

# Módulo 4: Camada de Rede

## Objetivos do módulo:

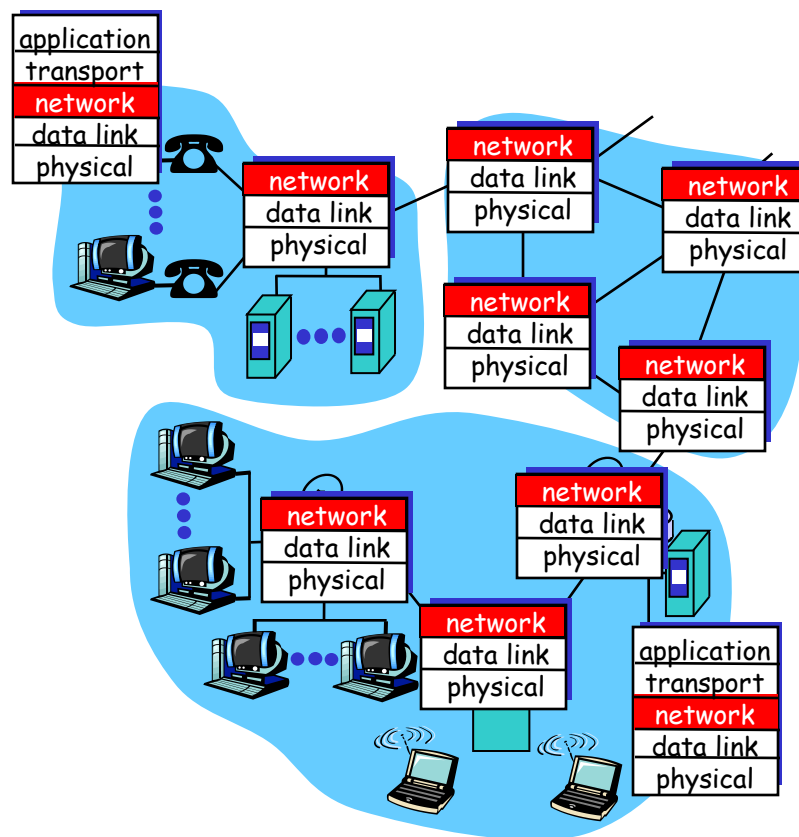
- ❑ Compreender os princípios por trás dos serviços da camada de rede:
  - Modelos de serviço da camada de rede
  - Encaminhamento (forwarding) versus roteamento
  - Como um roteador funciona
  - roteamento (seleção de caminho)
  - Lidando com a escalabilidade
  - Tópicos avançados: IPv6, mobilidade
- ❑ instanciação, implementação na Internet

# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# Camada de Rede

- ❑ transporta segmento do host emissor para o host receptor
- ❑ No lado emissor encapsula segmentos em datagramas
- ❑ No lado receptor, entrega segmentos para a camada transporte
- ❑ Protocolos da camada de rede em *todos* os hosts, roteador
- ❑ Roteador examina campos do cabeçalho dos datagramas IP que passam por ele



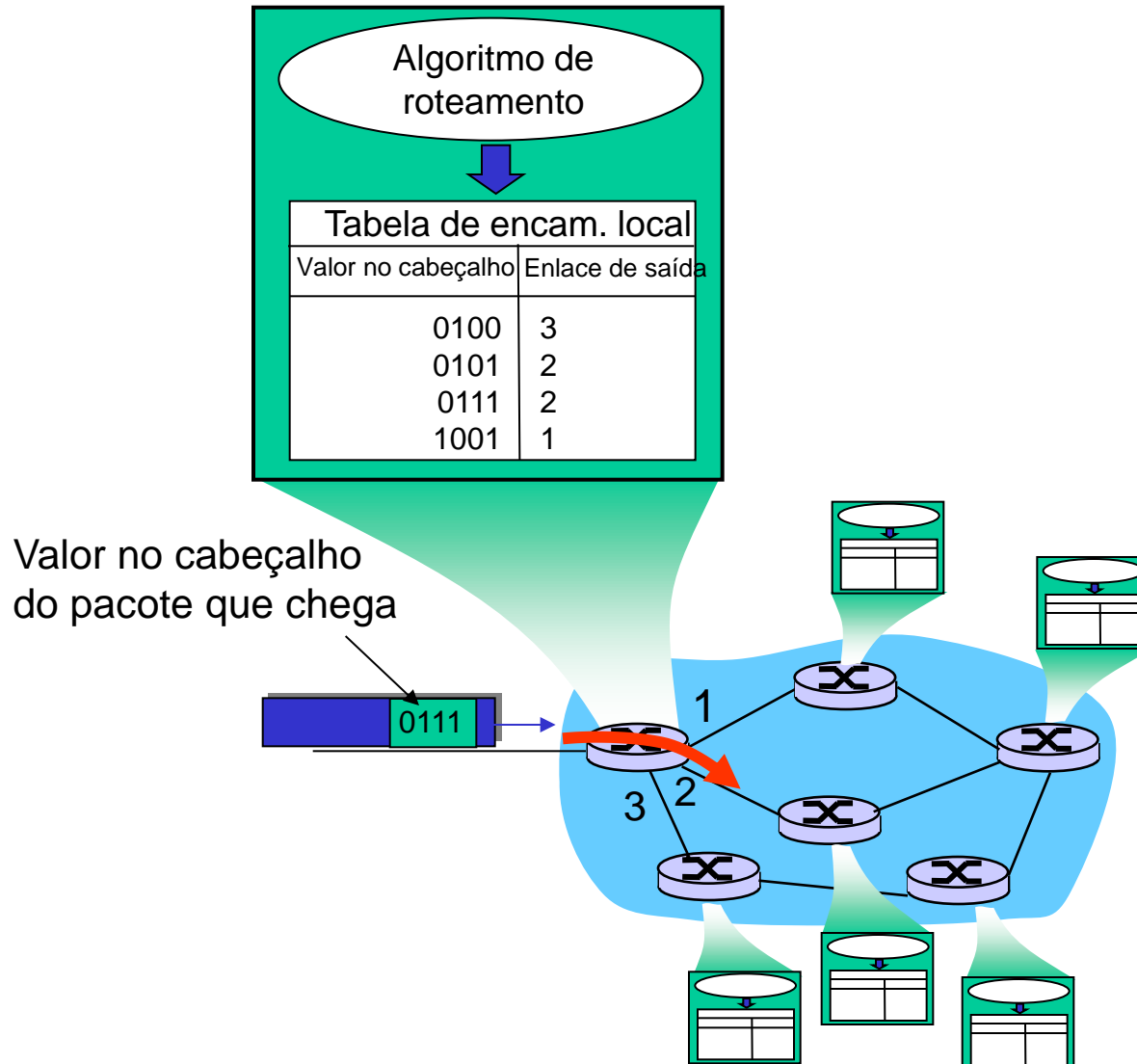
# Principais funções da camada de rede

- ❑ *forwarding*: encaminha pacotes que chegam ao roteador para a saída apropriada do roteador
- ❑ *roteamento*: determina rota a ser tomada por pacotes da fonte ao destino.
  - *Algoritmos de roteamento*

## analogia:

- ❑ *roteamento*: processo de planejamento de uma viagem da origem ao destino
- ❑ *forwarding*: processo de finalização completa de uma "escala" na viagem para o destino

# Ação e reação recíproca entre roteamento e forwarding (encaminhamento)



# Estabelecimento de conexão (virtual)

- ❑ 3ª função importante em algumas arquiteturas de rede:
  - ATM, frame relay, X.25
- ❑ Antes do envio do fluxo de datagramas, os dois hosts com auxílio dos roteadores intermediários estabelecem uma conexão virtual
  - Roteadores participam explicitamente do processo
- ❑ Serviço de conexão da camada de rede e da camada transporte:
  - **Rede:** entre dois hosts
  - **Transporte:** entre 2 processos

# Modelo de serviço de rede

**Q:** Qual o *modelo de serviço* para transporte direcionado dos datagramas do emissor para o receptor?

## Exemplos de serviço para datagramas individuais:

- ❑ Entrega garantida
- ❑ Entrega garantida com menos de 40 ms de atraso

## Exemplos de serviços para fluxo de datagramas:

- ❑ Entrega de datagramas na ordem correta
- ❑ Banda passante mínima garantida para o fluxo
- ❑ Restrições sobre mudança de espaçamento entre pacotes sucessivos (**jitter** = variação de atraso)

# Modelos de serviço da camada de rede:

Arquitetura de rede	Modelo de serviço	Banda	Garantias ?			Feedback sobre congestionamento
			Perda	Ordem	Timing	
Internet	best effort	não	não	não	não	não (inferido via perdas)
ATM	CBR	Taxa constante	sim	sim	sim	sem congestionamento
ATM	VBR	Taxa garantida	sim	sim	sim	sem congestionamento
ATM	ABR	Mínimo garantido	não	sim	não	sim
ATM	UBR	não	não	sim	não	não



# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# Serviço com e sem conexão da camada de rede

- ❑ Redes de datagrama provêem serviço de camada de rede sem conexão
- ❑ Redes de circuitos virtuais (VC) provêem serviço de rede orientado à conexão
- ❑ Análogo aos serviços da camada transporte, mas:
  - **Serviço:** host para host
  - **Sem escolha:** rede provê ou um ou outro
  - **Implementação:** no núcleo

# Circuitos Virtuais (VC)

“caminho da fonte para o destino se comporta de forma semelhante ao circuito da linha telefônica”

- performance-wise
- Ações da rede ao longo do caminho fonte-destino

- estabelecimento (teardown-encerramento) para cada chamada antes (depois) que os dados possam ser (foram) enviados
- Cada pacote carrega um identificador de circuito virtual (não o endereço do host de destino)
- *Todos os roteadores no caminho fonte-destino mantêm “estado” para cada conexão passando*
- enlace, recursos do roteador (banda, buffers) podem ser alocados para o circuito virtual

# Implementação do VC

Um VC consiste em:

1. Caminho da fonte ao destino
  2. Números de VCs, 1 número para cada enlace ao longo do caminho
  3. Entradas nas tabelas de encaminhamento dos roteadores ao longo do caminho
- ❑ Pacote pertencendo a um VC carrega consigo um número de VC.
  - ❑ O número de VC deve ser mudado em cada enlace.
    - Novo número de VC obtido da tabela de encaminhamento

# Tabela de Encaminhamento

Número do VC

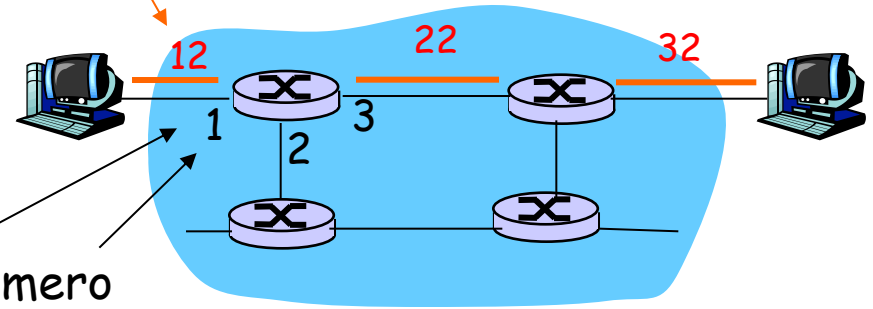


Tabela de encaminhamento  
no roteador indicado:

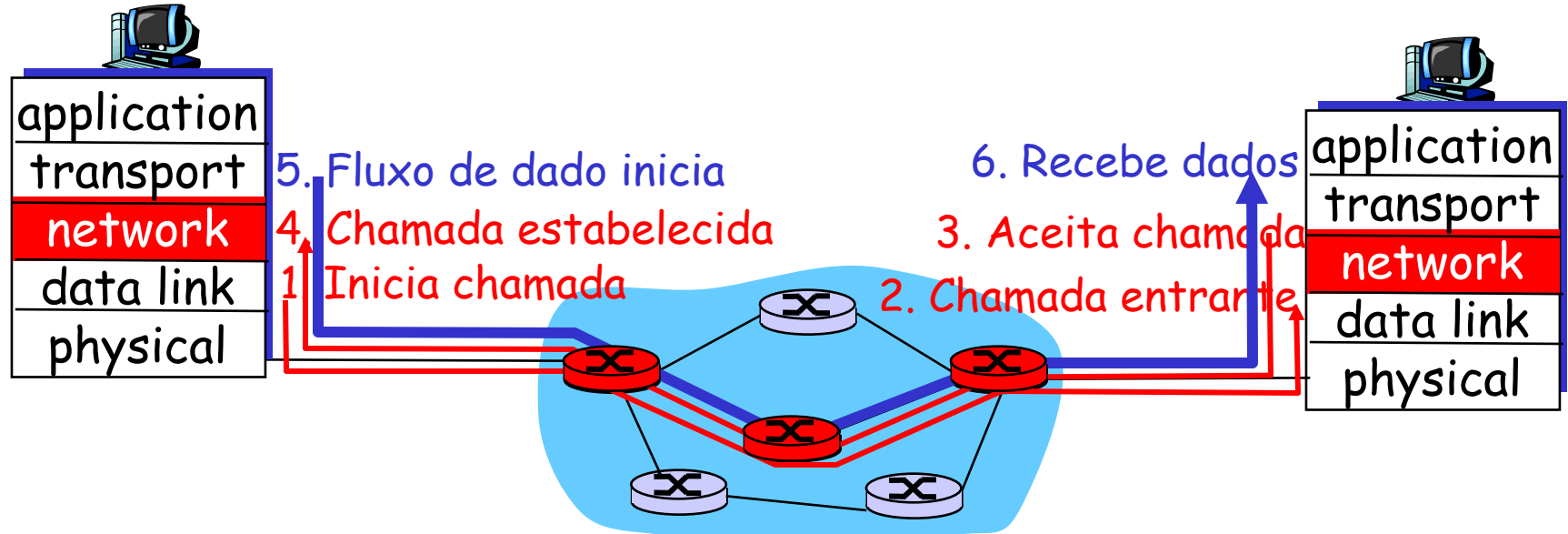
número  
da interface

Interface de entrada	# do VC de entrada	interface de saída	# do VC de saída
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

**Roteadores mantêm informação de estado da conexão!**

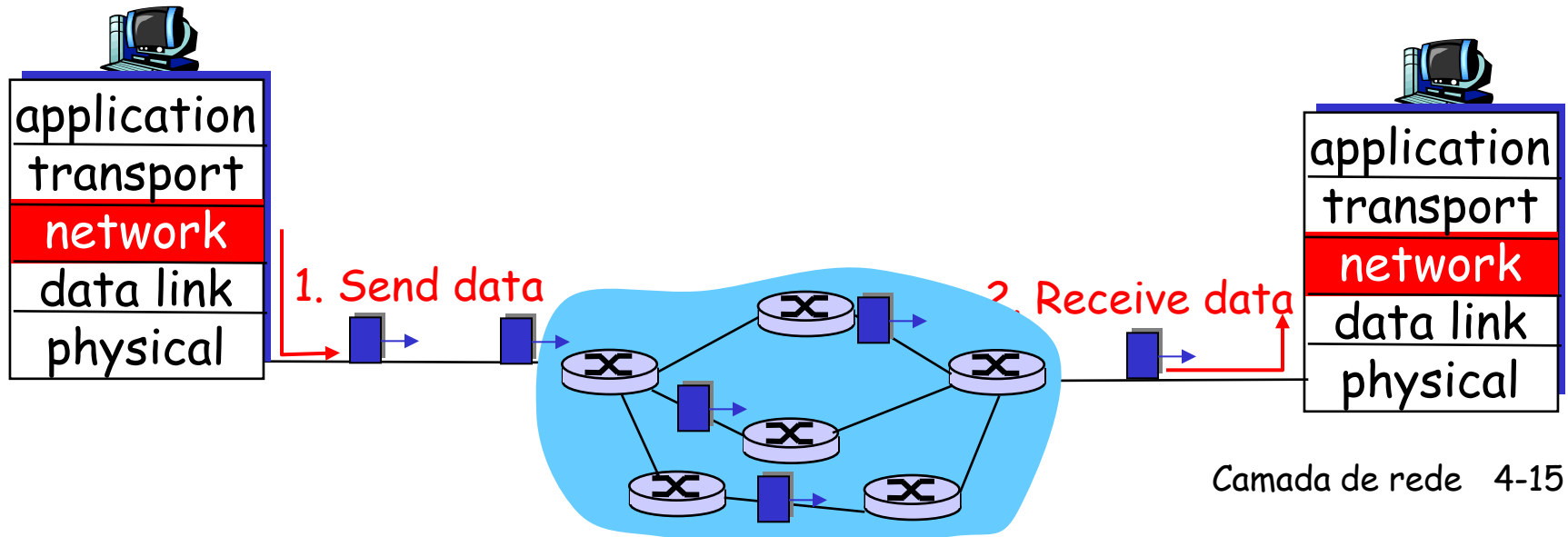
# Circuitos virtuais: protocolos de sinalização

- ❑ usados para estabelecer(setup), manter, encerrar (teardown) VC
- ❑ usados em redes ATM, frame-relay, X.25
- ❑ Não são usados na Internet atual



# Redes de datagramas

- ❑ Não há estabelecimento de conexão/chamada na camada de rede
- ❑ roteadores: não há estado sobre conexões fim-à-fim
  - Não existe o conceito de "conexão" na camada de rede
- ❑ pacotes são encaminhados com base no endereço de destino
  - pacotes entre mesmo par fonte-destino podem tomar caminhos diferentes



# Tabela de encaminhamento

4 bilhões de entradas possíveis

Range do endereço de destino	<u>Link Interface</u>
11001000 00010111 00010000 00000000 até 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 até 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 até 11001000 00010111 00011111 11111111	2
caso contrário	3



# Prefixo mais longo correspondente

<u>Prefix Match</u>	<u>Link interface</u>
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
caso contrário	3

## Exemplos

End. destino: 11001000 00010111 00010110 10100001 **Por qual interface?**

End. destino: 11001000 00010111 00011000 10101010 **Por qual interface?**

# Redes de Datagramas ou VCs: Por quê?

## Internet

- ❑ dados trocados entre computadores
  - Serviço "elástico", nenhum requisito de tempo estrito
- ❑ end systems "inteligentes"(computadores)
  - Pode se adaptar, realizar controle, recuperação de erros
  - Simplicidade dentro da rede, complexidade nas "bordas"
- ❑ Diferentes tipos de enlace
  - Características diferentes
  - Serviço uniforme difícil

## ATM

- ❑ Evolução do telefone
- ❑ Conversação humana:
  - Tempo estrito, requisitos de confiabilidade
  - Necessário para serviços garantidos
- ❑ end systems "burros"
  - telefones
  - complexidade dentro da rede

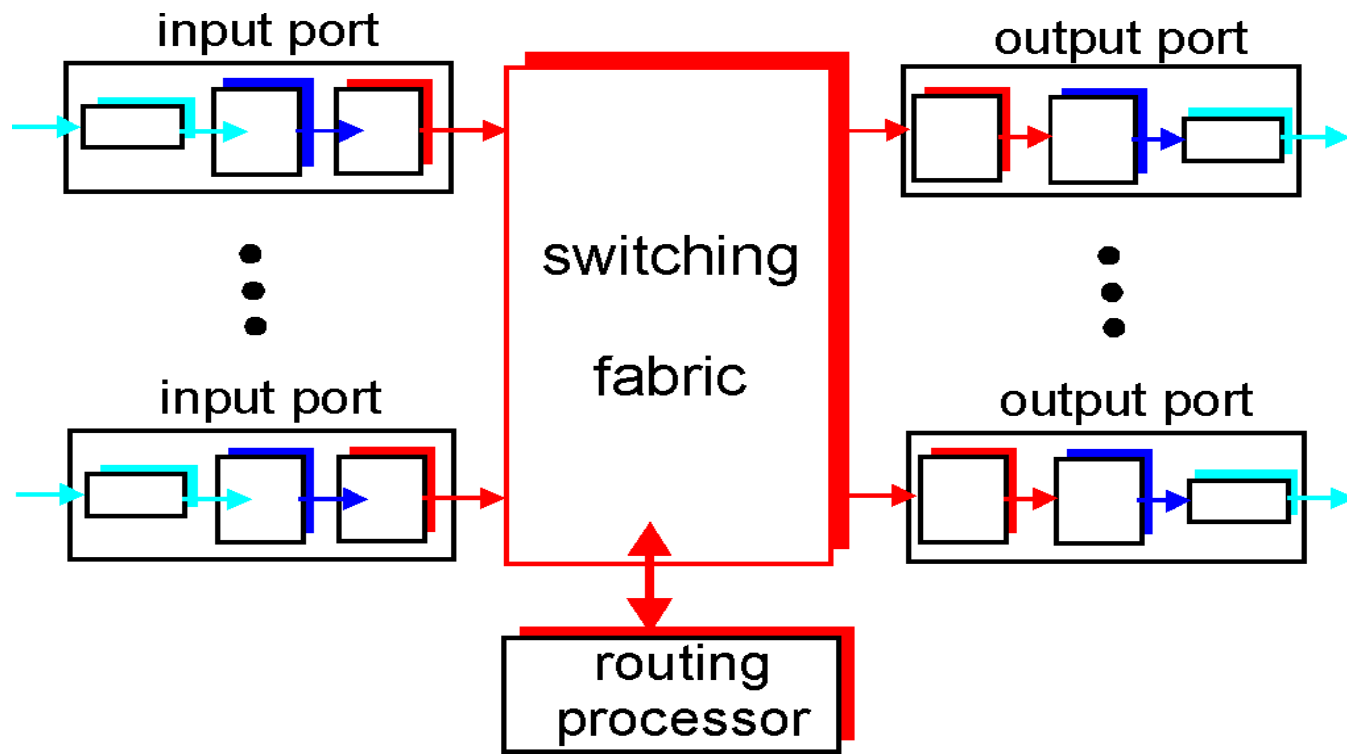
# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

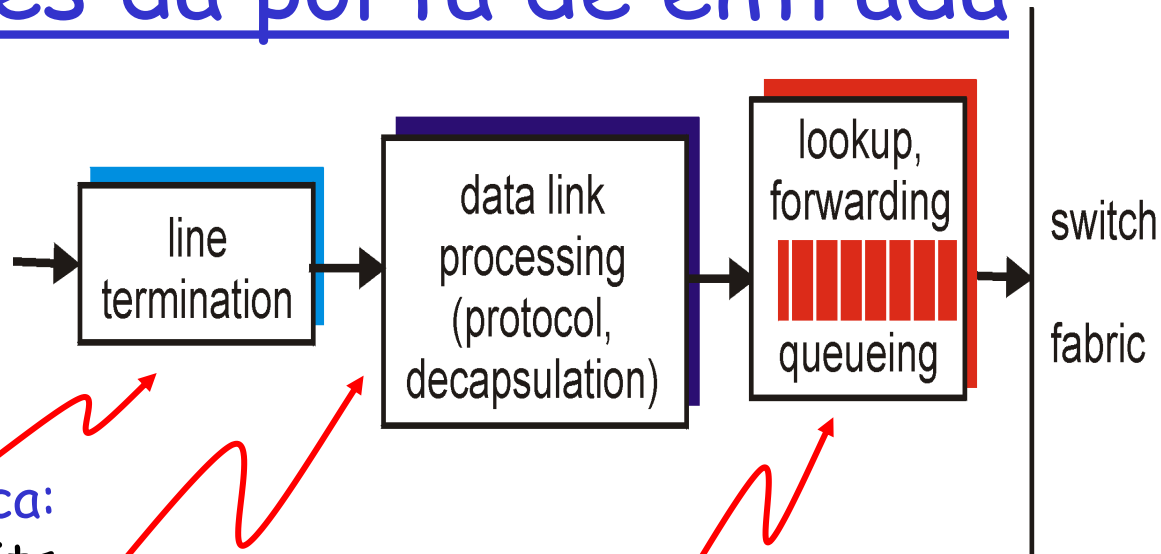
# Visão geral da arquitetura de um roteador

2 funções principais do roteador:

- ❑ Executar algoritmos/protocolos de roteamento (RIP, OSPF, BGP)
- ❑ *encaminhar* datagramas do enlace de entrada para o enlace de saída



# Funções da porta de entrada



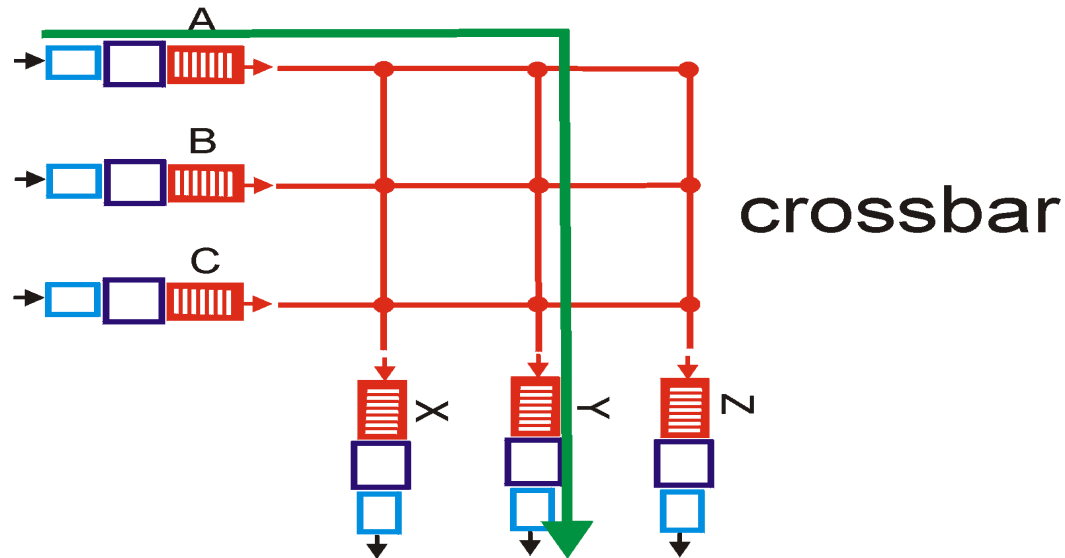
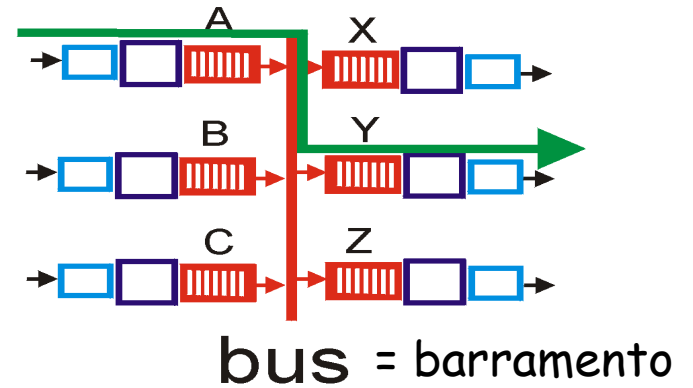
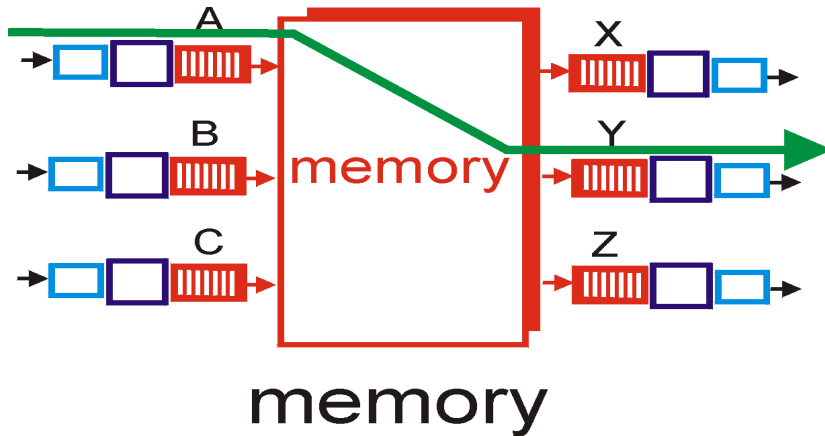
Camada física:  
Recepção de bits

Camada enlace:  
e.g., Ethernet  
(módulo 5)

## Comutação descentralizada:

- ❑ Dado destino de datagrama, procurar porta de saída usando a tabela de encaminhamento na memória da porta de entrada
- ❑ objetivo: completar processamento na porta de entrada na "velocidade da linha"
- ❑ fila (queuing): par o caso de datagramas chegarem mais rápido que a taxa de encaminhamento no switch fabric

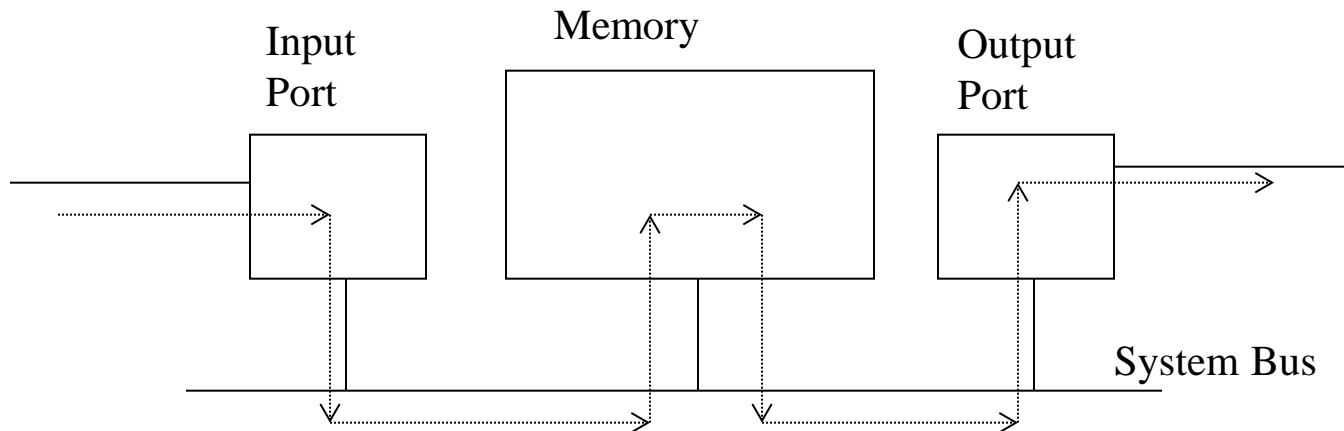
# 3 tipos de tecnologia de comutação



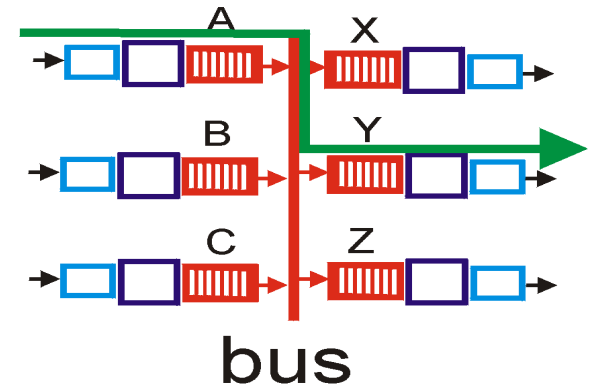
# Comutação Via Memória

## Roteadores de 1ª geração:

- ❑ computadores tradicionais com comutação sob controle direto da CPU
- ❑ Pacote copiado para a memória do sistema
- ❑ velocidade limitada pela banda passante da memória (cada datagrama cruza dois barramentos)



# Comutação via Barramento



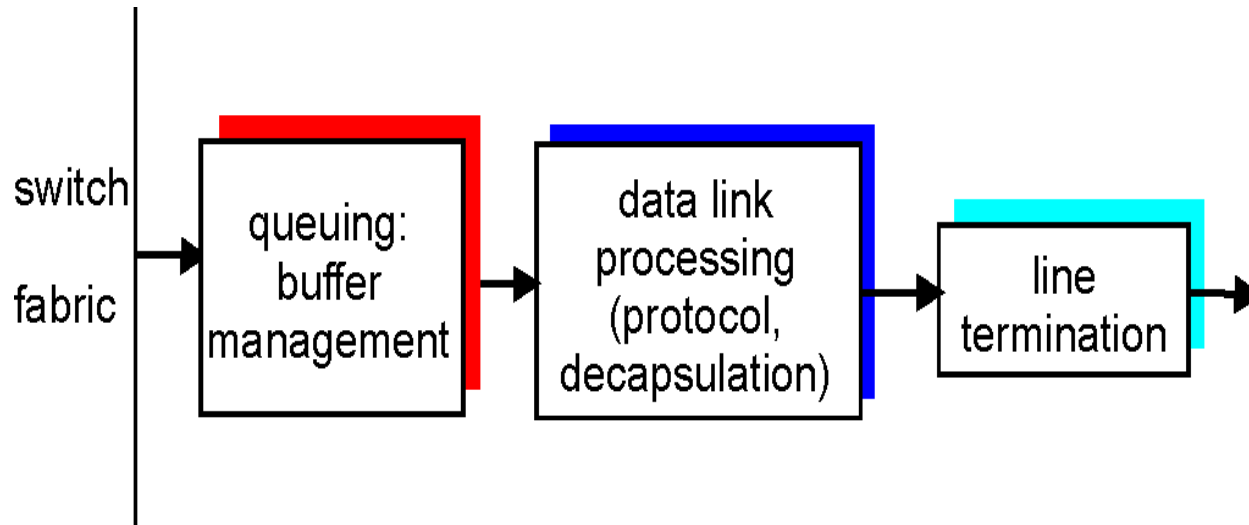
- ❑ datagrama da memória da porta de entrada para a memória da porta de saída via barramento compartilhado
- ❑ **bus contention:** velocidade de comutação limitada pela banda passante do barramento
- ❑ Barramento de 1 Gbps, Cisco 1900: velocidade suficiente para roteadores de acesso e de empresas (inadequado para acesso regional e para o backbone)



## Comutação Via uma rede de Interconexão (interna ao roteador)

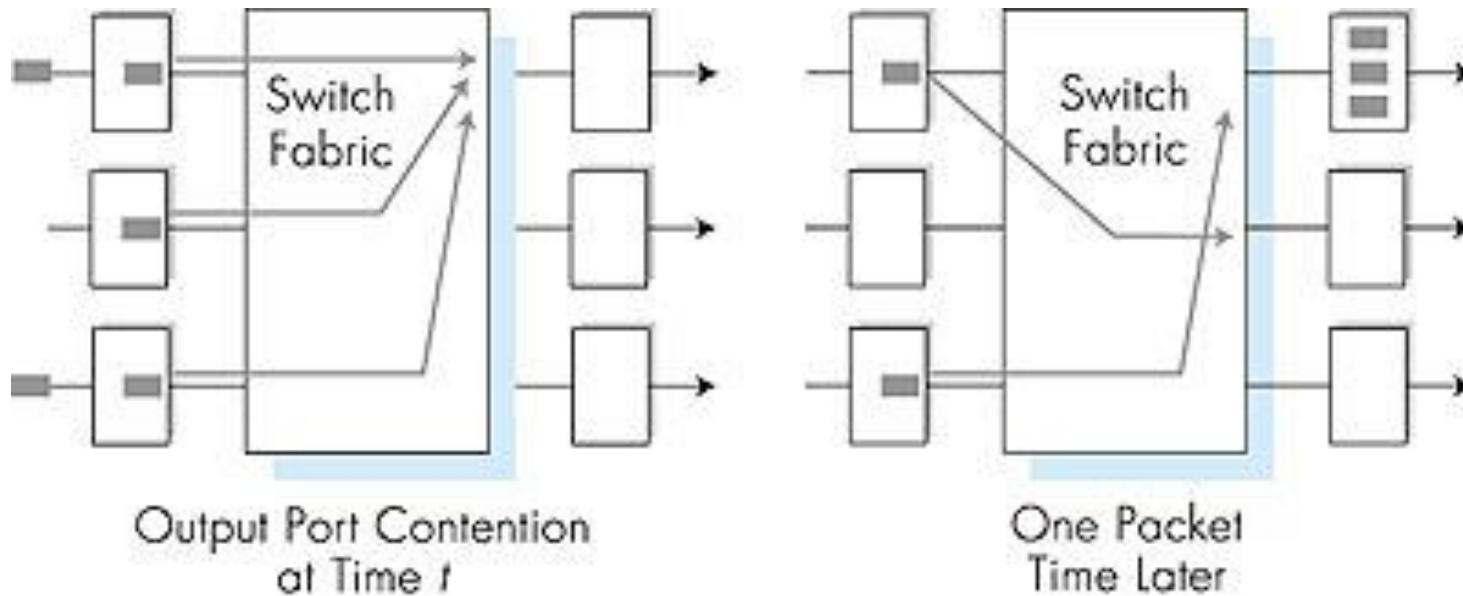
- ❑ “vencer” limitações de banda do barramento
- ❑ Redes Banyan, redes de interconexão inicialmente desenvolvidas para conectar processadores em sistemas multiprocessados
- ❑ Design avançado: fragmentação de datagramas em células de tamanho fixo, comuta células através da equipamento.
- ❑ Cisco 12000: comuta Gbps através de rede de interconexão

# Portas de saída



- ❑ **Buffering (fila)** requerido quando datagramas chegam do "switch fabric" mais rápido que a taxa de transmissão
- ❑ **Disciplina de escalonamento** escolhe datagramas da fila a serem transmitidos (prioridades)

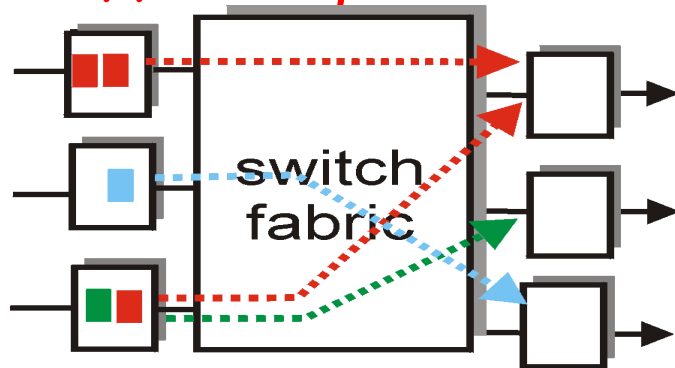
# Fila da porta de saída



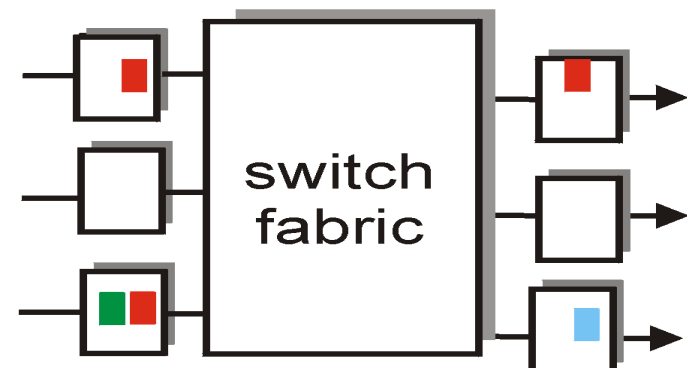
- ❑ "Búferização" quando taxa de chegada através do comutador excede a velocidade da linha de saída
- ❑ *Queueing/enfileiramento (atraso) e perda devido ao transbordamento (overflow) do buffer da porta de saída!*

# Fila da porta de entrada

- ❑ "Fabric" mais lenta que portas de entrada combinadas -> enfileiramento pode ocorrer nas filas de entrada
- ❑ **Head-of-the-Line (HOL) blocking:** datagrama enfileirado na frente da fila previne outros de serem encaminhados
- ❑ **atraso de fila e perda devido ao transbordamento do buffer da porta de entrada!**



output port contention  
at time t - only one red  
packet can be transferred



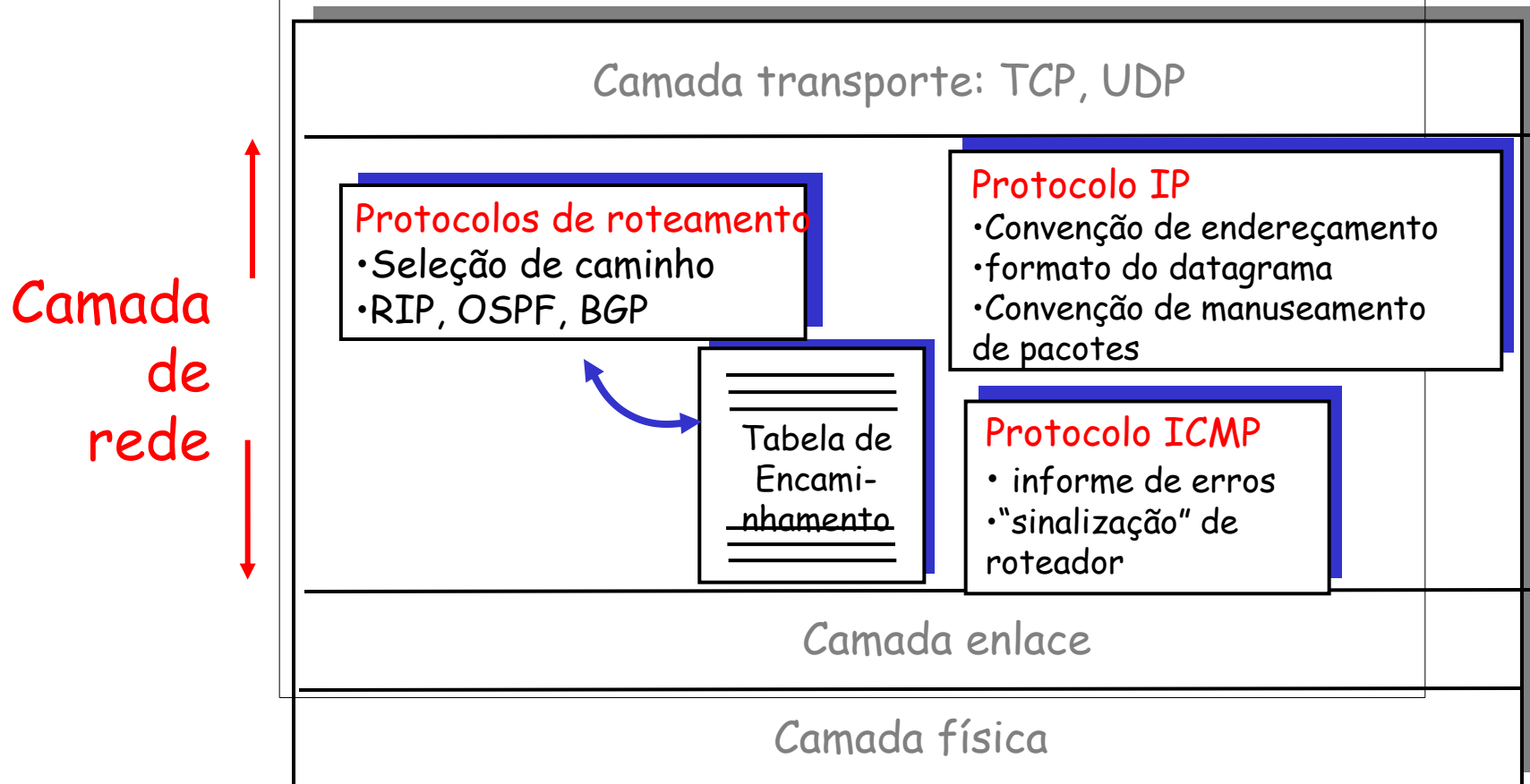
green packet  
experiences HOL blocking

# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# A camada de rede da Internet

Host, roteador -> funções da camada de rede:



# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# Formato do datagrama IP

No. da versão do protocolo IP

Tamanho do cabeçalho (bytes)

"tipo" de dados

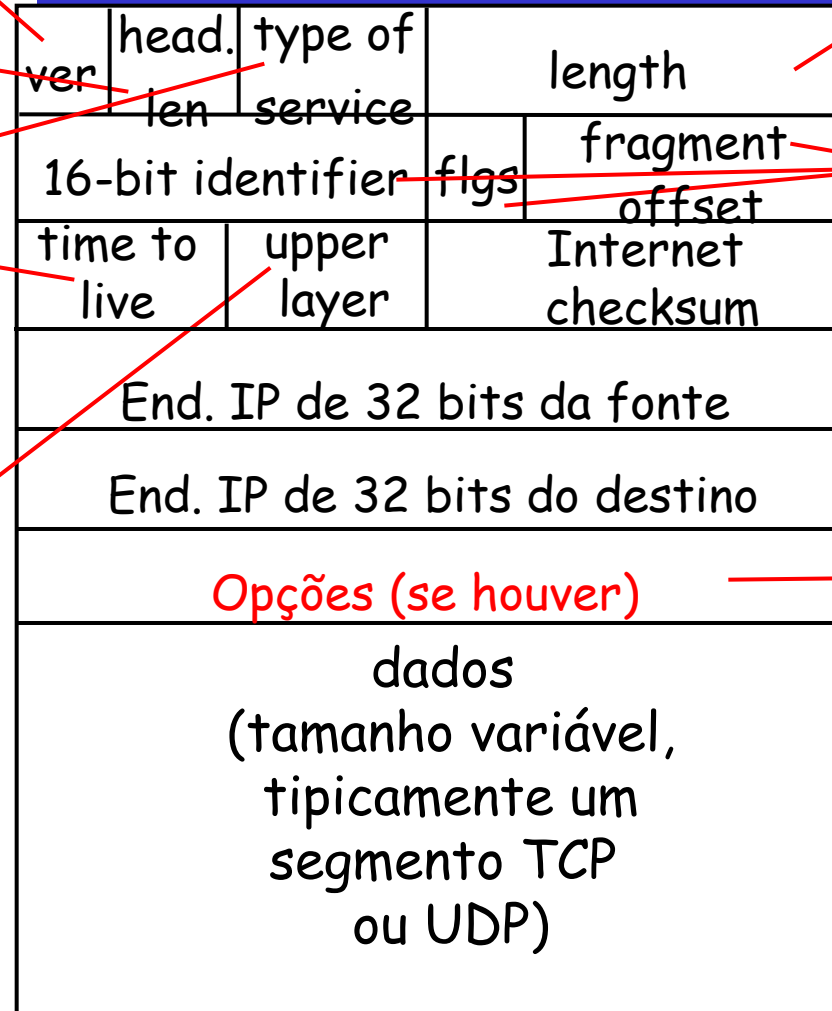
No. máx de saltos restantes (decrementado em cada roteador)

Protocolo da camada superior para o qual entregar o payload

## Qual o overhead com TCP/IP?

- 20 bytes do TCP
- 20 bytes do IP
- = 40 bytes + overhead da camada aplicação

← 32 bits →



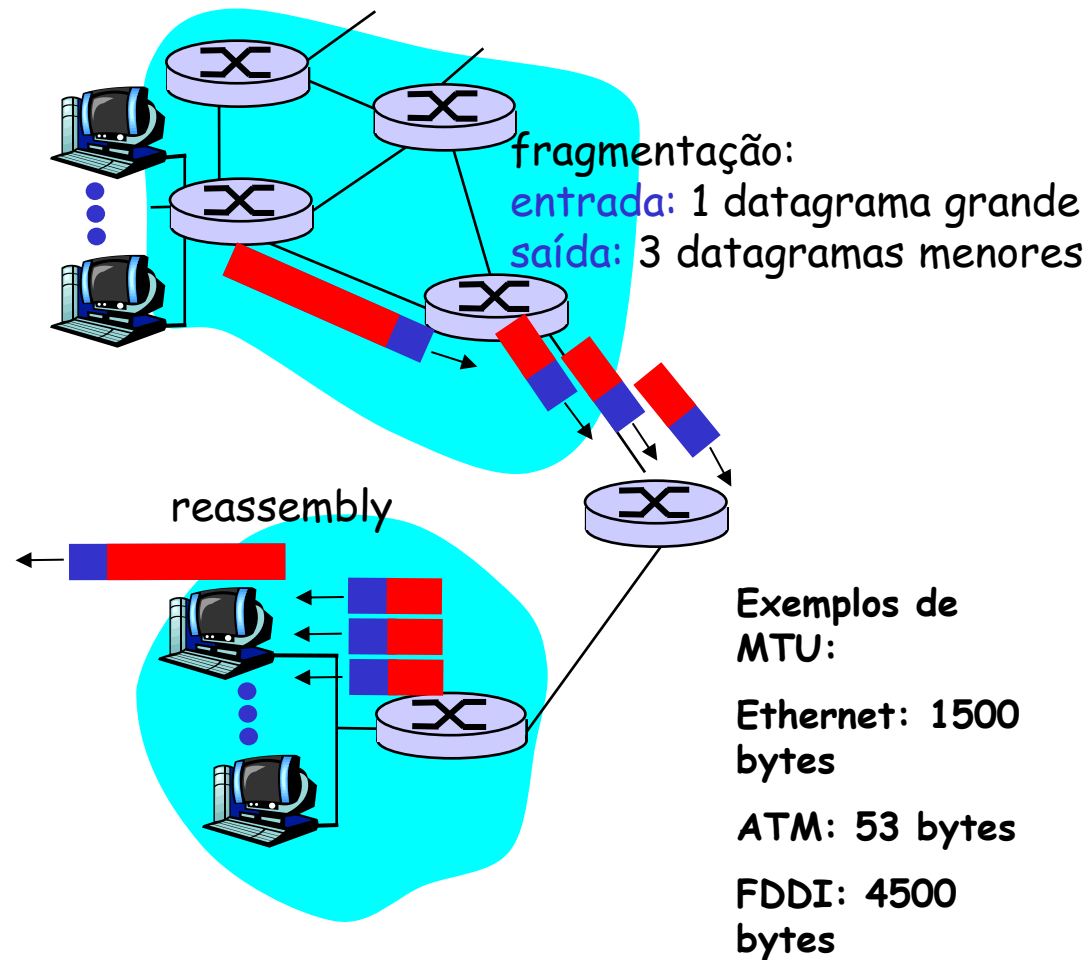
tamanho total do datagrama (bytes para fragmentação/remontagem)

E.g. timestamp (estampa de tempo), guarda rota tomada, especifica lista de roteadores a visitar.



# Fragmentação & Remontagem IP

- ❑ Enlace da rede possui MTU (tamanho máx. de transferência) - maior quadro possível no enlace.
  - diferentes tipos de enlaces, diferente MTUs
- ❑ datagrama IP "grande" dividido ("fragmentado") dentro da rede
  - 1 datagrama torna-se vários datagramas
  - "remontados" somente no destino final
  - Bits do cabeçalho IP usados para identificar, ordenar fragmentos relacionados



# Fragmentação & Remontagem IP

## Exemplo

- ❑ Datagrama de 4000 bytes (total)
- ❑ MTU = 1500 bytes

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

1 datagrama grande torna-se vários datagramas menores

1480 bytes no campo de dados

offset =  $1480/8$

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=185	

	length	ID	fragflag	offset	
	=1040	=x	=0	=370	

# Módulo 4: Camada de Rede

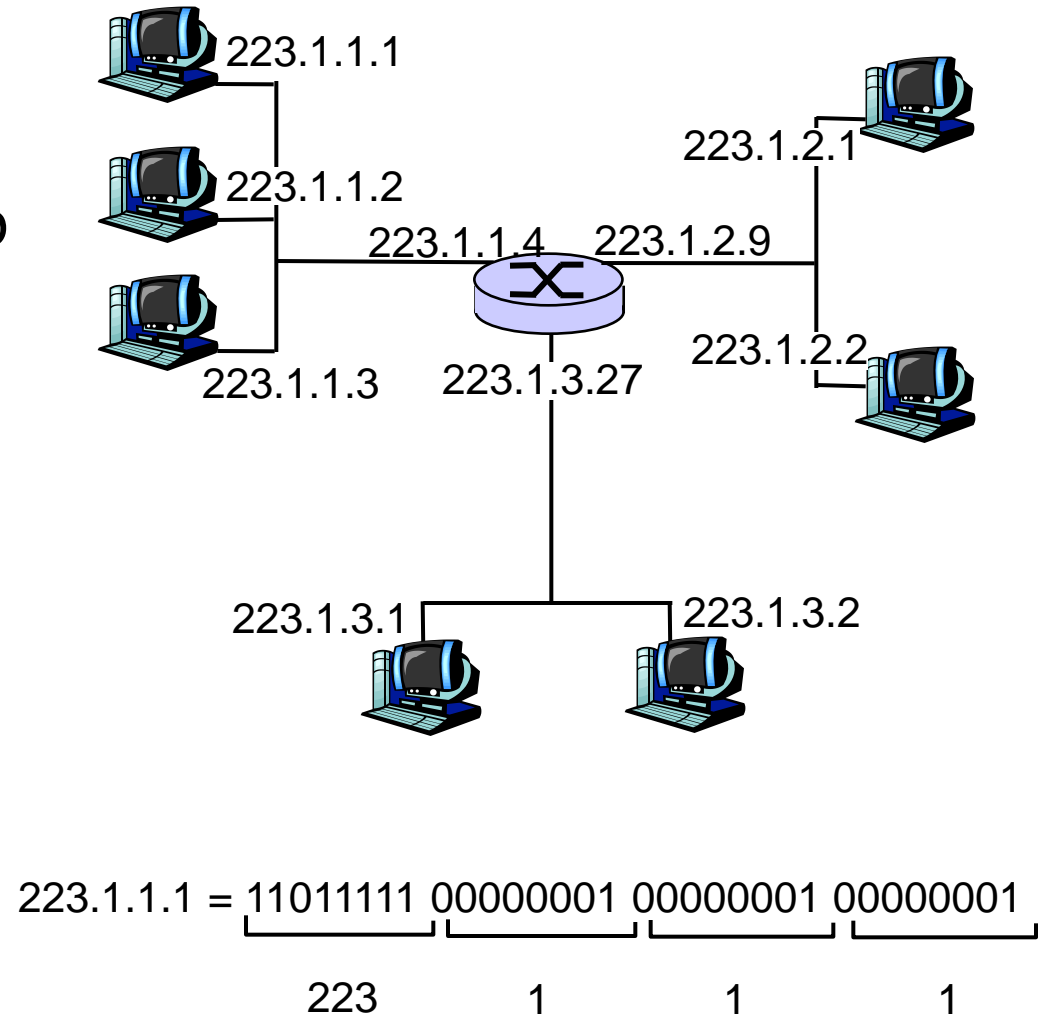
- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# Endereçamento IP: introdução

- A hierarquia de endereçamento IP:
  - Por questões de eficiência, o endereço IP é definido por duas partes, prefixo e sufixo;
  - O prefixo é responsável por determinar a rede que o computador está acoplado;
  - O sufixo é responsável por identificar um computador acoplado em cada rede;

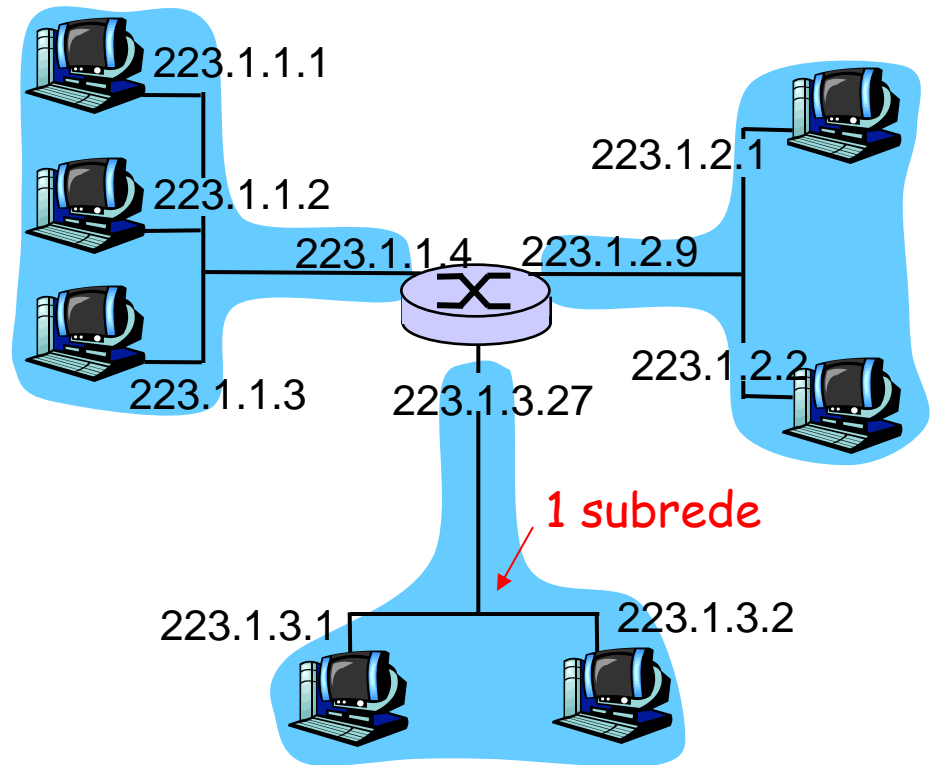
# Endereçamento IP: introdução

- **Endereço IP:**  
identificador de 32 bits para *interface* do host, roteador
- **interface:** conexão entre host/roteador e enlace físico
  - roteadores  
tipicamente possuem múltiplas interfaces
  - host tipicamente possui uma única interface
  - 1 endereço IP por interface



# Subredes

- Endereço IP:
  - Parte da subrede (bits de ordem mais elevada)
  - Parte do host (bits de ordem mais baixa)
- *O que é uma subrede?*
  - interfaces de dispositivos com mesma parte de subrede do endereço IP
  - Dispositivos podem fisicamente alcançar os outros sem ajuda de um roteador

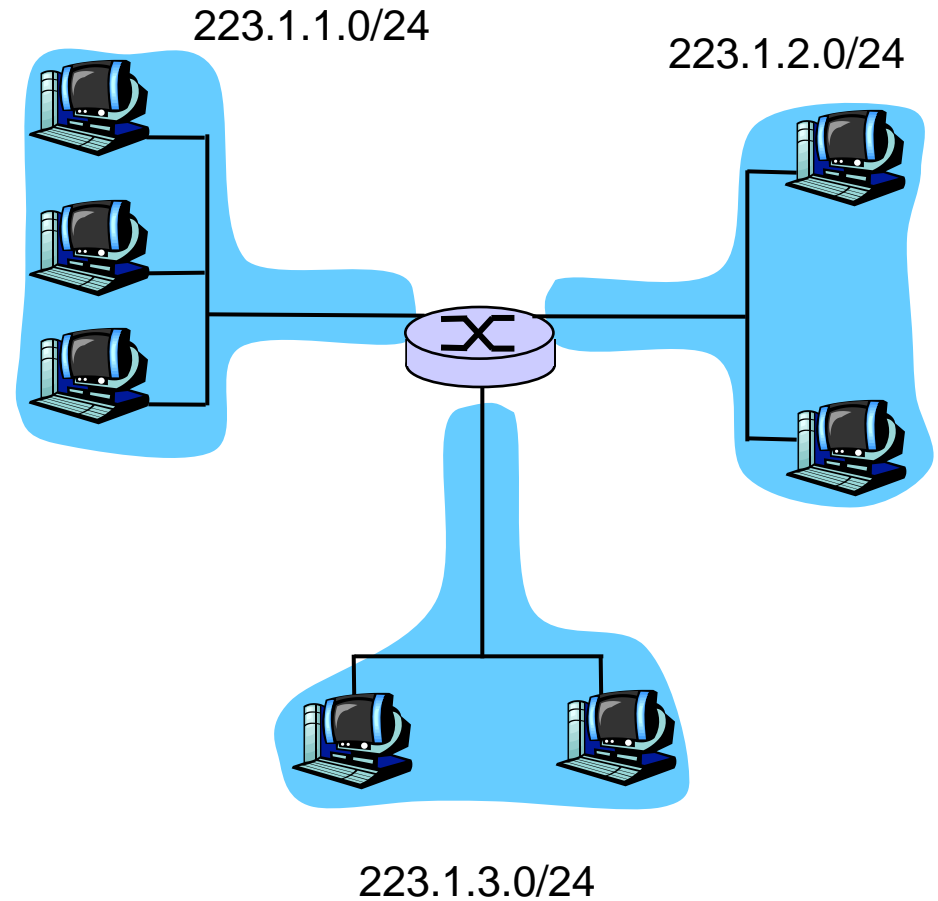


rede formada por 3 subredes

# Subredes

## Receita

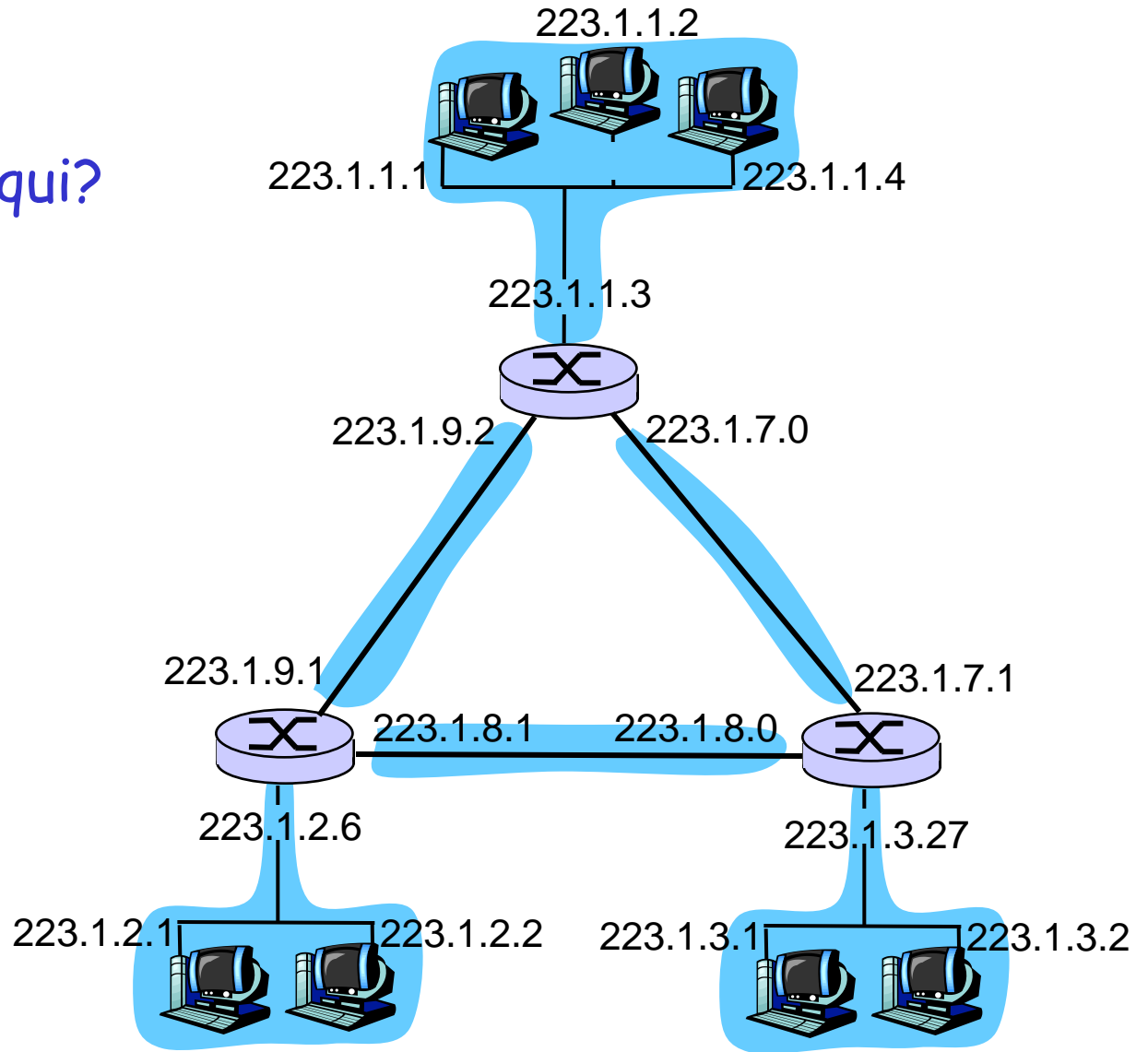
- Para determinar a subrede, retirar cada interface do seu host ou roteador, criando ilhas de redes isoladas. Cada rede isolada é chamada de **subrede**.



Máscara de rede: /24

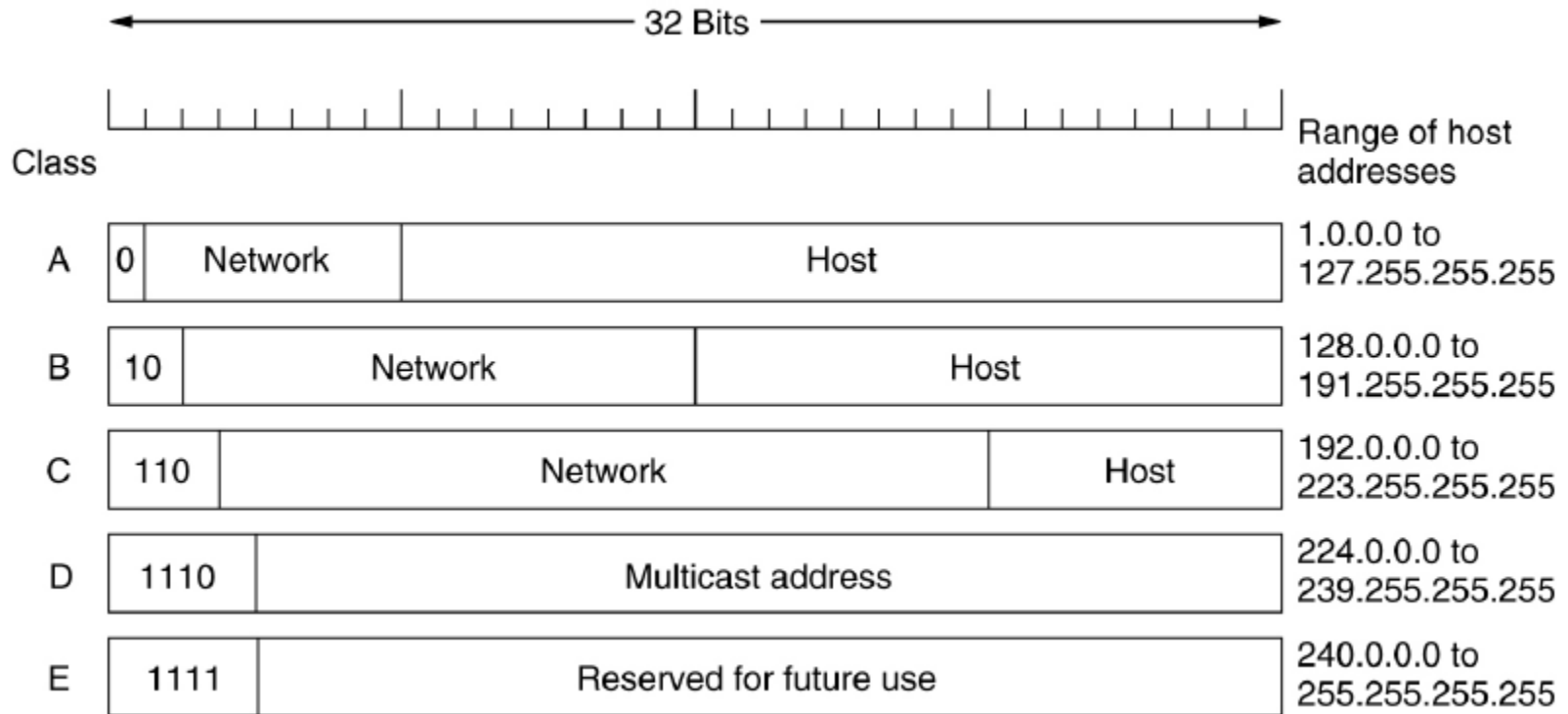
# Subredes

Quantas temos aqui?





# Endereçamento IP: Classes



Classe A: 8 bits para a subrede  
Classe B: 16 bits para a subrede  
Classe C: 24 bits para a subrede

# Endereçamento IP: Classes

- Desperdício de endereços IP
  - Classe C permite até  $2^8 - 2 = 254$  hosts/interfaces
  - Classe B permite até 65534 hosts/interfaces
  - Para organizações com até 2000 hosts é necessário alocar um endereço de rede classe B levando ao desperdício de mais de 63000 endereços que não serão usados !

# Endereçamento IP: CIDR

Padronizado em 1993 pelo IETF

## CIDR: Classless InterDomain Routing

- A porção no endereço que representa a subrede tem tamanho arbitrário
- Formato do endereço: **a.b.c.d/x**, onde **x** é o # de bits na porção do endereço que representa a subrede



200.23.16.0/23

# Endereçamento IP: CIDR

Padronizado em 1993 pelo IETF

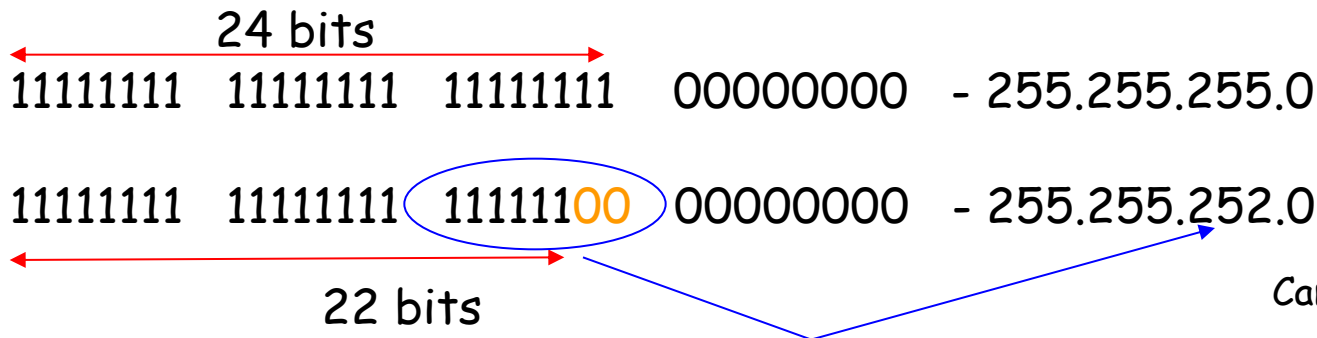
**CIDR: Classless InterDomain Routing**

- Exemplos

- 192.168.0.0/24 representa os 256 endereços IPv4 de 192.168.0.0 até 192.168.0.255 (este último é o endereço de broadcast da rede)
- 192.168.0.0/22 representa 1024 endereços IPv4 de 192.168.0.0 até 192.168.3.255 (este último é o endereço de broadcast da rede)

- Representação alternativa na forma decimal com pontos

- 192.168.0.0/24 pode ser escrito como 192.168.0.0/255.255.255.0
- 192.168.0.0/22 pode ser escrito como 192.168.0.0/255.255.252.0



# Endereços IP: como obter um?

Q: como hosts obtêm endereço IP?

- ❑ Colocado em um arquivo de configuração
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- ❑ **DHCP: Dynamic Host Configuration Protocol**: obtém dinamicamente um endereço através de um servidor
  - "plug-and-play"(mais em breve)

# Endereços IP: como obter um?

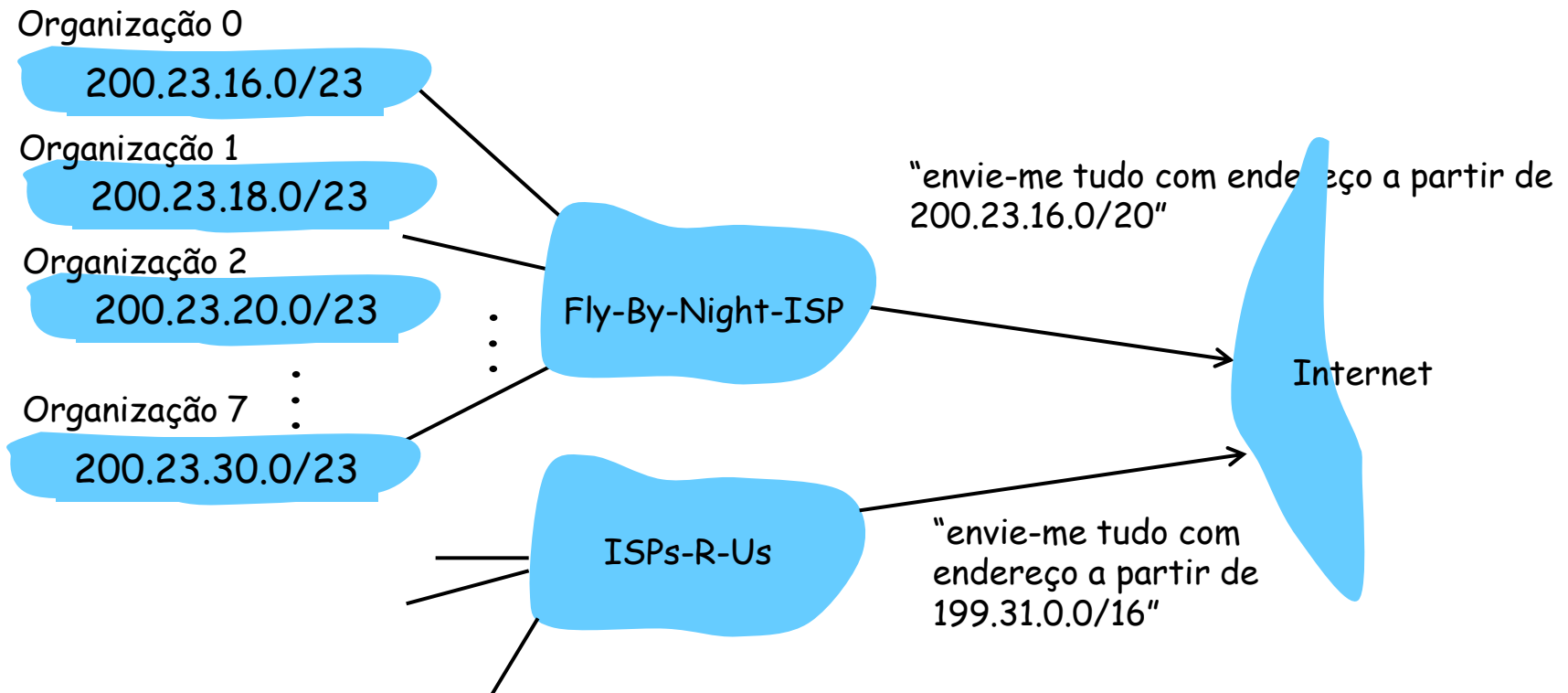
**Q:** como a rede obtém a parte de subrede do endereço IP?

**A:** pega porção alocada do espaço de endereço do seu ISP

Bloco do ISP	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organização 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organização 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organização 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...	.....	.....	.....	.....	.....
Organização 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

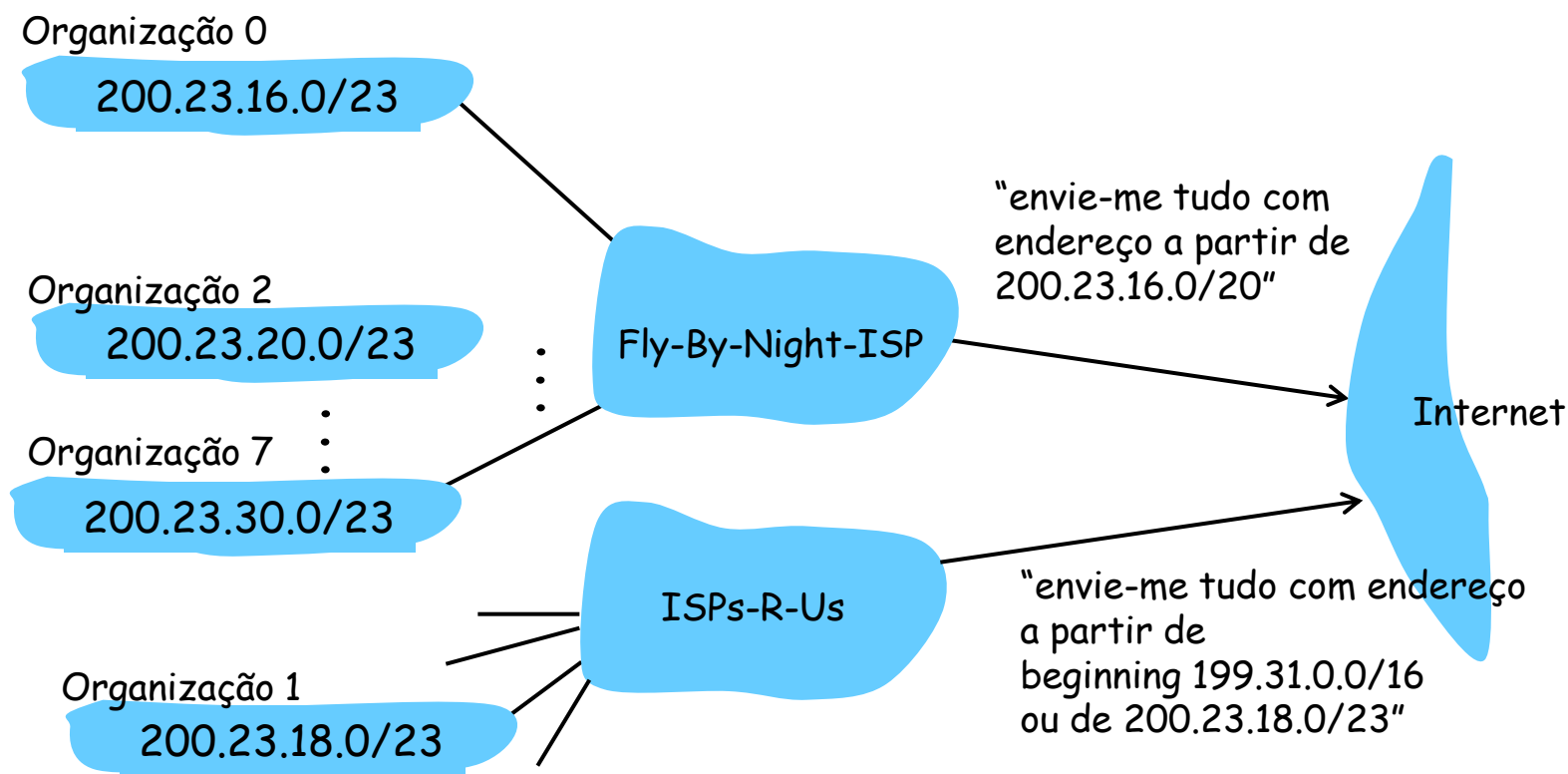
# Endereçamento Hierárquico: agregação de rotas

Endereçamento hierárquico permite uma eficiente divulgação de informações de roteamento:



# Endereçamento hierárquico: rotas mais específicas

ISPs-R-Us possui uma rota mais específica para a Organização 1





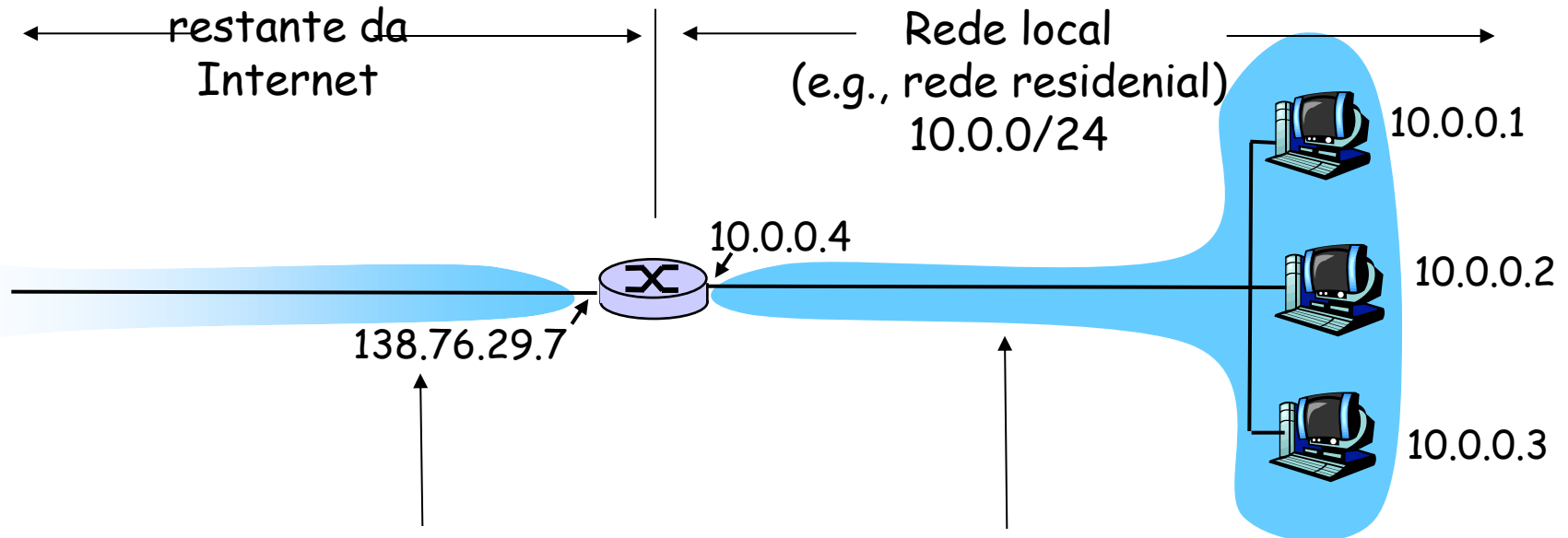
# Endereçamento IP: uma última palavra...

Q: Como um ISP consegue um bloco de endereços?

R: **ICANN**: Internet Corporation for Assigned Names and Numbers

- aloca endereços
- gerencia DNS
- atribui nomes de domínio, resolve disputas

# NAT: Network Address Translation



*Todos* os datagramas *deixando* a rede local possuem o *mesmo* endereço IP NAT de origem: 138.76.29.7, mas números de porta de origem diferentes

Datagramas com origem ou destino nesta rede possuem endereço 10.0.0/24 para fonte, destino (como usualmente)

# NAT: Network Address Translation

- **Motivação:** rede local usa somente um único endereço IP quando há necessidade de falar com o mundo externo:
  - range de endereços não são necessários do ISP: apenas um endereço IP para todos os dispositivos
  - Pode mudar endereço de dispositivos na rede local sem necessidade de notificar o mundo externo
  - Pode mudar o ISP sem mudar o endereço dos dispositivos na rede local
  - Dispositivos dentro da rede local não são explicitamente endereçáveis, ou seja, não são visíveis pelo mundo externo (um pouco mais de segurança).

# NAT: Network Address Translation

**Implementação:** roteador NAT deve:

- *Datagramas saindo para o mundo externo: substitui* (endereço IP fonte, # de porta) de todo datagrama sendo enviado para o mundo externo por (endereço IP NAT, novo # de porta)  
... Clientes/servidores remotos responderão usando (endereço IP NAT, novo # de porta) como endereço de destino.
- *Recorda (na tabela de tradução NAT)* os pares de equivalência (endereço IP fonte, # de porta) -- (endereço IP NAT, novo # de porta)
- *Datagramas chegando do mundo externo: substitui* (endereço IP NAT, novo # de porta) no campo "destino" de todos os pacotes provenientes do mundo externo com a informação correspondente (endereço IP fonte, # de porta) armazenada na tabela NAT

# NAT: Network Address Translation

Tabela de tradução NAT	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

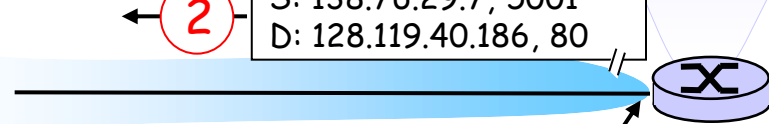
**1:** host 10.0.0.1  
Envia datagrama para  
128.119.40.186, 80

S: 10.0.0.1, 3345  
D: 128.119.40.186, 80

10.0.0.1  
10.0.0.2  
10.0.0.3

**2:** roteador NAT  
Muda ende. de ori-  
gem do datagrama  
de  
10.0.0.1, 3345 para  
138.76.29.7, 5001,  
e atualiza tabela

S: 138.76.29.7, 5001  
D: 128.119.40.186, 80



138.76.29.7

S: 128.119.40.186, 80  
D: 138.76.29.7, 5001

**3:** Resposta chega  
com endereço:  
138.76.29.7, 5001

10.0.0.4  
S: 128.119.40.186, 80  
D: 10.0.0.1, 3345

**4:** roteador NAT  
Muda endereço de destino  
do datagrama de  
138.76.29.7, 5001 para 10.0.0.1, 3345

# NAT: Network Address Translation

- ❑ Campo “# de porta” de 16 bits:
  - +60.000 conexões simultâneas com um único endereço IP externo!
- ❑ NAT é controverso:
  - roteadores devem processar somente até a camada 3
  - viola o argumento “fim-a-fim”
    - Possibilidade do uso de NAT deve ser levada em conta por projetistas de aplicações, eg, aplicações P2P
  - Esgotamento de endereços deve ser resolvido pelo IPv6

# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: **I**nternet **P**rotocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - **ICMP**
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# ICMP: Internet Control Message Protocol

- ❑ Usado por hosts & roteadores para comunicar informações do nível rede

- Reportagem de erro: host, rede, porta, protocolo inalcançável
- echo request/reply (usado pelo ping)

- ❑ Camada de rede "acima do" IP:

- ICMP mensagens transportadas em datagramas IP

- ❑ **Mensagem ICMP:** tipo, código mais 8 primeiros bytes do datagrama IP que causou o erro

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header



# Traceroute e ICMP

- ❑ Fonte envia uma série de segmentos UDP para o destino
    - O 1º possui TTL =1
    - O 2º possui TTL=2, etc.
    - Número de porta improvável
  - ❑ Quando enésimo datagrama chega ao enésimo roteador:
    - Roteador descarta datagrama
    - E envia para a fonte um mensagem ICMP (tipo 11, código 0)
    - Mensagem inclui nome do roteador & endereço
  - ❑ Quando a mensagem ICMP chega, fonte calcula o RTT
  - ❑ Traceroute faz isto 3 vezes
- Critério de Parada
- ❑ segmento UDP eventualmente chegará ao host de destino
  - ❑ O destino retorna a mensagem ICMP "host unreachable" (tipo 3, código 3)
  - ❑ Quando a fonte recebe esta msg ICMP, pára.

# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# IPv6

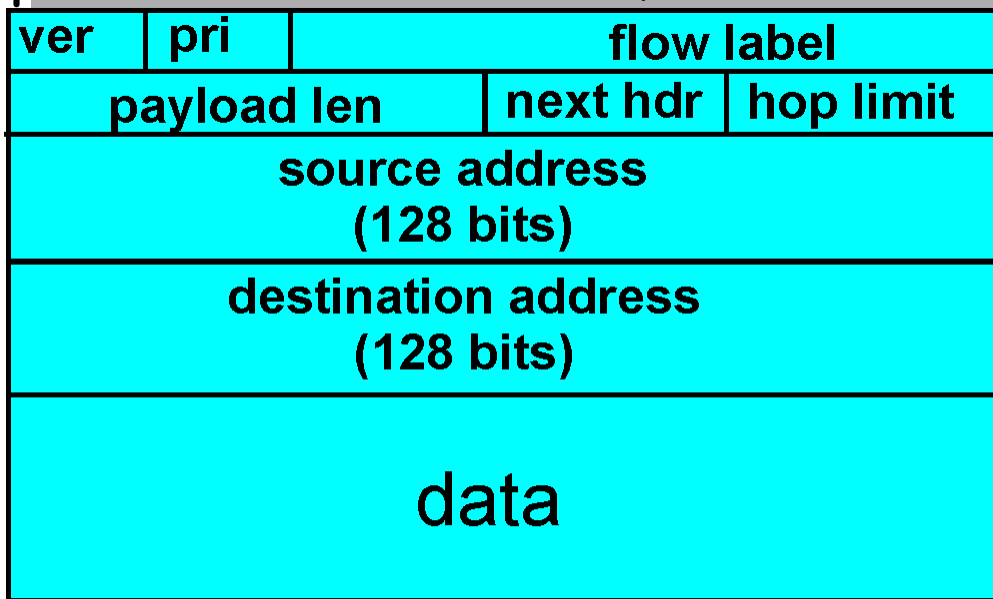
- **Motivação inicial:** *espaço de endereçamento de 32 bits estará completamente alocado em pouco tempo.*
  - **Motivação adicional:**
    - Formato do cabeçalho ajuda a fazer processamento/encaminhamento mais rápido
    - Mudanças no cabeçalho para facilitar QoS
- Formato do datagrama IPv6:**
- Cabeçalho de tamanho fixo de 40 bytes
  - Fragmentação não é permitida

# Cabeçalho IPv6 (Cont)

**Priority:** identifica prioridade dos datagramas

**Flow Label:** identifica datagramas no mesmo "fluxo."  
(conceito de "fluxo" não é bem definido).

**Next header:** identifica protocolo da camada superior para "data" -> dados (possui outras funções também)



*128 bits* equivalem à

$3,4 \times 10^{38}$  endereços  
únicos

← 32 bits →

# Outras mudanças com relação ao IPv4

- ❑ *Checksum*: inteiramente removido para reduzir tempo de processamento em cada salto
- ❑ *Options*: permitido, mas fora do cabeçalho, indicado pelo campo "Next Header"
- ❑ *ICMPv6*: nova versão do ICMP
  - Tipos adicionais de mensagens, e.g. "Pacote muito grande"
  - Funções de gerenciamento de grupos multicast

# Endereçamento IPv6 - Notação

- Cada um formado por 8 grupos com 4 dígitos hexadecimais
  - 2001:0db8:85a3:08d3:1319:8a2e:0370:7334
  
- 4 zeros em um grupo podem ser omitidos e substituídos por ::
  - 2001:0db8:85a3:0000:1319:8a2e:0370:7334 equivale à 2001:0db8:85a3:::1319:8a2e:0370:7334

# Endereçamento IPv6 - Notação

## □ Exemplo de endereços válidos e equivalentes

2001:0db8:0000:0000:0000:0000:1428:57ab (end. original)

2001:0db8:0000:0000:0000::1428:57ab (grupo de 4 zeros subst.)

2001:0db8:0:0:0:0:1428:57ab ("0000" por "0")

2001:0db8:0:0::1428:57ab (:: são equivalentes a 0000:0000)

2001:0db8::1428:57ab (:: são equivalentes a 0000:0000:0000:0000)

2001:db8::1428:57ab (zero à esquerda pode ser retirado)

As simplificações não podem gerar ambigüidades !

Exemplo: 2001:c00:0:0:5400:0:0:9 ----- > 2001:c00::5400::9 ????

# Endereçamento IPv6 - Notação

## □ Em uma URL devemos fazer como segue

- `http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]/`

Qual a razão do uso de colchetes na URL ?

`http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443/`

## □ Rede

- Segue notação CIDR

`2001:0db8:1234::/48` significa que rede possui faixa de endereços de

`2001:0db8:1234:0000:0000:0000:0000:0000` até

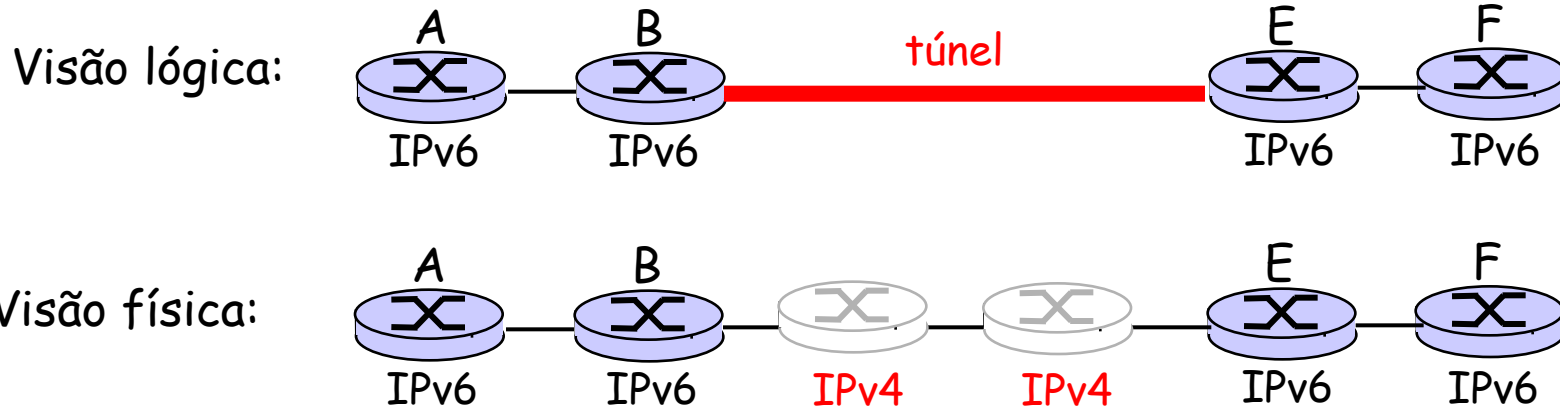
`2001:0db8:1234:FFFF:FFFF:FFFF:FFFF:FFFF`



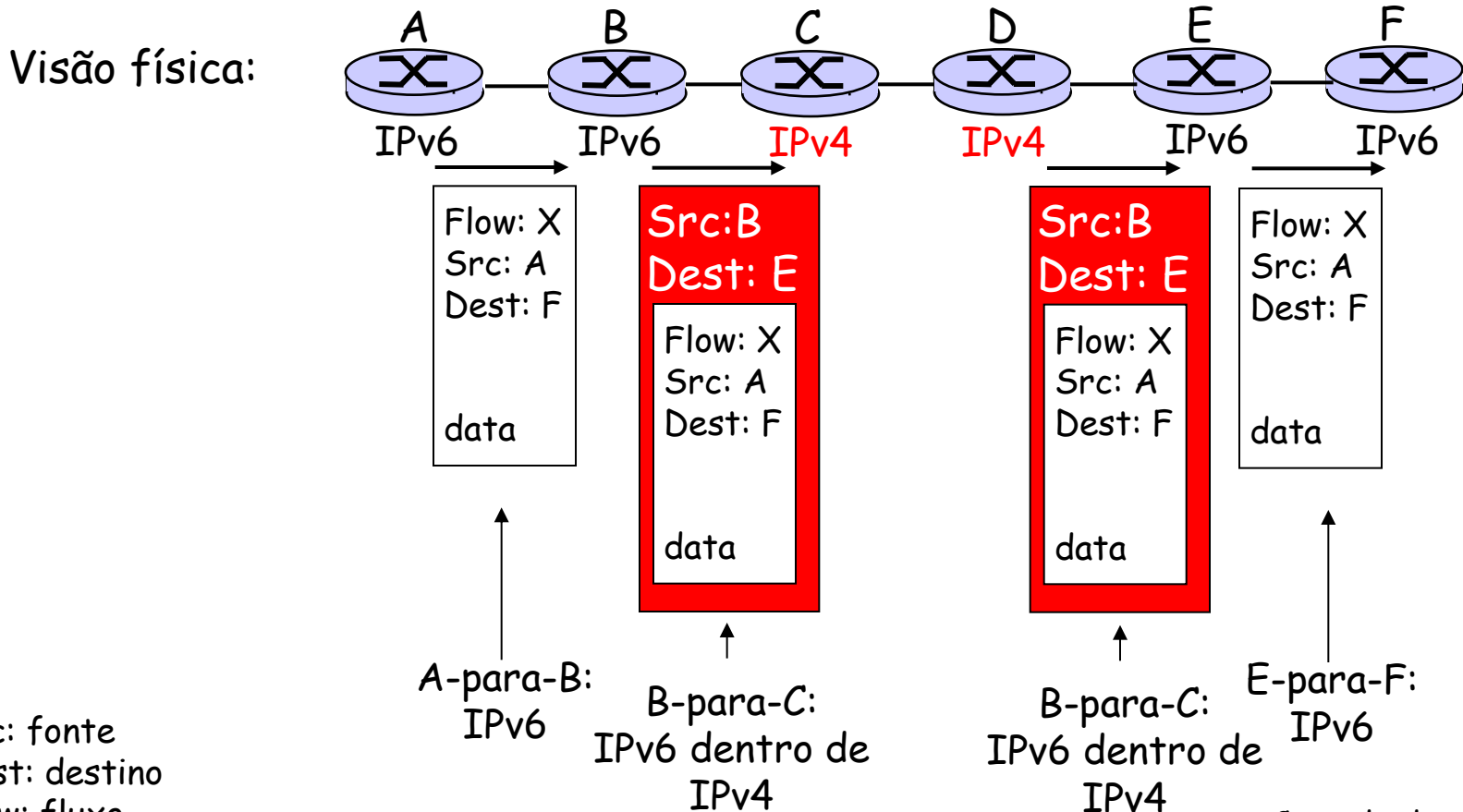
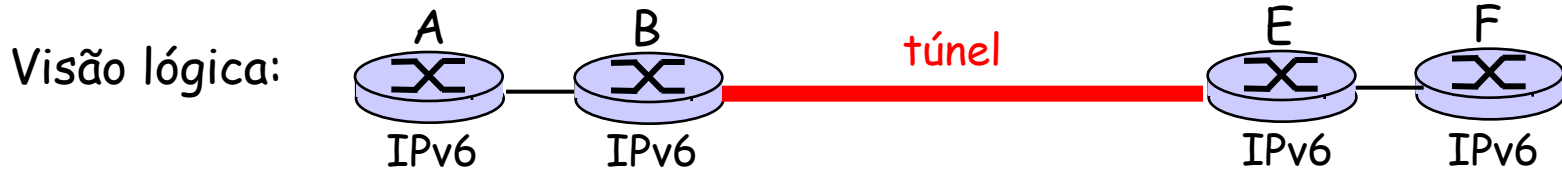
# Transição do IPv4 para o IPv6

- ❑ Todos os roteadores não podem se atualizados simultaneamente
  - dia do multirão de atualização é impossível
  - Como a rede irá operar com roteadores IPv4 e IPv6 misturados dentro dela?
- ❑ **Tunelamento:** IPv6 transportado como "payload" em datagramas IPv4 entre roteadores IPv4

# Tunelamento



# Tunelamento

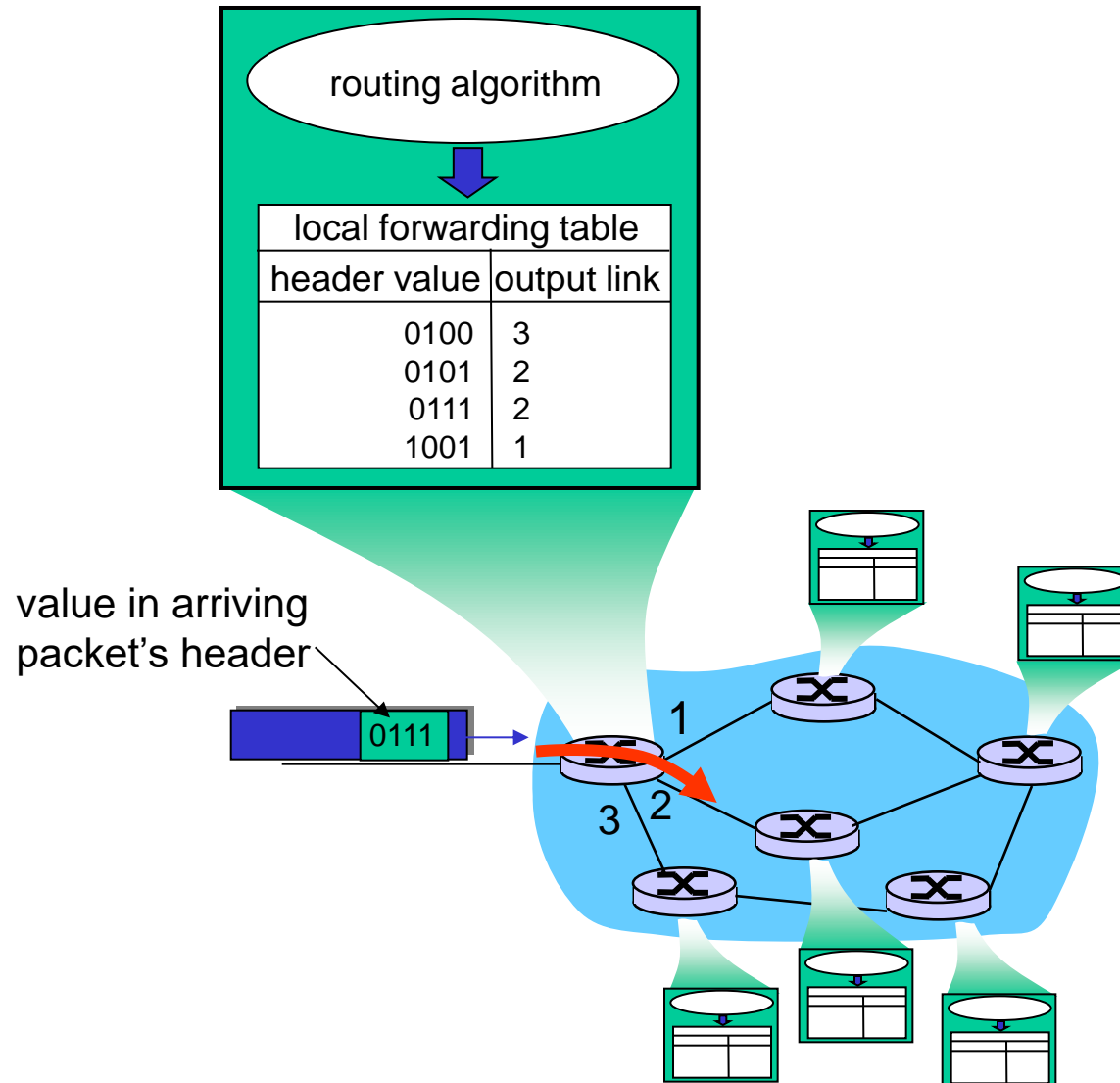


Src: fonte  
Dest: destino  
Flow: fluxo  
Data: dados

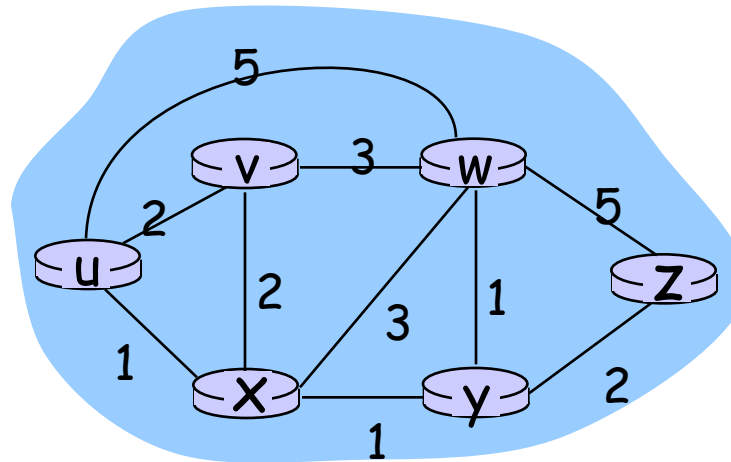
# Módulo 4: Camada de Rede

- ❑ 4.1 Introdução
- ❑ 4.2 redes de circuitos virtuais e de datagramas
- ❑ 4.3 O que há dentro de um roteador
- ❑ 4.4 IP: Internet Protocol - Protocolo Internet
  - Formato do datagrama
  - Endereçamento IPv4
  - ICMP
  - IPv6
- ❑ 4.5 Algoritmos de roteamento
  - Link state (Estado do enlace)
  - Distance Vector (Vetor-Distância)
  - Roteamento hierárquico
- ❑ 4.6 Roteamento na Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast e roteamento multicast

# Interplay between routing, forwarding



# Graph abstraction



Graph:  $G = (N,E)$

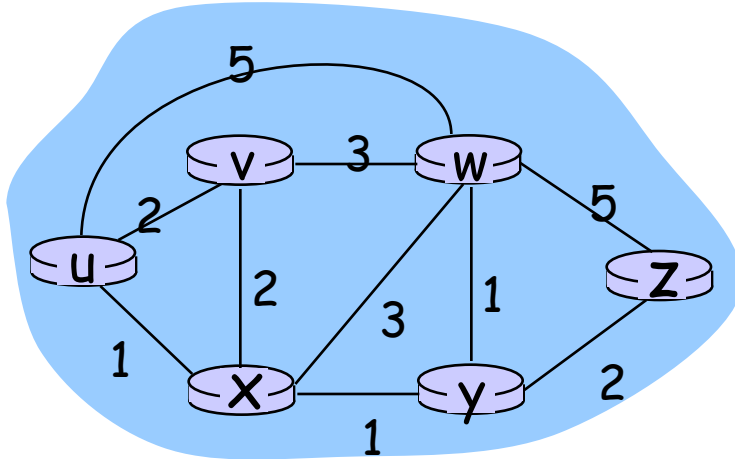
$N =$  set of routers =  $\{ u, v, w, x, y, z \}$

$E =$  set of links =  $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where  $N$  is set of peers and  $E$  is set of TCP connections

# Graph abstraction: costs



- $c(x,x')$  = cost of link  $(x,x')$ 
  - e.g.,  $c(w,z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

# Routing Algorithm classification

## Global or decentralized information?

### Global:

- ❑ all routers have complete topology, link cost info
- ❑ "link state" algorithms

### Decentralized:

- ❑ router knows physically-connected neighbors, link costs to neighbors
- ❑ iterative process of computation, exchange of info with neighbors
- ❑ "distance vector" algorithms

## Static or dynamic?

### Static:

- ❑ routes change slowly over time

### Dynamic:

- ❑ routes change more quickly
  - periodic update
  - in response to link cost changes



# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

# A Link-State Routing Algorithm

## Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives forwarding table for that node
- iterative: after  $k$  iterations, know least cost path to  $k$  dest.'s

## Notation:

- $c(x,y)$ : link cost from node  $x$  to  $y$ ;  $= \infty$  if not direct neighbors
- $D(v)$ : current value of cost of path from source to dest.  $v$
- $p(v)$ : predecessor node along path from source to  $v$
- $N'$ : set of nodes whose least cost path definitively known

# Dijsktra's Algorithm

1 **Initialization:**

2  $N' = \{u\}$

3 for all nodes  $v$

4 if  $v$  adjacent to  $u$

5 then  $D(v) = c(u,v)$

6 else  $D(v) = \infty$

7

8 **Loop**

9 find  $w$  not in  $N'$  such that  $D(w)$  is a minimum

10 add  $w$  to  $N'$

11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :

12  $D(v) = \min( D(v), D(w) + c(w,v) )$

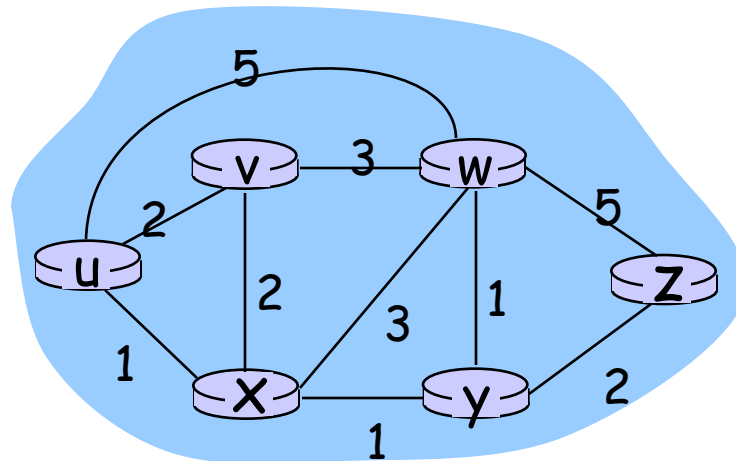
13 /\* new cost to  $v$  is either old cost to  $v$  or known

14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

15 **until all nodes in  $N'$**

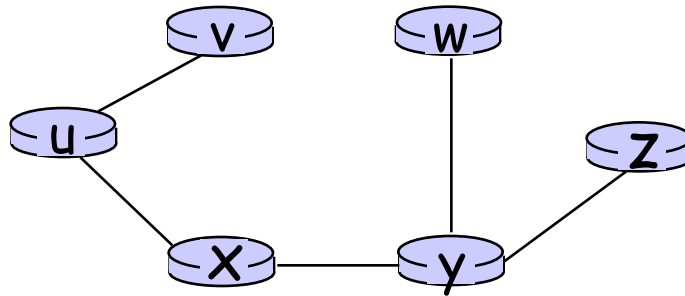
# Dijkstra's algorithm: example

Step	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	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



# Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

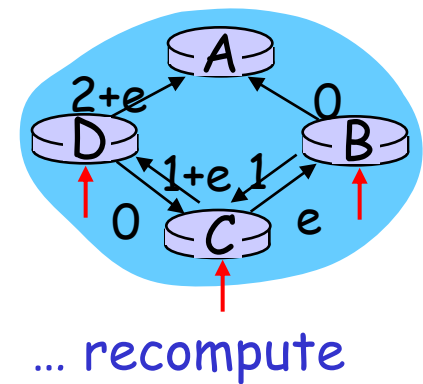
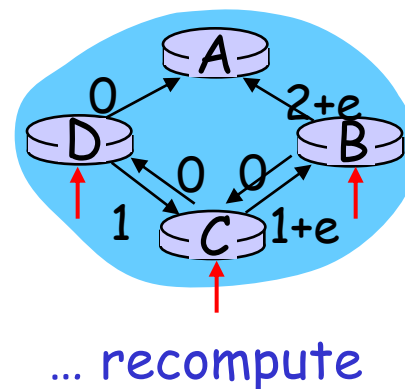
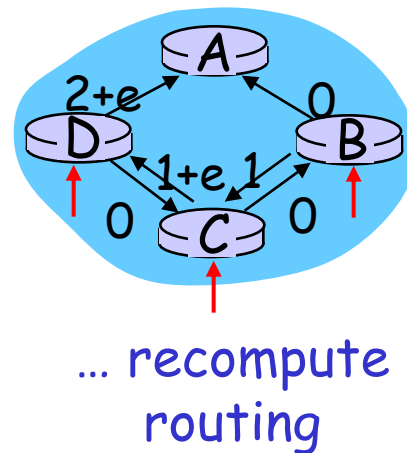
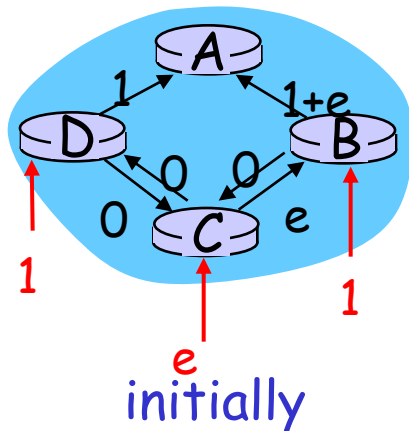
# Dijkstra's algorithm, discussion

**Algorithm complexity:** n nodes

- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$  comparisons:  $O(n^2)$
- more efficient implementations possible:  $O(n \log n)$

**Oscillations possible:**

- e.g., link cost = amount of carried traffic



# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

# Distance Vector Algorithm

## Bellman-Ford Equation (dynamic programming)

Define

$d_x(y) :=$  cost of least-cost path from  $x$  to  $y$

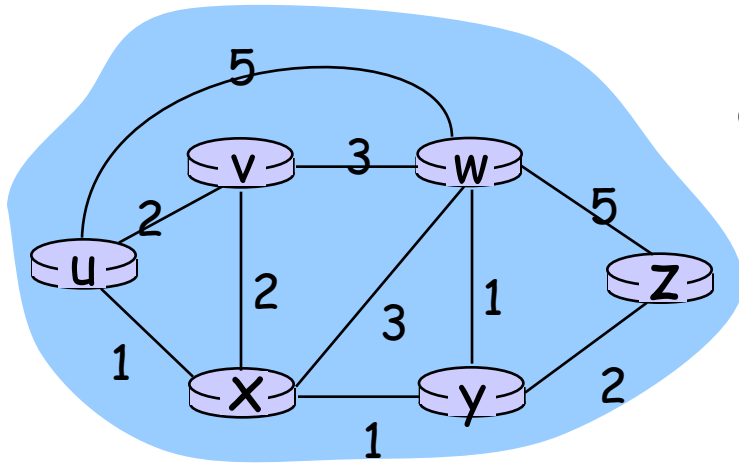
Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

where min is taken over all neighbors  $v$  of  $x$



# Bellman-Ford example



Clearly,  $d_v(z) = 5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

Node that achieves minimum is next hop in shortest path → forwarding table

# Distance Vector Algorithm

- $D_x(y)$  = estimate of least cost from  $x$  to  $y$
- Distance vector:  $D_x = [D_x(y): y \in N]$
- Node  $x$  knows cost to each neighbor  $v$ :  
 $c(x,v)$
- Node  $x$  maintains  $D_x = [D_x(y): y \in N]$
- Node  $x$  also maintains its neighbors' distance vectors
  - For each neighbor  $v$ ,  $x$  maintains  
 $D_v = [D_v(y): y \in N]$

# Distance vector algorithm (4)

## Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node  $x$  receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate  $D_x(y)$  converge to the actual least cost  $d_x(y)$

# Distance Vector Algorithm (5)

## Iterative, asynchronous:

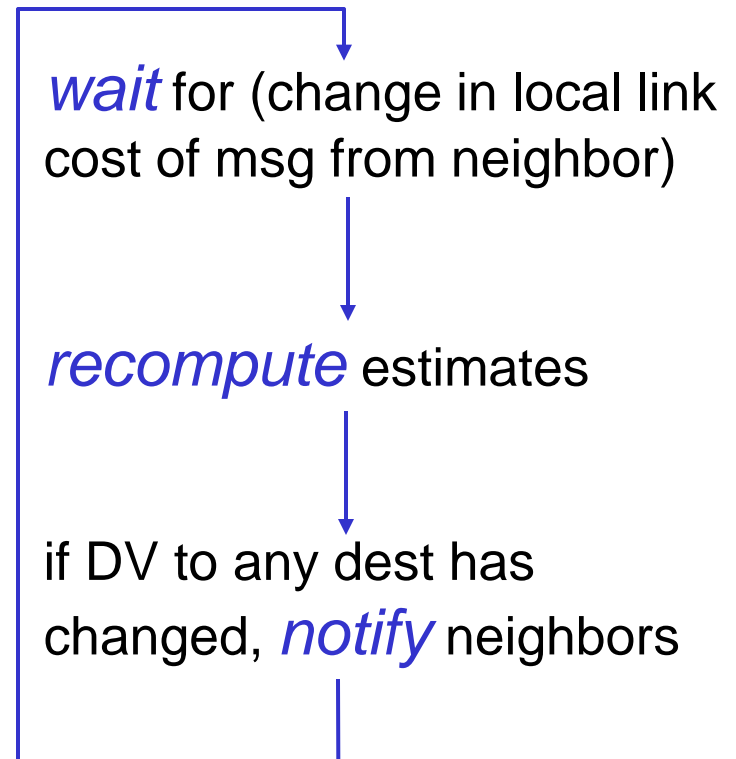
each local iteration caused by:

- ❑ local link cost change
- ❑ DV update message from neighbor

## Distributed:

- ❑ each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

## Each node:



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x table**

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

**node y table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

**node z table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

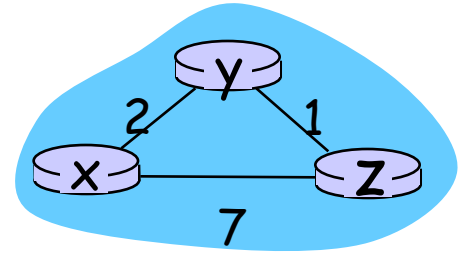
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

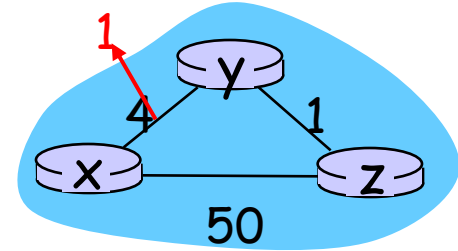


time →

# Distance Vector: link cost changes

## Link cost changes:

- ❑ node detects local link cost change
- ❑ updates routing info, recalculates distance vector
- ❑ if DV changes, notify neighbors



At time  $t_0$ ,  $y$  detects the link-cost change, updates its DV, and informs its neighbors.

At time  $t_1$ ,  $z$  receives the update from  $y$  and updates its table. It computes a new least cost to  $x$  and sends its neighbors its DV.

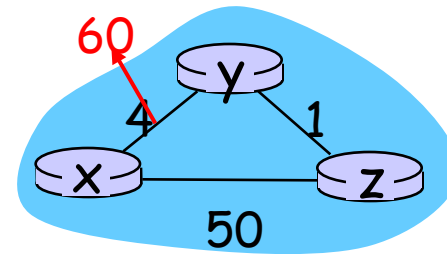
At time  $t_2$ ,  $y$  receives  $z$ 's update and updates its distance table.  $y$ 's least costs do not change and hence  $y$  does not send any message to  $z$ .

“good  
news  
travels  
fast”

# Distance Vector: link cost changes

## Link cost changes:

- ❑ good news travels fast
- ❑ bad news travels slow - "count to infinity" problem!
- ❑ 44 iterations before algorithm stabilizes: see text



## Poisoned reverse:

- ❑ If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❑ will this completely solve count to infinity problem?

# Comparison of LS and DV algorithms

## Message complexity

- LS: with  $n$  nodes,  $E$  links,  $O(nE)$  msgs sent
- DV: exchange between neighbors only
  - convergence time varies

## Speed of Convergence

- LS:  $O(n^2)$  algorithm requires  $O(nE)$  msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

**Robustness:** what happens if router malfunctions?

## LS:

- node can advertise incorrect *link* cost
- each node computes only its own table

## DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network



# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 **Routing algorithms**
  - Link state
  - Distance Vector
  - **Hierarchical routing**
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

# Hierarchical Routing

Our routing study thus far - idealization

- ❑ all routers identical
- ❑ network "flat"

... *not* true in practice

**scale:** with 200 million destinations:

- ❑ can't store all dest's in routing tables!
- ❑ routing table exchange would swamp links!

**administrative autonomy**

- ❑ internet = network of networks
- ❑ each network admin may want to control routing in its own network

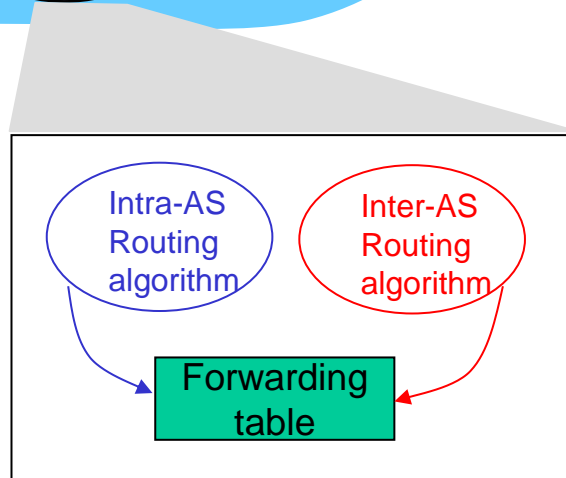
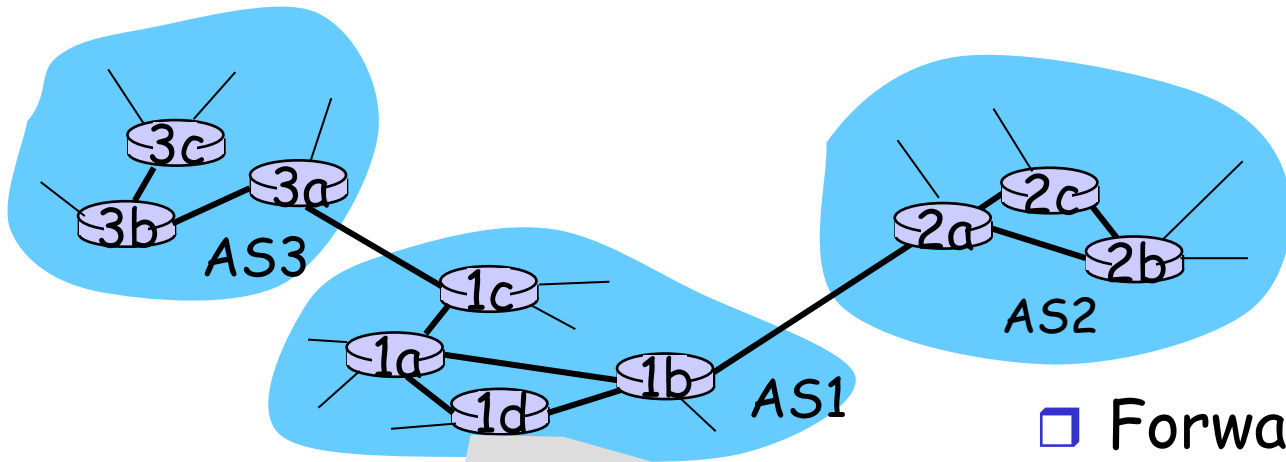
# Hierarchical Routing

- ❑ aggregate routers into regions, "autonomous systems" (AS)
- ❑ routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

## Gateway router

- ❑ Direct link to router in another AS

# Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

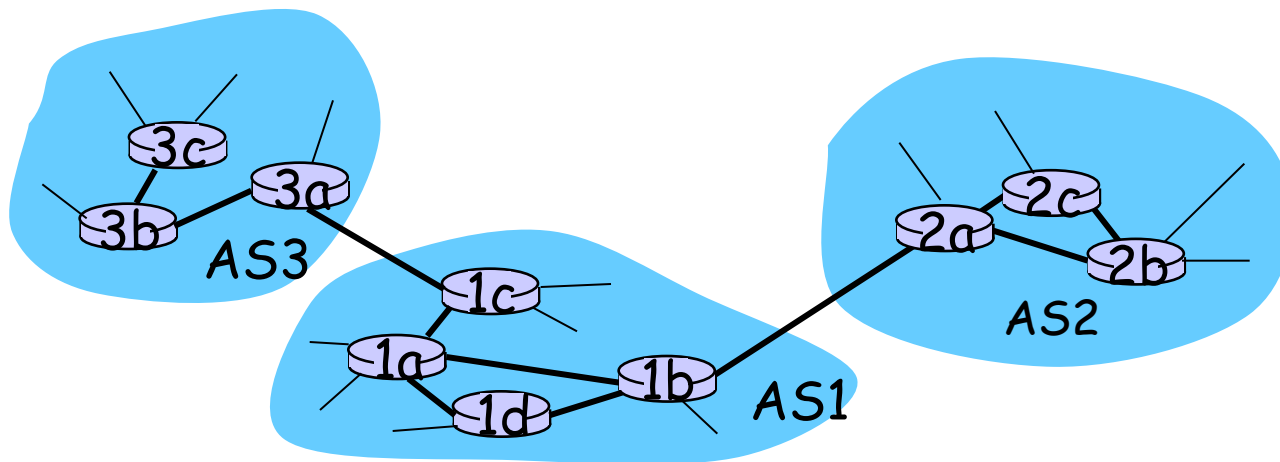
# Inter-AS tasks

- Suppose router in AS1 receives datagram for which dest is outside of AS1
  - Router should forward packet towards one of the gateway routers, but which one?

## AS1 needs:

1. to learn which dests are reachable through AS2 and which through AS3
2. to propagate this reachability info to all routers in AS1

**Job of inter-AS routing!**

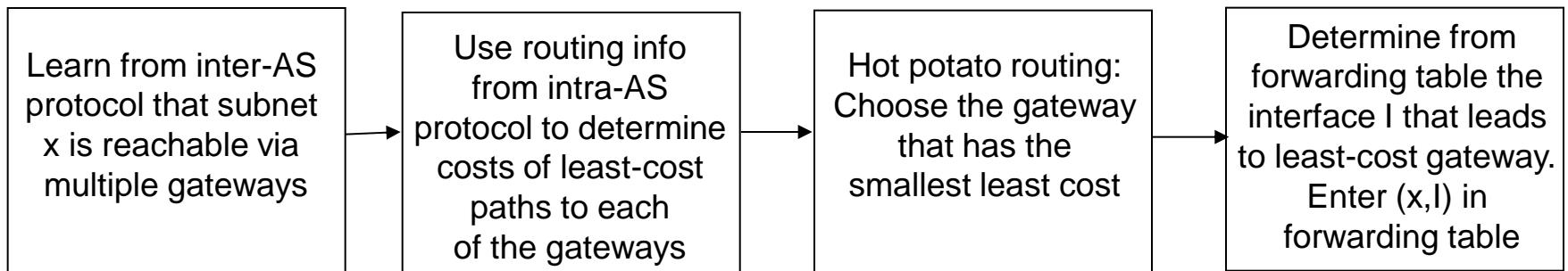


## Example: Setting forwarding table in router 1d

- ❑ Suppose AS1 learns from the inter-AS protocol that subnet  $x$  is reachable from AS3 (gateway 1c) but not from AS2.
- ❑ Inter-AS protocol propagates reachability info to all internal routers.
- ❑ Router 1d determines from intra-AS routing info that its interface  $I$  is on the least cost path to 1c.
- ❑ Puts in forwarding table entry  $(x, I)$ .

# Example: Choosing among multiple ASes

- ❑ Now suppose AS1 learns from the inter-AS protocol that subnet  $x$  is reachable from AS3 and from AS2.
- ❑ To configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest  $x$ .
- ❑ This is also the job on inter-AS routing protocol!
- ❑ **Hot potato routing:** send packet towards closest of two routers.



# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing



# Intra-AS Routing

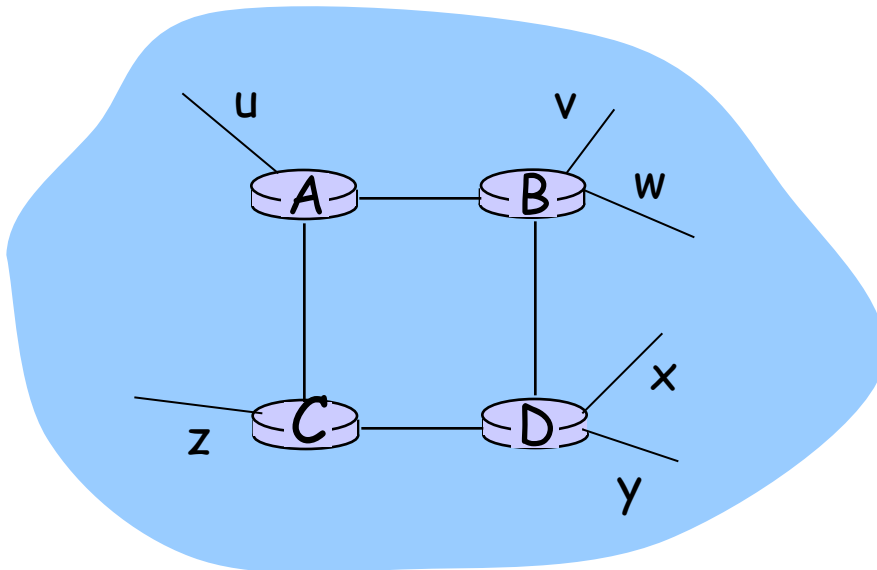
- ❑ Also known as **Interior Gateway Protocols (IGP)**
- ❑ Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

# RIP (Routing Information Protocol)

- ❑ Distance vector algorithm
- ❑ Included in BSD-UNIX Distribution in 1982
- ❑ Distance metric: # of hops (max = 15 hops)



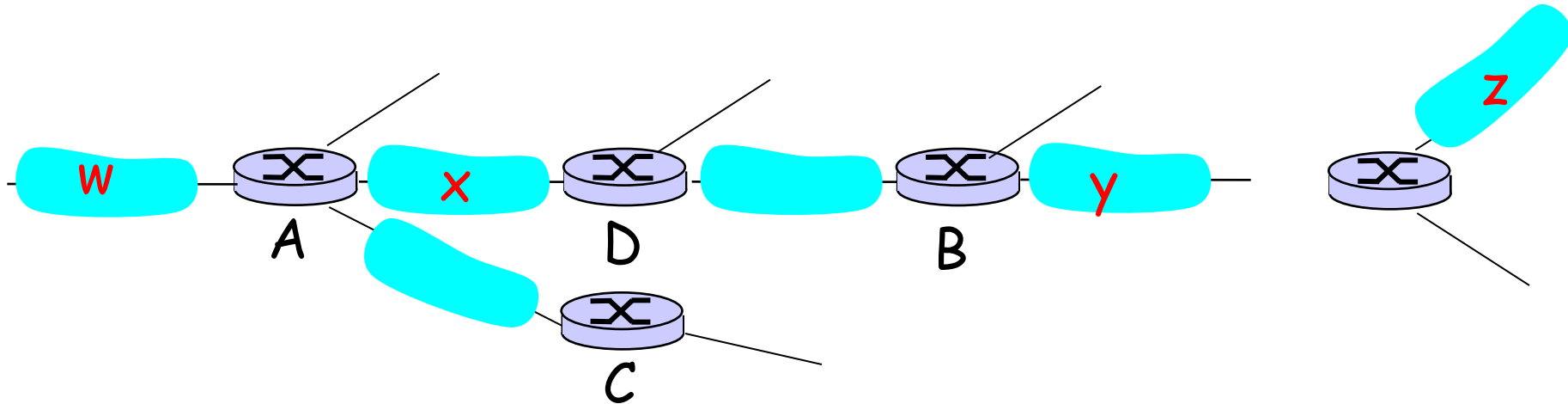
From router A to subsets:

<u>destination</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

# RIP advertisements

- ❑ Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called **advertisement**)
- ❑ Each advertisement: list of up to 25 destination nets within AS

# RIP: Example



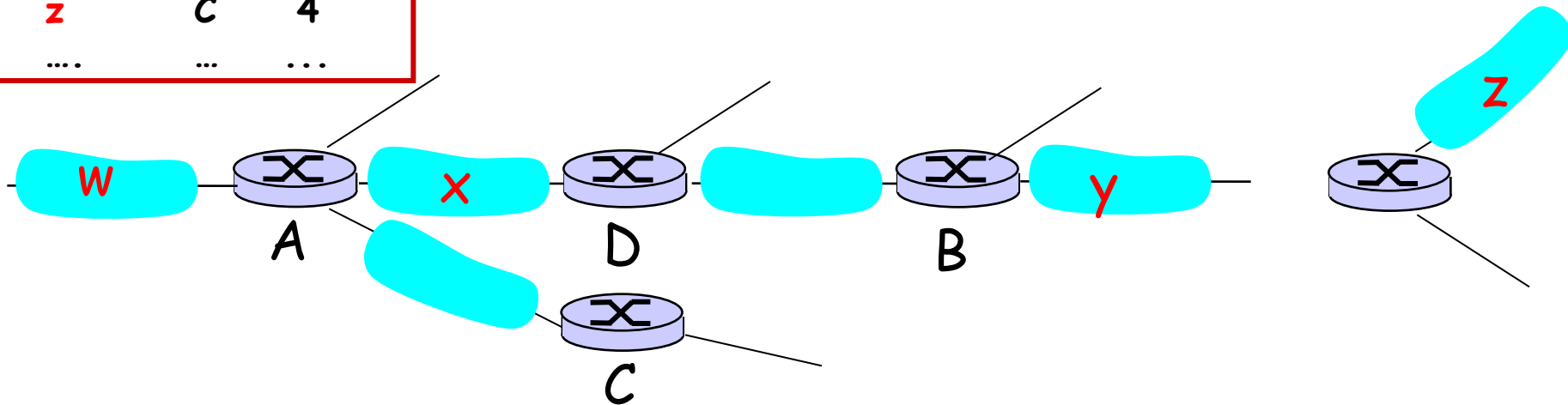
Destination Network	Next Router	Num. of hops to dest.
W	A	2
Y	B	2
Z	B	7
X	--	1
....	....	....

Routing table in D

# RIP: Example

Dest	Next	hops
w	-	1
x	-	1
z	C	4
....	...	....

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	<del>B</del> A	<del>7</del> 5
x	--	1
....	....	....

Routing table in D

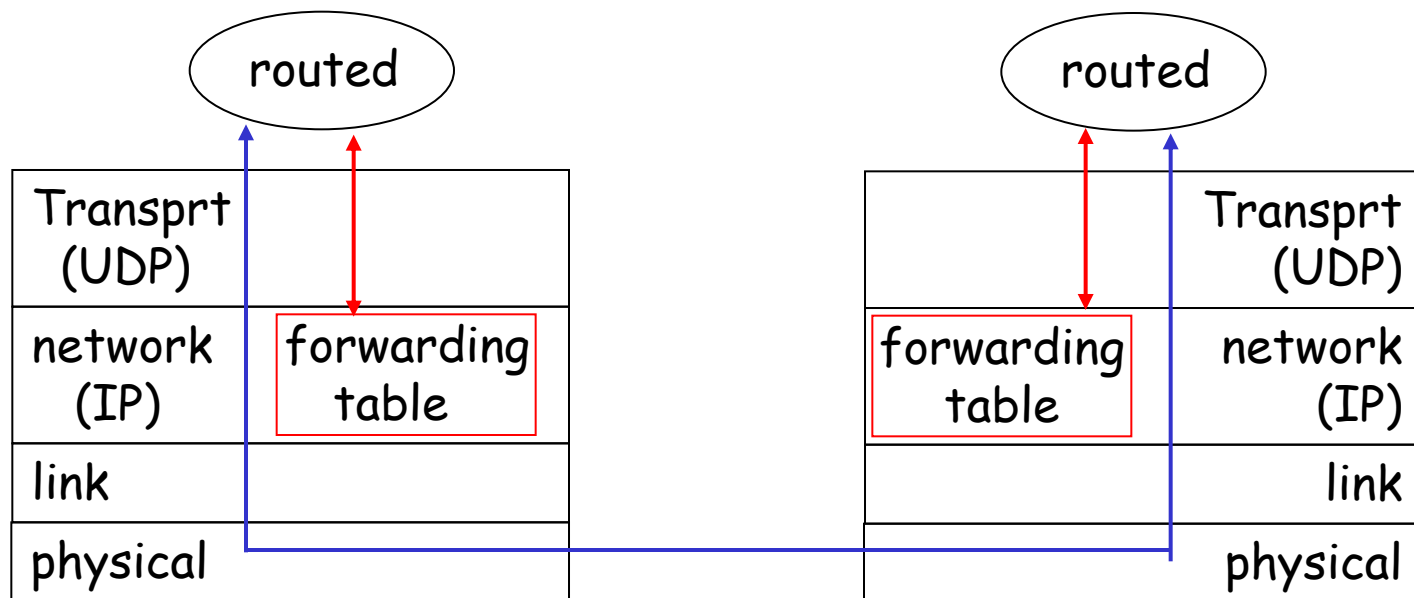
# RIP: Link Failure and Recovery

If no advertisement heard after 180 sec -->  
neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

# RIP Table processing

- ❑ RIP routing tables managed by **application-level** process called route-d (daemon)
- ❑ advertisements sent in UDP packets, periodically repeated





# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

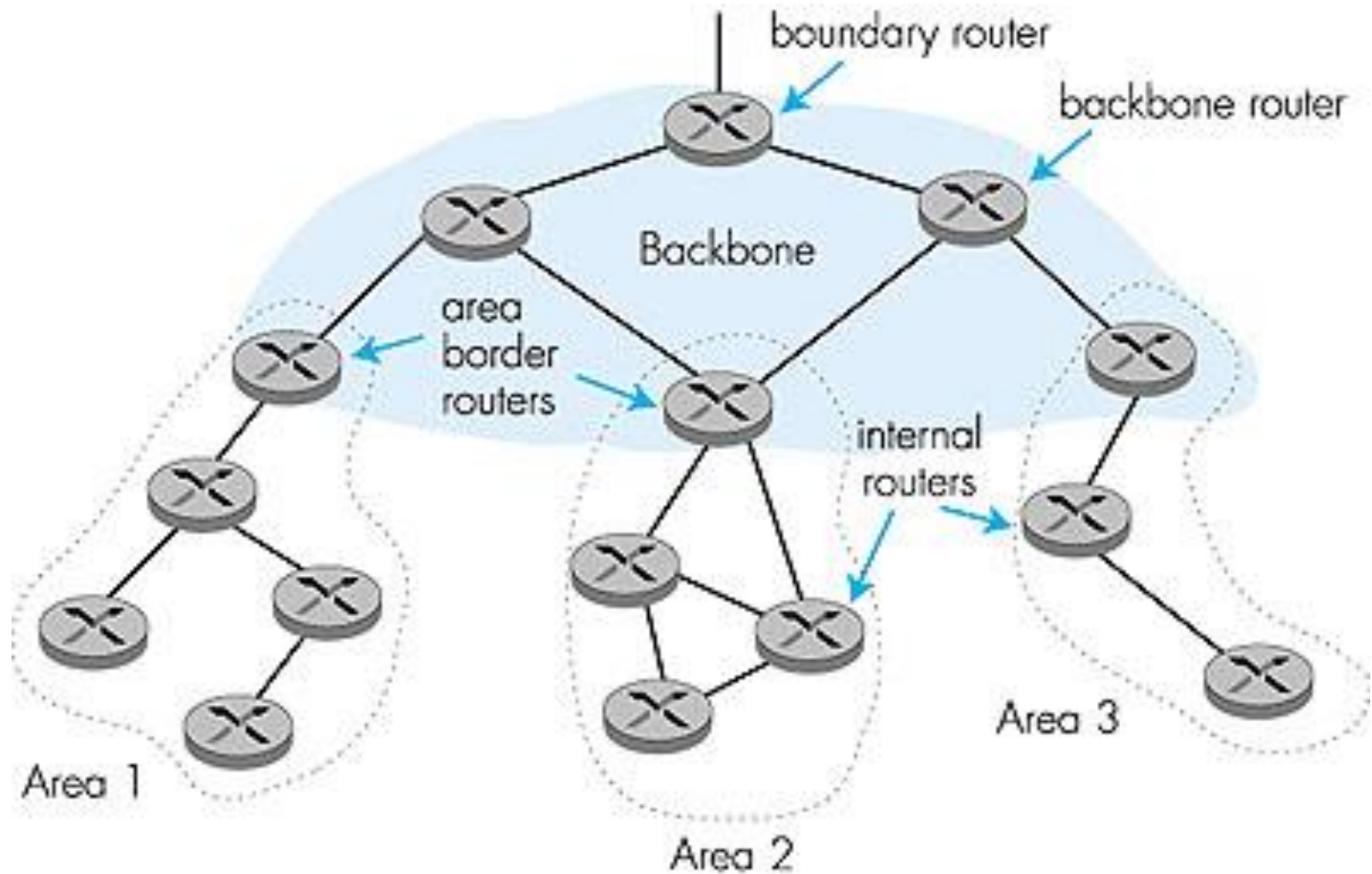
# OSPF (Open Shortest Path First)

- ❑ “open”: publicly available
- ❑ Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- ❑ OSPF advertisement carries one entry per neighbor router
- ❑ Advertisements disseminated to **entire** AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP)

# OSPF "advanced" features (not in RIP)

- ❑ **Security:** all OSPF messages authenticated (to prevent malicious intrusion)
- ❑ **Multiple same-cost paths** allowed (only one path in RIP)
- ❑ For each link, multiple cost metrics for different **TOS** (e.g., satellite link cost set "low" for best effort; high for real time)
- ❑ Integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- ❑ **Hierarchical** OSPF in large domains.

# Hierarchical OSPF



# Hierarchical OSPF

- ❑ **Two-level hierarchy:** local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- ❑ **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.
- ❑ **Backbone routers:** run OSPF routing limited to backbone.
- ❑ **Boundary routers:** connect to other AS's.

# Chapter 4: Network Layer

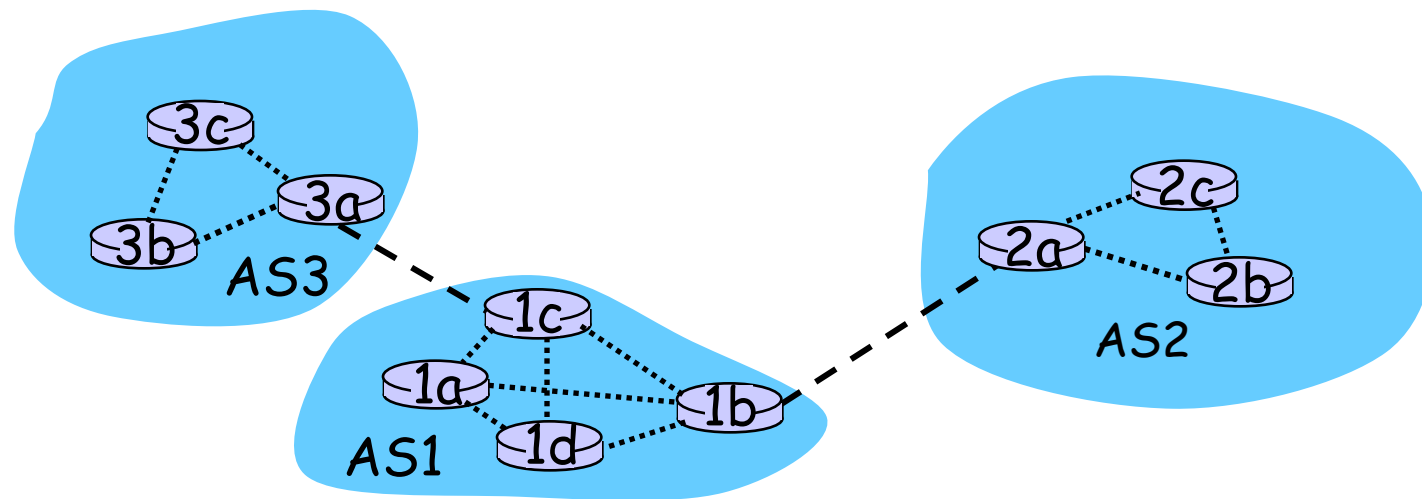
- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing

# Internet inter-AS routing: BGP

- ❑ **BGP (Border Gateway Protocol):** the de facto standard
- ❑ BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate the reachability information to all routers internal to the AS.
  3. Determine "good" routes to subnets based on reachability information and policy.
- ❑ Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*

# BGP basics

- ❑ Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP conctns: **BGP sessions**
- ❑ Note that BGP sessions do not correspond to physical links.
- ❑ When AS2 advertises a prefix to AS1, AS2 is **promising** it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



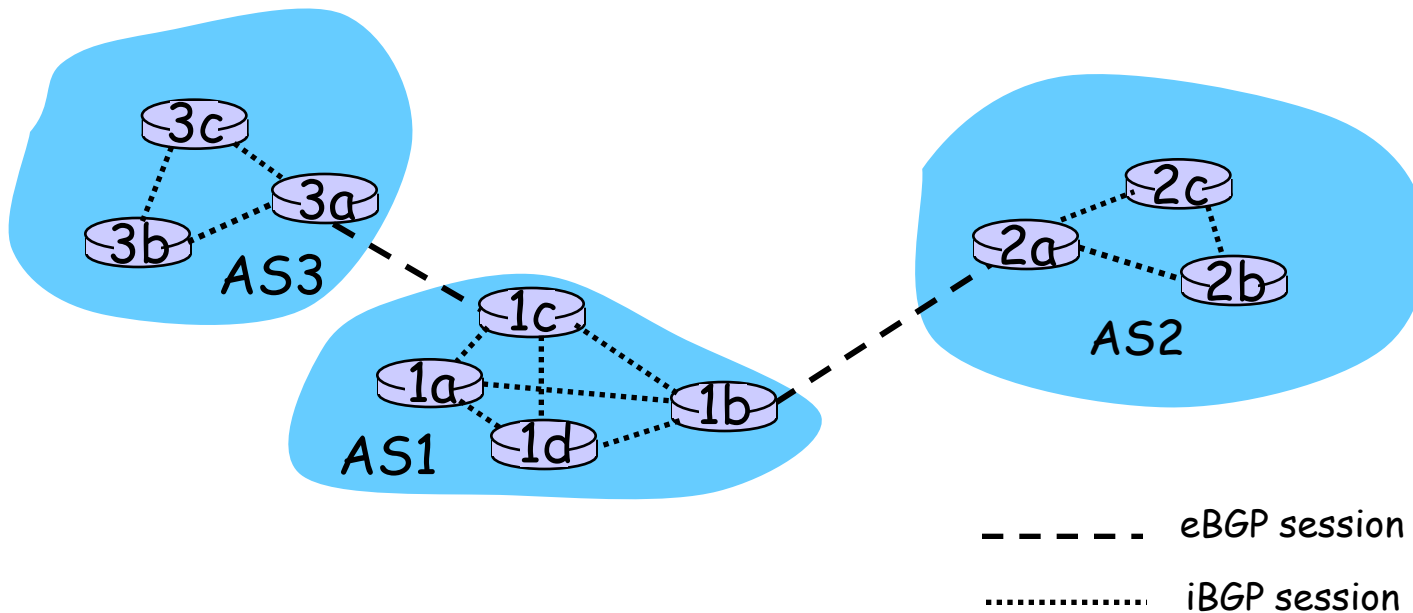
----- eBGP session

..... iBGP session



# Distributing reachability info

- ❑ With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- ❑ 1c can then use iBGP to distribute this new prefix reach info to all routers in AS1
- ❑ 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a eBGP session
- ❑ When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



# Path attributes & BGP routes

- ❑ When advertising a prefix, advert includes BGP attributes.
  - prefix + attributes = "route"
- ❑ Two important attributes:
  - **AS-PATH**: contains the ASs through which the advert for the prefix passed: AS 67 AS 17
  - **NEXT-HOP**: Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- ❑ When gateway router receives route advert, uses **import policy** to accept/decline.

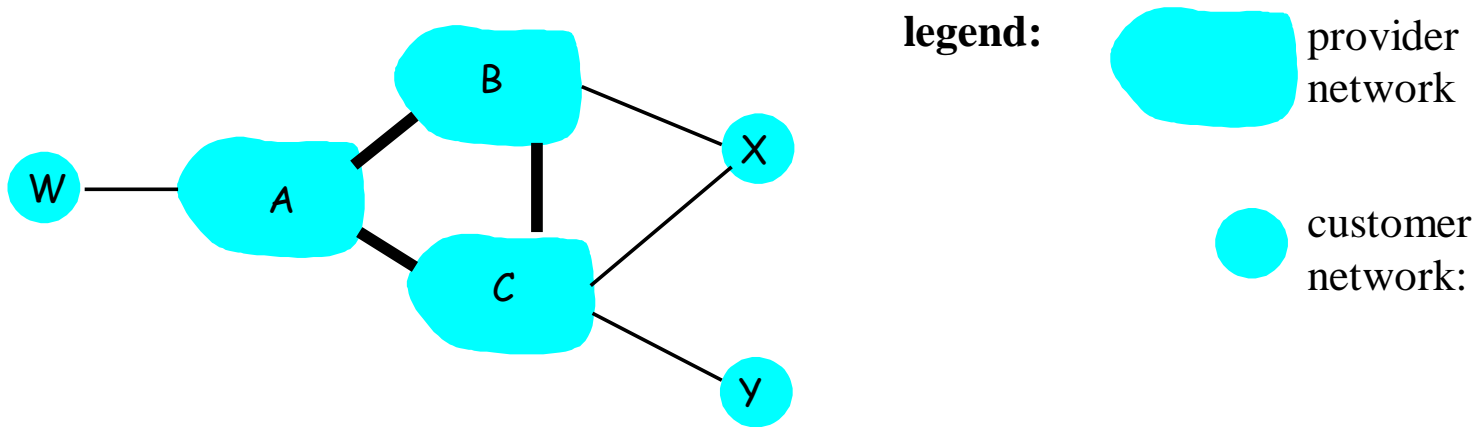
# BGP route selection

- ❑ Router may learn about more than 1 route to some prefix. Router must select route.
- ❑ Elimination rules:
  1. Local preference value attribute: policy decision
  2. Shortest AS-PATH
  3. Closest NEXT-HOP router: hot potato routing
  4. Additional criteria

# BGP messages

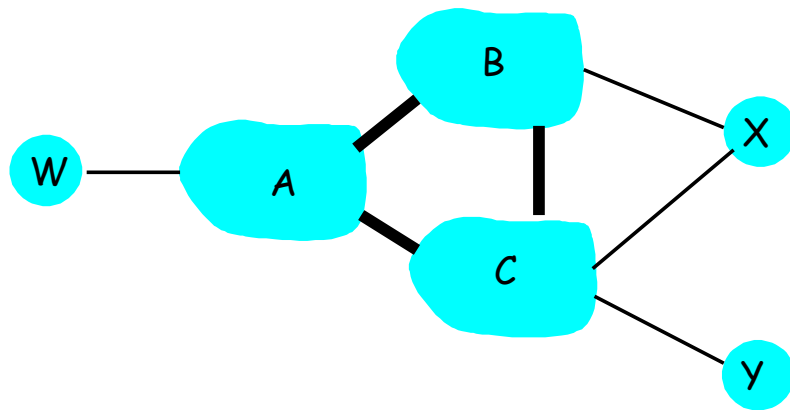
- ❑ BGP messages exchanged using TCP.
- ❑ BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE** keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection

# BGP routing policy




- A,B,C are **provider networks**
- X,W,Y are customer (of provider networks)
- X is **dual-homed**: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

## BGP routing policy (2)



legend:

 provider network

 customer network:

- ❑ A advertises to B the path AW
- ❑ B advertises to X the path BAW
- ❑ Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route **only** to/from its customers!

# Why different Intra- and Inter-AS routing ?

## Policy:

- ❑ Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- ❑ Intra-AS: single admin, so no policy decisions needed

## Scale:

- ❑ hierarchical routing saves table size, reduced update traffic

## Performance:

- ❑ Intra-AS: can focus on performance
- ❑ Inter-AS: policy may dominate over performance

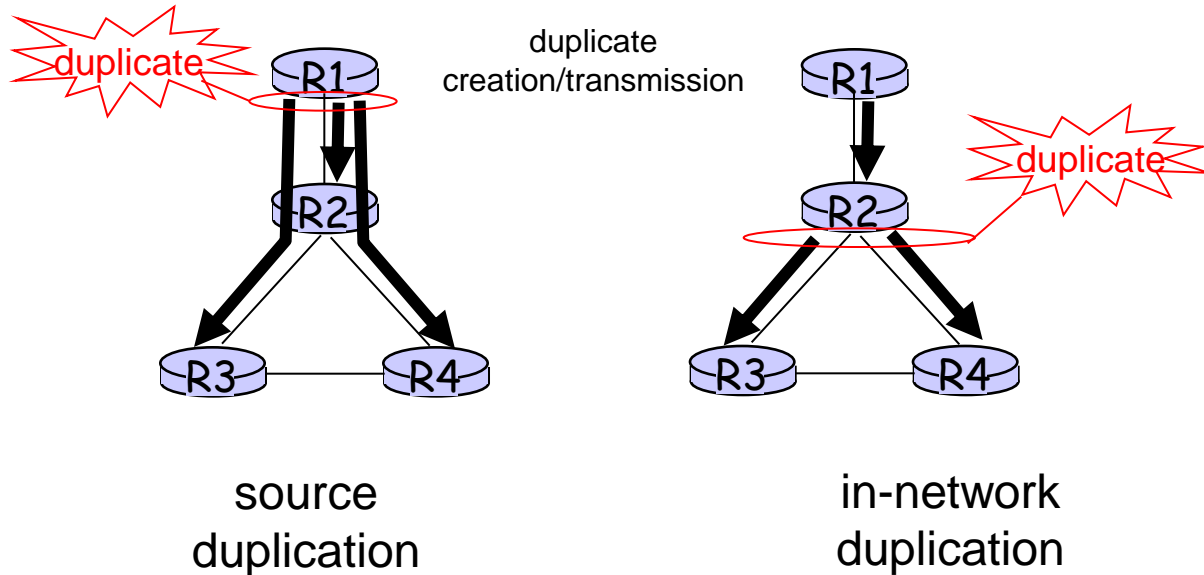
# Chapter 4: Network Layer

- ❑ 4.1 Introduction
- ❑ 4.2 Virtual circuit and datagram networks
- ❑ 4.3 What's inside a router
- ❑ 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- ❑ 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ 4.7 Broadcast and multicast routing



# Broadcast Routing

- ❑ Deliver packets from source to all other nodes
- ❑ Source duplication is inefficient:



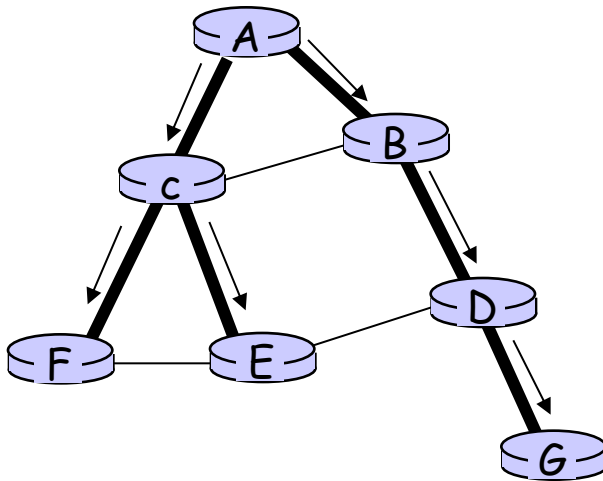
- ❑ Source duplication: how does source determine recipient addresses?

# In-network duplication

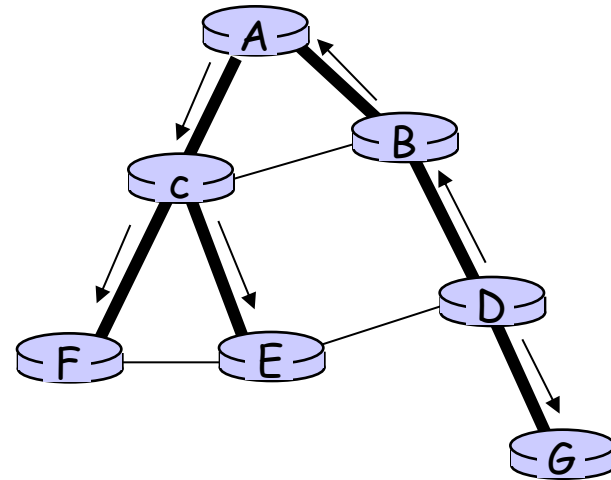
- ❑ Flooding: when node receives brdcst pckt, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- ❑ Controlled flooding: node only brdcsts pkt if it hasn't brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source
- ❑ Spanning tree
  - No redundant packets received by any node

# Spanning Tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree



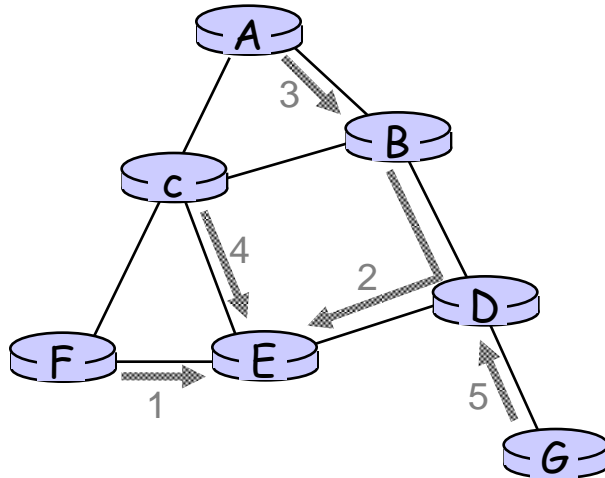
(a) Broadcast initiated at A



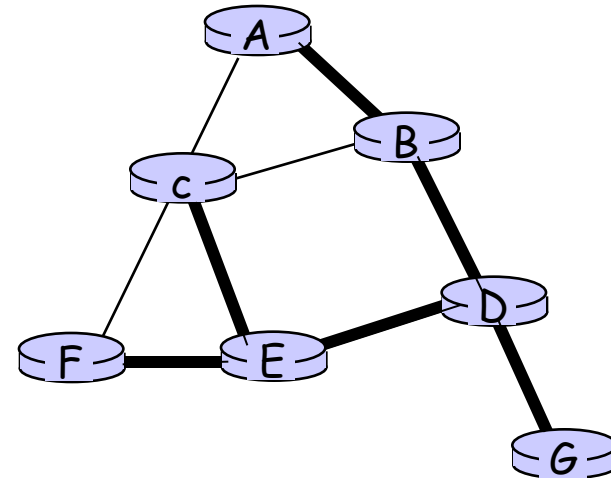
(b) Broadcast initiated at D

# Spanning Tree: Creation

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree



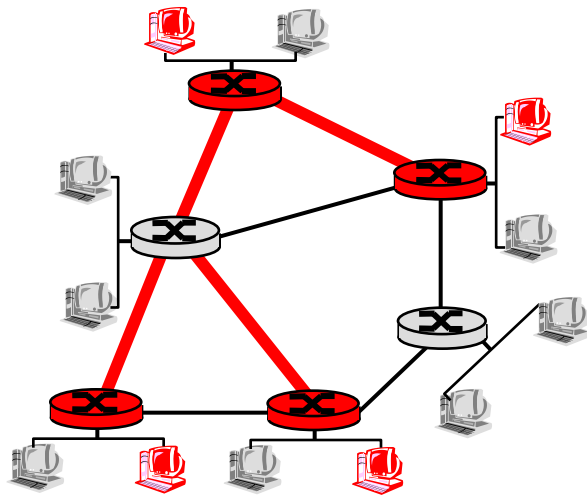
**(a) Stepwise construction of spanning tree**



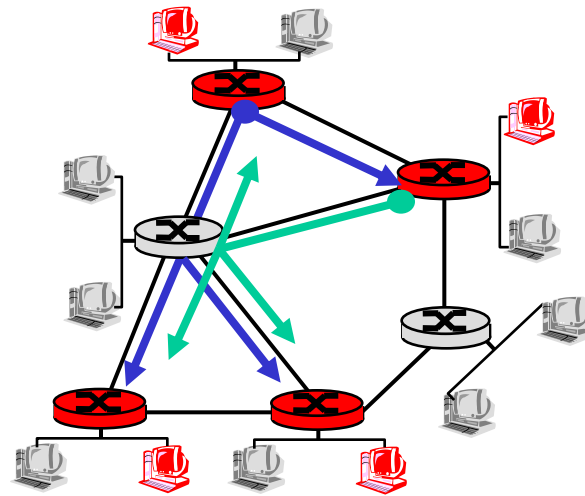
**(b) Constructed spanning tree**

# Multicast Routing: Problem Statement

- **Goal:** find a tree (or trees) connecting routers having local mcast group members
  - **tree:** not all paths between routers used
  - **source-based:** different tree from each sender to rcvrs
  - **shared-tree:** same tree used by all group members



Shared tree



Source-based trees

# Approaches for building mcast trees

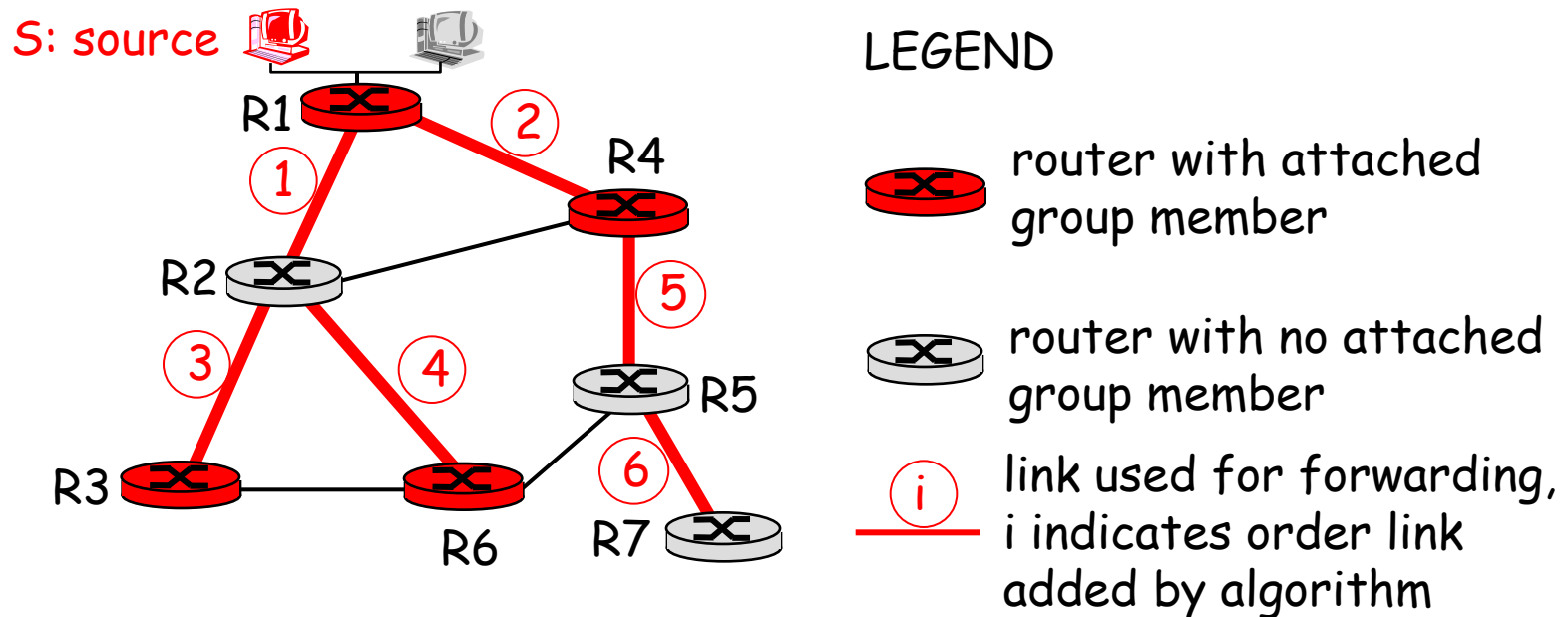
Approaches:

- **source-based tree:** one tree per source
  - shortest path trees
  - reverse path forwarding
- **group-shared tree:** group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

...we first look at basic approaches, then specific protocols adopting these approaches

# Shortest Path Tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm



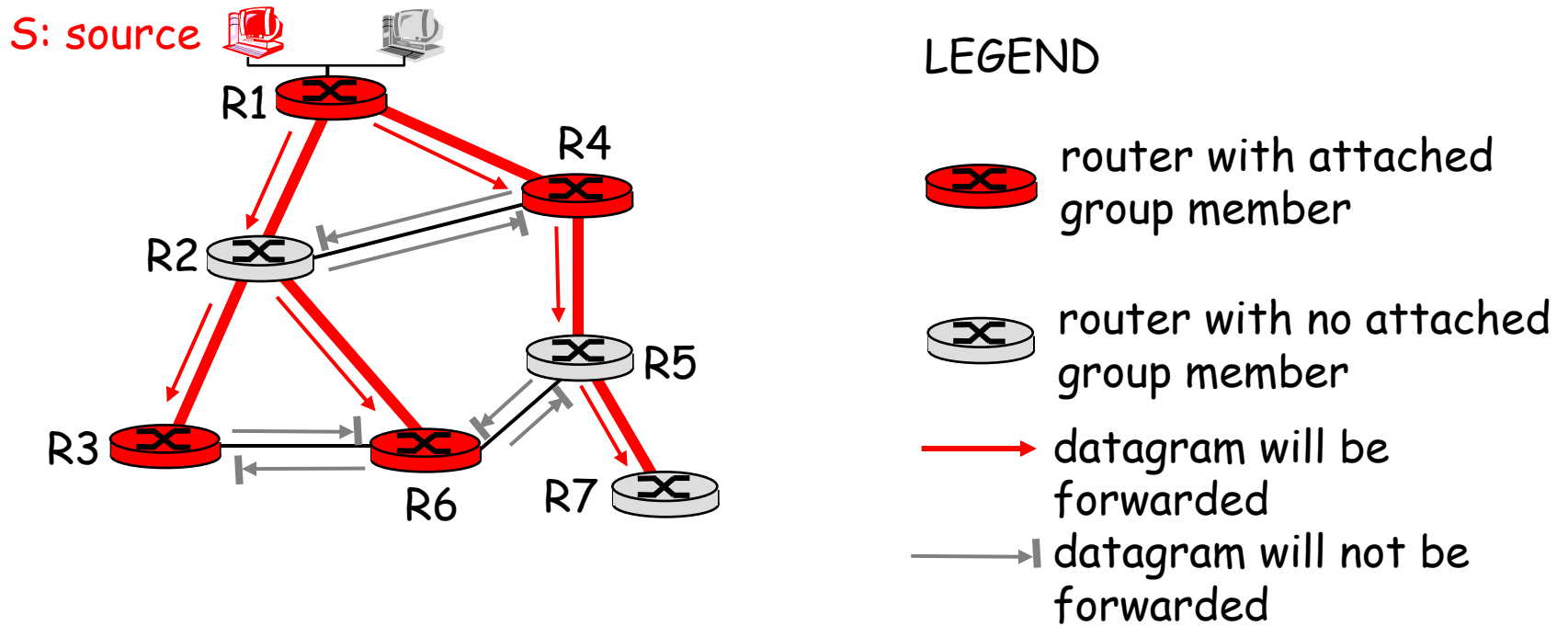
# Reverse Path Forwarding

- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

*if* (mcast datagram received on incoming link  
on shortest path back to center)  
*then* flood datagram onto all outgoing links  
*else* ignore datagram



# Reverse Path Forwarding: example

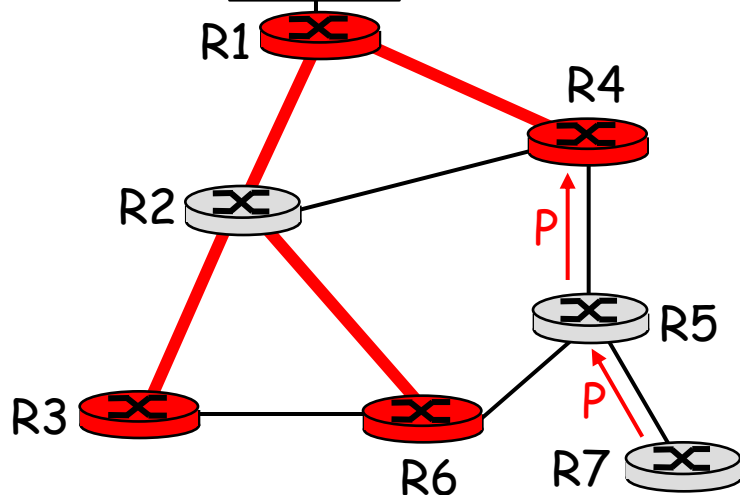


- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

# Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members

S: source



## LEGEND



router with attached group member



router with no attached group member



prune message



links with multicast forwarding

# Shared-Tree: Steiner Tree

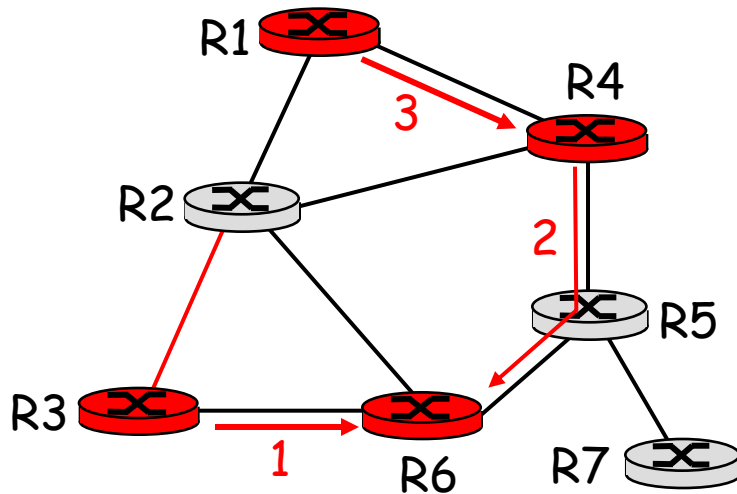
- ❑ **Steiner Tree:** minimum cost tree connecting all routers with attached group members
- ❑ problem is NP-complete
- ❑ excellent heuristics exists
- ❑ not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

# Center-based trees




- ❑ single delivery tree shared by all
- ❑ one router identified as "*center*" of tree
- ❑ to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

# Center-based trees: an example

Suppose R6 chosen as center:



## LEGEND

-  router with attached group member
-  router with no attached group member
-  path order in which join messages generated

# Internet Multicasting Routing: DVMRP

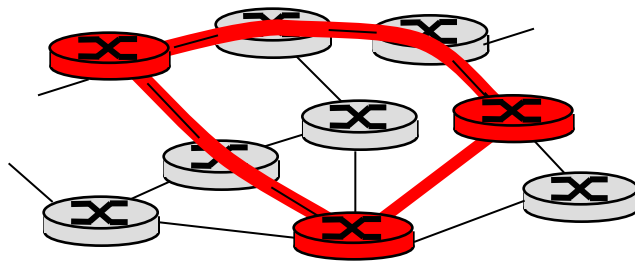
- **DVMRP**: distance vector multicast routing protocol, RFC1075
- ***flood and prune***: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

# DVMRP: continued...

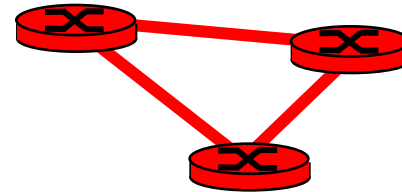
- soft state: DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: re prune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP

# Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- ❑ mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- ❑ normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- ❑ receiving mcast router unencapsulates to get mcast datagram



# PIM: Protocol Independent Multicast

- ❑ not dependent on any specific underlying unicast routing algorithm (works with all)
- ❑ two different multicast distribution scenarios :

## Dense:

- ❑ group members densely packed, in "close" proximity.
- ❑ bandwidth more plentiful

## Sparse:

- ❑ # networks with group members small wrt # interconnected networks
- ❑ group members "widely dispersed"
- ❑ bandwidth not plentiful

# Consequences of Sparse-Dense Dichotomy:

## Dense

- ❑ group membership by routers *assumed* until routers explicitly prune
- ❑ *data-driven* construction on mcast tree (e.g., RPF)
- ❑ bandwidth and non-group-router processing *profligate*

## Sparse:

- ❑ no membership until routers explicitly join
- ❑ *receiver-driven* construction of mcast tree (e.g., center-based)
- ❑ bandwidth and non-group-router processing *conservative*

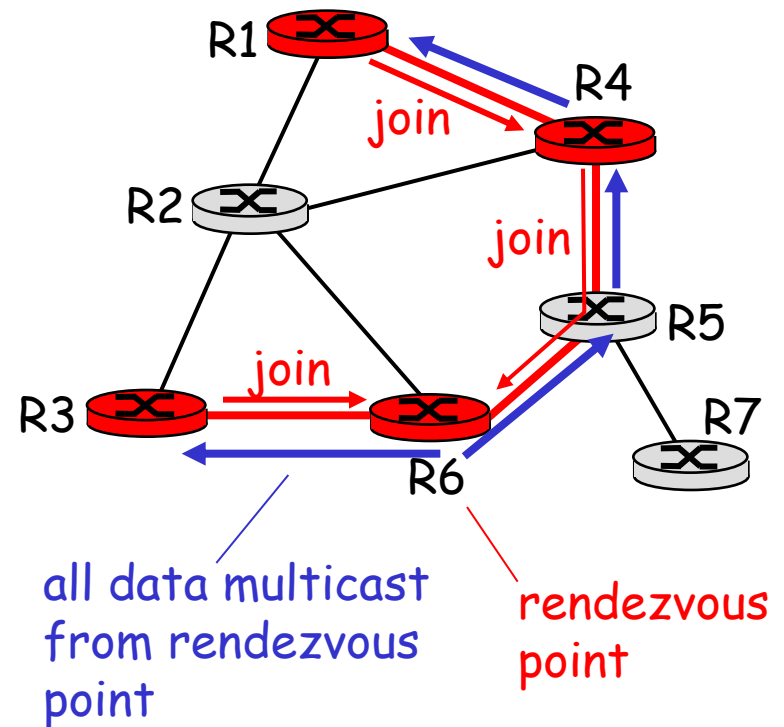
# PIM- Dense Mode

**flood-and-prune RPF**, similar to DVMRP but

- ❑ underlying unicast protocol provides RPF info for incoming datagram
- ❑ less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- ❑ has protocol mechanism for router to detect it is a leaf-node router

# PIM - Sparse Mode

- ❑ center-based approach
- ❑ router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- ❑ after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



# PIM - Sparse Mode

sender(s):

- ❑ unicast data to RP, which distributes down RP-rooted tree
- ❑ RP can extend mcast tree upstream to source
- ❑ RP can send *stop* msg if no attached receivers
  - "no one is listening!"

