STUDI IMPLEMENTASI ALGORITMA DIJKSTRA PADA PROTOKOL PERUTEAN *OPEN SHORTEST PATH FIRST* (OSPF)

Raden Aprian Diaz Novandi – NIM: 13505102

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
E-mail: if15102@students.if.itb.ac.id

Abstrak
Makalah ini membahas tentang studi implementasi algoritma Dijkstra pada salah satu protokol perutean (*routing protocol*) populer, yaitu *Open Shortest Path First* (OSPF). Protokol ini sering dipakai untuk perutean pada jaringan komputer lokal dengan skala menengah hingga besar, misalkan pada kantor-kantor yang menggunakan lebih dari 50 komputer beserta perangkat lainnya, atau perusahaan dengan banyak cabang di luar negeri. Para administrator jaringan berskala menengah hingga besar pun paling tidak pernah mengenal protokol perutean tersebut meski kadang belum pernah mengimplementasikannya secara langsung. Beberapa kelebihan yang menyebabkan protokol OSPF populer adalah kekuatan, keandalan (reliabilitas), keluwesan (fleksibilitas), dan kecocokan (kompatibilitas) yang dimilikinya dibandingkan dengan beberapa protokol lain yang sama-sama protokol jenis *Interior Gateway Protokol* (IGP), seperti *Routing Information Protocol* (RIP) maupun *Interior Gateway Routing Protocol* (IGRP).

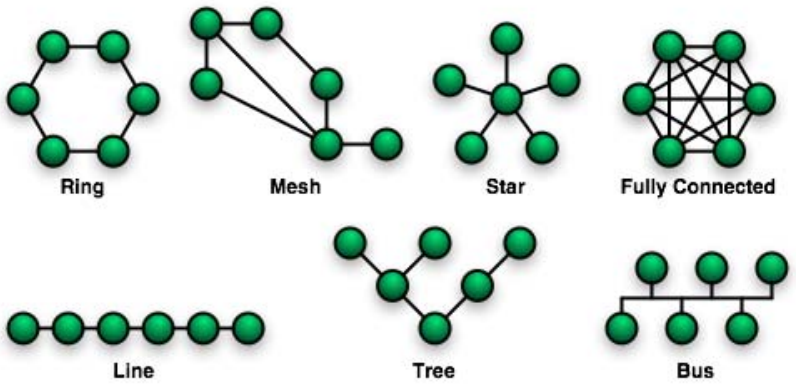
Algoritma Dijkstra diterapkan dalam protokol OSPF untuk memilih rute terbaik yang harus ditempuh oleh suatu paket data dari suatu alamat asal agar sampai di alamat tujuan dengan nilai satuan beban (*cost metric*) terkecil. Dalam teori graf, algoritma Dijkstra dimanfaatkan untuk memilih lintasan terpendek antara dua simpul dari suatu graf yang merupakan representasi topologi jaringan. Algoritma Dijkstra tersebut dijalankan oleh perute untuk mencari rute terbaik – dengan acuan nilai satuan beban terkecil – yang selanjutnya akan dimasukkan ke dalam tabel rute yang akan melaksanakan proses perutean paket data.

***Kata kunci**: perutean, protokol perutean, OSPF, algoritma Dijkstra, metric cost, lintasan terpendek*

1. Pendahuluan

Jaringan komputer terdiri atas beberapa komputer yang saling terhubung melalui saluran komunikasi, baik yang memanfaatkan kabel (contoh: kabel UTP, serat optik) maupun nirkabel (contoh: gelombang mikro, gelombang). Antara satu komputer dan komputer yang lain terkadang masing-masing berada di wilayah yang berbeda secara geografis (misalnya antarkota, antarpulau, antarbenua), dan tidak menutup kemungkinan masing-masing berada dalam wilayah geografis yang sama (misalnya berada dalam satu jaringan lokal). Pesan yang dikirim dari satu komputer ke komputer lainnya pada umumnya dipecah menjadi sejumlah paket data yang berukuran lebih kecil. Saluran komunikasi yang dipakai untuk mengantarkan paket data tersebut memiliki keterbatasan kecepatan transmisi. Agar paket data dapat sampai di alamat komputer yang diinginkan dengan besaran beban minimum (*minimum cost metric*), sistem jaringan komputer harus mampu melakukan pemilihan rute yang tepat untuk paket data tersebut. Proses pemilihan rute dari komputer asal ke komputer tujuan inilah yang disebut dengan perutean (*routing*).

Jaringan komputer dapat dimodelkan sebagai suatu graf berbobot terhubung. Setiap simpulnya melambangkan sebuah entitas komputer – bisa berupa komputer *workstation*, asisten digital pribadi (PDA), telepon seluler, ataupun beberapa piranti jaringan seperti perute (*router*), saklar (*switch*), dsb. –. Setiap simpul dalam graf memiliki alamat jaringan unik yang disebut alamat kontrol paut data (*DLC address*), atau kadang-kadang disebut dengan alamat kontrol akses media (*MAC address*). Sisi dalam graf melambangkan saluran komunikasi, sementara bobotnya menyatakan beberapa informasi yang meliputi lebatang pita (*bandwith*), waktu transmisi tunda (*delay transmission time*), jumlah hop, panjang lintasan secara fisik, keandalan, serta biaya komunikasi. Informasi-informasi tersebut dicakup dalam suatu besaran perutean (*routing metric*) yang dipakai sebagai acuan oleh algoritma perutean untuk menentukan apakah suatu rute bernilai lebih baik dibandingkan dengan rute-rute yang lain. Secara logis, rute yang baik adalah yang memiliki nilai satuan perutean minimum (*minimum routing metric*).



Gambar 1 Graf Topologi Jaringan

2. Dasar-dasar *Networking* dan Perutean

Salah satu hal penting dari komunikasi antara dua komputer adalah pemodelan komunikasi ini ke suatu bentuk representasi deskriptif. Artinya, representasi tersebut bisa menjelaskan bagaimana mekanisme komunikasi antara dua komputer tadi berlangsung. Salah satu model yang sering dipakai adalah *open system interconnection basic reference model* (*OSI basic reference model* atau selanjutnya dikenal dengan model OSI).

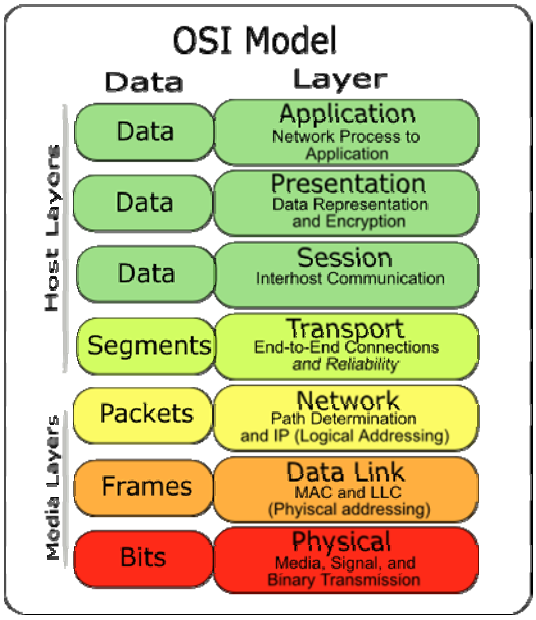
2.1 Model OSI

Model OSI adalah deskripsi abstrak dari komunikasi antarkomputer sekaligus desain protokol jaringan yang ada di dalamnya. Dalam model OSI ini, suatu entitas dalam jaringan komputer dibagi ke dalam tujuh lapis (*layer*), yang di setiap lapis ini, satu atau lebih entitas menjalankan fungsinya masing-masing secara berbeda, sehingga model OSI ini sering disebut juga dengan model referensi 7 lapis.

Tujuh lapis pada model OSI tersebut adalah:
1. Lapis fisik (*physical layer*)
2. Lapis taut data (*data link layer*)
3. Lapis jaringan (*network layer*)
4. Lapis transpor (*transport layer*)
5. Lapis sesi (*session layer*)
6. Lapis presentasi (*presentation layer*)
7. Lapis aplikasi (*application layer*)

Saat dua komputer berkomunikasi, tiap lapis OSI berkomunikasi dengan lapis yang sama pada perangkat yang lain (*peer process*). Misalnya: lapis aplikasi dari komputer A berkomunikasi dengan lapis aplikasi dari komputer B dengan meneruskan data melalui lapis-lapis yang lain. Masing-masing lapis aplikasi tidak peduli bagaimana lapis-lapis yang lain berfungsi, tetapi mereka bergantung kepada lapis-lapis tersebut untuk mendapatkan layanan.

Proses yang terjadi ketika suatu data dikirim dari aplikasi komputer sumber adalah pergerakan menurun dari data yang sudah berbentuk paket data melalui semua lapis. Ketika paket data telah mencapai lapis fisik, paket data tersebut siap dikirimkan melalui media komunikasi. Pada layer fisik, bit-bit paket data bisa berupa analog maupun digital, dan bisa dalam berbagai bentuk (contoh: listrik, cahaya, gelombang radio, dsb.). Dalam setiap pergerakan melalui lapis-lapis tadi, paket data dibungkus (dienkapsulasi) dengan informasi yang ditambahkan sebagai *header* atau *trailer*. Namun yang patut dicatat, data di dalam paket data tidak ada yang berubah.



Gambar 2 Model OSI

2.1.1 Lapis Fisik (*Physical Layer*)

Lapis fisik ini mendefinisikan semua spesifikasi listrik dan fisik untuk seluruh perangkat komunikasi. Berikut ini termasuk di dalamnya: rancangan pin, tegangan, dan spesifikasi kabel. Hub, pengulang (*repeater*), adapter jaringan (*network adapter*), dan Adapter Bus Hos (HBA yang dibagai dalam *Storage Area Network*) adalah perangkat-perangkat penting dalam lapis fisik ini. Fungsi dan layanan utama yang disediakan adalah:

1. Pendirian (*establishment*) dan pengakhiran (*termination*) dari suatu sambungan ke perantara komunikasi

2. Keikutsertaan (partisipasi) dalam suatu proses di mana sumber daya komunikasi dibagi secara mangkus di antara para pemakai. Contohnya adalah resolusi pendirian (*contention resolution*) dan kendali aliran (*flow control*).

3. Modulasi atau penukaran (konversi) antara representasi digital data yang ada di peralatan para pemakai (user) dan sinyal-sinyal koresponden yang dikirimkan melalui saluran komunikasi. Sinyal-sinyal ini dioperasikan melalui kabel fisik (contoh: tembaga dan serat optik) atau secara nirkabel (contoh: gelombang radio)

Bus-bus paralel SCSI beroperasi di lapis ini. Beberapa standar Ethernet lapis-fisik juga berada di lapis ini – di samping tergabung juga ke dalam lapis paut-data –. Hal yang sama berlaku pada jaringan lokal lainnya, seperti cincin token (*token ring*), *fiber distributed data interface* (FDDI), IEEE 802.11, dan juga jaringan pribadi (*personal area network*) seperti *bluetooth* dan IEEE 802.15.4.

2.1.2 Lapis Taut Data (*Data Link layer*)

Lapis taut data menyediakan kemampuan fungsional dan prosedural untuk melakukan pengiriman data antarentitas jaringan serta mendeteksi sekaligus membetulkan kesalahan yang mungkin terjadi di lapis fisik. Contoh media yang paling dikenal adalah Eternet. Contoh lainnya: *High-Level Data Link Control* (HDLC), *Analog-to-Digital Converter Control Protocol* (ADCCP) untuk jaringan alihpaket (*packet-switched networks*), dan Aloha untuk LAN. Pada IEEE 802 LAN, dan beberapa jaringan non-IEEE 802 seperti FDDI, layer ini terpecah menjadi lapis Media Access Control (MAC) dan lapis IEEE 802.2 *Logical Link Control* (LLC), yang berfungsi menyusun bit-bit dari layer fisik menjadi bentuk potongan-potongan data yang dikenal dengan sebutan bingkai data (*data frame*).

Lapis taut data adalah lapis yang menjadi tempat beroperasinya jembatan (*bridge*) dan saklar (*switch*) jaringan. Ketersambungan (konektivitas) disedikan hanya kepada simpul-simpul jaringan yang tersambung secara lokal dan membentuk domain lapis 2 untuk penerusan paket data baik secara *unicast* maupun *broadcast*. Protokol lain dimasukkan ke dalam bingkai data untuk menciptakan terowongan (*tunnel*), dan terpisah dengan 2 domain penerus tadi.

2.1.3 Lapis Jaringan (*Network Layer*)

Lapis jaringan menyediakan kemampuan fungsional dan prosedural untuk melakukan transfer VLDS (*variable length data sequences*) dari suatu alamat asal ke alamat tujuan lewat satu atau lebih jaringan sambil memelihara kualitas layanan yang diminta oleh lapis Transpor. Lapisan jaringan melakukan fungsi perutean jaringan, dan (kemungkinan besar) juga melakukan proses segmentasi/desegmentasi, dan melaporkan kegagalan fungsi pengiriman. Perute beroperasi di lapis ini serta melaksanakan pengiriman data melalui jaringan yang lebih luas. Skema ini disebut dengan skema pengalamatan logis, yang nilainya ditentukan oleh desainer jaringan. Contoh yang protokol yang paling dikenal adalah protokol Internet (IP). Di dunia nyata, analogi paling mirip dari lapis ini adalah pengirim surat dari titik A ke titik B.

2.1.4 Lapis Transpor (*Transport Layer*)

Lapis transpor menyediakan transfer data transparan di antara para pengguna terakhir (*end-user*). Lapis transpor membebastugaskan lapis di atasnya dari berbagai penanganan masalah, sambil menyediakan transfer data yang andal. Lapis transpor mengendalikan keandalan dari pautan yang diberikan melalui kendali aliran, segmentasi/desegmentasi, dan pengendalian kesalahan. Beberapa protokol berorientasi pada keadaan dan koneksi. Ini artinya bahwa lapis transpor dapat tetap meneruskan paket dan mengirimkan ulang paket data yang gagal. Contoh protokol yang paling sering ditemui di lapis 4 ini adalah *Transmission Control Protocol* (TCP). Lapis transpor adalah lapis yang mengubah pesan-pesan ke dalam segmen-segmen TCP atau *User Datagram Protocol* (UDP), *Stream Control Transmission Protocol* (SCTP), dsb. Analogi di dunia nyata dari lapis 4 ini adalah Kantor Pos, yang mengurus masalah pengiriman dan penggolongan surat serta bingkisan (parsel) yang akan dikirim.

2.1.5 Lapis Sesi (*Session Layer*)

Lapis sesi mengendalikan percakapan (dialog) antarkomputer. Lapis ini mendirikan, mengatur, dan mengakhiri koneksi antara aplikasi lokal dan aplikasi jarak jauh. Lapis ini juga menyediakan operasi-operasi baik yang bersifat dupleks maupun semidupleks, serta melakukan prosedur pemeriksaan, penundaan, pengakhiran, maupun pengulangan (*restart*). Model OSI memberikan tanggung jawab kepada lapis ini untuk pengaturan sesi yang bersifat "graceful close" – yang sebenarnya merupakan milik dari TCP –, pemeriksaan sesi, serta pemulihan (*recovery*), yang biasanya jarang dipakai pada protokol Internet.

2.1.6 Lapis Presentasi (*Presentation Layer*)

Lapis presentasi mengubah bentuk data untuk menyediakan antarmuka standar bagi lapis aplikasi. *Encoding* MIME, kompresi data, enkripsi data, serta manipulasi yang mirip dari presentasi dilakukan pada lapis ini untuk memberikan data sebagai layanan sampai pengembang protokol melihat kecocokan di dalamnya. Contoh: mengubah arsip teks berkode EBCDIC ke dalam arsip berkode ASCII, atau melakukan serialisasi objek dan struktur data (contoh: XML).

2.1.7 Lapis Aplikasi (*Application Layer*)

Lapis aplikasi menyediakan akses kepada pengguna (*user*) untuk memperoleh informasi pada jaringan melalui suatu aplikasi. Lapis ini adalah antarmuka utama bagi para pengguna untuk berinteraksi dengan aplikasi dan jaringan. Beberapa contoh protokol lapis aplikasi adalah Telnet, aplikasi yang memanfaatkan *File Transfer Protocol* (FTP), aplikasi yang memanfaatkan *Simple Mail Transfer Protocol* (SMTP), dan aplikasi yang memanfaatkan *Hypertext Transfer Protocol* (HTTP). Aplikasi yang dibangun untuk memanfaatkan suatu protokol, misalnya FTP, tidak boleh rancu dengan protokol itu sendiri – yang terletak di lapis sesi –.

2.2 Perutean

Pada bagian pendahuluan telah dijelaskan definisi dari perutean, yaitu pemilihan rute yang tepat untuk pengiriman paket data dari komputer asal ke komputer tujuan. Proses perutean ini memiliki beberapa protokol (tata cara), algoritma, dan skema yang berbeda satu sama lain. Dalam makalah ini protokol yang dibahas adalah protokol-protokol yang masuk kelompok *Interior Gateway Protocol* (IGP) yang bekerja di jaringan internal suatu organisasi atau perusahaan (definisi tentang jaringan internal bisa dibaca pada bagian 3 yang membahas lebih rinci masalah protokol perutean OSPF). Proses perutean didasarkan pada suatu tabel perutean, yaitu basis data pada perute yang menyimpan informasi mengenai topologi jaringan, serta nantinya akan dipakai untuk proses penerusan paket data hingga alamat tujuan dicapai.

2.3 Beberapa Protokol Perutean IGP

Beberapa protokol perutean populer yang masuk kelompok Interior Gateway Protocol (IGP) adalah:

1. *Routing Information Protocol* (RIP)
2. *Open Shortest Path First* (OSPF)
3. *Interior Gateway Routing Protocol* (IGRP)

2.3.1 RIP

Routing Information Protocol (RIP) adalah protokol yang memanfaatkan algoritma Bellman-Ford (kelompok protokol *distance-vector*) dalam pemilihan rute terbaiknya. Dibandingkan dengan protokol OSPF, protokol RIP memiliki tingkat kompleksitas komputasional yang lebih rendah, sehingga konsumsi sumber daya memorinya juga lebih rendah. Akan tetapi, konsekuensi yang ditimbulkan dari hal tersebut adalah bahwa penggunaan RIP hanya terbatas pada jaringan menengah ke bawah dengan jumlah *host* yang tidak terlalu besar. Untuk melihat hal yang lebih rinci tentang algoritma Bellman-Ford dan protokol *distance-vector*, silakan baca upabahasan algoritma Bellman-Ford (protokol *distance-vector*).

Perlu diketahui bahwa RIP tidak mengadopsi protokol *distance-vector* begitu saja, melainkan dengan melakukan beberapa penambahan pada algoritmanya agar kalang perutean dapat diminimalkan. *Split horizon* digunakan RIP untuk meminimalkan efek lambung (*bouncing*). Prinsip yang digunakan *split horizon* adalah: jika simpul A menyampaikan datagram ke tujuan X melalui simpul B, maka bagi B tidak masuk akal untuk mencapai tujuan X melalui A. Jadi, A tidak perlu memberitahu B bahwa X dapat dicapai B melalui A.

Untuk mencegah kasus menghitung sampai tak hingga, RIP menggunakan metode *Triggered Update*. RIP memiliki penghitung waktu (*timer*) untuk mengetahui kapan perute harus kembali memberikan informasi perutean. Jika terjadi perubahan pada jaringan, sementara *timer* belum habis, perute tetap harus mengirimkan informasi perutean karena dipicu oleh perubahan tersebut (*triggered update*). Dengan demikian, perute-perute dalam jaringan dapat dengan cepat mengetahui perubahan yang terjadi dan meminimalkan kemungkinan kalang loop (*routing loop*) terjadi.

2.3.2 OSPF

Open Shortest Path First (OSPF) adalah protokol yang memanfaatkan algoritma Dijkstra dalam pemilihan rute terbaiknya. OSPF sangat baik dalam penanganan jaringan dengan jumlah host yang sangat besar, karena pemeliharaan tabel perutean yang dibuat OSPF bersifat otomatis. Dibandingkan dengan protokol RIP, konsumsi sumber daya memori pada perute OSPF lebih besar. Untuk melihat pembahasan lebih lengkap mengenai OSPF, silakan baca bagian 3 yang khusus membahas mengenai mekanisme kerja protokol OSPF.

2.3.3 IGRP

Interior Gateway Routing Protocol (IGRP) adalah protokol yang dibuat oleh vendor Cisco. Biasa dimanfaatkan oleh perute Cisco atau perute milik vendor lain yang lisensi teknologinya dimiliki oleh Cisco. Protokol ini memiliki kemiripan sifat dengan RIP, yang sama-sama unggul di jaringan dengan jumlah *host* sedikit. Saat ini telah berkembang versi terbaru dari IGRP, yaitu EIGRP *(Enhanced Interior Gateway Routing Protocol*). Algoritma perutean yang dipakai oleh protokol IGRP maupun EIGRP adalah algoritma Bellman-Ford.

2.4 Algoritma Perutean

Algoritma perutean adalah cara yang dipakai oleh suatu protokol perutean untuk menentukan rute terbaik untuk mencapai alamat di jaringan tujuan. Secara umum ada dua macam algoritma populer yang sering dipakai oleh protokol perutean, yaitu algoritma Bellman-Ford (pada golongan protokol *distance vector*), serta algoritma Dijkstra (pada golongan protokol *link-state*). Pada upabahasan 2.4 ini, pembahasan lebih lengkap dikhususkan pada algoritma Bellman-Ford. Sementara, algoritma Dijkstra akan dibahasan lebih rinci pada upabahasan 4.

2.4.1 Algoritma Bellman-Ford (*Distance Vector Protocol*)

Algoritma Bellman-Ford menghitung lintasan terpendek dari suatu graf berbobot, yang mungkin pada sisinya memiliki bobot negatif. Algoritma Dijkstra menyelesaikan permasalahan yang sama dengan waktu yang lebih singkat, namun memiliki syarat bobot sisi graf tidak ada yang negatif. Maka dari itu, algoritma Bellman-Ford hanya dipakai ketika ada bobot sisi graf yang negatif.

Pseudokode algoritma Bellman-Ford:

```
function BellmanFord(list Simpul, list
Sisi, simpul asal)

   // Langkah 1: Inisialisasi graf
   for each simpul v dalam Simpul:
       if v adalah asal then v.jarak := 0
       else v.jarak := infinity
       v.pendahulu := null

   // Langkah 2: mengurangi sisi secara
berulang-ulang
   for i from 1 to jumlah simpul:
       for each sisi uv pada Sisi:
           u := uv.asal
           v := uv.tujuan // uv adalah
sisi dari u ke v
           if v.jarak > u.jarak +
uv.bobot:
               v.jarak := u.jarak +
uv.bobot
               v.pendahulu := u

   // Langkah 3: memeriksa siklus
berbobot negatif
   for each sisi uv pada Sisi:
       u := uv.asal
       v := uv.tujuan
       if v.jarak > u.jarak + uv.bobot:
           error "Graf mengandung siklus
berbobot negatif"
```

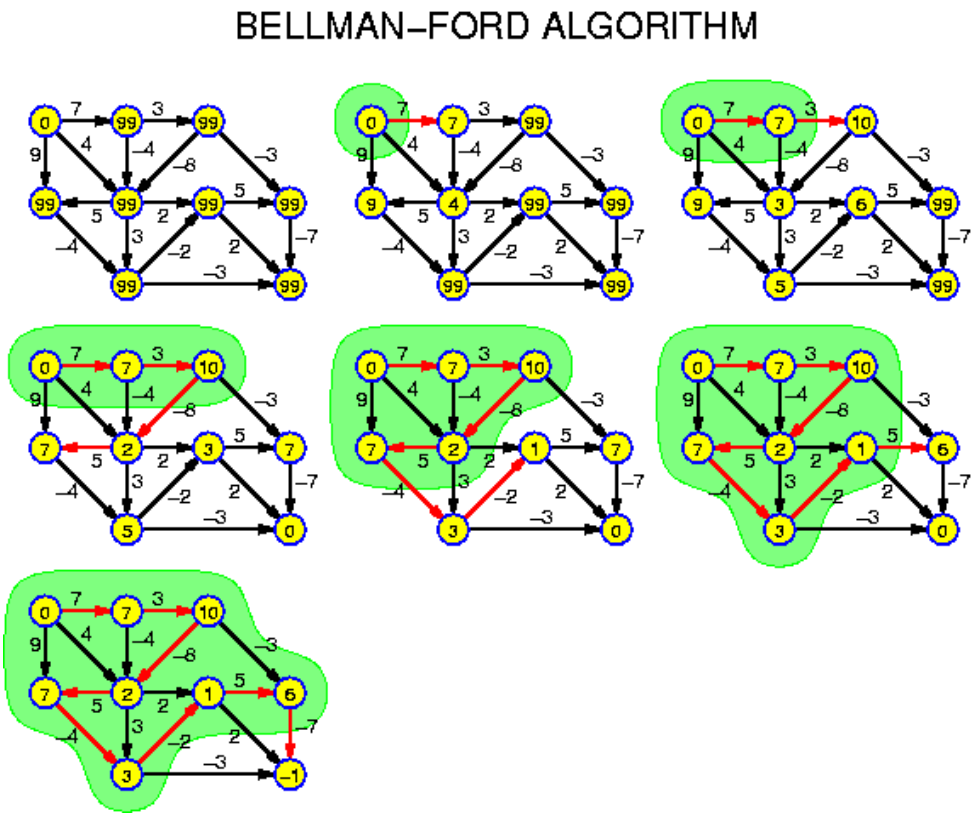
Algoritma Bellman-Ford biasa dipakai pada jenis protokol perutean *distance-vector*, misalnya RIP. Algoritma ini disebarkan karena melibatkan beberapa simpul (dalam hal ini perute) dalam suatu sistem Autonom (*Autonomous System*), sekumpulan jaringan IP dan perute di bawah kendali dari suatu entitas, yang mengatur kebijakan perutean di jaringan tersebut, umumnya jaringan tersebut berada dalam satu ISP (Penyedia Layanan Internet). Algoritma ini terdiri dari beberapa langkah, yaitu:

1. Setiap simpul menghitung jarak antara dirinya sendiri dengan semua simpul di dalam sistem autonom dan menyimpan informasi ini sebagai suatu tabel
2. Setiap simpul mengirimkan tabelnya ke semua simpul tetangga
3. Ketika suatu simpul menerima tabel jarak dari tetangganya, simpul tersebut akan menghitung rute terpendek ke semua simpul dan memperbarui tabelnya dan menyebarkan perubahan yang terjadi

Setiap perute dengan protokol *distance-vector* ketika kali pertama dijalankan hanya mengetahui cara perutean ke dirinya sendiri (informasi lokal) dan tidak mengetahui topologi jaringan tempatnya berada. Perute kemudian mengirimkan informasi lokal tersebut dalam bentuk *distance-vector* ke semua pautan yang terhubung langsung dengannya. Perute yang menerima informasi perutean menghitung *distance-vector*, menambahkan *distance-vector* dengan metrik pautan tempat informasi tersebut diterima, dan memasukkannya ke dalam entri tabel penerusan (forwarding) jika dianggap merupakan jalur terbaik. Informasi *routing* setelah penambahan metrik kemudian dikirim lagi ke seluruh antarmuka perute, dan ini dilakukan setiap selang waktu tertentu. Demikian

seterusnya sehingga seluruh perute di jaringan mengetahui topologi jaringan tersebut.

Protokol *distance-vector* memiliki kelemahan yang dapat terlihat apabila dalam jaringan ada pautan yang terputus. Dua kemungkinan kegagalan yang mungkin terjadi adalah efek *bouncing* dan menghitung sampai tak hingga (*counting to infinity*). Efek *bouncing* dapat terjadi pada jaringan yang menggunakan metrik yang berbeda pada minimal sebuah link. Pautan yang putus dapat menyebabkan kalang perutean (*routing loop*), sehingga datagram yang melewati pautan tertentu hanya berputar-putar di antara dua perute (*bouncing*) sampai umur (*time to live*) datagram tersebut habis.



Gambar 3 Contoh Ilustrasi Algoritma Bellman-Ford (menentukan lintasan terpendek dari simpul paling kiri atas ke simpul paling kanan bawah)

2.4.2 Algoritma Dijkstra (*Link-state Protocol*)

Algoritma Dijkstra adalah algoritma yang digunakan untuk menghitung jarak terpendek dari suatu simpul ke simpul yang lain pada kelompok protokol *link-state*, misalnya OSPF. Algoritma ini pada dasarnya bekerja pada suatu graf berbobot yang bobot sisi-sisinya tidak boleh ada yang negatif. Waktu komputasi yang diperlukan lebih singkat daripada algoritma Bellman-Ford.

Algoritma Dijkstra ini termasuk dalam golongan algoritma *greedy*, yang secara harafiah berarti tamak/rakus. Prinsip algoritma *greedy* adalah "Ambillah apa yang bisa Anda dapatkan sekarang!", serta penyelesaian masalahnya dilakukan langkah per langkah. Penjelasan lengkap mengenai algoritma Dijkstra bisa dilihat pada bahasan 4, yang memang khusus membahas mengenai penerapan algoritma Dijkstra pada protokol perutean OSPF.

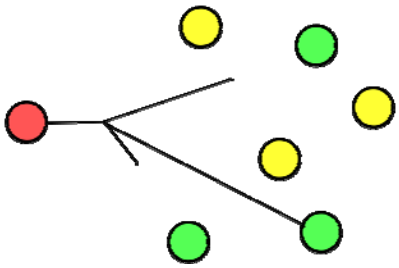
2.5 Skema Perutean (*Routing Scheme*)

Skema perutean adalah cara pengalamatan dan pengiriman paket informasi kepada setiap simpul dalam jaringan. Skema perutean terdiri atas: *anycast*, *broadcast*, *multicast*, dan *unicast*. Penjelasan masing-masing dari skema perutean bisa dilihat pada upabahasan berikut.

2.5.1 Anycast

Anycast adalah pengalamatan internet dan skema perutean yang merutekan data ke tujuan terpendek dan terbaik dilihat dari sisi topologi perutean. Di internet, *anycast* biasanya diterapkan oleh BGP untuk mengumumkan secara simultan daerah jelajah tujuan IP yang sama dari beberapa tempat berbeda di internet. Hasilnya, paket-paket data akan dialamatkan ke alamat tujuan dalam daerah jelajah ini ke titik

terdekat dalam suatu jaringan. Anycast lebih cocok untuk protokol-protokol yang bersifat nirkoneksi (misalnya UDP), dibandingkan dengan protokol yang berorientasi pada koneksi, misalnya TCP, ataupun protokol yang berbasis UDP namun tetap menjaga kondisi mereka, ketika penerima (*receiver*) yang dipilih untuk sumber yang diberikan mungkin berubah dari waktu ke waktu sebagai perubahan rute yang optimal.

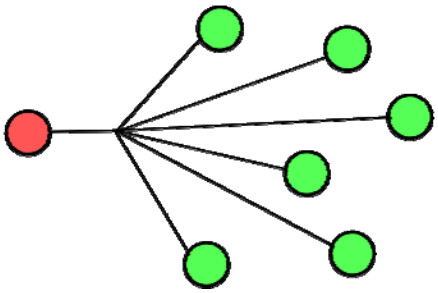


Gambar 4 Skema Perutean *Anycast*

2.5.2 Broadcast

Dalam jaringan komputer, *broadcasting* adalah pengiriman paket data yang akan diterima oleh setiap entitas dalam jaringan. Praktiknya, lingkup *broadcast* hanya terbatas pada daerah asal *broadcast* (*broadcast domain*). Tidak semua jaringan komputer mampu mendukung skema *broadcast*; contohnya, X.25 dan *frame relay*. *Broadcasting* hanya dibatasi pada teknologi LAN (*Local Area Network*); pada umumnya Eternet maupun cincin Token, yang hasil performa dari boradcasting tidak bisa luas seperti ketika berada dalam suatu WAN (*Wide Area Network*).

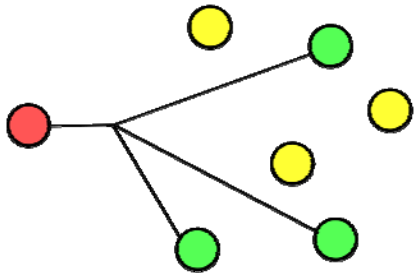
Baik Eternet maupun IPv4 memakai semua alamat *broadcast* untuk menunjukkan suatu paket *broadcast*, sementara itu Cincin Token menggunakan nilai tertentu pada bidang kendali IEEE 802.2. Karena pendekatannya yang bersifat "shotgun" pada penyebaran data, *broadcast* lambat laun tergantikan oleh *multicast*.



Gambar 5 Skema Perutean *Broadcast*

2.5.3 Multicast

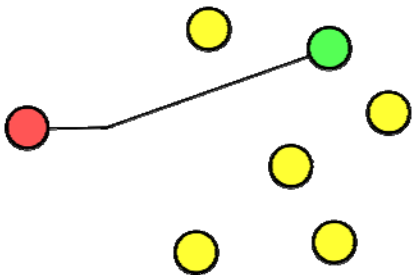
Multicast adalah pengiriman paket informasi ke suatu kelompok tujuan secara bersamaan memakai strategi paling mangkus untuk mengirimkan pesan yang melewati setiap pautan dalam jaringan tepat sekali dan hanya membuatkan gandaan (*copy*) ketika suatu entitas memintanya. Istilah *multicast* biasanya berhubungan dengan IP *multicast*, penerapan dari konsep *multicast* pada aras perutean IP, di mana perute menciptakan pohon distribusi merentang maksimum untuk proses pengiriman datagram ke alamat tujuan multicast dalam waktu yang sesungguhnya. IP Multicast adalah teknik untuk komunikasi yang melibatkan banyak entitas pada infrastruktur IP. Multicast memanfaatkan infrastruktur jaringan secara mangkus dengan meminta sumber untuk mengirimkan paket hanya sekali, bahkan jika yang membutuhkan paket tersebut adalah penerima (*receiver*) dalam jumlah yang besar.



Gambar 6 Skema Perutean *Multicast*

2.5.4 Unicast

Dalam jaringan komputer, *unicast* adalah pengirim paket informasi ke suatu tujuan tunggal. *Unicast* adalah kebalikan dari *broadcast*. Biasanya *unicast* dipakai untuk mendapatkan kemangkusan dari skema *broadcast*. Istilah ini juga mirip dengan analogi layanan *streaming* penyediaan konten. Server *unicast* menyediakan aliran ke pengguna tunggal (*single user*) dalam suatu waktu, sementara itu server *multicast* bisa mendukung pengguna yang banyak dengan melayani konten secara bersama-sama ke berbagai pengguna tersebut.



Gambar 7 Skema Perutean Unicast

3. Protokol OSPF

OSPF merupakan sebuah protokol perutean berjenis IGP (*interior gateway protocol*) yang hanya dapat bekerja dalam jaringan internal suatu organisasi atau perusahaan. Jaringan internal maksudnya adalah jaringan di mana pengguna masih memiliki hak untuk menggunakan, mengatur, dan memodifikasinya, atau dengan kata lain, pengguna masih memiliki hak administrasi terhadap jaringan tersebut. Jika pengguna sudah tidak memiliki hak untuk menggunakan dan mengaturnya, maka jaringan tersebut dapat dikategorikan sebagai jaringan eksternal. Selain itu, OSPF juga merupakan protokol perutean yang berstandar terbuka. Maksudnya, protokol perutean ini bukan ciptaan dari vendor mana pun, sehingga siapa pun dapat menggunakannya, perangkat mana pun dapat cocok dengannya, dan di mana pun protokol perutean ini dapat diimplementasikan.

OSPF merupakan protokol perutean yang menggunakan konsep perutean hierarkis, artinya OSPF membagi-bagi jaringan menjadi beberapa tingkatan. Tingkatan-tingkatan ini diwujudkan dengan menggunakan sistem pengelompokan area. Dengan menggunakan konsep perutean hierarki ini sistem penyebaran informasi dalam protokol OSPF menjadi lebih teratur dan tersegmentasi alias tidak menyebar ke mana-mana secara sembarangan. Efek dari keteraturan penyebaran perutean ini adalah penggunaan *bandwith* jaringan menjadi lebih mangkus, lebih cepat mencapai konvergensi, dan lebih akurat dalam menentukan rute-rute terbaik menuju ke sebuah lokasi.

Teknologi yang digunakan oleh routing protokol ini adalah teknologi *link-state* yang memang didesain untuk bekerja dengan sangat mangkus dalam proses pembaruan informasi rute. Prinsip perutean *link-state* sangat sederhana. Sebagai pengganti dalam menghitung rute terbaik dengan cara terdistribusi, semua perute mempunyai peta jaringan dan menghitung semua rute terbaik dari peta ini. Peta jaringan tersebut disimpan dalam sebuah basis data dan setiap *record* dalam basis data tersebut menyatakan sebuah pautan (*link*) dalam jaringan. *Record-record* tersebut dikirimkan oleh perute yang terhubung langsung dengan masing-masing pautan. Karena setiap perute perlu memiliki peta jaringan yang menggambarkan kondisi terakhir topologi jaringan yang lengkap, setiap perubahan dalam jaringan harus diikuti oleh perubahan dalam basis data *link-state* yang terletak di setiap perute. Perubahan status pautan yang dideteksi perute akan mengubah basis data *link-state* perute tersebut, kemudian perute mengirimkan perubahan tersebut ke perute - perute yang lain.

Hal ini membuat routing protokol OSPF menjadi sangat mangkus untuk memelihara jaringan berskala sedang hingga besar. Pengguna OSPF biasanya memang para administrator jaringan berskala sedang hingga besar. Jaringan dengan jumlah perute lebih dari sepuluh buah, dengan banyak lokasi-lokasi jarak jauh yang perlu juga dijangkau dari pusat, dan jumlah pengguna jaringan lebih dari lima ratus perangkat komputer, umumnya akan memilih untuk menggunakan protokol perutean ini.

3.1 Konektivitas Antarperute

Untuk memulai semua aktivitas OSPF dalam menjalankan pertukaran informasi perutean, hal pertama yang dilakukan adalah membentuk sebuah mekanisme komunikasi antarperute. Perute lain yang berhubungan langsung atau yang berada di dalam satu jaringan dengan perute OSPF disebut dengan perute tetangga (*neighbor router*).

Langkah pertama yang harus dilakukan sebuah perute OSPF adalah membentuk hubungan dengan perute tetangga. Perute OSPF mempunyai sebuah mekanisme untuk dapat menemukan perute tetangganya dan dapat membuka hubungan. Mekanisme tersebut disebut dengan istilah protokol Hello. Dalam membentuk hubungan dengan perute tetangganya, perute OSPF akan mengirimkan sebuah paket berukuran kecil secara periodik ke dalam jaringan atau ke sebuah perangkat yang terhubung langsung dengannya. Paket kecil tersebut dinamai dengan istilah paket Hello. Pada kondisi standar, paket Hello dikirimkan secara berkala setiap 10 detik sekali (dalam media penyiaran multiakses) dan 30 detik sekali dalam media *point-to-point*.

Paket Hello berisikan informasi seputar pernak-pernik yang ada pada perute pengirim. Paket Hello pada umumnya dikirim dengan menggunakan alamat *multicast* untuk menuju ke semua perute yang menjalankan OSPF (IP *multicast*: 224.0.0.5). Semua perute yang menjalankan OSPF pasti akan mendengarkan protokol Hello ini dan juga akan mengirimkan paket Hello-nya secara berkala. Cara kerja dari protokol Hello dan pembentukan perute tetangga terdiri dari beberapa jenis, tergantung dari jenis media yang perute OSPF jalankan.

3.2 Media Kerja Protokol OSPF

Penjelasan 3.1 menyebutkan bahwa OSPF harus membentuk hubungan dulu dengan perute tetangganya untuk dapat saling berkomunikasi

seputar informasi perutean. Untuk membentuk sebuah hubungan dengan perute tetangganya, OSPF mengandalkan protokol Hello. Namun uniknya cara kerja protokol Hello pada OSPF berbeda-beda pada setiap jenis media. Ada beberapa jenis media yang dapat meneruskan informasi OSPF, dan masing-masing memiliki karakteristik sendiri, sehingga OSPF pun bekerja mengikuti karakteristik mereka. Media tersebut adalah:

1. *Broadcast Multiaccess*
2. *Point-to-Point*
3. *Point-to-Multipoint*
4. *Non-broadcast Multiaccess (NBMA)*

3.2.1 Broadcast Multiaccess

Media jenis ini adalah media yang banyak terdapat dalam jaringan lokal atau LAN seperti misalnya Eternet, FDDI, dan cincin Token (*Token ring*). Dalam kondisi media seperti ini, OSPF akan mengirimkan lalu-lintas *multicast* dalam pencarian perute-perute tetangganya. Namun, ada yang unik dalam proses pada media ini, yaitu akan terpilihnya dua buah perute yang berfungsi sebagai *Designated Router* (DR) dan *Backup Designated Router* (BDR).

3.2.2 Point-to-Point

Teknologi *point-to-point* digunakan pada kondisi di mana hanya ada satu perute lain yang terkoneksi langsung dengan sebuah perangkat perute. Contoh dari teknologi ini misalnya pautan (*link*) serial. Dalam kondisi *Point-to-Point* ini, perute OSPF tidak perlu membuat *Designated Router* dan *Back-up*-nya karena hanya ada satu perute yang perlu dijadikan sebagai tetangga. Dalam proses pencarian neighbour ini, perute OSPF juga akan melakukan pengiriman paket Hello dan pesan-pesan lainnya menggunakan alamat *multicast* bernama AllSPFRouters (224.0.0.5).

3.2.3 Point-to-Multipoint

Media jenis ini adalah media yang memiliki satu antarmuka yang menghubungkannya dengan banyak tujuan. Jaringan-jaringan yang ada di bawahnya dianggap sebagai serangkaian jaringan *point-to-point* yang saling terkoneksi langsung ke perangkat utamanya. Pesan-pesan protokol perutean OSPF akan digandakan ke seluruh jaringan *Point-to-Point* tersebut.

Pada jaringan jenis ini, lalu-lintas OSPF juga dikirimkan menggunakan alamat IP *multicast*. Tetapi yang membedakannya dengan media berjenis *broadcast multi-access* adalah tidak adanya pemilihan *Designated Router* (DR) dan *Backup Designated Router* (BDR), karena sifatnya yang tidak meneruskan siaran (*broadcast*).

3.2.4 Non-broadcast Multiaccess (NBMA)

Media berjenis *nonbroadcast multiaccess* ini secara fisik merupakan sebuah serial line biasa yang sering ditemui pada media jenis *point-to-point*. Namun faktanya, media ini dapat menyediakan koneksi ke banyak tujuan; tidak hanya ke satu titik saja. Contoh dari media ini adalah X.25 dan *frame relay* yang mampu menyediakan solusi bagi kantor-kantor yang terpencar lokasinya. Di dalam penggunaan media ini pun ada dua jenis penggunaan, yaitu pada jaringan *partial mesh* dan *fully mesh*.

OSPF melihat media jenis ini sebagai media *broadcast multiaccess*. Namun pada kenyataannya, media ini tidak bisa meneruskan siaran ke titik-titik yang ada di dalamnya. Maka, dalam penerapan OSPF pada media ini, dibutuhkan konfigurasi DR dan BDR yang dilakukan secara manual. Dalam media jenis ini, yang menjadi DR dan BDR adalah perute yang memiliki hubungan langsung ke seluruh perute tetangganya. Semua lalu-lintas yang dikirimkan dari perute-perute tetangga akan digandakan oleh DR dan BDR untuk masing-masing perute dan dikirim dengan menggunakan alamat *unicast* atau layaknya proses OSPF pada media *point-to-point*.

3.3 Mekanisme Kerja OSPF

Secara garis besar, proses yang dilakukan protokol perutean OSPF mulai dari awal hingga dapat saling bertukar informasi ada lima langkah, yaitu:

1. Membentuk Perute yang Bersebelahan (*Adjacency Router*)
2. Memilih DR dan BDR jika diperlukan
3. Mengumpulkan semua keadaan (*state*) jaringan
4. Memilih rute terbaik untuk digunakan
5. Menjaga kemutakhiran informasi perutean

3.3.1 Membentuk Perute yang Bersebelahan (*Adjacency Router*)

Arti harafiah *adjacency router* adalah perute yang bersebelahan atau yang terdekat. Jadi proses pertama dari perute OSPF ini adalah menghubungkan diri dan saling berkomunikasi dengan para perute terdekat atau perute tetangga. Untuk dapat membuka komunikasi, protokol Hello akan bekerja dengan mengirimkan paket Hello.

Misalkan ada dua buah perute, Perute A dan B yang saling berkomunikasi memakai protokol OSPF. Ketika OSPF kali pertama bekerja, maka kedua perute tersebut akan saling mengirimkan Paket Hello dengan alamat *multicast* sebagai tujuannya. Di dalam Paket Hello terdapat sebuah *field* yang berisi identitas (ID) tetangga.

Misalkan perute B menerima paket Hello lebih dahulu dari perute A. Maka perute B akan mengirimkan kembali paket Hello-nya dengan disertai ID dari perute A. Ketika perute A menerima paket Hello yang berisikan ID dari dirinya sendiri, maka perute A akan menganggap perute B adalah *adjacent router* dan mengirimkan kembali paket Hello yang telah berisi ID perute B ke perute B. Dengan demikian perute B juga akan segera menganggap perute A sebagai *adjacent router*-nya. Sampai di sini *adjacency router* telah terbentuk dan siap melakukan pertukaran informasi perutean.
Contoh pembentukan *adjacency router* di atas hanya terjadi pada proses OSPF yang berlangsung pada media *point-to-point*. Namun, prosesnya akan lain lagi jika OSPF berlangsung pada media *broadcast multiaccess* seperti pada jaringan eternet. Karena media penyiaran akan meneruskan paket-paket Hello ke seluruh perute yang ada dalam jaringan, maka *adjacency router*-nya tidak hanya satu. Proses pembentukan *adjacency router* akan terus berulang sampai semua perute yang ada di dalam jaringan tersebut menjadi *adjacent router*.

Jika semua perute menjadi *adjacent router*, maka komunikasi OSPF akan meramaikan jaringan. *Bandwidth* jaringan menjadi tidak mangkus terpakai karena jatah untuk data yang sesungguhnya ingin lewat di dalamnya akan berkurang. Untuk itu pada media *broadcast multiaccess* akan terjadi lagi sebuah proses pemilihan perute yang menjabat sebagai "juru bicara" bagi perute-perute lainnya. Perute juru bicara ini sering disebut dengan istilah *Designated Router* (DR). Selain perute juru bicara, disediakan juga cadangan untuk perute juru bicara ini. Perute ini disebut dengan istilah *Backup Designated Router* (BDR). Langkah berikutnya adalah proses pemilihan DR dan BDR, jika memang diperlukan.

3.3.2 Memilih DR dan BDR jika diperlukan

Dalam jaringan *broadcast multiaccess*, keberadaan DR dan BDR sangat diperlukan. DR dan BDR akan menjadi pusat komunikasi seputar informasi OSPF dalam jaringan tersebut. Semua paket pesan yang ada dalam proses OSPF akan disebarkan oleh DR dan BDR. Maka itu, pemilihan DR dan BDR menjadi proses yang sangat penting. Sesuai dengan namanya, BDR merupakan "bayangan" dari DR. Artinya BDR tidak akan digunakan sampai terjadi masalah pada perute DR. Ketika perute DR bermasalah, maka posisi juru bicara akan langsung diambil alih oleh perute BDR, sehingga perpindahan posisi juru bicara akan berlangsung dengan halus.

Proses pemilihan DR maupun BDR tidak lepas dari peran penting paket Hello. Di dalam paket Hello ada sebuah *field* yang berisi ID dan nilai prioritas sebuah perute. Semua perute yang ada dalam jaringan *broadcast multi-access* akan menerima semua paket Hello dari semua perute yang ada dalam jaringan tersebut pada saat kali pertama OSPF berjalan. Perute dengan nilai prioritas tertinggi akan menang dalam pemilihan dan langsung menjadi DR. Perute dengan nilai prioritas di urutan kedua akan dipilih menjadi BDR. Status DR dan BDR ini tidak akan berubah sampai salah satunya tidak dapat berfungsi baik, meskipun ada perute lain yang baru bergabung dalam jaringan dengan nilai prioritas yang lebih tinggi.

Pada dasarnya, semua perute OSPF akan memiliki nilai prioritas 1. Kisaran nilai priority ini adalah mulai dari 0 hingga 255. Nilai 0 akan menjamin perute tersebut tidak akan menjadi DR atau BDR, sedangkan nilai 255 menjamin sebuah perute pasti akan menjadi DR. ID Perute biasanya akan menjadi penentu jika nilai prioritasnya sama. Jika dua buah perute memiliki nilai prioritas yang sama, maka yang menjadi DR adalah perute dengan nilai ID perute tertinggi dalam jaringan. Setelah DR dan BDR terpilih, langkah selanjutnya adalah mengumpulkan seluruh informasi jalur dalam jaringan.

3.3.3 Mengumpulkan semua keadaan (*state*) jaringan

Setelah terbentuk hubungan antarperute OSPF, kini saatnya untuk bertukar informasi mengenai keadaan (*state*) dan jalur-jalur yang ada dalam jaringan. Pada jaringan yang menggunakan media *broadcast multiaccess*, DR-lah yang akan melayani setiap perute yang ingin bertukar informasi OSPF dengannya. DR akan memulai lebih dulu proses pengiriman ini.

Pada jaringan *point-to-point* ada sebuah fasa yang menentukan perute yang lebih dulu melakukan pengiriman data *link-state* OSPF. Fasa ini akan memilih siapa yang akan menjadi tuan (*master*) dan siapa yang menjadi budak (*slave*) dalam proses pengiriman. Perute yang menjadi master akan melakukan pengiriman lebih dahulu, sedangkan perute *slave* akan mendengarkan lebih dulu. Fasa ini disebut dengan istilah *exstart state*. Perute master dan slave dipilih berdasarkan ID perute tertinggi dari salah satu router. Ketika sebuah perute mengirimkan paket Hello, ID perute masing-masing juga dikirimkan ke perute tetangga.

Setelah membandingkan dengan miliknya dan ternyata lebih rendah, maka perute tersebut akan segera terpilih menjadi *master* dan melakukan

pengiriman lebih dulu ke perute *slave*. Setelah fasa *exstart* lewat, maka perute akan memasuki fasa *exchange*. Pada fasa ini kedua buah perute akan saling mengirimkan paket deskripsi basis data (*database description packet*). Isi paket ini adalah ringkasan status untuk seluruh media yang ada dalam jaringan. Jika perute penerimanya belum memiliki informasi yang ada dalam paket deskripsi basis data, maka perute pengirim akan masuk dalam fasa *loading state*. Fasa *loading state* merupakan fasa di mana sebuah perute mulai mengirimkan informasi keadaan (*state*) secara lengkap ke perute tetangganya.

Setelah *loading state* selesai, maka perute-perute yang tergabung dalam OSPF akan memiliki informasi keadaan yang lengkap dan penuh dalam basis data keadaannya. Fasa ini disebut dengan istilah *full state*. Sampai dengan fasa ini, proses awal OSPF sudah selesai. Namun basis data keadaan (*state database*) belum bisa digunakan untuk proses penerusan (*forwarding*) data. Maka dari itu, perute akan memasuki langkah selanjutnya, yaitu memilih rute-rute terbaik menuju ke suatu lokasi yang ada dalam basis data keadaan tersebut.

3.3.4 Memilih rute terbaik untuk digunakan

Setelah informasi seluruh jaringan berada dalam basis data, maka kini saatnya untuk memilih rute terbaik untuk dimasukkan ke dalam tabel perutean. Jika sebuah rute telah masuk ke dalam tabel perutean, maka rute tersebut akan terus digunakan. Untuk memilih rute-rute terbaik, parameter yang digunakan oleh OSPF adalah beban (*cost*). Metrik *cost* biasanya akan menggambarkan seberapa dekat dan cepat sebuah rute. Nilai *cost* didapat dari perhitungan dengan rumus:

$$Cost = \frac{Cost\ of\ the\ link}{Bandwidth}$$

Perute OSPF akan menghitung semua *cost* yang ada dan akan menjalankan algoritma Dijkstra (*shortest path first*) – yang akan dijelaskan lebih lanjut pada bahasan berikutnya – untuk memilih rute terbaiknya. Setelah selesai, maka rute tersebut langsung dimasukkan dalam tabel perutean dan siap digunakan untuk penerusan (*forwarding*) paket data.

3.3.5 Menjaga kemutakhiran informasi perutean

Ketika sebuah rute sudah masuk ke dalam tabel perutean, perute tersebut harus juga menjaga kondisi (*state*) basis datanya. Hal ini bertujuan kalau ada sebuah rute yang sudah tidak valid, maka perute harus tahu dan tidak boleh lagi menggunakannya. Ketika ada perubahan *link-state* dalam jaringan, perute OSPF akan melakukan pembanjiran (*flooding*) terhadap perubahan ini. Tujuannya adalah agar seluruh perute dalam jaringan mengetahui perubahan tersebut. Sampai di sini semua proses OSPF akan terus berulang. Mekanisme seperti ini membuat informasi rute-rute yang ada dalam jaringan terdistribusi dengan baik, terpilih dengan baik, dan dapat digunakan dengan baik pula. Selain itu dengan konsep hierarki, administrator dapat membatasi ukuran basis data *link-state*-nya, sehingga menjadi tidak terlalu besar. Dengan demikian, proses CPU juga bisa menjadi lebih ringan.

4. Algoritma Dijkstra

Algoritma Dijkstra, yang ditemukan oleh pakar komputer asal Belanda, Edsger Dijkstra, adalah algoritma *greedy* yang menyelesaikan permasalahan mencari jarak terpendek dari suatu graf berarah dengan bobot sisi yang tidak negatif. Contoh analogi: jika simpul pada graf melambangkan suatu entitas komputer dalam jaringan dan bobot sisi melambangkan besaran beban (*metric cost*) antarentitas, algoritma Dijkstra dapat dipakai untuk mencari rute terbaik antara dua komputer dalam jaringan komputer.

4.1 Penjelasan Algoritma

Persoalan yang diselesaikan dengan memanfaatkan algoritma Dijkstra adalah persoalan menentukan lintasan terpedek pada sebuah graf berbobot G (V,E), di mana V adalah simpul, dan E adalah sisi dari graf, serta bobot pada E selalu positif.

Ada beberapa kasus pencarian lintasan terpendek yang diselesaikan menggunakan algoritma Dijkstra, yaitu: pencarian lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*), pencarian lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*), pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*), serta pencarian lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*). Untuk kasus pemutakhiran tabel perutean, persoalan pencarian lintasan terpendek yang diselesaikan adalah *singe-source shortest path*.

Penggunaan strategi *greedy* pada algoritma Dijkstra adalah:
Pada setiap langkah, ambil sisi berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain

yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek di antara semua lintasannya ke simpul-simpul yang belum terpilih.

4.2 Pseudokode

```
procedure Dijkstra(INPUT m: matriks, a : simpul awal)
{
    Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya.
    Masukan: matriks ketetanggaan (m) dari graf berbobot G dan simpul awal a
    Keluaran: lintasan terpendek dari a ke semua simpul lainnya
}

Kamus:
   s : array [1..n] of integer
   d : array [1..n] of integer
   i : integer

Algoritma:
     { Langkah 0 (inisialisasi: }
     traversal [1..n]
        s_i ← 0
        d_i ← m_ai

     { Langkah 1: }
     s_a ← 1
     d_a ← ∞

     { Langkah 2, 3, ..., n-1: }
     traversal [2..n-1]
        cari j  sedemikian sehingga s_j= 0 dan
        d_j = min {d1, d2, ..., dn}
        s_j ← 1 {simpul j sudah terpilih}
        perbarui d_i, untuk i = 1, 2, 3, s.d. n dengan:
           d_i(baru) = min{d_i(lama),d_j+ m_ji}
```

Kompleksitas algoritma Dijkstra adalah $O(n^2)$. Sehingga, untuk mencari semua pasangan simpul terpendek, total waktu asimptotik komputasinya adalah: $T(n) = n.O(n^2) = O(n^3)$.

4.3 Contoh Penelusuran Rute Terpendek

Di bawah ini terdapat beberapa *screenshot*, yaitu tabel perutean pada mesin *gateway* unit kegiatan mahasiswa ARC, dan beberapa contoh *tracerouting* (penelusuran rute terpendek) dari salah satu *host* di ARC ke beberapa tujuan populer seperti Google, dan Friendster, serta students.if.itb.ac.id.

```
Routing tables

Internet:
Destination        Gateway            Flags     Refs      Use   Netif Expire
default            167.205.23.3       UGS          0  8783859     rl0
66.90.92.104       167.205.23.3       UGH1         0        0     rl0
66.219.97.124      167.205.23.3       UGH1         0        0     rl0
81.233.243.81      167.205.23.3       UGH1         0        0     rl0
87.230.6.83        167.205.23.3       UGH1         0        0     rl0
127.0.0.1          127.0.0.1          UH           0      129     lo0
131.113            167.205.23.16      UG1          0        4     rl0
167.205            167.205.23.3       UG1          0  6795780     rl0
167.205.1/27       167.205.23.3       UG1          0      820     rl0
167.205.1.64/28    167.205.23.3       UG1          0    16330     rl0
167.205.1.74       167.205.23.3       UGH1         0       80     rl0
167.205.1.112/29   167.205.23.3       UG1          0      634     rl0
167.205.1.128/26   167.205.23.8       UG1          0  1056648     rl0
167.205.1.224/27   167.205.23.3       UG1          0     1061     rl0
167.205.3.0        ff:ff:ff:ff:ff:ff  UHLWb        1     1572     rl1 =>
167.205.3/26       link#2             UC           0        0     rl1
167.205.3.1        00:15:e9:d9:e6:42  UHLW         1    11294     lo0
167.205.3.2        00:15:e9:6a:56:67  UHLW         1  3789968     rl1     694
167.205.3.3        00:80:48:2d:a5:e8  UHLW         1  4648163     rl1       5
167.205.3.4        00:13:46:3b:ff:b0  UHLW         1 74641478     rl1     765
167.205.3.7        00:17:31:54:af:19  UHLW         1  1515222     rl1     954
167.205.3.9        00:16:36:86:0e:cb  UHLW         1     1932     rl1    1190
167.205.3.13       00:00:39:34:12:9f  UHLW         1    32918     rl1    1129
167.205.3.14       00:0c:61:00:00:00  UHLW         1  1727154     rl1     111
167.205.3.16       00:00:39:13:91:3f  UHLW         1    15375     rl1      30
167.205.3.27       00:10:5c:af:f8:dd  UHLW         1   966729     rl1     693
167.205.3.31       00:00:39:9b:f0:d1  UHLW         1    22032     rl1    1180
167.205.3.46       00:17:f2:2a:08:b6  UHLW         1   572374     rl1     449
167.205.3.60       00:0b:82:02:c5:79  UHLW         1     1652     rl1      57
167.205.3.63       ff:ff:ff:ff:ff:ff  UHLWb        1      108     rl1
167.205.3.64       ff:ff:ff:ff:ff:ff  UHLWb        1       92     rl2 =>
167.205.3.64/27    link#3             UC           0        0     rl2
167.205.3.72       00:11:95:63:4a:da  UHLW         1       39     rl2    1168
167.205.3.73       00:30:18:ac:37:3d  UHLW         1     7676     rl2     782
167.205.3.74       00:11:5b:a0:2d:d1  UHLW         1    43013     rl2    1066
167.205.3.75       00:a0:c9:4d:36:8e  UHLW         1    95727     rl2    1035
167.205.3.76       00:0c:6e:66:b4:df  UHLW         1     7547     rl2    1171
167.205.3.77       00:08:a1:33:38:98  UHLW         1    20349     rl2    1052
167.205.3.85       00:90:f5:51:0e:34  UHLW         1   377368     rl2    1037
:
```

Gambar 8 *Screenshot* Tabel Perutean (*Routing Tables*) mesin *gateway* ARC (167.205.3.1)

```
[admin@gtw] /home/admin# traceroute www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 66.102.7.99
traceroute to www.l.google.com (66.102.7.99), 64 hops max, 40 byte packets
 1  AI3-NOC-PAU-Cat6-Vlan.ITB.ac.id (167.205.23.3)  1.011 ms  0.653 ms  0.587 ms
 2  PAU-Cat6-FE4-1.ITB.ac.id (167.205.23.2)  0.943 ms  0.656 ms  0.431 ms
 3  gerbang.itb.ac.id (202.249.24.66)  0.621 ms  1.049 ms  0.662 ms
 4  sfc-sat2.ai3.net (202.249.25.244)  514.025 ms  523.706 ms  527.037 ms
 5  sfc-orochi.ai3.net (202.249.25.225)  512.950 ms  536.872 ms  557.961 ms
 6  sfc-gate.ai3.net (202.249.25.1)  522.219 ms  516.493 ms  512.498 ms
 7  gsr1.fujisawa.wide.ad.jp (202.249.26.114)  611.416 ms  538.525 ms  534.841 ms
 8  ve-100.foundry1.fujisawa.wide.ad.jp (203.178.137.91)  513.325 ms  566.279 ms  542.256 ms
 9  ve-4.cisco2.notemachi.wide.ad.jp (203.178.138.243)  514.227 ms  529.785 ms  598.674 ms
10  61.213.145.93 (61.213.145.93)  556.604 ms  524.030 ms  538.070 ms
11  61.213.162.87 (61.213.162.87)  516.180 ms
    61.200.92.119 (61.200.92.119)  532.720 ms  533.396 ms
12  xe-1-0-0.r20.tokyjp01.jp.bb.gin.ntt.net (61.213.162.229)  520.273 ms  514.558 ms  515.046 ms
13  p64-1-3-0.r20.sttlwa01.us.bb.gin.ntt.net (129.250.4.157)  632.275 ms  632.739 ms  653.716 ms
14  p16-6-0-0.r04.sttlwa01.us.bb.gin.ntt.net (129.250.2.69)  650.880 ms  637.913 ms  646.754 ms
15  so-3-3-1.cr1.sea1.us.above.net (64.125.12.29)  636.252 ms  658.985 ms  684.389 ms
16  so-3-2-0.mpr3.sjc2.us.above.net (64.125.28.182)  661.857 ms  635.313 ms  647.087 ms
17  so-0-0-0.mpr2.sjc2.us.above.net (64.125.27.246)  669.280 ms  642.154 ms  659.655 ms
18  so-4-0-0.mpr4.pao1.us.above.net (64.125.29.162)  654.402 ms  671.088 ms  633.711 ms
19  64.124.76.162.available (64.124.76.162)  762.324 ms  633.815 ms  633.888 ms
20  66.249.94.12 (66.249.94.12)  666.814 ms  696.680 ms  739.726 ms
21  72.14.233.129 (72.14.233.129)  728.473 ms  680.738 ms  637.389 ms
22  216.239.49.54 (216.239.49.54)  639.909 ms  660.032 ms
    72.14.233.129 (72.14.233.129)  671.134 ms
23  216.239.49.54 (216.239.49.54)  712.814 ms  678.172 ms *
24  mc-in-f99.google.com (66.102.7.99)  636.288 ms  636.851 ms  710.837 ms

[admin@gtw] /home/admin#
```

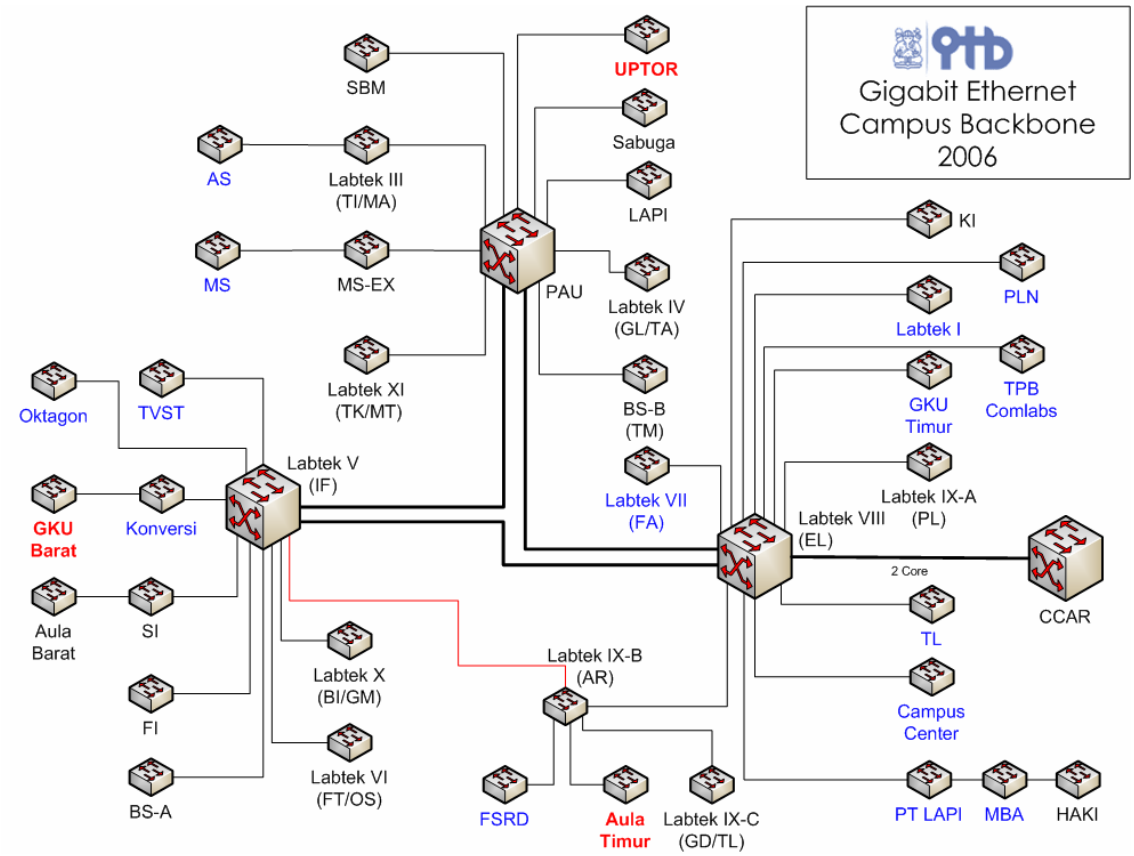
Gambar 9 *Screenshot* Penelusuran Rute Terpendek dari salah satu alamat IP di jaringan ARC (167.205.3.31) ke salah satu alamat IP Google (66.102.7.99)

```
[admin@gtw] /home/admin# traceroute www.friendster.com
traceroute: Warning: www.friendster.com has multiple addresses; using 209.11.168.201
traceroute to www.friendster.com (209.11.168.201), 64 hops max, 40 byte packets
 1  AI3-NOC-PAU-Cat6-Vlan.ITB.ac.id (167.205.23.3)  0.969 ms  0.738 ms  0.628 ms
 2  PAU-Cat6-FE4-1.ITB.ac.id (167.205.23.2)  0.605 ms  0.647 ms  0.431 ms
 3  gerbang.itb.ac.id (202.249.24.66)  0.612 ms  0.632 ms  0.933 ms
 4  sfc-sat2.ai3.net (202.249.25.244)  524.732 ms  516.079 ms  542.851 ms
 5  sfc-orochi.ai3.net (202.249.25.225)  530.171 ms  517.492 ms  580.425 ms
 6  sfc-gate.ai3.net (202.249.25.1)  532.418 ms  513.006 ms  540.754 ms
 7  gsr1.fujisawa.wide.ad.jp (202.249.26.114)  555.208 ms  532.129 ms  527.279 ms
 8  ve-100.foundry1.fujisawa.wide.ad.jp (203.178.137.91)  556.703 ms  511.146 ms  527.275 ms
 9  ve-4.cisco2.notemachi.wide.ad.jp (203.178.138.243)  526.361 ms  704.494 ms  615.955 ms
10  fa-2-3-3.a13.tokyjp01.jp.ra.gin.ntt.net (61.213.145.93)  550.560 ms  554.646 ms  532.024 ms
11  ge-7-1-3.a20.tokyjp01.jp.ra.gin.ntt.net (61.200.92.54)  525.818 ms
    ge-7-0-1.a20.tokyjp01.jp.ra.gin.ntt.net (61.213.162.54)  543.022 ms  549.813 ms
12  xe-1-0-0.r21.tokyjp01.jp.bb.gin.ntt.net (61.213.162.233)  541.708 ms  551.545 ms
    xe-2-0-0.r21.tokyjp01.jp.bb.gin.ntt.net (61.213.162.105)  517.947 ms
13  p64-2-1-0.r20.snjsca04.us.bb.gin.ntt.net (129.250.2.141)  672.515 ms  660.839 ms  653.622 ms
14  xe-1-4.r02.snjsca04.us.bb.gin.ntt.net (129.250.2.29)  653.130 ms  653.388 ms  635.623 ms
15  xe-0.internap.snjsca04.us.bb.gin.ntt.net (129.250.12.82)  631.160 ms  643.530 ms  650.700 ms
16  border2.pc2-0-bbnet2.sje.pnap.net (66.151.144.69)  652.434 ms  632.840 ms  664.266 ms
17  * friendster-4a.border2.sje.pnap.net (66.151.139.42)  677.702 ms  695.661 ms
18  www-nsin2.friendster.com (209.11.168.201)  701.249 ms  679.820 ms  633.765 ms

[admin@gtw] /home/admin#
```

Gambar 10 *Screenshot* Penelusuran Rute Terpendek dari salah satu alamat IP di jaringan ARC (167.205.3.31) ke salah satu alamat IP Friendster (209.11.168.201)

```
[admin@gtw] /home/admin# traceroute students.if.itb.ac.id
traceroute to students.if.itb.ac.id (167.205.32.4), 64 hops max, 40 byte packets
 1  AI3-NOC-PAU-Cat6-Vlan.ITB.ac.id (167.205.23.3)  0.823 ms  0.861 ms  0.887 ms
 2  LabtekV-Cat6-GEC-3.ITB.ac.id (167.205.62.6)  2.704 ms  0.758 ms  0.849 ms
 3  paspati.if.itb.ac.id (167.205.30.90)  1.245 ms  0.810 ms  0.661 ms
 4  students.if.itb.ac.id (167.205.32.4)  1.105 ms  1.068 ms  0.930 ms

[admin@gtw] /home/admin#
```

Gambar 11 *Screenshot* Penelusuran Rute Terpendek dari salah satu alamat IP di jaringan ARC (167.205.3.31) ke students.if.itb.ac.id (167.205.32.4)



Gambar 12 Peta Jaringan Internet ITB 2006

Dari Gambar 12, terlihat bahwa terdapat 4 perute utama ITB, yaitu perute Pusat Antaruniversitas (PAU) – yang merupakan gerbang komunikasi ITB, karena berhubungan langsung dengan para pemasok *bandwith* ITB –, perute Labtek V, perute Labtek VIII, serta perute CCAR (Rektorat). Perute utama tersebut terhubung dengan berbagai perute-perute dalam lingkup yang lebih kecil (misalnya perute Labtek X, Labtek VI, BS-A, Aula Barat, dsb.) yang relasinya bisa dilihat pada Gambar 12.

Misalkan, terdapat suatu *host* di jaringan unit kegitan mahasiswa ARC (167.205.3.31) yang akan berkomunikasi dan bertukar informasi dengan salah satu host di jaringan Informatika (167.205.32.4). Langkah-langkah yang dilakukan oleh *host* di jaringan ARC adalah:

- Host akan berkomunikasi dengan mesin gateway jaringan ARC (167.205.3.1)
- Mesin *gateway* akan berkomunikasi dengan Catalyst-6 PAU (167.205.23.3) yang masuk dalam perute PAU
- Perute PAU akan mencari rute terbaik ke jaringan Informatika dalam tabel peruteannya. Berdasarkan tabel perutean di perute PAU, perute selanjutnya ke jaringan IF adalah LabtekV-Cat6-GEC-3.ITB.ac.id (167.205.62.6)
- Perute LabtekV-Cat6-GEC-3.ITB.ac.id akan meneruskan paket data ke perute paspati.if.itb.ac.id (167.205.30.90)
- Perute paspati.if.itb.ac.id (167.205.30.90) selanjutnya langsung menyampaikan paket data ke tujuan (167.205.32.4)

Langkah-langkah penelusuran rute tersebut bisa dilihat dengan perintah **traceroute** setelah masuk log sebagai administrator pada mesin *gateway* ARC. Penelusuran rute untuk berbagai tujuan populer (seperti Google dan Friendster), bisa dilihat pada Gambar 9 dan Gambar 10. Alamat nomor 1 s.d. 23 pada Gambar 9, serta alamat nomor 1 s.d. 17 pada Gambar 10, adalah perute-perute yang harus dilalui oleh paket data agar sampai ke alamat tujuan yang dimaksud. Rute yang dilalui sudah terpilih melalui algoritma perutean yang dilakukan oleh masing-masing perute tadi. Di sinilah peran algoritma Dijkstra dalam penentuan rute tersebut, karena jaringan ITB, Google, maupun Friendster termasuk dalam golongan jaringan besar – ada sekitar 4000 komputer klien dalam jaringan komputer ITB – yang tentunya memakai OSPF sebagai protokol peruteannya.

5. Kesimpulan

Kesimpulan yang bisa diambil dari studi implementasi algoritma Dijkstra pada protokol perutean *Open Shortest Path First* (OSPF) adalah:

1. Protokol OSPF sangat cocok diterapkan pada jaringan yang memiliki jumlah *host* yang sangat banyak, karena kemampuannya dalam mengelola dan memperbarui tabel perutean untuk jaringan tersebut. Protokol OSPF pun juga cepat dalam penyesuaian terhadap perubahan topologi jaringan.

2. Dibandingkan dengan protokol RIP yang sama-sama populer dan sama-sama termasuk dalam kelompok Interior Gateway Protocol (IGP), protokol OSPF memang memiliki mekanisme lebih rumit dan kemungkinan besar memakan memori dan waktu yang lebih besar, namun sangat handal dalam penanganan jaringan dengan jumlah *host* yang sangat banyak. Sementara itu RIP meski sederhana dan lebih mudah penerapannya, namun kurang handal dalam penanganan jaringan besar dan kemungkinan besar menimbulkan kalang perutean (*routing loop*), meski RIP memiliki mekanisme sendiri untuk mencegahnya.

3. Algoritma Dijkstra – yang pada dasarnya dimanfaatkan untuk mencari jalur terpendek di antara dua simpul dalam graf berbobot tak negatif – digunakan untuk menentukan rute terbaik pada protokol OSPF. Algoritma Dijkstra akan menghitung rute dengan nilai beban terkecil (*minimum metric cost*) yang nantinya akan disimpan ke dalam basis data perutean.

DAFTAR PUSTAKA

[1] Anton Ellyas Priyambodo's Weblog. *http://antzon.wordpress.com/2006/02/28/ospf-routing-protokol-untuk-jaringan-lokal-part-1/*. Tanggal akses: 1 Januari 2007 pukul 14.47

[2] Munir, Rinaldi. (2003). *Bahan Kuliah IF2151 Matematika Diskrit*. Departemen Teknik Informatika, Institut Teknologi Bandung.

[3] Munir, Rinaldi. (2004). *Bahan Kuliah IF2251 Strategi Algoritmik*. Departemen Teknik Informatika, Institut Teknologi Bandung.

[4] OSI 7 Layer Model Tutorial (2006). *http://www.pcsupportadvisor.com/OSI_7_layer_model_page1.htm*. Tanggal akses: 30 Desember 2006 pukul 17.19

[5] White, Russ, et al. (2005). *Optimal Routing Design*. Cisco Press, Indianapolis, USA.

[6] Wikipedia. (2006). *http://en.wikipedia.org/wiki/Routing_protocols*. Tanggal akses: 30 Desember 2006 pukul 17.18

[7] Wikipedia. (2006). *http://en.wikipedia.org/wiki/OSI_model*. Tanggal akses: 30 Desember 2006 pukul 17.18

[8] Wikipedia. (2006). *http://en.wikipedia.org/wiki/Routing*. Tanggal akses: 30 Desember 2006 pukul 18.10

[9] Wikipedia (2006). *http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm*. Tanggal akses: 1 Januari 2007 pukul 14:16

[10] Wikipedia (2006). *http://en.wikipedia.org/wiki/Bellman-Ford_algorithm*. Tanggal akses: 1 Januari 2007 pukul 14:28

[11] Wikipedia (2006). *http://en.wikipedia.org/wiki/Open_shortest_path_first*. Tanggal akses: 1 Januari 2007 pukul 14:49