

William Stallings

Data and Computer

Communications

7th Edition



Bab 12

Routing



Routing in Circuit Switched Network

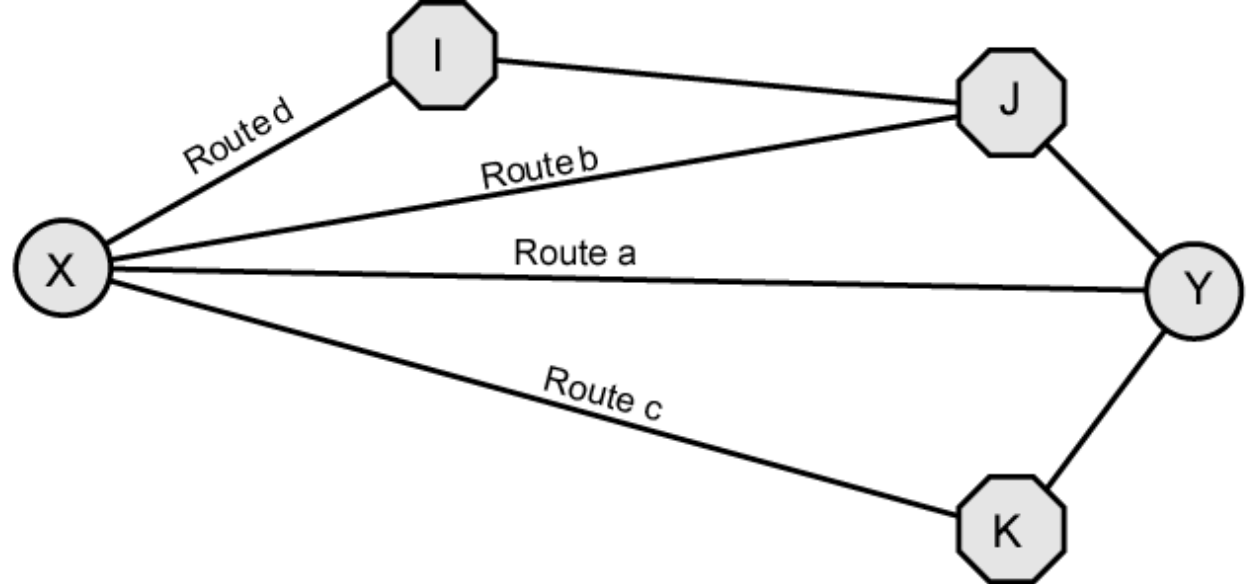
- Membuat banyak koneksi membutuhkan jalur menuju lebih dari satu switch
- Perlu untuk menemukan sebuah rute (*route*)
 - Efficiency
 - Resilience
- Switch telepon umum mempunyai struktur pohon
 - Roting static menggunakan pendekatan yang sama pada semua waktu
- Dynamic routing mengizinkan perubahan dalam routing, tergantung pada lalu lintas
 - Menggunakan sebuah peer structure untuk node



Alternate Routing

- Rute yang memungkinkan antara end offices yang ditentukan
- Membantu switch dalam memilih rute yang cocok
- Route yang dicatat berdasar preference order
- Perbedaan pengesetan dari rute (*route*) mungkin digunakan/dipakai pada waktu yang berbeda

Diagram Alternate Routing



Route a: X® Y
 Route b: X® J® Y
 Route c: X® K® Y
 Route d: X® I® J® Y

○ = end office

⬡ = intermediate switching node

(a) Topology

Time Period	First route	Second route	Third route	Fourth and final route
Morning	a	b	c	d
Afternoon	a	d	b	c
Evening	a	d	c	b
Weekend	a	c	b	d

(b) Routing table



Routing dalam Packet Switched Network

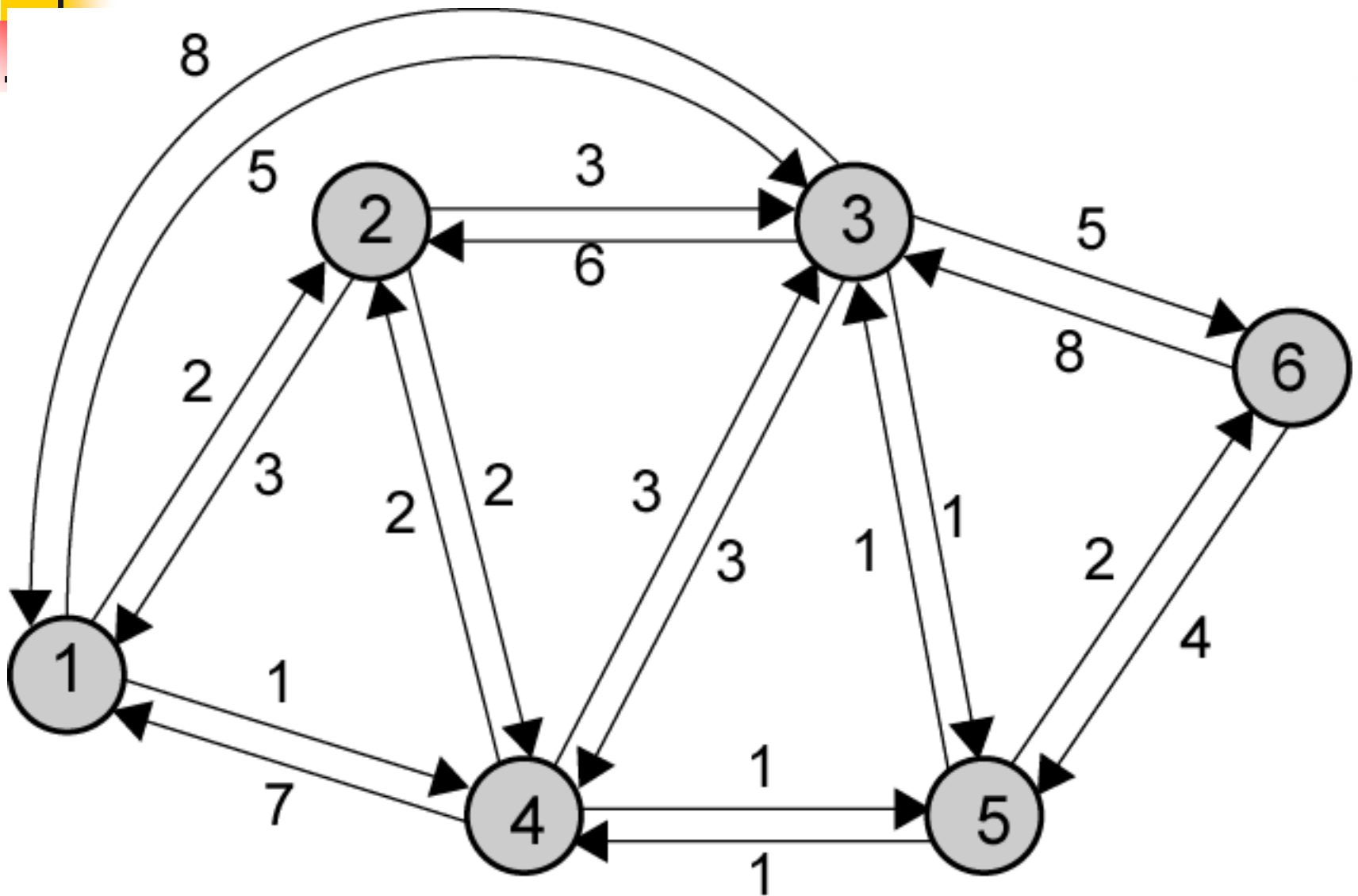
- Complex, aspek yang penting pada packet switched networks
- Karakteristik yang dibutuhkan
 - Ketepatan
 - Kesederhanaan
 - Ketahanan
 - Stabilitas
 - Fairness
 - Optimalisasi
 - Efisiensi



Kriteria Performance

- Digunakan untuk seleksi dari route
- Minimum hop
- Harga paling murah
 - Lihat Stallings appendix 10A untuk algoritma routing

Contoh Packet Switched Network





Keputusan waktu dan tempat

- Waktu
 - Packet atau virtual circuit basis
- Tempat
 - Didistribusikan
 - Dibuat oleh masing-masing node
 - Dipusatkan
 - Sumber

Sumber informasi jaringan dan Update Timing



- Keputusan dari routing biasanya berdasar pada pengetahuan/kapandaian dari jaringan (tidak selalu)
- Distributed routing
 - Nodes menggunakan lokal knowledge
 - Dapat mengumpulkan info dari node yang berdekatan
 - Dapat mengumpulkan info dari semua node pada sebuah potential route
- Central routing
 - Mengumpulkan info dari semua node
- Update timing
 - Ketika info jaringan yang disimpan node di-update
 - Fixed – tidak pernah di-update
 - Adaptive – update yang rutin



Strategi Routing

- Fixed
- Flooding
- Random
- Adaptive



Fixed Routing

- Satu rute permanen untuk setiap pasangan sumber sampai tujuan
- Menentukan route menggunakan algoritma biaya termurah (appendix 10A)
- Route fixed, sedikitnya sampai suatu perubahan di (dalam) topologi jaringan

Table Fixed Routing

CENTRAL ROUTING DIRECTORY

		From Node					
		1	2	3	4	5	6
To Node	1	—	1	5	2	4	5
	2	2	—	5	2	4	5
	3	4	3	—	5	3	5
	4	4	4	5	—	4	5
	5	4	4	5	5	—	5
	6	4	4	5	5	6	—

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6 Directory

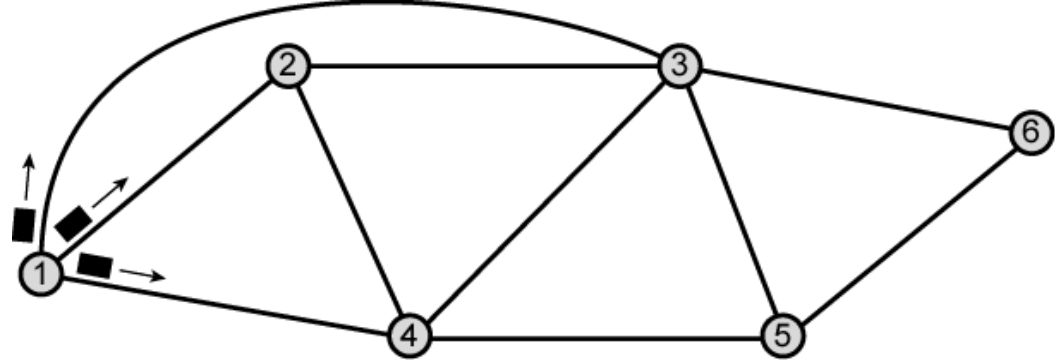
Destination	Next Node
1	5
2	5
3	5
4	5
5	5



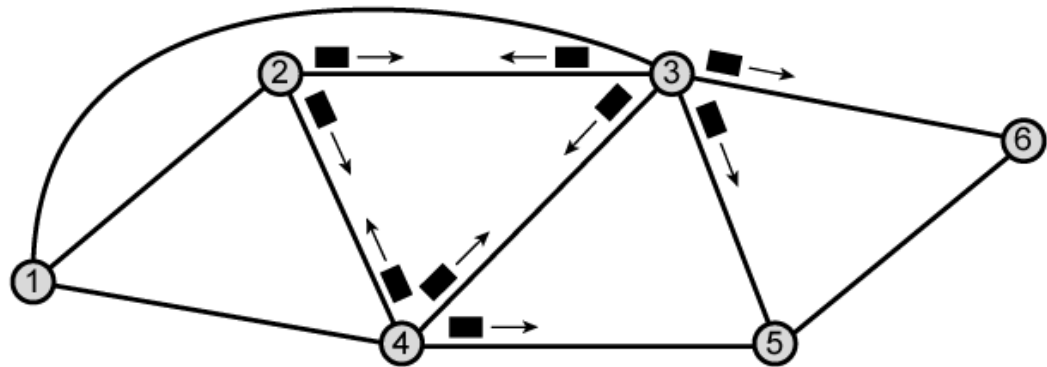
Flooding

- Tidak ada info jaringan yang dibutuhkan
- Paket dikirimkan oleh node ke setiap client yang lain (*neighbor*)
- Paket yang datang dikirim kembali pada setiap link selain jalur pengirim
- Secepatnya sejumlah salinan akan tiba di tujuan
- Masing-masing paket telah dinomori secara unik, jadi salinannya dapat dibuang/diputus
- Node dapat mengingat paket yang dikirimkan untuk menjaga agar paket tersebut tidak keluar dari jaringan
- Dapat memasukkan sebuah hop count kedalam paket

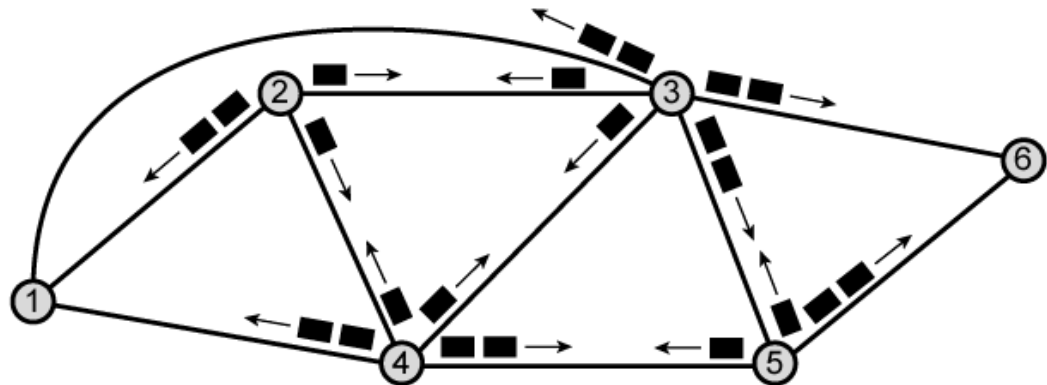
Contoh Flooding



(a) First hop



(b) Second hop



(c) Third hop



Properties of Flooding

- Semua rute yang mungkin dicoba
 - Sangat sempurna
- Sedikitnya satu paket akan dapat diambil hop count route terkecil
 - Dapat digunakan set up virtual circuit
- Semua node dilewati
 - Sangat berguna untuk mendistribusikan informasi (Contoh: routing)



Random Routing

- Node memilih satu jalur outgoing untuk pengiriman ulang paket yang datang
- Seleksi dapat diacak atau round robin
- Dapat memilih jalur outgoing berdasar pada perhitungan probabilitas
- Tidak ada info jaringan yang diperlukan
- Rute biasanya bukan least cost atau minimum hop



Adaptive Routing

- Digunakan oleh hampir seluruh jaringan paket switching
- Keputusan routing berubah ketika kondisi pada jaringan berubah
 - Kegagalan
 - Kemacetan pada jalur jaringan
- Membutuhkan informasi tentang jaringan
- Keputusan lebih kompleks
- Tradeoff antara kualitas informasi jaringan dan overhead
- Reaksi yang terlalu cepat dapat menyebabkan osilasi
- Terlalu lambat menjadi relevant



Adaptive Routing - Advantages

- Performance lebih baik
- Aid congestion control (Lihat bab 13)
- Sistem yang kompleks
 - Mungkin tidak ssuai dengan teori



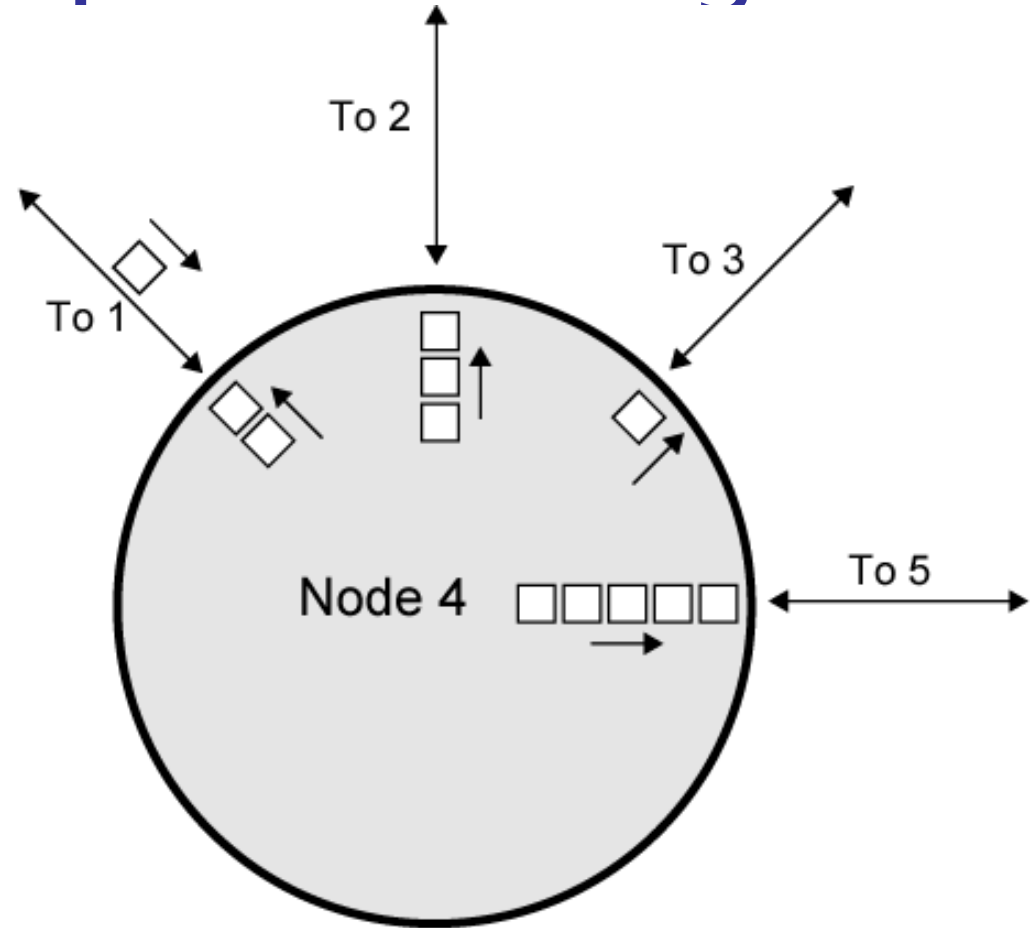
Klasifikasi

- Berdasar pada sumber informasi
 - Local (isolated)
 - Route menuju outgoing link dengan antrian terpendek
 - Dapat meliputi penyimpangan untuk masing-masing tujuan
 - Jarang digunakan – jangan memakai info mudah yang tersedia
 - Node yang berdekatan
 - Semua node

Isolated Adaptive Routing

Node 4's Bias
Table for
Destination 6

Next Node	Bias
1	9
2	6
3	3
5	0





Strategi ARPANET Routing (1)

- Generasi pertama
 - 1969
 - Distributed adaptive
 - Estimasi delay sebagai standart performance
 - Algoritma Bellman-Ford (appendix 10a)
 - Node mengubah delay vector dengan neighbors
 - Update routing table berdasar pada info yang datang
 - Tidak mempertimbangkan kecepatan jalur, hanya panjang antrian
 - Panjang antrian bukan sbuah ukuran yang bagus dari delay
 - Respond lambat untuk congestion



Strategi ARPANET Routing (2)

- Generasi kedua
 - 1979
 - Penggunaan delay sebagai standart performance
 - Delay dapat diukur secara langsung
 - Menggunakan algoritma Dijkstra(appendix 10a)
 - Good under light dan medium loads
 - Jika terjadi heavy loads, terjadi sedikit hubungan antara reported delay dan hal-hal sebelumnya



Strategi ARPANET Routing (3)

- Generasi ketiga
 - 1987
 - Biaya perhitungan link dirubah
 - Ukuran rata-rata delay tidak lebih dari 10 detik
 - Normalnya berbasis pada current value dan previous results



Perhitungan kerugian terkecil

- Berdasar pada routing decisions
 - Dapat meminimalisasi hop dengan masing-masing link cost 1
 - Dapat mempunyai link value yang berbanding terbalik terhadap kapasitas
- Jaringan yang diberikan oleh node-node dikoneksikan oleh bi-directional links
- Masing-masing link mempunyai kelemahan di setiap tujuan
- Menetapkan kerugian dari jalurantara dua node node seperti jumlah kerugian dari links traversed
- Untuk masing-masing pasangan dari node, menemukan sebuah jalur dengan kelemahan terkecil
- Kelemahan Link dalam tujuan yang berbeda mungkin dapat berbeda
 - Contoh : panjang dari antrian paket



Definisi algoritma Dijkstra

- Menemukan jalur terpendek dari sumber node yang diberikan ke semua node yang lain, Find shortest paths from given source node to all other nodes, dengan mengembangkan jalur yang dibutuhkan dari penambahan panjang jalur
- N = set of nodes dalam jaringan
- s = sumber node
- T = set of node yang paling jauh berdasarkan algoritma
- $w(i, j)$ = beban link dari node i ke node j
 - $w(i, i) = 0$
 - $w(i, j) = \infty$ jika dua node tidak terhubung secara langsung
 - $w(i, j) \geq 0$ jika dua node terhubung secara langsung
- $L(n)$ = biaya dari least-cost path dari node s dsampai node n yang diketahui
 - Pada akhirnya, $L(n)$ adalah biaya dari least-cost path dari s ke n



Metode algoritma Dijkstra

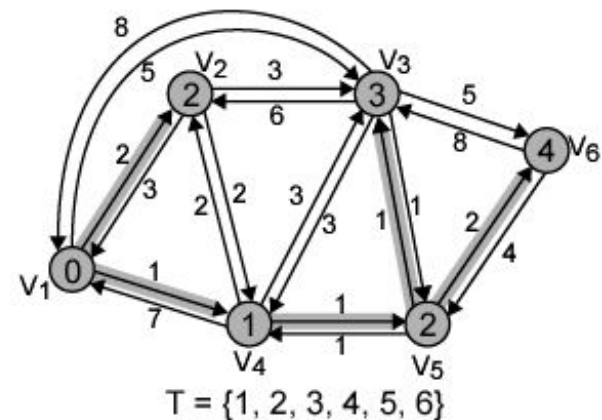
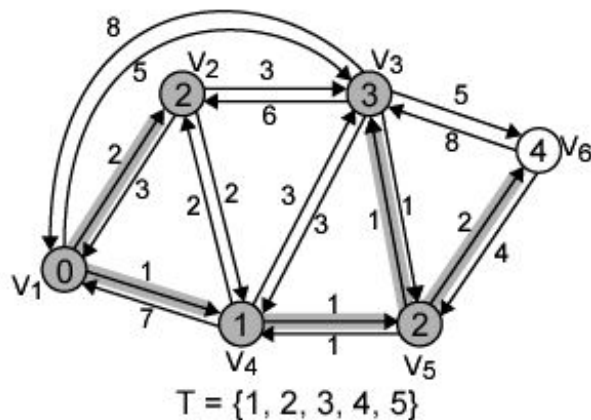
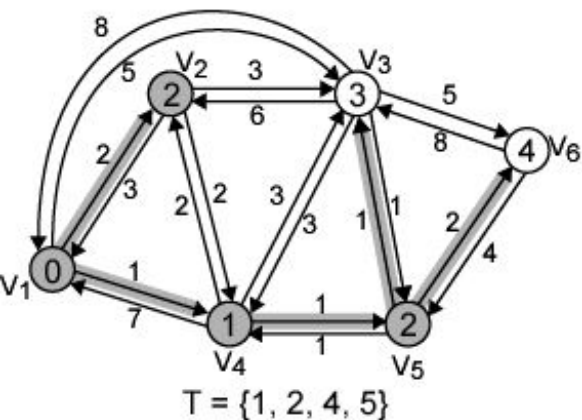
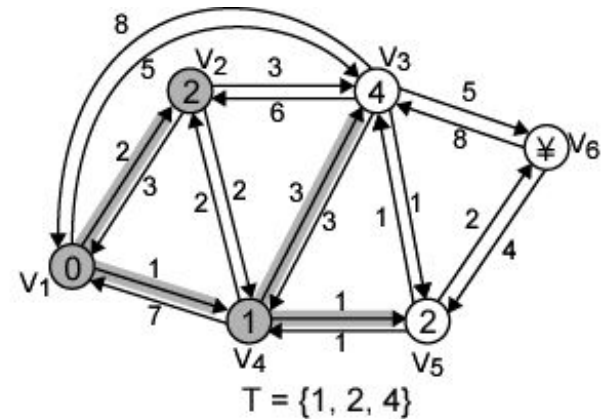
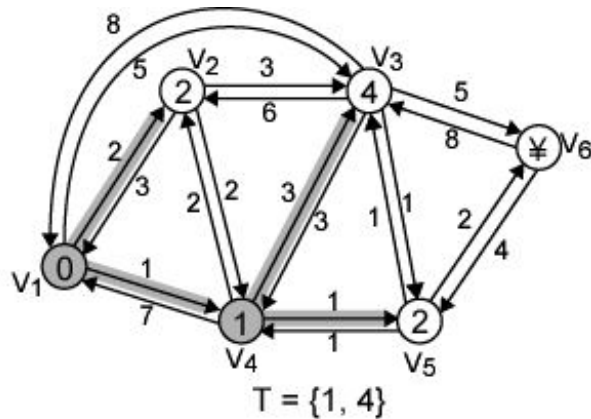
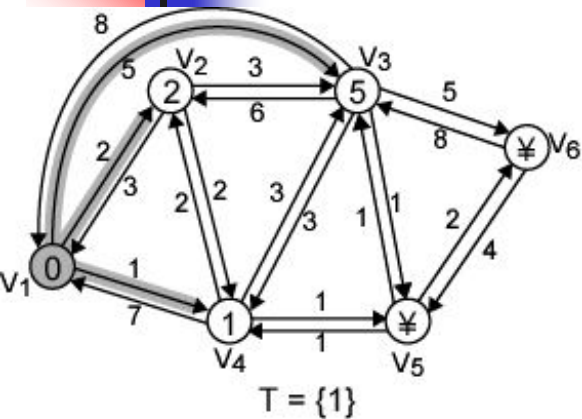
- Step 1 [inisialisasi]
 - $T = \{s\}$ Set of nodes so far incorporated consists of only source node
 - $L(n) = w(s, n)$ for $n \neq s$
 - Biaya awal jalur dari node yang bertetangga adalah biaya link
- Step 2 [memperoleh node berikutnya]
 - menemukan neighboring node tidak dalam T dengan least-cost path from s
 - Menggabungkan node kedalam T
 - Juga menggabungkan sisi yang bertabrakan pada node tersebut dan sebuah node di T yang memiliki kontribusi dalam path
- Step 3 [Update Least-Cost Paths]
 - $L(n) = \min[L(n), L(x) + w(x, n)]$ for all $n \notin T$
 - Jika nilainya minimum, jalur dari s ke n adalah jalur dari s ke x concatenated dengan edge dari x ke n
- Algoritma berakhir ketika semua node sudah ditambahkan ke T



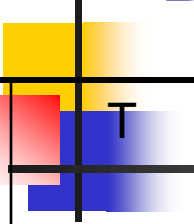
Catatan algoritma Dijkstra

- Pada akhirnya, nilai $L(x)$ yang berasosiasi dengan setiap node x adalah biaya (panjang) dari least-cost path dari s ke x .
- Tambahan, T menjelaskan least-cost path dari s ke setiap node lainnya
- Satu iterasi dari langkah 2 dan 3 menambahkan sebuah node baru pada T
 - Menjelaskan least cost path dari s ke node tersebut

Contoh algoritma Dijkstra



Penyelesaian dari Algoritma Dijkstra



Iteration	T	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
5	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
6	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6



Definisi Igoritma Bellman-Ford

- Menemukan jalur terpendek dari subject node yang diberikan ke constraint that paths contain at most one link
- Menemukan jalur terpendek dengan keterbatasan dari jalur dari dua link
- Dan seterusnya
- s = source node
- $w(i, j)$ = beban link dari node i ke node j
 - $w(i, i) = 0$
 - $w(i, j) = \infty$ Jika dua node tidak terhubung secara langsung
 - $w(i, j) \geq 0$ Jika dua node terhubung secara langsung
- h = jumlah maksimum dari links dalam jalur pada current stage dari algoritma
- $L_h(n)$ = biaya dari least-cost path dari s ke n dengan catatan dibawah h links

Metode algoritma Bellman-Ford

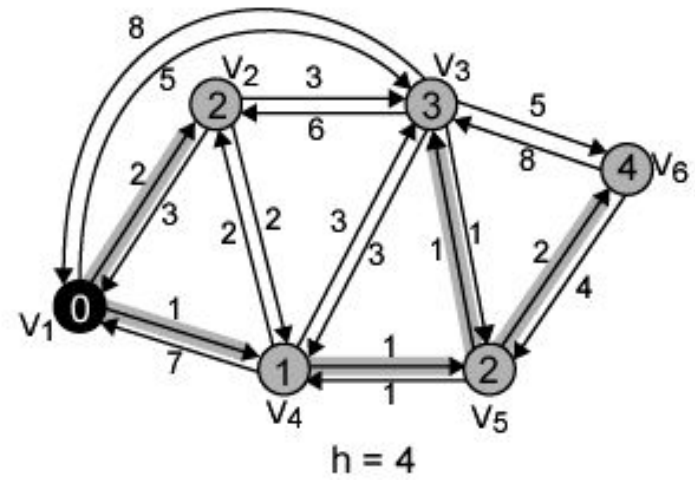
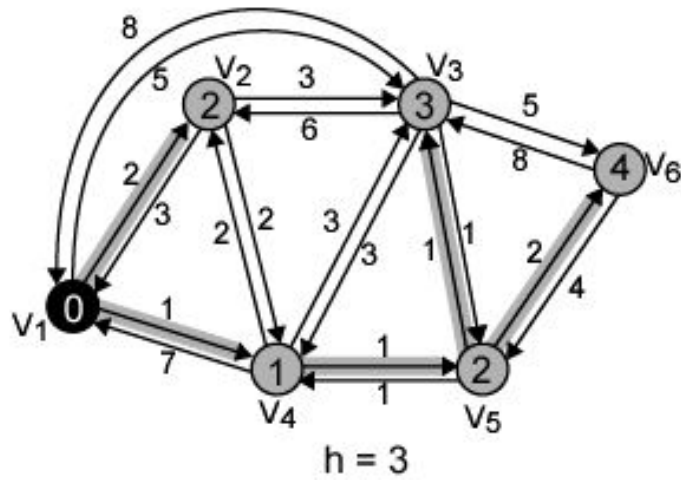
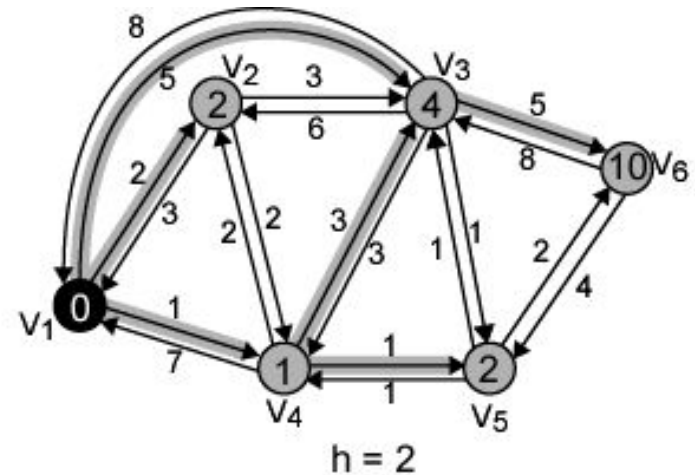
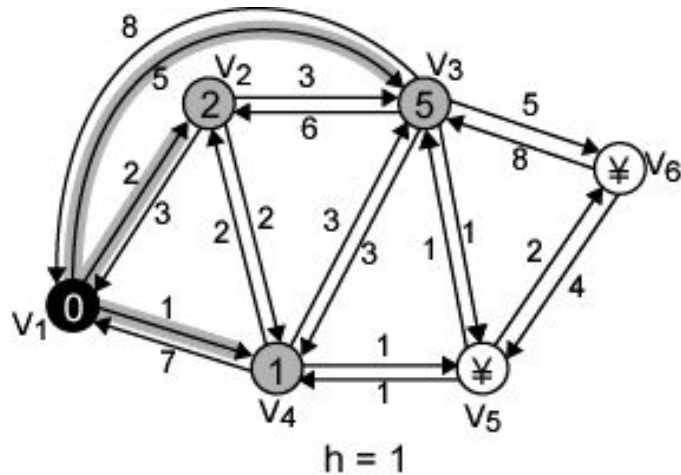
- Step 1 [Inisialisasi]
 - $L_0(n) = \infty$, for all $n \neq s$
 - $L_h(s) = 0$, for all h
- Step 2 [Update]
- For each successive $h \geq 0$
 - For each $n \neq s$, compute
 - $L_{h+1}(n) = \min_j [L_h(j) + w(j, n)]$
- Menghubungkan n dengan predecessor node j yang menghasilkan minimum
- Mengeliminasi koneksi yang lain pada n dengan predecessor node yang berbeda yang terbentuk pada iterasi sebelumnya
- Jalur dari s ke n berakhir dengan link dari j ke n



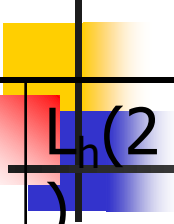
Catatan algoritma Bellman-Ford

- Pada masing-masing iterasi dari step 2 dengan $h=K$ dan untuk masing-masing tujuan node n , algoritma mengkompare jalur dari s ke n dengan panjang $K=1$ dengan jalur dari iterasi sebelumnya
- Jika jalur sebelumnya terpendek maka ditahan
- Sebaliknya jalur baru ditetapkan

Contoh algoritma Bellman-Ford



Contoh penyelesaian Bellman-Ford



h	$L_h(2)$)	Pat h	$L_h(3)$)	Path	$L_h(4)$)	Pat h	$L_h(5)$)	Path	$L_h(6)$)	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6



Comparison

- Penyelesaian dari dua persetujuan algoritma
- Information gathered
 - Bellman-Ford
 - Perhitungan untuk node n involves knowledge of link cost ke semua neighboring nodes plus total cost ke masing-masing neighbor dari s
 - Masing-masing node dapat dirawat set of costs dan jalur untuk setiap node yang lain
 - Dapat mengubah informasi dengan direct neighbors
 - dapatkan update costs dan pathsberdasar pada informasi dari neighbors dan knowledge of link costs
 - Dijkstra
 - Masing-masing node memerlukan topologi yang lengkap
 - Perlu diketahui link costs dari semua links dalam jaringan
 - Harus merubah informasi dengan semua node yang lain



Evaluasi

- Tergantung pada processing time dari algoritma
- Tergantung pada jumlah dari informasi yang dibutuhkan dari node yang lain
- Implementation specific
- Keduanya bertemu dibawah topologi static dan costs
- Bertemu pada solusi yang sama
- Jika link costs berubah, algoritma akan mencoba untuk mengecek kembali
- Jika link costs tergantung pada lalu lintas, dimana tergantung pada rute yang dipilih, kemudian diumpan balikkan
 - Mungkin hasilnya tidak stabil

Referensi



- Stalling bab 12
- Routing information from Comer D.
Internetworking with TCP/IP Volume 1,
Prentice Hall, Upper Saddle River NJ.