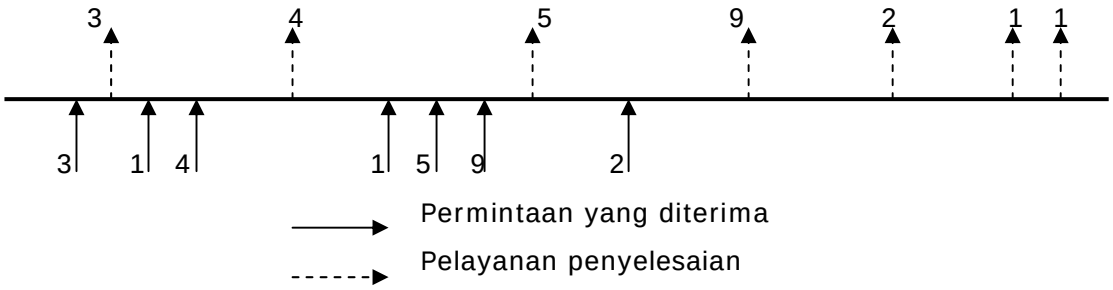
Nama : Yulius Nugroho Adi S
NIM : 23204112
UTS Tanggal 23 Agustus 2005
Remidi Tanggal 25 Agustus 2005

(I)5.6 PEMBATALAN TRANSAKSI (Menggagalkan Transaksi)

Ada dua macam pembatalan transaksi, termination dan nontermination. Suatu transaksi yang mendapat suatu pembatalan penghentian tidak dimulai lagi setelah dibatalkankan. Ini dapat terjadi untuk beberapa pertimbangan. Sebagai contoh, pemakai boleh memutuskan untuk membatalkan suatu transaksi. Misalkan ada suatu kesalahan operasi aritmatic (e.g, suatu percobaan untuk membagi dengan bilangan nol), atau transaksi yang mungkin telah tertunda untuk waktu lama sehingga nilai itu telah menurun ke nol.

Di dalam Membedakannya, ketika suatu transaksi dikenai suatu pembatalan nontermination, transaksi dimulai kembali. Pembatalan seperti itu dapat terjadi oleh karena pertentangan data dengan prioritas transaksi yang lebih tinggi lainnya, atau karena macet (*deadlock*). Hal itu sangat mungkin, tetapi, untuk suatu transaksi yang mendapat suatu pembatalan nontermination sangat banyak dikarenakan transaksi itu nilainya turun merosot menjadi nol, maka hal ini menjadi masalah pokok pada suatu penghentian (*termination*).

(II)Contoh 5.6



Gambar 5.5 Contoh Algoritma Elevator

Dalam gambar 5.5 lengan mengunjungi (mendatangi) track 3,4,5, dan 9 dalam perjalanan satu arah. Pada gilirannya ia berbalik arah ketika tidak ada permintaan yang ditunda untuk track yang lebih tinggi. Kemudian ia melayani akses permintaan untuk track 2, 1, dan 1.

Ada variasi dalam algoritma elevator, yaitu lengan melakukan scan dalam satu arah. Bila lengan sampai diujung, ia akan kembali tanpa melayani track dan mengulangi pelayanan dari awal. Jika algoritma

semacam ini digunakan untuk melayani permintaan seperti dalam gambar 5.5, track akan dikunjungi dalam urutan 3, 4, 5, 9, 1, 1, 2.

Algoritma elevator sudah dimodifikasi untuk digunakan dalam real-time system. Satu pendekatannya adalah mengklasifikasi permintaan kedalam prioritas kelas, tergantung kepada deadline karakteristik lain. Pada akhir pelayanan jika ada permintaan dari prioritas yang lebih tinggi dari yang lain, lengan akan pergi ke permintaan prioritas tertinggi, ia akan pergi ke track terdekat dan mulai melakukan eksekusi dari tempat tersebut. Jika tidak ada permintaan yang berprioritas tinggi, ia akan meneruskan algoritma elevator. Jika permintaan yang menunggu keputusan adalah berprioritas lebih rendah dari yang baru saja dilayani, ia akan memungut permintaan dengan prioritas yang lebih tinggi yang sedang menunggu keputusan dan memulai elevator dari titik tersebut. Pertanyaan sekarang adalah bagaimana menentukan level prioritas. Beberapa persiapan kerja adalah menandai dengan membuat jumlah level prioritas yang paling sedikit (misalnya 3) adalah yang terbaik, bagaimanapun tidak ada yang mengetahui bagaimana memilih performance yang optimal.

Variasi yang lain adalah sebagai berikut. Periksa setiap permintaan untuk melihat jika dapat dilihat. Tangkap akses yang dapat dilihat deadline terdepan dan pindahkan lengan ke arah tersebut hentikan semua jalan untuk melayani permintaan yang menunggu keputusan di jalan antaranya.

Kebijakan yang lain, mempertimbangkan antara deadline untuk masing-masing permintaan dan waktu yang diambil untuk memindahkan lengan untuk melayaninya. Kebijakan ini menjaga antrian yang di dalamnya permintaaan menunggu keputusan untuk disimpan dalam order deadline. Algoritma mempertimbangkan *q* element pertama dalam antrian, *q* disebut *window size.* Pembatasan pertimbangan ini mengurangi kompleksitas algoritma. Kita memesan setiap permintaan dalam antrian dan elemen ke *i* dalam beban w_i yang diberikan. Beban mencukupi dalam pertidaksamaan $w_i \leq w_j$ jika $i > j$. Jika δ_i adalah jarak lengan harus pindah dari posisi terakhir ke posisi permintaan ρ_i, prioritas permintaaan ρ_i diberikan oleh $p_i = 1/w_i\delta_i$. Pelayanan diberikan

kepada prioritas yang paling tinggi. Pembuat algoritma ini mengusulkan utuk menggunakan $w_i = \beta^{i-1}$, dimana $\beta \geq 1$adalah pilihan nilai konstan yang sesuai (secara simultan memperlihatkan bahwa β = 2 adalah sesuai) dan direkomendasikan memilih tiga atau empat *window size*. Tidak banyak diketahui tentang nilai β yang baik atau jika memilih beban yang lain yang terbaik.

Kebijakan diatas hanya mengambil pemesanan deadline kedalam *account.*Penjelasan sederhana adalah sebagai berikut:

d_i dinyatakan sebagai deadline yang dihubungkan dengan permintaan r_i. Tangkap sebuah nilai α tetap seperti $0 \leq \alpha \leq 1$dan melayani permintaan yang mempunyai nilai minimum dari fungsi $f(d_i \delta_i) = \alpha\delta_i + (1-\alpha)d_i$. α adalah parameter disain. Jika ini kecil, ia akan menaikkan pengaruh deadline, jika besar ia akan menaikkan pengaruh perpindahan lengan dalam memilih permintaaan berikut untuk dilayani. Penyusun kebijakan ini telah mendapatkan bahwa pemilihan α dalam range 0,7 – 0,8 sepertinya memberikan hasil yang baik

(III)

Heuristic that is a method using experience before all to do given duty. Pursuant to experience, duty will done quicker. Applying of using smart system technology or expert system.

(IV)

Key word thesis :

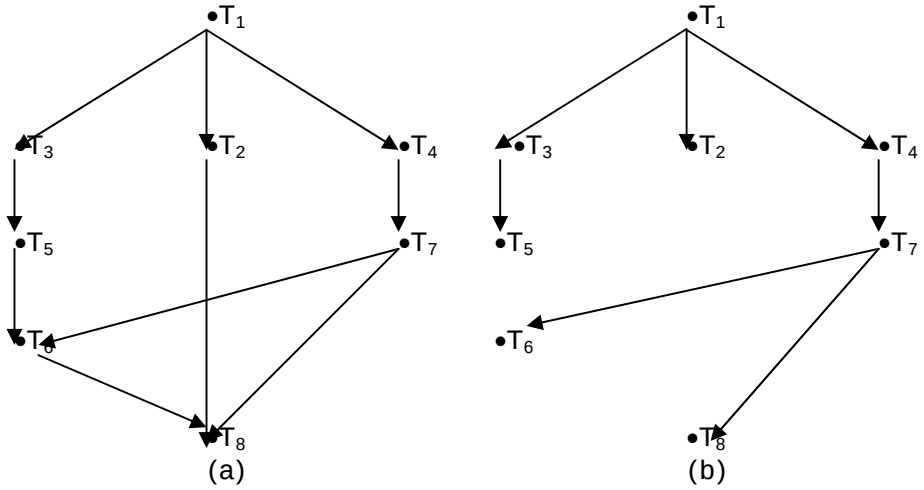
Wimax, WiFi, Wireless Metropolitan Area Networks (Wireless MAN), Topologi Kota Semarang.

TOEFL SCORE 500, Valid until April 15, 2007

Nama : Yulius Nugroho Adi S
NIM : 23204112
UTS Tanggal 23 Agustus 2005
Remidi Tanggal 25 Agustus 2005

Example 3.22.

Sekumpulan 8 task dengan waktu eksekusi berturut-turut 5, 6, 8, 1, 2, 4, 1, 2. Task **T_6, T_7, T_8** adalah tash OR. Grafik precedence dan grafik pemangkasan oleh MINPATH ditunjukkan pada gambar 3.22.



Gambar 3.22.
Transformasi grafik task oleh MINPATH: (a) grafik asli, (b) grafik yang dipangkas.

Algoritma MINPATH.
Selama A adalah sekumpulan OR task yang tidak kosong, lakukan :
Pilih task $T_i \in A$ yang tidak mendahului sebagai task OR
Tentukan k, yang memenuhi $L(k) \leq L(j)$ untuk semua $j \in P_i$.
Dalam G, pindahkan kembali semua sisi berakhir di T_i kecuali satu dari T_k
Beri label T_i sebagai AND task. (contoh 3.1)
Selesai

Nama : Yulius Nugroho Adi S
NIM : 23204112
UTS Tanggal 23 Agustus 2005
Remidi Tanggal 25 Agustus 2005

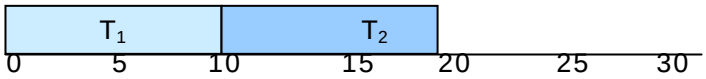
Example 3.27

Sistem dengan 2 task dengan parameter sebagai berikut :

	T_1	T_2
r_i	0	5
D_i	30	15
e_i	10	10

Ingat : T_1 EXCLUDES T_2

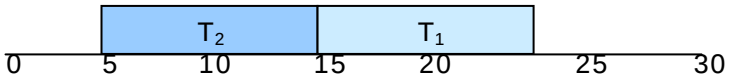
Penjadualan dengan inisial valid adalah :



Task T_2 tidak memenuhi deadline-nya dan keterlambatannya adalah 20 – 15 = 5. K(2) = {1}.GAP(1,2) = 10. Batasan terendah pada yang paling lambat ditunjukkan sebagai berikut :

$$\min_{k \in K(2)} \{f(2) + GAP(1,2) \ D_k\} = 0$$

Kita membuat suatu titik yaitu G_1 (2) = {1}, maka kita menambahkan relasi T_2 PRECEDES T_1. Saat algoritma penjadualan dijalankan dalam kondisi ini, kita memiliki penjadualan seperti pada gambar yang memenuhi semua deadline-nya.



Algoritma akan mengambil penjadualan ini dan berhenti.