UNIVERSITAS INDONESIA

MODIFIKASI ALGORITMA *LINK STATE ROUTING* PADA JARINGAN KOMPUTER

SKRIPSI

RADEN RARA DIAN PUSPITA MURTI
0305017046

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
JUNI 2011

UNIVERSITAS INDONESIA

MODIFIKASI ALGORITMA *LINK STATE ROUTING* PADA JARINGAN KOMPUTER

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains

RADEN RARA DIAN PUSPITA MURTI
0305017046

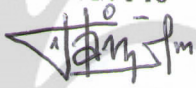
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
JUNI 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**:**	**Raden Rara Dian Puspita Murti**
NPM	**:**	**0305017046**
Tanda Tangan	**:**	
Tanggal	**:**	**14 Juni 2011**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Raden Rara Dian Puspita Murti
NPM	:	0305017046
Program Studi	:	Matematika
Judul Skripsi	:	Modifikasi Algoritma *Link State Routing* pada Jaringan Komputer

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia

DEWAN PENGUJI

Pembimbing I :	Dra. Siti Aminah, M.Kom.	(.....)
Pembimbing II :	Drs. Suryadi M.T., M.T.	(.....)
Penguji I :	Dr. Alhadi Bustamam, M.Kom.	(.....)
Penguji II :	Dhian Widya, S.Si, M.Kom.	(.....)
Penguji III :	Dr. Yudi Satria, M.T.	(.....)

Ditetapkan di : Depok
Tanggal : 14 Juni 2011

KATA PENGANTAR/UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi syarat untuk mencapai gelar Sarjana Sains Departemen Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada :

(1) Dra. Siti Aminah, M.Kom sebagai dosen pembimbing I yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) Drs. Suryadi M.T., M.T. sebagai dosen pembimbing II yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(4) orang tua dan keluarga saya yang telah memberikan bantuan;
(5) semua pihak yang telah membantu saya;
(6) Yanuar Singgih Saputra yang telah berdiskusi dengan saya tentang jaringan komputer;
(7) Ferdy Jamanta yang telah memodifikasi program tabel2; dan
(8) teman-teman di Departemen Matematika yang telah banyak membantu saya.

Saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 14 Juni 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama	:	Raden Rara Dian Puspita Murti
NPM	:	0305017046
Program Studi	:	Matematika
Departemen	:	Matematika
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam
Jenis Karya	:	Skripsi

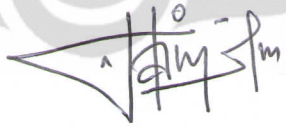
demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Modifikasi Algoritma *Link State Routing* pada Jaringan Komputer

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 14 Juni 2011
Yang menyatakan



(Raden Rara Dian Puspita Murti)

ABSTRAK

Nama : Raden Rara Dian Puspita Murti
Program Studi : Matematika
Judul : Modifikasi Algoritma *Link State Routing* pada Jaringan Komputer

Penelitian ini membahas modifikasi algoritma *link state routing* pada jaringan komputer. Jaringan komputer dalam hal ini direpresentasikan dalam bentuk graf. Algoritma *link state routing* yang biasa dipakai adalah algoritma yang menghasilkan tabel *routing* dengan menggunakan algoritma Dijkstra dalam menemukan lintasan terpendek. Pada modifikasi algoritma *link state routing*, sebelum digunakan algoritma Dijkstra, graf dikelompokkan terlebih dahulu dengan menggunakan MST *clustering*, yang dalam membentuk *cluster* menggunakan algoritma Zahn. Pada modifikasi algoritma *link state routing* ini, entri dari tabel *routing* berkurang, sehingga proses di *router* menjadi lebih cepat.

Kata kunci :
algoritma *link state routing*, entri tabel *routing*, jaringan komputer, modifikasi algoritma *link state routing*, *router*

ABSTRACT

Name	:	Raden Rara Dian Puspita Murti
Study Program	:	Mathematics
Title	:	Modification of Link State Routing Algorithm on Computer Network

This research is about modification of link state routing algorithm on computer network. Computer network in this case is represented as graph. The link state routing algorithm that usual to be used is algorithm that produces routing table by using Dijkstra's algorithm in finding shortest path. In modification of link state routing algorithm, before using Dijkstra's algorithm, graph is grouped formerly by using MST clustering, which in forming cluster using Zahn's algorithm. In this modification of link state routing algorithm, routing table entries reduce, so process in router becomes faster.

Key words :
computer network, link state routing algorithm, modification of link state routing algorithm, router, routing table entries

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan komputer semakin hari semakin pesat. Pada mulanya, orang menggunakan komputer terbatas pada hanya mengerjakan tugas-tugas perseorangan tanpa ada komunikasi data antara komputer yang satu dengan yang lainnya. Namun, pada perkembangannya sampai saat ini kebutuhan terhadap komunikasi data semakin meningkat sehingga diperlukan komputer yang dapat terhubung dengan komputer lainnya sehingga membentuk jaringan komputer. Jaringan komputer semakin meningkat. Berdasarkan data dari www.internetworldstats.com, pada 31 Desember 2000, pengguna *Internet* di dunia sebanyak 360.985.492. Di tahun 2010, pengguna *Internet* meningkat menjadi 1.966.514.816. Terjadi peningkatan sebesar 444,8 %.

Jaringan komputer adalah hubungan antara satu komputer dengan komputer lain dimana terjadi pengiriman dan penerimaan data. Jaringan komputer mempunyai manfaat yaitu berbagi pakai (*resource sharing*). Berbagi pakai bertujuan agar seluruh program, peralatan, dapat digunakan oleh setiap orang yang ada pada jaringan tanpa terpengaruh oleh lokasi dan pemakainya. Manfaat lainnya yaitu mendapatkan keandalan tinggi dengan mempunyai sumber-sumber alternatif yang tersedia dan menghemat uang (Sugeng, 2006).

Jaringan komputer dikategorikan menjadi tiga yaitu *Local Area Network* (LAN), *Metropolitan Area Network* (MAN), dan *Wide Area Network* (WAN). Banyak jaringan di dunia ini seringkali menggunakan perangkat keras dan perangkat lunak yang berbeda. Banyak orang ingin menggabungkan seluruh jaringan yang ada sehingga kebutuhan komunikasi antar komputer akan mudah dipenuhi. Jawaban semua ini adalah dengan *internetwork* atau *internet*. Jika sistem yang dihubungkan tidak *compatible*, maka diperlukan mesin *gateway* untuk keperluan penerjemahan, dan untuk menyampaikan paket yang diterima (Sugeng, 2006). Usaha dari menghubungkan antara LAN satu dengan LAN lainnya untuk membentuk jaringan

WAN satu dengan WAN lainnya sehingga terbentuk jaringan WAN yang besar dinamakan *internetworking* (Sugeng, 2006).

Internetworking terdiri dari jaringan yang menggunakan protokol yang berbeda/tidak sejenis sehingga menjadi sangat kompleks (Sugeng, 2006). Hampir semua protokol berjalan di atas protokol *Transmission Control Protocol/Internet Protocol* (TCP/IP). Tetapi, ada juga yang berjalan di atas SNA IBM, beberapa perusahaan telepon mengoperasikan jaringan ATM, beberapa LAN ada yang menggunakan Novell NCP/IPX (Tanenbaum, 2003). Untuk mengubah jaringan tidak sejenis menjadi jaringan sejenis diperlukan protokol yang menghubungkan antara satu komputer dengan komputer lain yang dinamakan *Transmission Control Protocol/Internet Protocol* (TCP/IP) dan hasilnya adalah *internet* (Sugeng, 2006).

Komputer yang satu dengan yang lainnya dapat saling terhubung menggunakan alamat *Internet Protocol* (IP) yang bahasa awamnya disebut kode pengenal komputer pada jaringan. Alamat IP merupakan komponen penting pada *internet* karena tanpa alamat IP seseorang tidak akan dapat terhubung ke *internet* (Sugeng, 2006). Paket dikirim dari satu alamat IP ke alamat IP lainnya berdasarkan alamat IP yang terdapat dalam tabel *routing*.

Tabel *routing* adalah tabel yang terdapat di *router*, yang terdiri dari dua kolom. Kolom pertama berisi alamat-alamat individu *Internet* atau *range-range* alamat yang dinyatakan dalam urutan bilangan. Kolom kedua, berisi *hop* berikutnya yang cocok dari lintasan ke alamat yang berkorespondensi (Clark, 2003). *Hop* adalah transmisi satu titik ke titik lainnya dalam sebuah rangkaian yang diperlukan untuk mendapatkan sebuah pesan dari titik A ke titik B pada sebuah jaringan (Howe, 1997, p.1).

Alamat IP harus unik. Alamat IP ini berjumlah terbatas karena jumlah *bit* pada alamat IP pun terbatas. Alamat IP versi 4 mempunyai batas maksimum 32 *bit* dan alamat IP versi 6 mempunyai batas maksimum 128 *bit*. Saat ini jaringan komputer semakin bertambah sehingga alamat IP bertambah. Jika alamat IP bertambah maka rute pada jaringan bertambah, sehingga entri-entri tabel *routing* bertambah. Hal ini mengakibatkan ukuran tabel *routing* pada *router* semakin besar

sehingga waktu pemrosesan di *router* menjadi lama. Untuk mengatasinya, diperlukan cara untuk meminimalkan rute pada jaringan dan mereduksi ukuran tabel *routing* sehingga tabel *routing* menjadi minimal dan proses di *router* menjadi cepat.

Untuk membuat tabel *routing*, dapat digunakan algoritma *link state routing* yang didalamnya menggunakan algoritma Dijkstra. Berdasarkan informasi dari http://www.ecgalerycomputer.co.cc, terdapat beberapa masalah pada algoritma *link state routing*, yaitu membutuhkan sumber daya yang tinggi dalam mendapatkan informasi topologi jaringan, menghasilkan *traffic* yang sangat tinggi pada saat pertama kali membanjiri jaringan, dan memungkinkan *delay* atau *lost* yang dapat menyebabkan jaringan tidak konsisten sehingga dilakukan modifikasi terhadap algoritma tersebut. Modifikasi tersebut dinamakan algoritma *link state routing* dengan *clustering* untuk mereduksi ukuran tabel *routing* dengan cara membagi sebuah jaringan komputer yang besar menjadi kelompok-kelompok jaringan yang lebih kecil. Kelompok tersebut dinamakan sebagai daerah (*region*). Pembagian daerah tersebut mereduksi ukuran tabel *routing*.

1.2 Perumusan Masalah

a. Bagaimana proses pembuatan tabel *routing* di suatu jaringan komputer dengan algoritma *link state routing*?

b. Bagaimana usaha untuk memodifikasi algoritma *link state routing* sebagai alternatif dalam membuat tabel *routing* di suatu jaringan komputer?

1.3 Tujuan Penelitian

a. Untuk mengetahui proses pembuatan tabel *routing* di suatu jaringan komputer dengan algoritma *link state routing*.

b. Untuk memodifikasi algoritma *link state routing* sebagai alternatif dalam membuat tabel *routing* di suatu jaringan komputer.

1.4 Manfaat Penelitian

a. Algoritma *link state routing* dengan *clustering* merupakan alternatif algoritma dalam *router* untuk mereduksi ukuran tabel *routing*.

b. Mempercepat proses pengiriman data dalam jaringan komputer.

1.5 Batasan Penelitian

a. Model topologi jaringan menggunakan model topologi bintang.

b. Protokol menggunakan protokol TCP/IP.

c. Alamat IP menggunakan alamat IP versi 4 (IPv4) yang mempunyai 32 *bit*.

1.6 Model Operasional Penelitian

Model operasional penelitian ini adalah studi kepustakaan/studi literatur, yaitu mencari bahan pustaka yang sesuai dengan topik penelitian tentang jaringan komputer kemudian membuat model sistem jaringan dalam bentuk graf terhubung dan tidak berarah. Setelah itu dilakukan simulasi program terhadap algoritma-algoritma yang digunakan untuk meminimalkan rute pada topologi jaringan.

1.7 Sistematika Penulisan

Sistematika penulisan pada skripsi ini terdiri dari Bab 1 Pendahuluan, Bab 2 Tinjauan Pustaka, Bab 3 Algoritma *Link State Routing*, Bab 4 Implementasi dan Simulasi, Bab 5 Kesimpulan dan Saran.

Pada Bab 1 Pendahuluan terdapat Latar Belakang, Perumusan Masalah, Tujuan Penelitian, Manfaat Penelitian, Batasan Penelitian, Model Operasional Penelitian, dan Sistematika Penulisan.

Pada Bab 2 Tinjauan Pustaka terdapat Jaringan Komputer, Representasi Jaringan Komputer dalam Bentuk Graf, MST *Clustering*, dan Pembentukan Tabel *Routing*.

Pada Bab 3 Algoritma *Link State Routing* terdapat Algoritma *Link State Routing* dan Modifikasi Algoritma *Link State Routing*.

Pada Bab 4 Implementasi dan Simulasi terdapat Implementasi dan Simulasi Algoritma *Link State Routing* pada Jaringan Komputer dan Implementasi dan Simulasi dari Modifikasi Algoritma *Link State Routing* pada Jaringan Komputer.

Pada Bab 5 terdapat Kesimpulan dan Saran.

BAB 2
TINJAUAN PUSTAKA

Pada bab 2 ini, dijelaskan mengenai jaringan komputer, representasi jaringan komputer dalam bentuk graf, MST *Clustering*, dan pembentukan tabel *routing*.

2.1 Jaringan Komputer

Penggabungan dari komputer dan komunikasi antar komputer berpengaruh pada cara pengaturan sistem-sistem komputer. Konsep dari "*computer center*" sebagai sebuah ruangan dengan sebuah komputer besar sekarang sudah kuno. Model kuno dari sebuah komputer tunggal untuk mengerjakan tugas-tugas komputasi digantikan dengan beberapa komputer yang saling terhubung. Sistem dengan beberapa komputer saling terhubung untuk mengerjakan tugas-tugas komputasi dinamakan jaringan komputer.

Dua komputer dikatakan saling terhubung jika ada pengiriman dan penerimaan data melalui media transmisi yang menghubungkan kedua komputer tersebut. Hubungan antar komputer dapat menggunakan kabel, *fiber optics*, *microwaves*, *infrared*, dan satelit komunikasi (Tanenbaum, 2003).

2.1.1 Keuntungan-keuntungan Menggunakan Jaringan Komputer

Berikut ini beberapa keuntungan jaringan komputer sehingga orang tertarik pada jaringan komputer dan menggunakannya.

Keuntungan-keuntungan menggunakan jaringan komputer :

1. Berbagi pakai sumber daya

Berbagi pakai bertujuan untuk membuat semua program, data, dan peralatan tersedia ke sembarang orang pada jaringan tanpa memperhatikan lokasi fisik dari sumber daya dan pengguna.

Contoh : Anggap seseorang berada 1000 km jauh dari datanya tetap dapat mengakses data.

2. Keandalan tinggi

Jaringan menyediakan keandalan tinggi dengan mempunyai sumber-sumber alternatif.

Contoh : Anggap semua *file* dapat digandakan pada dua atau tiga mesin, sehingga jika satu mesin tidak tersedia (karena kerusakan perangkat keras), salinan masih dapat digunakan.

3. Biaya rendah/menghemat uang

Penggunaan jaringan komputer dapat menghemat uang dalam hal pemakaian atau pembelian perangkat lunak.

4. Komunikasi

Sebuah jaringan komputer dapat menyediakan sebuah media komunikasi di antara orang yang terpisah. Menggunakan jaringan komputer, mudah untuk dua atau lebih orang untuk menulis sebuah laporan bersama, yaitu ketika satu pembuat membuat perubahan ke dokumen yang dijaga *online* yang lainnya dapat melihat perubahan dengan segera, daripada menunggu beberapa hari untuk sebuah surat (Shinde, 2009).

2.1.2 Topologi-topologi Jaringan Komputer

Sebuah topologi adalah sebuah bentuk koneksi fisik untuk menghubungkan setiap komputer pada sebuah jaringan (Akib, p.1). Menurut Shinde, topologi pada jaringan komputer, yaitu :

1. Topologi Bintang (*Star Topology*)
2. Topologi Bus (*Bus Topology*)
3. Topologi *Mesh* (*Mesh Topology*)
4. Topologi Cincin (*Ring Topology*)
5. Topologi Pohon (*Tree Topology*)

Berikutnya akan dibahas mengenai kategori-kategori dari jaringan komputer.

2.1.3 Kategori-kategori dari Jaringan Komputer

Jaringan komputer dikategorikan menjadi tiga, yaitu LAN, MAN, dan WAN. Berikut dijelaskan secara singkat ketiga kategori tersebut.

1. *Local Area Network* (LAN)

   Sebuah *Local Area Network* umumnya sebuah jaringan dalam sebuah kantor, gedung, kampus yang berjarak beberapa kilometer. Alasan pembuatan LAN untuk berbagi pakai seperti *disk*, *printer*, *program*, dan *data*. LAN dapat juga digunakan untuk berkomunikasi. LAN mempunyai *data rate* dari 4 Mbps sampai ratusan Mbps. LAN dapat menggunakan topologi bintang, bus, atau cincin. Contoh : *Ethernet* LAN, *Token Bus* LAN, *Token Ring* LAN, *Fiber Distributed Data Interface* (Akib, 2009, p.1).

2. *Metropolitan Area Network* (MAN)

   Sebuah *Metropolitan Area Network* adalah jaringan komputer pada sebuah kota. MAN dapat berupa sebuah jaringan tunggal seperti jaringan televisi kabel atau jaringan yang terdiri dari banyak LAN yang tergabung dalam sebuah jaringan yang lebih besar sehingga sumber daya dapat dibagi pada LAN ke LAN seperti pada komputer ke komputer. Sebuah MAN biasanya dimiliki dan dioperasikan oleh perusahaan seperti perusahaan telepon. Sebuah MAN mempunyai cakupan geografi lebih luas dibandingkan dengan sebuah LAN dan jaraknya berkisar antara 10 kilometer hingga beberapa ratus kilometer.

3. *Wide Area Network* (WAN)

   Sebuah *Wide Area Network* dibangun untuk menghubungkan sistem komputer pada cakupan geografi yang luas, seperti negara, benua, bahkan dunia. Kecepatan WAN berkisar antara 1,5 Mbps hingga 100 Gbps. WAN menggunakan alat-alat komunikasi publik atau pribadi atau kombinasi antara keduanya sehingga dapat menjangkau cakupan geografi yang luas. Contoh WAN adalah *internet* yang mempunyai sebuah hubungan ke jaringan-jaringan komputer di negara lain (Shinde, 2009).

Jaringan LAN, MAN, WAN yang saling terhubung membentuk jaringan yang besar yang dinamakan *internet*.

2.1.4 *Internet*

Banyak jaringan komputer dengan perangkat keras dan perangkat lunak yang berbeda-beda. Orang yang terhubung dalam suatu jaringan ingin berkomunikasi dengan orang lain yang berada di jaringan yang berbeda. Pemenuhan dari keinginan ini membutuhkan hal yang berbeda dan sering juga jaringan yang *incompatible* dapat dihubungkan dengan menggunakan mesin yang bernama *gateway* untuk membuat koneksi dan menerjemahkan baik dalam perangkat keras ataupun perangkat lunak. Sebuah kumpulan dari jaringan yang saling terhubung dinamakan *internetwork* atau *internet* (Tanenbaum, 2003).

2.1.5 Model Referensi

Terdapat dua model referensi yang merupakan arsitektur jaringan komputer. Hal ini berguna untuk mengurangi kerumitan rancangan sehingga jaringan diorganisasikan sebagai suatu tumpukan lapisan (*layer*) atau level. Jumlah, nama, isi, dan fungsi setiap lapisan dapat berbeda dari jaringan yang satu dengan jaringan yang lain. Tujuan suatu lapisan adalah untuk memberikan layanan kepada lapisan yang berada di atasnya. Hukum dan konvensi yang dipakai dalam pembicaraan antar lapisan pada mesin yang berbeda adalah menggunakan protokol. Protokol adalah sebuah persetujuan semua pihak yang berkomunikasi tentang bagaimana komunikasi tersebut harus dilakukan. Selanjutnya, terdapat dua model referensi, yaitu model referensi OSI (*Open System Interconnection*) dan model referensi TCP/IP (Sugeng, 2006).

1. Model Referensi OSI
    a. Lapisan Fisik (*Physical Layer*)
    b. Lapisan *Data Link* (*Data Link Layer*)
    c. Lapisan Jaringan (*Network Layer*)

d. Lapisan *Transport* (*Transport Layer*)

e. Lapisan Sesi (*Session Layer*)

f. Lapisan Presentasi (*Presentation Layer*)

g. Lapisan Aplikasi (*Application Layer*)

2. Model Referensi TCP/IP

a. Lapisan *Host to Network* (*Host to Network Layer*)

b. Lapisan *Internet* (*Internet Layer*)

c. Lapisan *Transport* (*Transport Layer*)

d. Lapisan Aplikasi (*Application Layer*)

Perbedaan pada model referensi OSI dan TCP/IP terletak pada jumlah lapisan (Tanenbaum, 2003).

2.1.6 Pengalamatan pada Jaringan *Internet*

Jaringan komputer besar yang dinamakan *internet* mempunyai alamat untuk dapat terhubung antara satu komputer dengan komputer lainnya. Alamat *Internet Protocol* disingkat alamat IP merupakan kode pengenal pada komputer. Penelitian ini menggunakan alamat IP versi 4. Selanjutnya, akan dibahas mengenai pengalamatan *Internet* dengan IP versi 4 (IPv4).

Pengalamatan *Internet* IPv4

Agar *Internet Protocol* (IP) dapat mengantarkan paket-paket IP ke tujuan-tujuan yang diinginkan, masing-masing alat yang terhubung ke sebuah jaringan IP harus secara unik diidentifikasi oleh sebuah alamat IP. Alamat IP versi 4 mempunyai panjang 32 *bit*. Alamat IP versi 4 digunakan dalam IP *packet header* untuk mengidentifikasi sumber dan tujuan dari paket dan penting untuk komunikasi data menggunakan *Internet Protocol*.

Alamat IPv4 dapat diklasifikasikan ke dalam dua jenis alamat, yaitu IP publik dan alamat IP privat. Alamat IP publik unik. Kedua versi mempunyai sebuah format umum biasanya ditulis sebagai deret dari empat bilangan bulat desimal (masing-masing nilai antara 0 dan 255) dipisahkan oleh titik.

Format alamat IPv4

d.d.d.d

dengan d bilangan bulat desimal antara 0 sampai 255 (Clark, 2003).

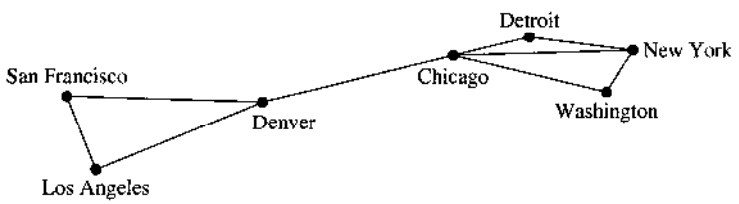
2.2 Representasi Jaringan Komputer dalam Bentuk Graf

Sebelum dibahas representasi jaringan komputer dalam bentuk graf, berikut ini penjelasan mengenai teori-teori yang digunakan.

Definisi 1

Sebuah graf $G(V, E)$ terdiri dari V sebuah himpunan tidak kosong dari simpul dan E sebuah himpunan dari busur. Masing-masing busur mempunyai satu atau dua simpul yang dihubungkan dengan titik-titik akhirnya. Sebuah busur yang demikian dikatakan terhubung dengan titik-titik akhirnya.

Sekarang, anggap bahwa sebuah jaringan disusun dari pusat-pusat data dan hubungan-hubungan komunikasi antara komputer-komputer. Kita dapat menggambarkan lokasi dari masing-masing pusat data oleh sebuah titik dan masing-masing hubungan komunikasi oleh sebuah segmen garis seperti ditunjukkan oleh Gambar 2.1. Jaringan komputer ini dapat dimodelkan dengan sebuah graf yang mana simpul-simpul dari graf mewakili pusat-pusat data dan busur-busur mewakili hubungan-hubungan komunikasi.



Gambar 2.1 Sebuah Jaringan Komputer

(Rosen, 2007)

Catat bahwa masing-masing busur dari graf yang mewakili jaringan komputer ini menghubungkan dua simpul berbeda. Yaitu, tidak ada busur menghubungkan sebuah simpul dengan dirinya sendiri. Lebih jauh, tidak ada dua busur berbeda menghubungkan pasangan simpul yang sama. Sebuah graf yang mana masing-masing busur menghubungkan dua simpul berbeda dan dimana tidak ada dua busur menghubungkan pasangan simpul yang sama dinamakan sebuah **graf sederhana**.

Sebuah graf dikatakan **graf tidak berarah** karena busur-busur mereka tidak mempunyai arah. Sedangkan jika busur-busur pada graf mempunyai arah maka graf dikatakan **graf berarah** (Rosen, 2007).

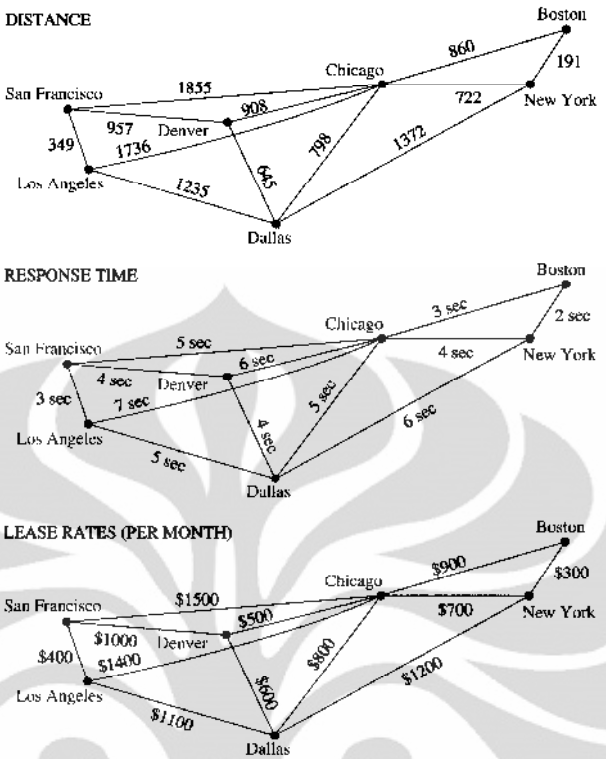
Berikutnya, akan dibahas mengenai sirkuit pada graf. Misalkan v dan w menjadi simpul dalam graf. Sebuah **jalan** antara v dan w dalam graf adalah sebuah barisan berganti-ganti hingga $v = v_0, e_1, v_1, e_2, v_2, e_3, ..., e_n, v_n = w$ dari simpul-simpul dan busur-busur dari graf sehingga masing-masing busur e_i dalam barisan menghubungkan simpul v_{i-1} ke simpul v_i. Sebuah **jalan tertutup** dalam sebuah graf adalah sebuah jalan dimana sebuah simpul akan kembali ke dirinya sendiri. Sebuah jalan tertutup yang mana tidak ada busur yang berulang adalah sebuah **sirkuit** (Balakrishnan, 1997).

Definisi 2

Dua simpul u dan v dalam sebuah graf tidak berarah G dinamakan bertetangga dalam G jika u dan v adalah titik-titik akhir dari sebuah busur dari G. Jika e dihubungkan dengan $\{u,v\}$, busur e dinamakan hadir dengan simpul u dan v. Busur e terhubung ke u dan v. Simpul u dan v dinamakan dari sebuah busur yang dihubungkan dengan $\{u,v\}$.

Graf Terboboti

Graf yang mempunyai sebuah bilangan yang dipetakan ke masing-masing busurnya dinamakan graf terboboti. Graf ini digunakan untuk memodelkan jaringan-jaringan komputer. Seperti yang tampak pada Gambar 2.2.



Gambar 2.2 Graf-graf Terboboti Memodelkan sebuah Jaringan Komputer

(Rosen, 2007)

Selanjutnya, akan dibahas mengenai algoritma untuk mencari lintasan terpendek dari simpul sumber ke simpul tujuan.

2.2.1 Algoritma Dijkstra

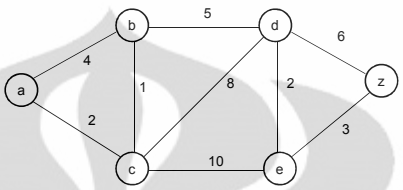
Ada beberapa algoritma berbeda untuk menemukan sebuah lintasan terpendek antara dua simpul dalam sebuah graf terboboti. Sekarang, akan dihadirkan sebuah algoritma yang ditemukan oleh Edsger Dijkstra di tahun 1959. Versi yang akan digambarkan menyelesaikan masalah ini dalam graf-graf terboboti tidak berarah dimana semua bobotnya positif.

Algoritma Dijkstra

1. **procedure** *Dijkstra*(G: graf sederhana terhubung terboboti dengan semua bobot positif)
   {G mempunyai simpul $a = v_0, v_1, ..., v_n = z$ dan bobot $w(v_i, v_j)$ dimana $w(v_i, v_j) = \infty$ jika $\{v_i, v_j\}$ bukan sebuah busur di G}
2. **for** $i := 1$ **to** n
3. $L(v_i) := \infty$
4. $L(a) := 0$
5. $S := \varnothing$
   {label sekarang diinisialisasi sehingga label dari a adalah 0 dan semua label lain adalah ∞ dan S adalah himpunan kosong}
6. **while** $z \notin S$
7. **begin**
8. $u :=$ sebuah simpul tidak dalam S dengan $L(u)$ minimal
9. $S := S \cup \{u\}$
10. **for** semua simpul v tidak dalam S
11. **if** $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
    {ini menambahkan sebuah simpul ke S dengan label minimal dan perbarui label-label dari simpul-simpul tidak dalam S}
12. **end** {$L(z) =$ panjang dari sebuah lintasan terpendek dari a ke z}

(Rosen, 2007).

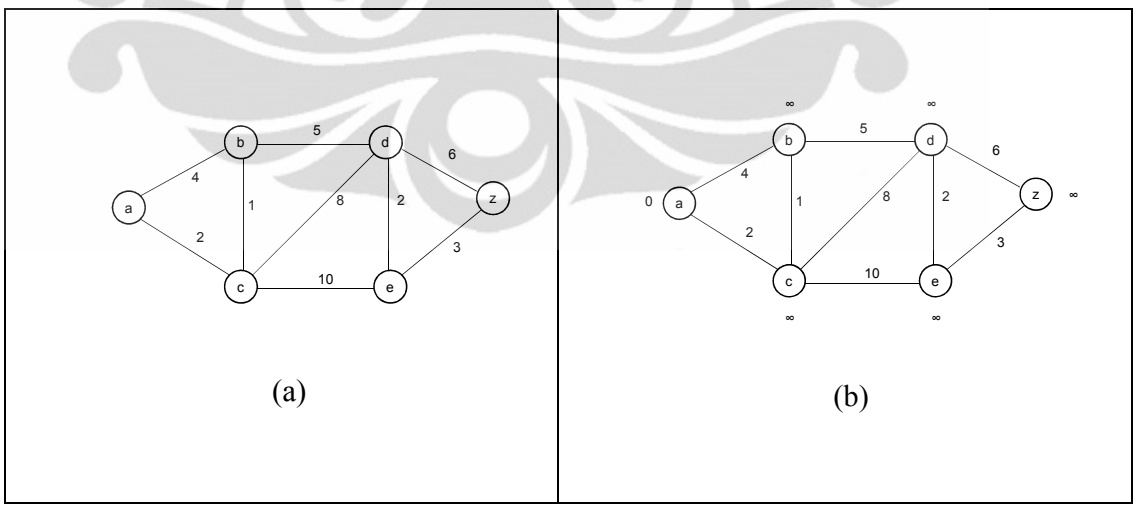
Berikut contoh penggunaan algoritma Dijkstra untuk menemukan lintasan terpendek dari simpul a ke simpul z.



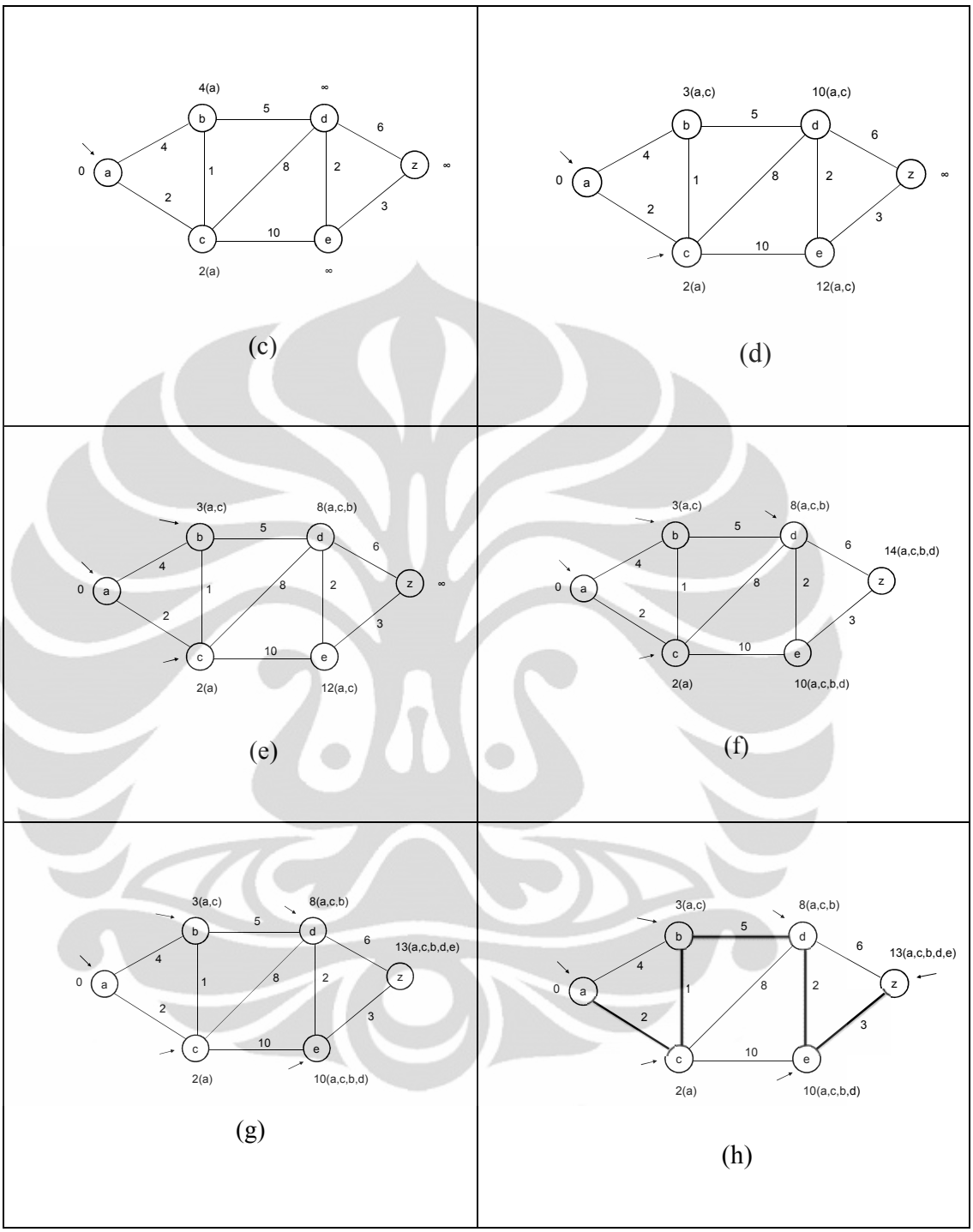
Gambar 2.3 Graf Terboboti G_1 dengan Enam Simpul

Contoh 2.1

Diberikan graf terboboti seperti graf G_1 pada Gambar 2.3. Akan dicari lintasan terpendek dari a ke z.



Gambar 2.4 Penggunaan Algoritma Dijkstra untuk Menemukan Lintasan Terpendek dari a ke z



Gambar 2.4 Penggunaan Algoritma Dijkstra untuk Menemukan Lintasan Terpendek dari a ke z (Lanjutan)

Penjelasan Gambar 2.4 :

(a) Graf G_1 pada Gambar 2.3.

(b) Pertama, labeli semua simpul dengan ∞ kemudian inisialisasi simpul sumber a dengan 0. (Baris 1 sampai 4 dalam algoritma Dijkstra).

Proses pada baris ke 6 sampai 12 dalam algoritma Dijkstra sebagai berikut :

(c) Labeli simpul yang keluar dari a, yaitu simpul b dengan 4(a) dan c dengan 2(a). Simpul-simpul d, e, dan z tetap ∞. Arah panah menunjukkan simpul yang sedang dijalankan.

(d) Karena label c, yaitu 2(a) lebih kecil daripada label b, yaitu 4(a) maka perjalanan menuju simpul c. Kemudian, labeli simpul yang keluar dari c, yaitu simpul b dengan 3(a,c), simpul d dengan 10 (a,c), dan simpul e dengan 12(a,c). Selanjutnya, dipilih label terkecil, yaitu b dengan 3(a,c).

(e) Perjalanan sudah berada di simpul b. Satu-satunya busur yang keluar dari simpul b adalah busur yang menuju simpul d. Labeli simpul d dengan 8(a,c,b). Perjalanan menuju simpul d.

(f) Perjalanan sudah sampai di simpul d. Ada tiga busur yang keluar dari simpul d, yaitu menuju simpul c, e, dan z. Tetapi karena simpul c sudah dilalui, label pada simpul c tidak diperbarui atau simpul c tidak dilalui lagi. Labeli simpul e dengan 10(a,c,b,d) dan simpul z dengan 14(a,c,b,d). Karena label simpul e lebih kecil daripada label simpul z, maka dipilih simpul e.

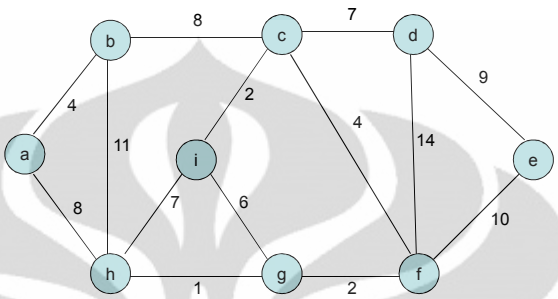
(g) Perjalanan sudah sampai di simpul e. Satu-satunya busur yang belum dilalui adalah simpul z. Perbarui label pada simpul z dengan 13(a,c,b,d,e). Kemudian, pilih simpul z.

(h) Perjalanan sudah mencapai simpul z. Diperoleh lintasan terpendek dari simpul a ke simpul z adalah 13.

Dengan menggunakan algoritma Dijkstra pada graf G_1 diperoleh panjang lintasan terpendek dari a ke z sebesar 13.

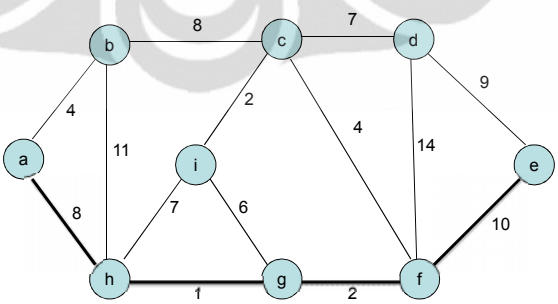
Contoh 2.2

Akan dicari lintasan terpendek untuk graf berikut.



Gambar 2.5 Graf Terboboti G_2 dengan Sembilan Simpul

Dengan menggunakan algoritma Dijkstra pada graf G_2 diperoleh panjang lintasan terpendek dari a ke e sebesar 21 dengan rute a-h-g-f-e yang ditunjukkan pada Gambar 2.6.



Gambar 2.6 Lintasan Terpendek dari Graf G_2

2.2.2 Pohon Perentangan Minimum (*Minimum Spanning Tree*)

Berikut ini diberikan definisi dari pohon.

Definisi 3

Sebuah pohon adalah sebuah graf tidak berarah terhubung dengan tidak ada sirkuit sederhana.

Pohon Perentangan

Pohon perentangan mengandung sebuah subgraf terhubung dengan jumlah busur minimum yang mengandung semua simpul dari graf asli sederhana. Di bawah ini definisi formal dari pohon perentangan.

Definisi 4

Misalkan G adalah graf sederhana. Sebuah pohon perentangan dari G adalah sebuah subgraf dari G yang merupakan sebuah pohon yang mengandung setiap simpul dari G.

Pohon Perentangan Minimum

Sebuah pohon dikatakan pohon perentangan minimum jika jumlah dari bobot-bobot busurnya minimum.

Berikut diberikan definisi tentang pohon perentangan minimum.

Definisi 5

Sebuah pohon perentangan minimum (*minimum spanning tree*) dalam sebuah graf terboboti terhubung adalah sebuah pohon perentangan yang mempunyai jumlah terkecil yang mungkin dari bobot busurnya.

Untuk mencari pohon perentangan minimum dalam sebuah graf dapat digunakan algoritma Kruskal yang ditemukan oleh Joseph Kruskal pada tahun 1956.

Algoritma Kruskal

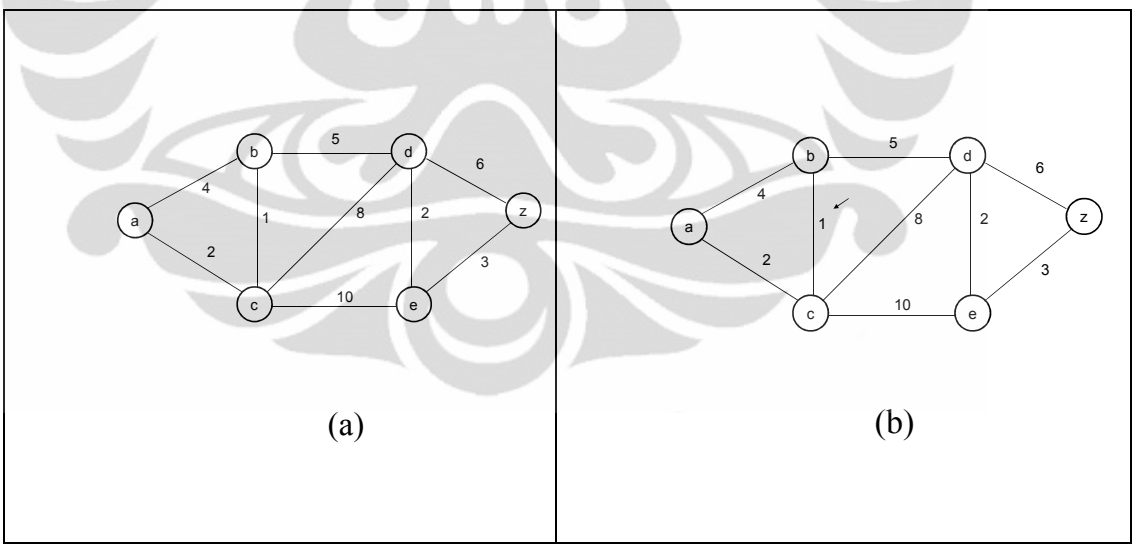
```
1.  procedure Kruskal(G: graf tidak berarah terhubung
    terboboti dengan n simpul)
    G := graf kosong
2.  for i := 1 to n−1
3.  begin
    e := sembarang busur dalam G dengan bobot terkecil
    yang tidak membentuk sebuah sirkuit sederhana
    ketika ditambahkan ke T
    T := T dengan e ditambahkan
    end {T adalah sebuah pohon perentangan minimum dari
    G }
```

(Rosen, 2007).

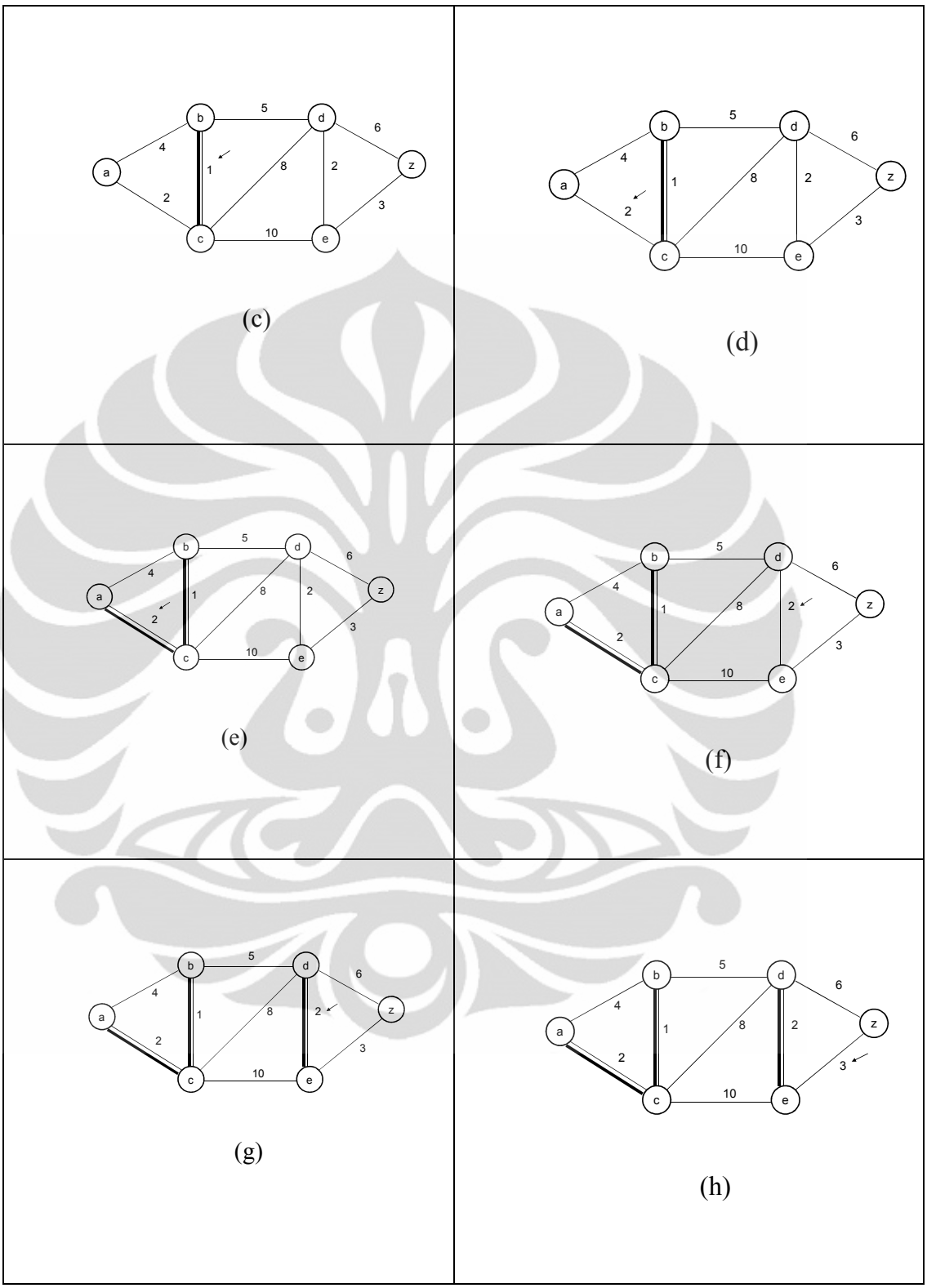
Berikut adalah contoh penggunaan algoritma Kruskal pada graf G_1 (Gambar 2.3).

Contoh 2.3

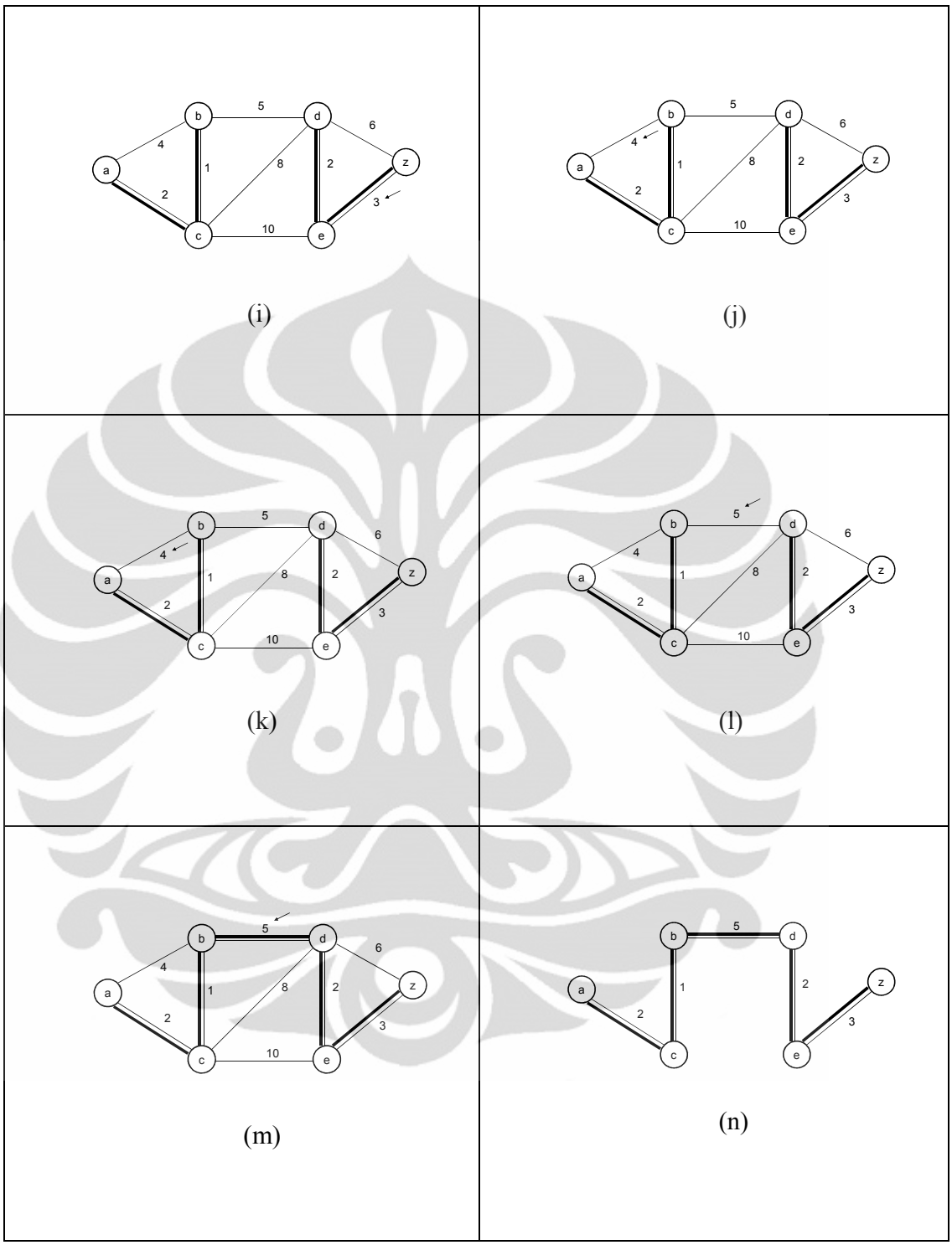
Diberikan graf terboboti seperti graf G_1 pada Gambar 2.3.



Gambar 2.7 Penggunaan Algoritma Kruskal untuk Memperoleh Pohon Perentangan Minimum



Gambar 2.7 Penggunaan Algoritma Kruskal untuk Memperoleh Pohon Perentangan Minimum (Lanjutan)



Gambar 2.7 Penggunaan Algoritma Kruskal untuk Memperoleh Pohon Perentangan Minimum (Lanjutan)

Penjelasan Gambar 2.7.

a. Graf G_1 pada Gambar 2.3.

Proses pada baris kedua dan ketiga dalam algoritma Kruskal sebagai berikut :

b. Pertama, periksa busur yang mempunyai bobot yang paling minimum, yaitu busur {b,c}.

c. Karena tidak terbentuk lingkaran maka beri garis tebal atau busur {b,c} dipilih.

d. Periksa busur kedua yang mempunyai bobot minimum yang belum diperiksa. Ada dua busur yaitu {a,c} dan {d,e}. Dapat diperiksa busur {a,c} terlebih dahulu.

e. Karena {a,c} tidak membentuk lingkaran, beri garis tebal pada busur {a,c}.

f. Periksa busur {d,e}.

g. Karena busur {d,e} tidak membentuk lingkaran, beri garis tebal pada busur {d,e}.

h. Periksa busur {e,z}.

i. Karena busur {e,z} tidak membentuk lingkaran, beri garis tebal pada busur {e,z}.

j. Periksa busur {a,b}.

k. Karena busur {a,b} dapat membentuk lingkaran jika dimasukkan maka busur {a,b} tidak diberi garis tebal.

l. Periksa busur {b,d}.

m. Karena busur {b,d} tidak membentuk lingkaran maka beri garis tebal.

n. Pohon perentangan minimum yang terbentuk.

2.3 MST *Clustering*

Pada skripsi ini digunakan metode *clustering* Pohon Perentangan Minimum Zahn. Zahn mengusulkan untuk membangun sebuah pohon perentangan minimum dan menghapus busur-busur tidak konsisten, yaitu nilai-nilai bobot busur-busur yang secara penting lebih besar daripada bobot rata-rata dari keseluruhan bobot di dalam pohon. Ukuran ketidakkonsistenan digunakan ke masing-masing busur untuk mendeteksi dan menghilangkan busur-busur tidak konsisten yang menghasilkan

sebuah himpunan dari subpohon yang saling lepas, masing-masing subpohon akan mewakili sebuah kelompok terpisah (Prasad dan Rajulu, 2010).

Dalam penelitian ini digunakan algoritma Zahn untuk membagi graf besar ke dalam kelompok graf yang lebih kecil yang dinamakan subgraf.

Algoritma Zahn

Langkah 1 Bentuk pohon perentangan minimum (*Minimum Spanning Tree*) untuk himpunan dari n simpul yang diberikan

Langkah 2 Identifikasi busur tidak konsisten (*inconsistent edge*) pada pohon perentangan minimum

Langkah 3 Hilangkan busur tidak konsisten (*inconsistent edge*) untuk membentuk komponen-komponen terhubung yang disebut *cluster*.

Keterangan :

Sebuah busur dikatakan busur tidak konsisten (*inconsistent edge*) jika bobotnya lebih besar daripada rata-rata bobot busur seluruhnya. Jadi, busur-busur yang tidak konsisten akan dibuang untuk membentuk *cluster-cluster*.

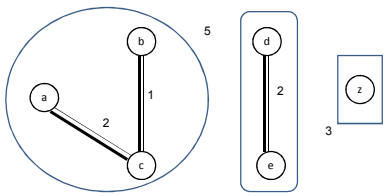
Contoh 2.4

Berikut contoh pohon perentangan minimum pada graf G_1 (Gambar 2.3) yang akan dibentuk *cluster*-nya.

Dari contoh pohon perentangan minimum pada graf G_1 (Gambar 2.3) diperoleh :

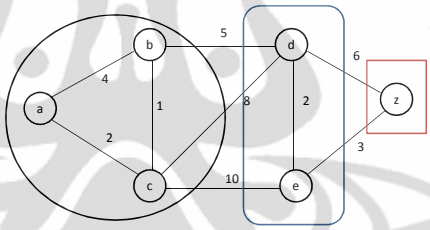
Rata-rata dari seluruh bobot pada pohon perentangan minimum = $\frac{2+1+5+2+3}{5} = \frac{13}{5} = 2,6$.

Busur-busur tidak konsisten yaitu busur berbobot 3 dan 5 karena lebih besar daripada rata-rata. Busur-busur tersebut dihapus dari graf untuk membentuk *cluster*.



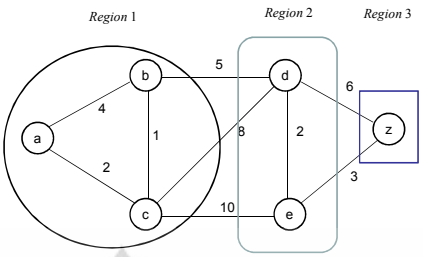
Gambar 2.8 Beberapa *Cluster* yang Dihasilkan dari MST *Clustering* terhadap Graf G_1

Sehingga diperoleh tiga *cluster*, yaitu {a,b,c}, {d,e}, dan {z}. Graf yang sudah terkelompokkan terlihat pada Gambar 2.9.



Gambar 2.9 Graf Terkelompokkan

Lengkapnya, beri nama *cluster* dengan *Region*, seperti terlihat pada Gambar 2.10.



Gambar 2.10 Graf Terkelompokkan sesuai *Region*

2.4 Pembentukan Tabel *Routing*

Pada jaringan komputer, untuk menyampaikan data dari satu komputer ke komputer lainnya dibutuhkan sebuah lintasan terpendek dari komputer sumber ke komputer tujuan. Untuk menyampaikan data tersebut dibutuhkan sebuah *router* yang berisi tabel *routing*. Tabel *routing* sendiri adalah tabel yang terdapat di *router*, yang terdiri dari dua kolom. Kolom pertama berisi alamat-alamat individu *Internet* atau *range-range* alamat yang dinyatakan dalam urutan bilangan. Kolom kedua, berisi *hop* berikutnya yang cocok dari lintasan ke alamat yang berkorespondensi (Clark, 2003). *Hop* adalah transmisi satu titik ke titik lainnya dalam sebuah rangkaian yang diperlukan untuk mendapatkan sebuah pesan dari titik A ke titik B pada sebuah jaringan (Howe, 1997, p.1).

Contoh 2.5

Untuk graf G_1 pada Gambar 2.3 yang belum dikelompokkan ke dalam *cluster* akan dibentuk tabel *routing* yang berisi semua kemungkinan simpul tujuan dari simpul sumber a dan hop berikutnya pada lintasan terpendek yang diperoleh untuk mencapai simpul tujuan.

Diperoleh tabel *routing* sebagai berikut.

Tabel 2.1 Tabel Penuh untuk a

Tujuan	Hop Berikutnya
a	-
b	c
c	c
d	c
e	c
z	c

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 3, lintasan a-c-b, hop berikutnya c
3. Dari simpul sumber a menuju c, jarak terpendek 2, lintasan a-c, hop berikutnya c
4. Dari simpul sumber a menuju d, jarak terpendek 8, lintasan a-c-b-d, hop berikutnya c
5. Dari simpul sumber a menuju e, jarak terpendek 10, lintasan a-c-b-d-e, hop berikutnya c
6. Dari simpul sumber a menuju z, jarak terpendek 13, lintasan a-c-b-d-e-z, hop berikutnya c

Untuk graf G_1 yang sudah dikelompokkan pada Gambar 2.10, diperoleh tabel *routing* sebagai berikut.

Tabel 2.2 Tabel Ter-*cluster* untuk a

Tujuan	Hop Berikutnya
a	-
b	c
c	c
Region 2	c
Region 3	c

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 3, lintasan a-c-b, hop berikutnya c
3. Dari simpul sumber a menuju c, jarak terpendek 2, lintasan a-c, hop berikutnya c
4. Dari simpul sumber a menuju *Region* 2, jarak terpendek 8, lintasan a-c-b-d, hop berikutnya c
5. Dari simpul sumber a menuju *Region* 3, jarak terpendek 13, lintasan a-c-b-d-e-z, hop berikutnya c

BAB 3
ALGORITMA *LINK STATE ROUTING*

Pada skripsi ini dibahas dua algoritma untuk membentuk tabel *routing*. Algoritma pertama dinamakan algoritma *link state routing* dan algoritma kedua dinamakan algoritma *link state routing* dengan *clustering*.

3.1 Algoritma *Link State Routing*

Algoritma *link state routing* adalah algoritma yang bertujuan untuk mengisi tabel *routing* dengan rute terbaik dan terkini. Algoritma tersebut digunakan sampai saat ini karena konvergensinya yang cepat dibandingkan algoritma lain, tidak rentan terhadap *routing loops*, tidak rentan terhadap informasi yang salah karena informasi pada pihak pertama saja yang disiarkan.

Ide dari *link state routing* sangat sederhana dan dapat dinyatakan sebagai lima bagian. Menurut Tanenbaum, tahap pada algoritma *link state routing* sebagai berikut :

1. Temukan tetangga dan pelajari alamat jaringan mereka
2. Ukur *delay* atau biaya ke masing-masing tetangganya
3. Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya
4. Kirim paket ini ke semua *router* lain
5. Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

(Tanenbaum, 2003).

Berikut algoritma *link state routing* untuk membuat tabel *routing*.

Algoritma *Link State Routing*

MASUKAN : Sembarang graf terboboti
KELUARAN : Tabel *routing* dengan rute minimal
PROSES :
1. Temukan tetangga dan pelajari alamat jaringan mereka
2. Ukur *delay* atau biaya ke masing-masing tetangganya
3. Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya
4. Kirim paket ini ke semua *router* lain
5. Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

Contoh penggunaan algoritma *link state routing* :

Untuk memperoleh tabel *routing* yang akan disimpan di *router* dapat digunakan algoritma *link state routing* dari model jaringan graf G_2 sebagai berikut :

1) Temukan tetangga dan pelajari alamat jaringan mereka

   Router a mengenali tetangganya, yaitu *router* b dan *router* h. *Router* b mengenali tetangganya, yaitu *router* a, *router* c, dan *router* h. *Router* c dan seterusnya mengenali tetangganya masing-masing.

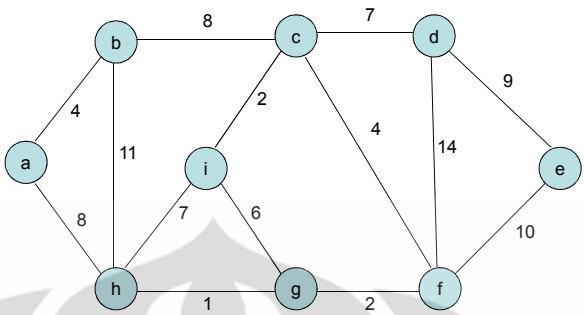
2) Ukur *delay* atau biaya ke masing-masing tetangganya

   Masing-masing *router* mengukur *delay* atau biaya ke masing-masing tetangganya.

3) Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya

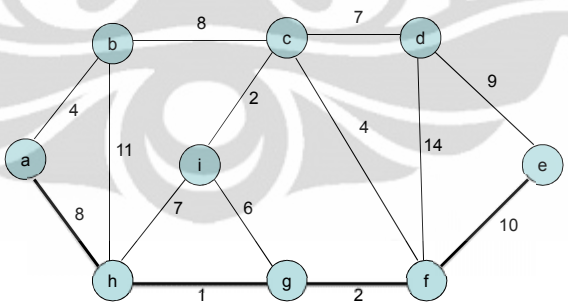
4) Kirim paket ini ke semua *router* lain

Setelah tahapan 1 sampai 4 dijalankan diperoleh graf yang menggambarkan *router*. Misalkan *router* dapat digambarkan dalam bentuk graf seperti berikut.



Gambar 3.1 Graf Terboboti G_2 dengan Sembilan Simpul

5) Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra



Gambar 3.2 Lintasan Terpendek dari Graf G_2

Diperoleh jarak terpendek dari a ke e sebesar 21 dengan lintasan terpendek a-h-g-f-e.

Untuk memperoleh tabel *routing* dihitung jarak terpendek dari sumber ke semua kemungkinan tujuan.

Setelah tahapan kelima dijalankan diperoleh tabel *routing* sebagai berikut.

Tabel 3.1 Tabel Penuh untuk a

Tujuan	Hop Berikutnya
a	-
b	b
c	b
d	b
e	h
f	h
g	h
h	h
i	b

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 4, lintasan a-b, hop berikutnya b
3. Dari simpul sumber a menuju c, jarak terpendek 12, lintasan a-b-c, hop berikutnya b
4. Dari simpul sumber a menuju d, jarak terpendek 19, lintasan a-b-c-d, hop berikutnya b
5. Dari simpul sumber a menuju e, jarak terpendek 21, lintasan a-h-g-f-e, hop berikutnya h
6. Dari simpul sumber a menuju f, jarak terpendek 11, lintasan a-h-g-f, hop berikutnya h

7. Dari simpul sumber a menuju g, jarak terpendek 9, lintasan a-h-g, hop berikutnya h
8. Dari simpul sumber a menuju h, jarak terpendek 8, lintasan a-h, hop berikutnya h
9. Dari simpul sumber a menuju i, jarak terpendek 14, lintasan a-b-c-i, hop berikutnya b

3.2 Modifikasi Algoritma *Link State Routing*

Walaupun algoritma *link state routing* mempunyai beberapa keuntungan dibandingkan algoritma lain, masih terdapat beberapa masalah dalam penggunaannya sehingga perlu dilakukan modifikasi terhadap algoritma *link state routing*. Masalah-masalah tersebut antara lain :

1. Membutuhkan sumber daya *router* yang tinggi baik *power* maupun memori.
2. Menghasilkan *traffic* yang sangat tinggi saat pertama kali membanjiri jaringan (*flooding*). Akan tetapi, jika konfigurasi inisialisasi ini sudah stabil maka *traffic* dari *link state* ini sangat kecil.
3. Memungkinkan *delay* atau bahkan *lost* menyebabkan jaringan inkonsisten. Hal ini umumnya menjadi masalah pada jaringan yang besar jika bagian-bagian jaringan datang *online* pada saat yang berbeda atau jika *link bandwidth* antar *link* berbeda (misal pada jaringan ISP yang lebar akan berbeda dengan jaringan lainnya). Masalah inilah yang biasanya merupakan masalah terbesar.

Untuk mengatasi masalah yang ada pada algoritma *link state routing*, topologi jaringan yang besar akan dibagi-bagi menjadi kelompok-kelompok jaringan yang lebih kecil. Modifikasi tersebut diharapkan dapat memecahkan masalah yang ada pada pembuatan tabel *routing* dan dapat mengurangi ukuran dari tabel *routing*.

Modifikasi algoritma *link state routing* adalah algoritma *link state routing* dengan *clustering* yang pada tahapan terakhir dari *link state routing* akan digunakan algoritma Kruskal untuk membentuk pohon perentangan minimum (*minimum spanning tree*) pada sembarang jaringan komputer yang telah digambarkan dalam bentuk graf. Setelah pohon perentangan minimum terbentuk, rata-rata dari bobot

dalam graf dihitung. Kemudian, busur yang mempunyai bobot lebih besar dari rata-rata dihapus sehingga akan terbentuk kelompok atau *cluster* yang diberi nama *region*. Selanjutnya, dibuat tabel *routing* untuk jaringan komputer yang diteliti.

Berikut algoritma *link state routing* dengan *clustering* untuk membuat dan mengurangi ukuran tabel *routing*.

Algoritma *Link State Routing* dengan *Clustering*

MASUKAN : Sembarang graf terboboti
KELUARAN : Tabel routing dengan rute minimal
PROSES :

1. Temukan tetangga dan pelajari alamat jaringan mereka
2. Ukur *delay* atau biaya ke masing-masing dari tetangganya
3. Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya
4. Kirim paket ini ke semua *router* lain
5. Gunakan algoritma Kruskal untuk membentuk pohon perentangan minimum (*minimum spanning tree*)
6. Hitung rata-rata dari seluruh bobot pada pohon perentangan minimum
7. Hapus busur yang mempunyai bobot lebih besar daripada rata-rata
8. Terbentuk *cluster*
9. Beri nama *cluster* yang sudah terbentuk dengan nama *Region*
10. Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

Contoh penggunaan algoritma *link state routing* dengan *clustering* :

Akan dijalankan algoritma *link state routing* dengan *clustering* sebagai berikut :

1) Temukan tetangga dan pelajari alamat jaringan mereka

   Router a mengenali tetangganya, yaitu *router* b dan *router* h. *Router* b mengenali tetangganya, yaitu *router* a, *router* c, dan *router* h. *Router* c dan seterusnya mengenali tetangganya masing-masing.

2) Ukur *delay* atau biaya ke masing-masing tetangganya

Masing-masing *router* mengukur *delay* atau biaya ke masing-masing tetangganya.

3) Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya

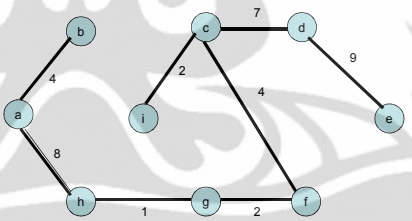
4) Kirim paket ini ke semua *router* lain

Setelah tahapan 1 sampai 4 dijalankan diperoleh graf yang dapat menggambarkan *router*.

Misalkan *router* dapat digambarkan dalam bentuk graf G_2 seperti pada Gambar 3.1.

5) Gunakan algoritma Kruskal untuk membentuk pohon perentangan minimum (*minimum spanning tree*)

Diperoleh pohon perentangan minimum (*minimum spanning tree*) berikut.

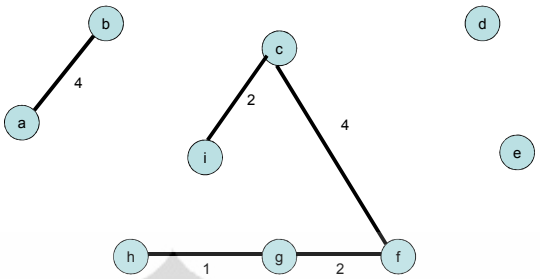


Gambar 3.3 Pohon Perentangan Minimum

6) Hitung rata-rata dari seluruh bobot pada pohon perentangan minimum

$$\text{Rata-rata} = \frac{4+8+1+2+2+4+7+9}{8} = \frac{37}{8} = 4,625$$

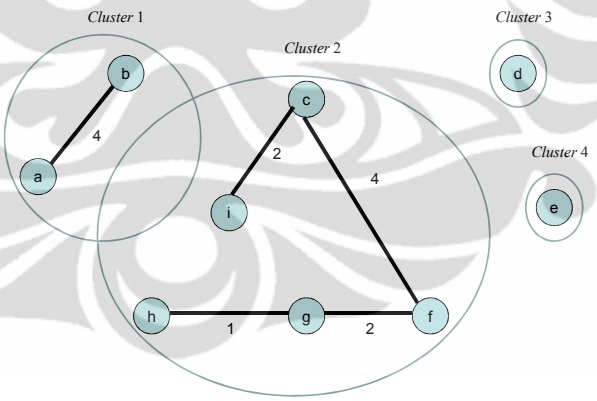
7) Hapus busur yang mempunyai bobot lebih besar daripada rata-rata diperoleh hasil MST *clustering*



Gambar 3.4 Beberapa *Cluster* yang Dihasilkan dari MST *Clustering* terhadap Graf G_2

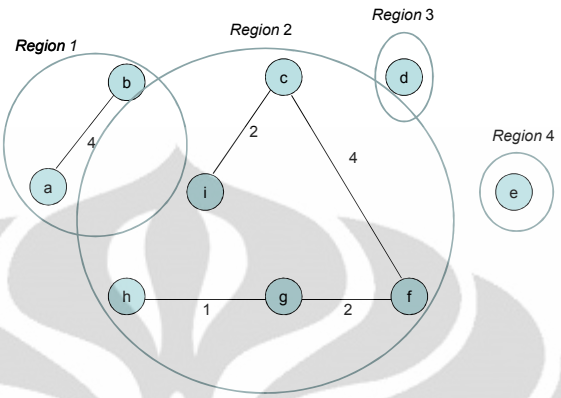
8) Terbentuk *cluster*

Terbentuk empat *cluster*, yaitu {a,b}, {c,f,g,h,i}, {d}, dan {e}.



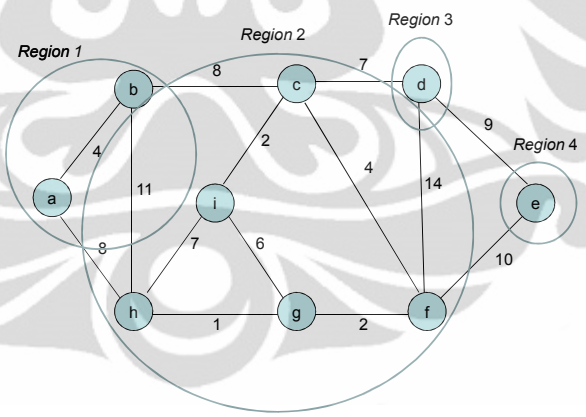
Gambar 3.5 Graf Terkelompokkan

9) Beri nama *cluster* yang sudah terbentuk dengan nama *Region*



Gambar 3.6 Graf dengan nama *Region*nya

Graf yang sudah terkelompokkan terlihat pada Gambar 3.7.



Gambar 3.7 Graf Terkelompokkan sesuai *Region*

10) Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

Setelah tahapan 1 sampai 10 dijalankan diperoleh tabel *routing* sebagai berikut.

Tabel 3.2 Tabel Ter-*cluster* untuk a

Tujuan	Hop Berikutnya
a	-
b	b
Region 2	h
Region 3	b
Region 4	h

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 4, lintasan a-b, hop berikutnya b
3. Dari simpul sumber a menuju *Region* 2, jarak terpendek 8, lintasan a-h, hop berikutnya h
4. Dari simpul sumber a menuju *Region* 3, jarak terpendek 19, lintasan a-b-c-d, hop berikutnya b
5. Dari simpul sumber a menuju *Region* 4, jarak terpendek 21, lintasan a-h-g-f-e, hop berikutnya h

BAB 4
IMPLEMENTASI DAN SIMULASI

Pada bab ini, akan dibahas implementasi dan simulasi algoritma *link state routing* dan modifikasi algoritma *link state routing*, yaitu algoritma *link state routing* dengan *clustering*. Implementasi dan simulasi pada sebuah contoh graf dan data *router* di FMIPA UI yang digambarkan dalam bentuk graf. Simulasi dilakukan dengan menggunakan perangkat lunak Matlab 7.8.0 (R2009a) untuk menampilkan jaringan komputer dalam bentuk graf dan hasil berupa tabel *routing*.

4.1 Implementasi dan Simulasi Algoritma *Link State Routing* pada Jaringan Komputer

Pembentukan tabel *routing* menggunakan prosedur berikut.

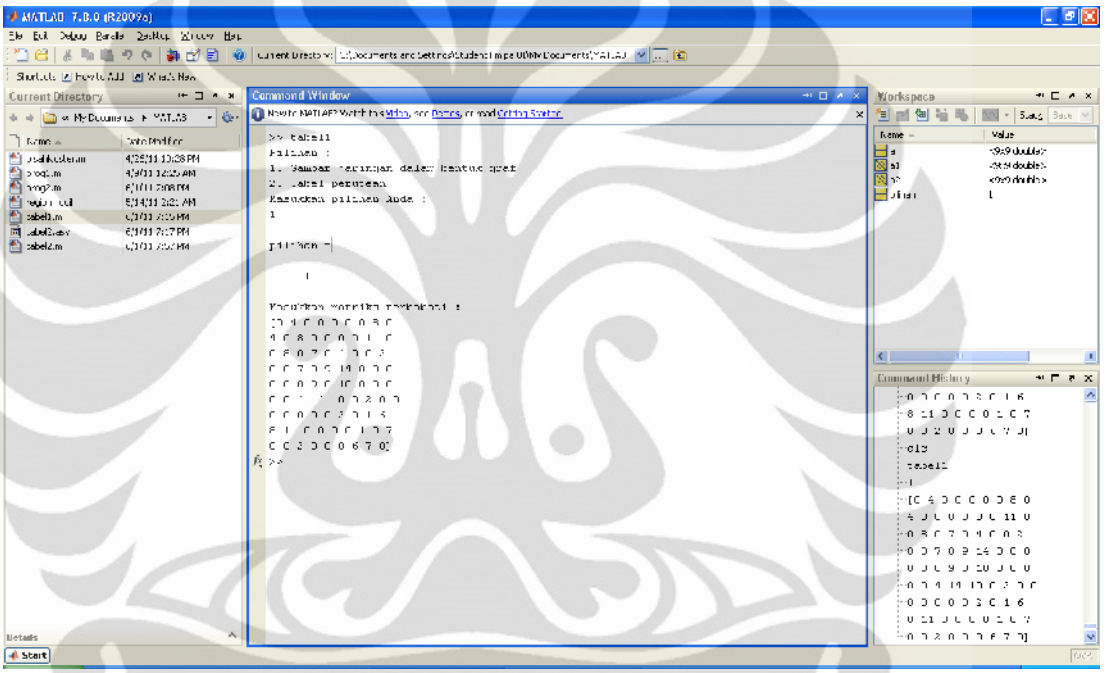
Algoritma *Link State Routing*

MASUKAN : Sembarang graf terboboti
KELUARAN : Tabel *routing* dengan rute minimal
PROSES :

1. Temukan tetangga dan pelajari alamat jaringan mereka
2. Ukur *delay* atau biaya ke masing-masing tetangganya
3. Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya
4. Kirim paket ini ke semua *router* lain
5. Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

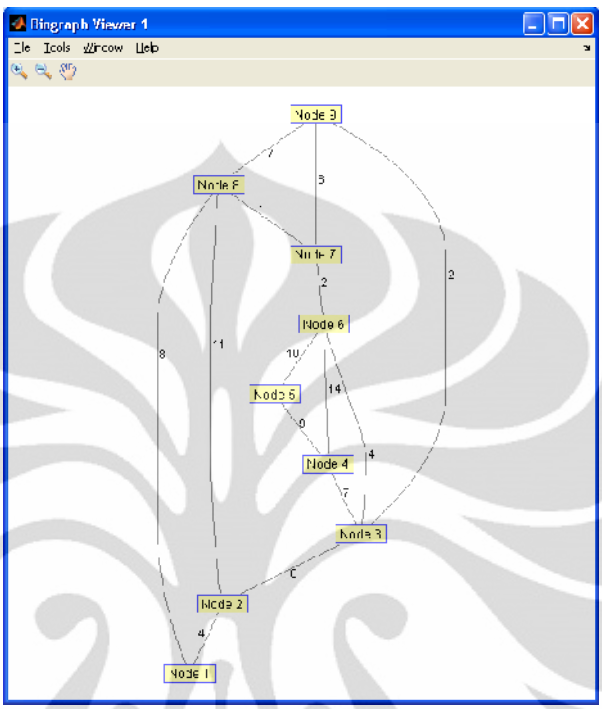
Setelah tahap 1 sampai 4 dijalankan, diperoleh peta jaringan komputer yang dapat digambarkan dalam graf. Selanjutnya, pada tahap kelima digunakan algoritma Dijkstra untuk membentuk tabel *routing*. Pembentukan tabel *routing* dapat menggunakan program tabel1 yang terdapat pada Lampiran 1.

Akan dilakukan simulasi terhadap contoh graf G_2 pada Gambar 2.5 dengan menjalankan program tabel1 pada *Command Window*. Dimisalkan simpul a dengan simpul 1, simpul b dengan simpul 2, simpul c dengan simpul 3, simpul d dengan simpul 4, simpul e dengan simpul 5, simpul f dengan simpul 6, simpul g dengan simpul 7, simpul h dengan simpul 8, dan simpul i dengan simpul 9. Dipilih pilihan 1, yaitu 1. Gambar jaringan dalam bentuk graf. Diperoleh



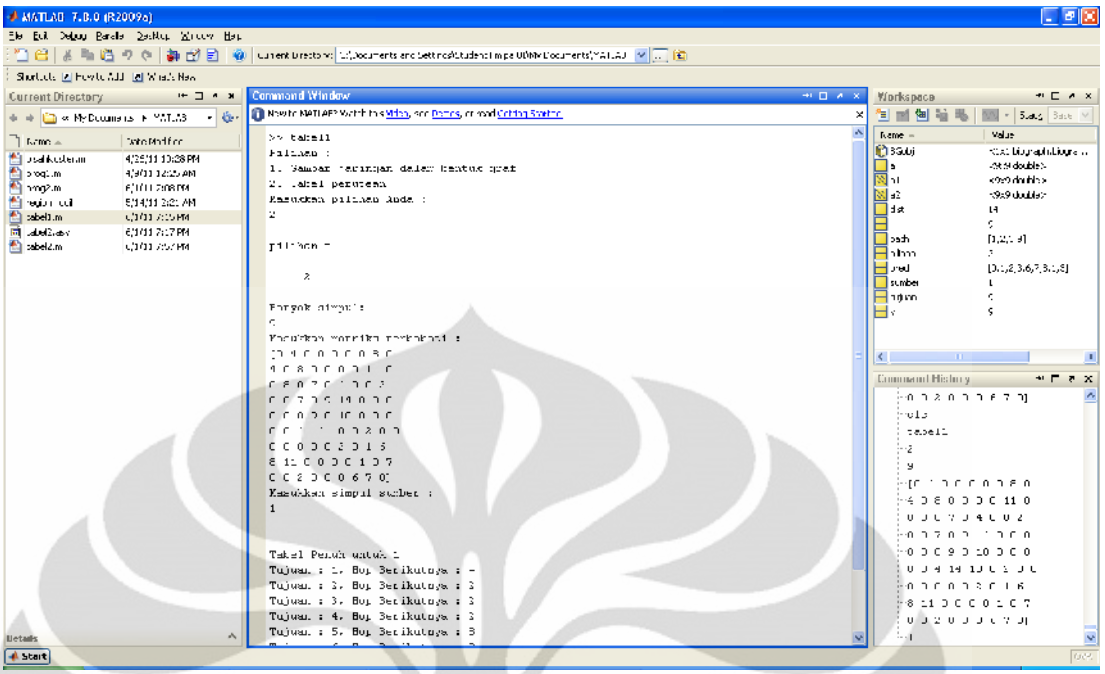
Gambar 4.1 Gambar *Command Window* Hasil Pilihan 1 pada Graf Terboboti Sembilan Simpul

Diperoleh graf sebagai berikut.

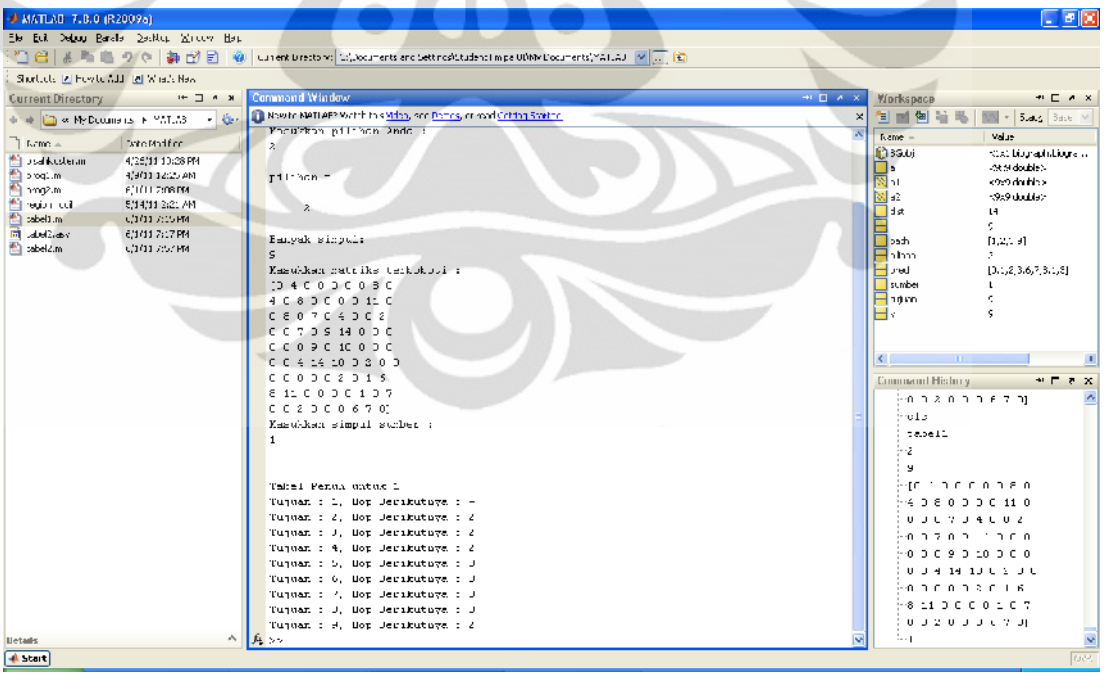


Gambar 4.2 Graf Hasil Pilihan 1

Selanjutnya, program dijalankan kembali dengan mengetik tabel1 pada *Command Window* dan dipilih pilihan 2, yaitu 2. Tabel perutean. Diperoleh



Gambar 4.3 Gambar *Command Window* Hasil Pilihan 2 pada Graf Terboboti Sembilan Simpul



Gambar 4.4 Kelanjutan Gambar *Command Window* Hasil Pilihan 2 pada Graf Terboboti Sembilan Simpul

Diperoleh tabel *routing* untuk simpul sumber 1, dinamakan Tabel Penuh untuk 1, dalam hal ini di graf G_2 pada Gambar 2.5 untuk simpul sumber a, dinamakan Tabel Penuh untuk a.

Tabel 4.1 Tabel Penuh untuk a

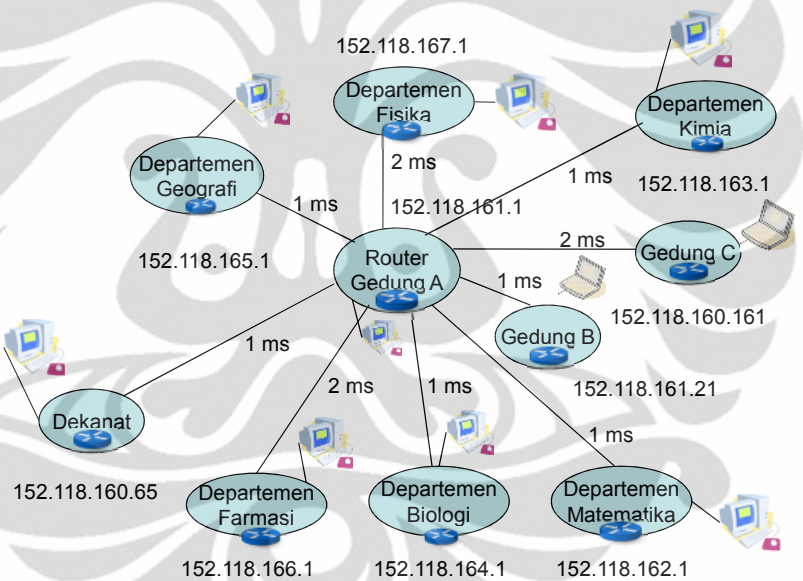
Tujuan	Hop Berikutnya
a	-
b	b
c	b
d	b
e	h
f	h
g	h
h	h
i	b

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 4, lintasan a-b, hop berikutnya b
3. Dari simpul sumber a menuju c, jarak terpendek 12, lintasan a-b-c, hop berikutnya b
4. Dari simpul sumber a menuju d, jarak terpendek 19, lintasan a-b-c-d, hop berikutnya b
5. Dari simpul sumber a menuju e, jarak terpendek 21, lintasan a-h-g-f-e, hop berikutnya h
6. Dari simpul sumber a menuju f, jarak terpendek 11, lintasan a-h-g-f, hop berikutnya h

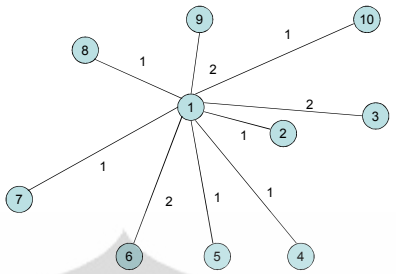
7. Dari simpul sumber a menuju g, jarak terpendek 9, lintasan a-h-g, hop berikutnya h
8. Dari simpul sumber a menuju h, jarak terpendek 8, lintasan a-h, hop berikutnya h
9. Dari simpul sumber a menuju i, jarak terpendek 14, lintasan a-b-c-i, hop berikutnya b

Selanjutnya, simulasi program untuk peta jaringan komputer di FMIPA UI. Berikut ini peta jaringan komputer di FMIPA UI.



Gambar 4.5 Waktu *Response* (*Response Time*) pada Jaringan Komputer di FMIPA UI

Peta jaringan komputer tersebut dapat digambarkan dalam bentuk graf sebagai berikut.



Gambar 4.6 G_3 Graf Data *Router* di FMIPA UI

Keterangan :

Misalkan

1 mewakili *router* di Gedung A yang beralamat IP 152.118.161.1

2 mewakili *router* di Gedung B yang beralamat IP 152.118.161.21

3 mewakili *router* di Gedung C yang beralamat IP 152.118.160.161

4 mewakili *router* di Departemen Matematika yang beralamat IP 152.118.162.1

5 mewakili *router* di Departemen Biologi yang beralamat IP 152.118.164.1

6 mewakili *router* di Departemen Farmasi yang beralamat IP 152.118.166.1

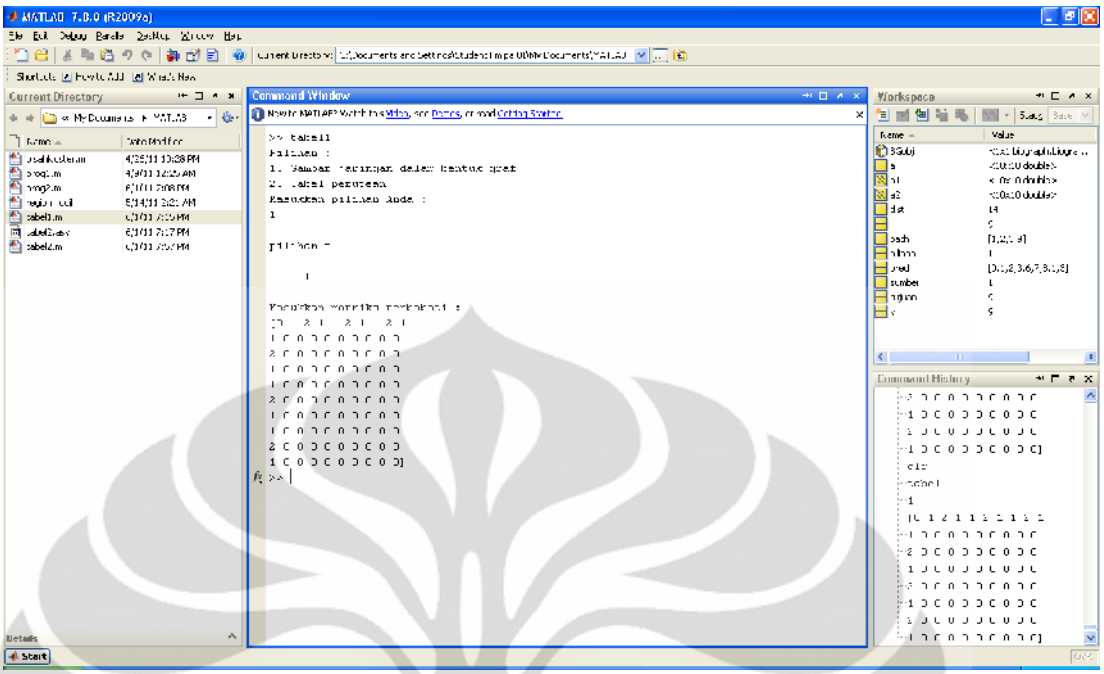
7 mewakili *router* di Dekanat yang beralamat IP 152.118.160.65

8 mewakili *router* di Departemen Geografi yang beralamat IP 152.118.165.1

9 mewakili *router* di Departemen Fisika yang beralamat IP 152.118.167.1

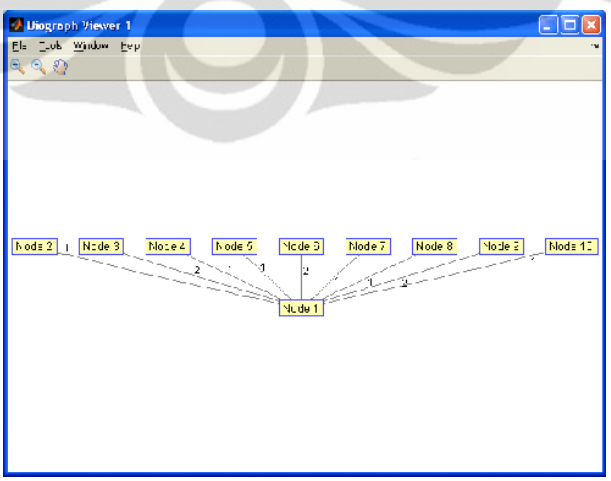
10 mewakili *router* di Departemen Kimia yang beralamat IP 152.118.163.1

Untuk menjalankan program, diketik tabel1 pada *Command Window*. Dipilih pilihan 1, yaitu 1. Gambar jaringan dalam bentuk graf. Diperoleh



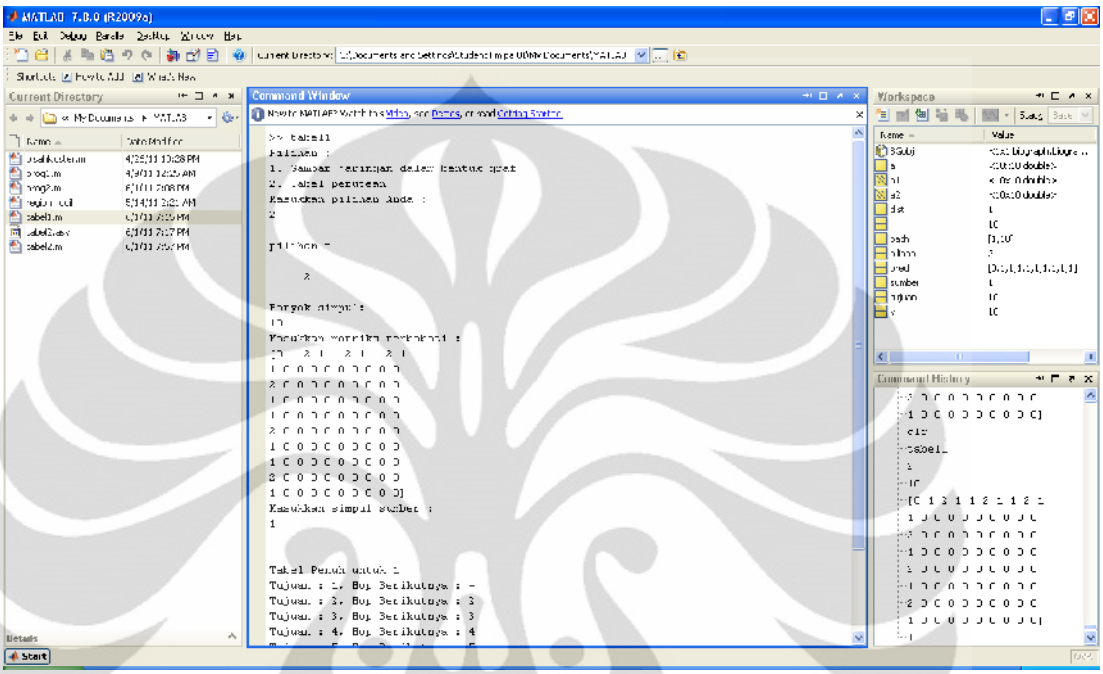
Gambar 4.7 Gambar *Command Window* Hasil Pilihan 1 pada Graf Data *Router* di FMIPA UI

Diperoleh graf sebagai berikut.

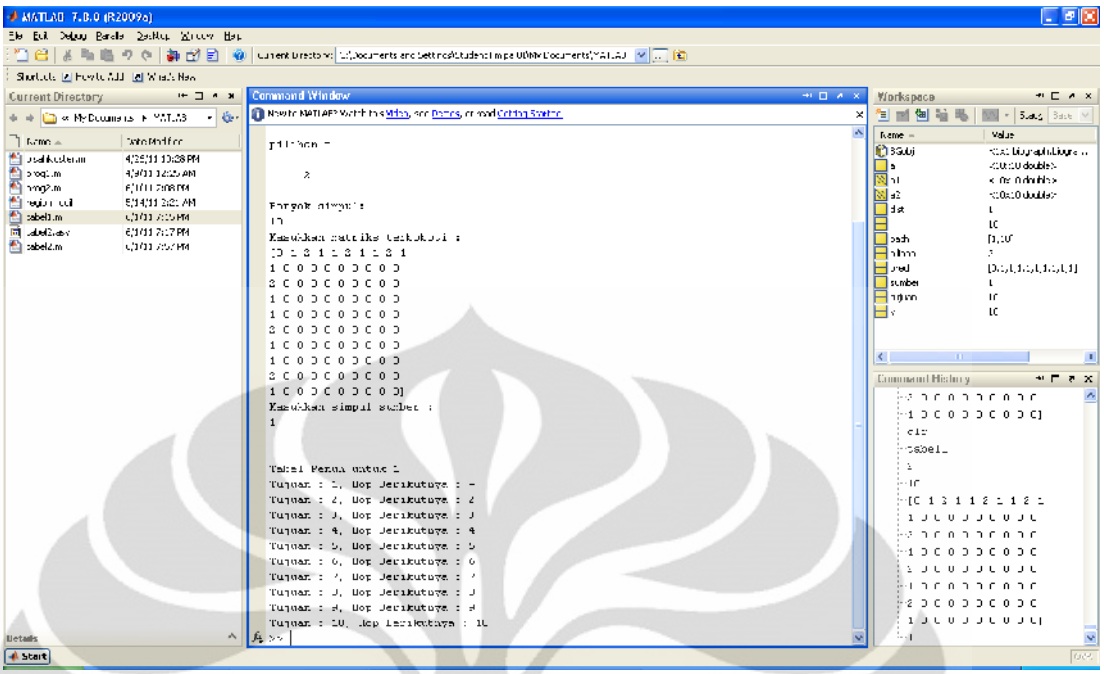


Gambar 4.8 Graf Hasil Pilihan 1

Selanjutnya, program dijalankan kembali dengan mengetik tabel1 pada *Command Window* dan dipilih pilihan 2, yaitu 2. Tabel perutean. Diperoleh



Gambar 4.9 Gambar *Command Window* Hasil Pilihan 2 pada Graf Data *Router* di FMIPA UI



Gambar 4.10 Kelanjutan Gambar *Command Window* Hasil Pilihan 2 pada Graf Data *Router* di FMIPA UI

Dari hasil program diperoleh tabel *routing* sebagai berikut :

Tabel 4.2 Tabel Penuh untuk 1

Tujuan	Hop Berikutnya
1	-
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Keterangan :

1. Dari simpul sumber 1 menuju 1, jarak terpendek 0, lintasan 1, hop berikutnya -
2. Dari simpul sumber 1 menuju 2, jarak terpendek 1, lintasan 1-2, hop berikutnya 2
3. Dari simpul sumber 1 menuju 3, jarak terpendek 2, lintasan 1-3, hop berikutnya 3
4. Dari simpul sumber 1 menuju 4, jarak terpendek 1, lintasan 1-4, hop berikutnya 4
5. Dari simpul sumber 1 menuju 5, jarak terpendek 1, lintasan 1-5, hop berikutnya 5
6. Dari simpul sumber 1 menuju 6, jarak terpendek 2, lintasan 1-6, hop berikutnya 6

7. Dari simpul sumber 1 menuju 7, jarak terpendek 1, lintasan 1-7, hop berikutnya 7
8. Dari simpul sumber 1 menuju 8, jarak terpendek 1, lintasan 1-8, hop berikutnya 8
9. Dari simpul sumber 1 menuju 9, jarak terpendek 2, lintasan 1-9, hop berikutnya 9
10. Dari simpul sumber 1 menuju 10, jarak terpendek 1, lintasan 1-10, hop berikutnya 10

4.2 Implementasi dan Simulasi Modifikasi Algoritma *Link State Routing* pada Jaringan Komputer

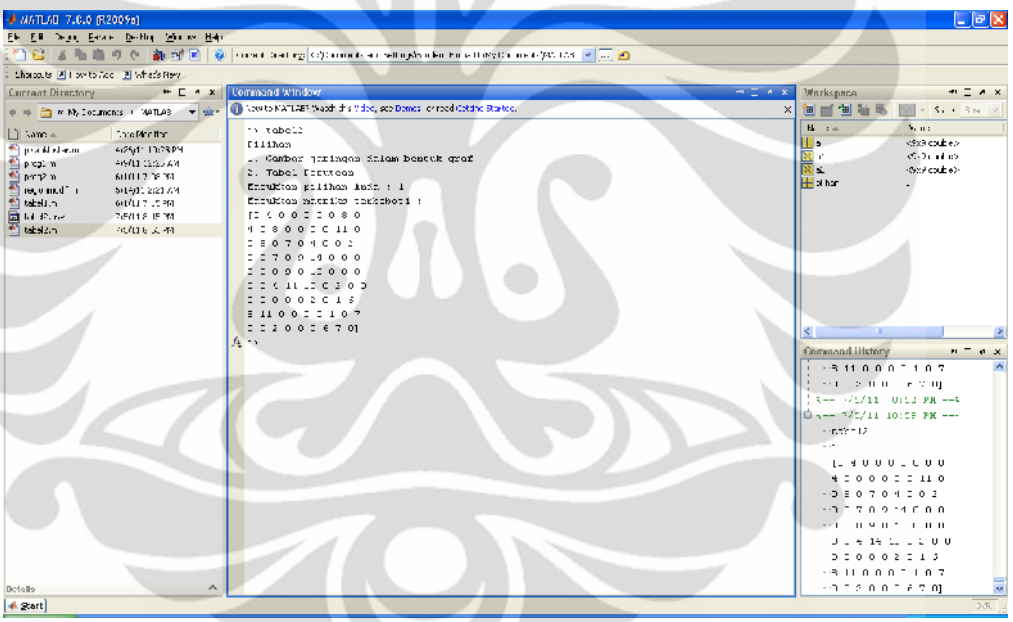
Pembentukan tabel *routing* menggunakan prosedur berikut.

Algoritma *Link State Routing* dengan *Clustering*

MASUKAN : Sembarang graf terboboti
KELUARAN : Tabel routing dengan rute minimal
PROSES :
1. Temukan tetangga dan pelajari alamat jaringan mereka
2. Ukur *delay* atau biaya ke masing-masing dari tetangganya
3. Bangun sebuah paket yang mengatakan semua harus pelajari alamat jaringan tetangganya
4. Kirim paket ini ke semua *router* lain
5. Gunakan algoritma Kruskal untuk membentuk pohon perentangan minimum (*Minimum Spanning Tree*)
6. Hitung rata-rata dari seluruh bobot pada pohon perentangan minimum
7. Hapus busur yang mempunyai bobot lebih besar daripada rata-rata
8. Terbentuk *cluster*
9. Beri nama *cluster* yang sudah terbentuk dengan nama *Region*
10. Hitung lintasan terpendek ke setiap *router* lain dengan menggunakan algoritma Dijkstra

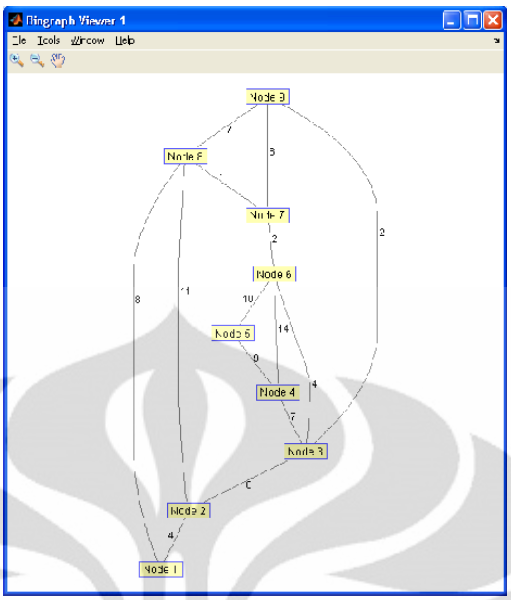
Setelah tahap 1 sampai 4 dijalankan, diperoleh peta jaringan komputer yang dapat digambarkan dalam bentuk graf. Tahap 5 sampai 8 dijalankan dengan menggunakan program tabel2 yang terdapat pada Lampiran 2.

Akan dilakukan simulasi terhadap contoh graf G_2 pada Gambar 2.5 dengan menjalankan program tabel2 pada *Command Window*. Dimisalkan simpul a dengan simpul 1, simpul b dengan simpul 2, simpul c dengan simpul 3, simpul d dengan simpul 4, simpul e dengan simpul 5, simpul f dengan simpul 6, simpul g dengan simpul 7, simpul h dengan simpul 8, dan simpul i dengan simpul 9. Dipilih pilihan 1, yaitu 1. Gambar jaringan dalam bentuk graf. Diperoleh



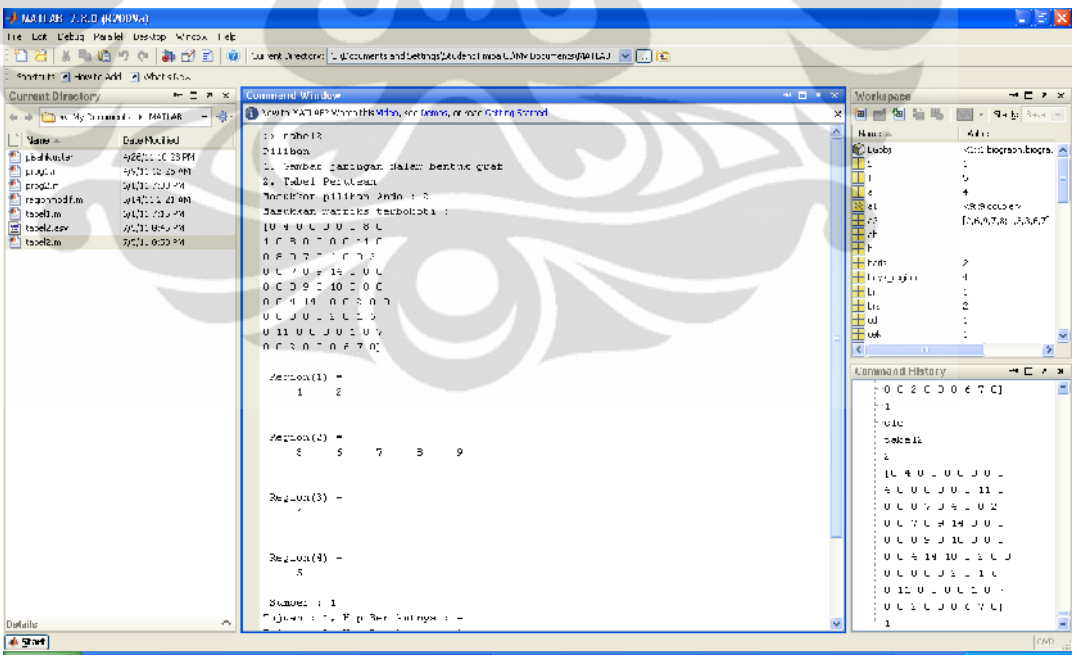
Gambar 4.11 Gambar *Command Window* Hasil Pilihan 1 pada Graf Terboboti Sembilan Simpul

Diperoleh graf sebagai berikut.

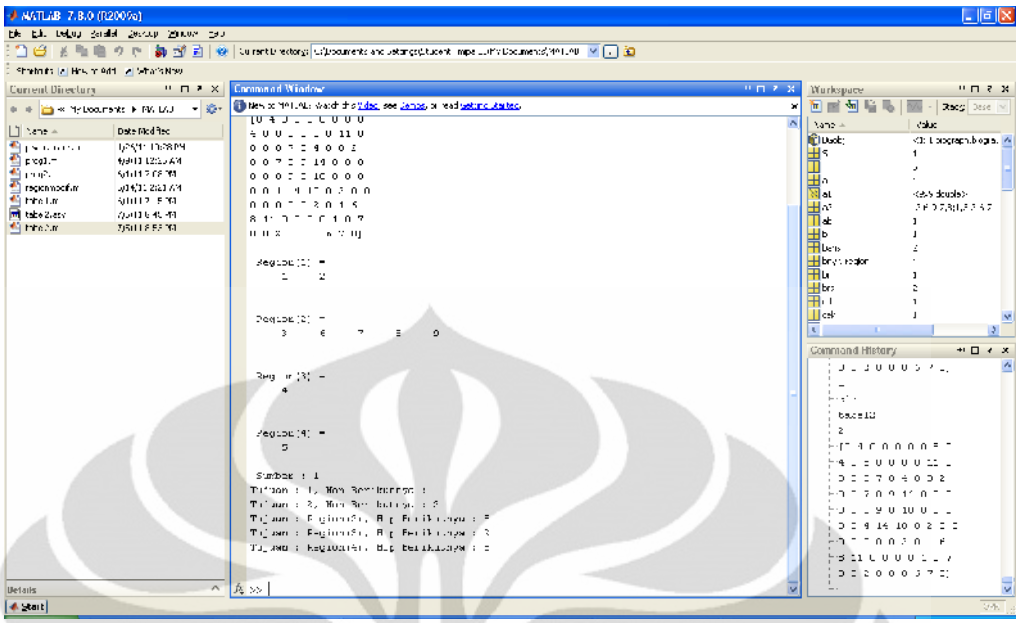


Gambar 4.12 Graf Hasil Pilihan 1

Selanjutnya, program dijalankan kembali dengan mengetik tabel2 pada *Command Window* dan dipilih pilihan 2, yaitu 2. Tabel perutean. Diperoleh

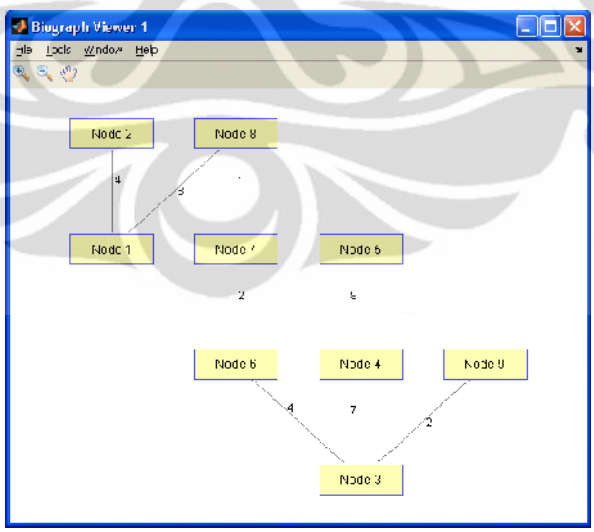


Gambar 4.13 Gambar *Command Window* Hasil Pilihan 2 pada Graf Terboboti Sembilan Simpul



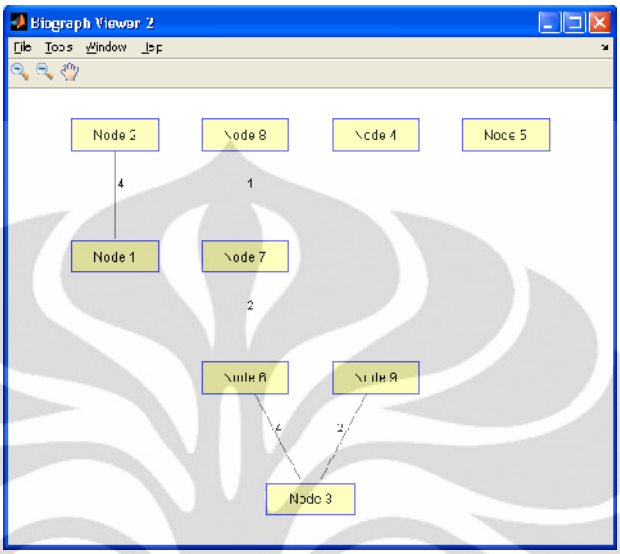
Gambar 4.14 Kelanjutan Gambar *Command Window* Hasil Pilihan 2 pada Graf Terboboti Sembilan Simpul

Diperoleh pohon perentangan minimum



Gambar 4.15 Pohon Perentangan Minimum

Dan hasil MST *clustering*



Gambar 4.16 Hasil MST *Clustering*

Diperoleh tabel *routing* untuk simpul sumber 1, dinamakan Tabel Ter-*cluster* untuk 1, dalam hal ini di graf G_2 pada Gambar 2.5 untuk simpul sumber a, dinamakan Tabel Ter-*cluster* untuk a.

Tabel 4.3 Tabel Ter-*cluster* untuk a

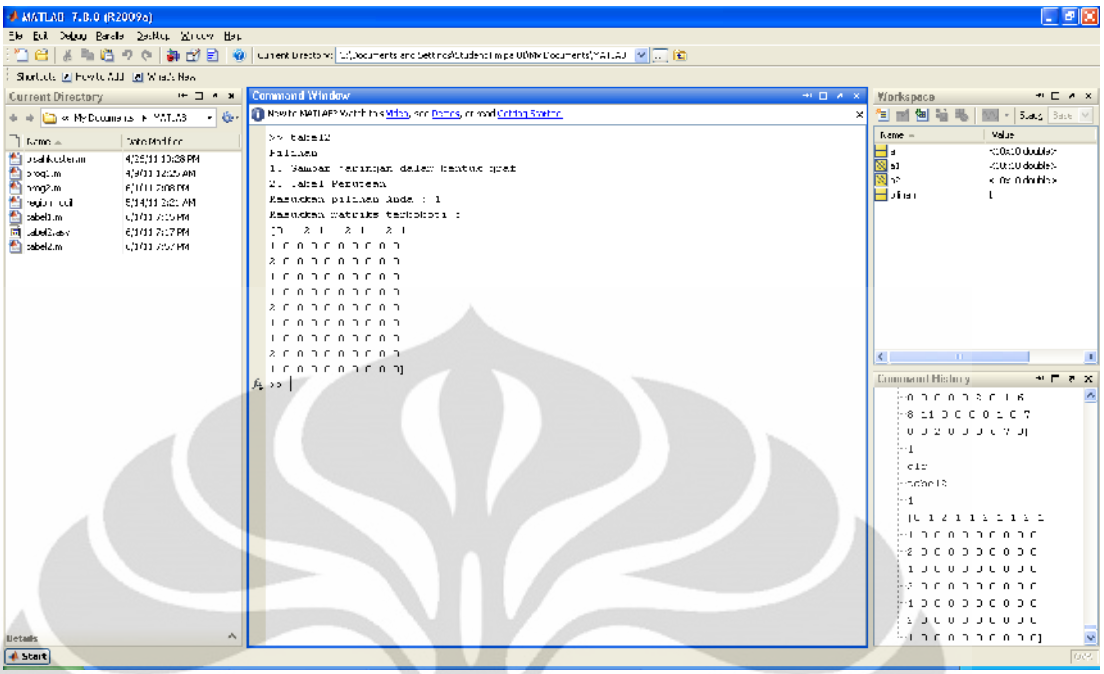
Tujuan	Hop Berikutnya
a	-
b	b
Region 2	h
Region 3	b
Region 4	h

Keterangan :

1. Dari simpul sumber a menuju a, jarak terpendek 0, lintasan a, hop berikutnya -
2. Dari simpul sumber a menuju b, jarak terpendek 4, lintasan a-b, hop berikutnya b
3. Dari simpul sumber a menuju *Region* 2, jarak terpendek 8, lintasan a-h, hop berikutnya h
4. Dari simpul sumber a menuju *Region* 3, jarak terpendek 19, lintasan a-b-c-d, hop berikutnya b
5. Dari simpul sumber a menuju *Region* 4, jarak terpendek 21, lintasan a-h-g-f-e, hop berikutnya h

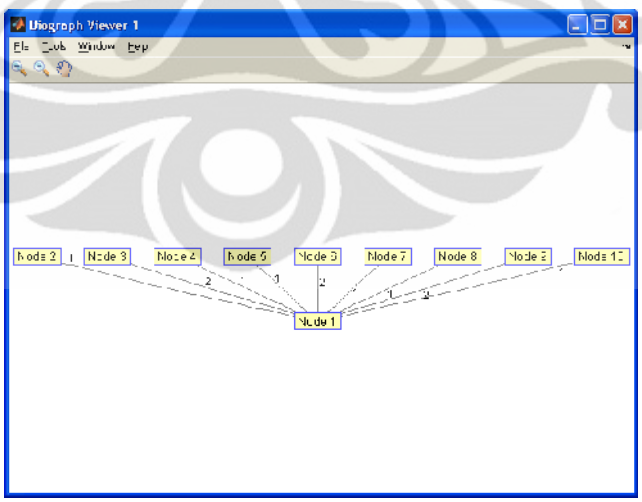
Selanjutnya, simulasi program untuk peta jaringan komputer di FMIPA UI.

Untuk menjalankan program, diketik tabel2 pada *Command Window*. Dipilih pilihan 1, yaitu 1. Gambar jaringan dalam bentuk graf. Diperoleh



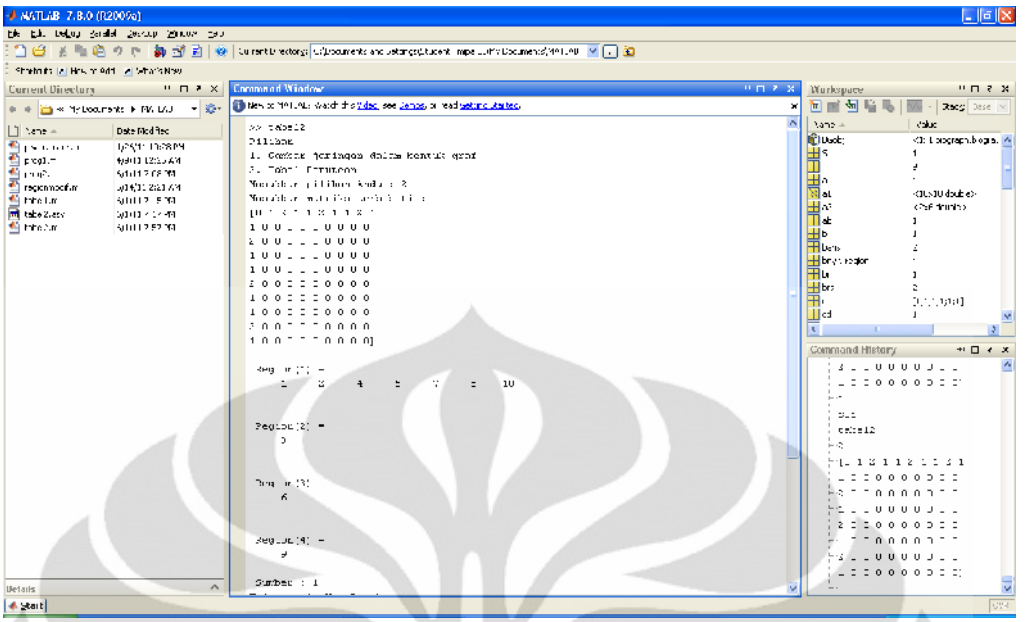
Gambar 4.17 Gambar *Command Window* Hasil Pilihan 1 pada Graf Data *Router* di FMIPA UI

Diperoleh graf sebagai berikut.

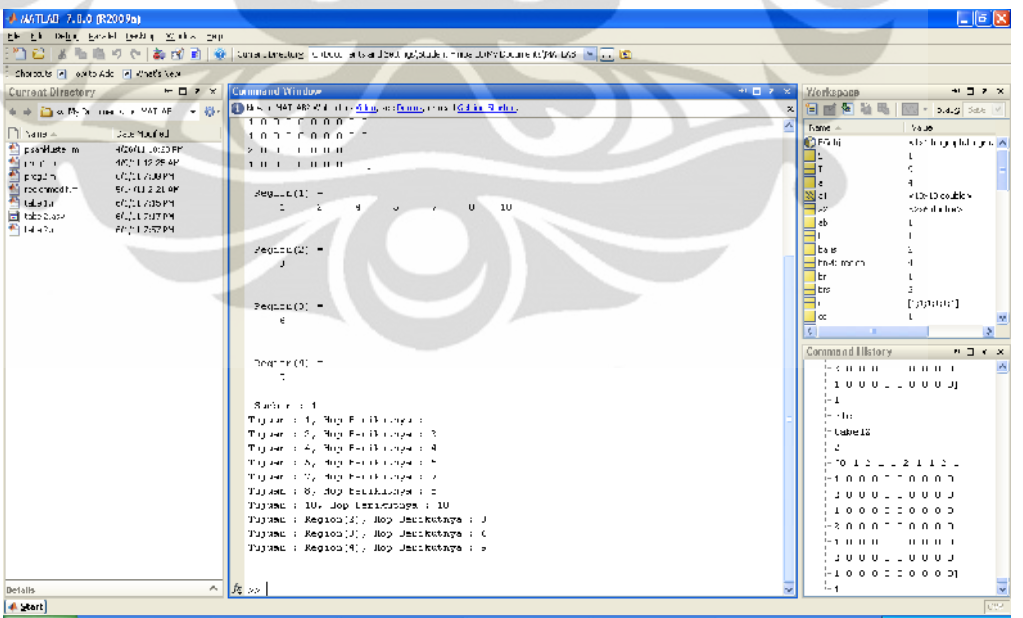


Gambar 4.18 Graf Hasil Pilihan 1

Selanjutnya, program dijalankan kembali dengan mengetik tabel2 pada *Command Window* dan dipilih pilihan 2, yaitu 2. Tabel perutean. Diperoleh

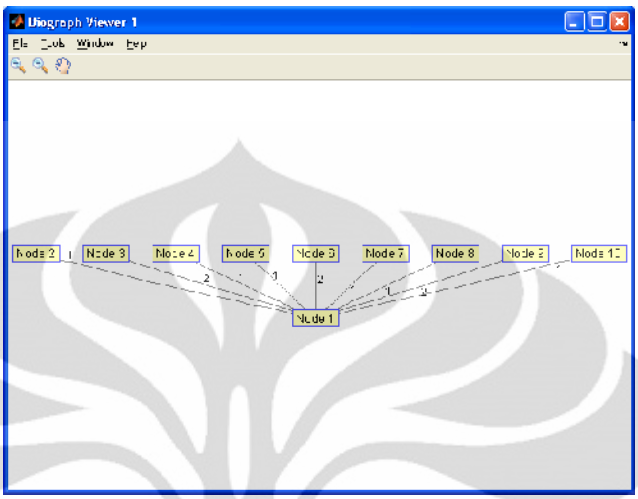


Gambar 4.19 Gambar *Command Window* Hasil Pilihan 2 pada Graf Data *Router* di FMIPA UI



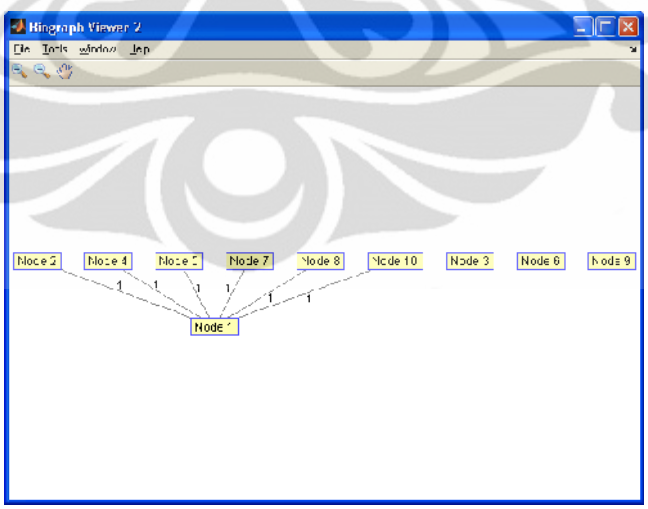
Gambar 4.20 Kelanjutan Gambar *Command Window* Hasil Pilihan 2 pada Graf Data *Router* di FMIPA UI

Diperoleh pohon perentangan minimum



Gambar 4.21 Pohon Perentangan Minimum

Dan hasil MST *clustering*



Gambar 4.22 Hasil MST *Clustering*

Dari hasil program diperoleh tabel *routing* sebagai berikut :

Tabel 4.4 Tabel Ter-*cluster* untuk 1

Tujuan	Hop Berikutnya
1	-
2	2
4	4
5	5
7	7
8	8
10	10
Region 2	3
Region 3	6
Region 4	9

Dengan menjalankan kedua algoritma, yaitu algoritma *link state routing* dan algoritma *link state routing* dengan *clustering* dapat diketahui bahwa ukuran tabel *routing* dengan menggunakan algoritma *link state routing* dengan *clustering* akan lebih kecil dari atau sama dengan ukuran tabel *routing* yang diperoleh dengan menggunakan algoritma *link state routing*.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Pada algoritma *link state routing* terdapat beberapa keuntungan sehingga digunakan sampai saat ini. Tetapi masih terdapat beberapa kerugian dan masalah antara lain, membutuhkan sumber daya yang tinggi dalam mendapatkan informasi topologi jaringan, menghasilkan *traffic* yang sangat tinggi pada saat pertama kali membanjiri jaringan, dan memungkinkan *delay* atau *lost* menyebabkan jaringan tidak konsisten. Pada algoritma *link state routing* digunakan algoritma Dijkstra untuk menemukan lintasan terpendek.
2. Untuk mengatasi masalah yang ada pada penggunaan algoritma *link state routing*, pada penelitian ini dilakukan modifikasi terhadap algoritma *link state routing*. Modifikasi algoritma *link state routing* dinamakan algoritma *link state routing* dengan *clustering*. Pada algoritma *link state routing* dengan *clustering*, jaringan komputer yang besar dibagi-bagi menjadi jaringan komputer yang lebih kecil. Hal tersebut dilakukan untuk mereduksi ukuran tabel *routing*. Pada algoritma *link state routing*, graf yang besar sebagai representasi jaringan komputer dibagi-bagi terlebih dahulu menjadi kelompok-kelompok graf yang lebih kecil dengan menggunakan algoritma Zahn untuk memperoleh *cluster* pada graf yang sudah berbentuk pohon perentangan minimum yang diperoleh dengan menggunakan algoritma Kruskal. Setelah graf terbagi menjadi kelompok-kelompok yang lebih kecil digunakan algoritma Dijkstra untuk menemukan lintasan terpendek.
3. Setelah implementasi dan simulasi, dapat diketahui bahwa ukuran tabel *routing* dari algoritma *link state routing* dengan *clustering* lebih kecil atau sama dengan ukuran tabel *routing* dari algoritma *link state routing*.

5.2 Saran

Algoritma *link state routing* dengan *clustering* sebaiknya digunakan pada jaringan yang menggunakan topologi selain topologi bintang.

DAFTAR PUSTAKA

Akib, Faisal. "Fiber Distributed Data Interface." Style Sheet.
http://teknik-informatika.com
diperoleh (Mei 2011)

Akib, Faisal. "Topologi Jaringan : Macam-macam Topologi Jaringan." Style Sheet.
http://teknik-informatika.com
diperoleh (2011)

Aminah, Siti. (2000). *Penggunaan Metoda Clustering dalam Pemetaan Kualitas Pendidikan SMTA di Indonesia berdasarkan Nilai UMPTN*.

Balakrishnan, V.K. (1997). *Schaum's Outline of Theory and Problems of Graph Theory*. McGraw-Hill Companies, Inc.

Byers, John, dkk. (2005). *Optimizing IP Address Assignment on Network Topologies*. Flux Technical Note.

Clark, Martin P. (2003). *Data Networks, IP, and the Internet : networks, protocols, design, and operation*. West Sussex : John Wiley and Sons Ltd.

Howe, Denis. "hop Definition." Style Sheet.
www.learnthat.com/define/view.asp?id=6194
diperoleh (12 Februari 2011)

Prasad, K. Rajendra, Rajulu, P. Govinda. (2010). *A Survey on Clustering Technique for Datasets using Efficient Graph Structures*. International Journal of Engineering Science and Technology.

Rosen, Kenneth H. (2007). *Discrete Mathematics and Its Applications, Sixth Edition*. New York : McGraw-Hill.

Shinde, S.S. (2009). *Computer Network*. New Delhi : New Age International.

Sugeng, Winarno. (2006). *Jaringan Komputer dengan TCP/IP*. Bandung : Informatika.

Tanenbaum, Andrew S. (2003). *Computer Networks*. New Jersey : Prentice Hall.

Thoyib, Suryadi M. (1997). *TCP/IP dan Internet : Sebagai Jaringan Komunikasi Global*. Jakarta : P.T. Elex Media Komputindo.

"World Internet Users and Population Stats". Style Sheet.
http://www.internetworldstats.com/stats.htm
diperoleh (17 Februari 2011)

http://www.ecgalerycomputer.co.cc/2010/08/link-state-routing.html
diperoleh (17 Juni 2011)

Kode Program Tabel *Routing* Algoritma *Link State Routing*

```
fprintf('Pilihan :\n')
fprintf('1. Gambar jaringan dalam bentuk graf\n')
fprintf('2. Tabel perutean\n')
pilihan=input('Masukkan pilihan Anda :\n')
if (pilihan==1)
    a=input('Masukkan matriks terboboti :\n');
    a1=sparse(a);
    a2=tril(a1);
    view(biograph(a2,[],'ShowArrows','off','ShowWeights','on'))
elseif (pilihan==2)
v=input('Banyak simpul:');
a=input('Masukkan matriks terboboti :\n');
a1=sparse(a);
BGobj=biograph(a1);
sumber=input('Masukkan simpul sumber : ');
fprintf('\n\n')
fprintf('Tabel Penuh untuk %d \n', sumber)
for i=1:v
tujuan=i;
[dist,path,pred]=shortestpath(BGobj,sumber,tujuan,'Method','Dijkstra
');
if(tujuan==sumber)
    fprintf('Tujuan : %d, Hop Berikutnya : - \n', i)
elseif(tujuan==path(2))
    fprintf('Tujuan : %d, Hop Berikutnya : %d \n', i, path(2))
else
   fprintf('Tujuan : %d, Hop Berikutnya : %d \n', i, path(2))
end
end
end
```

Kode Program Tabel *Routing* Algoritma *Link State Routing* dengan *Clustering*

```
clear all;


fprintf('Pilihan\n')
fprintf('1. Gambar jaringan dalam bentuk graf\n')
fprintf('2. Tabel Perutean\n')
pilihan=input('Masukkan pilihan Anda : ');
if(pilihan==1)
    a=input('Masukkan matriks terboboti :\n');
    a1=sparse(a);
    a2=tril(a1);
    view(biograph(a2,[],'ShowArrows','off','ShowWeights','on'))
else if(pilihan==2)
        a=input('Masukkan matriks terboboti :\n');


        v = size(a,1);
        a1=sparse(a);
        BGobj=biograph(a1);
        [tree, pred] = minspantree(BGobj,'Method','Kruskal');

view(biograph(tree,[],'ShowArrows','off','ShowWeights','on'))
        sum1=sum(tree);
        sum2=sum(sum1);
        m=sum2/nnz(tree);
tree(tree>m)=0;

view(biograph(tree,[],'ShowArrows','off','ShowWeights','on'))
```

```
[row,col] = find(full(tree));
a2 = [row';col'];
[baris kolom] = size(a2);

z = 1;
j = 1;
while j < kolom + 1
temp = a2(:,j);
cek = a2;
cek(:,j) = 0;
reg(z).data = temp;
for m = 1 : baris
    for i = 1 : baris
        for k = 1 : kolom
            if temp(m) == cek(i,k)
                reg(z).data = [reg(z).data cek(:,k)];
            end
        end
    end
```

```
end
[brs klm] = size(reg(z).data);

count = 1;
while count < klm
    temp = reg(z).data(:,count+1);
    cek = a2;
    cek(:,1:j+klm-1) = 0;
    for m = 1 : baris
        for i = 1 : baris
            for k = 1 : kolom
                if temp(m) == cek(i,k)
                    reg(z).data = [reg(z).data cek(:,k)];
                end
            end
        end
    end
    count = count + 1;
end
if klm == 1
    j = j + 1;
else
    j = j + count;
end
```

```
if reg(z).data(end) ~= a2(end)
    z = z + 1;
else
    j = j + kolom;
    z = z + 1;
end
end

node = (1:1:v);

for y = 1 : z-1
    reg(y).hasil(1)=reg(y).data(1);
    [ab,cd]=size(reg(y).hasil);
    [m,n]=size(reg(y).data);
    uk = m*n;
    w = 2;
    for r = 1 : v
        if node(r) == reg(y).hasil(1)
            node(r) = 0;
        end
    end
    for q = 2 : uk
        cek = 0;
        [br,kl] = size(reg(y).hasil);
```

```
            for loop = 1 : kl
                if reg(y).hasil(loop) == reg(y).data(q)
                    cek = 1;
                end
            end
            if cek == 0
                reg(y).hasil(w) = reg(y).data(q);
                f = w;
                w = w + 1;
            end
            for r = 1 : v
                if node(r) == reg(y).hasil(f)
                    node(r) = 0;
                end
            end
        end
    end

    for t = 1 : z-1
        fprintf('\n Region(%d) = \n',t);
        reg(t).hasil = sort(reg(t).hasil);
        disp(reg(t).hasil)
    end
```

```
[row,column]=size(node);


        for loop = 1 : column
            if node(loop) ~= 0
                reg(z).hasil = node(loop);
                fprintf('\n Region(%d) = \n',z);
                disp(reg(z).hasil)
                z = z + 1;
            end
        end

        bnyk_region = z-1;


        S = input(' Sumber : ');


        pointer = 0;
        for a = 1 : bnyk_region
            [m,n] = size(reg(a).hasil);
            for b = 1 : n
                if S == reg(a).hasil(b)
                    pointer = a;
                    uk = n;
                end
            end
```

```
end

        T = reg(pointer).hasil;
        BGobj = biograph(a1);

        for d = 1 : uk
            [dist,path,pred] = shortestpath(BGobj,S,T(d));
            if size(path,2) > 1
                hop = path(2);
                fprintf('Tujuan : %d, Hop Berikutnya :
%d\n',T(d),hop);
            else
                fprintf('Tujuan : %d, Hop Berikutnya : -\n',T(d));
            end

        end

        for e = 1 : bnyk_region
          if e ~= pointer
            T = reg(e).hasil;
            BGobj = biograph(a1);
            [dist,path,pred] = shortestpath(BGobj,S,T(1));
            d_path = dist;
            s_path = path;
            for j = 2 : size(T,2)
```

```
[dist,path,pred]=shortestpath(BGobj,S,T(j));
                if dist < d_path
                    d_path = dist;
                    s_path = path;
                end
            end
            if size(s_path,2) > 1
                hop = s_path(2);
                fprintf('Tujuan : Region(%d), Hop Berikutnya :
%d\n',e,hop);
            else
                fprintf('Tujuan : Region(%d), Hop Berikutnya : -
\n',e);
            end

        end
    end
end

fprintf('\n\n');
end
```