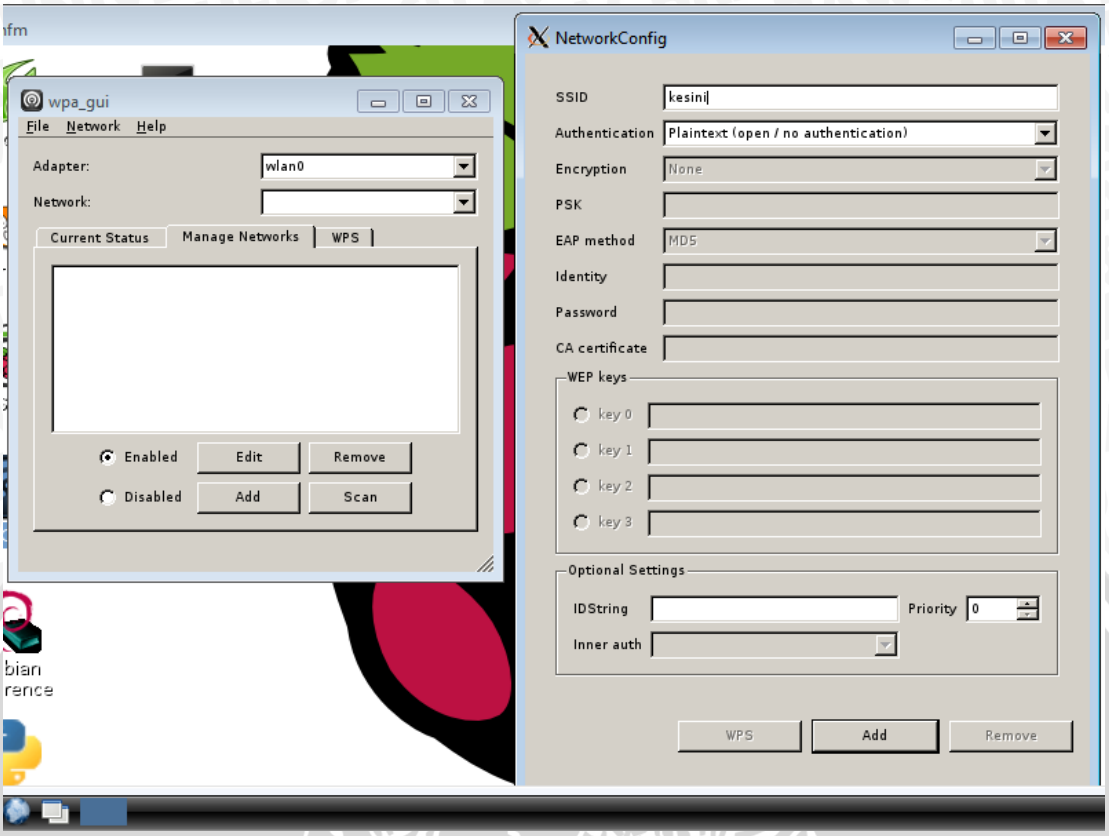

pi@raspberrypi: ~
root@raspberrypi:/home/pi# /etc/init.d/ibrdtnd start
root@raspberrypi:/home/pi#
root@raspberrypi:/home/pi# sudo nano /etc/ibrdtn/ibrdtnd.conf

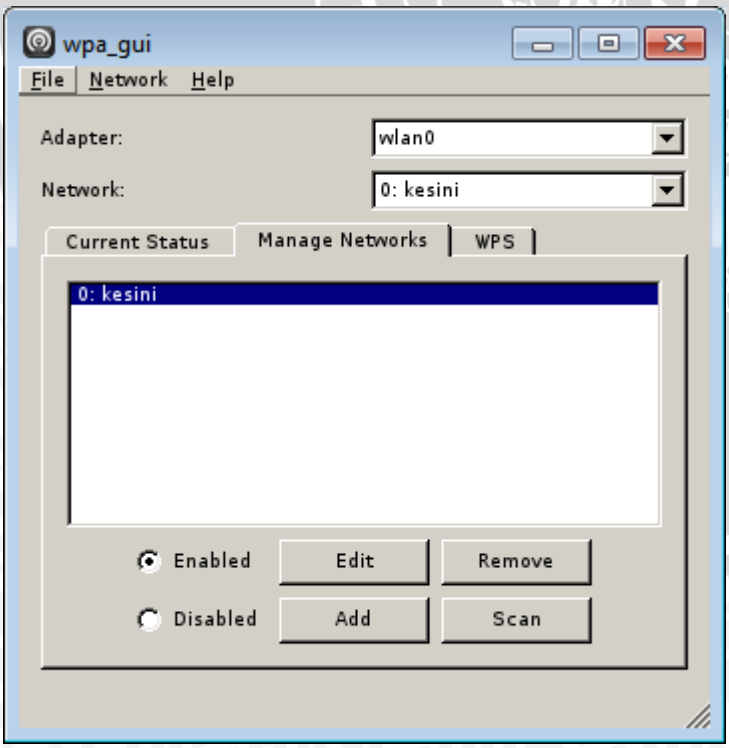

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi ~ $ sudo nano /etc/ibrdtn/ibrdtnd.conf
pi@raspberrypi ~ $ dtnd -i wlan0 --badclock -c /etc/ibrdtn/ibrdtnd.conf
Mon Jul 1 13:35:00 2013 INFO: Configuration: /etc/ibrdtn/ibrdtnd.conf
Mon Jul 1 13:35:00 2013 INFO: IBR-DTN daemon 0.8.0 (build 57054M)
Mon Jul 1 13:35:00 2013 INFO: use logfile for output: /var/log/ibrdtn.log
Mon Jul 1 13:35:00 2013 INFO: Local node name: dtn://raspberrypi
Mon Jul 1 13:35:00 2013 INFO: using simple bundle storage in /media/usb1/ibrdtn/bundles
Mon Jul 1 13:35:00 2013 INFO: 0 Bundles restored.
Mon Jul 1 13:35:00 2013 INFO: using BLOB path: /home/pi/ibrdtn/blob
Mon Jul 1 13:35:00 2013 INFO: DiscoveryAgent: broadcast mode 255.255.255.255:4551
Mon Jul 1 13:35:00 2013 INFO: DiscoveryAgent: bind to interface wlan0
Mon Jul 1 13:35:00 2013 INFO: StandByManager: IPNDAgent adopted
Mon Jul 1 13:35:00 2013 INFO: Using default routing extensions
Mon Jul 1 13:35:00 2013 INFO: Forwarding of bundles enabled.
Mon Jul 1 13:35:00 2013 INFO: StandByManager: TCPConvergenceLayer adopted
Mon Jul 1 13:35:00 2013 INFO: TCP ConvergenceLayer added on wlan0:4556
Mon Jul 1 13:35:00 2013 INFO: API initialized using tcp socket: <any>:4550


pi@raspberrypi: ~
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 07b8:8188 AboCom Systems Inc
Bus 001 Device 005: ID 0781:5567 SanDisk Corp. Cruzer Blade
pi@raspberrypi ~ $ ifconfig
eth0 Link encap:Ethernet HWaddr b8:27:eb:2c:ec:9a
inet addr:192.168.212.102 Bcast:192.168.212.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1300 Metric:1
RX packets:3196 errors:0 dropped:0 overruns:0 frame:0
TX packets:2113 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:2714206 (2.5 MiB) TX bytes:210861 (205.9 KiB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1104 (1.0 KiB) TX bytes:1104 (1.0 KiB)
wlan0 Link encap:Ethernet HWaddr 90:61:0c:0c:68:82
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0

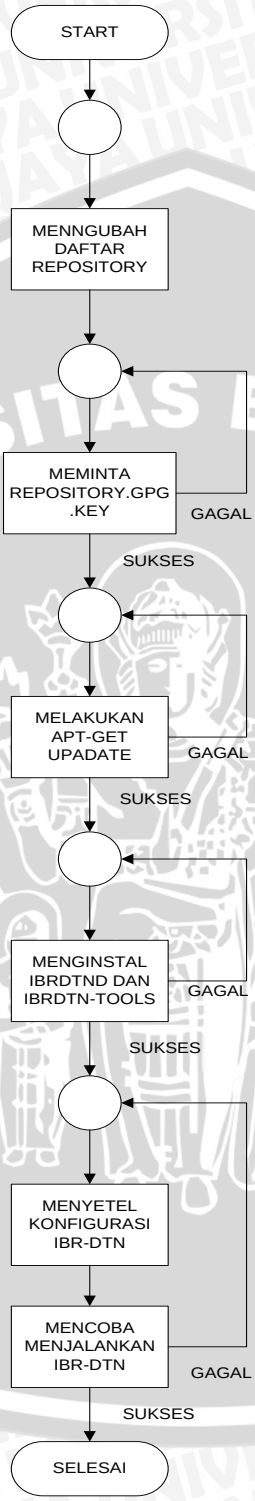

pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/rc.local
#
# By default this script does nothing.
# Print the IP address
_IP=$(hostname -I) || true
if ["$_IP"]; then
printf "My IP address is %s\n" "$_IP"
fi
exit 0
sudo iwconfig wlan0 essid "kesini"
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell


pcmanfm
Midori
LXTerminal
Scratch
OCR Resources
Pi Store
IDLE
WiFi Config
Python Games
Debian Reference


wpa_gui
File Network Help
Adapter: wlan0
Network:
Current Status Manage Networks WPS
Enabled Edit Remove
Disabled Add Scan
NetworkConfig
SSID kesini
Authentication Plaintext (open / no authentication)
Encryption None
PSK
EAP method MD5
Identity
Password
CA certificate
WEP keys
key 0
key 1
key 2
key 3
Optional Settings
IDString
Priority 0
Inner auth
WPS Add Remove


wpa_gui
File Network Help
Adapter: wlan0
Network: 0: kesini
Current Status Manage Networks WPS
0: kesini
Enabled Edit Remove
Disabled Add Scan

Lampiran 3. Diagram alir instalasi IBR-DTN



Lampiran 4. Konfigurasi IBR-DTN

```
#####################################
# IBR-DTN daemon                    #
#####################################

#

# the local eid of the dtn node

# default is the hostname

#

#local_uri = dtn://node.dtn

#

# specifies an additional logfile

#

logfile = /var/log/ibrdtn.log

#

# timezone offset in hours

#

#timezone = +7



#

# Limit the block size of all bundles

# The value accepts different multipliers.

# G = 1,000,000,000 bytes

# M = 1,000,000 bytes

# K = 1,000 bytes

#
```

```
limit_blocksize = 1.5G

#

# Limit the offset of predated timestamps to a max
value.

# Bundles with an invalid timestamp will be rejected.

#

limit_predated_timestamp = 604800

#

# Limit the max. lifetime of a bundle-

# Bundles with a lifetime greater than this value will
be rejected.

#

limit_lifetime = 604800

# set the user for the daemon

user = dtnd

# bind API to a named socket instead of an interface

#api_socket = /tmp/ibrdtn.sock

# define the interface for the API, choose any to bind
on all interfaces

api_interface = any

# define the port for the API to bind on

#api_port = 4550

#

# enable fragmentation support

# (default is disabled)

#
```

```
fragmentation = yes

#####################################

# storage configuration             #

#####################################

#

# define a folder for temporary storage of bundles

# if this is not defined bundles will processed in
memory
#
blob_path = /tmp/blob/
#

# define a folder for persistent storage of bundles
# if this is not defined bundles will stored in memory
only
#
storage_path = /media/123/ibrdtn/bundles/
#

# defines the storage module to use

# default is "simple" using memory or disk (depending
on storage_path)

# storage strategy. if compiled with sqlite support,
you could change

# this to sqlite to use a sql database for bundles.
#
storage = default

#

# Limit the size of the storage.

# The value accepts different multipliers.

# G = 1,000,000,000 bytes

# M = 1,000,000 bytes
```

```
# K = 1,000 bytes

#

limit_storage = 2.5G


#####################################

# statistic logging                 #

#####################################

#

# types: stdout | syslog | plain | csv | stat | udp

#

# statistic_type = stat

# statistic_interval = 2

# statistic_file = /tmp/ibrdtn.stats

# statistic_address = 127.0.0.1

# statistic_port = 1234


#####################################

# convergence layer configuration   #

#####################################
#

# discovery over UDP/IP

#

# You can specify an address to listen to for discovery
announcements.

# Multicast addresses are supported. If no address is
specified the default

# broadcast address is used.
```

```
#

#discovery_address = 224.0.0.1

# Specify the timeout. If no further discovery
announcement is received

# for x seconds, the daemon assume that the node has
went away.

#discovery_timeout = 5

# use short IPND beacons

#discovery_short = 0

# specify the discovery mechanism to use

# 0 = DTN2 compatible discovery

# 1 = IPND version 0

# 2 = IPND version 1 (default)

#discovery_version = 2

# To disable discovery announcements, set this option
to zero.

# (default is 1)

#

#discovery_announce = 1

#

# a list (seperated by spaces) of names for convergence
layer instances.

#

#net_interfaces = lan0

#

# dynamic rebind to network interfaces
```

```
# If this feature is enabled, then a socket will be
rebound on the

# network interface if the address has changed or the
interface is

# temporarily down.

#

#net_rebind = yes

#

# Try to connect to other nodes each x seconds.

# This option keeps connections up all the time.
#

#net_autoconnect = 60

#

# configuration for a convergence layer named lan0

#

#net_lan0_type = tcp                    # we want
to use TCP as protocol

#net_lan0_interface = eth0              # listen on
interface eth0

#net_lan0_port = 4556                   # with port
4556 (default)

#net_lan0_discovery = yes               # enable
discovery (default)

#

# configuration for a convergence layer named lan1

#

#net_lan1_type = udp                    # we want to
use UDP as protocol
```

```
#net_lan1_interface = eth0                  # listen on
interface eth0

#net_lan1_port = 4556                       # with port
4556 (default)

#net_lan1_discovery = no                 # disable discovery

#

# TCP tuning options

#

# NODELAY option in TCP disables the nagle algorithm,
if set to yes (default).

#tcp_nodelay = yes

#

# The bundles are split into chunks while they are
transmitted over TCP. This

# parameter defines the size of these chunks (4096 is
the default).

#tcp_chunksize = 4096

#

# The timeout for idle TCP connection in seconds. 0 =
disabled

#tcp_idle_timeout = 0

#######################################

# routing configuration                #

#######################################

#

# routing strategy

#
```

```
# values: default | epidemic | flooding | prophet

#

# In the "default" the daemon only delivers bundles to
neighbors and static

# available nodes. The alternative module "epidemic"
spread all bundles to

# all available neighbors. Flooding works like
epidemic, but do not send the

# own summary vector to neighbors. Prophet forwards
based on the probability

# to encounter other nodes (see draft-irtf-dtnrg-
prophet-09).

#

#routing = default

#

# forward bundles to other nodes (yes/no)

#

#routing_forwarding = yes

#

# static routing rules

# - a rule is a regex pattern

# - format is <target-scheme> <routing-node>

#

# route all bundles for "dtn://*.moon.dtn/*" to
dtn://router.dtn

#route1 = ^dtn://[[:alpha:]].moon.dtn/[[:alpha:]]
dtn://router.dtn
```

```
#

# static connections

# for configure static connections it is important to
begin with "static1_"

# and count up ("static2_", "static3_", ...)

#

### node-five.dtn ###

#static1_address = 10.0.0.5             # the node has
the address 10.0.0.5

#static1_port = 4556                    # accept
bundles on port 4556

#static1_uri = dtn://node-five.dtn # eid of the node is
"dtn://node-five.dtn"

#static1_proto = tcp                    # reachable
over TCP

#static1_immediately = yes              # connect
immediately to this node

### node-ten.dtn ###

#static2_address = 192.168.0.10         # the node has
the address 10.0.0.10

#static2_port = 4556                    # accept
bundles on port 4556

#static2_uri = dtn://node-ten.dtn  # eid of the node is
"dtn://node-ten.dtn"

#static2_proto = udp                    # reachable
over UDP

#static1_immediately = no               # connect on-
demand to this node

### prophet configuration ###
```

```
#prophet_p_encounter_max = 0.7          #affects how
strong the predictability is
                                        #increased on an
encounter

#prophet_p_encounter_first = 0.5        #the
predictability of a neighbor on the
                                       #first encounter

#prophet_p_first_threshold = 0.1        #lowest
predictability when neighbors
                                       #predictabilities
are forgotten

#prophet_beta = 0.9                     #Weight of the
transitive property

#prophet_gamma = 0.999                  #Determines how
quickly predictabilities
                                        #age

#prophet_delta = 0.01                   #(1-delta) is the
maximum predictability

#prophet_time_unit = 1                  #time unit in
seconds

#prophet_i_typ = 300                    #typical time
interval between two node
                                        #encounters

#prophet_next_exchange_timeout = 60     #timeout how
often handshakes should be
                                       #executed

#prophet_forwarding_strategy = GRTR     #The forwarding
strategy used GRTR | GTMX

#prophet_gtmx_nf_max = 30               #Maximum times to
forward in the GTMX
                                        #strategy



#####################################

# bundle security protocol            #
```

```
####################################

#

# the level specifies the security constains

#

# 0 = no constrains (default)

# 1 = accept only BAB authenticated bundles

# 2 = accept only encrypted bundles

# 3 = accept only BAB authenticated and encrypted
bundles

#security_level = 0

#

# bab default key

#

#security_bab_default_key = /etc/ibrdtn/bpsec/default-
bab-key.mac

#

# key path

#

#security_path = /etc/ibrdtn/bpsec/keys

#

# TLS for TCP convergence layer

# Authentication and encryption (optional) support for
every

# tcp connection between the daemons.

#
```

```
# certificate signed by the authority (public key)

#security_certificate = /etc/ibrdtn/tls/local.crt

# local TLS key

#security_key = /etc/ibrdtn/tls/local.key

# path to trusted certificates

#security_trusted_ca_path = /etc/ibrdtn/tls/

# set to 'yes' if tcp connections without TLS are not
allowed

#security_tls_required = yes


# set to 'yes' to disable encryption in the TLS streams

#security_tls_disable_encryption = yes

#######################################

# time synchronization                #

#######################################

#

# set to yes if this node is connected to a high
precision time reference

# like GPS, DCF77, NTP, etc.

#

#time_reference = yes

#

# set the quality of time decrease tick

#

#time_qot_tick = 0

#
```

```
# request time synchronization on discovery
#
#time_sync_on_discovery = yes
#
# announce time sync capabilities in discovery messages
#
#time_discovery_announcements = yes
#
# Parameters for the QoT aging process.
#
#time_sigma = 1.001
#time_sync_level = 0.15
#
# Adjust the clock of the host on each sync
#
#time_set_clock = no

#####################################
# DHTNameService settings           #
#####################################
#
# Enable the DHT, if it was compiled
# Default is no
#
```

```
#dht_enabled = yes

#

# Set the udp port, the DHT should working on

# Default is 9999

# If Port is 0, a random Port will be chosen for each
run

#

#dht_port = 9999


#

# Here you can choose a static DHT ID, which is very
common

# Default is none -> a random ID per run will be
generated

#dht_id = <randomstring>

#

# Enables DHT on IPv4 socket

# Default is yes

#

#dht_enable_ipv4 = yes

#

# Enables DHT on IPv6 socket

# Default is yes

#

#dht_enable_ipv6 = yes

#
```

```
# Bind the DHT to a specific IPv4 Address
# Default is the any device
#
#dht_bind_ipv4 = 127.0.0.1
#
# Bind the DHT to a specific IPv6 Address
# Default is the any device
#
#dht_bind_ipv6 = ::1
#
# Specify the file, where the DHT can save all good
nodes
# for faster restart on next session
# Default is no file, but it should be set
#
#dht_nodes_file = <filepath>


#
# Enable DNS Bootstrapping for the DHT
#
#dht_bootstrapping = yes
#
# DNS Bootstrapping by giving domain names of wellknown
nodes
#dht_bootstrapping_domains = [domain] [...]
```

```
#
# Example:
#dht_bootstrapping_domains = dtndht.ibr.cs.tu-bs.de
#
# Default is an empty string
#dht_bootstrapping_domains =
#
# IP Bootstrapping from wellknown IP (and port)
addresses of nodes
#dht_bootstrapping_ips = [ip [port]]; [ip [port]]; ...
#
# Example:
#dht_bootstrapping_ips = 192.168.0.1; 192.168.0.2 8888;
#
# Default is an empty string
#dht_bootstrapping_ips =
#
# Blacklist support of the DHT can be switch on and off
#
# Default is yes
#dht_blacklist = yes
#
# Announcing myself on the DHT
#
```

```
# Default is yes

#dht_self_announce = yes

#

# Minimum necessary rating of a DHT information

#

# The lowest rating is 0: the node information has been
sent by only one DHT node

# The maximum rating is 10 (for single lookups) and
means: 10 or more different DHT nodes sent the
information

#

# If the rating of an incoming information is lower, it
will be ignored

#

# Default is 1

#dht_min_rating = 1

#

# Allow announcing neighbours

#

# Default is yes

#dht_allow_neighbour_announcement = yes

#

# Allow all neighbours announce them to be neighbour to
me

# For privacy reasons, you could turn this off

#

# Default is yes
```

```
#dht_allow_neighbours_to_announce_me = yes

#

# Ignoring the neighbour information sent by a node,
found on the DHT

#

# Default is no

#dht_ignore_neighbour_informations = no
```

Lampiran 5. Konfigurasi /etc/rc.local

```
#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

# By default this script does nothing.


sudo iwconfig wlan0 essid "kesini"
```