Aula 15 - NAT, ICMP e IPv6, Roteamento (I)

Diego Passos

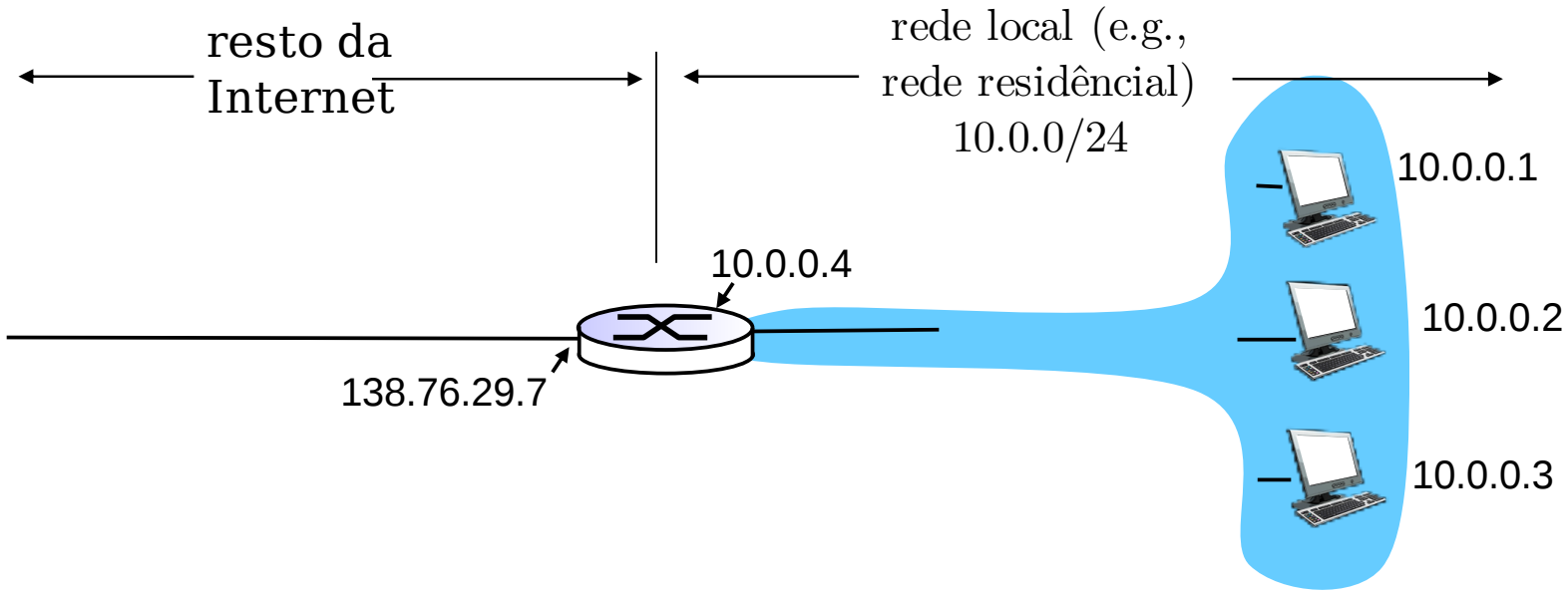
Universidade Federal Fluminense

Redes de Computadores

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

NAT

NAT: *Network Address Translation*



- **Todos** os datagramas **deixando** a rede local possuem o mesmo único endereço de origem: 138.76.29.7.
  - Diferenciação através do **número de porta de origem**.
- Datagramas com origem ou destino nesta rede possuem endereços de origem, destino da sub-rede 10.0.0/24.

NAT: Motivação

- Rede local pode utilizar um único endereço, do ponto de vista do mundo externo.
  - Não é necessária uma faixa de endereços do ISP: um único endereço IP para todos os dispositivos.
  - Pode-se alterar os endereços dos dispositivos locais sem notificação ao mundo externo.
  - Pode-se mudar de ISP sem que os endereços dos dispositivos locais sejam alterados.
  - Dispositivos dentro da rede local não são explicitamente endereçáveis, visíveis ao mundo externo.
    - Um (pequeno) benefício de segurança.
- NAT consegue lidar com a **escassez de endereços IPv4**.

NAT: Implementação

- Um roteador que realiza NAT precisa:
  - **Datagramas que saem: substituir** `(IP de origem, porta de origem)` de cada datagrama para `(IP do roteador, nova porta de origem)`.
    - Nó remoto respoderá utilizando `(IP roteador, nova porta de origem)` como destino.
  - **Armazenar (na tabela NAT)** todo mapeamento feito entre `(IP de origem, porta de origem)` e `(IP roteador, nova porta de origem)`.
  - **Datagramas que chegam: substituir** `(IP roteador, nova porta de origem)` nos campos de destino do pacote por `(IP de origem, porta de origem)` armazenado na tabela NAT.

NAT: Exemplo

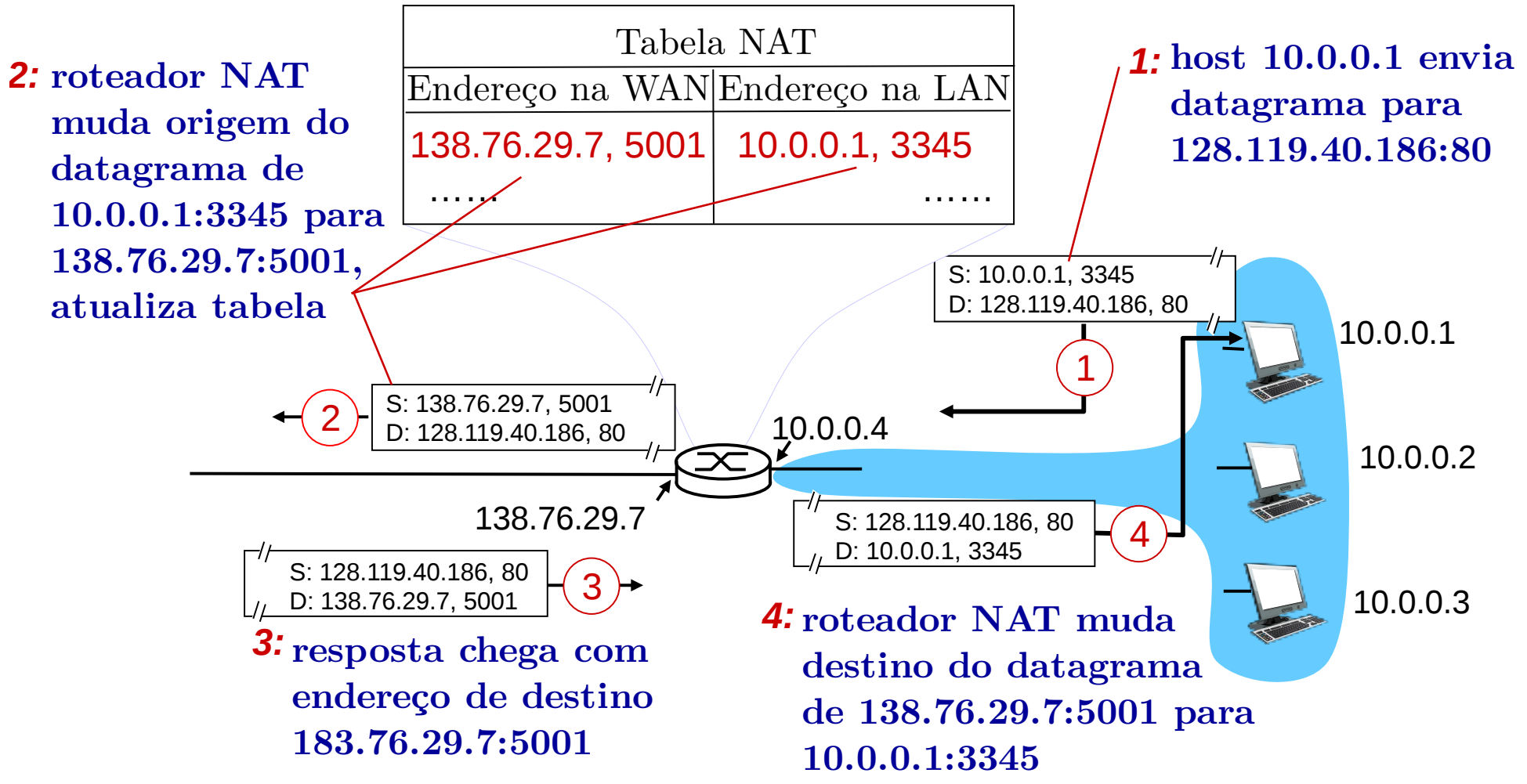


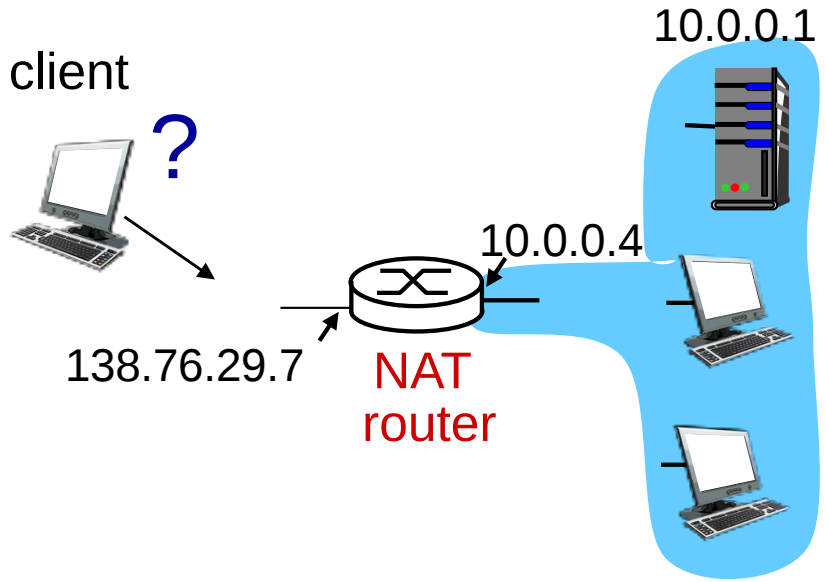
Tabela NAT
Endereço na WAN
Endereço na LAN
138.76.29.7, 5001
10.0.0.1, 3345
.....
.....
2: roteador NAT muda origem do datagrama de 10.0.0.1:3345 para 138.76.29.7:5001, atualiza tabela
1: host 10.0.0.1 envia datagrama para 128.119.40.186:80
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
1
2
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
10.0.0.4
138.76.29.7
10.0.0.1
10.0.0.2
10.0.0.3
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
4
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
3
3: resposta chega com endereço de destino 183.76.29.7:5001
4: roteador NAT muda destino do datagrama de 138.76.29.7:5001 para 10.0.0.1:3345

NAT: Análise

- Campo de número de porta: 16 bits.
  - 65000 conexões simultâneas usando um único endereço IP!
- NAT é controverso:
  - Roteadores só deveriam processar até a camada 3 (camada de rede).
  - NAT viola o argumento fim-a-fim.
    - Muitas vezes, o **NAT precisa ser levado em consideração por projetistas de aplicações**, *e.g.*, aplicações P2P.
  - Escassez de endereços deve ser resolvida pela adoção do IPv6.

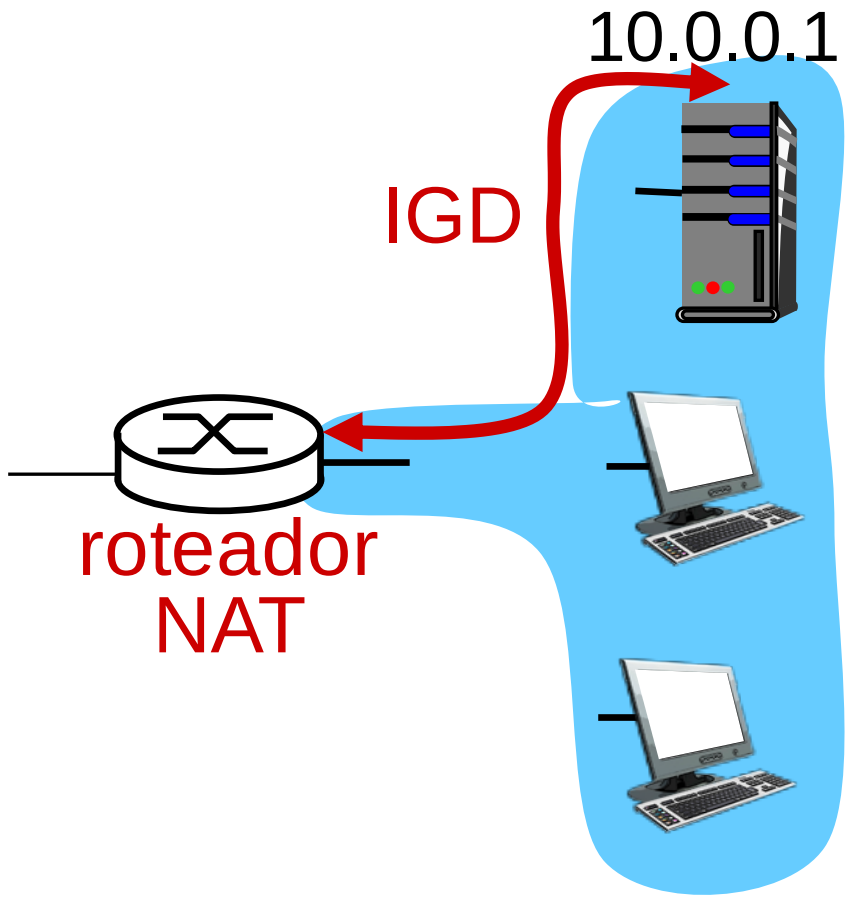
NAT *Traversal* (I)

- Cliente quer se conectar ao servidor com endereço 10.0.0.1.
  - Endereço 10.0.0.1 local para a LAN (cliente não pode usá-lo como endereço de destino).
  - Apenas um endereço visível externamente: 138.76.29.7.
- **Solução 1:** configurar NAT estaticamente para encaminhar conexões que chegam para uma dada porta para o servidor.
  - *e.g.*, `(138.76.29.7, porta 2500)` sempre é traduzido (e encaminhado) para `(10.0.0.1, porta 25000)`.



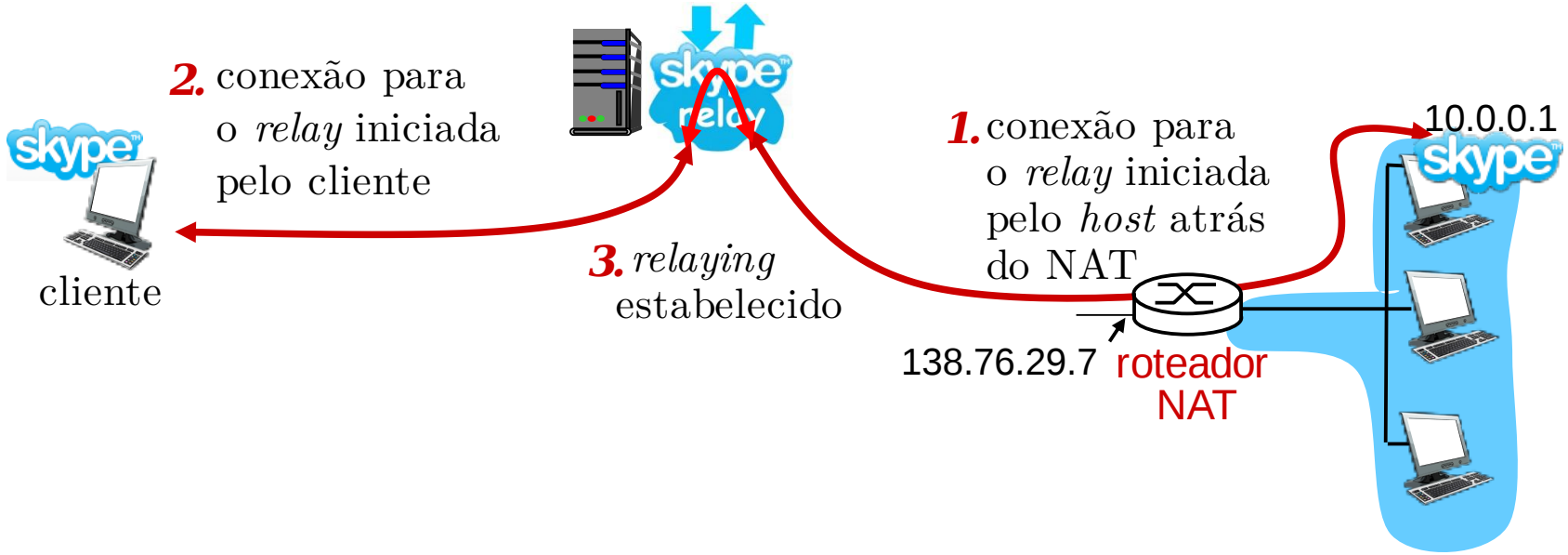
NAT *Traversal* (II)

- **Solução 2:** *Internet Gateway Device Protocol* (IGD).
  - Parte do *Universal Plug and Play* (UPnP).
- Permite que *host* atrás de NAT:
  - Aprenda endereço IP público (138.76.29.7).
  - Adicione/remova mapeamentos de porta (com tempos de *lease*).
- *i.e.*, automatizar configuração estática dos mapeamentos do NAT.



NAT *Traversal* (III)

- **Solução 3:** *relaying* (usado, por exemplo, no Skype).
  - Cliente atrás do NAT estabelece conexão com *host* intermediário.
  - Cliente externo se conecta ao mesmo *host* intermediário.
  - *Host* intermediário (*relay*) faz a ponte entre pacotes das duas conexões.



ICMP

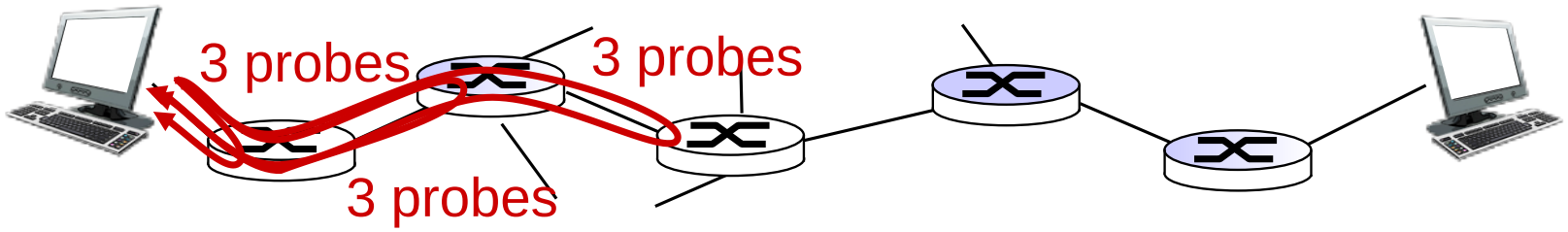
ICMP: *Internet Control Message Protocol*

- Usado por *hosts* e roteadores para comunicar informações no nível de rede.
  - Reportar erros: *host*, rede, porta, protocolo inalcançáveis.
  - *Echo request/reply*: usado pelo utilitário `ping`.
- Protocolo de camada de rede, mas "sobre o IP":
  - Mensagens ICMP transportadas em datagramas IP.
- **Mensagem ICMP:** tipo, código, além dos primeiros 8 bytes do datagrama IP que causaram o erro.

Tipo	Código	Descrição
0	0	*echo reply*
3	0	Rede de destino inalcançável
3	1	*Host* de destino inalcançável
3	2	Protocolo de destino inalcançável
3	3	Porta de destino inalcançável
3	6	Rede de destino desconhecida
3	7	*Host* de destino desconhecido
4	0	*Source quench* (controle de congestionamento, não usada)
8	0	*echo request*
9	0	anúncio de rota
10	0	descoberta de rota
11	0	TTL expirado
12	0	Cabeçalho IP com erros

Traceroute e ICMP

- Origem envia série de segmentos UDP para o destino.
  - Primeiro com TTL = 1.
  - Segundo com TTL = 2, etc.
  - Utiliza porta de destino pouco provável.
- Quando *n-ésimo* conjunto de datagramas chega ao *n-ésimo* roteador:
  - TTL expira, roteador descarta datagrama.
  - Envia mensagem ICMP reportando erro à origem (tipo 11, código 0).
  - Mensagem ICMP inclui nome e endereço IP do roteador.
- Quando mensagem ICMP chega, origem registra o RTT.
- **Critério de parada:**
  - Segmento UDP eventualmente chega ao destinatário.
  - Destinatário envia mensagem ICMP do tipo "porta inalcançável" (tipo 3, código 3).
  - Origem para.



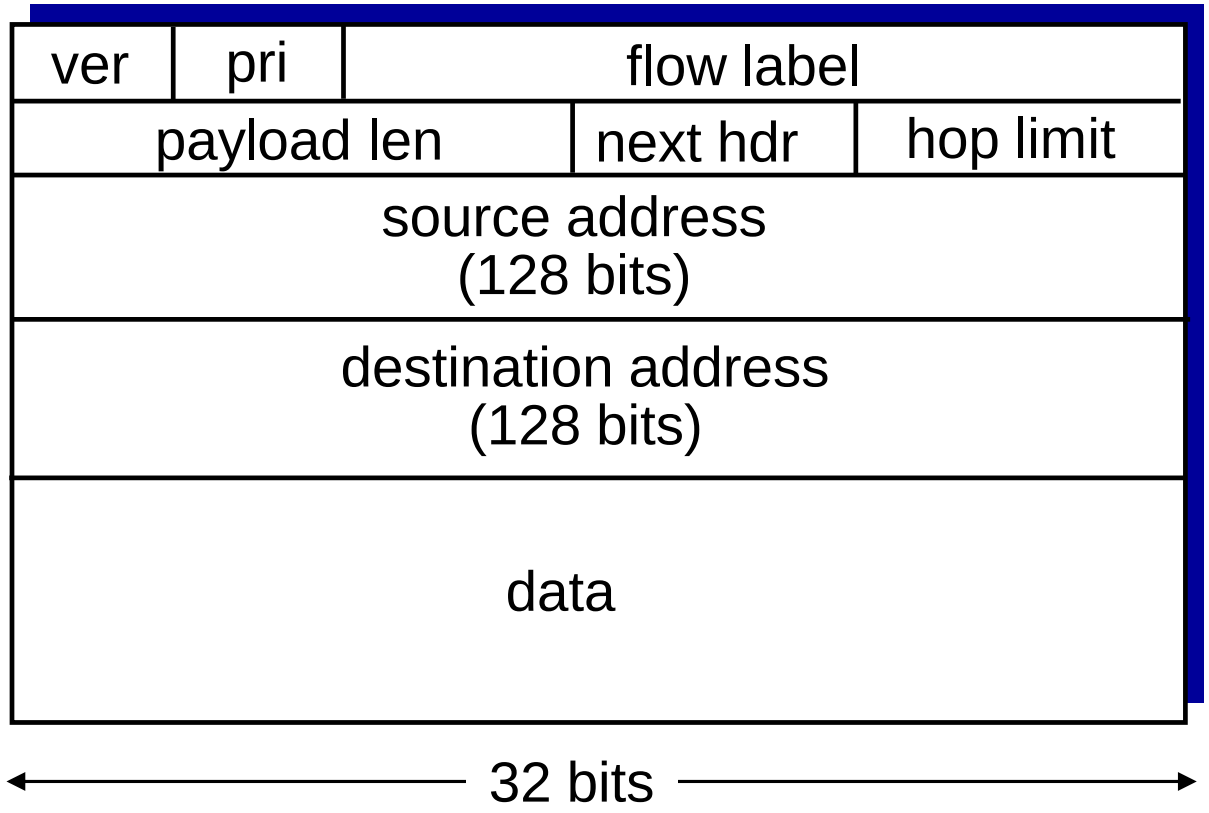
IPv6

IPv6: Motivação

- **Motivação inicial:** espaço de endereçamento de 32 bits será esgotado em breve.
  - *i.e.*, todos os endereços serão alocados.
- Motivações adicionais:
  - Formato do cabeçalho facilita e acelera o processamento/encaminhamento.
  - Alterações no cabeçalho para facilitar QoS.
- **Formato do datagrama IPv6**:
  - Cabeçalho de tamanho fixo, com 40 bytes.
  - Não permite fragmentação.

IPv6: Formato do Datagrama

- **pri:** identifica prioridade do datagrama em relação a outros gerados pela mesma origem.
- **flow label:** identifica datagramas pertencentes a um mesmo "fluxo" (conceito de fluxo não é bem definido).
- **next header:** identifica protocolo para a carga útil do pacote.

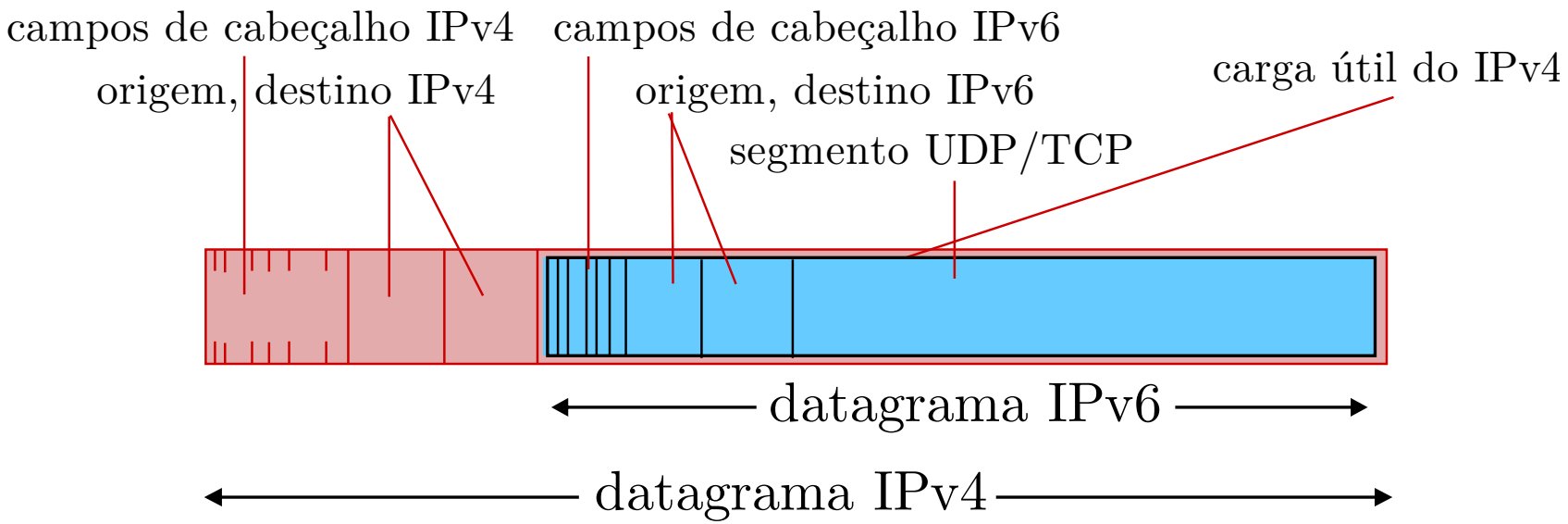


Outras Mudanças em Relação ao IPv4

- **Checksum:** completamente removido para reduzir tempo de processamento em cada salto.
- **Opções:** permitidas, mas **fora do cabeçalho**, indicado pelo valor do campo **next header**.
- **ICMPv6:** nova versão do ICMP.
  - Tipos de mensagem adicionais, *e.g.*, "Pacote Muito Grande".
  - Funções de gerenciamento de grupos multicast.

Transição do IPv4 para o IPv6

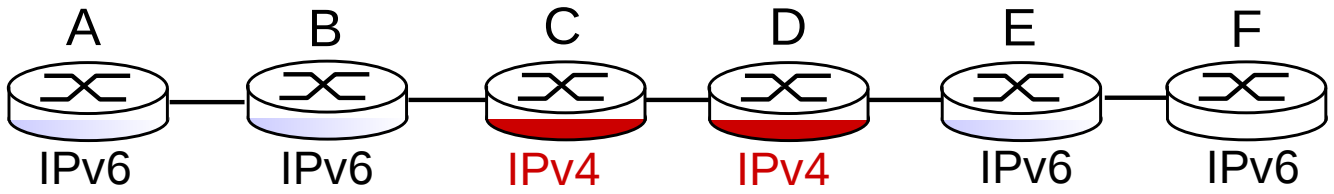
- Impossível atualizar todos os roteadores do mundo simultaneamente.
  - Não existe um “dia oficial de migração”.
  - Como a rede pode operar com roteadores IPv4 e IPv6 misturados?
- **Tunelamento:** datagramas IPv6 carregados como carga útil em datagramas IPv4 encaminhados por roteadores IPv4.



Tunelamento

visão lógica:

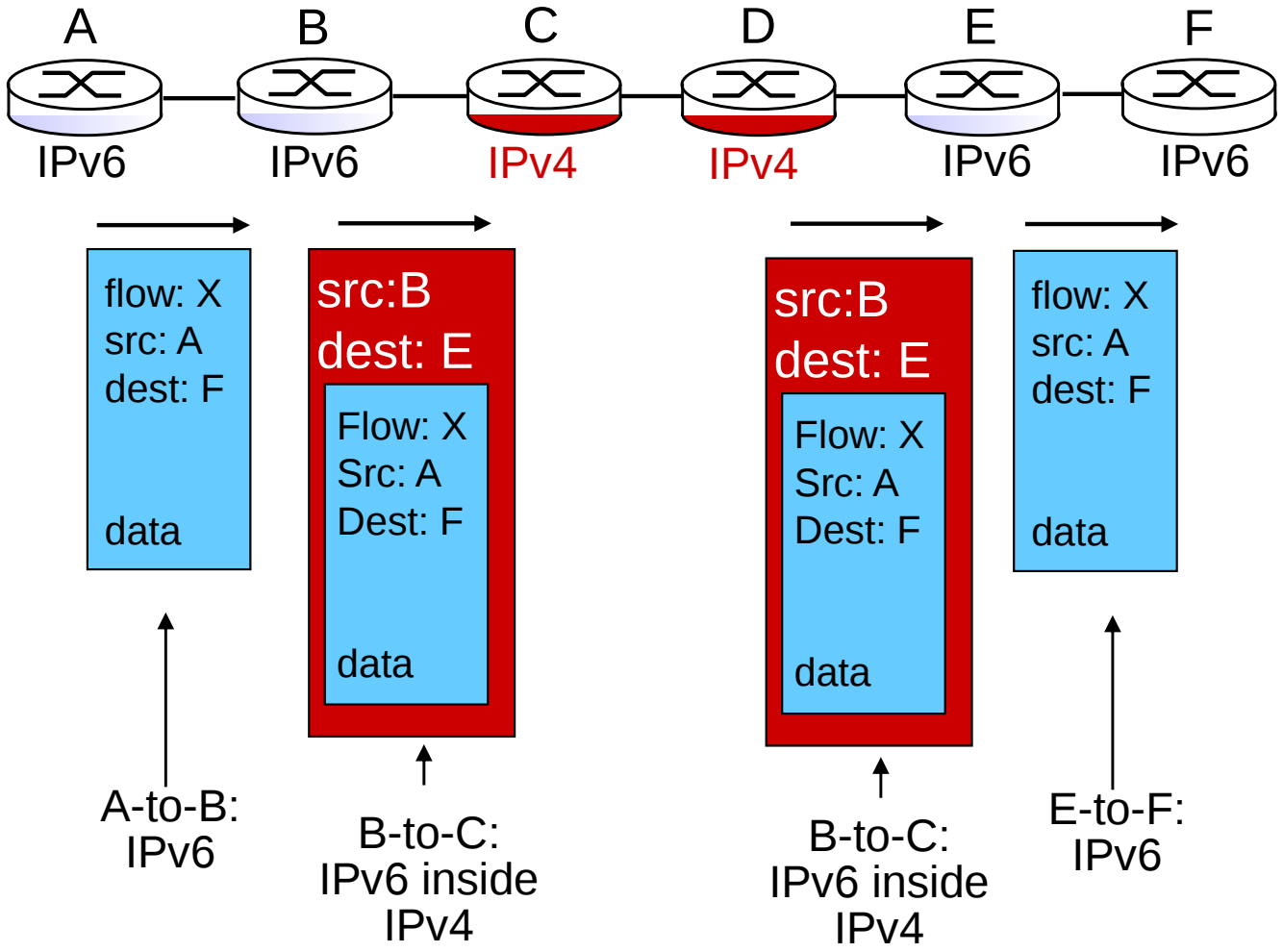
visão física:



Tunelamento

visão lógica:

visão física:

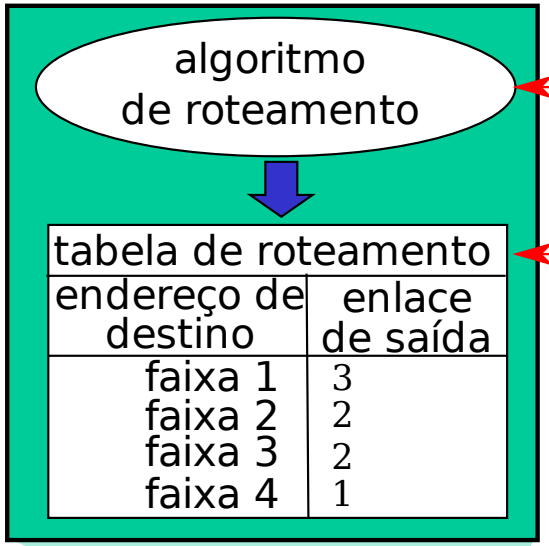


IPv6: Adoção

- Estimativas do US National Institute of Standards [2013]:
  - ~3% dos roteadores IP da indústria.
  - ~11% dos roteadores do governo americano.
- **Tempo (muito!) longo para implantação, uso.**
  - 20 anos e contando!
  - Pense nas mudanças no nível de aplicação nos últimos 20 anos: web, facebook, Netflix, ...
  - **Por quê?**

Roteamento: Introdução

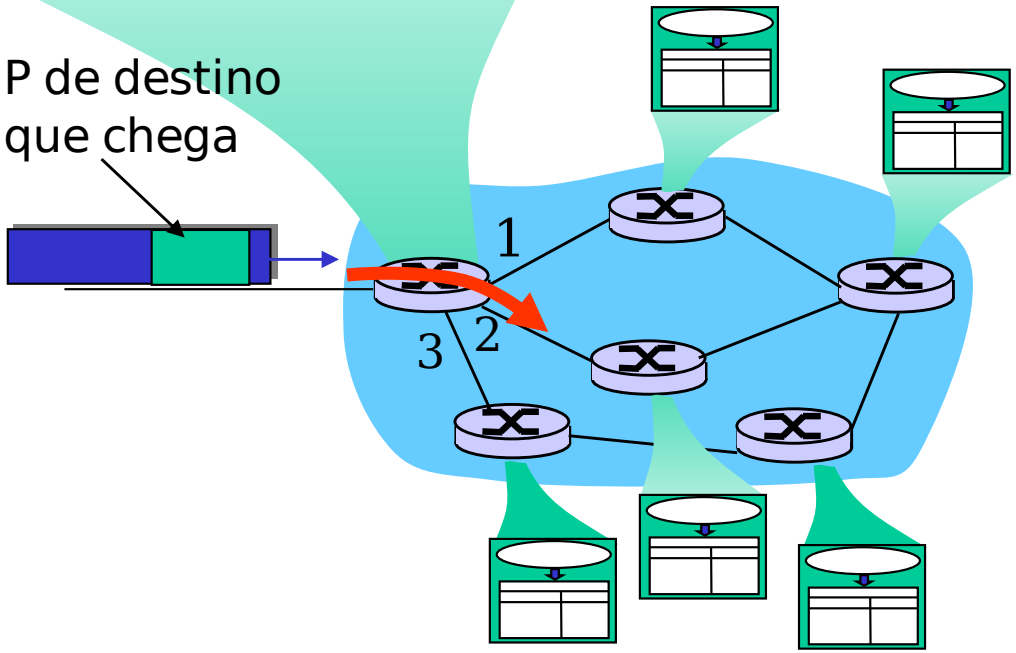
Sinergia entre Roteamento e Encaminhamento



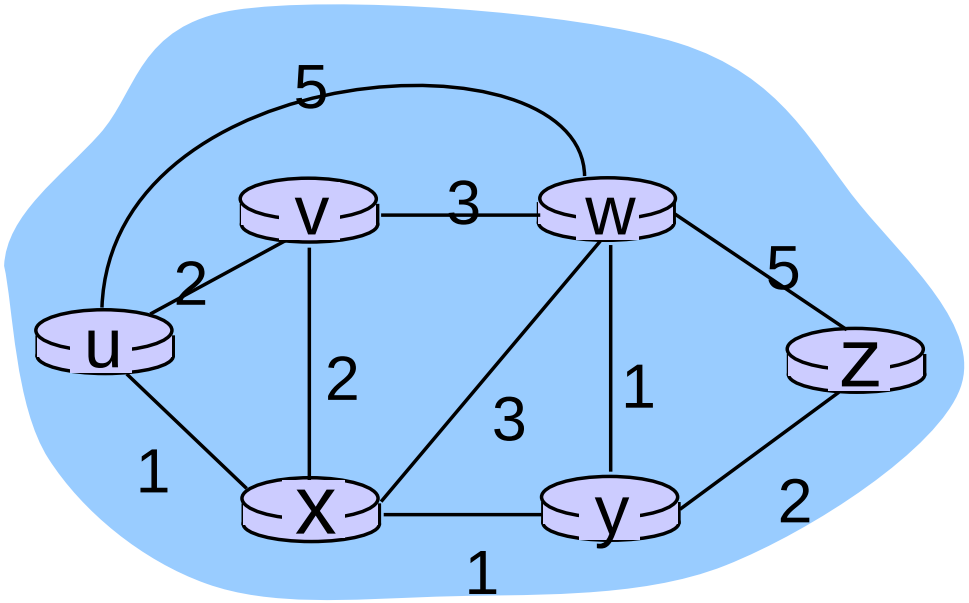
algoritmo de roteamento determina caminho fim-a-fim pela rede

tabela de roteamento determina encaminhamento local neste roteador

endereço IP de destino do pacote que chega



Abstração de Grafo

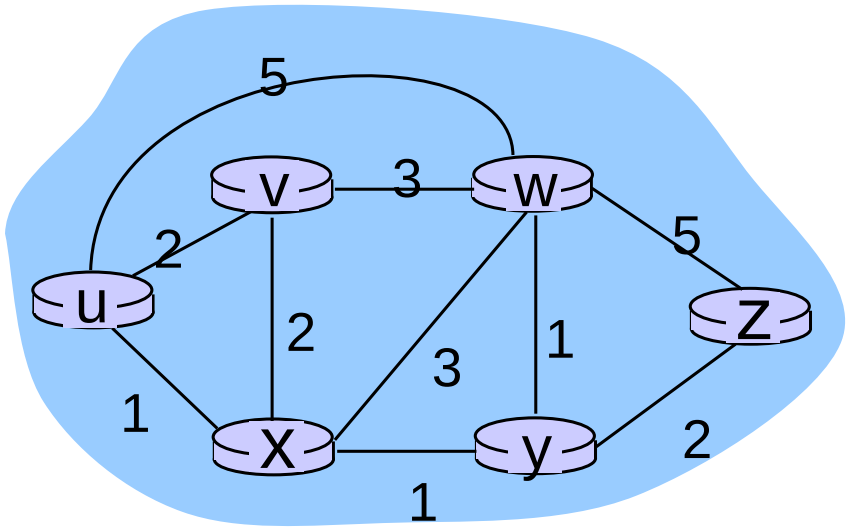


- Grafo: G = (N, E).
- N = conjunto de roteadores = {u, v, w, x, y, z}.
- E = conjunto de enlaces = {(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)}

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.

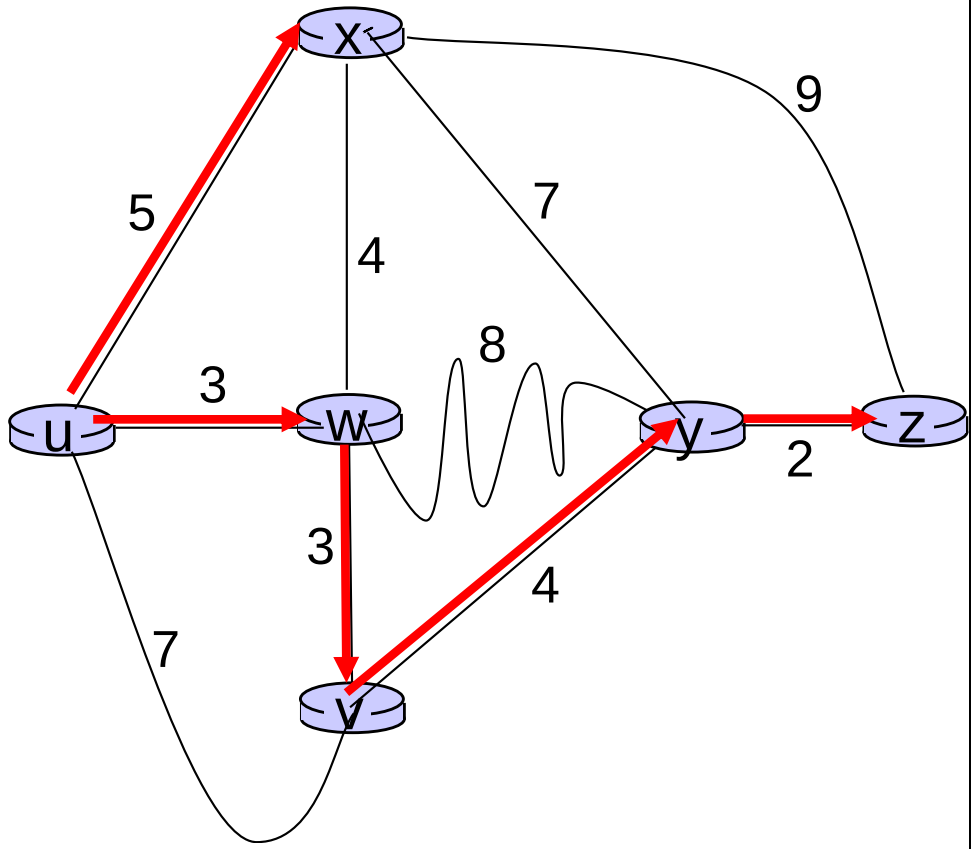
- **Notação:**
  - $c(x, y)$: custo do enlace do nó x para nó y.
    - infinito, se x e y não são vizinhos.
  - $D(v)$: custo atualmente conhecido para o caminho da origem até v.
  - $p(v)$: predecessor de v no caminho da origem para v.
  - N': conjunto de nós para os quais definitivamente conhecemos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

		$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
Passo	N'	$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
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					

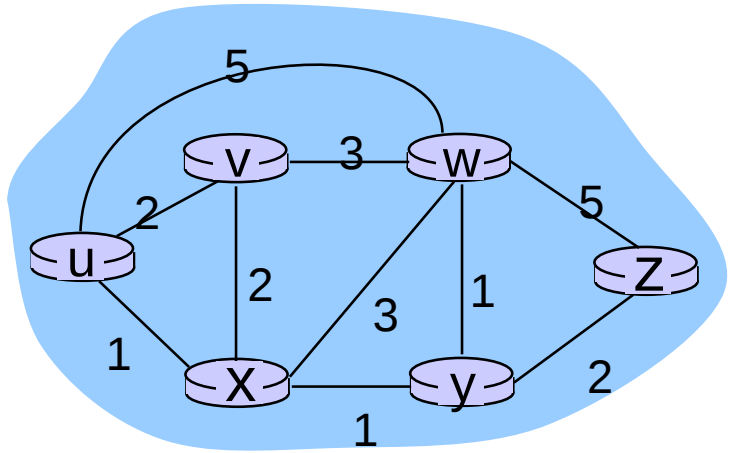


Notas

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

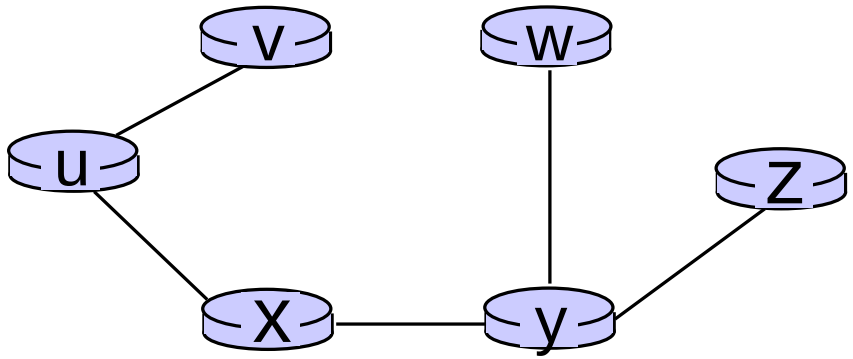
Algoritmo de Dijkstra: Outro Exemplo (I)

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:

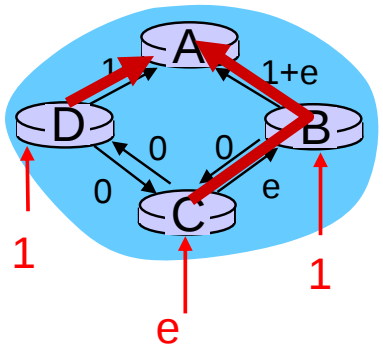


- Tabela de roteamento resultante:

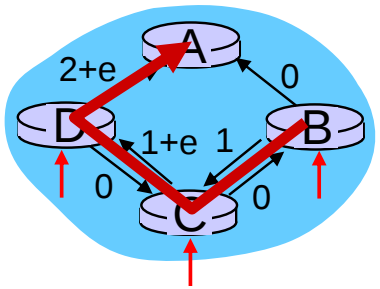
Destino	Enlace
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Algoritmo de Dijkstra: Discussão

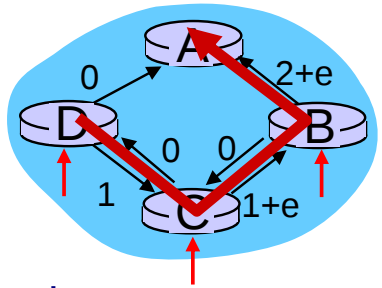
- **Complexidade do algoritmo:** com n nós.
  - Cada iteração: verifica todos os possíveis nós $w \notin N'$.
  - $\frac{n(n+1)}{2}$ comparações: $O(n^2)$.
  - Implementações mais eficientes são possíveis: $O(n \cdot log\ n)$.
- **Podem ocorrer oscilações:**
  - *e.g.* custos dos enlaces iguais a quantidade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



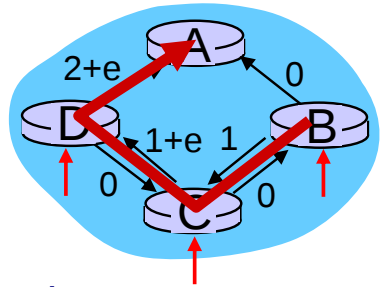
inicialmente



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