Aula 18 - Roteamento: Introdução, Algoritmos de Estado de Enlace

Diego Passos

Universidade Federal Fluminense

Redes de Computadores I

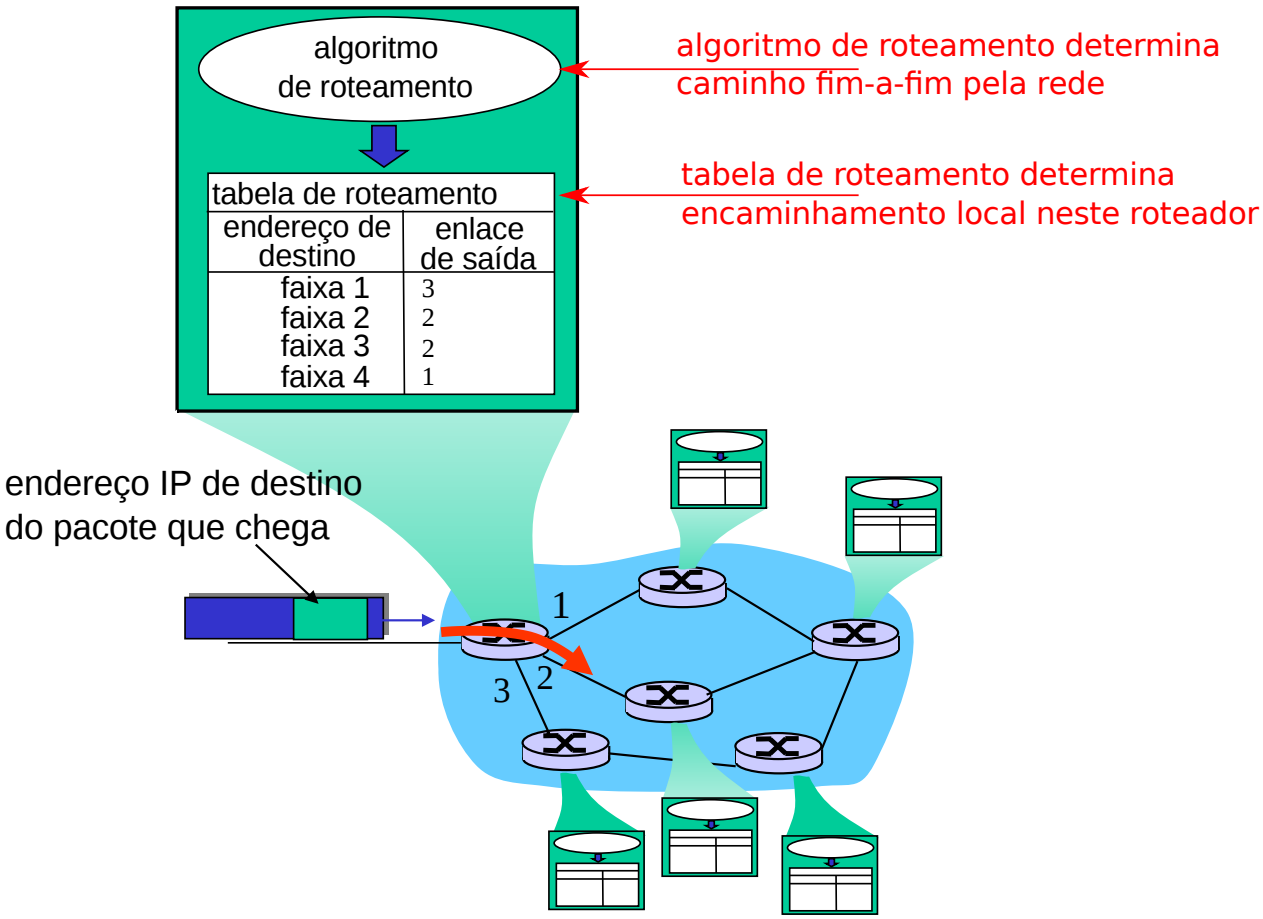
Material adaptado a partir dos slides originais de J.F Kurose and K.W. Ross.

Revisão da Última Aula...

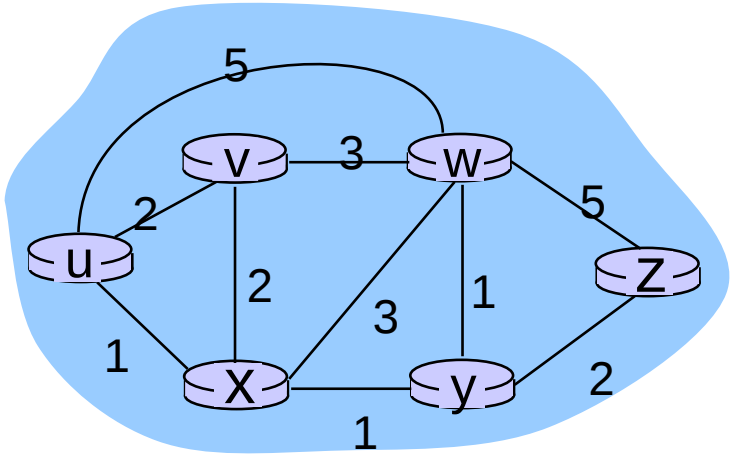
- **NAT**:
  - Tradução de endereços.
  - Rede local *vs.* rede externa.
  - Endereços privados *vs.* públicos, roteáveis.
  - Pacote sai: IP e porta de origem alterados.
  - Pacote entra: IP e porta de destino são alterados.
  - Tabela NAT: armazana mapeamentos.
- NAT: Motivação.
  - **Escassez de IPs**.
  - Independência dos endereços do ISP.
  - Segurança.
- NAT *Traversal*:
  - Conexão de fora para dentro do NAT.
  - Entradas estáticas na tabela.
  - Protocolo IGD.
  - *Relaying* (aplicação).
- **ICMP:**
  - Gerência do IP.
  - Informações, condições de erro.
  - Diversas tipos de mensagens.
  - Suporte a algumas ferramentas usuais.
- **IPv6: Motivações.**
  - Mais endereços.
  - Menor *overhead* de processamento.
- **IPv6: diferenças**.
  - Cabeçalho fixo.
  - Fragmentação não permitida.
  - Melhor suporte a QoS.
  - ICMPv6.
- **IPv6: Transição.**
  - Gradual, coexistência com IPv4.
  - Solução: tunelamento.

Roteamento: Introdução

Sinergia entre Roteamento e Encaminhamento



Abstração de Grafo

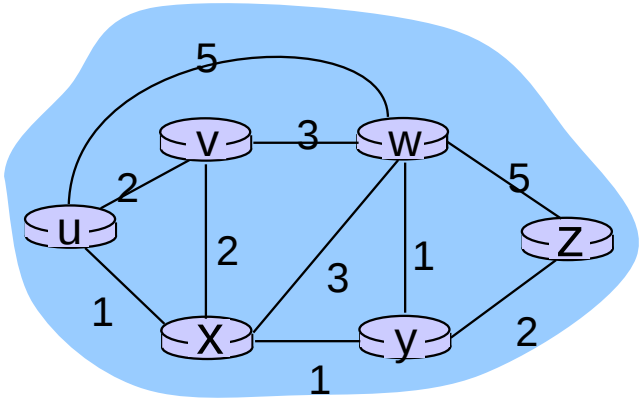


- Grafo: G = (N, E).
- N = conjunto de roteadores = {u, v, w, x, y, z}.
- E = conjunto de enlaces = {(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)}

Nota

Grafos são uma abstração útil em outros contextos de redes de computadores, *e.g.*, P2P, onde N é o conjunto de pares e E é o o conjunto de conexões TCP.

Abstração de Grafo: Custos



- $c(x, x')$ = custo do enlace (x, x').
  - *e.g.*, $c(w, z) = 5$.
- Custo pode ser sempre 1, ou inversamente proporcional à banda, ou inversamente proporcional ao congestionamento.

- Custo de um caminho $(x_1, x_2, x_3, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$

Pergunta chave: qual é o caminho de **custo mínimo** entre u e z?
Algoritmo de roteamento: algoritmo que encontra o caminho de custo mínimo.

Classificação de Algoritmos de Roteamento

- **Perguta:** informação global ou decentralizada?
- **Global:**
  - Todos os roteadores tem **visão completa da topologia, custos de enlaces.**
  - Algoritmos baseados em "Estado de Enlace".
    - *Link State*, em inglês.
- **Decentralizada:**
  - Roteador **conhece vizinhos** fisicamente conectados por enlaces, custos dos enlaces para vizinhos.
  - Processo iterativo de computação, troca de informação com vizinhos.
  - Algoritmos baseados em "Vetor de Distâncias".
    - *Distance Vector*, em inglês.

- **Pergunta:** estático ou dinâmico?
- **Estático:** rotas mudam pouco com o tempo.
- **Dinâmico:** rotas mudam rapidamente.
  - Em resposta a mudanças nos custos dos enlaces.
  - Atualização periódica.
- **Pergunta:** pró-ativo ou reativo?
- **Pró-ativo:** rotas encontradas antes de serem necessárias.
- **Reativo:** rotas encontradas sob demanda.

Algoritmos Baseados em Estado de Enlaces

Um Algoritmo de Roteamento Baseado em Estado de Enlaces

- **Algoritmo de Dijkstra:**
  - Topologia da rede, custos dos enlaces conhecidos por todos os nós.
    - Conseguido através da difusão do "estado dos enlaces".
    - Todos os nós tem a mesma informação.
      - Ou deveriam ter.
    - Computa caminho de custo mínimo de um nó ("origem") para todos os outros nós.
      - Resulta na tabela de roteamento para aquele nó.
    - Iterativo: depois de k iterações, conhece o caminho mínimo para k destinos.

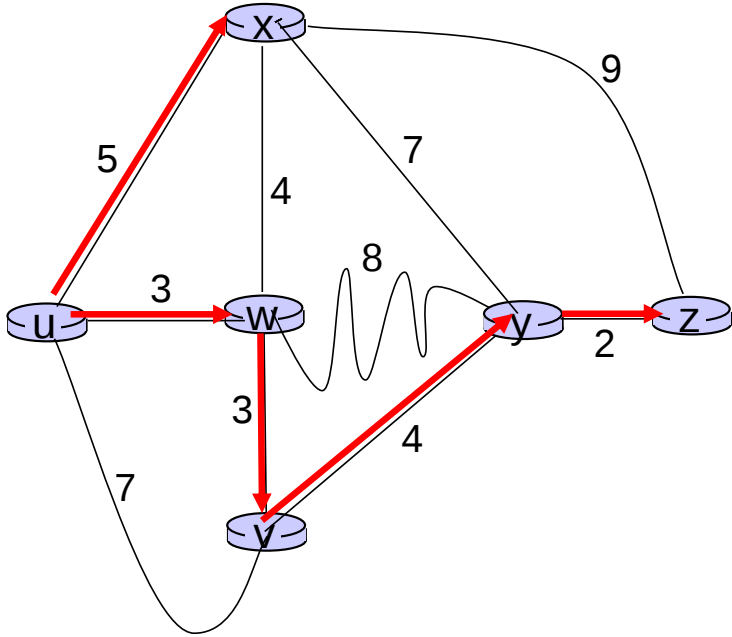
- **Notação:**
  - $c(x, y)$: custo do enlace do nó x para nó y.
    - infinito, se x e y não são vizinhos.
  - $D(v)$: custo atualmente conhecido para o caminho da origem até v.
  - $p(v)$: predecessor de v no caminho da origem para v.
  - N': conjunto de nós para os quais definitivamente conhecemos o melhor custo.

Algoritmo de Dijkstra

1	$N' \leftarrow \{u\}$	// Caminho para origem é trivial
2	$\forall v \in N$	// Para os demais nós
3	**se** $(u, v) \in E$	// É vizinho da origem?
4	**então** $D(v) \leftarrow c(u, v)$	// Conhecemos um caminho
5	**senão** $D(v) = \infty$	// Ainda sem caminho
6		
7	**Repita**	// Loop principal
8	Encontre $w \notin N'$ tal que D(w) seja mínimo	// Melhor caminho provisório
9	$N' \leftarrow N' \cup \{w\}$	// Se torna definitivo
10	$\forall v$ tal que $v \notin N' \wedge (w, v) \in E$	// Vizinhos de w ainda provisórios
11	$D(v) \leftarrow min(D(v), D(w) + c(w, v))$	// Anterior ou passando por w?
12	**Até que** $N' = N$	// Até adição de todos os nós

Algoritmo de Dijkstra: Exemplo

Passo	N'	$D(v)$ $p(v)$	$D(w)$ $p(w)$	$D(x)$ $p(x)$	$D(y)$ $p(y)$	$D(z)$ $p(z)$
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

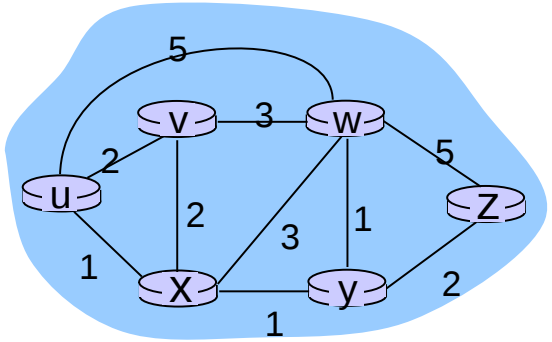


Notas

- Constrói uma **árvore de caminhos de custo mínimo** percorrendo predecessores.
- Podem existir empates. Critério de desempate é arbitrário.

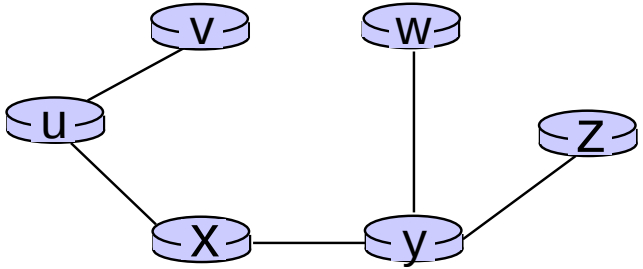
Algoritmo de Dijkstra: Outro Exemplo (I)

Passo	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Algoritmo de Dijkstra: Outro Exemplo (II)

- Árvore de caminhos de menor custo formada a partir de u:

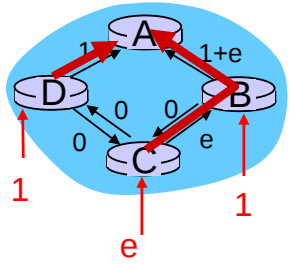


- Tabela de roteamento resultante:

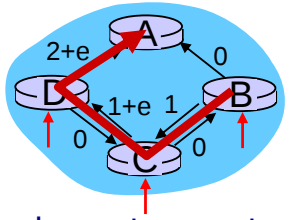
Destino	Enlace
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Algoritmo de Dijkstra: Discussão

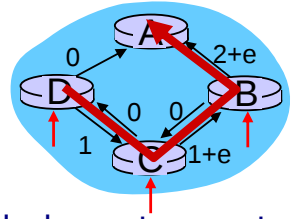
- **Complexidade do algoritmo:** com n nós.
  - Cada iteração: verifica todos os possíveis nós $w \notin N'$.
  - $\frac{n(n+1)}{2}$ comparações: $O(n^2)$.
  - Implementações mais eficientes são possíveis: $O(n \cdot log\ n)$.
- **Podem ocorrer oscilações:**
  - *e.g.* custos dos enlaces iguais a quantidade de tráfego transportado:



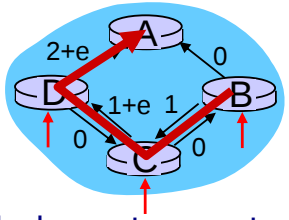
inicialmente



dados estes custos, encontrar novas rotas... resultando em novos custos



dados estes custos, encontrar novas rotas... resultando em novos custos



dados estes custos, encontrar novas rotas... resultando em novos custos

Resumo da Aula...

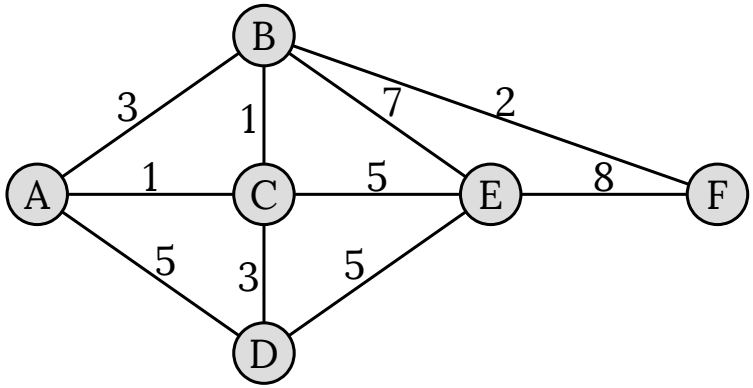
- **Roteamento:**
  - Encontrar caminhos fim-a-fim.
  - **Construir tabela de roteamento**.
    - Consultada no encaminhamento.
- **Grafos**: usados como abstração para representar a rede.
  - Roteadores são nós.
  - Enlaces são arestas.
  - Podem ter pesos: **medida de qualidade do enlace**.
    - Relacionado a banda, atraso, congestionamento, ...
- **Classificações:**
  - Estado de Enlace *vs.* Vetor de Distâncias.
  - Dinâmico *vs.* Estático.
- **Roteamento baseado em Estado de Enlaces:**
  - Algoritmo de Dijkstra.

Próxima Aula...

- Prosseguiremos o estudo sobre roteamento.
- Mais dois temas:
  - Algoritmos baseados em Vetor de Distâncias.
  - Roteamento hierárquico.

Exercícios

1. Execute o Algoritmo de Dijkstra a partir do nó D do grafo abaixo:



2. Suponha que uma instituição possua a sub-rede `199.67.44.0/22`. A instituição é dividida em 5 departamentos, com as seguintes necessidades:
   - **Departamento A:** 50 *hosts*.
   - **Departamento B:** 400 *hosts*.
   - **Departamento C:** 50 *hosts*.
   - **Departamento D:** 32 *hosts*.

   Projete uma divisão do bloco de endereços em sub-redes menores que atendam às necessidades dos departamentos **evitando desperdício de endereço**. Assuma que cada *host* demanda exatamente um endereço IP.