PENANGANAN MASALAH COUNT-TO-INFINITY PADA ROUTING JARINGAN YANG MENERAPKAN ALGORITMA BELLMAN-FORD

Bramianha Adiwazsha

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganeca 10 Bandung, Indonesia
e-mail: if17106@students.if.itb.ac.id

ABSTRAK

Dalam konteks jaringan komputer, *routing* adalah proses untuk meneruskan paket data dari suatu jaringan ke jaringan lain yang dituju dan sebaliknya. Untuk menentukan rute terpendek pada saat *routing*, cara yang paling umum digunakan adalah dengan menggunakan algoritma Bellman-Ford secara rekursif untuk menentukan jalur router mana yang harus dipilih untuk mencapai jaringan tujuan dengan cost paling kecil. Jumlah router yang ada dalam jaringan dapat berjumlah sangat besar sehingga menyimpan informasi semua router dalam tiap table akan sangat tidak efisien dilihat dari sudut pandang pemakaian memori, sehingga algoritma Bellman-Ford yang dipakai menerapkan prinsip program dinamis yang dapat menyimpan status cost routing dalam tiap langkah yang diambil sehingga dapat menghemat pemakaian memori. Namun dalam proses routing menggunakan algoritma Bellman-Ford ini memiliki kelemahan yaitu mungkin terjadi kesalahan count-to infinity yang dapat timbul saat suatu router yang berada dalam jalur routing mati dan keadaan ini tidak dapat dideteksi oleh algoritma Bellman-Ford. Pada makalah ini akan dibahas mengenai penerapan algoritma Bellman-Ford dalam proses *routing* dan penanganan yang dapat diambil untuk mengatasi masalah count-to-infinity yang mungkin timbul.

Kata kunci: Routing, algoritma Bellman-Ford, count-to-infinity, program dinamis.

1. PENDAHULUAN

Ada dua kelompok algoritma yang dapat digunakan untuk menentukan jalur routing pada jaringan yaitu algoritma *link state* dan algoritma *distance vector*.

Pada *link state*, seluruh router yang berada dalam jaringan harus sudah diketahui terlebih dahulu keterhubungan maupun beban/cost pada jalur yang menghubungkan tiap router. Kemudian di masing-masing router akan dihitung router mana saja yang dapat dicapai oleh router tersebut dan jalur mana yang harus diambil untuk mendapatkan rute terpendek. Informasi jalur yang dada tersebut kemudian disimpan di tiap router di jaringan. Namun cara ini kurang efisien karena jumlah router yang terhubung saat ini umumnya sangat besar sehingga menyimpan informasi setiap router yang ada akan memerlukan ruang memori yang sangat besar dan waktu yang diperlukan untuk mengkakulasikan jalur setiap kali ada penambahan router baru terlalu besar.

Pada *distance vector*, router tidak perlu memelihara informasi mengenai seluruh router yang ada. Namun hanya cukup informasi router yang menjadi tetangganya saja (terhubung secara langsung tanpa perantara router lain). Dengan cara ini informasi yang harus disimpan tidaklah terlalu besar, dan untuk mendapatkan jalur dengan bobot terkecil digunakan algoritma Bellman-Ford yang mendapatkan informasi bobot jalur terkecil dengan cara rekursif ke router berikutnya. Hal ni dapat dicapai dengan menyimpan informasi jalur terpendek yang dapat dicapai router.

2. BELLMAN-FORD PADA ROUTING

2.1 Bellman-Ford yang Terdistribusi

Algoritma Bellman-Ford yang digunakan secara tersdistribusi adalah berdasar pada ide bahwa jika suatu node router B terdapat dalam jalur terdekat antara node A dan C, maka jalur dari node A ke node B pasti juga merupakan jalur terdekat. Demikian pula jalur dari node C ke node B.

Penggunaan algoritmanya adalah sebagai berikut:
Didefinisikan:

$d_x(y)$ = jarak terdekat antara node x ke node y.
$c(x,y)$ = jarak antara node x ke node y.

Inisialisasi:

$d_x(v) = \infty$, untuk semua $v \neq x$

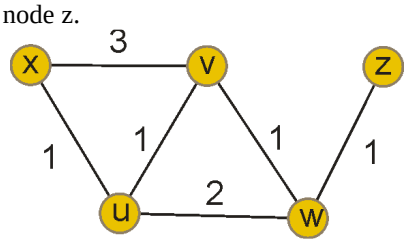
$$d_x(x) = 0$$

Update nilai:
untuk dapat mencapai node z dari x

$$d_x(z) = \min \{c(x,v) + d_v(z) \}$$

Dimana v merupakan semua node tetangga dari node x.

Contoh:
Pada jaringan contoh_1 dicari jalur terdekat dari node router x ke node z.



Gambar 1. Topologi jaringan contoh_1

Kita ketahui bahwa $d_u(z) = 3$ dan $d_v(z) = 2$

Jalur terdekat dari x ke z:

$$d_x(z) = \min \{c(x,u) + d_u(z) , c(x,v) + d_v(z)\}$$
$$= \min \{1 + 3 , 3 + 2\}$$
$$= \min \{4 , 5\}$$

Maka didapatlah bobot jalur terdekat dari node x ke z adalah 4 dan router berikutnya yang harus diambil dari node x adalah node u.

Dari contoh di atas terlihat bahwa node x memelihara bobot jalur minimum ke node z yang di dalamnya sekaligus memelihara bobot jalur minimum dari node u dan v (yang merupakan tetangga dari node x) ke node z. Dalam hal ini algoritma Bellman-Ford yang digunakan didefinisikan secara rekursif.

2.2 Program Dinamis pada *Distance Vector Protocol*

Seperti yang sudah disebutkan sebelumnya, *distance vector* menggunakan algoritma Bellman-Ford seperti di atas untuk menghitung jarak terdekat. Hal ini dilakukan secara terdisribusi, dan setiap node menghitung semua jalur ke setiap node lainnya.

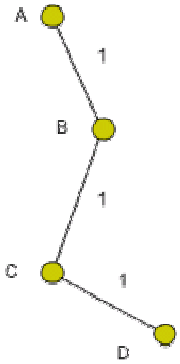
Dengan langkah ini, secara intuitif algoritma membandingkan jarak yang didapatkan dari node lain dan mengupdate data jalur terpendek dalam routing tabel masing-masing node. Informasi yang dipertahankan dalam routing tabel mencakup 3 hal: node yang dapat dituju, node terdekat berikutnya untuk mencapai node tujuan tersebut, dan bobot untuk mencapainya. Dimana informasi dari perhitungan awal dipertahankan untuk dapat digunakan dalam langkah berikutnya.

Dalam penerapan di dunia nyata, acuan yang digunakan untuk menjari jalur terdekat adalah suatu parameter yang tidak selalu jarak. Namun juga bisa menggunakan kapasitas lebar jalur, delay pengiriman, maupun kecepatan transfer.

2.3 Masalah Count-to-Infinity

Namun begitu, penggunaan algoritma Bellman-Ford ini dapat gagal mendapatkan hasil berupa jalur dengan bobot terkecil. Salah satunya adalah jika algoritma mendapati keadaan *count-to-infinity*.

Berikut adalah ilustrasi masalah *count-to-infinity* pada jaringan contoh_2



Gambar 2. Keadaan awal jaringan contoh_2

Dari topologi jaringan di atas node akan memiliki routing tabel sebagai berikut

Node	Node tujuan	Bobot terpendek
A	B	1
	C	2
	D	3

Node	Node tujuan	Bobot terpendek
B	A	1
	C	1
	D	2

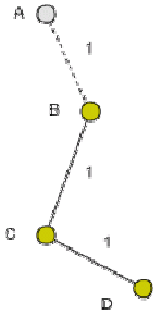
Node	Node tujuan	Bobot terpendek
C	A	2
	B	1
	D	1

Node	Node tujuan	Bobot terpendek
D	A	3
	B	2
	C	1

Tabel 1 Routing tabel contoh_2 pada keadaan awal

Apabila suatu ketika jalur antara node A dan B putus, maka routing tabel pada node akan diupdate. Node B sadar bahwa jalur dari B ke A sudah putus, namun B melihat bahwa node C dapat mencapai node A dalam 2 kali hop. Sehingga node B merasa dapat mencapai node A melalui C dan kemudian mengbah bobot untuk menuju A menjadi 3 (hasil dari $c(B,C) + d_c(A)$).

Tidak ada cara bagi node B untuk tahu bahwa sebenarnya node C dapat mencapai A juga harus melalui B. Selanjutnya node C melihat bahwa jarak dari node B ke A telah berubah, dan mengganti isi routing tabel untuk mencapai A menjadi 4.



Gambar 3. Jalur A-B pada contoh_2 putus

Bila hal di atas terus berlanjut maka pada akhirnya kondisi penambahan bobot akan berulang menuju tak hingga (*count-to-infinity*) dan akhirnya mematikan laju seluruh jaringan. Karena bobot terus dihitung ulang tak terbatas dan menghabiskan memori router.

Dan routing tabel akan menjadi seperti sebagai berikut

Node	Node tujuan	Bobot terpendek
A	B	∞
	C	∞
	D	∞

Node	Node tujuan	Bobot terpendek
B	A	Terus bertambah
	C	1
	D	2

Node	Node tujuan	Bobot terpendek
C	A	Terus bertambah
	B	1
	D	1

Node	Node tujuan	Bobot terpendek
D	A	Terus bertambah
	B	2
	C	1

Tabel 2 Routing tabel contoh_2 pada *count-to-infinity*

3. PENANGANAN MASALAH COUNT-TO-INFINITY

Masalah count-to-infinity yang disebabkan oleh penggunaan algoritma Bellman-Ford terdistribusi ini dapat diselesaikan dengan beberapa metode tambahan yang disertakan dalam pemakaian algoritma.

3.1 Menentukan maksimum distribusi penghitungan

Cara ini membatasi sejauh/ sedalam apakah proses rekursi penghitungan bobot jalur node dilakukan. Dengan menentukan nilai maksimum node terjauh yang dihitung, kita dapat mengurangi kemungkinan terjadinya count-to-infinity yang berpengaruh ke seluruh jaringan. Namun hal ini berarti membatasi ukuran jaringan yang dapat ditangani oleh algoritma kita.

Semisal kita menentukan jumlah maksimum penghitungan distribusi hingga hop ke 16 (seperti standar yang digunakan dalam metode *Routing Information Protocol*) maka algoritma routing yang kita gunakan tidak dapat mencapai sub jaringan yang terpisah lebih dari 15.

3.2 Split Horizon

Metode ini berarti bahwa jika node A mengetahui bahwa rute menuju C adalah melalui node B, maka node A tidak perlu mengirim informasi *distance vector* node C kepada node B saat melakukan update routing.

Jadi tidak semua informasi dikirimkan, yang dikirimkan ke node B hanyalah informasi bobot node yang bisa dicapai node A yang tidak melalui B.

3.3 Poisoned Reverse

Metode ini adalah teknik tambahan yang digunakan bersama split horizon. Metode split horizon dimodifikasi sehingga menjadi bukan tidak mengirimkan informasi, namun informasi mengenai bobot node tersebut diubah menjadi infinite (∞).

Split horizon yang menggunakan poisoned reverse mencegah terjadinya loop router yang hanya melibatkan dua buah node saja. Tapi loop yang terjadi pada node lebih dari dua masih mungkin terjadi.

3.4 Update dengan pemicu

Saat node menerima update, node tersebut menghitung kembali jalur terpendek. Jika hasil perhitungan menunjukkan perubahan terhadap routing tabel mengenai ute terpendek, maka update dikirimkan ke node lain.

3.5 Penghitung waktu

Penghitung waktu diaktifkan saat:

- Batas waktu untuk suatu rute telah kadaluwarsa
- Rute yang tadinya dapat dicapai berubah menjadi tidak dapat dicapai

Saat hitungan waktu mundur mencapai 0, rute tersebut dihilangkan dari data dan dianggap mati.

IV. KESIMPULAN

Telah kita lihat bahwa penggunaan algoritma yang menerapkan metode pemrograman dinamis dapat diterapkan untuk memecahkan masalah pemilihan jalur terpendek pada routing. Namun algoritma Bellman-Ford yang digunakan ternyata masih mempunyai kelemahan yaitu masalah *count-to-infinity*.

Namun masalah tersebut dapat diatasi dengan melakukan sedikit penambahan pada proses penghitungan yang dilakukan, dalam hal ini terutama mengenai informasi yang digunakan ntuk perhitungan jalur terpendek (isi routing tabel).

REFERENSI

[1] Luntovsky, Count to Infinity problem http://wiki.uni.lu/secan-lab/Count-To-nfinity+Problem.html Diakses tanggal 04 Januari 2010 pukul 19.00 WIB

[2] Wikipedia, Bellman-Ford algorithm http://en.wikipedia.org/wiki/Bellman_Ford Diakses tanggal 05 januari 2010 pukul 15.00 WIB

[3] Christian Huitema. *Routing in the Internet*, 2nd Edition. Prentice Hall, Upper Saddle River, 2000. ISBN 0-13-022647-5

[4] J.M. Jaffe and F. Moss. *A Responsive Distributed Routing Algorithm for Computer Networks*. IEEE Transactions on Communications COM-30:1758-1762, July 1982.

[5] Rinaldi, Munir, *Diktat Kuliah IF2251Strategi Agoritmik,* Program Studi Teknik Informatika ITB, 2005.