Halaman Judul

LAPORAN PENELITIAN

ANALISIS KECEPATAN DOWNLOAD FILE BERBAGAI PROTOKOL JARINGAN

oleh
W A G I T O, S.T., M.T.
NIDN : 0522126901
NPP : 961080

Mendapat Bantuan Biaya Penelitian dari Puslitbang dan PPM
Semester Ganjil 2014/2015

Sekolah Tinggi Manajemen Informatika dan Komputer
AKAKOM YOGYAKARTA
Tahun 2015

HALAMAN PENGESAHAN

1.	a. Judul Penelitian	:	Analisis Kecepatan Download File Berbagai Protokol Jaringan
	b. Bidang Ilmu	:	Jaringan Komputer
	c. Kategori	:	Implementasi Jaringan Komputer
2.	Ketua Peneliti		
	a. Nama	:	Wagito, S.T., M.T.
	b. NIDN	:	0522126901
	c. NPP	:	961080
	d. Pangkat/Golongan	:	Pembina Tk 1 / IV B
	e. Jabatan Fungsional	:	Lektor Kepala
	f. Jurusan/Prodi	:	Teknik Informatika
	g. Alamat Institusi	:	Jalan Raya Janti Karang Jambe, Yogyakarta
5.	Waktu Penelitian	:	6 bulan
6.	Biaya Penelitian	:	

Yogyakarta, Maret 2015

Mengetahui

Ketua Prodi

Febri Nova Lenti, S.Si., M.T.
NPP. 961079

Ketua Peneliti

Wagito, S.T., M.T.
NPP. 961080

Menyetujui

Kepala Puslitbang dan PPM
STMIK AKAKOM

Dra. Syamsu Windarti, M.T., Apt
NIP. 19660710 199303 2 001

Kata Pengantar

Puji syukur kepada Allah S.W.T. karena telah melimpahkan rahmat, hidayah dan taufik-Nya Berkat pertolongan dan tuntunan-Nya serta dengan berbagai usaha akhirnya penelitian ini berhasil diselesaikan dengan baik.

Penelitian yang berjudul Analisis Kecepatan Download Berbagai Protokol Jaringan dilakukan untuk meneliti pengaruh penggunaan berbagai macam protokol jaringan terhadap kecepatan *download file*. Proses *download file* memerlukan koneksi terus-menerus yang berpotensi menghabiskan *bandwidth* jaringan. Hal ini dapat mengganggu aktivitas Internet lain. Oleh karena itu perlu ditentukan cara paling baik dalam melakukan *download file.* Pemilihan protokol jaringan paling cepat dalam melakukan proses *download file* menjadi sesuatu yang penting.

Hasil penelitian ini masih banyak kekurangannya, sehingga kritik dan saran yang membangun untuk lebih mengembangkan hasilnya sangat diharapkan. Semoga hasil penelitian ini bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Penulis

Daftar Isi

Daftar Gambar

ABSTRAK

Proses *download file* memerlukan koneksi terus-menerus yang berpotensi menghabiskan *bandwidth* jaringan. Hal ini dapat mengganggu aktivitas Internet lain. Oleh karena itu perlu ditentukan cara paling baik dalam melakukan *download file.* Pemilihan protokol jaringan paling cepat dalam melakukan proses *download file* menjadi sesuatu yang penting. Penelitian bertujuan untuk membandingkan kecepatan *download file* menggunakan bermacam-macam protokol jaringan.

Proses *download file* dapat dilakukan menggunakan bermacam-macam protokol jaringan. Protokol jaringan yang biasa dipakai untuk *download file* adalah HTTP, HTTPS, FTP, SFTP, SCP dan TFTP. Masing-masing protokol jaringan punya perilaku yang berbeda dalam melakukan proses *download file*. Suatu protokol perlu proses *login* sebelum *download*, Protokol lain perlu proses enkripsi pada saat proses *download*. Secara keseluruhan, perilaku tersebut akan mempengaruhi kecepatan dalam melakukan proses *download file*.

Beberapa hasil diperoleh pada penelitian. Tipe *file* tidak berpengaruh terhadap kecepatan *download file.* Semakin besar ukuran *file* menyebabkan kecepatan *download file* cenderung semakin kecil. Protokol jaringan tidak secara signifikan berpengaruh pada perbedaan kecepatan *download file* namun protokol TFTP cenderung paling cepat. Untuk protokol yang umum dipakai pada jaringan Internet, protokol SCP menunjukkan nilai kecepatan *download file* yang paling tinggi.

Kata kunci: kecepatan, *download*, protokol

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pada saat ini jaringan Internet sudah menjadi kebutuhan kebanyakan orang. Hampir seluruh sektor kehidupan tidak terlepas dari peran jaringan Internet. Orang menggunakan jaringan Internet untuk berbagai keperluan mulai dari mencari informasi, menyebarkan informasi, mengirim e-mail, menerima e-mail, berbincang secara *online*, melakukan *download file*, melakukan *upload file*, dan sebagainya. Bahkan keperluan melihat acara televisi, mendengarkan siaran radio, mendengarkan alunan musik, menggunakan telepon serta melihat video juga dapat dilakukan melalui jaringan Internet. Pendek kata, jaringan Internet sudah menguasai banyak perilaku kehidupan manusia sehari-hari.

Masing-masing aktivitas dalam berhubungan dengan jaringan Internet memerlukan protokol jaringan. Untuk mencari informasi biasanya dilakukan dengan aktivitas *browsing*. Menurut kamus istilah komputer dan informatika (Andino Maseleno, 2003), *browsing* adalah aktivitas menjelajah beberapa situs di Internet. Aktivitas *browsing* menjadi aktivitas yang paling banyak dilakukan pada saat berhubungan dengan jaringan Internet. Aktivitas *browsing* pada jaringan Internet umumnya menggunakan protokol jaringan HTTP dan HTTPS. Protokol

HTTP umumnya menggunakan port nomor 80 sedangkan protokol HTTPS umumnya menggunakan port nomor 443. Informasi yang ditampilkan umumnya berbentuk dokumen teks HTML.

Untuk *browsing* biasanya digunakan perangkat lunak *browser*. *Browser* yang umum dipakai untuk *browsing* antara lain Mozilla Firefox, Internet Explorer, Opera, Google Chrome, Konqueror dan sebagainya. *Browser* harus didukung oleh bahasa sekrip untuk dokumen Internet. Bahasa sekrip yang umumnya didukung oleh *browser* adalah HTML, CSS dan Java Script. *browser* juga harus mendukung protokol jaringan untuk akses Internet. Protokol jaringan yang umum didukung oleh *browser* adalah HTTP, HTTPS dan FTP *anonymous*. Pada *browser* Konqueror ada tambahan dukungan protokol SFTP dan SMB. *browser* Internet Explorer hanya dapat berjalan pada sistem operasi Windows. *browser* Konqueror hanya dapat berjalan pada sistem operasi Linux. *Browser* Mozilla Firefox, Google Chrome dan Opera dapat berjalan pada beberapa sistem operasi seperti Windows dan Linux.

Aktivitas yang umumnya menyertai *browsing* adalah *download*. Menurut kamus istilah komputer dan informatika (Andino Maseleno, 2003), *download* adalah mengambil *file* atau melakukan *transfer file* dari satu komputer ke komputer lainnya. *download* dilakukan untuk mengambil *file* dari suatu situs penyedia untuk disimpan pada komputer.

Aktivitas *download* biasa dipakai untuk mengambil *file* musik, video, buku, perangkat lunak dan sebagainya. *file* musik yang biasa di*download*

misalnya *file* mp3. *file* video yang biasa di*download* misalnya mp4. *file* buku yang biasa di*download* umumnya berbentuk pdf. *file* buku yang biasa di*download* misalnya pdf. *file* perangkat lunak yang biasa di*download* biasanya dalam bentuk exe. Selain itu, *file* juga dapat diwujudkan dalam bentuk *file* terkompresi zip.

Aktivitas *download* dapat dilakukan menggunakan beberapa protokol misalnya HTTP, HTTPS, FTP, SCP, TFTP dan sebagainya. Masing-masing punya perilaku yang berbeda-beda dalam melakukan proses *download*. Protokol HTTP dan HTTPS biasanya dapat digunakan untuk *download file* tanpa melakukan *login*. Protokol FTP biasanya digunakan untuk proses *download file* melalui proses *login*, walaupun bisa diatur tanpa melakukan proses *login*. Protokol SCP biasa digunakan untuk proses *download file* dengan melibatkan faktor keamanan data yaitu enkripsi dan dekripsi.

Seringkali suatu situs Internet menyediakan beberapa cara untuk menyediakan fasilitas *download* bagi para penggunanya. Misalnya saja situs www.kernel.org (2014) yang menyediakan perangkat lunak *kernel* sistem operasi Linux. Situs ini menyediakan protokol HTTP, FTP dan RSYNC untuk melakukan *download*. Contoh lain adalah situs kambing.ui.ac.id (2014) milik Universitas Indonesia. Situs ini menyediakan protokol HTTP dan FTP untuk *download file*. Pada situs lain, *download* disediakan dengan variasi protokol jaringan yang lain. Secara umum, situs penyedia *file* biasanya menyediakan beberapa protokol untuk melakukan proses *download file*.

Jika tersedia beberapa protokol jaringan untuk proses *download file*, maka

pengguna dapat menggunakan salah satu. Pemilihan protokol yang dipakai untuk *download file* umumnya berdasarkan kecepatan *download*. Pertimbangan kecepatan bisa dipakai apabila jaringan Internet tidak melakukan blok pada protokol jaringan tertentu. Karena alasan keamanan jaringan, suatu sistem jaringan tidak mengizinkan akses ke jaringan Internet menggunakan suatu protokol tertentu.

Proses *download file* dapat dilakukan menggunakan perangkat lunak *browser* atau perangkat lunak khusus. Perangkat lunak khusus *download* misalnya Download Accelerator (DAC), Internet Download Manager (IDM) dan GetRight. Penggunaan perangkat lunak khusus memungkinkan penggunaan fitur khusus *download* misalnya resam dan multi koneksi. Fitur resam memungkinkan melakukan pengulangan proses *download file* tanpa dimulai dari awal. Fitur multi koneksi memungkinkan penggunaan beberapa koneksi serentak ketika melakukan proses *download*. Fitur resam dan multi koneksi mengesankan proses *download* menjadi lebih cepat.

Proses *download file* memerlukan koneksi terus-menerus (*continouos connection*). Koneksi terus-menerus punya potensi untuk menghabiskan *bandwidth* jaringan. Dengan demikian proses *download* dapat mengganggu aktivitas Internet lain seperti *browsing*. Oleh karena itu perlu ditentukan pemilihan protokol yang paling cepat dalam melakukan proses *download file*.

Pada penelitian ini dicoba untuk membandingkan kecepatan *download file* menggunakan bermacam-macam protokol jaringan yang umum dipakai untuk

proses *download file* yaitu HTTP, HTTPS, FTP, SCP dan TFTP. Setelah dibandingkan, tentunya akan diketahui protokol jaringan yang paling cepat dalam melakukan *download file*. Untuk melakukan proses *download file*, digunakan perangkat lunak bantu Curl yang mendukung beberapa protokol jaringan. Untuk mengamati proses *download file*, disusun algoritme untuk mengamati kecepatan *download file* menggunakan berbagai protokol.

Pengamatan terhadap pengaruh penggunaan protokol jaringan terhadap kecepatan *download* harus dilakukan tanpa pembatasan *bandwidth* jaringan. Pengamatan kecepatan *download* melalui jaringan Internet, tidak dapat menampilkan hasil yang benar, karena adanya pembatasan *bandwidth*. Dengan demikian, perlu disusun suatu sistem jaringan yang dilengkapi dengan *server* yang mendukung bermacam-macam protokol jaringan.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian adalah bagaimana pengaruh berbagai variasi protokol jaringan pada proses *download file*. Protokol jaringan yang diamati untuk proses *download* adalah FTP, HTTP, HTTPS, SFTP, SCP dan TFTP.

1.3 Batasan Masalah

Batasan yang perlu diperhatikan dalam kaitan dengan kemungkinan

masalah yang muncul penelitian adalah:

- Penelitian dititikberatkan pada pengamatan kecepatan *download file* melalui berbagai protokol jaringan.
- Protokol jaringan yang diamati untuk *download file* meliputi FTP, HTTP, HTTPS, SFTP, SCP serta TFTP.
- Pengamatan dilakukan untuk proses *download file* dengan berbagai variasi ukuran *file*.
- Pengamatan dilakukan untuk proses *download file* yang sudah atau belum mengalami kompresi
- Waktu yang diperlukan untuk proses *login server* diabaikan dalam pengamatan kecepatan *download file*.

1.4 Tujuan Penelitian

Penelitian analisis kecepatan *download* melalui berbagai protokol jaringan adalah sebagai berikut.

- Menyusun algoritme untuk mengamati kecepatan *download file* menggunakan berbagai protokol.
- Membandingkan kecepatan *download file* menggunakan bermacam-macam protokol jaringan.
- Mengetahui protokol jaringan yang paling cepat dalam melakukan proses *download file*.

1.5 Manfaat Penelitian

Manfaat penelitian berkaitan dengan analisis kecepatan *download* melalui berbagai macam protokol adalah dapat mengetahui protokol jaringan yang paling cepat dalam melakukan *download file*. Pengetahuan ini sangat bermanfaat untuk menentukan pemilihan protokol paling baik untuk proses *download file* dari beberapa situs Internet. Pengetahuan ini juga sangat bermanfaat untuk menentukan pilihan protokol jaringan pada pengelolaan sistem *server* penyedia layanan *file*

1.6 Target Luaran

Hasil penelitian direncanakan dilakukan publikasi dan seminasi pada kegiatan ilmiah. Publikasi yang diinginkan adalah pada jurnal ilmiah nasional yang sudah akreditasi. Jika sulit masuk pada jurnal akreditasi, maka paling tidak bisa masuk jurnal ilmiah nasional.

BAB 2 TINJAUAN PUSTAKA

Beberapa penelitian berkaitan dengan kecepatan *transfer* data telah dilakukan dan dipublikasikan dalam bentuk jurnal. Penelitian-penelitian tersebut berbeda-beda pada objek yang diteliti. Kecepatan *transfer file* dapat dibandingkan berdasarkan medium transmisi yang dipakai. Kecepatan *transfer file* juga dapat dibandingkan berdasarkan penyedia layanan Internet. Namun masih belum ditemukan penelitian yang membandingkan kecepatan *transfer file* berdasarkan jenis protokol jaringan yang dipakai.

Penelitian Sugeng Riyadi dan Harjono tentang analisis perbandingan kecepatan *download* pada GSM dibandingkan kecepatan *download* beberapa provider GSM. Pada penelitian diperoleh bahwa ada perbedaan kecepatan *download* pada provider yang dibandingkan. Pada penelitian ini tidak disebutkan protokol jaringan yang digunakan untuk proses *download*. Analisis perbedaan kecepatan dititikberatkan didasarkan pada perbedaan penyedia layanan seluler.

Penelitian Sony Bahagia Sinaga tentang analisis perbandingan kecepatan *transfer* data menggunakan kabel UTP dan Wifi dibandingkan kecepatan *transfer* data melalui dua medium transmisi berbeda. Pada penelitian ini diperoleh hubungan antara jarak dengan kecepatan *download* dan *upload*, bahwa semakin jauh jarak *server* maka nilai kecepatan *download*/*upload* semakin kecil dan

sebaliknya semakin dekat jarak *server* maka nilai kecepatan *download*/*upload* semakin besar. k memutuskan media apa yang akan digunakan untuk melakukan suatu *transfer* data. Menurut penelitian ini penggunaan kabel UTP menghasilkan kecepatan yang lebih baik. Penelitian ini membandingkan kecepatan *transfer* data melalui dua medium transmisi yang berbeda yaitu kabel UTP dan Wifi.

Hasil penelitian Gede Wahyudi dan Trisna Hanggara tahun 2013 tentang analisis perbandingan kinerja antara NFS dan PDC Samba menunjukkan adanya perbedaan hasil antara dua protokol yang berbeda. Menurut penelitian ini diketahui bahwa kecepatan *transfer* data menggunakan protokol NFS lebih baik dibanding Samba. Pada penelitian ini, *transfer* data dilakukan dengan teknik berbagi (*sharing*) *file*.

Penelitian tentang implementasi VPN PPTP untuk integrasi jaringan dicoba dirancang integrasi dua jaringan dengan memanfaatkan protokol PPTP. Salah satu yang diteliti pada penelitian ini adalah konsekuensi integrasi jaringan pada beberapa aspek implementasi jaringan yaitu pertukaran data antar jaringan menggunakan *sharing file*, pemendekan jalur *routing*, koneksi HTTP dan koneksi basisdata. Pada penelitian ini belum dilakukan pengamatan lebih detail berkaitan dengan aspek kecepatan *transfer* data antara dua komputer yang terhubung melalui kanal VPN PPTP.

Penelitian analisis kecepatan *transfer* data melalui VPN PPTP mengamati pengaruh kompresi dan enkripsi pada *transfer* data melalui kanal PPTP. Penelitian belum memperhitungkan penggunaan berbagai variasi protokol jaringan

komputer. Pada penelitian ini diamati pengaruh kompresi dan enkripsi pada *transfer* data melalui kanal PPTP menggunakan protokol FTP. Pada penelitian ini, belum dibandingkan variasi penggunaan protokol jaringan untuk proses *transfer* data melalui kanan VPN PPTP.

Proses *transfer* data dapat dilakukan menggunakan berbagai macam protokol jaringan seperti FTP, SFTP, HTTP, HTTPS, TFTP dan sebagainya. Masing-masing protokol punya perilaku yang berbeda dalam *transfer* data. Penelitian ini dikembangkan untuk mengetahui kecepatan *transfer* data pada protokol jaringan yang punya perilaku mirip dalam proses *transfer* data.

BAB 3 TEORI

3.1 Protokol Jaringan

Sama seperti dalam komunikasi manusia, berbagai jaringan dan komputer protokol harus mampu berinteraksi dan bekerja sama untuk komunikasi jaringan untuk menjadi sukses. Sekelompok protokol yang saling terkait yang diperlukan untuk melakukan fungsi komunikasi disebut protokol. Protokol suite dilaksanakan oleh host dan perangkat jaringan baik perangkat lunak, perangkat keras atau keduanya (Cisco, 2015).

Salah satu cara terbaik untuk membuat visualisasi bagaimana protokol *suite* dalam berinteraksi dapat dilihat interaksi sebagai *stack*. Sebuah protokol *stack* menunjukkan bagaimana protokol *suite* individu diimplementasikan. Protokol yang dilihat dari segi lapisan, dengan masing-masing tingkat pelayanan yang lebih tinggi tergantung pada fungsi didefinisikan oleh protokol ditampilkan dalam tingkat yang lebih rendah. Lapisan bawah tumpukan berusaha dengan memindahkan data melalui jaringan dan menyediakan layanan untuk lapisan atas, yang fokus pada isi pesan yang dikirim.

Kasus tersebut dapat ditunjukkan dengan menggambarkan lapisan untuk aktivitas yang terjadi dalam komunikasi *face to face*. Pada lapisan bawah, lapisan

fisik, dapat dilihat dua orang, masing-masing dengan suara yang bisa mengatakan kata-kata keras. Pada lapisan kedua, lapisan aturan, dimiliki kesepakatan untuk berbicara dalam bahasa yang sama. Pada lapisan atas, lapisan isi, ada kata-kata yang benar-benar diucapkan. Ini adalah isi dari komunikasi.

Apakah dapat disaksikan percakapan ini, tidak akan benar-benar dapat dilihat lapisan mengambang. Penggunaan lapisan adalah model yang menyediakan cara untuk memudahkan dalam memecah tugas yang kompleks menjadi beberapa bagian dan menjelaskan bagaimana bekerja.

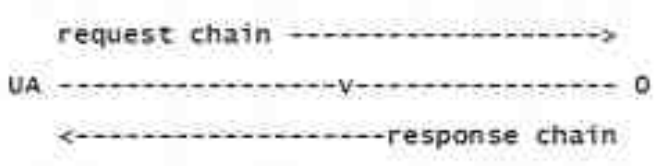
3.1.1 HTTP

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol tingkat aplikasi untuk distribusi serta kolaborasi, sistem informasi *hypermedia*. HTTP telah digunakan oleh informasi *World Wide Web* global dimulai sejak tahun 1990. Versi pertama HTTP, disebut sebagai HTTP/0.9, adalah sebuah protokol sederhana untuk transfer data mentah di Internet. HTTP/1.0, seperti yang didefinisikan oleh RFC 1945, meningkatkan protokol dengan mengizinkan pesan dalam format pesan *MIME-like*, yang mengandung meta informasi tentang data yang ditransfer dan pengubah pada semantik *request/response*. Namun, HTTP/1.0 tidak cukup mempertimbangkan efek dari *proxy* hierarki, *caching*, kebutuhan untuk koneksi *persistent*, atau *virtual host*. Selain itu, perkembangan aplikasi implementasi tak lengkap yang menyebut dirinya "HTTP/1.0" telah mengharuskan perubahan versi protokol agar dua aplikasi yang saling komunikasi

masing-masing mampu menentukan kemampuan sesungguhnya. Spesifikasi ini mendefinisikan protokol acuan untuk "HTTP/1.1". Protokol ini mencakup kebutuhan yang lebih ketat dibanding HTTP/1.0 untuk menjaga kelayakan implementasi masa depan. (Fielding, R., 1999)

HTTP adalah protokol *request/response*. Suatu klien mengirimkan permintaan ke *server* dalam bentuk metode permintaan, URI dan versi protokol, diikuti dengan pesan *MIME-like* yang mengandung permintaan pengubah, informasi klien, dan isi tubuh yang mungkin melalui sambungan dengan *server*. *Server* memberi respon dengan baris status, termasuk versi protokol pesan dan kode sukses atau kesalahan, diikuti dengan pesan *MIME-like* yang mengandung informasi *server*, entitas meta informasi dan mungkin isi *entity-body*.

Kebanyakan komunikasi HTTP dimulai oleh agen pengguna dan terdiri dari permintaan untuk diterapkan ke sumber daya pada beberapa *server* asal. Dalam kasus yang paling sederhana, hal ini dapat dicapai melalui koneksi tunggal (v) antara *user agent* (UA) dan *server* asal (O).



Gambar 3.1 Proses Permintaan HTTP (Fielding, R., 1999)

Sebuah situasi yang lebih rumit terjadi ketika satu atau lebih perantara

yang hadir dalam *request/response*. Ada tiga bentuk umum dari perantara: *proxy*, *gateway*, dan *tunnel*. *Proxy* adalah agen penerus, menerima permintaan untuk URI dalam bentuk mutlak, menulis ulang seluruh atau sebagian dari pesan, dan meneruskan permintaan yang dilakukan format ulang ke *server* yang diidentifikasi oleh URI. Sebuah *gateway* adalah agen penerima, bertindak sebagai lapisan di atas beberapa *server* lain (s) dan, jika perlu, menerjemahkan permintaan ke protokol *server* yang mendasari. Sebuah *tunnel* bertindak sebagai titik *estafet* antara dua koneksi tanpa mengubah pesan; *tunnel* digunakan komunikasi perlu melewati perantara (seperti *Firewall*) bahkan ketika perantara tidak dapat memahami isi pesan.

Gambar 3.2 Permintaan Melalui Proxy (Fielding, R., 1999)

Gambar 3.2 menunjukkan tiga perantara (A, B, dan C) antara *user agent* dan *server* asal. Permintaan atau respon pesan yang melintasi seluruh rantai akan melewati empat koneksi yang terpisah.

Komunikasi HTTP biasanya terjadi melalui koneksi TCP/IP. *Port* default adalah TCP 80, tetapi *port* lain dapat digunakan. Hal ini tidak menghalangi HTTP diimplementasikan di atas protokol lainnya di Internet atau jaringan lain. HTTP hanya menganggap transportasi yang handal; protokol yang menyediakan jaminan

tersebut dapat digunakan; pemetaan HTTP/1.1 *request* dan struktur respon ke unit transportasi data dari protokol yang dimaksud adalah di luar lingkup spesifikasi.

3.1.2 HTTPS

HTTP [RFC2616] pada awalnya digunakan secara luas di Internet. Namun, peningkatan penggunaan HTTP untuk aplikasi yang sensitif diperlukan langkah-langkah keamanan. SSL, dan penggantinya TLS [RFC2246] dirancang untuk memberikan keamanan *channel-oriented*. (Rescorla, E., 2000)

Agen yang bertindak sebagai klien HTTP juga harus bertindak sebagai klien TLS. Klien harus melakukan koneksi ke *server* pada *port* yang sesuai dan kemudian mengirim TLS *ClientHello* untuk memulai gandengan TLS. Ketika gandengan TLS telah selesai. Klien kemudian dapat memulai permintaan HTTP pertama. Semua data HTTP harus dikirim sebagai TLS "data aplikasi". Perilaku HTTP normal, termasuk koneksi tetap harus diikuti.

TLS menyediakan fasilitas untuk penutupan sambungan aman. Ketika peringatan penutupan valid diterima, implementasi dapat yakin bahwa tidak ada data lebih lanjut akan diterima pada koneksi itu. TLS implementasi harus memulai pertukaran peringatan penutupan sebelum menutup sambungan. Implementasi TLS mungkin, setelah mengirimkan peringatan penutupan, menutup koneksi tanpa menunggu rekan untuk mengirim peringatan penutupan, menghasilkan '*incomplete close*'. (Rescorla, E., 2000)

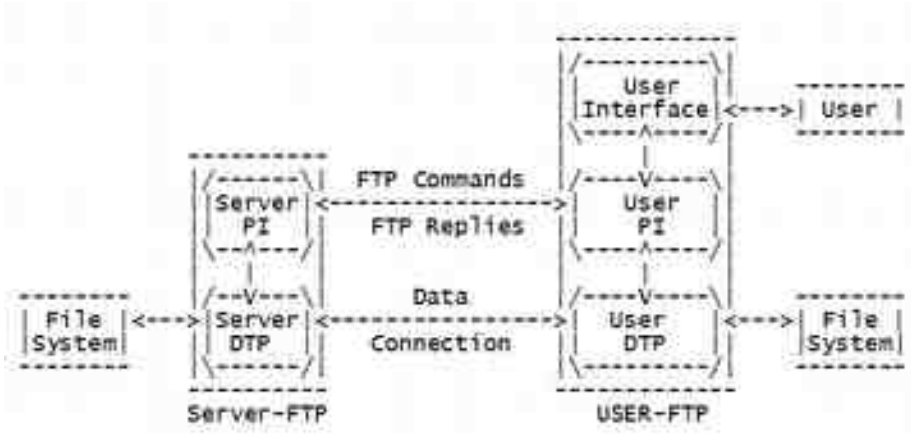
Data pertama yang *server* HTTP mengharapkan untuk menerima dari klien

adalah produksi *Request-Line*. Data pertama yang diharapkan *server* TLS (dan karenanya *server* HTTP/TLS) untuk menerima adalah *ClientHello*. Akibatnya, praktik umum telah menjalankan HTTP/TLS melalui *port* terpisah untuk membedakan mana protokol yang digunakan. Ketika HTTP/TLS sedang dijalankan melalui koneksi TCP/IP, *port default* adalah 443. Hal ini tidak menghalangi HTTP/TLS ditempatkan pada *port* transportasi lain. TLS hanya menganggap aliran data berorientasi koneksi yang dapat diandalkan. HTTP/TLS dibedakan dari HTTP URI dengan menggunakan pengenal protokol 'https'. (Rescorla, E., 2000)

3.1.3 FTP

Tujuan dari FTP adalah untuk mempromosikan berbagi *file* (program komputer dan/atau data), untuk mendorong penggunaan tak langsung atau implisit (melalui program) dari komputer *remote*, untuk melindungi pengguna dari variasi dalam sistem penyimpanan *file* antar *host*, dan untuk melakukan transfer data andal dan efisien. Meski pun digunakan secara langsung oleh pengguna di terminal, FTP dirancang terutama untuk digunakan oleh program.

Dengan definisi tersebut, model layanan FTP dapat digambarkan dalam diagram Gambar 3.3.



Gambar 3.3 Model FTP (Postel, J., 1985)

catatan:

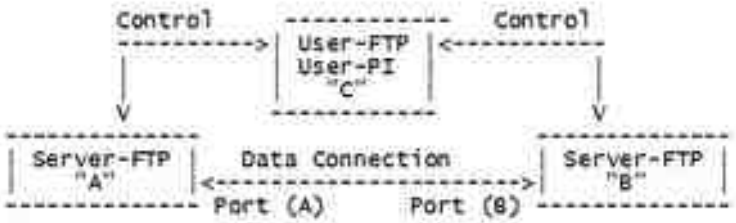
- sambungan data dapat digunakan di kedua arah
- sambungan data tidak perlu ada sepanjang waktu.

Dalam model yang digambarkan dalam Gambar 3.3, *interpreter user-protocol* memulai koneksi kontrol. Koneksi kontrol mengikuti protokol Telnet. Pada inisiasi pengguna, perintah FTP standar dihasilkan *user*-PI dan dikirim ke proses *server* melalui koneksi kontrol. (Pengguna dapat membuat sambungan kontrol langsung ke *server*-FTP dan menghasilkan perintah standar FTP mandiri, melewati proses *user*-FTP.). Balasan standar dikirim *server*-PI ke *user*-PI melalui koneksi kontrol dalam bentuk tanggapan perintah.

Perintah FTP menentukan parameter untuk koneksi data (*port* data, modus transfer, jenis representasi dan. struktur) dan sifat operasi sistem *file* (menyimpan, mengambil, menambah, menghapus dan sebagainya). *User*-DTP atau

rancangannya harus "mendengarkan" pada *port* data tertentu. *Server* memulai koneksi data dan transfer data sesuai dengan parameter tertentu. Perlu dicatat bahwa *port* data tidak perlu di *host* yang sama yang memulai perintah FTP melalui koneksi kontrol, tetapi pengguna atau proses *user*-FTP harus memastikan "mendengarkan" pada *port* data tertentu. Seharusnya juga dicatat bahwa koneksi data dapat digunakan untuk mengirim dan menerima secara simultan.

Dalam situasi lain pengguna mungkin ingin melakukan transfer *file* antara dua *host*, meski pun salah satu merupakan *host* lokal. Pengguna memasang koneksi kontrol untuk kedua *server* dan kemudian mengatur untuk koneksi data di antara dua *server* tersebut. Dengan cara ini, kontrol informasi akan diteruskan ke *user*-PI tetapi data ditransfer antara proses transfer data *server*. Berikut ini adalah model interaksi *server-server*.



Gambar 3.4 Interaksi Server-Server (Postel, J., 1985)

Protokol meminta syarat bahwa koneksi kontrol terbuka selama transfer data sedang berlangsung. Ini adalah tanggung jawab pengguna untuk meminta penutupan koneksi kontrol ketika selesai menggunakan layanan FTP, sementara

itu adalah *server* yang mengambil tindakan. *Server* dapat membatalkan transfer data jika koneksi kontrol ditutup tanpa perintah.

3.1.4 TFTP

TFTP adalah protokol yang sangat sederhana yang digunakan untuk melakukan transfer *file*. Setiap paket *non-terminal* diakui secara terpisah. Protokol ini awalnya dirancang oleh Noel Chiappa, dan dirancang ulang oleh dia, Bob Baldwin dan Dave Clark, dengan komentar dari Steve Szymanski. Revisi dilakukan saat dokumentasi termasuk modifikasi yang berasal dari diskusi dengan dan saran dari Larry Allen, Noel Chiappa, Dave Clark, Geoff Cooper, Mike Greenwald, Liza Martin, David Reed, Craig Milo Rogers (USC-ISI), Kathy Yellick, dan penulis. Pengakuan dan transmisi skema terinspirasi oleh TCP, dan mekanisme *error* disarankan oleh pesan EFTP batalkan PARC. (Sollins, K., 1992)

TFTP merupakan protokol sederhana untuk mentransfer *file* dan karena itu bernama *Trivial File Transfer Protocol* atau TFTP. TFTP telah diimplementasikan di atas *User Datagram Protokol* (UDP atau *Datagram*), sehingga dapat digunakan untuk memindahkan *file* antara komputer pada jaringan yang berbeda menerapkan UDP. (Hal ini seharusnya tidak mengecualikan kemungkinan pelaksanaan TFTP di atas protokol *datagram* lainnya.) TFTP dirancang untuk menjadi kecil dan mudah diimplementasikan. Oleh karena itu, tidak memiliki sebagian besar fitur dari FTP biasa. Satu-satunya hal yang dapat dilakukan adalah membaca dan menulis *file* (atau surat) dari/ke *server* jauh. TFTP

tidak bisa mendaftar direktori dan saat ini tidak memiliki ketentuan untuk autentikasi pengguna. Secara umum dengan protokol Internet lainnya, melewati 8 *bit byte* data. (Sollins, K., 1992)

Tiga modus transfer yang saat ini didukung oleh protokol TFTP adalah sebagai berikut: (Sollins, K., 1992)

- netascii (Ini adalah ascii sebagaimana didefinisikan dalam "*USA Standard Code For Information Interchange*" dengan modifikasi yang ditentukan dalam "*Telnet Protocol Spesification*"). Ascii yang dipakai 8 *bit*.
- oktet *raw* 8 *bit byte*. (Mous ini menggantikan modus "biner" versi sebelumnya)
- mail, karakter netascii dikirim ke pengguna tidak dalam bentuk *file*. (Modus *mail* adalah usang dan tidak boleh dilaksanakan atau digunakan.).

Kebanyakan kesalahan menyebabkan pemutusan sambungan. Kesalahan ditandai dengan mengirimkan paket *error*. Paket ini tidak diakui, dan tidak dipancarkan (yaitu, *server* TFTP atau pengguna dapat menghentikan setelah mengirim pesan kesalahan), sehingga ujung sambungan mungkin tidak mendapatkannya. Oleh karena itu *timeout* digunakan untuk mendeteksi penghentian tersebut ketika paket *error* telah hilang. Kesalahan yang disebabkan oleh tiga jenis kegiatan: tidak mampu memenuhi permintaan (misalnya, *file* tidak ditemukan, pelanggaran akses atau tidak ada pengguna tersebut), menerima paket yang tidak dapat dijelaskan oleh keterlambatan atau duplikasi dalam jaringan

(misalnya, paket salah terbentuk), dan kehilangan akses ke sumber daya yang diperlukan (misalnya, disk atau akses ditolak selama transfer). (Sollins, K., 1992)

3.2 Server

Dalam jaringan komputer, *server* adalah komputer yang dirancang untuk memproses permintaan dan mengirimkan data ke komputer lain melalui jaringan lokal atau Internet. *Server* jaringan biasanya dikonfigurasi dengan pemroses, memori dan kapasitas penyimpanan tambahan untuk menangani beban permintaan klien. (Bradley, M., 2015)

3.2.1 Server HTTP

Koneksi Internet yang paling banyak digunakan oleh pengguna Internet adalah koneksi menggunakan protokol HTTP. Pengguna dapat memanfaatkan program klien HTTP (*browser*) seperti Netscape, Internet Explorer, Mozilla, Opera dan lain-lain untuk mempermudah koneksi HTTP.

Selain memerlukan klien HTTP, yang lebih penting lagi adalah bahwa untuk koneksi HTTP memerlukan program *server* HTTP. *Server* HTTP merupakan program yang dapat menerima koneksi HTTP serta melayani permintaan tersebut. Program ini akan bekerja untuk melayani seluruh permintaan koneksi HTTP yang ditujukan pada *server* dan mengirimkan tanggapan kepada klien HTTP. Program *server* HTTP akan selalu mendengarkan permintaan pada port nomor 80 yang merupakan port untuk koneksi HTTP.

Salah satu program *server* HTTP yang banyak digunakan adalah Apache yang dibuat oleh *Apache Software Foundation*. Program Apache merupakan standar program untuk *server* HTTP pada sistem operasi Linux. Program Apache berkembang sedemikian rupa, sehingga sampai versi terakhir sudah mendukung banyak hal yang diperlukan untuk koneksi HTTP. Versi terakhir program Apache dapat diperoleh dari situs http://www.apache.org dan dapat di*download* secara gratis.

Instalasi paket apache memerlukan paket lain untuk memenuhi dependensi paket. Paket yang diperlukan untuk memenuhi dependensi instalasi paket apache adalah lynx, libmm1, apache-common, apache-modules dan apache-conf.

Paket apache merupakan paket inti yang berisi program *daemon* httpd yang berlaku sebagai *server* HTTP. Paket lynx berisi program aplikasi klien HTTP atau *browser* mode teks yaitu program lynx. Program ini bekerja pada mode teks dan hanya dapat menampilkan teks. Walaupun sederhana, program lynx sangat berguna untuk menguji apakah *server* HTTP sudah bekerja secara benar. Paket libmm1 berisi pustaka MM yang berguna dalam menyederhanakan penggunaan memori secara berbagi antar cabang proses. Paket apache-common berisi *file-file* khusus yang digunakan secara bersama oleh beberapa paket termasuk terutama apache. Paket apache-modules berisi modul standar yang biasa digunakan untuk menjalankan *daemon* httpd. Paket apache-conf berisi *file* konfigurasi untuk jalannya *daemon* httpd dan beberapa program untuk administrasi *server* HTTP seperti apachectl.

Konfigurasi program Apache *server* HTTP diatur dalam beberapa *file*. *File* konfigurasi program Apache biasanya sudah diatur secara *default* sedemikian, sehingga *server* HTTP dapat langsung dijalankan. Konfigurasi *server* HTTP dengan Apache terutama diletakkan dalam dua direktori yaitu direktori /etc/httpd/ dan /var/www. Direktori /etc/httpd berisi *file* konfigurasi untuk mengatur jalannya *server* HTTP, sedangkan direktori /var/www digunakan untuk meletakkan dokumen yang akan ditampilkan oleh *server* HTTP.

Direktori /etc/httpd secara *default* sudah berisi *file-file* konfigurasi untuk menjalankan *daemon* httpd. *File* konfigurasi utama yang diletakkan di bawah direktori /etc/httpd/conf serta digunakan untuk mengatur jalannya *daemon* httpd adalah httpd.conf. *File* ini berisi pengarah-pengarah yang digunakan untuk mengatur jalannya *daemon* httpd. Beberapa konfigurasi penting httpd.conf disajikan pada paragraf berikut.

- ServerType: menentukan tipe *server* yang digunakan. Pengarah mempunyai dua nilai yaitu inetd apabila daemon httpd dijalankan dari *super internet daemon* (xinetd) dan standalone apabila *daemon* httpd dijalankan secara mandiri.
- ServerRoot: menentukan direktori teratas untuk menyimpan *file* konfigurasi, *file error* dan *file log*.
- ServerName: menentukan nama *server* yang akan dipakai untuk identitas *server* dan akan dikirimkan kepada klien HTTP. Nama ini harus merupakan nama valid sesuai dengan DNS *server*.

- DocumentRoot: menentukan tempat yang mana dokumen situs disimpan.
- Include: menyertakan *file* konfigurasi lain. Konfigurasi ini mungkin akan menimpa konfigurasi *default* pada httpd.conf.
- MaxClients : menentukan jumlah maksimal koneksi klien secara simultan.
- ServerAdmin: pengaturan alamat e-mail pengelola *server*.

3.2.2 Server HTTPS

Layanan koneksi HTTPS memerlukan keberadaan klien HTTPS dan *server* HTTPS. Klien HTTPS dapat berupa *browser* atau pun perangkat lunak khusus seperti Curl. *Server* HTTPS yang sekarang umum digunakan dan menjadi standar pada jaringan Internet adalah Nginx.

Nginx memiliki satu proses master dan beberapa proses pekerja. Tujuan utama dari proses master adalah membaca dan mengevaluasi konfigurasi, serta memelihara proses pekerja. Proses pekerja yang sebenarnya melakukan proses terhadap permintaan. Nginx mempekerjakan mekanisme model *event-based* dan *OS-dependent* efisien mendistribusikan permintaan antara proses pekerja. Jumlah pekerja proses ditentukan dalam *file* konfigurasi dan dapat tetap untuk konfigurasi tertentu atau secara otomatis disesuaikan dengan jumlah yang tersedia core CPU. (nginx documentation, 2015)

Cara Nginx dan modul-modulnya bekerja ditentukan dalam *file* konfigurasi. Secara *default*, *file* konfigurasi bernama nginx.conf dan ditempatkan pada direktori /usr/local/nginx/conf, /etc/nginx, atau /usr/local/etc/ nginx.

Nginx terdiri dari modul yang dikendalikan oleh pengarah yang ditentukan dalam *file* konfigurasi. Pengarah dibagi menjadi pengarah sederhana dan pengarah blok. Sebuah instruksi sederhana terdiri dari nama dan parameter yang dipisahkan oleh spasi dan diakhiri dengan tanda titik koma (;). Blok pengarah memiliki struktur yang sama sebagai pengarah sederhana, tapi bukan berakhir dengan titik koma tapi satu kumpulan instruksi tambahan yang dibatasi tanda { dan }. Jika blok pengarah memiliki pengarah lain di dalam batas, maka hal itu disebut konteks (contoh: *events*, http, *server*, dan lokasi).

Pengarah yang ditempatkan dalam *file* konfigurasi di luar setiap konteks dianggap dalam konteks utama. *Event* dan pengarah http berada dalam konteks utama, *server* http sedangkan lokasi di dalam *server*.

Nginx dapat dikonfigurasi sebagai *server* HTTP atau pun *server* HTTPS. Untuk konfigurasi *server* HTTPS, parameter ssl harus diaktifkan pada soket yang mana *server* mendengarkan pada blok *server* dan lokasi sertifikat *server* dan *file* kunci pribadi harus ditentukan: (nginx documentation, 2015)

```
server {
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:NULL:MD5;
    ...
}
```

Sertifikat server *adalah* entitas publik. Hal ini dikirim ke setiap klien yang terhubung ke *server*. Kunci privat adalah badan yang aman dan harus disimpan dalam *file* dengan akses terbatas, namun, harus dapat dibaca oleh master proses

Nginx. Kunci privat bisa juga disimpan dalam *file* yang sama dengan sertifikat:

```
ssl_certificate     www.example.com.cert;
ssl_certificate_key www.example.com.cert;
```

dalam hal ini hak akses *file* juga harus dibatasi. Meski pun sertifikat dan kunci disimpan dalam satu *file*, hanya sertifikat yang dikirim ke klien.

Pengarah ssl_protocols dan ssl_ciphers dapat digunakan untuk membatasi koneksi untuk menyertakan hanya versi yang kuat dan cipher SSL / TLS. Sejak versi 1.0.5, Nginx menggunakan "ssl_protocols SSLv3 TLSv1" dan "ssl_ciphers HIGH:NULL:MD5" secara *default*, sehingga konfigurasi secara eksplisit hanya masuk akal untuk versi Nginx sebelumnya. Sejak versi 1.1.13 dan 1.0.12, Nginx menggunakan "ssl_protocols SSLv3 TLSv1 menggunakan TLSv1.1 TLSv1.2" secara *default*.

3.2.3 Server FTP

Untuk jaringan komputer, proses transfer *file* salah satunya dapat ditangani melalui protokol FTP. Supaya dapat melakukan proses transfer *file* dengan protokol FTP, *user* harus masuk pada suatu *server* melalui proses autentikasi yang biasanya menggunakan nama *user* dan *password* tertentu. Setelah berhasil masuk pada sistem *server*, yang dapat dilakukan tidak hanya sekadar transfer *file* tetapi dapat melakukan aktivitas lain yang diizinkan oleh protokol FTP seperti membuat direktori, menghapus direktori, menghapus *file*, mengirim *file*, *download file* dan sebagainya tergantung izin yang diberikan.

Yang diperlukan untuk proses transfer *file* melalui protokol FTP, bagi *server* yang harus memberikan layanan adalah program *server* FTP, sedangkan bagi klien yang akan melakukan koneksi adalah program klien FTP. *Server* FTP adalah program yang dapat melayani permintaan koneksi melalui protokol FTP. Klien FTP adalah program yang dapat digunakan untuk melakukan koneksi melalui protokol FTP. Salah satu program untuk *Server* FTP adalah ProFTPD. Untuk aplikasi klien FTP variasinya cukup banyak baik yang mode teks maupun mode grafis seperti gFTP.

ProFTPD (*Professional FTP Daemon*) merupakan program *server* FTP yang dibuat untuk beberapa varian Unix termasuk juga Linux. Kode program untuk ProFTPD dapat diperoleh dari situs http://www.proftpd.org dan dapat di*download* dari situs tersebut. Dari situs tersebut dapat diperoleh versi terbaru ProFTPD dan dapat dikompilasi dengan mudah.

Program ProFTPD dikemas dalam paket proftpd. Paket proftpd berisi semua program yang diperlukan untuk menjalankan dan mengelola server FTP seperti proftpd, ftpshut, ftpcount, ftptop dan ftpwho. Program utama yang dikemas dalam paket tersebut adalah proftpd (proftp *daemon*) yang merupakan program yang berlaku sebagai *server* FTP. Selain itu juga sudah disediakan file untuk konfigurasi dasar proftpd, sehingga bisa langsung dijalankan secara mudah.

ProFTPD dibuat sedemikian, sehingga mudah untuk dikonfigurasi. Konfigurasi ProFTPD diatur dalam dua file yaitu proftpd.conf dan ftpusers yang keduanya berada di bawah direktori /etc. *File* proftpd.conf berisi pengaturan

konfigurasi untuk jalannya *daemon* proftpd, sedangkan *file* ftpusers digunakan untuk mencatat semua *user* yang tidak boleh digunakan untuk *login* melalui perangkat lunak klien FTP.

Secara *default file* konfigurasi /etc/proftpd.conf sudah disediakan dan dapat digunakan sebagai konfigurasi untuk menjalankan *daemon* proftpd. Konfigurasi penting dalam *file* proftpd.conf dituliskan pada paragraf berikut ini.

- ServerName: menentukan nama *server* yang digunakan. Nama ini dapat berupa keterangan yang dibatasi oleh tanda petik ganda dan akan muncul pada saat *user* masuk dalam *server* FTP.
- ServerType: menentukan tipe *server* yang digunakan. Pengarah ServerType dapat mempunyai dua kemungkinan nilai yaitu standalone yang mana *daemon* proftpd dijalankan secara mandiri atau inetd yang mana *daemon* proftpd dijalankan dari program *super server* xinetd.
- DefaultServer: mengendalikan konfigurasi *server* yang digunakan ketika suatu koneksi ditujukan untuk suatu alamat IP tertentu yang bukan alamat IP primer dan alamat IP *virtual host*. Jika bernilai on, maka semua layanan FTP menggunakan konfigurasi *server default*.
- AllowStoreRestart: mengizinkan atau menolak *resume* terhadap transfer *file* ke *server* (*upload*). Jika pengarah ini bernilai on, maka *resume upload file* diizinkan.
- Port: menentukan nomor *port* yang digunakan untuk koneksi FTP. Secara *default* bernilai 21.

- Umask: menentukan mask *default* pada saat *user* membuat *file* atau direktori pada *server*.
- MaxInstances: menentukan jumlah proses anak (*thread*) yang diizinkan untuk dibuat. pengarah ini akan menentukan jumlah *user* yang dapat masuk melalui koneksi FTP dari klien. Pengarah ini hanya berlaku apabila *daemon* proftpd dijalankan dengan mode standalone. Jika dijalankan dengan mode inetd, maka pengaturan ini harus dilakukan dari xinetd.
- AllowOverwrite: mengatur apakah *file* yang sudah ada diizinkan untuk ditimpa dengan *file* yang ditransfer.
- PersistentPasswd: menentukan bagaimana cara *daemon* proftpd menangani autentikasi. Jika bernilai on, maka *daemon* proftpd akan berusaha untuk membuka *file* /etc/passwd dan /etc/group.

3.2.4 Server SSH

Program sshd (SSH *Daemon*) adalah *daemon* untuk program ssh (*secure shell*). Program ssh digunakan untuk *login* dalam suatu komputer dari jauh (secara *remote*) dan mengeksekusi perintah pada komputer tersebut. Program ssh ini mirip dengan telnet, tetapi punya pengendalian terhadap keamanan. Program ssh menyediakan komunikasi aman terenkripsi antara dua *host* yang tidak saling kenal melalui jaringan yang umumnya tidak aman. Dengan alasan keamanan, fungsi program telnet sudah mulai ditinggalkan diganti dengan ssh.

Program sshd adalah *daemon* yang menunggu koneksi dari klien pada port

22. Program ini akan membuat cabang *daemon* baru untuk tiap koneksi yang datang. Masing-masing *daemon* menangani secara mandiri pertukaran kunci, enkripsi, autentikasi, eksekusi perintah dan pertukaran data. Penerapan program sshd mendukung protokol SSH versi 1 dan versi 2 secara serentak.

Pada SSH versi 1, masing-masing host punya kunci RSA tertentu (umumnya 1024 *bit*) yang digunakan untuk mengenali host. Ketika suatu *daemon* muncul (akibat permintaan koneksi), daemon akan menghasilkan sebuah kunci RSA *server* (umumnya 768 *bit*). Kunci ini umumnya diperbarui tiap jam apabila digunakan dan tidak pernah disimpan dalam *disk*.

Ketika suatu klien menghubungi *server*, *daemon* akan menanggapi dengan kunci publik *server* dan *host* yang dimiliki. Klien akan membandingkan kunci RSA *host* dengan simpanan yang dimiliki untuk menentukan apakah kunci tersebut tidak berubah. Kemudian klien akan menciptakan angka random 256 *bit*. Klien akan melakukan enkripsi angka random tersebut menggunakan kunci *host* dan kunci server serta mengirimkan angka terenkripsi tersebut ke server. Klien dan *server* selanjutnya menggunakan angka random tersebut sebagai kunci sesi yang selanjutnya digunakan untuk enkripsi komunikasi pada sesi tersebut.

Selanjutnya klien dan *server* masuk pada dialog autentikasi. Klien mencoba untuk autentikasi sendiri menggunakan autentikasi *rhost*. Autentikasi *rhost* dikombinasikan dengan autentikasi RSA host, autentikasi RSA *challenge-response* atau autentikasi berdasar password. Autentikasi *rhost* biasanya dibuat *disable* karena umumnya tidak aman, tetapi dapat dibuat *enable* jika diinginkan.

Pada protokol SSH versi 2 juga terdapat proses yang mirip dengan protokol versi 1. Masing-masing host punya kunci RSA dan DSA tertentu yang digunakan untuk mengenali suatu *host*. Perbedaannya, pada saat suatu *daemon* diciptakan (karena ada permintaan koneksi), *daemon* tidak menciptakan kunci *server*. Keamanan disediakan dengan suatu perjanjian kunci Diffie-Hellman. Kunci ini akan menghasilkan suatu kunci sesi yang digunakan bersama.

Protokol versi 2 dilengkapi dengan metode autentikasi kunci publik berbasis *user* (*PubkeyAuthentication*) atau *host* klien (*HostbaseAuthentication*), autentikasi *password* dan metode berbasis *challenge-response*. Dengan cara seperti di atas, koneksi SSH menjadi relatif aman karena semua data koneksi dienkripsi dengan algoritme tertentu.

Salah satu program *server* SSH adalah openssh. OpenSSH dibuat oleh Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt dan Dug Song. OpenSSH memerlukan beberapa paket lain yaitu Zlib, openSSL dan secara opsional memerlukan PAM.

Konfigurasi server SSH disimpan dalam direktori /etc/ssh/ baik *file* konfigurasi sshd_config maupun *file-file* kunci autentikasi. *File* sshd_config digunakan untuk menyimpan konfigurasi *server* SSH. Secara *default*, *file* sshd_config sudah diatur sedemikian, sehingga *daemon* sshd dapat berjalan secara benar. Namun demikian dapat diatur sesuai dengan keinginan. Berikut ini diberikan beberapa opsi yang sering diatur dalam *file* konfigurasi tersebut.

- Port: menentukan nomor *port* yang mana *daemon* sshd mendengarkan,

dimungkinkan memberi lebih dari satu nomor *port*. *Default* bernilai 22.

- Protocol: menentukan versi protokol yang digunakan *daemon* sshd. Secara *default daemon* sshd menggunakan versi 2 dan 1 dan ditulis 2,1.
- ListenAddress: menentukan alamat IP antarmuka untuk mendengarkan koneksi SSH. Jika tidak ditentukan, maka *daemon* sshd akan mendengarkan pada seluruh antarmuka.
- HostKey: menentukan *file* untuk menyimpan kunci RSA dan DSA.
- PermitRootLogin: menentukan apakah *account* root diizinkan *login* atau tidak. *Default* bernilai yes.
- PermitEmptyPasswords: menentukan apakah *password* kosong diizinkan. *Default* bernilai no.

3.2.5 Server TFTP

Tftpd adalah *server* untuk *Trivial File Transfer Protocol*. Protokol TFTP banyak digunakan untuk mendukung *booting* perangkat *remote diskless*. *Server* biasanya dimulai oleh inetd, tetapi juga dapat dijalankan mandiri.

Penggunaan layanan TFTP tidak memerlukan *account* atau *password* pada sistem *server*. Karena kurangnya informasi autentikasi, tftpd akan memungkinkan hanya file dibaca publik (o+r) untuk diakses, kecuali opsi --permissive ditentukan. *File* dapat ditulis hanya jika mereka sudah ada dan dapat ditulis secara terbuka, kecuali opsi --create ditentukan. Catatan bahwa ini memperluas konsep publik untuk memasukkan semua pengguna di semua *host* yang dapat dicapai melalui

kerja jaringan; ini mungkin tidak sesuai pada semua sistem dan implikasinya harus dipertimbangkan sebelum mengaktifkan layanan TFTP. Biasanya, beberapa jenis *Firewall* atau solusi *packet-filter* harus digunakan. Jika tepat disusun (dapat dilihat pada *output* in.tftpd --version) tftpd akan *query hosts_access database* untuk informasi kontrol akses. Ini mungkin lambat; situs yang membutuhkan kinerja maksimum mungkin ingin kompilasi tanpa pilihan ini dan bergantung pada Firewall atau paket berbasis *kernel-filte*r gantinya.

Untuk menjalankan *server* Tftpd digunakan bantuan *server* Xinetd. Oleh sebab itu, konfigurasi Tftpd menyatu dengan *server* Xinetd, Konfigurasi Tftpd diatur pada *file* /etc/xinetd.d/tftp. Dengan menghilangkan yang tidak penting, isi file tftp adalah sebagai berikut.

```
service tftp
{
     disable                 = no
     socket_type             = dgram
     protocol                = udp
     wait                    = yes
     user                    = root
     server                  = /usr/sbin/in.tftpd
     server_args             = -s /var/lib/tftpboot
     per_source              = 11
     cps                     = 100 2
     flags                   = IPv4
}
```

Beberapa pengaturan penting dalam konfigurasi *server* Tftpd dapat diuraikan sebagai berikut.

- disable: menentukan apakah layanan akan disable atau tidak.
- socket_type: menentukan tipe socket.
- protocol: menentukan protokol yang digunakan oleh layanan.

- wait: menentukan apakah layanan berupa *single-threaded* (yes) atau *multi-threaded* (no).
- user: menentukan user id yang menjalankan proses.
- group: menentukan group id yang menjalankan proses.
- instances: menentukan jumlah layanan yang dapat aktif secara simultan.
- server: menentukan program untuk menjalankan layanan.
- server_args: menentukan argumen yang dilewatkan pada server.
- port: menentukan nomor port untuk layanan.
- redirect: mengizinkan layanan TCP untuk diarahkan ke host lain.
- bind: mengizinkan pembatasan suatu layanan melalui antarmuka tertentu.
- interface: sama dengan bind.
- per_source: menentukan jumlah instan maksimum tiap alamat IP.
- cps: membatasi laju koneksi yang masuk.
- umask: mengatur UMASK yang diturunkan dari layanan.
- enabled: menentukan apakah layanan akan dibuat enable.

3.3 Curl

Curl adalah perangkat lunak untuk mentransfer data dari atau ke *server*, menggunakan salah satu protokol yang didukung (HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP atau FILE). Perintah ini dirancang

untuk bekerja tanpa interaksi pengguna. Curl menawarkan fitur-fitur yang berguna seperti dukungan *proxy*, autentikasi pengguna, *upload* FTP, HTTP *post*, koneksi SSL, *cookies*, *resume* transfer *file* dan banyak lagi. Curl ini didukung oleh libcurl untuk semua fitur transfer terkait.

Sintaksis URL tergantung protokol. Penjelansan rinci dapat ditemukan dalam RFC 3986. *User* dapat menentukan beberapa URL atau bagian dari URL dengan menulis bagian *set* dalam kurung seperti:

```
http://site.{one,two,three}.com
```

User dapat juga mendapatkan urutan seri alfanumerik dengan menggunakan tanda [ ] seperti dalam:

```
ftp://ftp.numericals.com/file[1-100].txt
ftp://ftp.numericals.com/file[001-100].txt (dengan awalan nol)
ftp://ftp.letters.com/file[a-z].txt
```

Tidak ada urutan bersarang yang didukung pada saat ini, tetapi *user* dapat menggunakan beberapa yang bersebelahan:

```
http://any.org/archive[1996-1999]/vol[1-4]/part{a,b,c}.html
```

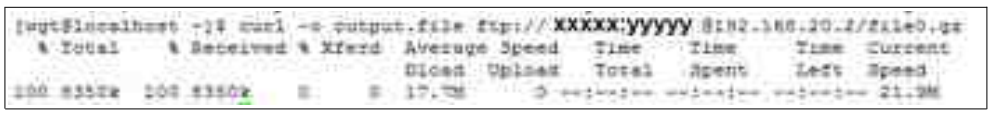
User dapat menentukan setiap jumlah URL pada baris perintah. Perintah akan diambil secara berurutan dalam urutan tertentu. Pada Curl versi 7.15.1 *user* juga dapat menentukan langkah *counter* untuk rentang, sehingga *user* bisa mendapatkan setiap nomor N atau surat:

```
http://www.numericals.com/file[1-100:10].txt
http://www.letters.com/file[a-z:2].txt
```

Jika *user* menetapkan URL tanpa tulisan `protokol://`, Curl akan mencoba menebak apa protokol yang diinginkan *user*. Secara *default* akan

menggunakan HTTP tetapi bisa dicoba protokol lain berdasarkan frekuensi penggunaan prefiks nama *host*. Sebagai contoh, untuk nama *host* dimulai dengan "ftp." akan diarahkan untuk menggunakan protokol FTP. Pengarah akan melakukan yang terbaik untuk menggunakan apa yang diinginkan *user*.

Curl akan mencoba untuk menggunakan kembali koneksi untuk beberapa transfer *file*, sehingga semakin banyak *file* dari *server* yang sama tidak akan melakukan beberapa gandengan. Hal ini meningkatkan kecepatan. Tentu saja hal ini hanya dilakukan pada *file* yang ditentukan pada baris perintah tunggal dan tidak dapat digunakan antara memanggil Curl terpisah.



Gambar 3.5 Tampilan Curl

Curl biasanya menampilkan ukuran kemajuan (*progress*) selama operasi, menunjukkan jumlah data yang ditransfer, kecepatan transfer dan perkiraan waktu yang tersisa, dan lain-lain. Namun, karena tampilan Curl data ini ke terminal secara *default*, jika *user* memanggil Curl untuk melakukan operasi dan itu adalah tentang untuk menulis data ke terminal, Curl mematikan meteran kemajuan karena kalau tidak itu akan mengacaukan keluaran pencampuran ukuran kemajuan dan respon data. Jika diinginkan ukuran kemajuan untuk permintaan HTTP *post* atau *put*, *user* harus mengarahkan keluaran respon ke suatu *file*, menggunakan *redirect*

shell seperti (>), -o [file] atau sejenisnya.

BAB 4 METODE PENELITIAN

4.1 Bahan Penelitian

Bahan yang digunakan dalam penelitian berupa *file* yang punya tipe teks dan gz. *file* teks mewakili jenis *file* yang belum mengalami kompresi, sedangkan *file* gz mewakili *file* yang sudah mengalami kompresi. Dua jenis *file* tersebut dianggap dapat mewakili banyak tipe lain.

Ukuran *file* yang dipakai dalam penelitian meliputi bermacam-macam ukuran. Ukuran *file* dipilih sedemikian, sehingga waktu yang diperlukan untuk proses *transfer* cukup untuk diukur. Ukuran *file* juga dipilih sedemikian, sehingga waktu *download file* dapat mengabaikan waktu yang dipakai untuk proses *login*. Sebelumnya dilakukan uji coba awal untuk pemilihan dan penentuan seberapa kira-kira ukuran *file* yang sesuai.

4.2 Alat

Alat yang digunakan pada penelitian berupa perangkat keras dan perangkat lunak. Perangkat lunak yang digunakan dalam penelitian adalah sebagai berikut.

- Sistem Operasi Linux Mandriva 2010.
- Perangkat lunak virtualisasi Virtual Box OSE 4.0.6

- Perangkat lunak *download* Curl 7.19.6
- *Server* FTP Proftpd 1.3.3
- *Server* Web Apache 2.2.4
- *Server* HTTPS Nginx 0.8.17
- *Server* SSH OpenSSH 5.3
- *Server* TFTP Tftpd 5.0.3

Selain perangkat lunak yang sudah disebutkan, pada penelitian masih diperlukan perangkat lunak pendukung yaitu: bash, cat, tail, awk dan mcedit. Perangkat lunak pendukung digunakan dalam proses pencuplikan data.

Perangkat keras yang digunakan dalam penelitian adalah komputer personal dengan spesifikasi sebagai berikut.

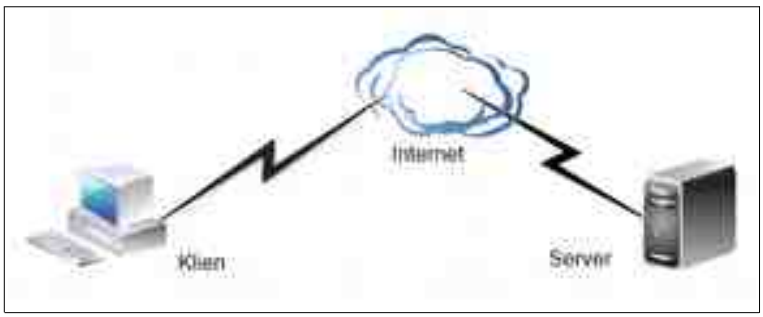
- Prosesor Intel Pentium(R) *Dual-Core* CPU E5700 @ 3.00GHz
- RAM 1 GB.
- Tipe sistem 32 bit.

4.3 Jalan Penelitian

4.3.1 Rancangan Perangkat Keras

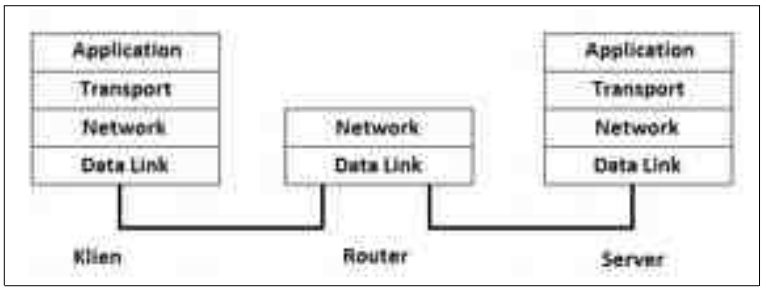
Pada penelitian terlibat satu komputer sebagai klien, satu komputer sebagai *server* dan sistem jaringan. Komputer klien berfungsi untuk menjalankan perangkat lunak *download file*. Komputer *server* berfungsi untuk menjalankan perangkat lunak *server*. *Cloud* (awan) menggambarkan suatu sistem jaringan

Internet yang menghubungkan Klien dengan *Server*. Diagram jaringan yang akan diterapkan untuk penelitian ditunjukkan pada Gambar 4.1.



Gambar 4.1 Jaringan Server dan Klien

Model TCP/IP digunakan untuk menjelaskan komunikasi data yang berkaitan dengan diagram jaringan pada Gambar 4.1. Secara skematis, model TCP/IP untuk menjelaskan komunikasi data sistem jaringan pada Gambar 4.1 ditampilkan pada Gambar 4.2.



Gambar 4.2 Model TCP/IP *Server* dan Klien

Jumlah *layer* yang dialami oleh paket data pada masing-masing komputer

berbeda-beda. Pada komputer Klien dan *server*, terdapat empat *layer* yang dialami oleh paket data. *Layer* tersebut adalah *link*, *network*, *transport* dan *application*. Pada *server* sebetulnya bisa tidak melewati *layer application* pada saat *download file* karena pengguna tidak terlibat pada sisi *server*. Namun demikian, supaya dapat memberi layanan proses *download file*, komputer *server* perlu dikonfigurasi oleh pengguna.

Sistem jaringan yang terdapat pada Gambar 4.1 dapat dianggap sebagai suatu komputer *router*. Dengan demikian, pada saat proses *download*, paket data akan melewati dua *layer*. *Layer* tersebut adalah *link* dan *network*. Namun demikian, apabila pada sistem jaringan terdapat komputer *Firewall*, maka paket data dapat mencapai *layer transport*.

4.3.2 Rancangan Perangkat Lunak

Komputer Klien menggunakan sistem operasi Linux Mandriva 2010. Pada komputer Klien dipasang perangkat lunak untuk keperluan *download file* serta mengamati kecepatan *download* yang diperoleh. Perangkat lunak yang dipasang pada komputer Klien terutama adalah Curl. Perangkat lunak Curl dapat menjalankan fungsi *download* menggunakan berbagai protokol jaringan antara lain HTTP, HTTPS, FTP, SFTP, SCP dan TFTP.

Komputer *Server* juga menggunakan sistem operasi Linux Mandriva 2010. Pada komputer *Server* dipasang perangkat lunak untuk memberi layanan *download file*. Pada komputer *Server* dipasang beberapa perangkat lunak *server*

yaitu: Apache sebagai *server* HTTP, Nginx sebagai *server* HTTPS, ProFTPD sebagai *server* FTP, OpenSSH sebagai *server* SSH (sebagai penyedia layanan protokol SCP) dan Tftpd sebagai *server* TFTP.

Komputer *Router* menggunakan sistem operasi Linux Mandriva 2010. Pada komputer *Router* tidak dipasang perangkat lunak khusus karena sistem operasi Linux secara bawaan sudah mendukung fungsi *router*. Komputer *Router* hanya bersifat meneruskan paket data yang melewatinya.

4.3.3 Rancangan Tahapan Penelitian

Penelitian dilakukan dalam beberapa tahapan. Masing-masing tahapan akan dibahas pada bagian selanjutnya. Tahapan-tahapan yang dilakukan pada penelitian ini dapat diuraikan sebagai berikut.

- Menyiapkan komputer *Server*
- Menyiapkan komputer Klien
- Menyiapkan komputer Router
- Konfigurasi Mandriva 2010 pada komputer *Server*
- Konfigurasi Mandriva 2010 pada komputer *Router*
- Konfigurasi Mandriva 2010 pada komputer Klien
- Menyiapkan *file* percobaan pada komputer *Server*
- Konfigurasi *server* HTTP
- Konfigurasi *server* HTTPS
- Konfigurasi *server* FTP

- Konfigurasi *server* SSH
- Konfigurasi *server* TFTP
- Instalasi perangkat lunak *download* Curl pada Klien
- Menyusun program sekrip untuk mencuplik kecepatan *download file*
- Percobaan *download file* dengan berbagai macam ukuran menggunakan beberapa protokol jaringan
- Percobaan *download file* dengan berbagai macam tipe menggunakan beberapa protokol jaringan
- Menghitung rata-rata kecepatan.
- Menampilkan grafik hubungan antara kecepatan *download*, ukuran *file* dan protokol jaringan
- Menampilkan grafik hubungan antara kecepatan *download*, tipe *file* dan protokol jaringan.
- Analisis terhadap hasil percobaan *download file*.

Konfigurasi jaringan melibatkan sistem pengkabelan. Faktor pengkabelan bisa menjadi sesuatu yang sulit diperhitungkan dalam *download file*. Pemasangan kabel yang tidak sempurna menyebabkan kecepatan *download file* yang diamati menjadi tidak sesuai. Faktor lain yang berkaitan dengan pengkabelan adalah kualitas kabel. Kualitas kabel yang tidak sesuai spesifikasi juga berpengaruh pada kecepatan *download file.*

Berkaitan dengan faktor pengkabelan adalah faktor perangkat keras kartu

jaringan. Kartu jaringan yang tidak bekerja dengan baik menyebabkan pengamatan terhadap kecepatan *download file* tidak sesuai. Dalam praktiknya cukup sulit untuk menentukan apakah suatu kartu jaringan masih bekerja dengan baik atau tidak.

Penelitian ini mencoba mengabaikan faktor yang muncul dari perangkat keras yaitu pengkabelan dan kartu jaringan yang tidak bekerja dengan sempurna. Diharapkan faktor yang berpengaruh hanya muncul dari pertimbangan perangkat lunak. Faktor perangkat lunak yang dimaksud dalam hal ini adalah faktor protokol jaringan.

Asumsi ini menjadikan faktor yang dimunculkan dari *layer link* model TCP/IP diabaikan. Pengabaian faktor pada *layer link* menyebabkan hasil kecepatan *transfer* data yang terukur hanya dipengaruhi oleh *layer network*, *layer transport* dan *layer application*. Ketiga *layer* tersebut berkaitan dengan pengaruh perangkat lunak pada kecepatan *transfer* data. Perangkat lunak yang berpengaruh dalam pengertian ini adalah sistem operasi serta program aplikasi.

Pada penelitian ini digunakan perangkat lunak Virtual Box untuk menjalankan komputer *Server*, Klien dan sistem Jaringan. Penggunaan Virtual Box dapat mengabaikan *layer link* model TCP/IP dalam perhitungan kecepatan *transfer* data. Dengan demikian, semua kerumitan perhitungan kecepatan *transfer* data yang dipengaruhi faktor perangkat keras dapat diabaikan.

BAB 5 IMPLEMENTASI DAN PEMBAHASAN

5.1 Implementasi

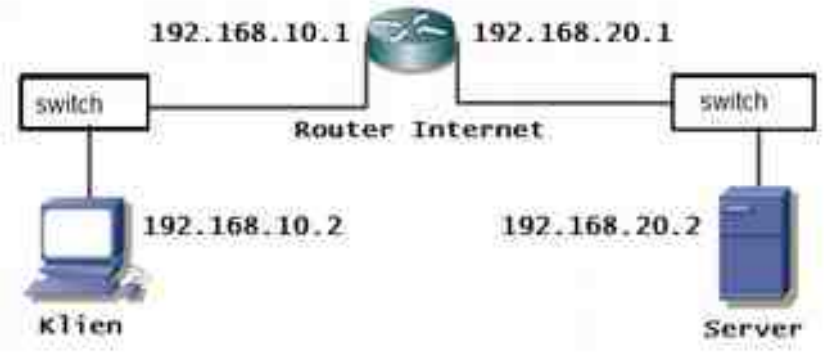
Konfigurasi jaringan yang diperlukan dalam penelitian ini melibatkan satu *server*, satu klien, satu *router* dan sistem pengkabelan. Pada penelitian ini, sistem pengkabelan diabaikan karena komputer *server* dan *router* dijalankan pada VirtualBox. *Server* dan *router* dijalankan sebagai komputer *virtual*, sedangkan klien dijalankan dari komputer *real*. Untuk menjalankan komputer *server* dan *router* digunakan perintah

```
$ VBoxManage startvm server --type headless
$ VBoxManage startvm internet --type headless
```

Perintah pertama digunakan untuk menjalankan komputer *virtual server*, sedangkan perintah kedua digunakan untuk menjalankan komputer *virtual router*. Perintah yang digunakan untuk menjalankan *server* maupun *router* dapat diterapkan pada *shell user*.

Pengaturan *device network* pada komputer server menggunakan *Internal Network* dan dihubungkan pada *switch* pertama. Pengaturan *device network* pertama pada *router* menggunakan *Internal Network* dan dihubungkan pada *switch* pertama, sedangkan *device network* kedua menggunakan *Host-only Adapter*. Pengaturan ini dan alamat IP sesuai dengan diagram jaringan yang

ditunjukkan pada Gambar 5.1.



Gambar 5.1 Rancangan Jaringan

Server berlaku sebagai *server* HTTP, HTPS, FTP, SFTP, SCP dan TFTP yang melayani proses *download file*. Antarmuka jaringan pada *server* diberi alamat IP 192.168.20.2/24. Klien berlaku sebagai tempat untuk mengukur kecepatan *download file*. Antarmuka pada klien diberi alamat IP 192.168.10.2/24. Router berlaku sebagai simulasi jaringan Internet. Supaya tidak terjadi proses translasi alamat jaringan, pada *router* tidak diberlakukan sebagai *Firewall*. Hal demikian dimaksudkan untuk meminimalkan pengaruh selain faktor tipe protokol. Pada *router* hanya terjadi proses *routing* paket. Antarmuka pada *router* yang berada satu jaringan dengan klien diberi alamat IP 192.168.10.1/24, sedangkan antarmuka pada *router* yang berada satu jaringan dengan komputer *server* diberi alamat IP 192.168.20.1/24.

5.1.1 Konfigurasi Klien

Komputer klien digunakan untuk mengukur kecepatan *download file*. Komputer klien menggunakan sistem operasi Linux *distro* Mandriva 2010. Pada komputer klien dipasang perangkat lunak Curl. Perangkat lunak Curl digunakan untuk proses *download file.* Keuntungan menggunakan perangkat lunak Curl adalah: dapat digunakan untuk proses *download file* secara *command line,* dapat menampilkan kecepatan transfer *file* yang digunakan, dapat mendukung proses *download file* menggunakan beberapa tipe protokol dan dapat digabungkan dalam suatu sekrip program. Secara keseluruhan, perangkat lunak Curl dapat memudahkan proses pengambilan data.

Komputer klien terhubung dengan *router* melalui suatu *switch*. Komputer klien diberi alamat IP 192.168.10.2/24. Alamat IP *Gateway* untuk komputer klien seharusnya 192.168.10.1/24, namun karena pada percobaan digunakan VirtualBox untuk menjalankan *server*, pada komputer klien cukup dibuat *routing* menuju *server*. Pemberian alamat IP pada komputer klien diatur pada *file* /etc/sysconfig/network-scripts/ifcfg-eth0 yang isinya adalah sebagai berikut.

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.10.2
NETMASK=255.255.255.0
GATEWAY=192.168.10.1
ONBOOT=yes
METRIC=10
MII_NOT_SUPPORTED=no
USERCTL=no
RESOLV_MODS=no
IPV6INIT=no
IPV6TO4INIT=no
ACCOUNTING=no
```

Hasil konfigurasi alamat IP adalah sebagai berikut.

```
# ifconfig eth0

eth0 Link encap:Ethernet  HWaddr 08:00:27:68:50:A7
     inet addr:192.168.10.2 Bcast:192.168.10.255 Mask:255.255.255.0
     inet6 addr: fe80::a00:27ff:fe68:50a7/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:227 errors:0 dropped:0 overruns:0 frame:0
     TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:21708 (21.1 KiB)  TX bytes:19401 (18.9 KiB)
```

Hasil disunting untuk menghilangkan *Shell Linux* [root@localhost] dan merapikan tampilan tanpa mengurangi inti isinya.

Pemberian tabel routing menuju server pada klien menggunakan perintah sebagai berikut.

```
# route add -net 192.168.20.0/24 gw 192.168.10.1
```

Perintah tersebut harus dijalankan pada *shell superuser*. Maksud perintah tersebut adalah bahwa paket data yang menuju jaringan 192.168.20.0/24 dilewatkan melewati *Gateway* yang punya alamat IP 192.168.10.1 (alamat IP pada antarmuka kedua *router*)

5.1.2 Konfigurasi Server

Komputer *server* disiapkan untuk melayani permintaan *download* dari komputer klien. Komputer *server* diatur supaya berada pada jaringan yang sama dengan komputer klien. Hal demikian untuk mengurangi pengaruh translasi alamat jaringan pada proses *download*. Diharapkan proses *download* hanya dipengaruhi oleh perbedaan tipe protokol yang dipakai.

Komputer *server* menggunakan sistem operasi Linux *distro* Mandriva 2010. Komputer *Server* diberi alamat IP 192.168.20.2/24 dan alamat IP *Gateway*

192.168.20.1/24. Pemberian alamat IP pada komputer *Server* diatur pada *file* /etc/ sysconfig/network-scripts/ifcfg-eth0 yang isinya adalah sebagai berikut.

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.20.2
NETMASK=255.255.255.0
GATEWAY=192.168.20.1
ONBOOT=yes
METRIC=10
MII_NOT_SUPPORTED=no
USERCTL=no
RESOLV_MODS=no
IPV6INIT=no
IPV6TO4INIT=no
ACCOUNTING=no
```

Hasil konfigurasi alamat IP pada *server* adalah sebagai berikut.

```
# ifconfig eth0

eth0   Link encap:Ethernet  HWaddr 08:00:27:5C:A1:27
       inet addr:192.168.20.2  Bcast:192.168.20.255  Mask:255.255.255.0
       inet6 addr: fe80::a00:27ff:fe5c:a127/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:174 errors:0 dropped:0 overruns:0 frame:0
       TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:17906 (17.4 KiB)  TX bytes:12905 (12.6 KiB)
```

Hasil konfigurasi alamat IP *Gateway* dapat dilihat pada tampilan tabel *routing* komputer *server* sebagai berikut.

```
# route -n

Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref   Use Iface
192.168.20.0  0.0.0.0       255.255.255.0   U     5      0     0   eth0
169.254.0.0   0.0.0.0       255.255.0.0     U     5      0     0   eth0
0.0.0.0       192.168.20.1  0.0.0.0         UG    5      0     0   eth0
```

Supaya kerja *server* lebih ringan, mode kerja yang diterapkan pada *server* adalah mode teks. Seluruh layanan bekerja pada mode teks dan bekerja secara latar. Layanan yang disiapkan pada *server* adalah layanan *download file*. Komputer *server* melayani permintaan *download file* menggunakan protokol HTTP, HTTPS, FTP, SFTP, SCP dan TFTP.

5.1.3 Konfigurasi Server HTTP

Layanan permintaan protokol HTTP ditangani oleh perangkat lunak Apache. Versi Apache yang digunakan pada penelitian ini adalah 2.2.4. Versi ini merupakan bawaan dari Linux Mandrva 2010.

Konfigurasi Apache dituliskan dalam *file* /etc/httpd/conf/httpd.conf. Isi *file* tersebut secara lengkap dapat dilihat pada Lampiran. Konfigurasi Apache dapat menggunakan nilai-nilai default. Namun ada sedikit modifikasi untuk pemberian nama *server* melalui pengarah ServerName. Ada beberapa konfigurasi penting berkaitan dengan layanan *server* HTTP.

```
.............
.............
ServerName   server
Listen       80
ServerRoot "/etc/httpd"
.............
.............
DocumentRoot "/var/www/html"
<Directory "/var/www/html">
    Options -Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
.............
.............
```

Pengarah ServerName digunakan untuk menentukan nama *server* HTTP. Nama ini harus dicantumkan pada file /etc/hosts. Apabila pengarah ServerName tidak ditentukan, maka akan menimbulkan pesan peringatan pada saat *server* HTTP dijalankan. Meski pun tidak terlalu berpengaruh pada jalannya *server*, namun lebih baik pengarah ini perlu ditentukan.

Pengarah Listen digunakan untuk menentukan nomor *port* kerja *server*

HTTP. Nomor *port* secara *default* untuk *server* HTTP adalah 80. Apabila nomor *port* tidak ditentukan dengan nilai *default* 80, maka pada saat akses dari klien harus ditentukan secara eksplisit berapa nilai *port* yang ditentukan.

Pengarah ServerRoot digunakan untuk menentukan letak direktori tempat konfigurasi *server* HTTP. Umumnya, konfigurasi *server* HTTP Apache diletakkan pada direktori /etc/httpd. *File* konfigurasi httpd.conf juga diletakkan pada direktori conf yang merupakan sub direktori.

Pengarah DocumentRoot digunakan untuk menentukan letak dokumen yang akan diakses. Pada konfigurasi ini, dokumen dirancang untuk diletakkan pada direktori /var/www/html. Karena nantinya akan diakses dari luar *server*, akses direktori /var/www/html harus diatur lebih lanjut. Pengaturan pada direktori ini ditentukan pada pengarah <Directory "/var/www/html">.

5.1.4 Konfigurasi Server HTTPS

Layanan permintaan protokol HTTPS ditangani oleh perangkat lunak Nginx. Versi Nginx yang digunakan pada penelitian ini adalah 0.8.17. Nginx merupakan perangkat lunak yang umum digunakan untuk menyediakan layanan HTTPS.

Konfigurasi Nginx diatur pada dua *file* utama yaitu /etc/nginx/nginx.conf dan /etc/nginx/conf.d/ssl.conf. Konfigurasi pada nginx.conf dibiarkan default. Isi konfigurasi nginx.conf dan ssl.conf secara lengkap disertakan pada Lampiran. Beberapa konfigurasi penting *server* HTTPS Nginx adalah sebagai berikut.

```
.............
.............
Http {
  .............
  .............
  server {
      listen       443;
      server_name  server;
      .............
      .............
      location / {
          root   /usr/share/nginx/html;
          index  index.html index.htm;
      }
      .............
      .............
   }
}
.............
.............
```

Pengarah Listen digunakan untuk menentukan nomor *port* tempat bekerja *server* HTTPS. Nilai *default* pengarah Listen adalah 443. Apabila nilai Listen diubah dari nilai *default*, maka pada saat akses tidak dapat dipanggil menggunakan protokol https.

. Pengarah Location / digunakan untuk menentukan letak dokumen yang nantinya akan diakses dari luar server. Pada pengarah ini diatur direktori root /usr/ share/nginx/html dan dokumen yang akan ditampilkan apabila tidak dituliskan secara eksplisit. Umumnya dokumen yang ditampilkan apabila tidak disebutkan secara eksplisit adalah index.htm atau index.html.

Pada *file* konfigurasi ssl.conf, yang perlu diatur adalah pengarah penggunaan SSL (*Secure Socket Layer*). Pengarah ssl dibuat tidak aktif.

```
ssl          off
```

Pengarah SSL harus dimatikan supaya pada saat *download file* tidak diminta untuk memasukkan sertifikat akses. Permintaan sertifikat akses tentunya akan mengganggu proses *download file*.

5.1.5 Konfigurasi Server FTP

Layanan permintaan protokol FTP ditangani oleh perangkat lunak ProFTPD. Versi Proftpd yang digunakan adalah 1.3.3. Proftpd dapat dijalankan secara *standalone* atau pun melewati server Xinetd. Pada penelitian ini diatur agar Proftpd dijalankan secara *standalone.*

Konfigurasi ProFTPD dituliskan dalam *file* /etc/proftpd.conf. Isi *file* tersebut secara lengkap dapat dilihat pada Lampiran. Bagian penting untuk konfigurasi proftpd.conf adalah sebagai berikut.

```
.............
.............
ServerType                      standalone
Port                            21
.............
.............
```

Pengarah ServerType digunakan untuk menentukan tipe server yang akan dijalankan. Pengarah Port digunakan untuk menentukan nomor port untuk menjalankan server.

5.1.6 Konfigurasi Server SSH

Layanan permintaan protokol SCP ditangani *server* SSH. Perangkat lunak *server* SSH yang umum dipakai sebagai standar pada jaringan Internet adalah OpenSSH. Versi OpenSSH yang dipakai pada penelitian ini adalah 5.3. OpenSSH merupakan perangkat lunak standar untuk melayani permintaan protokol SSH baik SSH versi 1 maupun SSH versi 2.

Konfigurasi OpenSSH dituliskan pada *file* /etc/ssh/sshd_config. Isi *file* konfigurasi ini secara lengkap disertakan pada Lampiran. Ada beberapa konfigurasi OpenSSH penting yang perlu dibahas berkaitan dengan pelaksanaan penelitian.

```
.............
.............
Port 22
ListenAddress 0.0.0.0
Protocol 2
PermitRootLogin yes
PermitEmptyPasswords yes
.............
.............
```

Pengarah Port digunakan untuk menentukan nomor *port* tempat *server* SSH bekerja. Nilai default untuk *port* kerja server SSH adalah 22. Nilai *port* dibiarkan *default* untuk memudahkan penggunaan protokol SCP.

Pengarah ListenAddress digunakan untuk menentukan alamat IP antarmuka tempat *server* SSH mendengarkan permintaan dari luar *server*. Nilai pengarah ini diberi nilai 0.0.0.0 yang artinya *server* SSH mendengarkan permintaan pada seluruh antarmuka yang ada pada *server*.

Pengarah PermitRootLogin digunakan untuk menentukan izin akses *root.* Pada penelitian ini diatur agar nilai pengarah ini bernilai yes, artinya *root* dapat langsung melakukan login dari luar *server*. Pengaturan ini dimaksudkan untuk kemudahan dalam konfigurasi *server*. Pada *server* sesungguhnya, nilai pengarah ini dibuat menjadi no untuk meningkatkan keamanan.

Pengarah PermitEmptyPasswords digunakan untuk menentukan izin akses *user* tanpa menggunakan *password*. Nilai pengarah diberi yes supaya *server* dapat menerima *login user* tanpa menggunakan *password*. Pengaturan ini dimaksudkan

supaya pengukuran kecepatan *download file* yang menjadi objek penelitian tidak terlalu dipengaruhi oleh waktu yang diperlukan untuk melakukan *login*.

Pengaturan berikutnya berkaitan dengan percepatan waktu *login* adalah pemetaan alamat IP komputer yang terlibat dalam proses autentikasi. Pemetaan ini dituliskan pada *file* /etc/hosts. Isi *file* ini adalah sebagai berikut.

```
..........
..........
127.0.0.1       localhost
..........
192.168.20.1    gateway1
192.168.20.2    server
192.168.10.1    gateway2
192.168.10.2    klien
..........
..........
```

Pada penelitian terlibat tiga komputer utama yaitu *router*, *server* dan klien. Tiga komputer utama dikenalkan pada *server* supaya proses autentikasi yang memerlukan resolusi nama komputer menjadi lebih cepat. Pada bagian luar terdapat dua antarmuka yaitu gatetay1 yang diberi alamat IP 192.168.20.1/24 dan gateway2 yang diberi alamat IP 192.168.10.1/24. *Server* diberi alamat IP 192.168.20.2/24 sedangkan klien diberi alamat IP 192.168.10.2/24.

5.1.7 Konfigurasi Server TFTP

Layanan permintaan protokol TFTP ditangani oleh Tftpd. Versi Tftpd yang dipakai pada penelitian ini adalah 5.0.3. Perangkat lunak Tftpd yang dipakai merupakan standar yang dipakai pada Linux Mandriva 2010.

Untuk menjalankan *server* Tftpd digunakan bantuan *server* Xinetd. Oleh sebab itu, konfigurasi Tftpd menyatu dengan *server* Xinetd, Konfigurasi Tftpd

diatur pada *file* /etc/xinetd.d/tftp. Dengan menghilangkan yang tidak penting, isi *file* tftp adalah sebagai berikut.

```
service tftp
{
        disable                 = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /var/lib/tftpboot
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

Beberapa pengaturan yang perlu dijelaskan adalah pengarah disable = no untuk menentukan *server* dalam keadaan aktif. Pengarah server = /usr/sbin/in.tftpd menentukan program *server* untuk menjalankan layanan tftp. Pengarah server_args = -s /var/lib/tftpboot untuk menentukan letak direktori tempat menyimpan dokumen yang akan diakses.

5.1.8 Konfigurasi Router Internet

Router Internet digunakan sebagai *router* penghubung dari jaringan *server* dengan jaringan klien. *Router* Internet sepenuhnya difungsikan hanya sebagai *router,* tidak difungsikan sebagai *Firewall*. Hal demikian dimaksudkan untuk menghilangkan pengaruh translasi alamat jaringan dalam penelitian ini.

Antarmuka jaringan yang menghadap jaringan klien yaitu ether1 diberi alamat IP 192.168.10.1/24. Alamat IP ini merupakan alamat IP *Gateway* komputer klien. Antarmuka jaringan yang menghadap jaringan *server* yaitu ether2 diberi alamat IP 192.168.20.1/24. Alamat IP ini menjadi alamat *Gateway*

komputer *server.* Konfigurasi jaringan dilakukan dengan beberapa perintah sebagai berikut.

```
> ip address add address=192.168.10.1/24 interface=ether1
> ip address add address= 192.168.20.1/24 interface=ether2
```

Perintah tersebut sudah disunting dengan menghilangkan tanda s*hell* Mikrotik [admin@MikroTik] untuk menghemat penulisan dan merapikan tampilan tanpa mengurangi inti isinya.

Hasil konfigurasi alamat IP dan tabel *routing* pada *router* Internet dapat dilihat pada tampilan berikut.

```
> ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   192.168.10.1/24    192.168.10.0    ether1
 1   192.168.20.1/24    192.168.20.0    ether2

> ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS      PREF-SRC      GATEWAY         DISTANCE
 1 ADC  192.168.10.0/24  192.168.10.1  ether2                 0
 2 ADC  192.168.20.0/24  192.168.20.1  ether1                 0
```

Tabel *routing* yang ada pada *router* Internet dihasilkan secara otomatis saat antarmuka ether1 dan ether2 diberi alamat IP. Kode untuk untuk dua *route* yang tercipta adalah ADC yaitu status yang merupakan singkatan dari *active dynamic connect*.

5.2 Pembahasan

Pada penelitian ini digunakan beberapa *file* dengan ukuran dan tipe yang berbeda. Uji coba transfer data dilakukan pada dua jenis *file* yaitu *file* teks dan *file*

gz. *File* teks mewakili transfer data untuk *file* yang belum mengalami kompresi, sedangkan *file* gz mewakili transfer data yang sudah mengalami kompresi. Ukuran *file* dipilih yang cukup besar. Sedemikian sehingga kecepatan transfer data dapat diamati secara lebih baik. Daftar *file* dan ukuran yang dipakai untuk penelitian disajikan pada Tabel 5.1.

Tabel 5.1 Daftar File

No	file	Size	. .	No	file	Size
1	file0.txt	79206023		6	file1.gz	17097953
3	file1.txt	158412046		7	file2.gz	34186353
3	file2.txt	316824092		8	file3.gz	68377785
4	file3.txt	633648184		9	file4.gz	136767033
5	file4.txt	1267296368		10	file5.gz	273535718

Sisi sebelah kiri merupakan *file* teks berekstensi *.txt, sedangkan sisi sebelah kanan merupakan *file* terkompresi berekstensi *.gz.

5.2.1 Uji Koneksi Jaringan

. Untuk memastikan jaringan sudah berjalan sesuai perancangan, dilakukan uji coba koneksi jaringan. Selanjutnya dilakukan uji coba hubungan antara klien dan *server* baik menggunakan uji ping maupun traceroute. Hasil uji ping dari klien menuju *server* adalah sebagai berikut.

```
# ping 192.168.20.2 -c 4
PING 192.168.20.2 (192.168.20.2) 56(84) bytes of data.
64 bytes from 192.168.20.2: icmp_seq=1 ttl=63 time=0.488 ms
64 bytes from 192.168.20.2: icmp_seq=2 ttl=63 time=0.486 ms
64 bytes from 192.168.20.2: icmp_seq=3 ttl=63 time=0.339 ms
64 bytes from 192.168.20.2: icmp_seq=4 ttl=63 time=0.492 ms

--- 192.168.20.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.339/0.451/0.492/0.066 ms
```

Hasil uji ping menunjukkan bahwa komputer klien dan *server* sudah terhubung secara betul dan berfungsi secara baik.

Hasil uji traceroute dari komputer klien menuju komputer *server* dapat dilihat sebagai berikut.

```
# traceroute 192.168.20.2
traceroute to 192.168.20.2 (192.168.20.2), 30 hops max, 60 byte packets
 1  192.168.10.1 (192.168.10.1)  0.187 ms  0.185 ms  0.193 ms
 2  192.168.20.2 (192.168.20.2)  1.135 ms  1.172 ms  1.179 ms
```

Hasil uji traceroute menunjukkan bahwa rute yang dilalui paket data sudah betul yaitu dari komputer klien melewati *router* Internet dan akhirnya sampai pada komputer *server*.

5.2.2 Sekrip Download

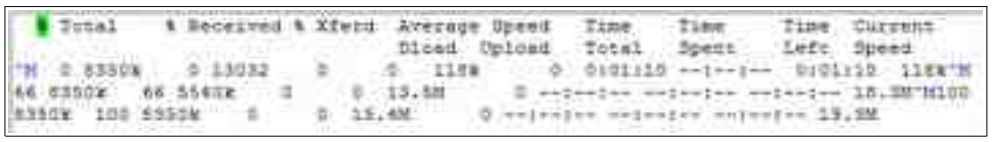
Pada penelitian akan diuji pengaruh berbagai macam protokol jaringan pada proses *download* file. Perangkat utama untuk proses *download file* adalah Curl. Perangkat lunak Curl punya dua fitur yang berguna pada penelitian ini yaitu mendukung berbagai protokol jaringan serta dapat menampilkan kecepatan *download* rata-rata pada terminal. Dukungan banyak protokol pada proses *download* memudahkan dalam hal penggunaan perangkat lunak yang tidak perlu berubah ketika protokol jaringan diubah. Dukungan tampilan kecepatan *download* memudahkan dalam pengukuran kecepatan *download file*.

Tampilan kecepatan *download file* dari Curl dapat ditangkap menggunakan pembelok arah > dari sekrip Bash. Salah satu contoh pengolahan awal untuk

mencuplik data *download* menggunakan protokol FTP dan membelokkan hasilnya pada *file* hasil.txt adalah sebagai berikut.

```
$ curl -o output.file ftp://xxx:yyy@192.168.20.2/file0.gz &> hasil.txt
```

Namun *file* hasil.txt masih berupa *file* yang perlu diolah terlebih dahulu. Apabila hasil *file* hasil.txt dilihat menggunakan *editor* mcedit, maka berbentuk seperti pada Gambar 5.2.

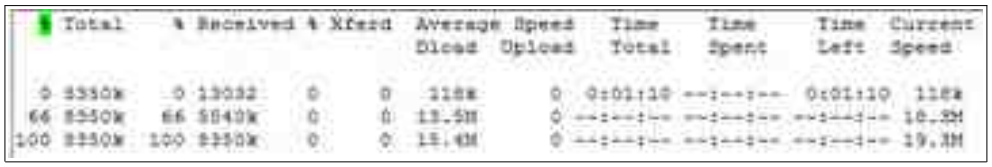


Gambar 5.2 Hasil Awal Tangkapan Curl

Pada *file* tersebut terdapat karakter ^M yang mengganggu dalam tampilan sistem Linux. Hasil *file* hasil.txt harus diubah supaya menjadi *file* Linux dengan bantuan perangkat lunak dos2unix, Perintah yang digunakan untuk maksud tersebut adalah sebagai berikut.

```
$ dos2unix hasil.txt
```

Perintah ini akan menghilangkan karakter ^M yang tidak diperlukan dalam *file* Linux. Hasil akhir *file* hasil.txt yang berisi informasi kecepatan *download file* ditunjukkan pada Gambar 5.3



Gambar 5.3 Tampilan Akhir Hasil Tangkapan Curl

Pada Gambar 5.3 dapat dilihat kecepatan rata-rata yang diperoleh selama proses *download file*. Data yang diperlukan adalah kecepatan rata-rata *download file* pada baris paling akhir kolom tujuh. Nilai ini dapat dicuplik menggunakan kombinasi perintah tail dan awk sebagai berikut

```
# tail -1 hasil.txt| awk '{ print $7}'
```

Perintah tail -1 untuk melihat menampilkan baris nomor satu dari belakang. Perntah awk '{ print $7}' untuk menampilkan kolom ke tujuh dengan pembatas kolom *default* berupa spasi.

Pencuplikan data dilakukan pada berbagai tipe protokol, tipe *file* dan ukuran *file*. Hal demikian akan cukup merepotkan apabila pencuplikan dilakukan manual satu demi satu, Untuk memudahkan dalam pencuplikan data kecepatan *download file*, dibuat skrip Bash sebagai berikut.

```
aprotocol="http https ftp scp sftp tftp"
afile="file0.txt file1.txt file2.txt file3.txt file4.txt \
       file1.gz file2.gz file3.gz file4.gz file5.gz"

server="192.168.20.2"
user="xxx"
pass="yyy"
tmpfile="tmp.txt"

#variasi protocol jaringan
for protocol in `echo $aprotocol`
do
  dev="data_download-$protocol.csv"
  #dev="/dev/stdout"

  echo `date` > $dev

  sleep 40

  #variasi tipe file
  for file in `echo $afile`
  do
    echo "$protocol;$file;"
    echo -n "$protocol;$file;" >>$dev

    #jumlah experimen
```

```
    for exp in  {1..10}
    do
      #hapus file
      rm -f *.txt >/dev/null
      rm -f *.gz >/dev/null

      #download file & membaca rate
      if [ $protocol='http' ]
      then
        curl -o $file http://$server/file/$file 2> $tmpfile
      elif [ $protocol='https' ]
      then
        curl -o $file http://$server:443/file/$file 2> $tmpfile
      elif [ $protocol='ftp' ]
      then
        curl -o $file ftp://$user:$pass@$server/file/$file 2> $tmpfile
      elif [ $protocol='scp' ]
      then
        curl scp://$user:$pass@$server/home/$user/file/$file 2> $tmpfile
      elif [ $protocol='sftp' ]
      then
        curl sftp://$user:$pass@$server/home/$user/file/$file 2> $tmpfile
      elif [ $protocol='tftp' ]
      then
        curl -o $file tftp://$server/$file 2> $tmpfile
      fi
      dos2unix $tmpfile
      rate=`tail -1 $tmpfile | awk '{print $7}'`
      echo -n "$rate;" >>$dev
    done
    echo >>$dev
  done
done
```

Sekrip program yang ditampilkan sedikit dimodifikasi dari program sesungguhnya untuk menyembunyikan *username* dan *password*. Hal demikian untuk menjaga keamanan server yang dipakai untuk penelitian. *Username* yang dituliskan dalam program adalah xxx, sedangkan *password* yang dituliskan adalah yyy.

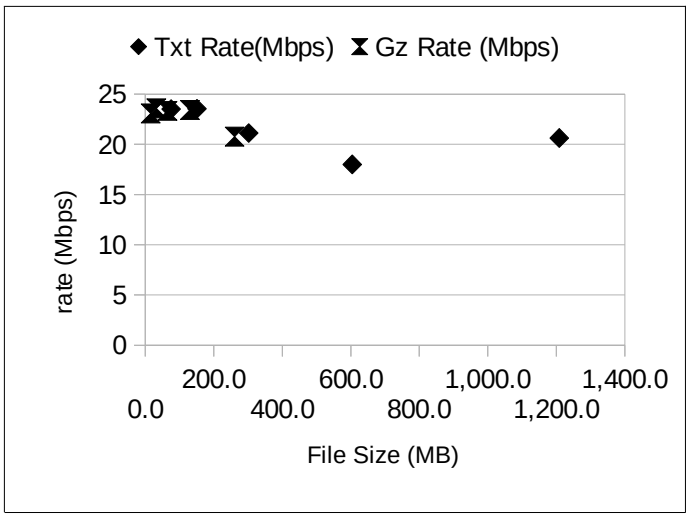
5.2.3 Percobaan Download File

Pada percobaan *download file* dicoba dilakukan *download* terhadap *file* teks (belum mengalami kompresi) dan *file* gz (sudah mengalami kompresi). Masing-asing sebanyak lima *file* dengan ukuran yang berbeda. Tiap-tiap *file* dilakukan pengambilan data sebanyak lima puluh kali. Hasil pengambilan data

dapat dilihat pada Lampiran.

Pengaruh Tipe File

Pengaruh tipe *file* pada *download file* menggunakan berbagai macam protokol diuji menggunakan dua tipe *file* yaitu teks dan gz. Tipe *file* teks mewakili *file* tidak terkompresi, sedangkan *file* gz mewakili *file* terkompresi.



Gambar 5.4 Pengaruh Enkripsi Upload File Teks

Hasil uji pengaruh tipe *file* pada *download file* ditunjukkan pada Gambar 5.4. Proses *download file* dilakukan menggunakan protokol HTTP.

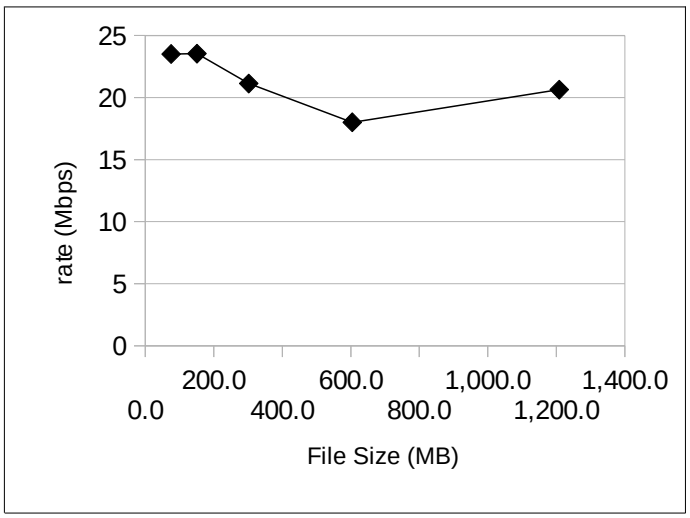
Hasil percobaan pengaruh tipe *file* terhadap kecepatan *download file* menunjukkan bahwa tipe *file* tidak berpengaruh pada kecepatan *download file*. Tidak ada perbedaan yang signifikan kecepatan *download file* untuk berbagai variasi ukuran file.

Pada penggunaan jenis protokol lain (HTTPS, FTP, SFTP, SCP, TFTP) juga menunjukkan bahwa tidak ada perbedaan kecepatan *download file* yang signifikan pada dua tipe *file* yang berbeda. Hasil lengkap uji pengaruh tipe *file* pada kecepatan *download file* ditampilkan pada Lampiran.

Pengaruh Ukuran File

Pengaruh ukuran pada kecepatan *download file* dapat dilihat pada Gambar 5.5. Pengaruh ukuran *file* diuji pada tipe *file* teks. Proses *download file* dilakukan menggunakan protokol HTTP.

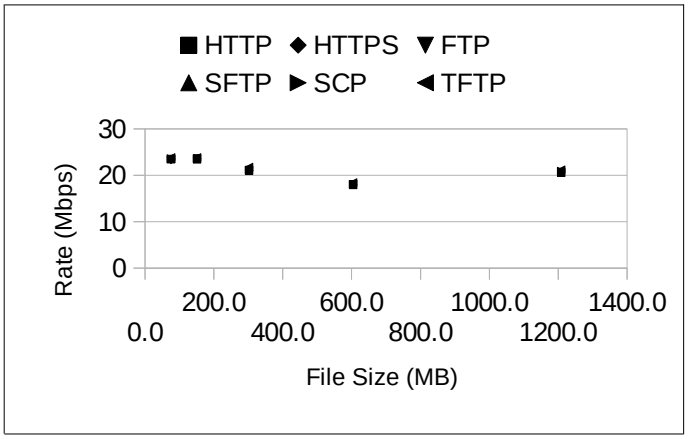
Pengaruh ukuran *file* pada kecepatan *download file* menunjukkan bahwa semakin besan ukuran *file* menyebabkan kecepatan *download file* cenderung semakin kecil. Pada penelitian tertangkap data bahwa pada ukuran *file* 604,3 MB kecepatan *download file* mencapai titik terendah yaitu sekitar 18.006 Mbps. Gejala ini perlu diteliti lebih lanjut apakah ada sesuatu yang berhubungan.



Gambar 5.5 Pengaruh Ukuran Pada Download File

Pengaruh Perbedaan Protokol

Pengaruh perbedaan protokol jaringan pada kecepatan *download file* ditunjukkan pada Gambar 5.6. Proses *download file* dilakukan menggunakan protokol HTTP, HTTPS, FTP, SFTP, SCP dan TFTP.



Gambar 5.6 Pengaruh Protokol Pada Download File

Pengaruh protokol jaringan pada proses *download file* menunjukkan bahwa tidak ada perbedaan secara signifikan antar protokol yang diuji. Namun demikian dapat dipetakan untuk menentukan kecenderungan pengaruh protokol jaringan yang lebih cepat. Pengaruh protokol jaringan pada kecepatan *download file* dapat ditunjukkan pada Tabel 5.2. Kolom pertama menunjukkan variasi ukuran *file*, sedangkan baris pertama menunjukkan variasi protokol jaringan yang digunakan untuk uji coba.

Tabel 5.2 Pemetaan Pengaruh Protokol Jaringan Kecepatan

File Size (MB)	HTTP	HTTPS	FTP	SFTP	SCP	TFTP
75,5	23,504	23,402	23,332	23,512	23,572	23,834
151,1	23,542	23,536	23,57	23,442	23,568	23,826
302,1	21,132	20,992	21,016	21,006	21,058	21,746
604,3	18,006	17,934	18,036	17,956	18,044	18,412
1208,6	20,636	20,682	20,574	20,598	20,906	21,172

Tabel 5.2 menunjukkan bahwa protokol jaringan yang paling cepat dalam proses *download file* adalah TFTP. Protokol TFTP menggunakan *datagram* dalam koneksi jaringan. Protokol *datagram* lebih ringan dalam koneksi jaringan karena tidak memerlukan adanya gandengan. Namun protokol TFTP umumnya dipakai untuk protokol *boot* yaitu proses *booting* melalui jaringan dan jarang digunakan untuk *download file* lewat jaringan Internet.

Apabila protokol TFTP ditiadakan dalam uji kecepatan download file, maka hasilnya ditunjukkan pada Tabel 5.3.

Tabel 5.3 Uji Coba Dengan Mengabaikan Protokol TFTP

File Size (MB)	HTTP	HTTPS	FTP	SFTP	SCP
75,5	23,504	23,402	23,332	23,512	23,572
151,1	23,542	23,536	23,57	23,442	23,568
302,1	21,132	20,992	21,016	21,006	21,058
604,3	18,006	17,934	18,036	17,956	18,044
1208,6	20,636	20,682	20,574	20,598	20,906

Tabel 5.3 menunjukkan bahwa kecepatan *download file* yang paling besar adalah cenderung pada penggunaan protokol SCP. Protokol ini dilayani oleh *server* SSH. Protokol SCP adalah jenis protokol yang dilengkapi fitur keamanan.

Dengan demikian, antar protokol jaringan yang diuji, tidak terlihat perbedaan yang signifikan dalam proses *download file*. Namun demikian, protokol yang cenderung lebih cepat dalam proses *download file* adalah TFTP. Jika percobaan hanya mempertimbangkan protokol yang umum dipakai untuk jaringan Internet, maka protokol SCP cenderung memberikan nilai kecepatan yang paling besar.

BAB 6 KESIMPULAN

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari hasil pembahasan dan percobaan dalam penelitian ini adalah sebagai berikut.

- Penelitian berhasil disusun metode untuk mengamati kecepatan *download file* menggunakan berbagai variasi protokol jaringan.
- Hasil percobaan tentang pengaruh tipe *file* terhadap kecepatan *download file* menunjukkan bahwa tipe *file* tidak berpengaruh pada kecepatan *download file*.
- Pengaruh ukuran *file* pada kecepatan *download file* menunjukkan bahwa semakin besan ukuran *file* menyebabkan kecepatan *download file* cenderung semakin kecil.
- Pengaruh protokol jaringan pada proses *download file* menunjukkan bahwa tidak ada perbedaan secara signifikan namun protokol TFTP cenderung menunjukkan hasil yang paling cepat.
- Pada jenis protokol yang umum dipakai untuk *download file* pada jaringan Internet, protokol SCP menunjukkan nilai kecepatan yang paling tinggi.

6.2 Saran

Saran yang diajukan untuk pengembangan dan penelitian lebih lanjut dari penelitian ini adalah sebagai berikut.

- Perlu dilakukan penelitian lebih lanjut tentang pengaruh ukuran *file* terhadap kecepatan *download file*. Saran ini didasarkan pada gejala bahwa pada variasi ukuran tertentu, kecepatan *download file* mencapai nilai terendah.
- Perlu dilakukan penelitian lebih lanjut pengaruh penggunaan protokol jaringan untuk kecepatan *upload file.* Yang perlu dipertimbangkan pada saran ini adalah bahwa tidak semua protokol jaringan yang dipakai pada penelitian mendukung proses *upload file.*

Daftar Pustaka

----------, 2015, Nginx Documentation, http://www.nginx.org/docs

Andino Maseleno, 2003, *Kamus Istilah Komputer dan Informatika*, IlmuKomputer.Com

Bradley M., 2015, *Protocol (network)*, http://compnetworking.about.com/od/networkprotocols/g/protocols.htm

Cisco CCNA, 2014, Introduction To Networks, CCNA 1 Course, Cisco Networking Academy, http://www.netacad.com

Fielding, R., et al., 1999, RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, The Internet Society, W3C/MIT, http://tools.ietf.org/html/rfc2616

Gede Wahyudi, Trisna Hanggara, 2013, A*nalisis Perbandingan Kinerja Antara Network file System (NFS) dan Primary Domain Controller (PDC) Samba*, Jurnal Ilmu Komputer, ISSN: 1979-5661, volume 6, nomor 1, UNUD Bali

Postel, J., Reynolds. J., 1985, RFC 959, *File Transfer Protocol (FTP)*, The Internet Society, ISI, http://tools.ietf.org/html/rfc959

Rescorla, E., 2000, RFC 2818, HTTP (Hypertext Transfer Protocol) Over TLS, The Internet Society, RTFM, Inc., , http://tools.ietf.org/html/rfc2818

Sollins, K., 1992, RFC 1350, The TFTP Protocol (Revision 2), The Internet Society, MIT, http://tools.ietf.org/html/rfc1350

Sony Bahagia Sinaga, 2012, *Analisa Perbandingan Kecepatan transfer Data Menggunakan Kabel UTP dan Wifi dengan Metode Stop & Wait Automatic Repeat Request*, Jurnal Ilmiah Pelita Informatika Budi Darma, ISSN: 2301-9425, volume II, nomor 1, STMIK Budi Darma Medan

Sugeng Riyadi, Harjono Harjono, 2013, A*nalisis Perbandingan Kecepatan download pada GSM*, jurnal Informatika Juita, ISSN: 2086-9398, volume 2, nomor 3, UMP Purwokerto

Wagito, 2012, *Implementasi VPN PPTP Untuk Integrasi Jaringan,* Puslitbang PPM, STMIK AKAKOM

Wagito, 2013, *Analisis Kecepatan transfer Data Point-To-Point Tunneling Protocol,* Puslitbang PPM, STMIK AKAKOM