IMPLEMENTASI METODE STORE AND FORWARD PADA HYPERTEXT TRANSFER PROTOCOL (HTTP)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Gigih Wicaksono
NIM: 115060801111073

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI METODE STORE AND FORWARD PADA HYPERTEXT TRANSFER PROTOCOL (HTTP)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Gigih Wicaksono
NIM: 115060801111073

Skripsi ini telah diuji dan dinyatakan lulus pada
10 November 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Aswin Suharsono S.T, M.T	Adhitya Bhawiyuga S.Kom, M.S
NIK: 201102 840919 1 001	NIK: 201405 890720 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP : 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 1 September 2016

Muhammad Gigih Wicaksono

NIM: 115060801111073

KATA PENGANTAR

Dengan menyebut nama Allah SWT Yang Maha pengasih dan Penyayang. Puji Syukur penulis panjatkan kehadirat Allah SWT karena dengan rahmat, hidayah serta kekuatan-Nya, penulis dapat menyelesaikan skripsi dengan judul "**Implementasi metode *store and forward* pada *HTTP (Hypertext Transfer Protocol)***". Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar sarjana komputer di Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya Malang.

Penyusun skripsi ini juga tidak lepas dari bantuan semua pihak yang terus memberikan semangat, kritik, saran, bimbingan, serta doa. Oleh karena itu, ucapan terima kasih penulis sampaikan kepada:

1. Kedua orang tua penulis, Bapak Drs. Ahmad Nurudin,Ak. dan Ibu Endang Setyawati yang telah memberikan semangat, motivasi, teguran, kasih sayang serta dukungan secara moral serta finansial. Adik Penulis Ghassani Gita Nirwanawati yang telah memberi semangat dan bantuan tiada henti dalam proses pengerjaan skripsi.
2. Kepada keluarga ibu di Ponorogo yang menyamangati penulis dan membantu membiayai penulis kuliah selama masa pengerjaan skripsi hingga selesai. Penulis sangat-sangat terbantu hingga akhirnya dapat menyelesaikan skripsi
3. Bapak Aswin Suharsono, S.T., M.T. selaku dosen pembimbing 1 yang telah sabar dan semangat dalam membimbing, memberikan ilmu, mengarahkan penulis dalam pembuatan skripsi hingga selesai, serta membantu dukungan secara finansial ketika penulis sempat mendapat masalah keuangan ditengah-tengah pengerjaan skripsi
4. Bapak Adhitya Bhawiyuga, S.Kom, M.S. selaku dosen pembimbing 2 yang memberikan ilmu, saran serta arahan dalam penyelesain laporan skripsi.
5. Kepada seluruh dosen di fakultas ilmu komputer yang telah memberi ilmu kepada saya selama masa perkuliahan baik ilmu tertulis maupun ilmu yang tidak tertulis.
6. Bapak Ronny Usman Santoso, S.T yang telah senantiasa membantu saya ketika menghadapi kebuntuan dalam pengerjaan mulai awal hingga akhir dalam pengerjaan skripsi.
7. Dani Tri Astiti, S.Pd yang telah dan selalu memberi semangat, motivasi, kasih sayang tiada hentinya setiap hari dan memberi kritik dan saran kepada penulis selama penulis melakukan pengerjaan skripsi hingga selesai.
8. Mas Safril B Nugroho, Fredy Dwi Setyawan, Piter Prabadi, S.Psi dan Ahmad Rizal H yang telah memberi saran, ilmu, pengalaman serta menyemangati penulis dalam mengerjakan skripsi.

9. Kepada teman-teman baik penulis yang telah mendo'akan penulis dan memberi dukungan agar penulis dapat dengan cepat menyelesaikan skripsi.

Semoga seluruh jasa dan amal baik yang telah diberikan kepada penulis mendapat balasan dari Allah SWT. Dengan segala kerendahan hati penulis sangat sadar bahwa skripsi ini jauh sekali dari kata sempurna. Penulis berharap skripsi ini dapat bermanfaat bagi masyarakat umum ataupun pihak tertentu yang membutuhkan serta pembaca khususnya mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya

Malang, 31 Juli 2016

Muhammad Gigih Wicaksono
super.mgw@gmail.com

ABSTRAK

Perkembangan internet saat ini sangat pesat. Di Desa maupun daerah terpencil perkembangan teknologi masih minim. *Delay Tolerant Network* (DTN) merupakan sebuah solusi yang ditawarkan untuk mengatasi masalah ini. Namun karena terjadi *Internet Ossification*, teknologi DTN tidak dapat diimplementasikan pada jaringan yang ada saat ini karena dapat merubah banyak infrastruktur yang sudah ada. Untuk mengatasi hal semacam ini, perlu dibuat sebuah arsitektur baru yang dapat mengatasi hal semacam ini tanpa merubah infrastruktur yang ada. Oleh karena itu diambil metode *store and forward* yang dimiliki DTN. Metode store and forward ini akan bekerja pada protokol yang umum digunakan saat ini. Protokol ini adalah protokol *HTTP*, *HTTP* dianggap lebih sederhana dan dapat berjalan di banyak jenis *Transport layer* seperti *TCP* maupun *SCTP*.

Implementasi Metode *store and forward* pada *HTTP* ini dimulai dengan membuat routing yang jelas dan pasti, agar tidak ada kesalahan dalam pengiriman file. File yang di *upload* akan disimpan pada folder khusus dan tiap file memiliki *header IP* pada *filename*. *Header IP* pada *filename* ini nanti yang akan memastikan sebuah file adalah milik *node* tertentu. Implementasi ini akan diuji beberapa *OS* (*Operating System*) agar dapat digunakan secara universal, *OS* yang digunakan adalah Windows 10, Linux dan Raspbian untuk Raspberry pi. Implementasi ini diharapkan dapat menjadi solusi *Internet ossification* saat ini.

Kata kunci: *Store and Forward, Hypertext Transfer Protocol, Delay Tolerant Network, Raspbery pi, Internet Ossification*

ABSTRACT

Development of the internet is very fast. Rural and remote areas in the technological development is still minimal. Delay Tolerant Network (DTN) is a proposed solution to overcome this problem. However, due to Internet Ossification, DTN technology can not be implemented on existing networks today because it can change a lot of existing infrastructure. To fix this problem, it should be made a new architecture that can fix with this sort of thing without changing the existing infrastructure. Therefore taken a store and forward method owned by DTN. Store and forward method will work on a common protocol used today. This protocol is the HTTP protocol, HTTP is considered simple and can run on many types of transport layer such as TCP and SCTP.

Implementation Methods store and forward on this HTTP begins with made a fix and definite routing, so that there are no errors in the transmission of files. The uploaded file will be stored in a special folder and every file has an IP header on the filename. IP Header on this filename later that will ensure a file is the property of a particular node. This implementation will be tested in multiple OS (Operating System) that can be used universally, the OS used is Windows 10, Linux and Raspbian for Raspberry pi. This implementation is expected to be a solution ossification internet today.

Keywords : *Store and Forward, Hypertext Transfer Protocol, Delay Tolerant Network, Raspbery pi, Internet Ossification*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet berkembang pesat hampir diseluruh Indonesia. Warga di kota sudah menikmati akses internet dengan mudah. Kendala pengembangan akses internet ke daerah terpencil maupun di desa terdapat pada infrastrkturnya. Kendala ini menyebabkan pengembang jaringan internet tidak berani melakukan investasi. Sebuah solusi yang ditawarkan mengatasi masalah ini adalah menggunakan teknologi DTN (Delay Tolerant Network). DTN awalnya dibuat untuk mengatasi delay yang tinggi dalam proses komunikasi antar planet. Teknologi ini dirancang untuk kondisi jaringan yang menantang (Fall, 2003) serta mendukung operasi pada jaringan tanpa memperdulikan waktu tunda yang lama (Warthman, 2003). Pengiriman data-nya, DTN menggunakan konsep store and forward, dimana data yang akan dikirimkan disimpan pada sebuah perantara tertentu kemudian di teruskan ke penerima data.

Dalam perkembangan dan riset oleh Lloy Wood, terdapat permasalahan serius pada DTN. Masalah yang terjadi pada DTN, terdapat pada cara pengiriman file yang menggunakan *Bundling/Bundle* yang dikenal dengan istilah Protokol *Bundle*. Protokol *Bundle* bermasalah karena memiliki beberapa kekurangan. Kekurangannya adalah tidak dapat mendeteksi error pengiriman file dan tidak memiliki identifikasi konten yang dikirimkan. Sehingga, walaupun DTN di implementasikan pada jaringan saat ini, hal ini akan menyebabkan masalah di kemudian hari khususnya pada proses pengiriman file-nya.

Riset mengenai DTN yang sudah dibuat dengan judul Implementasi *web service* pada Delay Tolerant Network (Nutria, 2015). Fokus yang ingin dihasilkan dari riset miliknya ini adalah, bagaimana kerja web service pada DTN. Nutria menggunakan web service, karena merupakan sebuah API (*Application Programming Interface*) yang universal. Dengan menggunakan *web service* pengiriman *request* dan respon berupa file JSON dan lebih struktur. Teknologi ini menyebabkan data yang dikirimkan antara Klien dan server memiliki format yang lebih mudah dibaca dan diolah. Namun Riset milik Nutria ini hanya dapat mengirimkan 1 buah jenis file, yaitu jenis gambar dengan ekstensi JPEG, PNG, JPG. Hal ini disebabkan karena ia menggunakan jenis encode dan decode base64. Dalam perkembangannya Internet didunia saat ini mengalami Internet Ossification.

Internet ossification merupakan sebuah kondisi dimana tingginya tingkat penolakan terhadap inovasi internet (Stephen Baucke, 2007). Penolakan ini terjadi karena infrastruktur internet sudah dibangun sangat luas hampir diseluruh dunia, sehingga apabila ada teknologi baru seperti DTN sangat susah masuk karena harus merubah infrastruktur yang ada. Oleh karena itu diperlukan sebuah teknologi baru yang dapat diimplementasikan pada jaringan yang ada saat ini tanpa merubah

infrastruktur internet yang ada saat ini. Solusi yang ditawarkan adalah dengan menggunakan metode store and forward.

Metode store and forward merupakan sebuah metode pengiriman file yang digunakan oleh DTN. Metode store and forward ini memastikan bahwa setiap file yang dikirim akan disimpan terlebih dahulu pada sebuah node perantara sebelum diteruskan ke penerima. Setiap node pada metode store and forward ini memiliki penyimpanan khusus yang berfungsi untuk menyimpan file yang masuk ke node tersebut. (warthman, 2012).

Berdasarkan penjelasan diatas, untuk mengatasi *Internet ossification* perlu menggunakan metode store and forward. Agar tidak terjadi perubahan infrastruktur yang banyak, maka metode store and forward ini akan diimplementasikan pada sebuah protokol umum yang sering digunakan saat ini, yaitu adalah protokol HTTP. HTTP dianggap lebih sederhana dan dapat berjalan di banyak jenis Transport layer seperti TCP maupun SCTP (Lloyd Wood, 2009). Protokol HTTP ini nanti akan menjadi penghubung antar node satu dengan node yang lainnya.

1.2 Rumusan masalah

Berdasarkan paparan latar belakang tersebut, maka masalah yang dapat dirumuskan adalah :

1. Bagaimana metode *store and forward* bekerja pada *HTTP* ?
2. Bagaimana mengimplementasikan metode *store and forward* pada *HTTP* dapat bekerja di *platform* Linux, Raspberry Pi dan Windows ?
3. Bagaimana per*forma* pengiriman data dengan *store and forward* dengan menggunakan protokol *HTTP* ?

1.3 Tujuan

Tujuan yang ingin dicapai dalam pembuatan skr*ip*si ini adalah sebagai berikut:

1. Membuat sebuah sistem dengan konsep *store and forward* dengan protokol *HTTP*
2. Mengimplentasikan sistem dengan konsep *store and forward* pada *platform* Linux, Linux versi *Raspi* dan *Windows*
3. Waktu tempuh pengiriman *file*

1.4 Manfaat

Penulisan tugas akhir ini diharapkan mempunyai manfaat yang baik serta berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut :

1. Umum
    - Menyediakan sebuah sistem baru yang mirip kerja *DTN* dengan menggunakan kegunaan *Web Server*.
    - Menawarkan sebuah solusi dalam pengiriman *file*

2. Khusus
   a. Penulis
      - Dapat lebih memahami cara kerja *DTN*
      - Dapat megembangkan sistem yang mirip kerja *DTN*
      - Menambah pengetahuan dan pengalaman baru terkait dengan pembuatan sistem pengiriman *file* yang menggunakan *protocol HTTP*

   b. Pembaca

      Mendapatkan wawasan terkait *DTN*, cara kerja *DTN* dan metode *store and forward* apabila menggunakan *protocol HTTP*.

1.5 Batasan masalah

Untuk menghindari semakin melebarnya masalah, maka dari rumusan masalah yang telah dipaparkan, batasan masalah yang diterapkan adalah sebagai berikut:

1. Pembahasan difokuskan pada bagaimana merancang sistem *store and forward* dengan menggunakan protokol *HTTP*.
2. Pengujian dilakukan dengan menggunakan 3 *Node* ,2 Komputer dan 1 *Raspberry Pi*
3. Menggunakan *Web Server* sebagai alternatif penghubung antar *node* untuk pengiriman *file*.
4. Maksimal *file* yang dikirimkan adalah 8 MB

1.6 Sistematika pembahasan

Sistematika pembahasan ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, yaitu sebagai berikut:

BAB I : PENDAHULUAN

Pada bab ini dijabarkan latar belakang penulis memilih topik skripsi ini, rumusan masalah mengenai perancangan sebuah sistem yang menganut cara kerja *DTN* serta dijelaskan mengenai batasan masalah agar tidak melebar dari rumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II : LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang teori-teori yang digunakan serta kajian pustaka yang mengenai pengertian *Web Server*. *DTN*, Raspberry Pi, Protokol *HTTP* perangkat lunak pendukung sistem, ulasan mengenai referensi paper yang terkait dengan penelitian ini, bahasa pemrograman PHP dan Python, kajian mengenai bahasa pemrograman PHP dan Python, serta bahan kajian lain yang mendukung.

BAB III : METODOLOGI

Membahas tentang metode yang digunakan dalam penulisan yang terdiri dari studi literatur, analisa kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem dan analisa hasil pengujian, serta pengambilan kesimpulan dan saran. Serta membahas tentang cara kerja sistem. Analisa kebutuhan pengiriman *file* dengan metode *store and forward.*

BAB IV : REKAYASA PERSYARATAN / KEBUTUHAN

Membahas tentang hasil rancangan dari sistem yang dibuat serta membuat analisis pengiriman *file* yang dikerjakan.

BAB V : PERANCANGAN DAN IMPLEMENTASI

Memuat proses dan hasil dari implementasi metode *store and forward* pada HTTP

BAB VI : PENGUJIAN

Menunjukkan hasil pengujian dari implementasi yang telah dibuat

BAB VII : PENUTUP

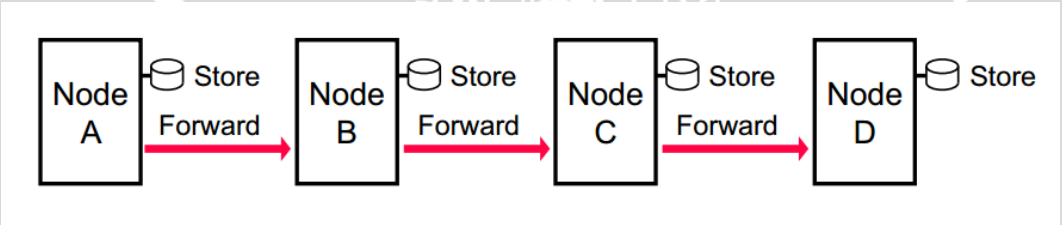
Memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Store and Forward

Store-Forward merupakan salah satu teknik yang dikenalkan pertama kali melalui RFC 2305 oleh Internet Engineering Task Force(IETF) dan juga sama dengan standarsisasi ITU-T.37. Dengan menggunakan metode store and forward ini dapat dilakukan beberapa konfigurasi yang dapat menciptakan teknologi alternatif dari kondisi jaringan yang sudah ada. (Achmad Subhan K).

Store and forward adalah sebuah metode yang digunakan oleh Teknologi Delay Tolerant Network untuk komunikasi antar planet. Metode store and forward ini sebenarnya sudah digunakan untuk komunikasi kuno seperti *post* maupun *pony-express*. Dalam perkembangannya metode store and forward ini dipakai dibeberapa kondisi seperti pengiriman file antar server maupun file antar pc yang berbeda(warthman 2013).



Gambar 2.1 Alur *Store and Forward* pada DTN (warthman, 2013)

Berdasar gambar diatas metode store and forward ini dapat bekerja perlu dibuat sebuah media penyimpanan khusus pada setiap node. Media penyimpanan ini nantinya akan menjadi wadah penerimaan file dari node sebelumnya. File akan dikirimkan dan diteruskan ke node selanjutnya, apabila node selanjutnya dalam keadaan aktif atau hidup. Apabila node selanjutnya dalam keadaan mati atau tidak aktif maka file akan disimpan oleh node yang memiliki file hingga, node selanjutnya aktif berdasar jalur routing yang sudah dibuat(Warthman,2013). Metode Store and forward ini sangat cocok digunakan untuk kondisi jaringan yang memiliki tingkat latensi tinggi (warthman, 2013).

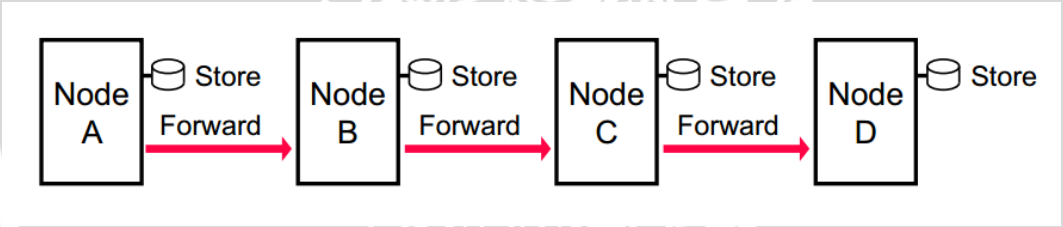
2.2 Delay Tolerant Network (DTN)

Delay Tolerant Network (*DTN*) merupakan sebuah arsitektur pada jaringan komputer yang tidak mempermasalahkan waktu *delay* yang tinggi. *DTN* dirancang untuk kondisi jaringan yang menantang (Fall, 2003). *DTN* menyediakan suatu mekanisme yang dapat menghubungkan suatu jaringan regional yang memiliki karakteristik delay yang berbeda-beda (Fall, 2003).

Protokol internet yang ada saat ini belum dapat menjangkau semua karakteristik internet yang dibutuhkan pengguna. Beberapa karakteristik yang belum dapat terpenuhi antara lain (Warthman, 2003) :

- *Intermittent Connectivity*, jika tidak terdapat koneksi *end-to-end* antara sumber dan tujuan–*network partitioning*–maka komunikasi *end-to-end* menggunakan protocol *TCP/IP* tidak mungkin dilakukan.
- *Long or Variable delay,* menyebabkan waktu tunda yang panjang antar satu *node* dengan *node* lainnya. Sehingga hal tersebut dapat menghambat proses pengiriman *ACK* yang seharusnya dikirimkan secara cepat.
- *Asymmetric Data Rates*, internet mendukung mode asimetris laju data dua arah untuk pengguna TV kabel ataupun *asymmetric DSL (Digital Subscriber Line) access*. Akan tetapi jika laju asimetris terlalu besar, maka komunikasi juga tidak dapat dilakukan.
- *High Error Rates*, pengiriman ulang paket data akibat kesalahan pengiriman data, dapat membuat lalu lintas pengiriman data semakin padat. Maka dari itu semakin sedikit kesalahan pengiriman data, maka komunikasi antara sumber dan tujuan akan semakin lancar.

DTN mengatasi keempat masalah diatas dengan menggunakan metode *store and forward message switching*. Berikut adalah cara kerja *DTN*



Gambar 2.2 Alur pengiriman *DTN*

Sumber : Warthman 2003

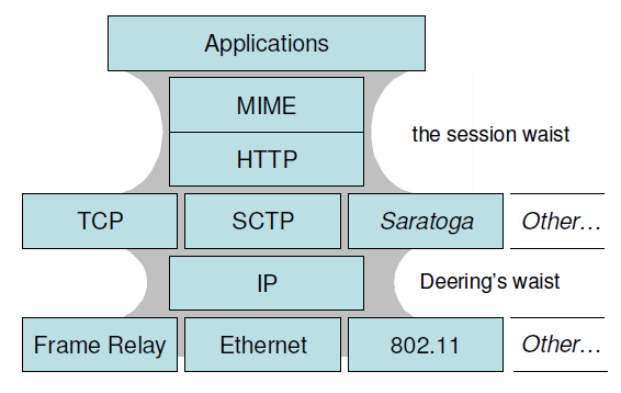
node pengirim (A) mengirim paket data menuju *node* penerima(D), namun paket data yang dikirim tadi di simpan dahulu di *node* kurir(B dan C). *Node* kurir ini akan mengirim paket data ke *node* penerima apabila *node* penerima siap untuk menerima paket data. Pada proses pengiriman *DTN*, tiap *node* memerlukan *storage* yang tetap dalam memproses *file* yang dikirim. *Storage* yang tetap ini d*IP*erlukan karena beberapa hal sebagai berikut (Warthman,2003):

- Dalam waktu beberapa saat terkadang tidak ada komunikasi yang terhubung antar *node*.
- Sebuah *node* terkadang dapat mengirimkan *file* lebih cepat atau lebih lambat dari *node* lainnya.

Sebuah pesan ketika sudah di transmisi, maka perlu untuk ditransmisi ulang apabila terjadi sebuah kesalahan dalam pengiriman, atau ketika *node* selanjutnya menolak untuk melanjutkan pesan.

2.3 Moving DTN over *HTTP*

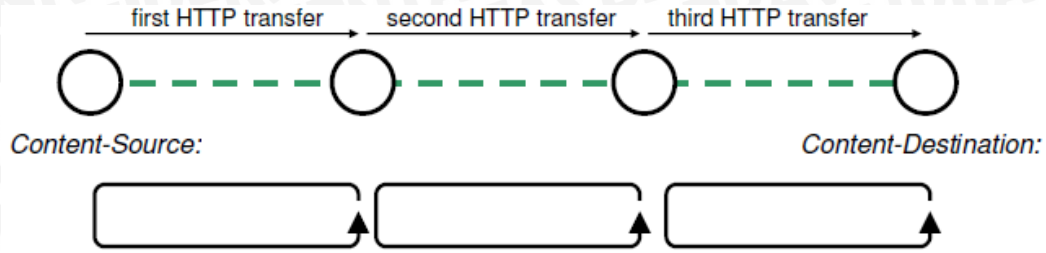
Kajian pustaka pada penelitian ini adalah membahas paper "***Moving data in DTNs with HTTP and MIME***" yang dilakukan oleh Lloyd Wood, Peter Holliday, Daniel Floreani dan Ioannis Psaras. Dasar penelitiannya adalah karena banyak komunikasi yang terjadi di dunia *internet* dengan menggunakan aplikasi berbasis *web*. Penelitian yang dilakukan adalah mengenai konsep pengiriman data yang mirip kerja *DTN* namun menggunakan *MIME* (*MultIPurpose Internet Mail Extensions*)



Gambar 2.3 *Protocol waist* pada jam pasir
Sumber : Lloyd Wood 2009

Gambar diatas menjelaskan protokol *HTTP* dan *MIME* menjadi *protokol waists* pada struktur jam pasir jaringan. Tidak ada perkembangan yang drastis pada protokol *HTTP / Transport layer. MultIPurpose Internet Mail Extensions (MIME)* menyediakan cara sederhana untuk menggambarkan jenis data yang dikirim dan penggunaannya. *Email* dan *web* menggunakan *MIME* ekstensif untuk membawa *file* yang berbeda.

Protokol yang digunakan oleh Lloyd adalah protokol *HTTP*. (*HTTP*) secara universal diakui sebagai metode sederhana untuk membawa objek yang ada pada *MIME*. *HTTP* juga protokol umum yang dapat dikembangkan. Lloyd menjelaskan upaya untuk menunjukkan independensi *HTTP* dari *TCP* dengan menggunakan *HTTP* melalui protokol transportasi lainnya. *HTTP* memiliki sebuah kelemahan yaitu sangat tidak cocok dalam kondisi jaringan yang memiliki latensi yang tinggi. Protokol transport sangat cocok dalam lingkungan *DTN.* Karena itu Konsep *HTTP-DTN* dapat dibuat. Konsep ini menjalankan *DTN* dengan bantuan protokol *HTTP*. Konsep kerja *HTTP-DTN* ini mirip *DTN* pada umumnya. Paket data dikirim antar *node* dengan bantuan protokol *HTTP.*



Gambar 2.4. *HTTP-DTN* transfer antar *node*
Sumber : Lloyd 2009

2.4 Internet Ossification

Internet ossification adalah sebuah kondisi dimana tingginya tingkat penolakan terhadap inovasi *internet* (Stephen Baucke, 2007). *Internet Ossification / ossification* sebenarnya adalah sebuah tahap evolusi alami dalam perkembangan teknologi yang sangat sukses. *Internet ossification* merupakan masalah yang serius bagi komunitas pengembang jaringan dan negara (David Taylor,2005) *Internet* saat ini berkembang sangat jauh dan bagus, namun banyak kendala dalam implementasi untuk kehidupan sehari-hari. Layanan pada *internet* saat ini terbatas pada kebutuhan yang di perlukan. Hal ini menambahkan sebuah fakta, setiap kali fungsi tambahan sederhana seperti pengiriman paket tidak diimplementasikan pada lapisan jaringan yang rendah tetapi pada *protocol* lapisan yang lebih tinggi (Schindler,Felix2010).

Kasus sederhana terhadap *internet ossification* ini adalah *IP*v6 yang sudah ditemukan. *IP*v6 awalnya diciptakan dengan alasan *IP*v4 dimasa mendatang tidak dapat mencukupi kebutuhan *IP* didunia. Namun, sampai kini implementasi *IP*v6 didunia belum maksimal.

Kasus lain terhadap *Internet Ossification* adalah pada pengimplementasian DTN. Penelitian terkait DTN sangat banyak, namun hingga kini teknologi DTN tidak diimplementasikan secara luas kepada masyarakat. Hal ini terjadi karena, apabila DTN di implementasikan pada jaringan saat ini akan merubah infrastruktur jaringan yang ada. Apabila merubah struktur jaringan yang ada akan membutuhkan waktu yang lama dan biaya yang cukup tinggi.

2.5 Web Server

2.5.1 Pengertian Web Server

Web Server merupakan sebuah perangkat lunak dalam *Server* yang berfungsi menerima permintaan (*request*) berupa halaman *web* melalui *HTTP* atau *HTTPS* dari klien yang dikenal dengan *browser web* dan mengirimkan kembali (*response*) hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen HTML (Emka,2012). Dapat disimpulkan bahwa *web Server* merupakan pelayan (pemberi layanan) bagi *web client* (*browser*)

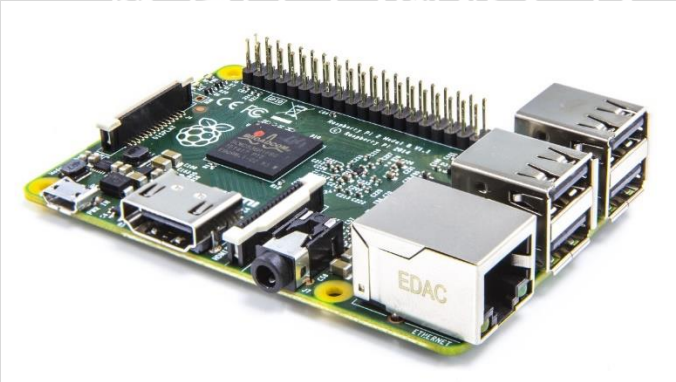
Pada Implementasi kali ini *web Server* yang digunakan adalah t*IP*e apache. **Apache** adalah *web Server* yang dapat dijalankan di banyak sistem operasi (Unix,BSD, Linux, Microsoft Windows dan Novell Netware serta plat*form* lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web*/www ini menggunakan *HTTP*

2.5.2 Fungsi Web Server

Fungsi utama dari *web Server* adalah untuk mentransfer atau memindahkan berkas yang diminta oleh pengguna melalui protokol komunikasi tertentu. (dedeerik,2015) Sebuah halaman *web* biasanya terdiri dari berbagai macam jenis berkas seperti gambar, *video*, teks, *audio*, *file* dan lain sebagainya, maka pemanfaatan *web Server* berfungsi juga untuk mentransfer keseluruhan aspek pemberkasan dalam halaman tersebut, termasuk teks, gambar, *video*, *audio*, *file* dan sebagainya.

HTTP (Hypertext Transfer Protocol) adalah protokol yang digunakan oleh *web Server* dan *web browser* untuk dapat berkomunikasi antara satu sama lain. Sedangkan *HTTPS (Hypertext Transfer Protocol Secure)* adalah merupakan versi aman (*secure*) dari *HTTP*. Biasanya protokol *HTTP* menggunakan *port* 80 dan protokol *HTTPS* menggunakan *port* 443.

2.6 Raspberry Pi



Gambar 2.5 Raspberry pi

Raspberry Pi sering disingkat dengan nama ***Raspi***, adalah komputer papan tunggal (*single-board circuit*; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresousi tinggi (wik*IP*edia). *Raspberry pi* memiliki sebuah sistem operasi khusus yang bernama *raspbian*, atau bisa di *install* dengan sistem operasi *windows 10 IoT core* maupun *linux* yang mendukung *32-bit ARM architecture* yaitu ***ARMv7-A.***

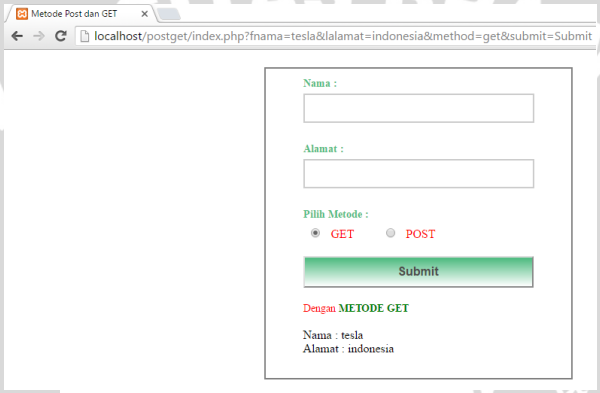
Pada Implementasi kali ini *raspberry pi* yang digunakan diinstall dengan Lubuntu 16.04 versi *raspbian*. *Lubuntu* digunakan dengan alasan memiliki booting yang cepat dan tidak memerlukan *RAM* yang tinggi untuk penggunaannya (*HelpUbuntu.com*). *Raspberry pi* pada implementasi ini digunakan sebagai *node*

kurir. *Node* kurir berfungsi untuk melanjutkan pengiriman dalam metode *store and forward*

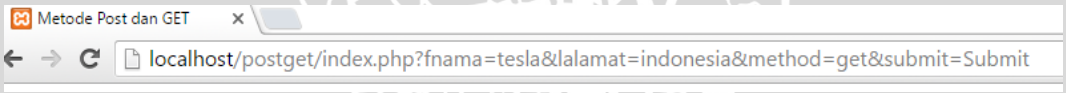
2.7 HTTP

2.7.1 GET

Get adalah metode untuk *request-response* antara *client* dan *Server*. Metode *GET* digunakan untuk melakukan permintaan data pada sumber tertentu. Dengan menggunakan metode *get* nilai dari variabel dapat dilihat langsung di dalam ***URL*** yang dikirimkan. Sebagai contoh apabila anda mengirimkan sebuah *request* ke *Server*, maka *Server* akan menampilkan seperti pada gambar dibawah ini



Gambar 2.6 *Form* submit Metode Get



Gambar 2.7 metode *GET* menampilkan variabel pada *URL*

Berikut syntax GET :

Tabel 2.1 Syntax GET

Syntax GET
`http://localhost/postget/index.php?fnama=tesla&lalamat=indonesia&method=get&submit=Submit`

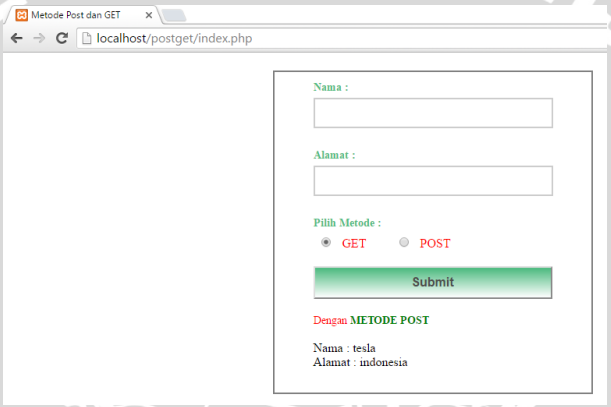
Berdasar gambar diatas *Server* membalas *request* klien dengan menunjukkan seluruh isi variabel yang diminta. Variabel yang diminta adalah "nama" dan "alamat". Metode *Get* sangat cocok digunakan apabila dalam memproses *form php* hanya untuk mengambil data atau sebuah variabel. Metode ini sangat cocok untuk proses menampilkan data saja bukan pengolahan data. Metode *Get* ini sangat sering digunakan pada sebuah *website* untuk mencari sebuah data pada *database*. (duniailkom,2015). Metode *Get* sangat tidak cocok digunakan untuk pemrosesan data penting seperti *data user*, transaksi keuangan dan *data* sensitif lainnya.

Beberapa hal yang bisa dilakukan oleh metode GET :

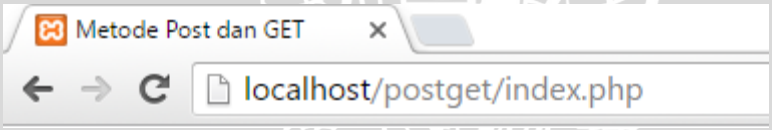
- *GET request* bisa di *cache*
- *GET request* dapat dilihat di riwayat *browser*
- *GET request* dapat di *bookmark*
- *GET request* memiliki batas karakter yang bisa digunakan (2047 karakter)
- *GET request* sangat baik digunakan hanya untuk mengambil *data*

2.7.2 POST

Post adalah metode untuk *request-response* antara *client* dan *Server*. Metode *post* digunakan untuk menyerahkan *data* untuk di proses ke sumber tertentu. Metode ini tidak menampilkan isi *form* pada *url*, sehingga metode ini sesuai untuk *data-data* yang bersifat sensitif seperti *username* dan *password*. Metode ini biasanya digunakan dalam melakukan pemrosesan *data* pada sebuah *website*. (duniailkom,2015)



Gambar 2.8 *Form submit Metode Post*



Gambar 2.9 Metode *post* tidak menampilkan isi variabel

Berikut Syntax POST :

Tabel 2.2 Syntax POST

Syntax POST
`http://localhost/postget/index.php`

Metode *post* sangat bagus jika digunakan dalam pemrosesan *data* penting dan sensitif, seperti pemrosesan *form register* dan *form login*. Metode ini banyak digunakan oleh *website-website* masa kini.
Beberapa hal yang bisa dilakukan dengan metode *post*

- *POST request* tidak pernah masuk dalam *cache*
- *POST request* tidak ada dalam riwayat *browser*
- *POST request* tidak dapat di *bookmark*
- *POST request* tidak memiliki batasan panjang *data*

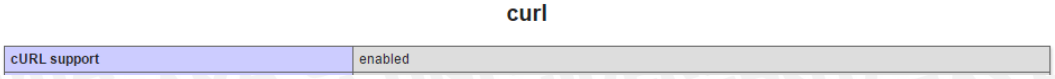
2.8 Cron

Cron adalah sebuah aplikasi bawaan sistem operasi yang dapat mengatur sebuah penjadwalan. Biasanya digunakan untuk dalam penjadwalan dalam sistem operasi, bisa *shutdown* pada waktu tertentu, *restart* pada waktu tertentu atau lainnya. Penggunaan *cron* ini hanya bisa dilakukan apabila kita sudah men set sebuah perintah ,atau *script shell* untuk menjalankan secara berkala pada waktu dan tanggal tertentu. Dalam perawatan sebuah *Server*, *cron* sangat sering diterapkan. *Cron* sangat cocok digunakan untuk melakukan penjadwalan yang berulang-ulang. *Cron* dapat digunakan pada semua sistem operasi, termasuk *android* dan perangkat *Raspberry pi*. *Android* yang berbasis *linux* dapat menggunakan *cron* dengan perintah *cron* yang sama juga seperti di *linux*. Penggunaan *cron* pada *Android* biasanya dilakukan untuk men *set up alarm*, *restart android* maupun untuk *shutdown* dan *turn on android* lagi.
Cron yang digunakan pada perangkat *raspi* ini sama dengan cara kerja *cron* pada *linux* selama menggunakan sistem operasi berbasis *Linux*. Apabila menggunakan *Windows 10 IoT core* maka *cron* yang digunakan sama dengan *task scheduler* milik *Windows*

2.9 Curl

Proyek perangkat lunak komputer yang menyediakan perpustakaan dan alat baris perintah untuk mentransfer *data* menggunakan berbagai protokol. Proyek Curl menghasilkan dua produk, *libcurl* dan *Curl*. Ini pertama kali dirilis pada tahun 1997. Nama awalnya dibuat nama aplikasi ini adalah "*see URL*" akhirnya di sederhanakan menjadi "*cURL/ CURL*".

Secara umum, *cURL* adalah perangkat lunak yang digunakan untuk mentransfer *data* dari dan ke *Server*. Sebenarnya ada banyak perangkat *transfer data* layaknya *cURL*, namun *cURL* memiliki fitur yang lebih lengkap diantara perangkat-perangkat lainnya. Diantaranya dukungan terhadap *HTTP*, *FTP*, SFTP, *SOCKS, TFTP, IMAP, POP3, SMTP*. *Curl* pada *XAMPP* berada satu paket dengan *php*. Cara mengetahui *curl* ada atau tidak pada *XAMPP windows* dan *LAMP Server Linux* adalah dengan masuk ke *PHP* info. *PHP* info akan menunjukkan seluruh fitur *PHP* yang aktif dan sudah berhasil jalan. *PHP curl* aktif seperti gambar dibawah ini.



Gambar 2.10 *php curl* dalam keadaan *enabled*

Pada *XAMPP* Windows, biasanya *PHP Curl* sudah menjadi satu paket yang terinstall dan setelah diinstall sudah dalam keadaan aktif (*enabled*). Pada *Linux* apabila kita *install Lamp Server*, *PHP Curl* tidak menjadi satu paket yang ter*install*,

sehingga kita perlu meng*install PHP Curl* dengan mengetikkan perintah sebagai berikut :

Tabel 2.3 perintah *install PHP Curl*

```
$ sudo apt-get install php-curl
```

Cara menjalankan *CURL* secara sederhana adalah seperti menggunakan *script* dibawah ini

Tabel 2.4 contoh *test url*

Script test CURL

```
1: <?php
2:
3: $ch = CURL_init("HTTP://www.example.com/");
4: $fp = fopen("example_homepage.txt", "w");
5:
6: CURL_setopt($ch, CURLOPT_FILE, $fp);
7: CURL_setopt($ch, CURLOPT_HEADER, 0);
8:
9: CURL_exec($ch);
10: CURL_close($ch);
11: fclose($fp);
12: ?>
```

Sumber : *HTTP*://php.net/manual/en/*CURL*.examples-basic.php

Pada *source code* diatas, *CURL* perlu dideklarasikan terlebih dahulu untuk mengakses alamat *web*site tertentu. Dilihat pada baris 3, $ch adalah variabel untuk deklarasi *session* CURL_init() untuk memanggil *web*site : *http*://example.com. Sedangkan variabel $fp pada baris ke 4 adalah untuk deklarasi *file* yang di *open* oleh *CURL* pada *web*site yang dituju. *File* yang diakses adalah *example_homepage.txt*. pada akhiran di tambah simbol "w" yang berarti *write* ulang (*overwrite*).

2.10 PHP

PHP adalah bahasa pemrograman *script Server-side* yang didesain untuk pengembangan *web*. Selain itu, *PHP* juga bisa digunakan sebagai bahasa pemrograman umum. *PHP* di kembangkan pada tahun 1995 oleh Rasmus Lerdorf, dan sekarang dikelola oleh The *PHP* Group. Situs resmi *PHP* beralamat di *http://www.php.net. PHP* disebut bahasa pemrograman *Server side* karena *PHP* d*i*proses pada komputer *Server*. Hal ini berbeda dibandingkan dengan bahasa pemrograman *client-side* seperti *JavaScript* yang d*I*Proses pada *web browser* (*client*).

Pada awalnya *PHP* merupakan singkatan dari *Personal Home Page*. Awalnya *PHP* digunakan untuk membuat *web*site pribadi. Dalam beberapa tahun perkembangannya, *PHP* menjelma menjadi bahasa pemrograman *web* yang powerful dan tidak hanya digunakan untuk membuat halaman *web* sederhana, tetapi juga *website* populer yang digunakan oleh jutaan orang seperti *wikipedia*, wordpress dan joomla. Saat ini *PHP* adalah singkatan dari *PHP*: *Hypertext*

Preprocessor, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya terdiri dari singkatan itu sendiri: PHP: Hypertext Preprocessor.(duniailkom,2014)

PHP dapat digunakan dengan gratis (*free*) dan bersifat *Open Source*. *PHP* dirilis dalam lisensi *PHP* License, sedikit berbeda dengan lisensi *GNU General Public License (GPL)* yang biasa digunakan untuk proyek *Open Source*.

2.11 Python

Python merupakan bahasa pemrograman yang dibuat oleh Guido Van Rossum pada tahun 1991. *Python* adalah sebuah bahasa pemrograman yang bersifat intrepeter, interaktif dan *object oriented*.

Python memiliki kepustakaan yang luas, dalam distribusi *Python* telah disediakan modul-modul 'siap pakai' untuk berbagai keperluan. *Python* juga memiliki tata bahasa yang jernih dan mudah dipelajari. Selain itu *Python* memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang *source code*. Sampai saat ini *Python* masih dikembangkan oleh *Python Software Foundation*. Bahasa *Python* mendukung hampir semua sistem operasi, bahkan untuk sistem operasi *Linux*, hampir semua distro-nya sudah menyertakan *Python* di dalamnya. *Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai *platform* sistem operasi.

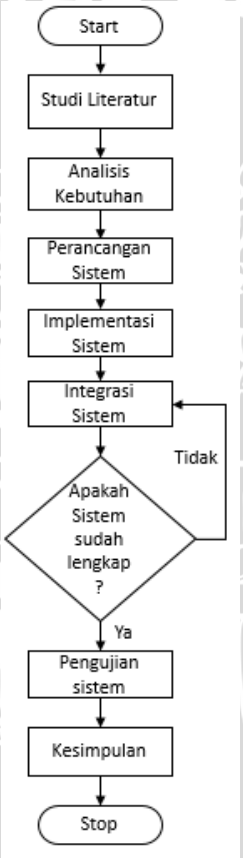
BAB 3 METODOLOGI

3.1 Jenis Penelitian

Jenis penelitiannya adalah Penelitian implementatif. Penelitian jenis ini merupakan sebuah kegiatan penelitian dalam rangka membuat implementasi topologi jaringan yang sistematis. Penelitian ini dimulai dari analisis kebutuhan, perancangan jaringan berdasarkan analisis kebutuhan, konstruksi dan perancangan jaringan serta pengujian hasil jadi rancangan jaringan.

3.1.1 Metodologi Penelitian

Metodologi penelitian yang akan dilakukan pada penelitian ini secara umum ditunjukkan pada diagram alir di bawah ini.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.2 Studi Literatur

Studi literatur dilakukan bertujuan untuk mempelajari serta memahami konsep-konsep sistem agar ketika dilakukan perancangan tidak terlalu mengalami kendala. Jenis literatur yang digunakan adalah artikel jurnal. Studi literatur ini meliputi pembahasan mengenai Implementasi jaringan didaerah terpencil ,seperti *DTN*. Pehamaman mengenai *Internet Osification.* Pembahasan mengenai referensi *paper* yang berkaitan dengan penelitian ini, pembahasan mengenai *web Server*,

pembahasan mengenai *HTTP* dan beberapa tools untuk otomatisasi perancangan sistem seperti *Cron* dan *CURL*. Pada tahap ini juga dilakukan proses pendefinisian masalah yang akan dibahas. Hal ini dilakukan agar mendapat pemahaman yang lebih mendalam terhadap pokok bahasan yang diangkat.

3.2.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menentukan apa saja yang dapat dilakukan oleh sistem. Pada tahap ini dijelaskan mengenai batasan sistem serta tujuan yang dapat dicapai oleh pengguna. Kebutuhan fungsional yang nantinya akan disediakan oleh rancangan sistem ini adalah sebagai berikut :

1. Sistem ini menyediakan pengiriman *file* dengan konsep *Store and Forward* dengan bantuan perangkat *Raspberry Pi*.
2. Dengan adanya konsep *store and forward* maka akan dibuat sistem otomatisasi untuk ping *node* dan pengiriman *file* yang sudah di upload. Otomatisasi sistem menggunakan *cron* dan *CURL*
3. Menggunakan protokol *HTTP* umum dibantu dengan *web Server* sebagai perantara pengiriman *file*
4. Sesuai dengan batasan masalah yang sudah dijelaskan sebelumnya, maka fungsi yang ada pada sistem adalah :

    - Pembahasan difokuskan pada bagaimana merancang sistem *store and forward* dengan menggunakan protokol *HTTP*.
    - Sistem ini menggunakan linux ubuntu
    - Pengujian dilakukan dengan menggunakan 3 *Node* ,2 Komputer dan 1 Raspberry Pi
    - Menggunakan *web Server* sebagai alternatif penghubung antar *node* untuk pengiriman *file*.
    - Maksimal *file* yang dikirimkan adalah 8 MB

3.2.2 Perancangan Sistem

Perancangan arsitektur sistem adalah tahap dimana penulis mulai membuat suatu sistem yang mampu memenuhi semua kebutuhan fungsional aplikasi dalam tugas akhir ini. Pada tahap ini dideskripsikan arsitektur sistem yang meliputi pembuatan simulasi jaringan sistem, proses aliran *data*, serta penjelasan komponen perangkat lunak yang digunakan dalam sistem. Teori-teori dari daftar pustaka digabungkan dengan ilmu yang didapat kemudian diimplementasikan untuk membuat dan serta mengembangkan sebuah arsitektur jaringan dengan konsep *Store and Forward* untuk pengiriman *file*

3.2.3 Implementasi Sistem

Implementasi sistem dilakukan dengan mengkonstruksikan serta menjalankan rancangan sistem yang telah dibuat. Implementasi dari hasil rancangan sistem diharapkan dapat merepresentasikan hasil rancangan sistem dari analisa kebutuhan yang ada. Implementasi sistem dilakukan dengan membuat

tiga buah *node*, dua *node* berupa *PC* dengan *OS Linux Lubuntu 16.04* pada *node* utama *dan Windows 10* pada *node* tujuan akhir*,* dan satu *node* menggunakan *raspberry pi*. Untuk perantara pengiriman *file* digunakan sebuah *web Server*. Untuk melakukan proses otomatisasi maka digunakan *cron* untuk melakukan ekseskusi *script python*.

3.2.4 Integrasi Sistem

Intergrasi sistem dilakukan untuk membentuk sistem secara lengkap. Proses integrasi adalah dengan melakukan proses pengiriman *file* antar *node* (Studi kasus pengiriman *file* dari *node* utama ke *node* akhir). Integrasi sistem akan selesai dilakukan apabila semua kebutuhan sistem sudah dilaksanakan dan sesuai dengan tujuan implementasi sistem

3.2.5 Pengujian Sistem dan Analisa hasil pengujian

Pengujian sistem dilakukan untuk mengetahui keberhasilan dan kekurangan yang dimiliki oleh rancangan jaringan yang telah dibuat. Pengujian sistem juga dilakukan untuk mengetahui tingkat keberhasilan pengiriman paket *data* antar *node* dengan konsep *store and forward.* Sementara analisa hasil pengujian dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan tujuan perancangan sistem tersebut.

Pada sistem kali ini ,hal yang diuji adalah sebagai berikut :

- Metode *store and forward* jalan atau tidak
- Pengiriman *file* jalan atau tidak (fungsional)
- Kecepatan pengiriman *file*

3.2.6 Pengambilan kesimpulan dan saran

Kesimpulan diambil berdasarkan semua tahapan yang telah dilalui dalam penelitian sehingga menghasilkan output yang sesuai dengan tujuan penelitian dan harapan peneliti. Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem telah selesai dilakukan serta didasarkan pada kesesuaian antara teori dan praktik. Kesimpulan diambil untuk menjawab rumusan permasalahan yang telah ditetapkan sebelumnya. Tahap terakhir adalah penulisan saran yang dimaksudkan untuk memperbaiki kekurangan-kekurangan yang terjadi dan menyempurnakan penulisan serta dapat dijadikan pertimbangan atas penelitian berikutnya.

BAB 4 REKAYASA PERSYARATAN / KEBUTUHAN

4.1 Pengiriman file

Pengiriman *file* antar komputer saat ini sangat berkembang. Pengiriman dapat dilakukan dengan menggunakan jaringan kabel maupun nirkabel. Pengiriman *file* akan gagal apabila salah satu komputer yang terhubung dalam jaringan terputus pada saat pengiriman *file* dilakukan. Metode *store and forward* yang ada pada *DTN* menjadi solusi dari masalah ini.

Metode *store and forward* bekerja untuk menyimpan *file* pada storage tertentu sebelum *file* dikirim. *File* akan dikirimkan apabila *node* tujuan *file* dalam keadaan hidup. *File* yang dikirim tidak akan pernah sampai ke *node* tujuan apabila *node* tujuan tidak hidup, sehingga *file* masih aman di simpan di *storage* tertentu.

Era saat ini protokol yang sangat umum digunakan adalah protokol *HTTP*. Protokol *HTTP* sangat banyak digunakan pada jenis jaringan maupun standar perangkat keras yang ada. Protokol *HTTP* juga sangat sering digunakan dalam pengiriman *file*.

Metode *store and forward* dan protokol *HTTP* menjadi solusi serta tempat implementasi sistem. Metode *store and forward* pada *HTTP* diharapkan dapat menjadi solusi banyak permasalahan pengiriman *file*.

4.2 Kebutuhan Sistem

4.2.1 Perangkat Keras

Perangkat keras yang dibutuhkan untuk membuat sistem adalah :

- 1 buah *PC* dengan *OS linux Lubuntu 16.04* (menggunakan 1 *Virtual machine*, Mesin *virtual* 1 untuk *node level 1* dengan *IP* statis "192.168.1.2") .
- 1 buah *Raspberry pi* yang di *install* dengan *OS linux* untuk raspberry yaitu Lubuntu 16.04 dengan *IP* statis yaitu "192.168.1.3" dan "192.168.1.4").
- 1 buah *PC* dengan *OS Windows 10*. Dengan *IP* "192.168.1.5", "192.168.1.6", "192.168.1.7" dan "192.168.1.8". *IP* dirubah menjadi statis yang disesuaikan dengan skenario pengiriman.
- *Access point* untuk menghubungkan antar *node*
- Kabel *LAN (Local Area Network)* untuk menghubungan *Access Point* ke *Raspberry pi*

4.2.2 Perangkat Lunak

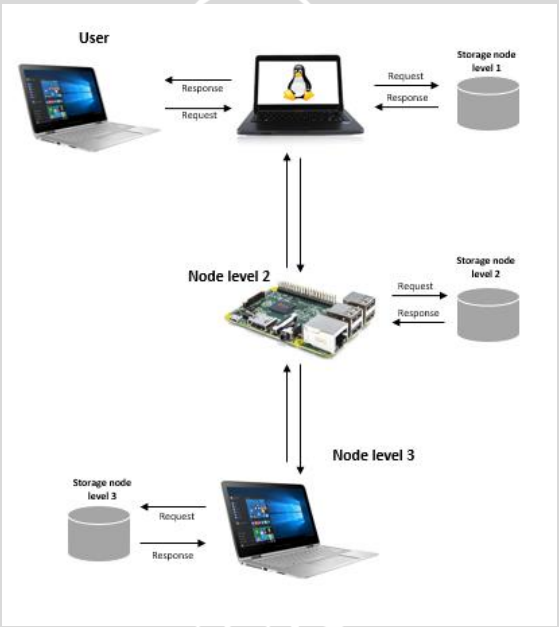
Perangkat lunak yang dibutuhkan untuk membuat sistem adalah :

- OS Linux Lubuntu 16.04 yang di*install* di PC maupun *Raspberry pi(Lubuntu* versi *Raspberry)*
- *OS* Windows 10
- *Web Server*

- *PHP*
- *Cron*
- *Python*
- *CURL*

4.3 Alur kerja sistem

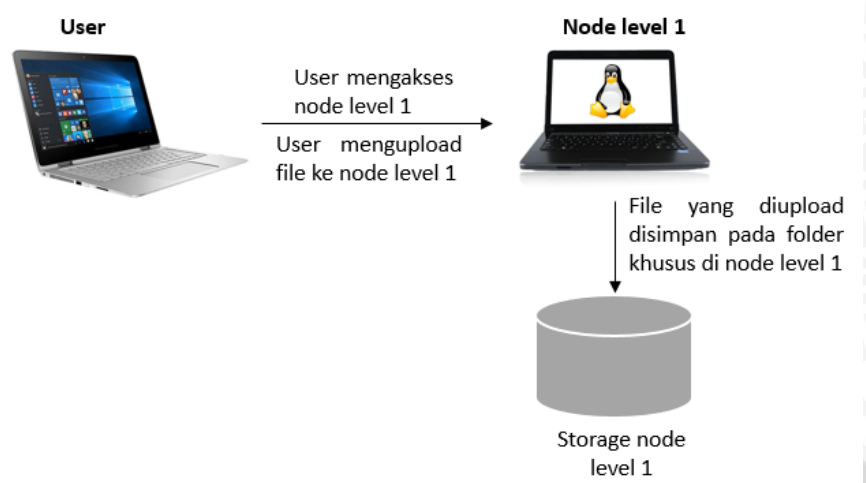
Sistem ini memiliki 3 level node, yaitu node level 1, level 2 dan level 3. Setiap level pada node memiliki perangkat tersendiri. *Node level 1* adalah *PC* (*Virtual*) dengan *OS Linux Lubuntu 16.04. Node level 2* adalah *Raspberry Pi* dengan *OS Lubuntu 16.04* versi *Raspbian* (*Raspberry Pi*). Sedangkan *node level 3* menggunakan *PC/Laptop* dengan *OS Windows 10.* Setiap *node* memiliki fungsi yang berbeda. Node level 1 merupakan node yang diakses oleh user untuk tempat user menyimpan file serta melanjutkan pengiriman file ke node tujuan. Node level 2 berfungsi sebagai node kurir, node ini akan melanjutkan pengiriman dari node level 1 menuju node level 3. Namun, karena routing dari sistem sudah jelas, maka pengiriman file tidak akan salah.



Gambar 4.1 Alur pengiriman antar *node*

4.3.1 User upload file

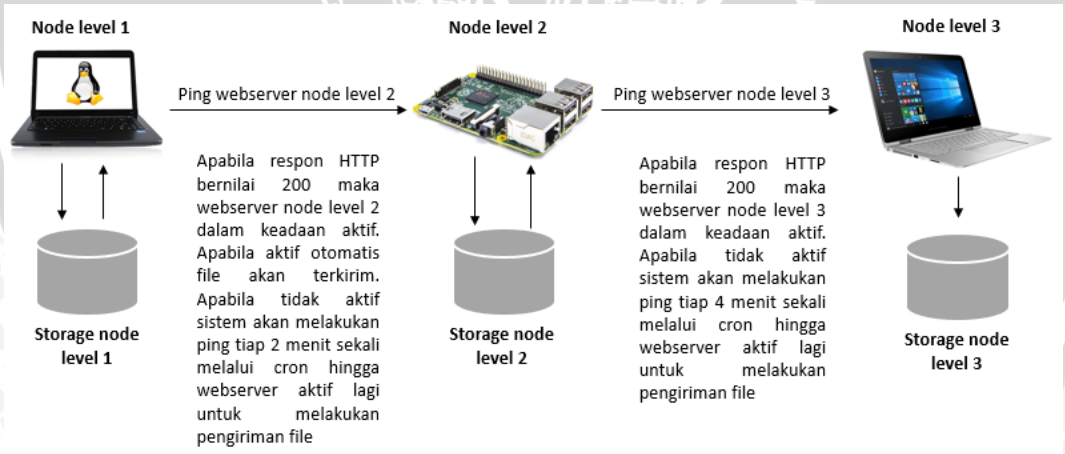
Node level 1 diakses oleh *user* dari luar *node*, *user* kemudian mengakses *form upload* pada *node* level 1. *User* kemudian meng*upload file* yang ingin dikirimkan menuju tujuan akhir. *File* yang berhasil di *upload* disimpan pada *folder* khusus yang terdapat pada *node level* 1.



Gambar 4.2 *Upload file* oleh *user* ke *node level* 1

File yang di*upload* di *node level* 1 semua memiliki tujuan alamat tertentu yang sudah ditentukan *user* ketika melakukan *upload data*. *File* yang telah di *upload* akan terkirim secara otomatis apabila *node level* 1 dan *node level* 2 *web Server*-nya dalam keadaan aktif. Bila kedua *web Server node level* 1 dan 2 dalam keadaan aktif dan *web Server node level* 3 dalam keadaan mati, maka *file* akan disimpan terlebih dahulu di *node level* 2 dan akan diteruskan ke *node level* 3 apabila *web Server node level* 3 dalam keadaan aktif

4.3.2 Pengiriman file



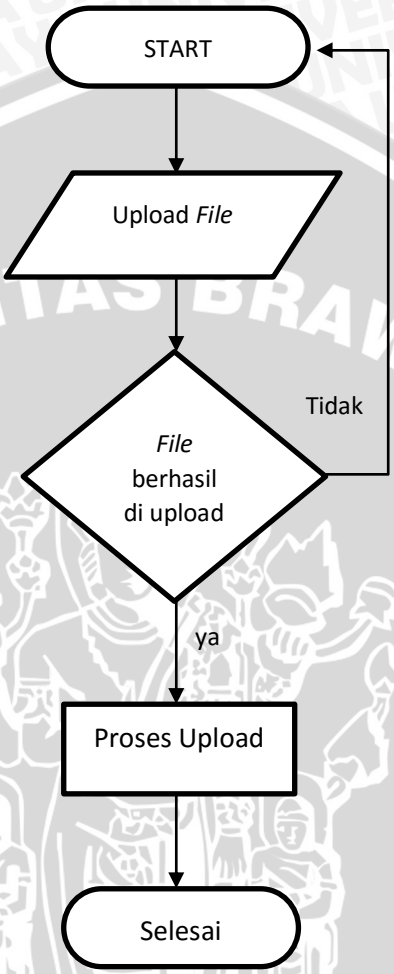
Gambar 4.3 Alur pengiriman *file* dari *node level* 1 menuju *node level* 3

Usai melakukan *upload file*, sistem secara otomatis melakukan proses ***ping web Server*** ke *node* selanjutnya. Proses *ping web Server* ini menggunakan bantuan *script Python*. Jika respon *HTTP* bernilai 200 maka *file* akan diteruskan ke *node* selanjutnya, apabila bernilai bukan 200, maka *file* disimpan terlebih dahulu. *File* disimpan pada *storage* khusus disetiap *node*.

4.4 Diagram

4.4.1 Flowchart upload file

Upload file adalah proses awal pada sistem dimana *user* meng*upload file* ke *node* utama. *User* meng*upload file* dengan cara mengakses *node* utama.



Gambar 4.4 Flowchart *Upload File*.

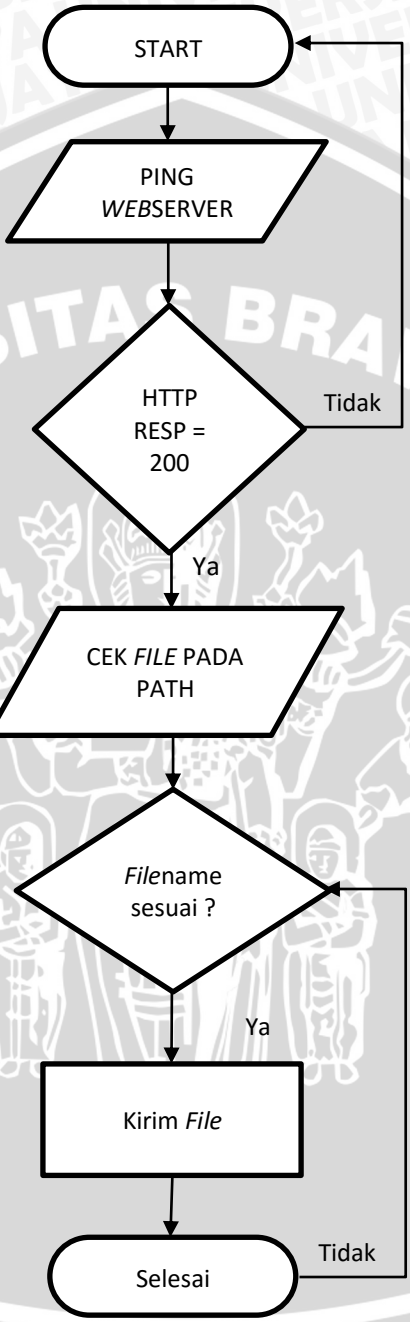
Penjelasan :

- *User* mengakses *form upload* untuk melakukan *upload file*
- *User* memilih *file* yang akan di*upload* beserta tujuan pengiriman *file*, apabila berhasil *upload* maka akan dilanjutkan proses *upload*, apabila tidak berhasil *upload* maka *user* kembali ke halaman *upload*

Apabila proses *upload* berhasil, maka *file* yang di*upload user* berhasil masuk ke *folder* khusus untuk menyimpan *file* yang di*upload*

4.4.2 Flowchart kirim file

Kirim *file* adalah proses dimana *file* dari *storage* pada *node* utama dikirim ke *node* selanjutnya berdasar header *IP* pada *file*name. Kirim *file* hanya akan berhasil apabila *web Server node* selanjutnya dalam keadaan aktif.



Gambar 4.5 Flowchart Kirim *file*

Penjelasan :

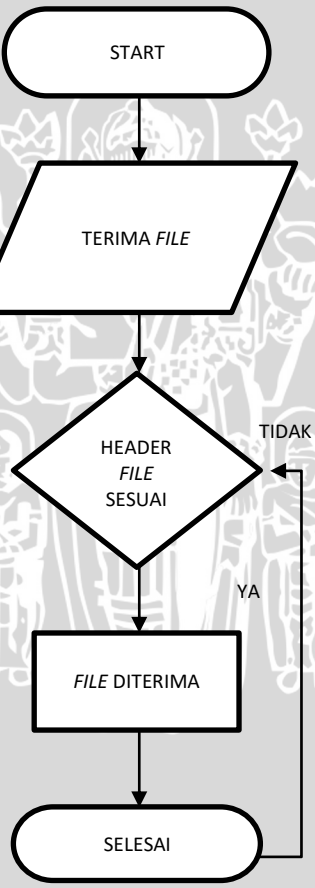
- *Node level* 1 / *node level* 2 melakukan pengecekan *web Server* dengan cara melakukan ping ke *web Server*

- *Node level* 1 / *node level* 2 menunggu respon *HTTP*, apabila respon bernilai 200 maka *web Server* tujuan yang di ping dalam keadaan aktif atau menyala. Apabila *web Server* tidak merespon dengan nilai 200 maka sistem kembali melakukan ping *web Server* ke *web Server* tujuan kembali.
- Apabila *web Server* tujuan aktif, sistem mengecek *file* pada *folder* penyimpanan *file* yang di *upload*.

Bila *filename* sesuai dengan tujuan alamat *IP* maka *file* akan dikirim, apabila tidak sesuai maka *file* tidak akan dikirim dan proses kirim selesai tanpa ada pengiriman *file*

4.4.3 Flowchart terima file

Terima *file* adalah sebuah proses akhir. Penerimaan *file* pada sebuah *node* berdasarkan dari header *IP* pada *file*name yang ter*upload* awal oleh *user*. Sebuah *node* tidak dapat menerima *file* apabila tidak terdapat header *IP* pada *filename*nya.



Gambar 4.6 Flowchart terima file

Penjelasan :

- Terima *file* akan terjadi apabila *web Server node* tujuan dan *node* pengirim dalam keadaan aktif.

Apabila header *filename* sesuai (alamat *IP* pada *filename* sesuai) maka *file* akan diterima oleh *node* penerima (*node* tujuan). Apabila tidak, maka proses terima *file* tidak terjadi, dan tidak ada *file* yang masuk ke penyimpanan *node* penerima.

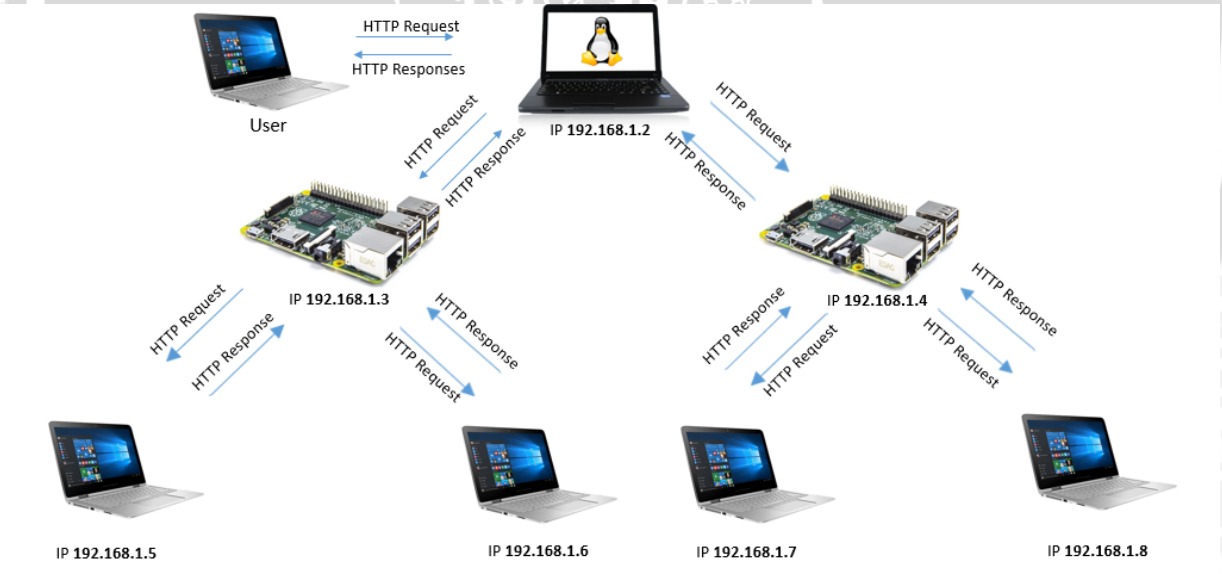
BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai perancangan dan hasil implementasi sistem yang telah di buat. Perancangan topologi jaringan, perancangan alur kerja sistem dan hasil implementasi yang dibuat.

5.1 Topologi jaringan

Topologi fisik jaringan yang dibuat menggunakan topologi *Star*. Topologi Star merupakan topologi yang umum digunakan pada jaringan yang ada saat ini. Topologi star membuat semua node dapat terhubung dengan node pusat dengan bantuan *hub*. Topologi secara logical menggunakan topologi *Tree*. Topologi tree digunakan untuk menjalankan proses *routing* metode *store and forward.*

Pada topologi ini terdapat 3 *level* jaringan. *Level* 1 adalah *node* utama, diakses langsung oleh *user*. *Level* 2 adalah *node* tengah (*Node* perantara) bertugas untuk menyimpan dan melanjutkan pengiriman *data* ke *node* selanjutnya. *Level* 3 adalah *node* akhir dimana *file* sampai pada tujuan akhir.

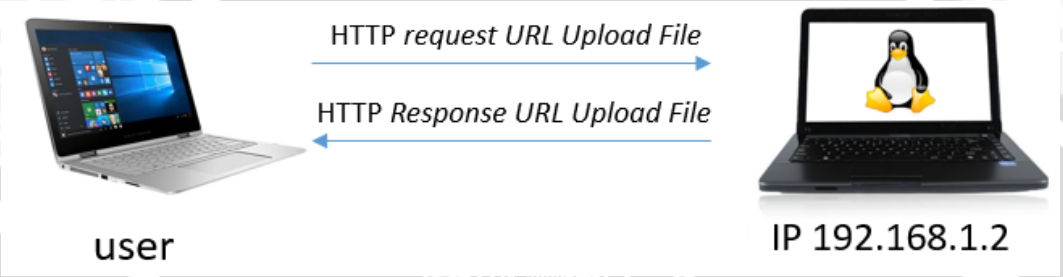


Gambar 5.1. Topologi Jaringan

Berdasar gambar diatas node pada *level* 1 berjumlah 1 buah. *Node* ini berupa sebuah *PC* dengan sistem operasi *Linux Lubuntu 16.04*. *Node* pada *level* 1 memiliki *IP* statis **"192.168.1.2"**. *Node* pada *level* 2 berjumlah 2 buah. *Node* ini berupa *Raspberry pi* yang di *install Linux Lubuntu 16.04*. Masing-masing *Raspberry pi* ini memiliki *IP* statis **"192.168.1.3"** untuk *node* 2 dan **"192.168.1.4"** untuk *node* 3. *Node* pada *level* 3 berjumlah 4 buah dan di*install* sistem operasi *Windows 10*. Tiap *node* memiliki *IP* statis, **"192.168.1.5"** untuk *node* 4, **"192.168.1.6"** untuk *node* 5, **"192.168.1.7"** untuk *node* 6 dan **"192.168.1.8"** untuk *node* 7.

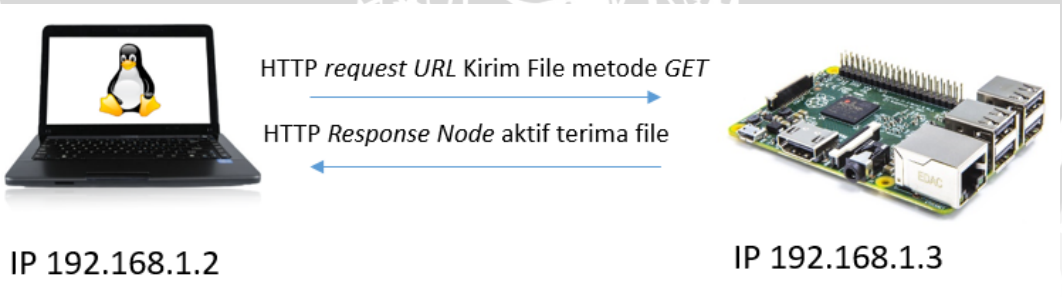
Routing pada arsitektur jaringan diatas adalah routing tipe statis. File yang teruplod di node level 1 akan dikirim ke node level 3 berdasarkan *routing* yang sudah dibuat, dan alur jalan pengiriman file sama seperti tanda panah pada gambar 5.1 Topologi jaringan. Sehingga, dalam proses pengiriman *file routing* tidak dapat berubah dan file pasti akan terkirim sesuai tujuan awal

Berdasar Gambar 5.1 diatas, mekanisme *Request* dan *respon* selalu terjadi antar *node*. Namun dalam kondisi jalannya sistem karena menggunakan metode *store and forward*, Selalu dilakukan *ping* otomatis antar *node* untuk memastikan bahwa node selanjutnya dalam keadaan aktif atau tidak aktif. Berikut adalah penjelasan mekanisme request – respon dalam sistem ini.



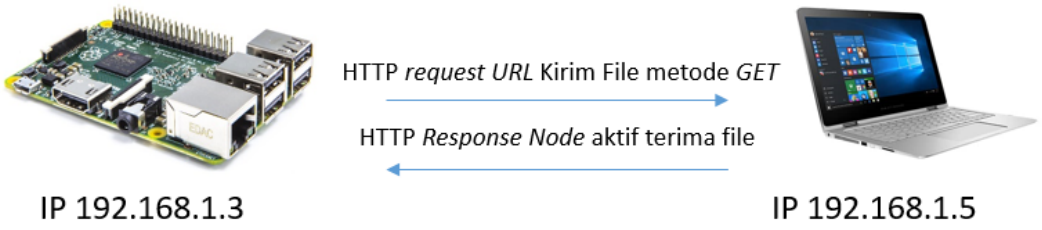
Gambar 5.2 Mekanisme Request Respon user ke Node level 1

Gambar 5.2 menjelaskan bahwa *User* melakukan *request* upload file ke node level 1. Metode HTTP Requestnya menggunakan GET. Apabila node level 1 dalam keadaan aktif, maka akan merespon dengan memberi konten upload file pada *user*. Apabila *Request – Response* terjadi, maka akan terjadi komunikasi antar 2 node ini, sehingga user dapat mengupload file kedalam sistem.



Gambar 5.3 HTTP Mekanisme *Request Respon* node level 1 menuju node level 2

Gambar 5.3 menjelaskan bahwa usai file milik *user* di *upload* ke *node* level 1, maka *node* level 1 akan melakukan komunikasi dengan *node* level 2 berdasarkan routing yang telah dibuat untuk sistem. *Node* level 1 melakukan request pengiriman file ke pada *Node* level 2. *Request* ini menggunakan metode *GET*, apabila *node* level 2 menerima *request* dari *node* level 1, maka *node* level 2 akan mengirimkan respon penerimaan file ke *node* level 1, kemudian file dikirimkan menuju *node* level 2.



Gambar 5.4 HTTP mekanisme Request respon dari node level 2 ke node level 3

Gambar 5.4 menjelaskan bahwa mekanisme request respon dari node level 2 ke node level 3 sama dengan request respon dari node level 1 ke node level 2. Hal ini dapat terjadi karena dalam sistem, node level 2 berdasar routing yang sudah dibuat meneruskan pengiriman file selanjutnya ke node level 3. Proses request respon hanya akan terjadi pada node yang sesuai dengan jalur routing yang sudah dibuat. Sehingga, tidak akan terjadi salah pengiriman file pada sistem.

5.2 Routing metode store and forward pada HTTP

Konfigurasi *routing* metode *store and forward* dibuat berdasarkan Topologi jaringan dengan *routing* Statis. Dengan menggunakan *file config.php,* di-*set Routing* antar *node* dengan menggunakan *IP* statis. *Node* pada *level* 1, memiliki 2 *child*. *Node* pada *level* 2 memiliki 1 *Parent* dan 2 *child*. *Node* pada *level* 3, memiliki 1 *parent*.

Tabel 5.1 Konfigurasi *Routing* Antar *Node*

Konfigurasi *Routing* Antar *Node* (config.*php*)

```
<?php
    $dataset=array(
            2 => array(
                "name" => "node1",
                "parent" => null
            ),
            3 => array(
                "name" => "node2",
                "parent" => 2
            ),
            4 => array(
                "name" => "node3",
                "parent" => 2
            ),
            5 => array(
                "name" => "node4",
                "parent" => 3
            ),
```

```
19:         6 => array(
20:                 "name" => "node5",
21:                 "parent" => 3
22:         ),
23:         7 => array(
24:                 "name" => "node6",
25:                 "parent" => 4
26:         ),
27:         8 => array(
28:                 "name" => "node7",
29:                 "parent" => 4
30:         )
31:     );
32: ?>
```

Penjelasan *Source code* :

Config.php adalah *script* untuk mengatur *Routing* dalam pengiriman *file*. Karena alamat *IP* yang digunakan adalah *IP* statis maka diambil adalah digit *IP* terakhir. *IP* statis yang digunakan adalah 192.168.1.xxx. *Node* awal memiliki alamt *IP* 192.168.1.2 dan *default gateway*-nya memiliki alamat *IP* 192.168.1.1.

Config.php berisi *script* untuk mendeklarasikan *parent and child* alamat tiap *IP* yang ada.

Baris 2 hingga baris 6 adalah contoh awal deklarasi dari *array* $*data*set yang berisi alamat *IP* 192.168.1.2 yang memiliki parent *null* dan dinamakan sebagai ***node 1*** . *Script* selanjutnya baris 7 hingga 10 berisi in*forma*si array bahwa *IP* dengan alamat *IP* 192.168.1.3 adalah *child* dari alamat *IP* 192.168.1.2 dan dinamakan sebagai *node* 2. Baris 11 hingga baris 31 berisi deklarasi tiap *node* yang berisi informasi parent and childnya sama seperti penjelasan baris 2 hingga baris ke 10. *Config*.*PHP* ini secara umum digambarkan pada gambar 5.1 Arsitektur jaringan, terdapat 3 *level* dalam arsitektur jaringannya. *Level* 1 adalah *Node* 1 (*node* utama). *Level* 2 adalah *node* 2 dan *node* 3 (*Node* kurir). *Level* 3 adalah *Node* 4, 5, 6 dan 7 (*Node* akhir/penerima).

Konfigurasi selanjutnya adalah konfigurasi tiap *node*. *Node* yang akan di konfigurasi adalah :

1. *Node level* 1
2. *Node level* 2
3. *Node level* 3

5.2.1 Konfigurasi node level 1

Node level 1 adalah *node* yang akan diakses oleh *user* terlebih dahulu. Pada *node* ini berisi fitur *upload* dan kirim *file*. Yang perlu dipersiapkan pada *node* ini adalah *Web Server*, *PHP* dan *cron*.

Web Server yang digunakan adalah milik Linux. Untuk menyiapkan *web Server* dan *PHP* ini bisa menggunakan command :

```
$ sudo apt-get install lamp-Server^
```

Kemudian dilanjutkan dengan meng*install CURL* untuk dapat mengirimkan *file* melewati protokol *HTTP*. Dilakukan dengan command sebagai berikut :

```
$ sudo apt-get install php-curl
```

Cron yang perlu disiapkan adalah *cron* untuk pengiriman *file*. *Cron* pada *node* utama diset untuk pengiriman *file* ke *node* 2 dan *node* 3 pada arsitektur/Topologi jaringan yang sudah dibuat. *Cron* akan memanggil *file* python untuk melakukan pengiriman. Berikut adalah code python untuk pengiriman *file* :

Tabel 5.2 *Script* python kirim.py

Contoh *script* python untuk pengiriman *file* ke *node* 2

```
1: $ #! python3
2: import subprocess
3: import time
4: import urllib.request
5: import os
6:
7: a= 1
8: hostname = "192.168.1.3"
9: resp= urllib.request.urlopen("HTTP://192.168.1.3").getcode()
10:
11: #and then check the response...
12: while a== 1:
13:    response = os.system("ping -n 1 " + hostname)
14:    if resp == 200:
15:          print(hostname, 'is up!')
16:
17:url=urllib.request.urlopen("HTTP://localhost/sfoh/kirim.PHP?IP=3")
18:
19:    a=0
20:    else:
21:          print(hostname, 'is down!')
```

```
22:         time.sleep(10)
```

Penjelasan *Source code* :

Source code diatas adalah *script* python yang digunakan untuk eksekusi *URL* : kirim.php dengan alamat "***http*://localhost/sfoh/kirim.*php*?*ip*=3**" pada tiap *node*. Pada *script* diatas dideklarasikan *hostname* pada baris 8 atau alamat tujuan dari pengiriman yaitu dengan alamat *IP* 192.168.1.3. *IP* 192.168.1.3 adalah alamat *IP node* 2, *node* 2 memiliki parent *node* 1 dengan alamat *IP* 192.168.1.2 dengan kata lain *script* python diatas berada pada *node* utama (*node level* 1). Pada baris 9 dideklarasikan *script pyhton urlopen* untuk eksekusi *file* kirim.php yang dieksekusi menuju alamat *IP* 192.168.1.3.

Baris 12 hingga baris 22 adalah proses eksekusi *script* python. Apabila kondisi a=1 maka *script* akan melakukan ping kepada *web Server* tujuan. Jika pengembalian respon *HTTP*-nya bernilai 200 maka *web Server* dalam keadaan nyala dan *script* kirim.php akan dieksekusi yang berarti *file* dengan *Routing* tujuan alamat *IP* 192.168.1.3 dikirim. Jika pengembalian respon *HTTP*-nya bernilai bukan 200 maka *web Server* tujuan dalam keadaan mati dan proses pengiriman tidak akan berjalan. Kesimpulannya adalah *script* python ini berfungsi pada sistem untuk eksekusi URL dengan alamat : "***http*://localhost/sfoh/kirim.*php*?*ip*=3**" dan memastikan *web Server* tujuan dalam keadaan aktif atau tidak.

5.2.2 Konfigurasi Node level 2 (node 2 dan node 3)

Node ini adalah *node level* 2 atau *node* tengah dalam arsitektur jaringan. Bertugas untuk menyimpan *data* atau *file* apabila ada *file* yang dikirimkan melalui *node* utama menuju *node level* 3. *Node level* 2 ini memerlukan *web Server, PHP, cron* serta yang paling utama adalah *file* penerima pengiriman dari *node* utama ke *node level* 2. Berikut *script* terima *file* di *node level* 2.

Tabel 5.3 *Script* terima *file*

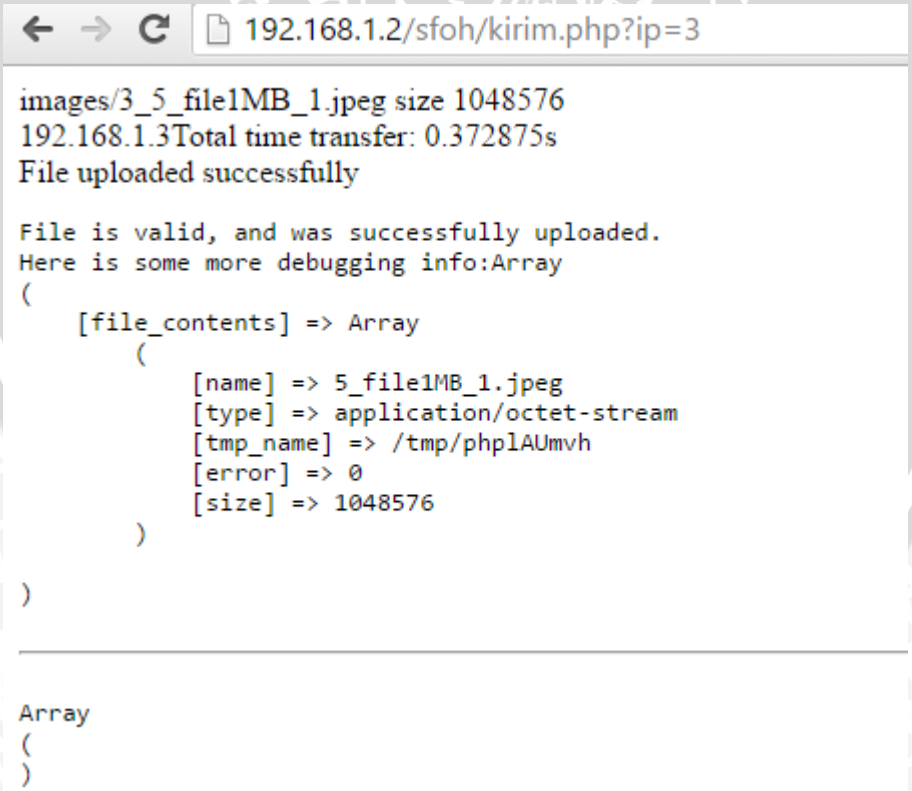
***Script* terima *file* pada *node level* 2**

```
1: <?PHP
2: $uploaddir = realpath('images')."/";
3:$uploadfile=$uploaddir.basename($_FILES['file_contents']['name
']);
4: echo '<pre>';
5:    if
(move_uploaded_file($_FILES['file_contents']['tmp_name'],
$uploadfile)) {
6:        echo "File is valid, and was successfully uploaded.\n";
7:    } else {
8:        echo "Possible file upload attACK!\n";
9:    }
```

```
10:    echo 'Here is some more debugging info:';
11:    print_r($_FILES);
12:    echo "\n<hr />\n";
13:    print_r($_POST);
14: print "</pr" . "e>\n";
15: ?>
```

Penjelasan Source code:

Script terima *file* ini adalah *script* yang dimiliki tiap *node* yang akan menerima *file*. *Script* ini berfungsi untuk menerima hasil pengiriman *file* yang dikirimkan oleh *node* pengirim. Pada baris 2 dilakukan deklarasi bahwa *file* yang diterima akan masuk kedalam *folder* images. Baris 3 menunjukkan proses penerimaan *file* yang dikirimkan. Baris 5 hingga baris ke 14 adalah statement if else yang menyatakan apabila *file* berhasil diterima dan disimpan pada *folder* images maka *file* dinyatakan sukses ter*upload* dan akan muncul beberapa info tentang *file* yang dinyatakan pada baris 10 hingga 13. Apabila *file* tidak berhasil terkirim maka hanya akan muncul pemberitahuan pada baris 8 dan baris 11 hingga 13 tidak akan muncul. Pesan berhasil *upload* dan tidak berhasil *upload* hanya akan muncul apabila *file* kirim.php dikirim secara manual



Gambar 5.5 *File* yang berhasil dikirim ke alamat *IP* tujuan 192.168.1.3

Gambar diatas menunjukkan ada beberapa *file* yang berhasil dikirimkan ke alamat *IP* 192.168.1.3. *File* yang dkirimkan akan menunjukkan in*form*asi yang ditunjukkan pada baris 10 hingga baris ke 13.

Node level 2 memiliki setting *cron* serta *web Server* yang sama dengan *Node* utama. *Cron* pada *level* 2 akan mengirimkan *file* ke *node* selanjutnya, sesuai dengan arsitektur jaringan yang telah di buat

5.2.3 Konfigurasi node level 3 (node 4, 5, 6 dan 7)

Node level 3 adalah *node* terakhir pada arsitektur jaringan yang dibuat. *Node* ini memerlukan *web Server* aktif dan *script* terima *file* untuk dapat menerima *file* yang dikirim. Berikut adalah *script* terima *file* pada *node* akhir

Tabel 5.4 *Script* terima *file* pada *node level* 3

***Script* terima *file* pada *node level* 3 (akhir)**

```
1: <?php
2: $uploaddir = realpath('images')."\\";
3:$uploadfile=$uploaddir.basename($_FILES['file_contents']['name
']);
4:echo '<pre>';
5:    if
(move_uploaded_file($_FILES['file_contents']['tmp_name'],
$uploadfile)) {
6:        echo "File is valid, and was successfully uploaded.\n";
7:    } else {
8:        echo "Possible file upload attack!\n";
9:    }
10:   echo 'Here is some more debugging info:';
11:   print_r($_FILES);
12:   echo "\n<hr />\n";
13:   print_r($_POST);
14: print "</pr" . "e>\n";
15: ?>
```

Penjelasan *source code* :

Secara umum *source code* terima *file* pada *node* 3 tidak jauh berbeda pada *node level* 2. Pada *node level* 3 yang berada di Windows, maka yang perlu dirubah hanya pada baris 2. Baris 2 hanya perlu mengganti "/" dengan "\\" agar dapat terbaca di Windows

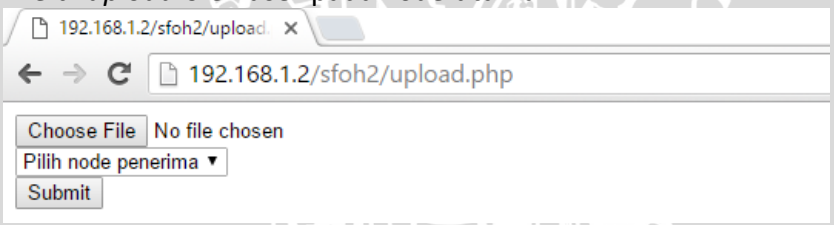
5.3 Implementasi metode *store and forward* pada HTTP

Implementasi metode *store and forward* pada *HTTP* dilakukan menggunakan minimal 3 *node* pada 3 *level* yang berbeda. Perangkat yang digunakan adalah PC Linux (*node level* 1 dan *node level* 3), *Raspberry pi* (*node level* 2). *IP* setiap *node* adalah statis dan sudah di konfigurasi sejak awal. Setiap *node* wajib memiliki *web Server*, *PHP*, *CURL* dan *cron* pada *node level* 1 dan 2. Setiap *node* perlu di lakukan konfigurasi dan *Routing* agar sistem berjalan dengan baik.

Script sistem ini dibuat menggunakan bahasa pemrograman *PHP*. Untuk pengiriman *file*nya dibantu dengan menggunakan bahasa pemrograman python dan *PHP*. Versi *PHP* yang digunakan adalah versi *PHP* 7.0 ke atas. Sistem ini memiliki fungsi *upload file* untuk *user*. *User* dapat melakukan *upload file* melewati *node* utama. *File* yang ter*upload* akan disimpan di *node* utama. *File* akan dikirimkan apabila *node* selanjutnya aktif. *Node* dikatakan aktif dan dapat menerima *file* yang dikirim apabila *web Server node* penerima aktif. Metode *store and forward* pada sistem ini bergantung pada proses pengiriman *file*. *Cron* di gunakan untuk melakukan otomatisasi pengiriman *file*, sehingga *user* tidak perlu melakukan pengiriman *file* secara manual. *User* juga tidak perlu menerima *file* secara manual, karena sistem melakukan otomatisasi penerimaan *file* yang dikirim.

Secara detail alur kerja sistem ini adalah sebagai berikut :

1. *File* di *upload* oleh *user* pada *node* utama



Gambar 5.6 *Form Upload*

2. *File* di simpan oleh sistem di *node* utama, *file* baru akan dikirimkan apabila *web Server node* selanjutnya aktif.

Tabel 5.5 potongan *script* python kirim.py

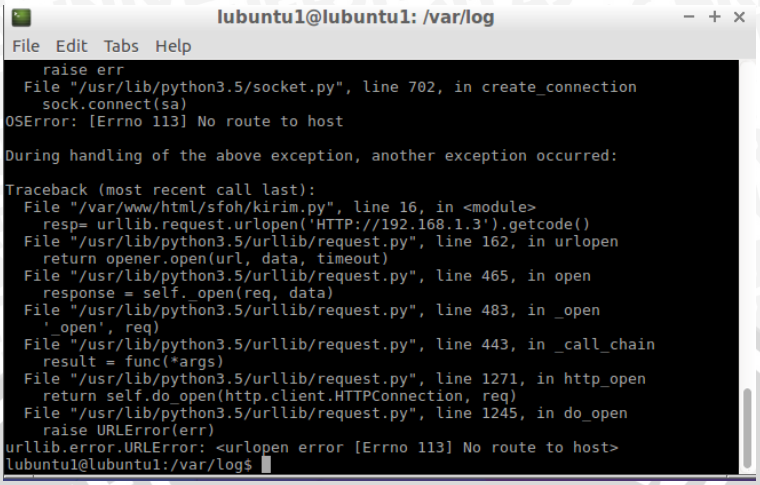
Potongan *script* python untuk ping *web Server*

```
1: while a== 1:
2:     response = os.system("ping -n 1 " + hostname)
3:     if resp == 200:
4:         print(hostname, 'is up!')
```

Penjelasan *source code* :

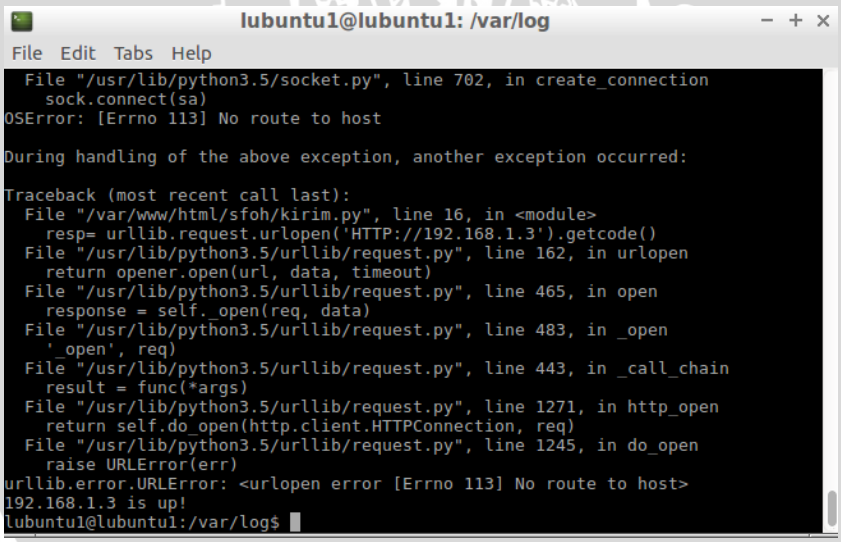
Pada kode diatas, apabila respon ping mendapat pengembalian nilai *HTTP* **"200"** maka *web Server* tujuan dalam keadaan hidup dan *file* baru dapat dikirim. Apabila pengembalian nilai *HTTP* bukan "200" atau tidak ada

pengembalian maka *web Server node* selanjutnya mati dan *file* di *store* pada *node* dimana *file* berada.



Gambar 5.7 *Web Server node* selanjutnya mati

Web Server node selanjutnya dalam keadaan mati sehingga memunculkan pesan *error* seperti gambar diatas. Pesan *error* diatas dihasilkan dari *script* python yang dieksekusi oleh *cron*



Gambar 5.8 *Web Server* tujuan dalam keadaan aktif/hidup

Pada gambar diatas *web Server* tujuan aktif. Hal ini ditunjukkan terdapat in*forma*si dibagian bawah bertuliskan : **"192.168.1.3 is up".** Apabila *web Server* tujuan dalam keadaan aktif maka *file* kirim.php dapat dieksekusi dan pengiriman *file* berjalan. *File* kirim.php dieksekusi oleh *script* python kirim.py

3. *File* dikirim apabila *web Server* aktif. Berikut adalah list *file* yang berada pada *node level* 1

Gambar 5.9 Daftar *File* pada *node level* 1

Berikut adalah *script* pengiriman *file* (kirim.php) :

Tabel 5.6 *script* kirim.php

***Script* kirim.php**

```
1: <?php
2: function getCURLValue($filename, $contentType, $postname){
3:     if (function_exists('CURL_file_create')) {
4:             return  CURL_file_create($filename,  $contentType,
$postname);
5:     }
6:     $value = "@{$filename};filename=" . $postname;
7:     if ($contentType) {
8:         $value .= ';type=' . $contentType;
9:     }
10:     return $value;
11: }
12: if (isset($_GET['IP'])){
13: $IP=$_GET['IP'];
14:    echo $IP;
15:    $imageSearch=glob('images/'.$IP.'*');
16:    if (!empty($imageSearch)){
17:          foreach ($imageSearch as $searchIP){
18:                $filepath = realpath($searchIP);
```

```
19:                $IPfull="192.168.1.".$IP;
20:                $url = "HTTP://".$IPfull."/sfoh/terima.PHP";
21:                $mimetype = mime_content_type($filepath); //get
file mime type
22:                $filename=  basename($searchIP);  //take  only
file name
23:                $pos=  strpos($filename,'_')+1;  //take  first
position of '_' in filename
24:                $length=   strlen($filename)-$pos;   //measure
length of string minus position of '_'
25:                $postname=     substr($filename,$pos,$length);
//get postname for posting to CURL
26:                $cfile                                          =
getCURLValue($filepath,$mimetype,$postname);   //convert file to
CURL object
27:                $filedata =  array('file_contents' => $cfile);
28:                $CURL = CURL_init();
29:                      CURL_setopt($CURL, CURLOPT_URL, $url);
30:                      CURL_setopt($CURL,
CURLOPT_HTTPHEADER,array('User-Agent: Opera/9.80 (Windows NT 6.2;
Win64;     x64)     Presto/2.12.388     Version/12.15','Referer:
HTTP://someaddress.tld','Content-Type:          multIPart/form-
data','Expect:'));
31:                      CURL_setopt($CURL,  CURLINFO_HEADER_OUT,
true);
32:                      CURL_setopt($CURL,
CURLOPT_SSL_VERIFYPEER, false); // stop verifying certificate
33:                      CURL_setopt($CURL,
CURLOPT_RETURNTRANSFER, true);
34:                      CURL_setopt($CURL,  CURLOPT_POST,  true);
// enable posting
35:                CURL_setopt($CURL,        CURLOPT_POSTFIELDS,
$filedata); // post images
36:                      CURL_setopt($CURL,
CURLOPT_FOLLOWLOCATION, true); // if any redirection after upload
37:          $result=CURL_exec($CURL);
38:          $header_info                                          =
CURL_getinfo($CURL,CURLINFO_HEADER_OUT);
39:                $header_size       =       CURL_getinfo($CURL,
CURLINFO_HEADER_SIZE);
40:                $header = substr($result, 0, $header_size);
41:                $body = substr($result, $header_size);
42:                if(!CURL_errno($CURL)) {
43:                      $info = CURL_getinfo($CURL);
```

```
44:                     if   ($info['HTTP_code']   ==   200   &&
unlink($searchIP))
45:                         echo "File uploaded successfully";
46:                 }
47:                 else{
48:                         echo CURL_error($CURL);
49:                 }
50:                 echo $result; //print transfer result
51:                 CURL_close($CURL);
52:         }
53:    }else echo "File not found!!!";
54: }else echo "IP search error!!!";
55: ?>
```

Script kirim.*PHP* berjalan apabila ping *web Server* berhasil dilakukan.

Penjelasan *source code* :

Script diatas adalah *script* untuk mengirimkan *file* dari *node* saat ini ke *node* selanjutnya. *Script* ini dieksekusi dengan bantuan *script* python kirim.py.

Penjelasan *source code*

Tabel 5.7 Potongan *script* kirim.php(1)

Potongan *script* kirim.php

```
2: function getCURLValue($filename, $contentType, $postname){
3:     if (function_exists('CURL_file_create')) {
4:             return  CURL_file_create($filename,  $contentType,
$postname);
5:     }
6:     $value = "@{$filename};filename=" . $postname;
7:     if ($contentType) {
8:         $value .= ';type=' . $contentType;
9:     }
10:     return $value;
11: }
```

Baris 2 hingga baris 11 adalah pembuatan fungsi untuk mengecek kompatibilitas fungsi *CURL_file_Create* apakah bisa dipakai pada *PHP* versi 5.5 keatas atau 7. Apabila menjalankan *file PHP* 5.5 keatas maka yang eksekusi adalah baris ke 2 sampai baris ke 4. Apabila versi *PHP* 5.5 kebawah maka yang dieksekusi adalah baris ke 6 hingga 11.

Tabel 5.8 Potongan *script* kirim.php(2)

Potongan *script* kirim.php

```
12: if (isset($_GET['IP'])){
13: $IP=$_GET['IP'];
14:   echo $IP;
15:   $imageSearch=glob('images/'.$IP.'*');
16:   if (!empty($imageSearch)){
17:        foreach ($imageSearch as $searchIP){
18:             $filepath = realpath($searchIP);
19:             $IPfull="192.168.1.".$IP;
20:             $url = "HTTP://".$IPfull."/sfoh/terima.PHP";
21:             $mimetype = mime_content_type($filepath); //get
file mime type
22:             $filename=  basename($searchIP);  //take  only
file name
23:             $pos=  strpos($filename,'_')+1;  //take  first
position of '_' in filename
24:             $length=  strlen($filename)-$pos;   //measure
length of string minus position of '_'
25:             $postname=    substr($filename,$pos,$length);
//get postname for posting to CURL
26:             $cfile                                    =
getCURLValue($filepath,$mimetype,$postname);   //convert file to
CURL object
27:             $filedata =  array('file_contents' => $cfile);
28:             $CURL = CURL_init();
29:                  CURL_setopt($CURL, CURLOPT_URL, $url);
30:                  CURL_setopt($CURL,
CURLOPT_HTTPHEADER,array('User-Agent: Opera/9.80 (Windows NT 6.2;
Win64;     x64)     Presto/2.12.388     Version/12.15','Referer:
HTTP://someaddress.tld','Content-Type:          multIPart/form-
data','Expect:'));
31:                  CURL_setopt($CURL,  CURLINFO_HEADER_OUT,
true);
32:                  CURL_setopt($CURL,
CURLOPT_SSL_VERIFYPEER, false); // stop verifying certificate
33:                  CURL_setopt($CURL,
CURLOPT_RETURNTRANSFER, true);
34:                  CURL_setopt($CURL,  CURLOPT_POST,  true);
// enable posting
35:              CURL_setopt($CURL,        CURLOPT_POSTFIELDS,
$filedata); // post images
36:                  CURL_setopt($CURL,
CURLOPT_FOLLOWLOCATION, true); // if any redirection after upload
37:        $result=CURL_exec($CURL);
```

```
38:         $header_info                                    =
CURL_getinfo($CURL,CURLINFO_HEADER_OUT);
39:                 $header_size        =       CURL_getinfo($CURL,
CURLINFO_HEADER_SIZE);
40:                 $header = substr($result, 0, $header_size);
41:                 $body = substr($result, $header_size);
42:                 if(!CURL_errno($CURL)) {
43:                         $info = CURL_getinfo($CURL);
44:                         if    ($info['HTTP_code']    ==    200    &&
unlink($searchIP))
45:                                 echo "File uploaded successfully";
46:                 }
47:                 else{
48:                         echo CURL_error($CURL);
49:                 }
50:                 echo $result; //print transfer result
51:                 CURL_close($CURL);
52:         }
```

pengecekan *IP* tujuan dideklarasikan pada baris ke 12 dan ke 13, kemudian sistem akan melakukan pengecekan pada *folder* ***images*** (alamat *images* pada *node* diberikan oleh sistem pada fungsi ***realpath)*** ini dideklarasikan pada baris ke 18**.** Namun sistem memulai pencarian atau pengecekan *filename* yang mengandung alamat *IP* tujuan pada baris ke 12. Kemudian dilanjut dengan proses mengeksekusi tiap *file* yang ada dimulai dari baris ke 16 hingga baris ke 52, pemrosesan *file* ini didapat pada baris ke 12. Usai pemrosesan *file* berhasil, maka *file* yang ada akan dikirimkan melalui *CURL*. *CURL* di deklarasikan dengan fungsi *CURL*l_init() yang perintahnya berada pada baris ke 28. *CURL* memerlukan beberapa opsi yang d*IP*erlukan dalam proses pengiriman *file*, opsi itu dideklarasikan mulai baris ke 29 hingga baris ke 36. Baris 42 hingga 49 adalah statement proses apakah *file* berhasil dikirim atau tidak. Baris 44 menjelaskan apabila respon *HTTP* bernilai 200 maka *file* berhasil dikirim dan *file* yang ada di *node* tempat kirim.*PHP* dieksekusi di hapus.

Gambar 5.10 *Script* kirim.php (dieksekusi manual) berjalan untuk melakukan pengiriman *file*

4. *File* diterima oleh *node* selanjutnya

Gambar 5.11 List *File* yang terkirim

File yang terkirim adalah *file* dengan kode ”11” didepan nama *file* pada gambar diatas. Karena *file* adalah milik *IP* 192.168.1.5 maka *file* akan dilanjutkan apabila *IP* 192.168.1.5 *web Server*-nya dalam keadaan hidup.

5. *Node level* 2 mengecek *IP* tujuan pada daftar *file* yang diterimanya

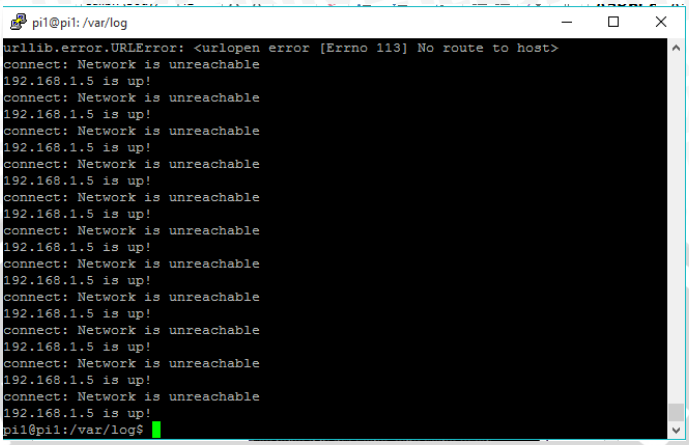
Node level 2 memiliki 6 item dengan list masing-masing tujuan sebagai berikut :

- *file* tujuan *IP* 192.168.1.5
- 1 *File* tujuan *IP* 192.168.1.2
- 1 *File* tujuan *IP* 192.168.1.6

Berdasar *Routing* jaringan yang dibuat, maka *node level* 2 dengan *IP* 192.168.1.3 hanya mengirimkan *file* ke *node level* 3 dengan *IP* 192.168.1.5 dan

192.168.1.6. Maka *Node level* 2 akan mencoba melakukan ping ke 2 *node level* bawah yang berada pada jaringannya.

Node level 2 dengan bantuan *script* Python melakukan proses ping *web Server* ke *node level* 3 tujuannya.



Gambar 5.12 *Web Server* tujuan *node level* 3 dengan *IP* 192.168.1.5 aktif

Karena yang aktif hanya *IP* 192.168.1.5 *node level* 2 mengirimkan *file* ke *node* akhir sebanyak 4 *file*.

Gambar 5.13 Daftar *file* di *node level* 2

Pada gambar diatas, pada *node level* 2 *file* yang memiliki kode ”11” sudah tidak ada, yang berarti *file* berhasil di teruskan ke *node* akhir. Hal ini terjadi karena *web Server node* akhir dalam kondisi hidup

Gambar 5.14 Daftar *file* pada *node* akhir

Pada gambar diatas *File* dengan kode "11" berhasil terkirim ke *node* tujuan akhir. Hal ini berarti tugas dari *node level* 1, *level* 2, *node level* 3 serta *Routing* bekerja secara baik.

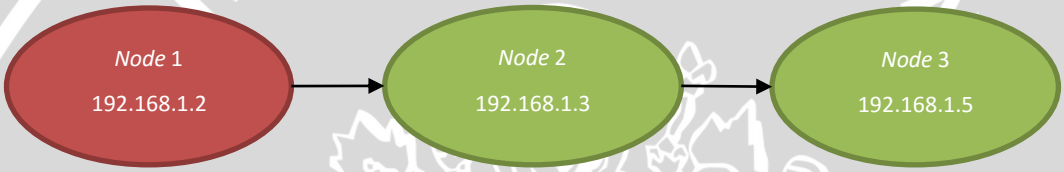
BAB 6 PENGUJIAN

Pada bab ini akan dijelaskan mengenai hasil pengujian dari implementasi yang sudah dibuat. Pengujian ini dilakukan untuk mengetahui kinerja pengiriman *file* dengan menggunakan metode *store and forward* pada *HTTP*. Serta untuk mengetahui seberapa lama waktu proses pengiriman *file*. Pengujian akan dilakukan dengan menggunakan beberapa skenario:

6.1 Pengujian metode store and forward

Pengujian kali ini akan dilakukan dengan menggunakan 4 skenario. Skenario ini dibuat untuk memastikan bahwa metode store and forward dapat berjalan atau tidak.

6.1.1 *Node level* 1 mati



Gambar 6.1 Skenario 1

Skenario pertama untuk uji coba metode *store and forward* adalah dengan mematikan *node level* 1. Skenario ini untuk mencari tahu apakah *file* yang ter*upload* akan terkirim apabila *node* 1 dalam keadaan mati.

Pada Skenario ini *web Server node level* 1 dimatikan, namun pc *level* 1 tetap dalam keadaan hidup. *Web Server* menjadi alat untuk mengetahui suatu *node* dalam keadaan mati atau tidak.

Proses Pengujian :

Berikut adalah konfigurasi tiap *node* pada skenario pertama kali ini :

Karena *Node level* 1 dalam keadaan mati, saya sudah melakukan *upload file* terlebih dahulu dari pc saya ke *node level* 1 (*IP* = 192.168.1.2).

Index of /sfoh/images

Name	Last modified	Size	Description
Parent Directory		-	
3_5_11Ant.Man.2015_HDRIP720pC.srt	2016-07-28 05:37	151K	
3_5_11DTN.pdf	2016-07-28 05:37	4.1M	
3_5_11KTP Mas GigihC.jpg	2016-07-28 05:40	4.6M	
3_5_11KatyPerryUnconditionallyB.flac	2016-07-28 05:39	21M	
3_6_template_skripsi_v1.1C.docx	2016-07-28 05:41	311K	
4_7_DJ Snake - Middle (Audio) ft. Bipolar Sunshine.mp3	2016-07-28 05:44	3.4M	
4_7_test.php	2016-07-28 05:43	1.9K	
4_7_th5SF9I4I0.jpg	2016-07-28 05:43	2.4K	

Apache/2.4.18 (Ubuntu) Server at 192.168.1.2 Port 80

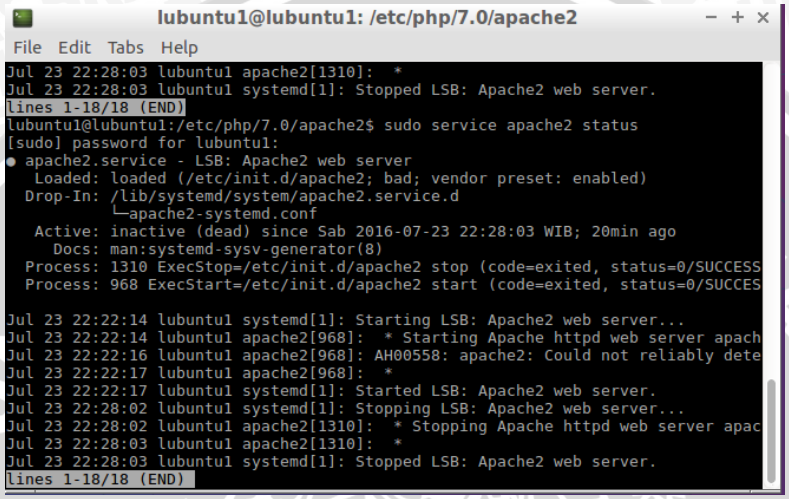
Gambar 6.2 Daftar *file* yang tersimpan di *node level* 1

Berdasar daftar diatas terdapat 4 *file* yang dikirim ke *node level* 3 dengan tujuan *IP* 192.168.1.5. Sebelum menuju *Node level* 3, *file* akan dikirim melalui *node level* 2 dengan *IP* 192.168.1.3.

Mengecek kondisi Tiap *node*

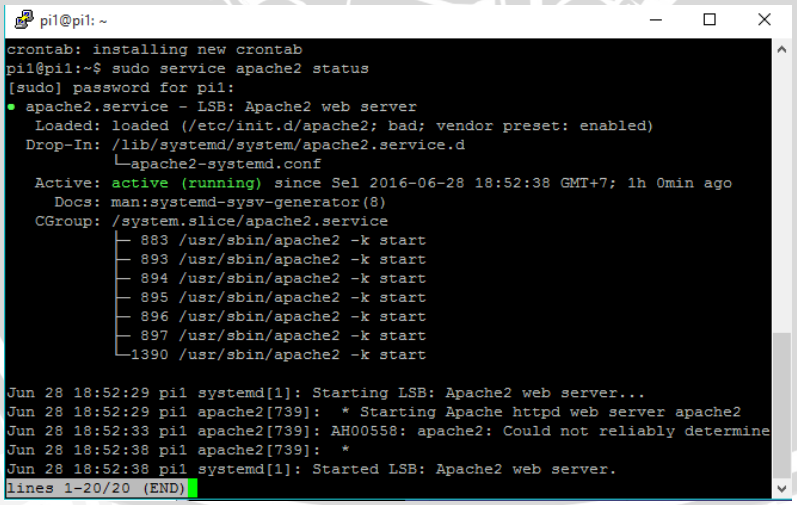
Kondisi tiap *node* berdasar skenario pengujian 1 :

Node level 1 : *Web Server* Mati (*IP* = 192.168.1.2)



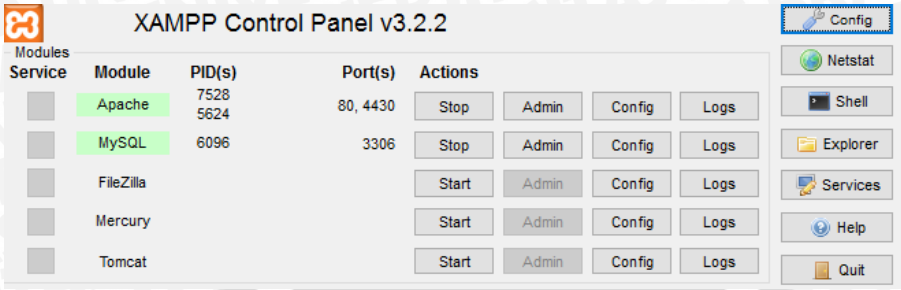
Gambar 6.3 *web Server Node level* 1 Mati

Node level 2 : *Web Server* Hidup (*IP* = 192.168.1.3)



Gambar 6.4 *Web Server Node Level* 2 Hidup

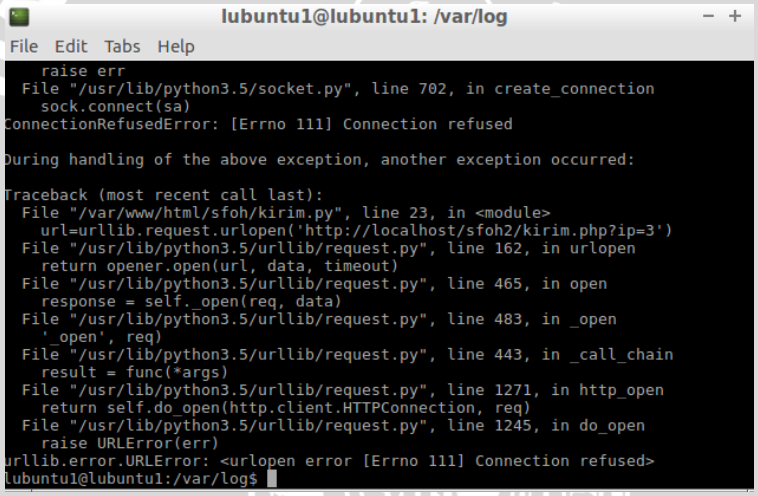
Node level 3 : *Web Server* Hidup (*IP* = 192.168.1.5)



Gambar 6.5 *Web Server node level* 3 hidup

Hasil dari pengujian adalah :

File di *node level* 1 tidak akan terkirim karena *web Server node level* 1 dengan *IP* 192.168.1.2 dalam keadaan mati, sehingga *file* yang di*upload* tetap akan tersimpan pada *node level* 1.



Gambar 6.6 Log pengiriman *File* menuju *node level* 2

Akibat *web Server node level* 1 dalam keadaan mati, tidak ada *script file* yang dieksekusi sehingga pengiriman *file* tidak berjalan.

Gambar 6.7 Daftar *file* yang tersimpan di *node level* 1

Kesimpulan dari pengujian 1 adalah :

Apabila *web Server node* utama mati, *file* yang ter-*upload* tidak akan terkirim. Karena *node level* 1 tidak dapat melakukan eksekusi *script* untuk pengiriman *file*. *File* yang ter-*upload* akan tetap disimpan oleh *node level* 1. *File* akan terkirim ke *node* selanjutnya apabila *node level* 1 dan *node level* 2 *web Server*-nya dalam keadaan aktif.

6.1.2 *Node level* 2 mati

Gambar 6.8 Skenario 2

Skenario kedua untuk uji coba metode *store and forward* adalah dengan mematikan *node level* 2. Skenario ini untuk mencari tahu apakah *file* yang ter*upload* bisa terkirim menuju tujuan akhir.

Skenario pengujian dua ini untuk memastikan apakah *Routing* pada sistem berjalan atau tidak. *Routing* sistem pada pengujian dua ini adalah apabila *file* dikirim menuju *node level* 3 dengan *IP* 192.168.1.5 harus melewati *node level* 2 dengan *IP* 192.168.1.3.

Pengujian ini akan memastikan bahwa tidak akan ada *file* yang terkirim ke tujuan akhir ataupun ke *node Level* 2. *Node level* 2 PC dalam keadaan hidup namun *web Server* dalam keadaan mati.

Proses Pengujian :

Berikut adalah konfigurasi tiap *node* pada skenario pertama kali ini :

Index of /sfoh/images

Name	Last modified	Size	Description
Parent Directory		-	
3_5_11Ant.Man.2015_HDRIP720pC.srt	2016-07-28 05:37	151K	
3_5_11DTN.pdf	2016-07-28 05:37	4.1M	
3_5_11KTP Mas GigihC.jpg	2016-07-28 05:40	4.6M	
3_5_11KatyPerryUnconditionallyB.flac	2016-07-28 05:39	21M	
3_6_template_skripsi_v1.1C.docx	2016-07-28 05:41	311K	
4_7_DJ Snake - Middle (Audio) ft. Bipolar Sunshine.mp3	2016-07-28 05:44	3.4M	
4_7_test.php	2016-07-28 05:43	1.9K	
4_7_th5SF9I4I0.jpg	2016-07-28 05:43	2.4K	

Apache/2.4.18 (Ubuntu) Server at 192.168.1.2 Port 80

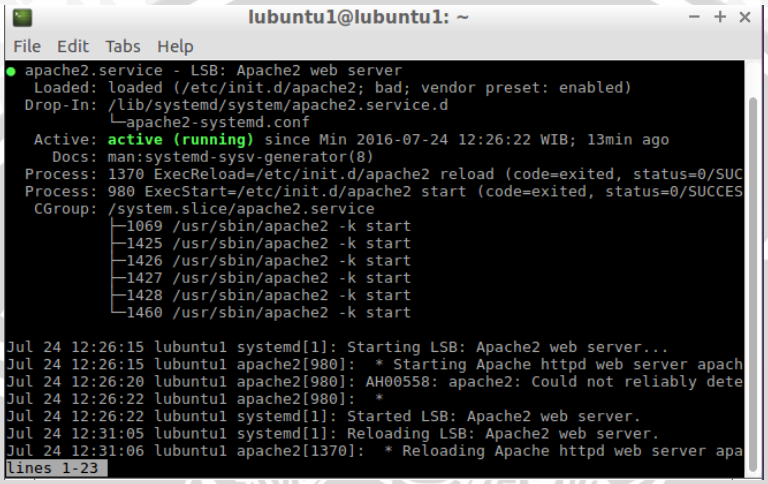
Gambar 6.9 Daftar *file* yang tersimpan di *node level* 1

Pada *node level* 1 kita ketahui masih terdapat 4 *file* dengan Tujuan *IP* 192.168.1.5. Selanjutnya kita perlu memastikan apakah kondisi *web Server* tiap *node* sebelum melakukan pengujian. Pada Pengujian kedua ini *web Server node level* 1 dalam keadaan aktif, *web Server node level* 2 dalam keadaan mati dan *web Server node level* 3 dalam keadaan hidup.

Mengecek kondisi tiap *node* :

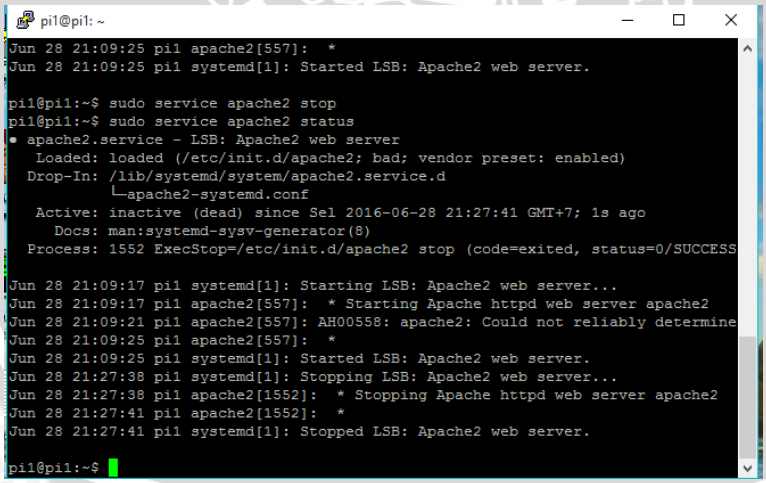
Kondisi Tiap *node* berdasar Skenario pengujian kedua adalah sebagai berikut :

Node Level 1 = *Web Server* Hidup (192.168.1.2)



Gambar 6.10 *Web Server Node level* 1 hidup

Node Level 2 = *Web Server* mati (*IP* = 192.168.1.3)



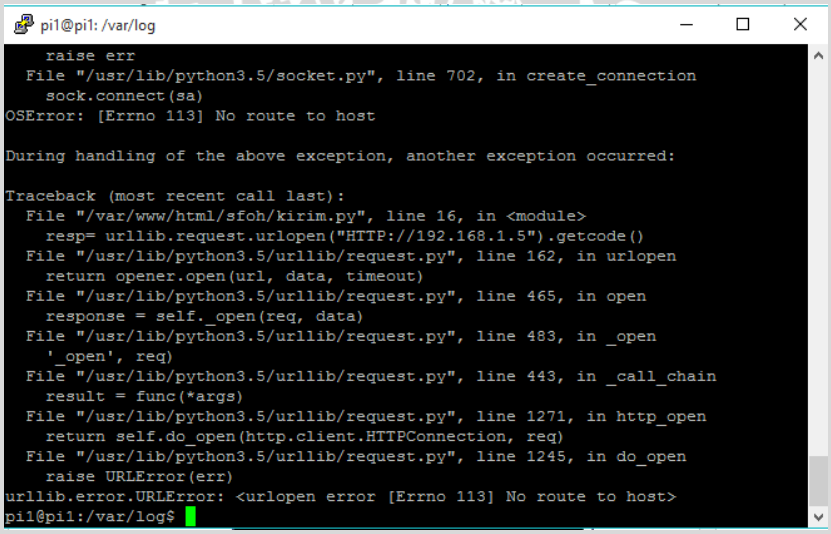
Gambar 6.11 *Web Server Node Level* 2 Mati

Node level 3 = *Web Server* hidup (*IP* = 192.168.1.5)

Gambar 6.12 *Web Server node level* 3 dalam keadaan hidup

Hasil dari pengujian adalah :

Proses pengiriman *file* yang sudah ter*upload* di *node level* 1 menuju *node* tujan akhir tidak terkirim. Hal ini disebabkan *node level* 2 dalam keadaan mati. Pada sistem *Routing* jaringan, semua *file* yang di*upload* dengan tujuan ke *IP* 192.168.1.5 akan dikirim melalui *node level* 2 dengan *IP* 192.168.1.3. Metode *Store and forward* dan *Routing* pada pengujian dua ini berhasil. *File* yang siap dikirim ke tujuan akhir tidak akan terkirim karena *node level* 2 pada (*IP* 192.168.1.3) dalam keadaan mati, sehingga *file* tetap disimpan pada *node level* 1 (*IP* 192.168.1.2).



Gambar 6.13 Log pengiriman tidak berhasil

Kesimpulan dari Pengujian 2 adalah :

Apabila *node* selanjutnya dalam keadaan mati, maka *file* yang akan dikirim tetap tersimpan di *node* dimana *file* berada. *File* hanya akan dapat terkirim apabila ke dua *node* terhubung. Sebuah *node* dinyatakan dalam keadaan hidup apabila *web Server* dalam keadaan hidup juga. Metode *store and forward* berhasil di terapkan pada pengujian ke dua ini.

6.1.3 *Node level* 3 mati

Gambar 6.14 Skenario 3

Skenario Ketiga untuk uji coba metode *store and forward* adalah dengan mematikan *node level* 3. Skenario ini dilakukan untuk mencari tahu apakah *file* yang ter*upload* bisa terkirim menuju *node* akhir.

Skenario pengujian tiga ini juga memastikan letak *file* apabila *node level* 1 dan *node level* 2 hidup sedangkan *node level* 3 mati. Dari dua pengujian sebelumnya kita ketahui bahwa *node level* 1 dan *node level* 2 tidak pernah terhubung. Pengujian sebelumnya menghasilkan in*forma*si *file* tetap tersimpan di *node level* 1 (*IP* = 192.168.1.2) karena tidak adanya pengiriman yang berhasil dilakukan.

Pengujian ketiga ini memastikan apabila *node level* 3 dalam keadaan mati *file* yang dikirim akan di store pada *node level* 2 (*IP* = 192.168.1.3) dan kembali di *forward* apabila *node level* 3 (*IP* = 192.168.1.5 atau *IP* = 192.168.1.6) dalam keadaan hidup. Uji coba tetap dilakukan untuk pengiriman *file* menuju *IP* = 192.168.1.5

Proses Pengujian :

Berikut adalah konfigurasi tiap *node* pada skenario pertama kali ini :

Index of /sfoh/images

Name	Last modified	Size	Description
Parent Directory		-	
3_5_11Ant.Man.2015_HDRIP720pC.srt	2016-07-28 05:37	151K	
3_5_11DTN.pdf	2016-07-28 05:37	4.1M	
3_5_11KTP Mas GigihC.jpg	2016-07-28 05:40	4.6M	
3_5_11KatyPerryUnconditionallyB.flac	2016-07-28 05:39	21M	
3_6_template_skripsi_v1.1C.docx	2016-07-28 05:41	311K	
4_7_DJ Snake - Middle (Audio) ft. Bipolar Sunshine.mp3	2016-07-28 05:44	3.4M	
4_7_test.php	2016-07-28 05:43	1.9K	
4_7_th5SF9I4I0.jpg	2016-07-28 05:43	2.4K	

Apache/2.4.18 (Ubuntu) Server at 192.168.1.2 Port 80

Gambar 6.15 Daftar *file* yang tersimpan di *node level* 1

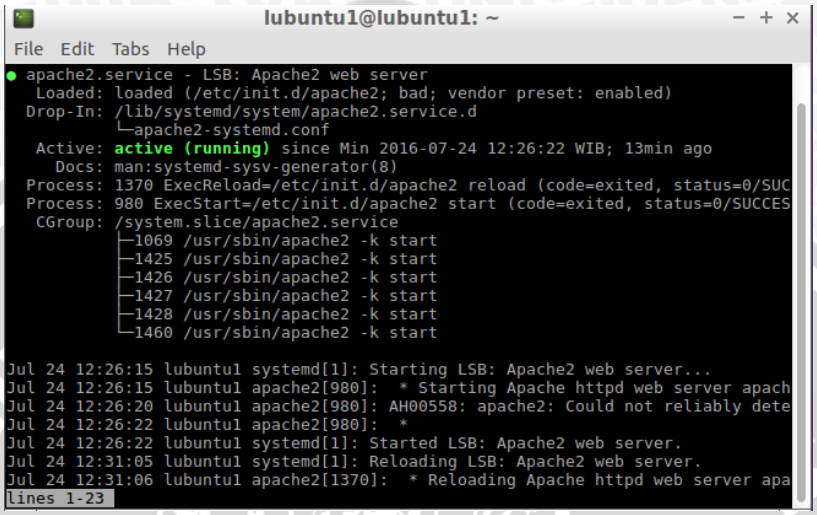
Pada *node level* 1 kita ketahui masih terdapat 4 *file* dengan Tujuan *IP* 192.168.1.5. Selanjutnya kita perlu memastikan apakah kondisi *web Server* tiap *node* sebelum melakukan pengujian. Pada Pengujian ketiga ini *web Server node*

level 1 dalam keadaan aktif, *web Server node level* 2 dalam keadaan aktif dan *web Server node level* 3 dalam keadaan mati

Mengecek kondisi tiap *node* :

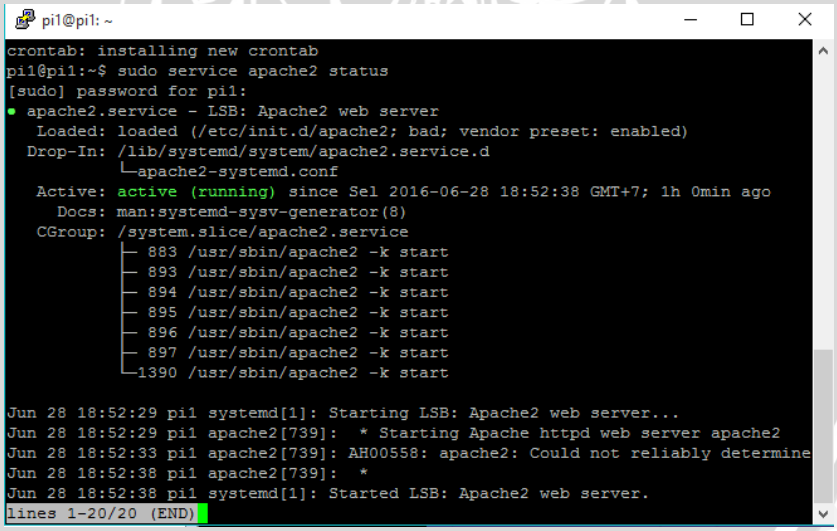
Kondisi Tiap *node* berdasar Skenario pengujian kedua :

Node Level 1 = *Web Server* Hidup (192.168.1.2)



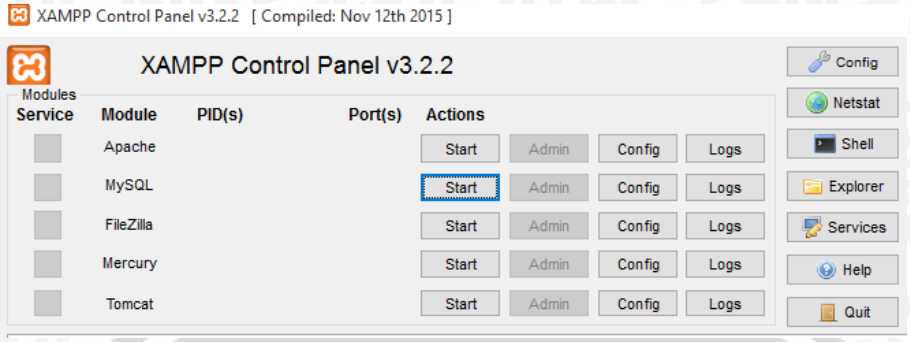
Gambar 6.16 *Web Server node level* 1 hidup

Node Level 2 = *Web Server* hidup (*IP* = 192.168.1.3)



Gambar 6.17 *Web Server node Level* 2 Hidup

Node level 3 = *Web Server* Mati (*IP* = 192.168.1.5)

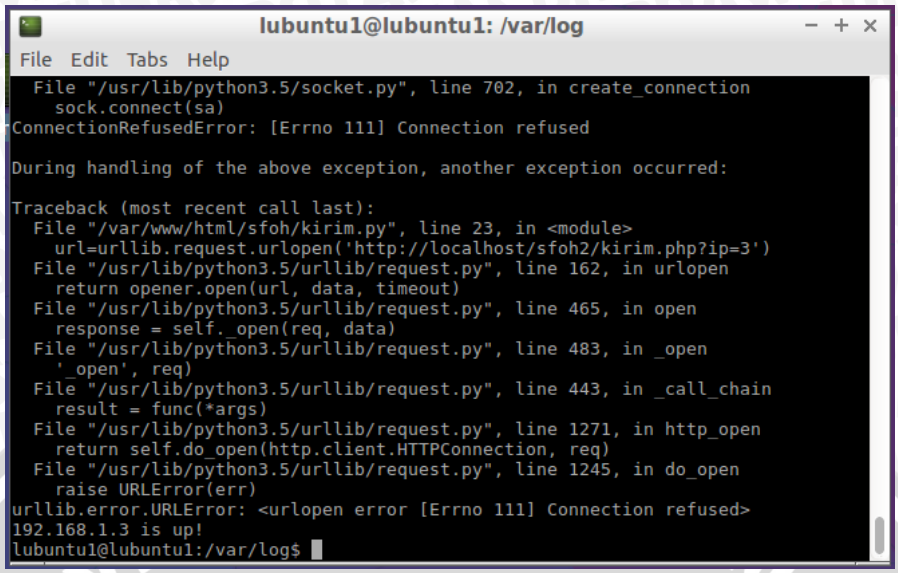


Gambar 6.18 *Web Server node level* 3 dalam keadaan mati

Hasil Pengujian adalah :

Karena *web Server* pada *node level* 1 (*IP* = 192.168.1) dan 2 (*IP* = 192.168.1.3) dalam keadaan hidup maka *file* yang dikirimkan menuju *node* 192.168.1.5 tersimpan dahulu di *node level* 2 (*IP* = 192.168.1.3). *File* akan dilanjut dikirim ke *node level* 3 (*IP* = 192.168.1.5) apabila *node level* 3 dalam keadaan hidup (*Web*sever Hidup). Metode *Store and forward* pada pengujian 3 ini berhasil.

Gambar 6.19 List *file* yang tersimpan di *node level* 1

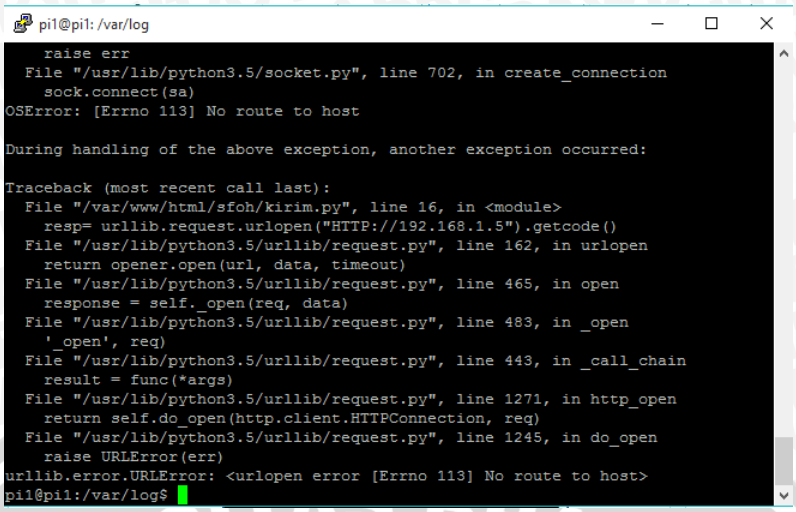


Gambar 6.20 *Web Server* pada *node level* 2 dalam kondisi up

Web Server node level 2 dalam keadaan hidup, sehingga *file* yang harus melalui *node level* 2 dengan tujuan *IP* = 192.168.1.3 akan terkirim ke *node level* 2.

Gambar 6.21 List *File* yang terkirim di *node level* 2

File yang tersimpan di *node level* 2 berjumlah 5 buah. 5 *file* ini 4 diantaranya memiliki tujuan akhir *IP* 192.168.1.5 dan 1 *file* memiliki tujuan akhir *IP* 192.168.1.6.



Gambar 6.22 Log *file* pengiriman ke *node level* 3

Pengiriman ke tujuan akhir gagal karena *web Server node level* 3 dalam keadaan tidak hidup.

Kesimpulan pengujian adalah :

Pengiriman *file* yang ter*upload* di *node level* 1 tidak akan terkirim langsung ke tujuan apabila *node* tujuan mati. Metode *store and forward* pada pengujian ke tiga ini berhasil

6.1.4 *Node level* 1, 2 dan 3 Hidup

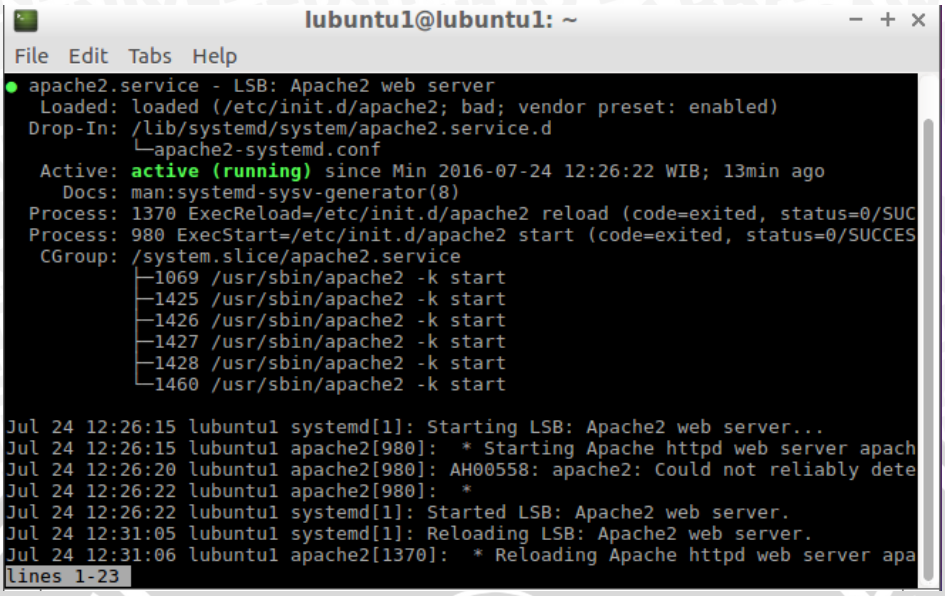
Gambar 6.23 Skenario 4

Skenario Keempat ini adalah pengujian dimana semua *node* dalam keadaan aktif. Skenario pengujian keempat ini dilakukan untuk memastikan bahwa *Routing* serta sistem berjalan dengan baik.

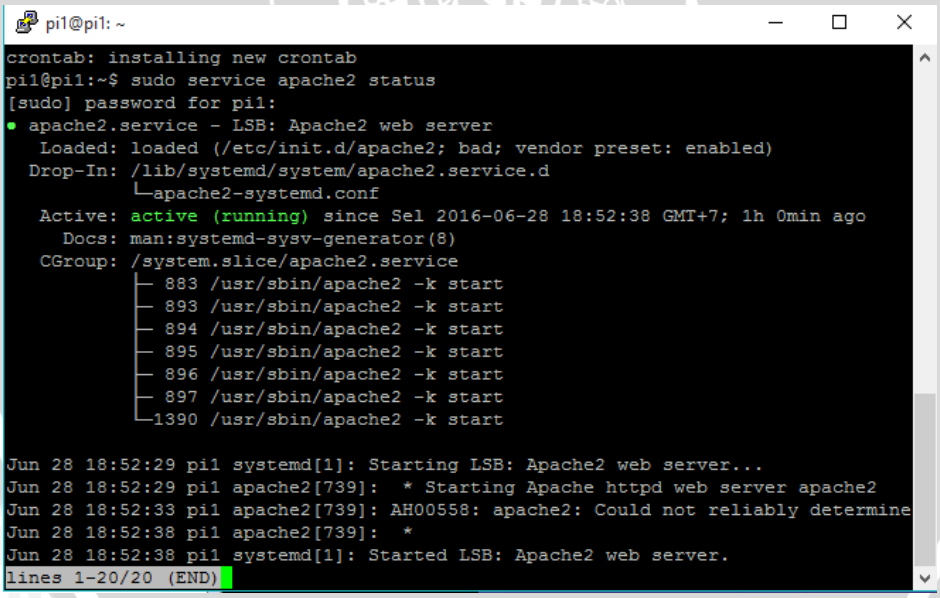
Uji coba dilakukan dengan mengirim *file* dari *node level* 1 *IP* = 192.168.1.2 menuju *node* akhir *level* 3 *IP* = 192.168.1.5

Berikut adalah konfigurasi tiap *node* pada skenario keempat :

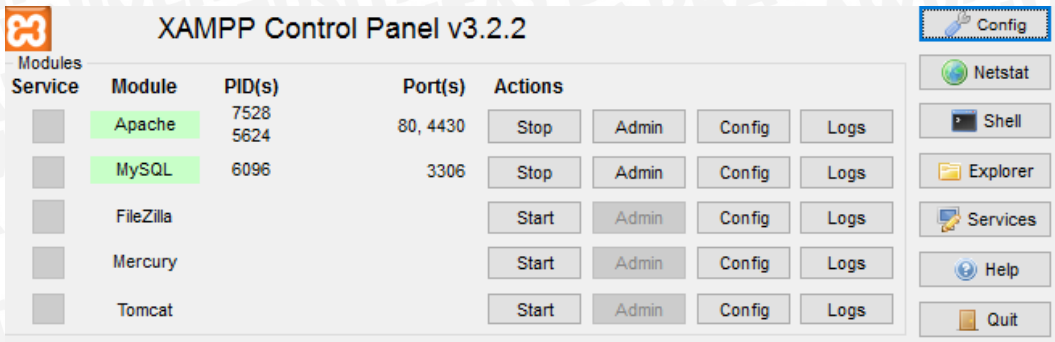
Node level 1 (192.168.1.2)



Gambar 6.24 *Web Server node level* 1 dalam keadaan hidup



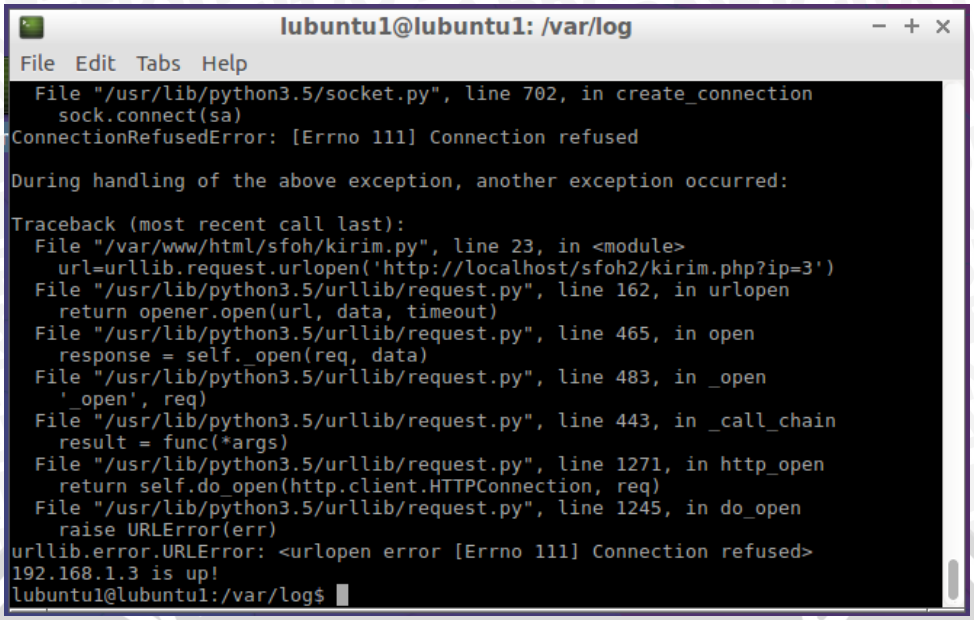
Gambar 6.25 *Web Server node level* 2 dalam keadaan hidup



Gambar 6.26 *Web Server node level* 3 dalam keadaan hidup

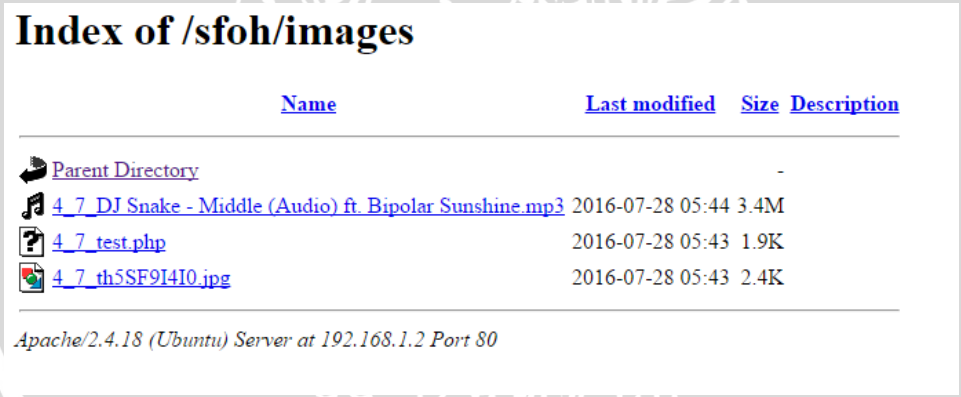
Gambar 6.27 Daftar *file* pada *node level* 1

Node level 1 memiliki beberapa list *file* yang memiliki tujuan masing-masing. Pada uji coba kali ini *file* akan dikirimkan menuju *node* tujuan 192.168.1.5. Berdasar *Routing* pada jaringan, *file* yang dikirim menuju 192.168.1.5 harus melewati *IP* 192.168.1.3. Sehingga semua *file* milik *IP node level* 2 dengan *IP* 192.168.1.3 akan dikirim terlebih dahulu.



Gambar 6.28 *Web Server node level* 2 dalam keadaan aktif.

Karena *web Server* dalam keadaan aktif maka *file* yang ada di *node level* 1 akan dikirimkan ke *node level* 2.

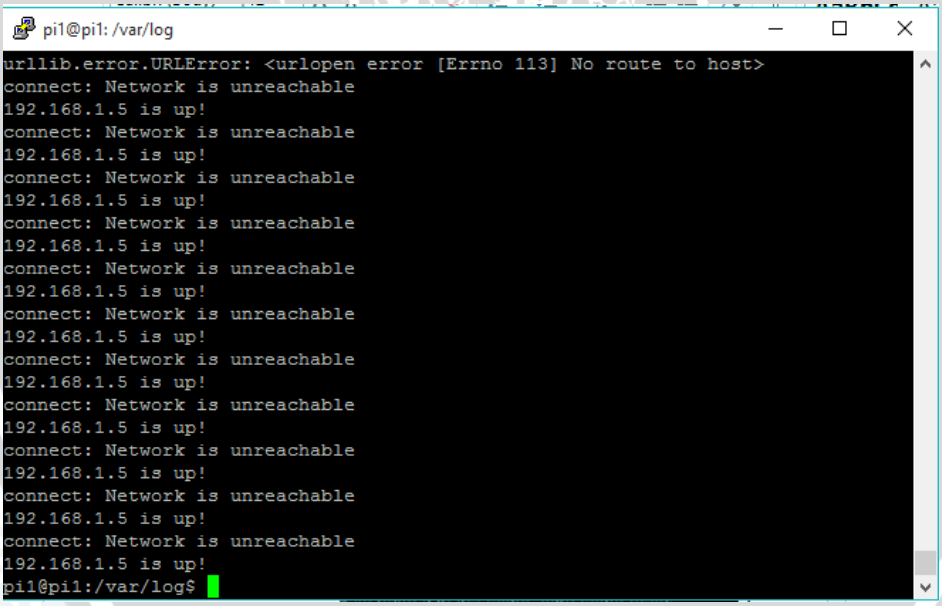


Gambar 6.29 List *file* pada *node level* 1 usai pengiriman *file*

File yang memiliki *filename* depan "3" semua terkirim ke *IP node level* 2 = 192.168.1.3.

Gambar 6.30 Daftar *file* pada *node level* 2

Node level 2 akan melanjutkan pengiriman ke *node* akhir apabila *node* akhir dalam keadaan hidup.



Gambar 6.31 *Web Server* tujuan akhir dalam keadaan aktif

Web Server node akhir aktif menjadi tanda bahwa *node level* 2 melakukan pengiriman *file* ke *node* akhir *IP* 192.168.1.5.

Gambar 6.32 Daftar *file* terkirim di *node* akhir

Pengiriman *file* yang di kirim dari *node level* 1 adalah 4 buah. 4 buah *file* itu terkirim semua dengan baik.

Daftar *file* yang dikirim dan sampai pada tujuan akhir adalah :

- 11Ant_Man_2015_HDR*IP*720p.srt
- 11*DTN*.pdf
- 11KTP Mas GigihC.jpg
- 11KattyPerryUnconditionallyOfficial.flac

Hasil pengujian ini adalah :

File yang di *upload* dapat terkirim ke tujuan akhir apabila *web Server node* level 1 , *node level* 2 dan *node level* 3 semua terhubung. Apabila tidak terhubung ketiganya, maka otomatis metode *store and forward* akan bekerja.

Kesimpulan pengujian ini adalah :

File dapat terkirim dengan baik dan metode *store and forward* berjalan dengan baik.

6.2 Pengujian waktu transfer file

Pengujian waktu *transfer file* dilakukan untuk mengetahui seberapa lama pengiriman *file* antar *node* apabila tanpa menggunakan metode *store and forward*. Pengujian ini dilakukan sebanyak 15 kali dengan melakukan pengiriman *file* sebesar 1MB, 5MB, 10MB, 15MB, 20MB.

Sebelum melakukan pengiriman *file*, kita perlu meng*upload* 5 *file* yang diperlukan. 5 File itu memiliki size 1 MB, 5 MB, 10 MB, 15MB, 20 MB

Tabel 6.1 Proses *upload file*

Ukuran *File*	**Status *upload***	
	Sukses	**Gagal**
1 MB	Sukses	-
5 MB	Sukses	-
10 MB	Sukses	-
15 MB	Sukses	-
20 MB	Sukses	-

Usai proses *upload* selesai, kita bisa melakukan pengiriman *file* menuju *node* selanjutnya. Berikut adalah rata-rata pengiriman file dari node level 1 menuju node level 2.

Tabel 6.2 Rata – rata Pengiriman 5 *file* dalam 1 waktu yang sama ke *node level* 2

Ukuran *File*	**Rata-rata Waktu Transfer**
1 MB	0,4201444 s
5 MB	2,121103267 s
10 MB	4,133738733 s
15 MB	6,153445333 s
20 MB	8,255388533 s

Tabel 6.3 Rata - rata Pengiriman 5 *file* dalam 1 waktu yang sama ke *node level* 3

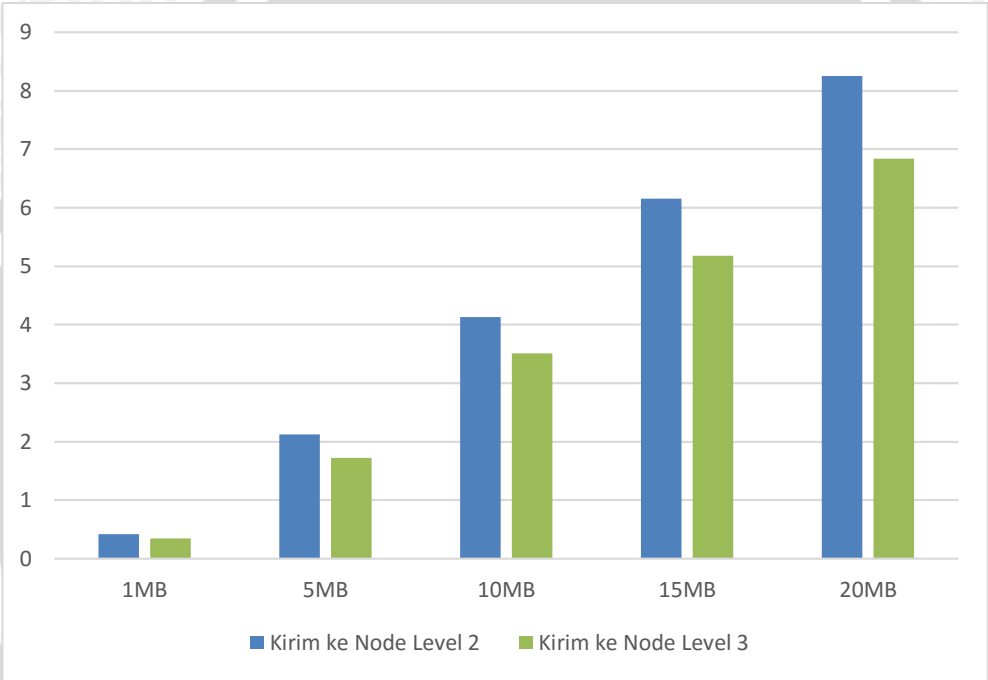
Ukuran *File*	**Waktu Transfer**
1 MB	0,3469098 s
5 MB	1,7240712 s
10 MB	3,513421133 s
15 MB	5,175677867 s
20 MB	6,834271133 s

Total *size* yang dikirimkan adalah 51 MB. Total waktu transfer yang dibutuhkan tiap node adalah sebagai berikut :

Tabel 6.4 Total rata-rata waktu transfer keseluruhan antar node

Node level 1 ke node level 2 (dalam detik)	Node level 2 ke node level 3 (dalam detik)
21.08382	**17.594351**

Berikut adalah grafik waktu transfer file antar node :



Gambar 6.33 Grafik waktu transfer rata – rata file antar node pada pengujian 2

Kesimpulan dari pengujian 2 adalah

Berdasar diatas, waktu transfer file antar node dari node level 1 ke node level 2 lebih lama daripada transfer file dari node level 2 ke node level 3. Hal ini dipengaruhi karena perbedaan perangkat ditiap nodenya. Secara keseluruhan rata-rata waktu yang dibutuhkan untuk mengirimkan file dari node level 1 ke node level 3 adalah **38.67 detik.**

6.3 Hasil pengujian

Hasil yang didapat dari 2 pengujian diatas adalah, metode store and forward yang dibuat dapat berjalan dalam pengiriman file. Apabila tidak terdapat 2 node yang terhubung secara langsung maka file tidak akan terkirim, namun disimpan pada folder khusus di setiap nodenya.

Pada pengujian 1, routing yang dibuat sudah dipastikan, sehingga, apabila ada node yang mati maka metode store and forward berfungsi. Pada

skenario 2, node level 1 dan 3 dalam keadaan hidup, namun node level 2 dalam keadaan mati, sehingga tidak ada file yang terkirim dan file tetap berada pada node level 1.

Pada pengujian 2, dilakukan pengiriman 5 file, dengan size 1MB, 5MB, 10MB, 15 MB dan 20 MB. Kelima file ini dikirim dalam waktu yang bersamaan dari node level 1 menuju node level 2. Hasil yang didapat, pengiriman dari node level 1 menuju node level 2 memerlukan waktu yang lebih lama daripada pengiriman dari node level 2 menuju node level 3. Hal ini dapat terjadi karena, perbedaan perangkat yang digunakan, sehingga memerlukan waktu lebih dalam melakukan pengiriman file.

BAB 7 PENUTUP

7.1 Kesimpulan

Implementasi metode *store and forward* pada *HTTP* berhasil dibuat. Implementasi ini berhasil berjalan di OS linux, Raspberry Pi dan Windows. Bahasa pemrograman *PHP* serta Python membantu implementasi ini dapat berjalan dengan baik. Hasil implementasinya adalah metode *store and forward* pada pengiriman file dapat berjalan dengan baik selama memiliki *Routing* yang jelas dan header *file* yang jelas pada *file*.

Implementasi metode *store and forward* pada *HTTP* ini sangat bergantung pada sebuah *web Server*. Apabila *web Server* pada sebuah *node* tidak aktif, maka sistem menganggap *node* tersebut dalam keadaaan mati.

Rata-rata Waktu tempuh yang diperlukan pada pengiriman file tanpa menggunakan metode *store and forward* dari node level 1 menuju node level 2 adalah sekitar 21,05 detik. Sedangkan rata-rata waktu tempuh file dari node level 2 menuju node level 3 adalah 17,29 detik. Secara keseluruhan pengiriman file dari node level 1 hingga node level 3 memerlukan waktu tempuh rata-rata sekitar 38.34 detik (diambil dari hasil pengujian).

7.2 Saran pengembangan lebih lanjut

Saran untuk pengembangan lebih lanjut adalah :

- *IP dinamis*

  Untuk pengembangan lebih lanjut, lebih baik menggunakan IP yang dinamis untuk setiap node. Hal ini bertujuan agar siapapun yang terhubung dalam satu jaringan yang sama dapat melakukan pengiriman maupun menerima file yang dikirim dalam jaringan yang sama.
- *Routing* otomatis

  Routing otomatis adalah, sistem yang dibuat hanya membaca seluruh *IP* yang tergabung pada suatu jaringan lokal. Sistem yang membuat *Routing* sendiri. Sehingga, paket bisa berada di *IP* yang berbeda. Hal ini akan mengurangi beban *Server* ketika melakukan pengiriman dalam jumlah banyak ke tujuan *IP* yang sama ataupun berbeda

- Diimplementasikan dengan menambah perangkat yang bergerak
- Digunakan pada pengiriman *file size* besar antar proxy atau antar *Server*.

DAFTAR PUSTAKA

Daerah terpencil 2015, [Penjelasan].
Tersedia di : *HTTP*://id.kopernik.ngo/help/apa-yang-dimaksud-dengan-daerah-terpencil-mengapa-kopernik-bekerja-untuknya [diakses 20 Oktober 2015]

Web Server 2015 , [Penjelasan].
Tersedia di : *HTTP*://emka.*web*.id/special/2012/apa-itu-*web-Server/* [diakses 20 Oktober 2015]

Web Server 2015, [Penjelasan].
Tersedia di : *HTTP*://www.dedeerik.com/pengertian-fungsi-serta-cara-kerja-*web-Server/* [diakses 20 Oktober 2015]

Taylor, David., Turner Jonathan., 2005 Towards a Diversified *Internet*. Washingtion University in Saint Louis

Schindler, Felix., 2010 *Weaknesses of today's Internet architecture*. Fakultas Ilmu Komputer, Teknik. Universitas Munchen

Wood, Lloyd., Holliday, Peter., Floreani, Daniel., Psaras, Ioannis., 2009. *Moving data in DTNs with HTTP and MIME*. Centre for Communication Systems Research University of Surrey

CURL 2014 [Program Komputer]
Tersedia di: *HTTP://PHP.net/manual/en/CURL.examples-basic.PHP* [diakses 4 Agustus 2016]

Move *Upload File* 2014 [Program Komputer]
Tersedia di: *HTTP://PHP.net/manual/en/function.move-uploaded-file.PHP* [diakses 25 Maret 2016]

CURL File upload 2014 [Program Komputer]
Tersedia di: *HTTPs://wiki.PHP.net/rfc/CURL-file-upload* [diakses 25 Maret 2016]

$_*FILE*S [Program Komputer]
Tersedia di: *http://php.net/manual/en/reserved.variables.files.php* [diakses 1 April 2016]

Python 2012 [Program Komputer]
Tersedia di: *http://stackoverflow.com/questions/8984287/execute-php-code-in-python* [diakses 1 Mei 2016]

Cron 2013 [Program Komputer]

Tersedia di: *HTTPs://en.wikIPedia.org/wiki/Cron* [diakses pada tanggal 1 Mei 2016]

Popa,Lucian., Ghodsi, Ali., Stoica,Ion., 2010. *HTTP* as the Narrow Waist of the Future *Internet*. U.C. Berkeley / ICSI 2010.

Sari,Nutria Iman., 2015. *Implementasi Web Service Pada Delay Tolerant Network.* S1. Teknik In*forma*tika, Filkom, Universitas Brawijaya Malang

Rahmania,Lidya amalia. 2013 . *Penerapan Delay Tolerant Network (DTN) untuk Sistem Konsultasi Kesehatan Jarak Jauh Berbasis Web.* S1. Teknik In*forma*tika, PTIIK, Universitas Brawijaya Malang

Lubuntu[Penjelasan]

Tersedia di :*HTTP*s://help.ubuntu.com/community/Lubuntu#System_requirements

Baucke, Stephan., 2007. Flexible Architecture for the Future *Internet*. Ericsson Research/2007

Metode Store and Forward [Penjelasan]

Warthman, Forest. Delay-and Disruption-Tolerant Networks (DTNs) v2.02. warthman Associates IPNSIG, 2013

Subhan,Achmad KH., Harsono Nonot, Fauziyah Anik, Latifah. 2009 . *Implementasi Metode Store-Forward Dengan Protokol SMTP Untuk Pengiriman Fax Pada Jaringan IP Sebagai Alternatif Migrasi Layanan Facsimile Pada Next Generation Network.* Politeknik Elektronika Negeri Surabaya, Institut Teknologi Sepuluh Nopember (ITS) Surabaya.