IMPLEMENTASI MIKROTIK SEBAGAI MANAJEMEN *BANDWIDTH*

Rr. Retna Trimantaraningsih dan Istiqomatul Muarifah
Jurusan Teknik Informatika, FTI, IST AKPRIND Yogyakarta
Email: rereningsih@yahoo.com

ABSTRAK

MikroTik *Router*OS™ merupakan OS turunan dari distro Linux Debian. Mikrotik di khususkan untuk *router* dan *gateway*. Untuk mengoperasikanya, mikrotik dapat diremot dari *client* dengan menggunakan aplikasi *winbox*. Instalasi mikrotik cukup mudah, dapat dilakukan dengan mode teks ataupun menggunakan *winbox*.

Berdasarkan hasil penelitian diketahui bahwa: 1) hasil perbandingan antara CBQ dan HTB, pembagian *bandwidth* dapat dilakukan berdasarkan IP ataupun berdasarkan port atau kelas, 2) perbedaan antara algoritma CBQ dan HTB terletak pada *scheduler* dan *estimator* yang digunakan pada keduanya. HTB menggunakan *scheduler* DRR (*Defiit Round Robin)* dan *estimator* menggunakan TBF (*Token Bucket Filter)*, sedangkan *scheduler* pada CBQ menggunakan WRR *(Weight Round Robbin)* dan *estimator* menggunakan EWMA *(Exponent Weight Moving Averege)*, 3) kestabilan dan kecepatan transfer data cenderung sama, tergantung alokasi *bandwidth* yang diberikan dan ukuran data, 4) jika tidak diberikan alokasi *bandwidth*, maka tidak bisa untuk *download* dan *upload* meskipun antar klien terhubung ke PC *router*, dan 5) pembagian *bandwidth* dengan menggunakan mikrotik lebih mudah, karena ada aplikasi untuk *remote* dari klien dengan menggunakan *winbox*.

Kata kunci: *mikrotik, router, gateway, bandwidth, winbox, HTB, CBQ*

PENDAHULUAN

Kebutuhan akses internet dewasa ini sangat tinggi, baik untuk mencari informasi, artikel, pengetahuan terbaru, atau bahkan hanya untuk *chating*. Pembagian *IP Address* sudah semakin menipis. Satu *IP Address* perlu berhubungan dengan *IP Address* lainnya yang berbeda *class* atau *subnet*, maka diperlukan proses *routing* untuk menghubungkan antar *IP Address*. *Routing* akan membuat sebuah rantai jaringan saling terhubung dan bisa berkomunikasi dengan baik, dan informasi yang tersedia di satu *IP Address* akan didapatkan di *IP Address* lainnya. Perangkat yang digunakan untuk proses *routing* biasa disebut *router*. *Router* terdiri dari *hardware* & *software* yang harus terpasang sinkron supaya bisa bekerja dengan baik. *Router broadband* bisa dipakai secara langsung tanpa harus diinstall, atau bisa juga menggunakan sebuah komputer yang diinstall *software* untuk membuat *router*, dengan catatan *hardware* mendukung untuk proses *routing*.

Mikrotik adalah salah satu vendor *hardware* dan *software* yang menyediakan fasilitas untuk membuat *router*. Salah satunya adalah Mikrotik *Router* OS, yaitu OS yang khusus digunakan untuk membuat *router* dengan cara menginstallnya ke komputer. Fasilitas yang disediakan dalam Mikrotik *Router* OS sangat lengkap untuk membangun sebuah *router* yang handal dan stabil. Mikrotik juga menyediakan fasilitas untuk manajemen *bandwidth*. *Router* digunakan untuk komunikasi, sedangkan manajemen *bandwidth* terkait dengan kecepatan akses. Kecepatan akses data sangat dipengaruhi oleh besar kecilnya *bandwidth*. Kebutuhan *bandwidth* untuk *user* yang melakukan *browsing* saja, tentu berbeda dengan *user* yang melakukan *download*. Oleh karena itu diperlukan manajemen *bandwidth* yang baik sesuai kebutuhan *user* yang berbeda-beda.

Penelitian ini membahas tentang manajemen *bandwidth* yang baik, pemanfaatan MikroTik *Router*OS™ untuk manajemen *bandwith*, kestabilan dan efisiensi *software* manajemen *bandwidth*, serta keuntungan penggunaan MikroTik *Router*OS™ untuk manajemen *bandwith*.

Sistem Operasi

Sistem operasi (OS) adalah perangkat lunak yang mengendalikan komponen komputer individual, juga bertugas untuk mengkoordinasikan program perangkat lunak lain yang berjalan pada komputer. Secara umum, OS adalah *software* pada lapisan pertama yang ditaruh pada memori komputer pada saat komputer dinyalakan. Sedangkan *software* lainnya dijalankan setelah OS berjalan, dan OS akan melakukan layanan inti umum untuk *software-software* itu. Layanan inti umum tersebut seperti akses ke *disk*, manajemen memori, penjadwalan tugas, dan antar-muka *user*, sehingga masing-masing *software* tidak perlu lagi melakukan tugas-tugas inti umum tersebut, karena dapat dilayani dan dilakukan oleh OS. Bagian kode yang melakukan tugas-tugas inti dan umum tersebut dinamakan dengan "*kernel*" suatu OS.

OS melakukan semua tugas-tugas penting dalam komputer, dan menjamin aplikasi-aplikasi yang berbeda dapat berjalan secara bersamaan dengan lancar. OS menjamin aplikasi *software* lainnya dapat menggunakan memori, melakukan input dan output terhadap peralatan lain, dan memiliki akses kepada sistem file. Apabila beberapa aplikasi berjalan secara bersamaan, maka OS mengatur jadwal yang tepat, sehingga sedapat mungkin semua proses yang berjalan mendapatkan waktu yang cukup untuk menggunakan prosesor (CPU) serta tidak saling mengganggu. Secara umum OS terdiri:

1. Mekanisme *Boot,* yaitu meletakkan kernel ke dalam memori.
2. *Kernel*, yaitu inti dari sebuah OS.
3. *Command Interpreter* atau *shell*, bertugas membaca input dari *user*.
4. *Library*, menyediakan fungsi dasar dan standar yang dapat dipanggil aplikasi lain.
5. *Driver* untuk berinteraksi dan mengontrol *hardware* eksternal.

Sebagian OS hanya mengizinkan satu aplikasi saja yang berjalan pada satu waktu, tetapi sebagian besar OS baru mengizinkan beberapa aplikasi berjalan secara simultan pada waktu yang bersamaan yang disebut *Multi-tasking OS*. Beberapa OS berukuran sangat besar dan kompleks, serta inputnya tergantung kepada input pengguna, sedangkan OS lainnya sangat kecil dan dibuat dengan asumsi bekerja tanpa intervensi manusia sama sekali. Tipe yang pertama sering disebut sebagai *Desktop OS*, sedangkan tipe kedua adalah *Real-Time OS.*

Program saling berkomunikasi antara satu dengan lainnya dengan antarmuka pemrograman aplikasi (*Application Programming Interface* atau disingkat dengan *API*. Dengan API inilah program aplikasi dapat berkomunikasi dengan OS. Sebagaimana manusia berkomunikasi dengan komputer melalui antarmuka *user*, program juga berkomunikasi dengan program lainnya melalui API. Walaupun demikian API sebuah komputer tidaklah berpengaruh sepenuhnya pada program-program yang dijalankan diatas *platform* OS tersebut. Misal, program yang dibuat untuk OS Windows 3.1 bila dijalankan pada OS Windows 95 dan generasi setelahnya akan terlihat perbedaan yang mencolok antara window program tersebut dengan program yang lain.

Setiap proses dalam sebuah OS mendapatkan sebuah PCB (*Process Control Block*) yang memuat informasi tentang proses tersebut, yaitu: sebuah tanda pengenal proses (*Process ID*) yang unik dan menjadi nomor identitas, status proses, prioritas eksekusi proses, dan informasi lokasi proses dalam memori. Prioritas proses merupakan suatu nilai atau besaran yang menunjukkan seberapa sering proses harus dijalankan oleh prosesor. Proses yang memiliki prioritas lebih tinggi, akan dijalankan lebih sering atau dieksekusi lebih dulu dibandingkan dengan proses yang berprioritas lebih rendah. Suatu OS dapat saja menentukan semua proses dengan prioritas yang sama, sehingga setiap proses memiliki kesempatan yang sama. Suatu OS dapat juga merubah nilai prioritas proses tertentu, agar proses tersebut akan dapat memiliki kesempatan lebih besar pada eksekusi berikutnya (misalnya: pada proses yang sudah sangat terlalu lama menunggu eksekusi, OS menaikkan nilai prioritasnya). Jenis status yang ditempatkan pada suatu proses pada setiap OS dapat berbeda-beda. Tetapi paling tidak ada 3 macam status yang umum, yaitu:

1. *Ready*, yaitu status dimana proses siap untuk dieksekusi pada giliran berikutnya.
2. *Running*, yaitu status dimana saat ini proses sedang dieksekusi oleh prosesor.
3. *Blocked*, yaitu status dimana proses tidak dapat dijalankan pada saat prosesor siap/bebas (http://id.wikipedia.org/wiki/Sistem_operasi, 2008).

Jaringan Komputer

Jaringan komputer adalah dua atau lebih komputer yang saling terhubung, bisa berbagi pakai file (data, *software*) dan peralatan (*modem*, *scanner*, CDROM, dll) jaringan pada beberapa lokasi (email, *link*, *video conferences*). Komponen jaringan meliputi:

1. Minimal ada 2 komputer
2. Antar muka jaringan pada setiap komputer (*NIC* atau *adapter*)
3. Media koneksi (kabel, gelombang radio, dll)
4. OS jaringan
5. Antar muka jaringan yang disebut NIC (*Network Interface Card*), berupa *Adapter Card*, *PC Card* atau *Compact Flash Card* yang menyebabkan komputer atau peralatan bisa terhubung ke jaringan.

***Local Area Network* (LAN)**

LAN merupakan jaringan yang meliputi area geografis yang relatif kecil. LAN dicirikan dengan kecepatan data yang relatif tinggi dan *error* yang relatif rendah. LAN menghubungkan *workstation*, perangkat jaringan, terminal, dan perangkat lain dalam area yang terbatas. LAN dimiliki oleh

pengguna dan tidak dioperasikan lewat sambungan sewa, meskipun LAN mungkin saja memiliki pintu gerbang ke PSTN atau jaringan swasta lainya.

***Metropolitan Area Network* (MAN)**

Pada dasarnya, MAN merupakan versi LAN yang berukuran lebih besar dan biasanya memiliki teknologi yang sama dengan LAN. MAN dapat mencakup kantor-kantor yang berdekatan atau dalam sebuah kota dan dapat dimanfaatkan untuk keperluan pribadi. MAN mampu menunjang data dan suara, bahkan dapat berhubungan dengan jaringan TV kabel. Alasan utama untuk memisahkan MAN sebagai kategori khusus adalah telah ditentukannya standar untuk MAN, dan standar ini sekarang sedang diimplementasikan.

***Wide Area Network* (WAN)**

WAN merupakan sebuah jaringan yang biasanya dibangun menggunakan jalur serial, yang mencakup area geografis yang luas, seringkali mencakup sebuah negara atau benua. WAN terdiri dari kumpulan mesin yang bertujuan untuk menjalankan program-program aplikasi pemakai. Pada sebagian besar WAN, *subnet-subnet* terdiri dari dua komponen yaitu kabel transmisi dan elemen *switch*. Kabel transmisi berfungsi untuk memindahkan bit-bit dari satu mesin ke mesin lainya, sedangkan *switch* adalah komputer khusus yang dipakai untuk menghubungkan dua kabel transmisi atau lebih.

Internet

Internet adalah jaringan komputer global yang memungkinkan dua komputer atau lebih berkoneksi untuk mentransfer file dan tukar menukar email dan pesan-pesan *real-time*. Internet merupakan landasan untuk *World Wide Web*. Internet juga merupakan kumpulan jaringan komputer yang berbeda-beda dan saling berhubungan di seluruh dunia. Semua komputer yang dihubungkan dengan internet dapat berkomunikasi satu sama lain dengan menggunakan *Transmission Control Protokol/Internet Protocol* (TCP/IP).

Gateway

Gateway adalah perangkat keras dan perangkat lunak interkoneksi jaringan yang beroperasi pada layer 7 OSI. *Gateway* menginterkoneksikan dua jaringan, *node* jaringan, *subnet*work, atau perangkat jaringan yang tidak bersesuaian. *Gateway* melakukan operasi konversi protokol dalam suatu spektrum yang luas dari fungsi-fungsi komunikasi.

Router

Router adalah perangkat keras yang memfasilitasi transmisi paket data melalui jaringan komputer. Fungsi *router* adalah sebagai penghubung antara dua atau lebih jaringan untuk meneruskan data dari satu jaringan ke jaringan lainnya. *Router* berbeda dengan *switch*. Sebagai ilustrasi perbedaan fungsi *router* dan *switch* adalah *switch* merupakan sebuah jalan, dan *router* merupakan penghubung antar jalan. Masing-masing rumah berada pada jalan yang memiliki alamat dalam suatu urutan tertentu. Dengan cara yang sama, *switch* menghubungkan berbagai macam alat, dimana masing-masing alat memiliki alamat IP sendiri pada sebuah LAN. *Routing* adalah proses membawa paket data dari satu *host* ke *host* yang lain tetapi berbeda *subnet*. *Router* merupakan perangkat yang menjembatani dua jaringan (http:// its.ac.id,2006).

Router banyak digunakan dalam jaringan berbasis teknologi protokol TCP/IP, dan *router* jenis itu disebut juga dengan ***IP Router***. Selain *IP Router*, ada lagi *AppleTalk Router*, dan masih ada beberapa jenis *router* lainnya. Internet merupakan contoh sebuah jaringan yang memiliki banyak *router IP*. *Router* dapat digunakan untuk menghubungkan banyak jaringan kecil ke sebuah jaringan yang lebih besar, yang disebut dengan *internetwork*, atau untuk membagi sebuah jaringan besar ke dalam beberapa *subnetwork* untuk meningkatkan kinerja dan juga mempermudah pengelolaannya. *Router* kadang juga digunakan untuk mengoneksikan dua buah jaringan yang menggunakan media yang berbeda (seperti halnya *router wireless* yang pada umumnya selain dapat menghubungkan komputer dengan menggunakan radio, juga mendukung penghubungan komputer dengan kabel UTP), atau berbeda arsitektur jaringan, seperti Ethernet ke *Token Ring*.

Router juga dapat digunakan untuk menghubungkan LAN ke sebuah layanan telekomunikasi seperti halnya telekomunikasi leased line atau *Digital Subscriber Line* (DSL). *Router* yang digunakan untuk menghubungkan LAN ke sebuah koneksi *leased line* seperti T1, atau T3, sering disebut sebagai *access server*. Sementara itu, *router* yang digunakan untuk menghubungkan jaringan lokal ke sebuah koneksi DSL disebut *DSL router*. *Router-router* jenis tersebut umumnya memiliki fungsi *firewall* untuk melakukan penapisan paket berdasarkan alamat sumber dan alamat tujuan paket tersebut, meski

beberapa *router* tidak memilikinya. *Router* yang memiliki fitur penapisan paket disebut juga dengan *packet-filtering router*. *Router* umumnya memblokir lalu lintas data yang dipancarkan secara *broadcast* sehingga dapat mencegah adanya *broadcast storm* yang memperlambat kinerja jaringan.
Secara umum, *router* dibagi menjadi dua buah jenis, yakni:

1. *Static router* (*router* statis), yaitu sebuah *router* yang memiliki tabel *routing* statis yang diset secara manual oleh administrator jaringan.
2. *Dynamic router* (*router* dinamis), yaitu sebuah *router* yang memiliki dan membuat tabel *routing* dinamis, dengan mendengarkan lalu lintas jaringan dan saling berhubungan dengan *router* lainnya.

Cara kerja *router* mirip dengan *bridge* jaringan, yakni meneruskan paket data jaringan dan dapat juga membagi jaringan menjadi beberapa segmen atau menyatukan segmen-segmen jaringan. Akan tetapi, *router* berjalan pada lapisan ketiga pada model OSI (lapisan jaringan), dan menggunakan skema pengalamatan yang digunakan pada lapisan itu, seperti halnya alamat IP. Sementara itu, *bridge* jaringan berjalan pada lapisan kedua pada model OSI (lapisan *data link*), dan menggunakan skema pengalamatan yang digunakan pada lapisan itu, yakni *MAC address*. *Bridge*, sebaiknya digunakan untuk menghubungkan segmen-segmen jaringan yang menjalankan protokol jaringan yang sama, Selain itu, *bridge* juga dapat digunakan ketika di dalam jaringan terdapat protokol-protokol yang tidak bisa melakukan *routing*. Sementara itu, *router* sebaiknya digunakan untuk menghubungkan segmen-segmen jaringan yang menjalankan protokol jaringan yang berebeda. Secara umum, *router* lebih cerdas dibandingkan dengan *bridge* dan dapat meningkatkan *bandwidth* jaringan, mengingat *router* tidak meneruskan paket *broadcast* ke jaringan yang dituju. *Router* paling sering digunakan untuk menghubungkan jaringan lokal ke internet (http://id.wikipedia.org/wiki/*Router*, 2008).

Bandwith

Bandwidth adalah kapasitas atau daya tampung kabel ethernet agar dapat dilewati trafik paket data dalam jumlah tertentu. *Bandwidth* juga bisa berarti jumlah konsumsi paket data per satuan waktu dinyatakan dengan satuan *bit per second* [bps]. *Bandwidth* internet disediakan oleh *provider* internet dengan jumlah tertentu tergantung sewa pelanggan. Dengan QoS dapat diatur agar *user* tidak menghabiskan *bandwidth* yang di sediakan oleh provider. Istilah *bandwidth* muncul dari bidang teknik elektro, dimana *bandwidth* mempresentasikan jarak keseluruhan atau jangkauan di antara sinyal tertinggi dan terendah pada kanal (*band*) komunikasi.

Pada dasarnya *bandwith* mempresentasikan kapasitas dari koneksi, semakin tinggi kapasitas, maka umumnya akan diikuti oleh kinerja yang lebih baik, meskipun kinerja keseluruhan juga tergantung pada faktor-faktor lain, misalnya *latency* yaitu waktu tunda antara masa sebuah perangkat meminta akses ke jaringan dan masa perangkat itu memberi izin untuk melakukan transmisi.

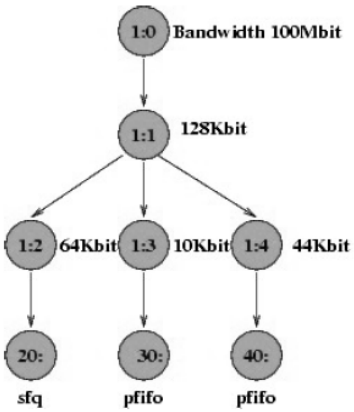
***Class Based Queue* (CBQ)**

Dalam *bandwidth management* dikenal istilah *queuing* yaitu penjadwalan bagaimana suatu data dikirimkan melewati jaringan. Semua data yang melewati jaringan akan dijadwalkan dan mengalami proses *shaping* sesuai dengan aturan yang berlaku pada jaringan tersebut. *Class-Based Queuing* (CBQ) adalah suatu mekanisme penjadwalan, bertujuan menyediakan *link sharing* antar agensi yang menggunakan jalur fisik yang sama, dan sebagai acuan untuk membedakan *traffic* yang memiliki prioritas yang berlainan. Dengan CBQ, setiap agensi dapat mengalokasikan *bandwidth* miliknya untuk berbagai jenis *traffic* yang berbeda, sesuai dengan pembagiannya yang tepat untuk masing-masing *traffic.* CBQ berinteraksi dengan *link sharing* yang memberikan keunggulan yaitu pemberian *bandwidth* yang tak terpakai bagi *leaf classnya* sebelum diberikan kepada agensi-agensi lain.

CBQ merupakan teknik klasifikasi paket data yang paling terkenal, mudah dikonfigurasi, memungkinkan sharing *bandwidth* antar *class* dan memiliki fasilitas *user* interface. CBQ mengatur pemakaian *bandwidth* jaringan yang dialokasikan untuk setiap *user*, pemakaian *bandwidth* yang melebihi nilai *set* akan dipotong (*shaping*), CBQ juga dapat diatur untuk *sharing* dan meminjam *bandwidth* antar *class* jika diperlukan. Parameter CBQ adalah:

1. *Avpkt*, adalah jumlah paket rata-rata saat pengiriman.
2. *Bandwidth*, adalah lebar *bandwidth* kartu ethernet biasanya 10-100Mbit
3. *Rate* yaitu, adalah kecepatan rata-rata paket data saat meninggalkan *qdisc*, *rate* merupakan parameter untuk mengeset *bandwidth*.
4. *Cell*, adalah peningkatan paket data yang dikeluarkan ke kartu ethernet berdasarkan jumlah byte, misalnya 800 ke 808 dengan nilai *cell* 8.

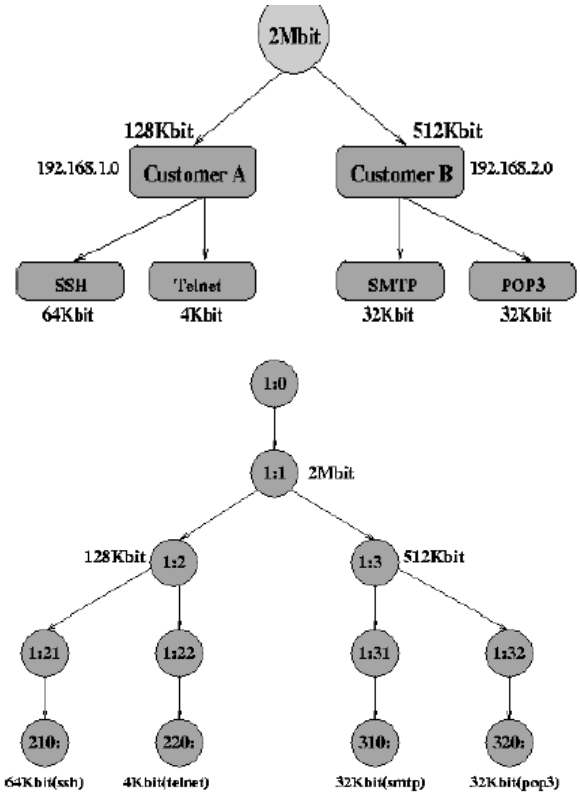
5. *Isolated/Sharing*. *Isolated* adalah parameter untuk mengatur agar *bandwidth* tidak bisa dipinjam oleh *class* lain yang memiliki tingkat yang sama/*sibling*. Parameter *Sharing* menunjukkan *bandwidth class* yang bisa dipinjam oleh *class* lain.
6. *Bounded/Borrow*. Parameter *Bounded* berarti *class* tidak dapat meminjam *bandwidth* dari klas lain, sedangkan *Borrow* adalah sebaliknya.



Gambar 1: Ilustrasi CBQ

Hierarchy Token Bucket (HTB)

Teknik antrian HTB mirip dengan CBQ, perbedaannya terletak pada jenis pilihan yang disediakan. HTB memiliki lebih sedikit pilihan saat konfigurasi dan lebih presisi. Teknik antrian HTB memberikan fasilitas pembatasan trafik pada setiap level maupun klasifikasi, *bandwidth* yang tidak terpakai bisa digunakan oleh klasifikasi yang lebih rendah. Susunan HTB dapat dilihat seperti suatu struktur organisasi dimana pada setiap bagian memiliki wewenang dan mampu membantu bagian lain yang memerlukan. Teknik antrian HTB cocok diterapkan pada perusahaan dengan banyak struktur organisasi.



Gambar 2: Ilustrasi HTB seperti susunan organisasi

Parameter HTB adalah:

1. *Rate*, yaitu parameter untuk menentukan *bandwidth* maksimum yang bisa dipakai oleh setiap *class*, jika *bandwidth* melebihi nilai "rate" maka paket data akan dipotong atau ditanggalkan (*drop*).
2. *Ceil*, yaitu parameter untuk menentukan peminjaman *bandwidth* antar *class* (kelas), peminjaman *bandwith* dilakukan oleh *class* lebih rendah ke kelas di atasnya, teknik ini disebut *link sharing*.

Proxy Server

Proxy server adalah sebuah komputer *server* atau program komputer yang dapat bertindak sebagai komputer lainnya untuk melakukan *request* terhadap *content* dari Internet atau intranet. *Proxy server* bertindak sebagai *gateway* terhadap dunia Internet untuk setiap komputer klien. *Proxy server* tidak terlihat oleh komputer klien, *user* yang berinteraksi dengan internet melalui sebuah *proxy server* tidak akan mengetahui bahwa sebuah *proxy server* sedang menangani *request* yang dilakukannya. *Web server* yang menerima *request* dari *proxy server* akan menginterpretasikan *request-request* tersebut seolah-olah *request* itu datang secara langsung dari komputer klien, bukan dari *proxy server*. Proxy server juga dapat digunakan untuk mengamankan jaringan pribadi yang dihubungkan ke sebuah jaringan publik (seperti Internet). *Proxy server* memiliki lebih banyak fungsi daripada *router* yang memiliki fitur *packet filtering* karena *proxy server* beroperasi pada level yang lebih tinggi dan memiliki kontrol yang lebih lengkap terhadap akses jaringan. *Proxy server* yang berfungsi sebagai sebuah "agen keamanan" untuk sebuah jaringan pribadi, umumnya dikenal sebagai *firewall*.

Fungsi dari *proxy* yaitu:

1. *Connection Sharing*, pengguna tidak langsung berhubungan dengan jaringan luar, atau internet tetapi harus melewati suatu *gateway* yang bertindak sebagai batas antara jaringan lokal dan jaringan luar.
2. *Caching*, *proxy server* memiliki mekanisme penyimpanan obyek-obyek yang sudah pernah diminta dari server-server di internet.
3. *Filtering*, bekerja pada layer aplikasi sehingga berfungsi sebagai *firewall* paket filtering yang digunakan untuk melindungi jaringan lokal dari serangan atau gangguan yang berasal dari jaringan internet.

Cara kerja dari *proxy* yaitu:

1. *Proxy server* memotong hubungan langsung antara pengguna dan layanan yang diakses.
2. Dilakukan pertama-tama dengan merubah alamat IP, membuat pemetaan alamat IP jaringan lokal ke alamat *IP proxy* yang digunakan untuk jaringan luar atau internet.
3. Pada prinsipnya hanya alamat *IP proxy* yang akan diketahui secara umum dan berfungsi sebagai *network address translator*.

MikroTik *Router*OS

MikroTik *Router*OS merupakan OS berbasis Linux yang diperuntukkan sebagai *network router*, didesain untuk memberikan kemudahan bagi penggunanya. Administrasinya biasa dilakukan melalui *Windows Application* (*Winbox*). Selain itu instalasi dapat dilakukan pada *Standard computer PC*. PC yang akan dijadikan *router* mikrotik tidak memerlukan *resource* yang tinggi untuk penggunaan standard, misalnya hanya sebagai *gateway*. Fasilitas pada mikrotik antara lain:

1. *Protocol routing* RIP, OSPF, BGP
2. *Statefull firewall*
3. *HotSpot for Plug-and-Play access*
4. *Remote winbox GUI admin*

Meskipun demikian, mikrotik bukanlah *software* yang free, artinya harus membeli lisensi untuk segala fasilitas yang disediakan. Instalasi mikrotik dapat dlakukan dengan tiga cara yaitu:

1. Installasi melalui NetInstall via jaringan
2. Installasi melalui *floppy disk*
3. Installasi melalui CD_ROM

Untuk *bandwidth management*, mikrotik menggunakan algoritma *Hierarchy Token Bucket* (HTB). *Bandwidth control* merupakan mekanisme untuk mengontrol alokasi data, *delay variability, timely delivery, dan delivery realibility.* Mikrotik mengikuti aturan *queuing*, yaitu:

1. *PFIFO - Packets First-In First-Out*
2. *BFIFO - Bytes First-In First-Out*
3. *SFQ - Stochastic Fairness Queuing*
4. *RED - Random Early Detect*
5. *PCQ - Per Connection Queue*
6. *HTB - Hierarchical Token Bucket*

Kebutuhan Sistem

Penelitian ini dilakukan di Laboratorium Komputer IST AKPRIND Yogyakarta. Untukuji coba sistem, digunakan *hardware* dan *software*, antara lain:

1. *Hardware*: Processor intel P4 3.06 GHz, RAM DDR II 256 MB, 2 *ethernal card*, kabel UTP, konektor RJ45, *hardisk* 40 GB, dll.
2. *Software*: OS Windows XP (untuk menjalankan winbox dalam pengaturan *bandwidth*), MikroTik RouterOS (sebagai *router* sekaligus *bandwidth management*), *bandwidth meter pro* (untuk monitoring *bandwidth*), dan Winbox (aplikasi kontrol melalui OS Windows dan *web interface* yang dapat di akses melalui *web browser*).

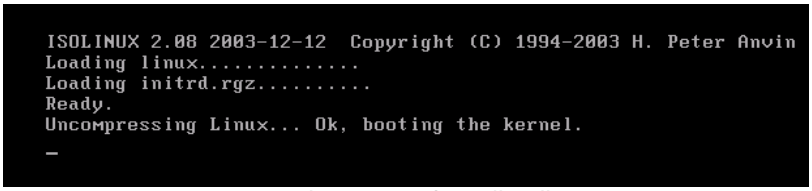
Langkah-langkah yang dilakukan dalam penelitian adalah:

1. Installasi mikrotik
2. Instalasi linux
3. Konfigurasi *router* dan gateway pada mikrotik dan linux
4. Seting *bandwidth* pada linux
5. Setting *bandwidth* menggunakan *winbox*

HASIL DAN PEMBAHASAN

Instalasi Mikrotik

Spesifikasi *hardware* minimum untuk installasi mikrotik minimum adalah prosesor P III 450Mhz, RAM 256 Mb, HD 20Gb. Proses installasi mikrotik pada penelitian ini dilakukan melalui CD-ROM setelah sebelumnya diset *boot* melalui CD-ROM. Jika berhasil, tampilan *booting* awal sistem mikrotik tampak seperti Gambar 3.

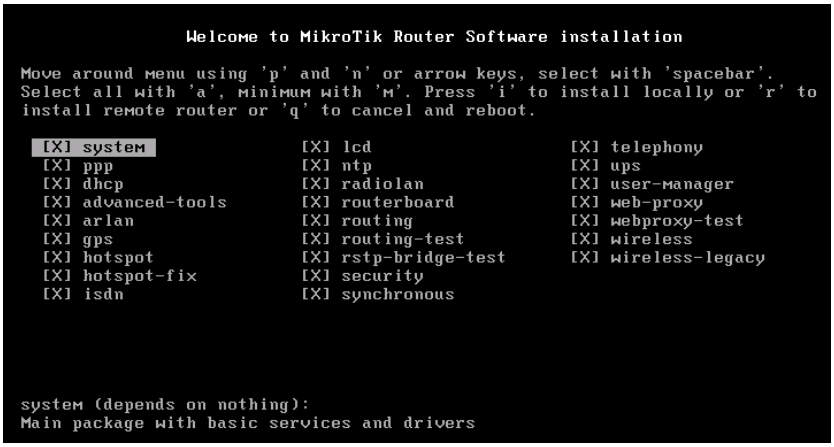


Gambar 3: *Booting* mikrotik

Jika *booting* berjalan dengan benar, maka tampilanya tampak seperti Gambar 4.

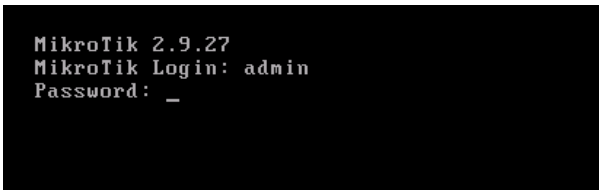
Gambar 4: Menu Instalasi

Menu instalasi menampilkan *service* yang disediakan oleh mikrotik. Untuk lebih mudahnya, bisa dipilih semua menu yang tersedia, seperti tampak pada Gambar 5.



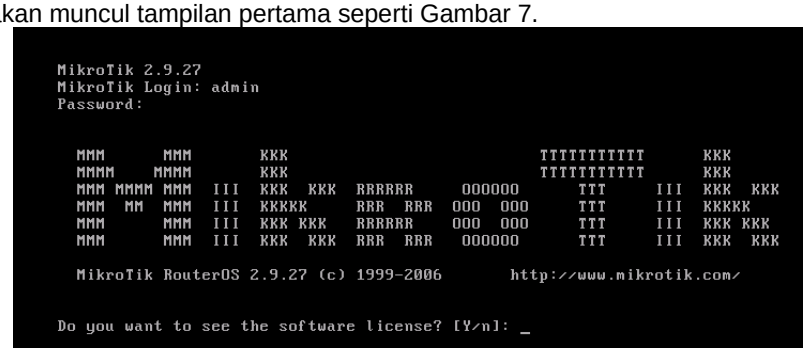
Gambar 5: Menu yang akan diinstal

Untuk melakukan installasi tekan tombol “n”, jika hanya untuk menambahkan *service* baru tekan tombol “y” agar konfigurasi sebelumnya tidak hilang. Selanjutnya perlu disiapkan ruang *hardisk* yang akan digunakan untuk mikrotik dengan cara menformatnya dan mengkopikan file-file yang dibutuhkan. Setelah proses pengkopian file selesai, sisem akan meminta *reboot*. Jika proses instalasi berjalan dengan benar, maka tampilan layar *reboot* akan muncul *user* login dan password seperti pada Gambar 6. Secara *default user* yang dipakai adalah *admin* dengan *password* kosong.



Gambar 6: Login Mikrotik

Selanjutnya akan muncul tampilan pertama seperti Gambar 7.



Gambar 7: Tampilan awal Mikrotik

Setelah proses instalasi selesai, langkah selanjutnya mengkonfigurasi PC sebagai *Router*/*gateway*, DNS, DHCP, serta *bandwidth management*.

Setting *IP Address*

Sebelum melakukan *setting IP Address*, perlu dicek interface yang ada untuk memastikan *interface* yang ada sudah terdeteksi oleh Mikrotik. Untuk mempermudah membedakan nama *ethernet*, perlu diganti penamaan nama *ethernet* seperti Gambar 8.

```
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX-RATE    TX-RATE    MTU
 0  R ether1                        ether            0          0          1500
 1  R ether2                        ether            0          0          1500
[admin@MikroTik] > interface
[admin@MikroTik] interface> set 0 name=publik
[admin@MikroTik] interface> set 1 name=lokal
[admin@MikroTik] interface> /
[admin@MikroTik] > system identity set name=ifach
[admin@ifach] > interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX-RATE    TX-RATE    MTU
 0  R publik                        ether            0          0          1500
 1  R lokal                         ether            0          0          1500
[admin@ifach] >
```

Gambar 8: Seting *interface*

Setelah terdeteksi oleh Mikrotik, selanjutnya di-*setting IP Address*, seperti Gambar 9.

```
[admin@ifach] > ip address add address=192.168.1.70 netmask=255.255.255.0 interf
ace=publik comment="ip ke internet"
[admin@ifach] > ip address add address=192.168.2.1 netmask=255.255.255.0 interfa
ce=lokal comment="ip lan"
[admin@ifach] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   ;;; ip ke internet
     192.168.1.70/24    192.168.1.0     192.168.1.255   publik
 1   ;;; ip lan
     192.168.2.1/24     192.168.2.0     192.168.2.255   lokal
```

Gambar 9: Seting *IP Address*

Konfigurasi Gateway

Setelah *IP Address* terkonfigurasi, selanjutnya dilakukan konfigurasi *gateway*, seperti Gambar 10.

```
[admin@ifach] > ip route add gateway=192.168.1.1
[admin@ifach] >
```

Gambar 10: Seting *routing*

Jika mikrotik digunakan sebagai *gateway server* maka harus di*masquerading*, seperti Gambar 11.

```
[admin@ifach] > ip firewall nat add chain=srcnat out-interface=publik action=masquerade
[admin@ifach] > ip firewall nat print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=srcnat out-interface=publik action=masquerade
```

Gambar 11: Seting firewall

Setting DNS (Domain Name System)

Selanjutnya yang harus dikonfigurasi adalah DNS, seperti Gambar 12.

```
[admin@ifach] > ip dns set primary-dns=202.134.2.5
[admin@ifach] > ip dns set primary-dns=202.134.2.5 allow-remote-requests=yes
[admin@ifach] > ip dns set secondary-dns=202.134.0.155 allow-remote-requests=yes
[admin@ifach] > ip dns print
           primary-dns: 202.134.2.5
         secondary-dns: 202.134.0.155
  allow-remote-requests: yes
             cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 17KiB
[admin@ifach] > ping 192.168.1.1
192.168.1.1 ping timeout
192.168.1.1 ping timeout
3 packets transmitted, 0 packets received, 100% packet loss
[admin@ifach] >
```

Gambar 12: Seting DNS

Seting Alokasi *Bandwidth*

Seting alokasi *bandwidth* yang disediakan oleh *winbox* adalah:

1. *Simple queue*. Ketentuan pengaturan *banwidth* ini cukup sederhana, besarnya *banwidth* bersifat *fixed* yaitu 64k, 128k, 512k, 1M, dan 2M.
2. *Queue tree*, merupakan pembagian *bandwidth* yang membawahi beberapa kelas. Sebelum membuat *queue tree* perlu dibuat *mungle* (untuk membuat *mark connection* dan *mark packet*) yang berfungsi untuk memonitor koneksi dan paket yang dilewatkan.

General Scheduler HTB-DRR (*Deficit Round Robin*)

DRR merupakan sebuah mekanisme penjadwalan pada HTB. Prinsip dasar dari DRR adalah suatu kelas berhak untuk mengirimkan paket dalam suatu putaran jika paket yang dimilikinya lebih kecil atau sama dengan ambang batas.

Estimator pada HTB-TBF (*Token Bucket Filter*)

TBF merupakan estimator yang digunakan oleh HTB untuk menentukan suatu kelas/prioritas berada dalam keadaan *underlimit, atlimit*, atau *overlimit*. TBF bekerja dengan algoritma ember token, jika token tidak tersedia di dalam ember maka paket-paket yang akan dikirim harus menunggu sampai tersedia token untuk mengirim paket yang menunggu.

General Scheduler CBQ-EWMA (*Exponential Weight Moving Average*)

EWMA merupakan estimator yang digunakan oleh CBQ dengan prinsip kerja menghitung waktu paket masuk dan menggunakan rata-rata untuk menghitungnya. Dalam EWMA dikenal isilah *avgidle* yang berarti selang waktu antara paket datang. EWMA berfungsi untuk mengetahui waktu *avgidle*.

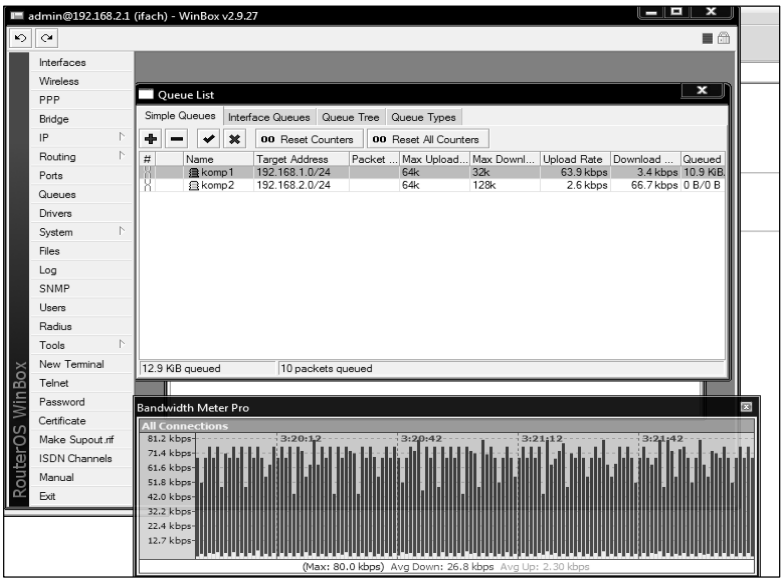
Estimator pada CBQ-WRR (*Weight Round Robbin*)

WRR merupakan mekanisme penjadwalan dalam CBQ dengan prinsip kerja hampir sama dengan DRR. WRR menggunakan metode *prio* untuk meneruskan paket. Sebelum diteruskan paket diklasifikasikan sesuai prioritas yang telah ditentukan.

Penggunaan HTB (*Hirarchy Token Bucket*)

Pada dasarnya Mikrotik telah menggunakan HTB untuk algoritma pembagian *bandwidth* yang terdapat pada menu di *winbox simple queue* dan *queue tree*.

Hasil Analisa Menggunakan HTB pada Mikrotik



Gambar 13: *Download Traffic* menggunakan HTB

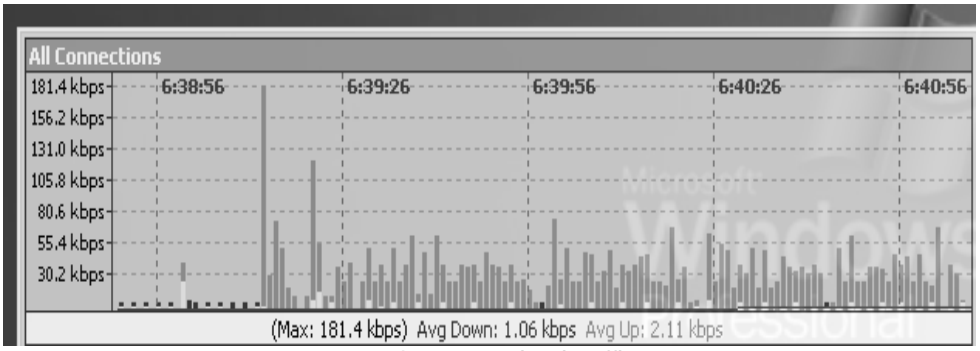
Hasil analisa menggunakan mikrotik didasarkan pada *proses upload* dan *download*, dengan alokasi untuk *upload* = 32K dan *download* = 64K, seperti Tabel 1 untuk *upload* = 32K dan *download* = 128K seperti Tabel 2.

Tabel 1: *Upload* = 64K dan *download* = 32K pada HTB

IP	*Upload*			*Download*		
	Ukuran file	Max	Min	Ukuran file	Max	Min
C1 192.168.1.4	622MB	4.4Mbps	0Mbps	4.75MB	164.4kbps	8kbps
	152MB	2.4Mbps	2.4Mbps	1.7MB	195.3kbps	16kbps

Tabel 2: *Upload* = 32K dan *download* = 128K pada HTB

IP	*Upload*			*Download*		
	Ukuran file	Max	Min	Ukuran file	Max	Min
C1 192.168.1.4	1.5MB	162.5kbps	8kbps	4.66MB	9.95kbps	6kbps
	14KB	124.5kbps	5kbps	1MB	56.4kbps	7kbps
	46MB	139.3kbps	20kbps	146KB	56.4kbps	6kbps



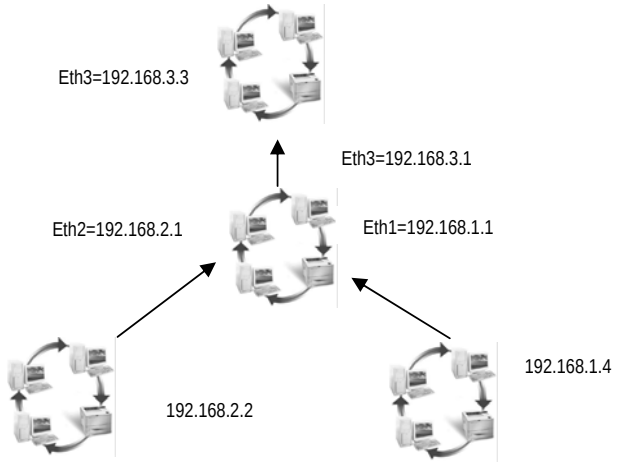
Gambar 14: *Upload traffic*

Dari hasil pengujian tampak bahwa:

3. Nilai maksimal *upload* dan *download* melebihi *bandwidth* yang ditentukan, karena adanya peminjaman *bandwidth* dari klien lain yang tidak terpakai.
4. Kecepatan *upload* dan *download* cenderung lebih stabil.

Penggunaan CBQ (*Class* Based Queue) pada Linux

Sebelum menjalankan CBQ, terlebih dahulu Linux dikonfigurasikan sebagai *router*. Bentuk topologi yang digunakan dalam penelitian ini adalah seperti Gambar 15.



Gambar 15: Topologi jaringan

Untuk konfigurasi *router* pada linux, pertama harus masuk sebagai "*super user*". Untuk melakukan perintah *routing*, periksa dahulu *interface network* yang tersedia dengan perintah:

```
root@ifach-desktop:~# ifconfig
```

Tabel *routing* akan berada dalam keadaan default, hal ini bisa dilihat dengan perintah:

```
root@ifach-desktop:~# route –n
```

Kemudian dilakukan konfigurasi *routing* sebagai berikut:

```
root@ifach-desktop:~# route add –net 192.168.1.0/24 gw 192.168.2.1
root@ifach-desktop:~# route add –net 192.168.2.0/24 gw 192.168.1.2
```

Untuk memeriksa apa yang telah dilakukan, dapat dilihat dengan perintah:

```
root@ifach-desktop:~# route –n
```

Kemudian dilakukan perintah untuk *forwarding*. Pastikan *ip_forwarding* dalam keadaan aktif, yaitu bernilai 1. Apabila masih bernilai 0, maka dapat dirubah dengan perintah:

```
root@ifach-desktop:~# cat /proc/sys/net/ipv4/ip_forward
root@ifach-desktop:~# gedit /proc/sys/net/ipv4/ip_forward
```

CBQ tidak secara *default* terdapat dalam paket linux, sehingga perlu dilakukan instalasi dan konfigurasi. Pertama harus masuk sebagai *root*, kemudian *download* file *cbq.init* menggunakan perintah:

```
root@ifach-desktop:~# wget http//:mesh.dl.sourceforge.net/sourceforge/cbqinit/cbq.init-v0.7.3
```

File yang telah di-*download* kemudian di-*copy* ke direktori **/usr/bin/** dengan perintah:

```
root@ifach-desktop:~# cp cbq.init-v0.7.3 /usr/bin/cbq.init
```

Kemudian dibuat file untuk eksekusi pada direktori tersebut dengan perintah:

```
root@ifach-desktop:~# chmod +x/usr/bin/bin/cbq.init
```

Kemudian dibuat file konfigurasinya dengan perintah:

```
root@ifach-desktop:~# mkdir /etc/sysconfig/cbq
root@ifach-desktop:~# gedit /etc/sysconfig/cbq/cbq-128.clint
```

dan diisi dengan:

```
DEVICE=eth0,100mbit,1mbit
RATE=28kbit
WEIGHT=2kbit
PRIO=5
RULE=192.168.1.0/24
```

Setelah dikompilasi, kemudian dijalankan *cbq.init* dan akan didapatkan informasi tentang berapa *byte* yang dikirim, berapa paket yang dikirim, berapa paket yang di-*drop*, *overlimit* dan *request*.

Hasil Analisa Pembagian *Bandwidth* Menggunakan CBQ

Hasil analisa menggunakan mikrotik didasarkan pada proses *upload* dan *download*, dengan alokasi untuk *upload* = 32K dan *download* = 64K seperti Tabel 3 dan untuk *upload* = 32K dan *download* = 128K seperti Tabel 4.

Tabel 3: *Upload* = 64K dan *download* = 32K pada CBQ

IP	*Download*			*Upload*		
	Ukuran File	Max	Min	Ukuran File	Max	Min
C1 192.168.1.4	622Mb	5.0Mbps	3.6Mbps	622Mb	5.2Mbps	3.2Mbps
	152Mb	5.0Mbps	3.6Mbps	152Mb	5.1Mbps	4.6Mbps
	1.28MB	67.6kbps	9.8kbps	240K	78.2kbps	8kbps

Tabel 4: *Upload* = 32K dan *download* = 128K pada CBQ

IP	*Download*			*Upload*		
	Ukuran File	Max	Min	Ukuran File	Max	Min
C1 192.168.1.4	6.25MB	77.3kbps	8.7kbps	4.75MB	71.2kbps	8.7kbps
	26KB	62.4kbps	9.8kbps	1.7MB	61.3kbps	13kbps
	1.28MB	66.6kbps	9.8kbps	240K	53.9kbps	9kbps

Dari hasil-hasil di atas tampak bahwa:

1. Pengaturan *bandwidth* pada CBQ lebih rumit.
2. Rata-rata pemakaian *bandwidth* untuk *download* dan *upload* seimbang, dikarenakan alokasi *bandwidth* yang dberikan untuk *download* dan *upload* tidak dibedakan.
3. Kecepatan *download* dan *upload* cenderung stabil.
4. Ukuran file mempengaruhi kecepatan dan kestabilan *banwidth*.
5. Alokasi *bandwidth* mempengaruhi kecepatan dan kestabilan *download* dan *upload*.
6. Kestabilan dan kecepatan *transfer* data cenderung sama, tergantung alokasi *bandwidth* yang diberikan dan ukuran data.
7. Jika tidak diberikan alokasi *bandwidth* tidak bisa untuk *download* dan *upload*, meskipun antar klien terhubung ke PC router.

KESIMPULAN

Hasil atas penelitian ini adalah sebagai berikut:

1. Pembagian *bandwidth* dapat dilakukan berdasarkan IP ataupun berdasarkan *port* atau *class*.

2. Perbedaan antara algoritma CBQ dan HTB terletak pada *scheduler* dan *estimator* yang digunakan. HTB menggunakan *scheduler DRR* (*Deficit Round Robin)* dan *estimator* menggunakan TBF (*Token Bucket Filter*), sedangkan *scheduler* pada CBQ menggunakan WRR *(Weight Round Robbin)* dan *estimator* menggunakan EWMA *(Exponent Weight Moving Averege).*
3. Kestabilan dan kecepatan transfer data cenderung sama, tergantung alokasi *bandwidth* yang diberikan dan ukuran data.
4. Jika tidak diberikan alokasi *bandwidth*, tidak bisa untuk *download* dan *upload*, meskipun antar klien terhubung ke PC *router*.
5. Pembagian *bandwidth* dengan menggunakan mikrotik lebih mudah, karena ada aplikasi untuk *remote* dari klien menggunakan *winbox*.

Saran

1. Penelitian ini hanya dilakukan pada jaringan lokal, selanjutnya bisa dikembangkan pada jaringan besar yang terkoneksi dengan internet.
2. Agar pebandingan algoritma HTB dan CBQ lebih fokus, sebaiknya menggunakan satu OS saja, misal Linux karena Linux mendukung untuk algoritma HTB dan CBQ.
3. Untuk mempermudah menjalankan CBQ, bisa dibuat aplikasi untuk mengatur *bandwidth* dengan model GUI *(graphic user interface)* seperti *winbox*.

DAFTAR PUSTAKA

Atma Ika, Satya, 2006**,** Mengenal dan Menggunakan Mikrotik Winbox, D@atakom Lintas Buana, Jakarta

Devera, Martin, 1997, http://luxik.cdi.cz/~devik/qos/htb

Devera, Martin, 2002, http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

HTB Means, http://luxik.cdi.cz/~devik/qos/htb/old/htbmeas1.htm

HTB Queuing Discipline, http://www.opalsoft.net/qos/DS-28.htm

Hubert, Bert, Classful Queueing Disciplines, http://lartc.org/howto/lartc.qdisc. classful.html

Instalasi, Konfigurasi dan Optimasi Mikrotik RouterOS http://www.kecoak-elektronik.net

Komunitas Indonesia Open Source, 2008, http://opensource.telkomspeedy.com /forum/index.php

Ropix, 2006, Mikrotik OS Untuk Bandwidth management, http://www.ilmukomputer.com

Santoso, Budi, 2007, Manajemen Bandwidth Internet dan Intranet, http://budisantoso.com

Yuda, Dewa, 2007, Hierarchical Token Bucket (HTB), http://omyudha.multiply.com/journal /item/8

Situs:

http://www.gnu.org/tc-htb.html
http://lartc.org/howto/lartc.qdisc.terminology.html
http://www.luxik.cdi.cz/~devik/qos/htb/
http://www.mikrotik.com/testdocs/ros/3.0/qos/queue.pdf"
http://www.mikrotik.com
http://digilib.petra.ac.id/ads-cgi/viewer.pl/jiunkpe/s1/info/2004/jiunkpe-ns-s1-2004-26400010-4191-router-chapter2.pdf
http:// www.vanillamist.com
http://www.wikipedia.com/Class_based_weighted_fair_queuing.htm
http://www.wikipedia.com/Hierarchical_Token_Bucket.htm
http://www.w3.org/1999/xhtml/perhitungan_bandwidth