PTC 2550 - Aula 18

4.4 Protocolos para aplicativos de conversação em tempo real

(Kurose, p. 587 - 626)
(Peterson, p. 444-454)

07/06/2017

Multimídia: estrutura de tópicos

4.1 Aplicativos de rede multimídia

4.2 *Streaming* de vídeo armazenado

- 4.2.1 Distribuição de vídeo e CDN
- 4.2.2 Ferramentas para reprodução contínua

4.3 Voz-sobre-IP

4.4 Protocolos para aplicativos de conversação em tempo real

4.5 Suporte de rede para multimídia

Suporte à rede para multimídia

Approach	Granularity	Guarantee	Mechanisms	Complex	Deployed?
Making best of best effort service	All traffic treated equally	None or soft	No network support (all at application)	low	everywhere
Differentiated service	Traffic "class"	None of soft	Packet market, scheduling, policing.	med	some
Per-connection QoS	Per-connection flow	Soft or hard after flow admitted	Packet market, scheduling, policing, call admission	high	little to none

Cor indica nível de utilização hoje.

Suporte à rede para multimídia

- Suporte ao multimídia usando “melhor esforço”

Dimensionando redes de "melhor esforço"

- *abordagem:* implanter capacidade de enlace suficiente de modo que congestionamentos não ocorrem, tráfego multimídia flui sem atraso ou perdas
  - baixa complexidade dos mecanismos de rede (usa a atual rede "melhor esforço")
  - altos custos para altas taxas
- desafios:
  - *network dimensioning* (arquitetura da rede; localização de roteadores e servidores) *bandwidth provisioning (quanta banda é suficiente?)*
- *Para isso é necessário:*
  - *estimar demanda de tráfego da rede*
  - Requisitos de desempenho bem definidos (QoS)
  - Modelos para predizer desempenho fim a fm para um dado modelo de carga; técnicas para encontrar a alocação de banda de custo mínimo que resulta nos requisitos atendidos
- Exemplos de pesquisa: aqui e aqui.

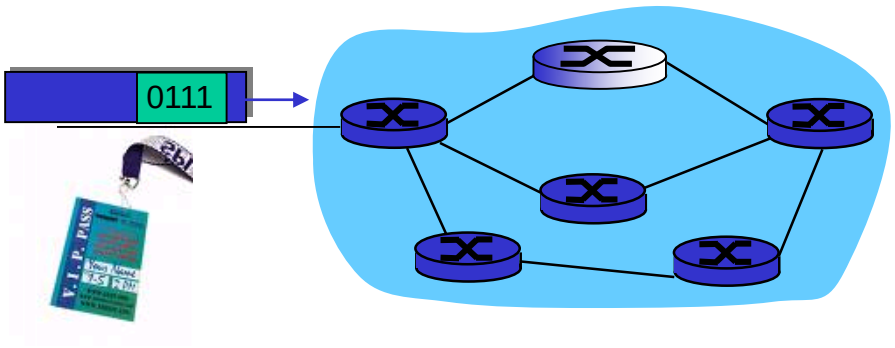
Suporte à rede para multimídia

- Suporte ao multimídia usando classes de serviços

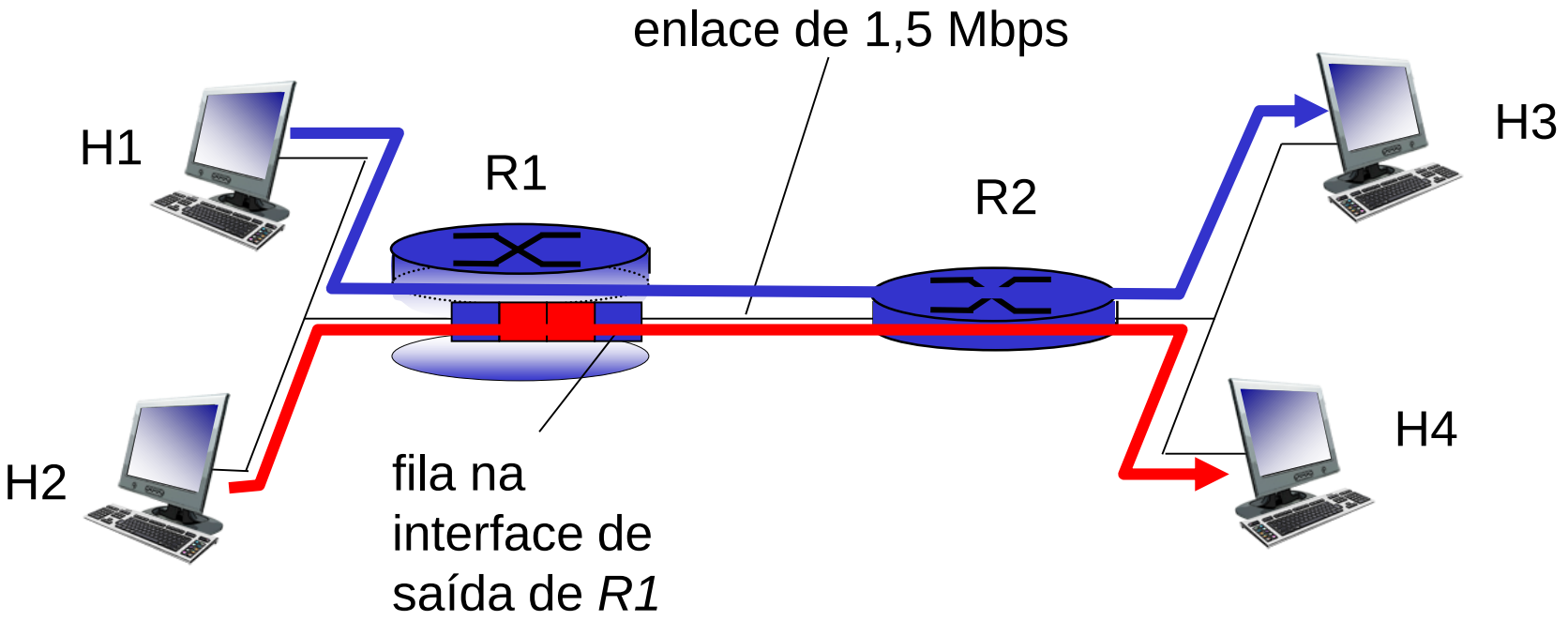
Fornecendo múltiplas classes de serviços

- até aqui: fazendo o possível com serviço de melhor esforço
  - modelo de serviço "tamanho único"
- alternativa: múltiplas classes de serviços
  - particionar o tráfego em classes
  - a rede trata diferentes classes de tráfego de forma diferente (analogia: primeira classe / classe executiva / classe econômica)

- granularidade serviço diferenciado entre múltiplas classes, não entre conexões individuais
- histórico: bits ToS – já previstos no *Internet Protocolo* [RFC 760] em 1980

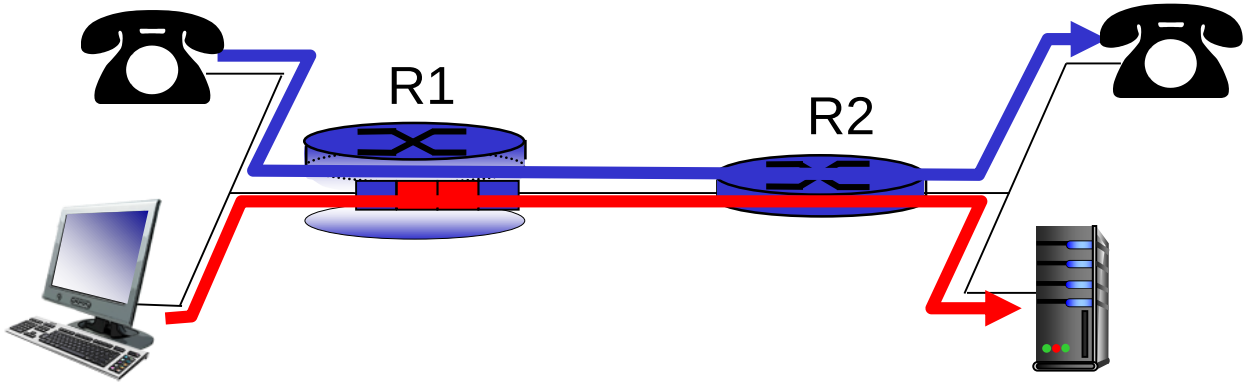


Múltiplas classes de serviço: cenário



Cenário 1: mistura de HTTP e VoIP

- Exemplo: VoIP de Mbps e HTTP compartilham enlace de1,5 Mbps.
  - rajadas HTTP podem congestionar enlace e causar perda de áudio
  - deseja-se dar prioridade ao áudio em relação ao HTTP

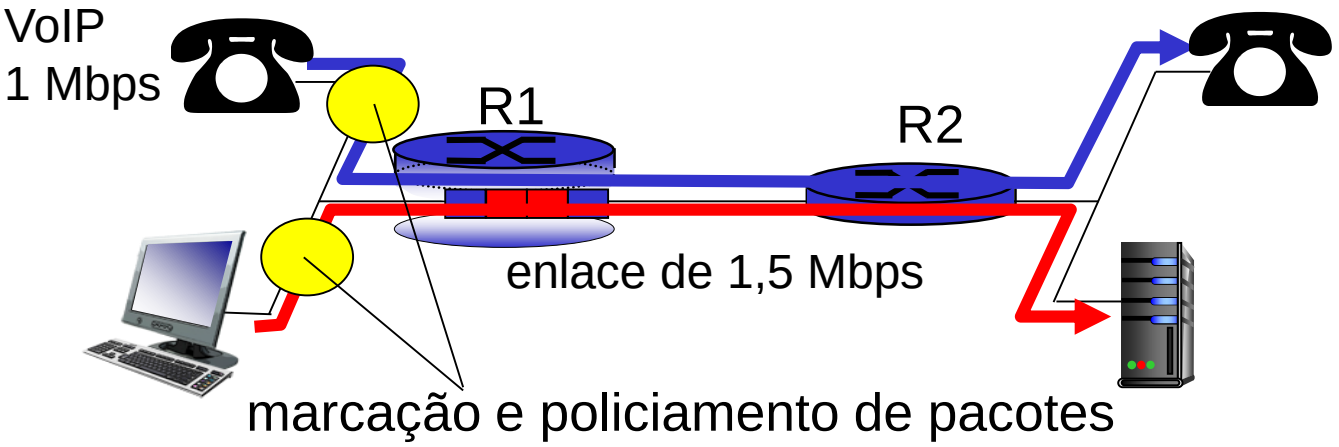


Princípio 1

Marcação de pacote necessária para roteador poder distinguir entre diferentes classes; e nova política para o roteador tratar pacotes de acordo

Princípios para garantir QoS (cont.)

- e se aplicações "não se comportarem"? (VoIP envia a taxas maiores do que as declaradas)
  - policiamento (*policing*): forçar fonte a aderir a alocações de banda
- *marcação* e *policiamento* na borda da rede

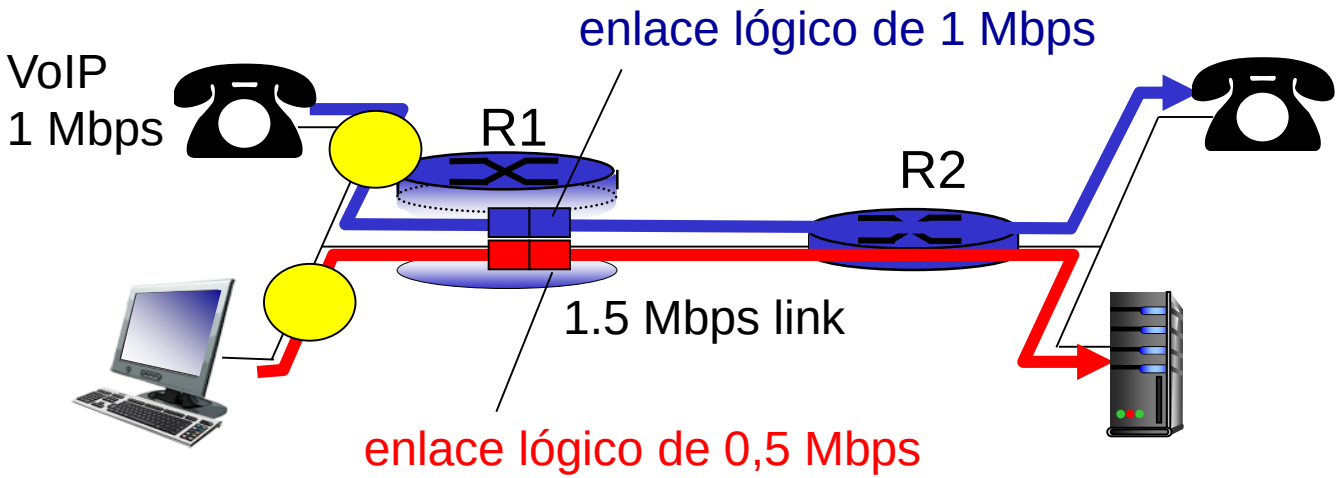


Princípio 2

Prover proteção (isolamento) de uma classe em relação às outras

Princípios para garantia de QoS (cont.)

- alocar largura de banda *fixa* (não-partilhável) para fluxo: uso ineficiente da banda se fluxo não usa recurso alocado ☹



Princípio 3

Mesmo provendo isolamento, o uso dos recursos da forma mais eficiente possível é sempre desejável.

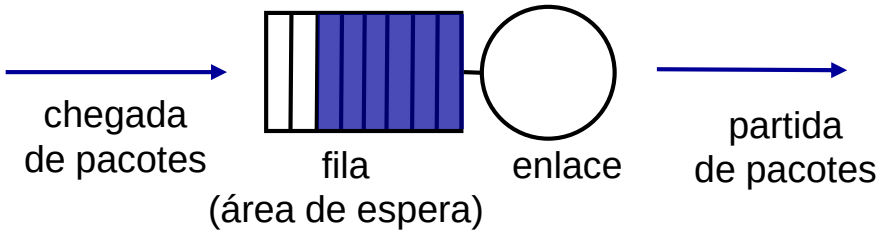
Mecanismos de agendamento e policiamento

- *Como implementar princípios??*

- *Mecanismos de agendamento e policiamento nos roteadores*

Mecanismos de agendamento

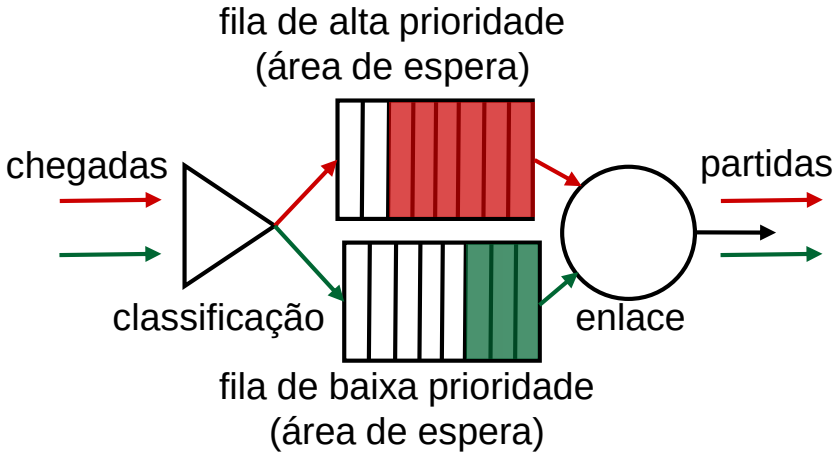
- *agendamento:* escolher próximo pacote a ser enviado pelo enlace
- *agendamento FIFO (first in first out) :* enviar na ordem de chegada à fila
  - exemplo do mundo real?
  - *politica de descarte:* se pacote chegam a uma fila cheia: qual descartar?
    - *tail drop:* elimina pacote chegando
    - *prioridade:* elimina/remove pacote baseado em prioridade
    - *aleatório:* elimina/remove aleatoriamente

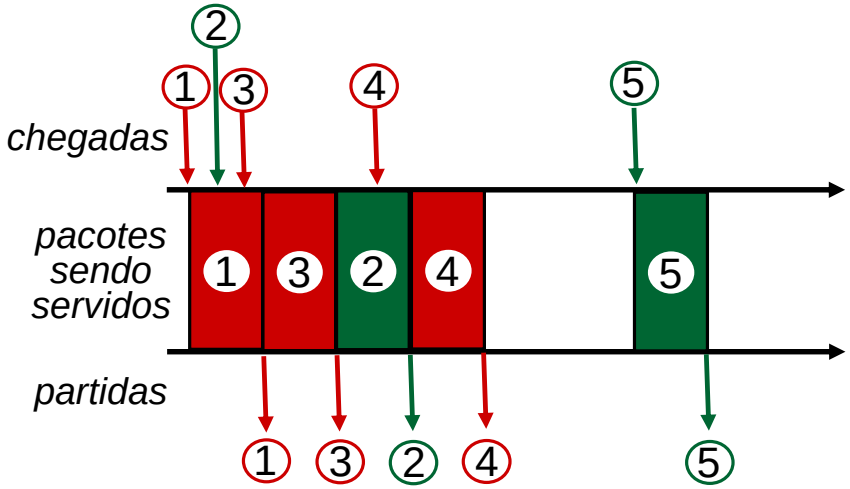


Mecanismos de agendamento: prioridade

agendamento por prioridade: enviar pacote em fila com maior prioridade

- múltiplas *classes*, com diferentes prioridades
  - classe pode depender de informações no cabeçalho como endereço IP de fonte/destino, número de porta, etc.
  - exemplo do mundo real?

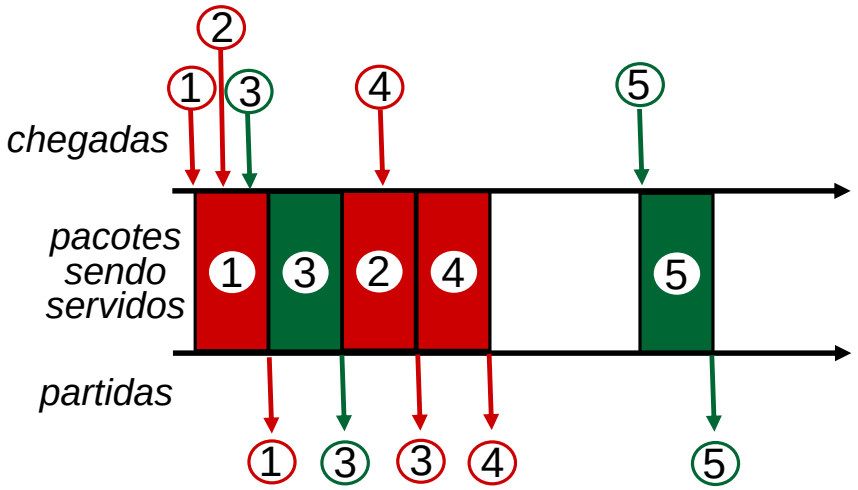




Mecanismos de agendamento (cont.)

agendamento Round Robin (RR) (rodízio):

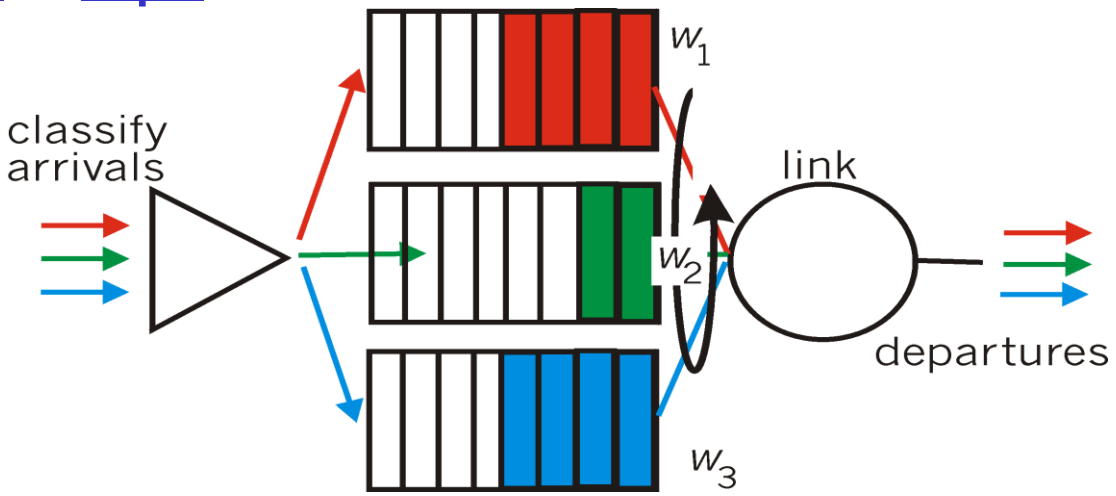
- múltiplas classes
- varre ciclicamente as filas, enviando um pacote complete de cada classe (se disponível)
- exemplo de tempo real?



Mecanismos de agendamento (cont.)

Weighted Fair Queuing (WFQ):

- *Round Robin* generalizado
- cada classe rece quantidade de serviço ponderado a cada ciclo
- exemplo do mundo real?
- Mais: aqui e aqui



Mecanismos de Policiamento

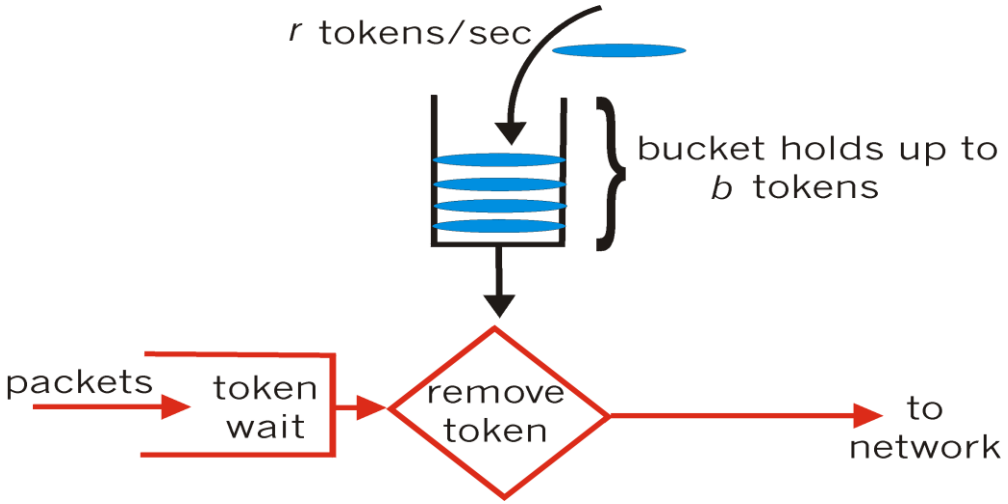
Objetivo: limitar tráfego de forma que ele não exceda parâmetros declarados

3 critérios comumente usados:

- *taxa media (de longo prazo) :* quantos pacotes podem ser enviados por unidade de tempo (no longo prazo)
  - questão chave: qual interval usar? 100 pacotes/s ou 6000 pacotes/min tem mesma taxa media (mas a 1a é mais restritiva…)
- *taxa de pico:* por exemplo, 6000 pacotes/min médio; 1500 pacotes/s de taxa de pico
- *(máxima) duração de rajada (burst):* máximo numero de pacotes enviados consecutivamente (sem intervalo oscioso)

Mecanismos de Policiamento: implementação

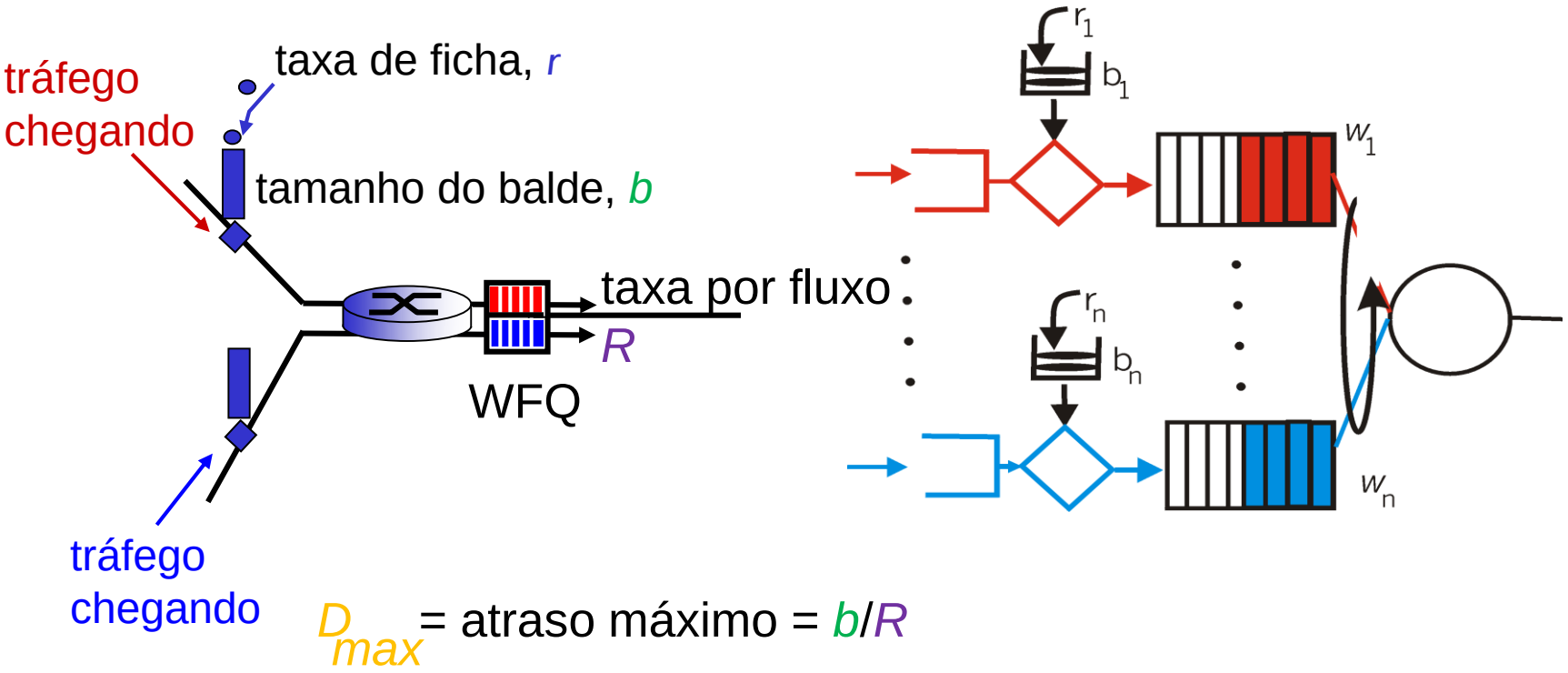
balde de fichas (token bucket ou leaky bucket): limita entrada a duração de rajada e taxa média especificada



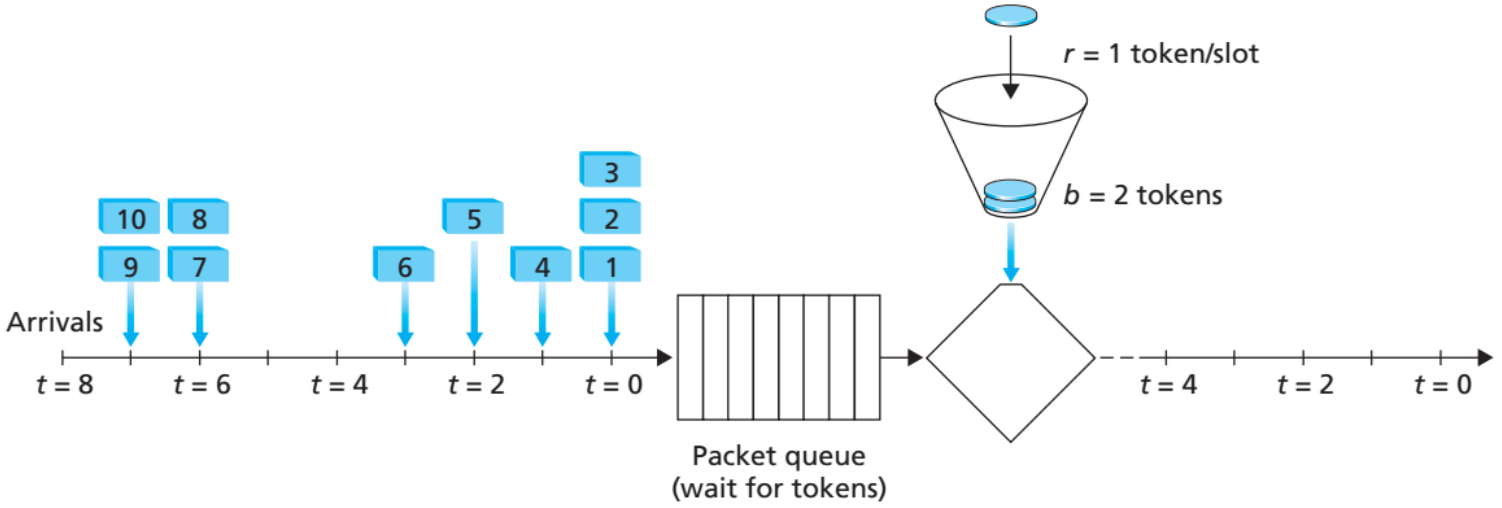
- balde tem capacidade para *b* fichas
- novas fichas são geradas à taxa de *r fichas/s* a menos que o balde esteja cheio
- *em um intervalo de duração t: número de pacotes admitidos é menor ou igual a (r t + b)*

Policiamento e garantia de QoS

- balde de fichas e WFQ podem ser combinadas para garantir limite máximo de atraso, i.e., garantia de QoS



P20. Consider the figure below, which shows a leaky bucket policer being fed by a stream of packets. The token buffer can hold at most two tokens, and is initially full at $t = 0$. New tokens arrive at a rate of one token per slot. The output link speed is such that if two packets obtain tokens at the beginning of a time slot, they can both go to the output link in the same slot. The timing details of the system are as follows:



1. Packets (if any) arrive at the beginning of the slot. Thus in the figure, packets 1, 2, and 3 arrive in slot 0. If there are already packets in the queue, then the arriving packets join the end of the queue. Packets proceed towards the front of the queue in a FIFO manner.
2. After the arrivals have been added to the queue, if there are any queued packets, one or two of those packets (depending on the number of available tokens) will each remove a token from the token buffer and go to the output link during that slot. Thus, packets 1 and 2 each remove a token from the buffer (since there are initially two tokens) and go to the output link during slot 0.
3. A new token is added to the token buffer if it is not full, since the token generation rate is $r = 1$ token/slot.
4. Time then advances to the next time slot, and these steps repeat.

Answer the following questions:

a. For each time slot, identify the packets that are in the queue and the number of tokens in the bucket, immediately after the arrivals have been processed (step 1 above) but before any of the packets have passed through the queue and removed a token. Thus, for the $t = 0$ time slot in the example above, packets 1, 2 and 3 are in the queue, and there are two tokens in the buffer.

b. For each time slot indicate which packets appear on the output after the token(s) have been removed from the queue. Thus, for the $t = 0$ time slot in the example above, packets 1 and 2 appear on the output link from the leaky buffer during slot 0.

Implementação: Differentiated services (*Diffserv*) [RFC 2475]

- deseja-se classes de serviço com "qualidades" diferentes
  - "comportamento como de fio" para VoIP
  - distinção relativa de serviços: assinatura platina, ouro, prata,…
- *Escalabilidade:* funções simples no núcleo da rede, funções relativamente complexas nos roteadores de borda (ou *hosts*)
  - sinalização, manter estado do roteador por fluxo é difícil para um grande número de fluxos
- Não definem-se classes de serviços, apenas prevê-se os componentes funcionais para construir classes de serviços

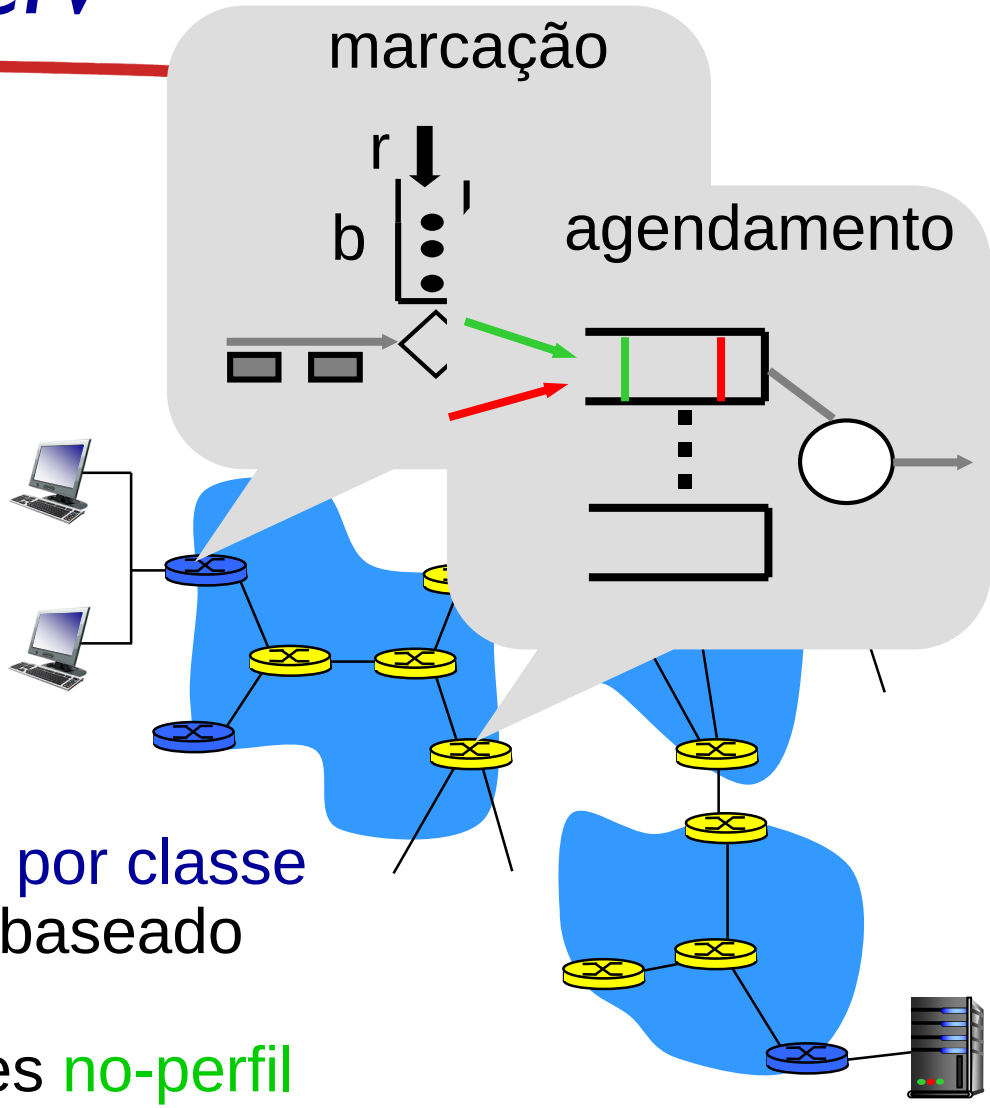
Arquitetura *Diffserv*

roteador de borda:

- gerenciamento de tráfego por fluxo
- marca pacotes como no-perfil e fora-do-perfil

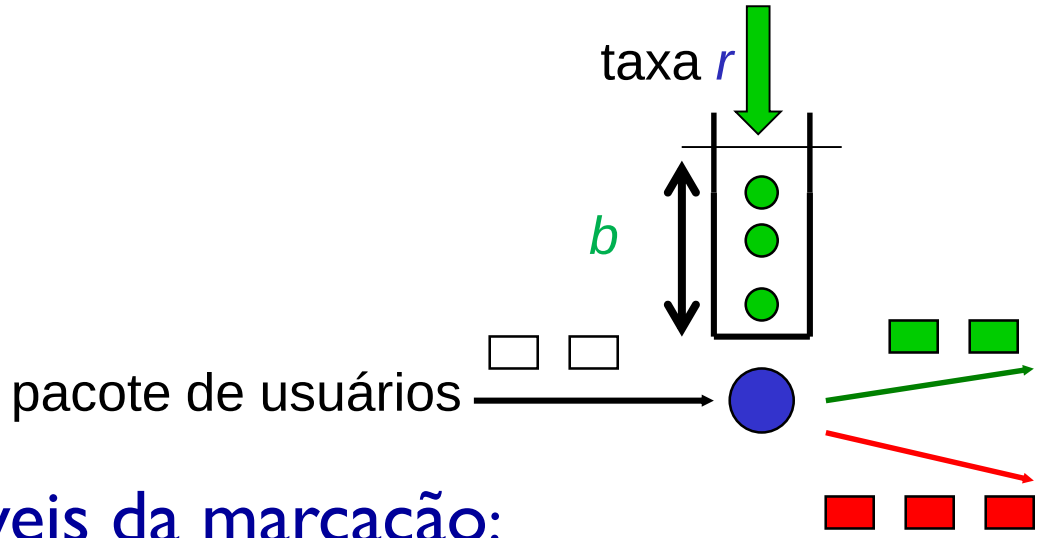
roteador do núcleo:

- gerenciamento de tráfego por classe
- agendamento e *buffering* baseado na marcação na borda
- preferência dada a pacotes no-perfil em relação aos pacotes *fora-do-perfil*



Marcação de pacotes nos roteadores de borda

- perfil: taxa pré-negociada *r*, capacidade do balde *b*
- marcação de pacote na borda baseada em um perfil *por fluxo*



usos possíveis da marcação:

- marcação baseada em classe: pacotes de diferentes classes marcados diferentemente
- marcação intraclasse: porção "de acordo" do tráfego marcada diferentemente da porção em desacordo

Marcação de pacotes *Diffserv*: detalhes

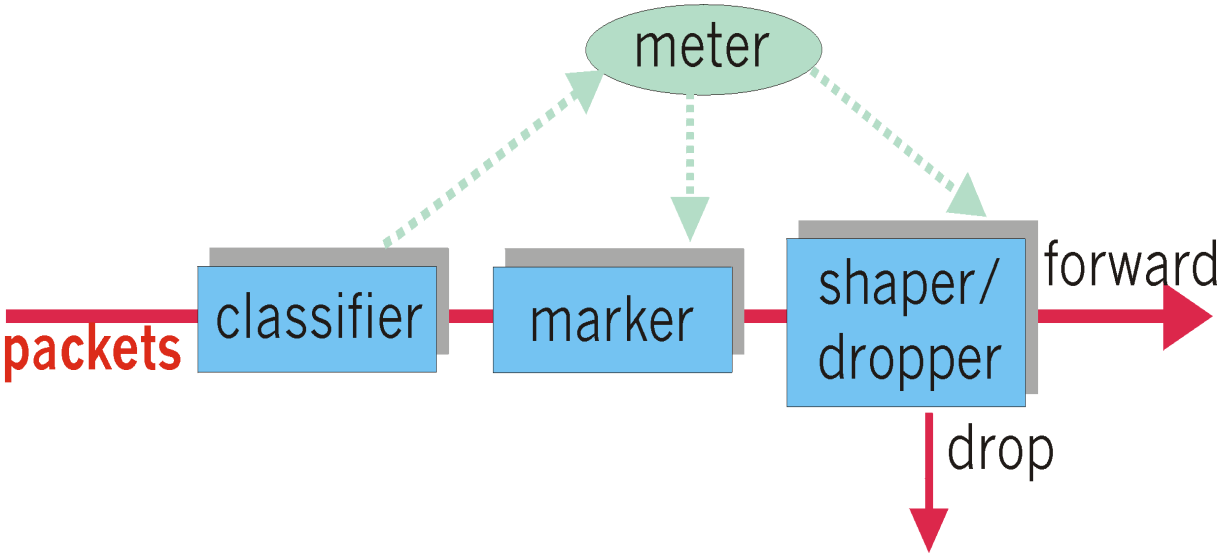
- pacotes são marcados no campo *Type of Service* (TOS) no IPv4, e Classe de Tráfego no IPv6 [RFC3260]
- 6 bits usados para *Differentiated Service Code Point* (DSCP)
  - determina PHB que o pacote receberá
  - 2 bits não usados

  - Analogia: credenciais na entrada de um evento

Classificação, condicionamento

pode ser desejável limitar a entrada de tráfego de algumas classes:

- usuário declara perfil de tráfego (taxa, tamanho de rajada)
- mede-se o tráfego, formatado se em não conformidade



Repasse *Per-hop Behavior* (PHB)

- PHB resulta em uma diferença *observável (mensurável)* no repasse de diferentes classes
- PHB não especifica qual mecanismo usar para garantir comportamento de desempenho PHB necessário
- Exemplos:
  - classe *A* recebe *x*% da largura de banda do enlace de saída em intervalos de tempo de duração especificada
  - pacotes classe *A* enviados antes de pacotes classe *B*

Repasse PHB

PHBs propostos:

- *expedited forwarding [RFC3246]*: taxa de envio de pacotes de uma classe iguala ou excede uma taxa especificada (DSCP 101 110 ou 46)
  - ençace lógico com taxa mínima garantida
- *assured forwarding [RFC2597]*: 4 classes de tráfego
  - cada uma tem garantida mínima quantidade de banda
  - cada com 3 partições de preferências de eliminação de pacotes (*drop*)

Assured Forwarding (AF) behavior group				
	Class 1	Class 2	Class 3	Class 4
Low drop probability	AF11 (DSCP 10)	AF21 (DSCP 18)	AF31 (DSCP 26)	AF41 (DSCP 34)
Med drop probability	AF12 (DSCP 12)	AF22 (DSCP 20)	AF32 (DSCP 28)	AF42 (DSCP 36)
High drop probability	AF13 (DSCP 14)	AF23 (DSCP 22)	AF33 (DSCP 30)	AF43 (DSCP 38)

Problemas para o *Diffserv*

- Redes não proprietárias: como garantir QoS fim-a-fim sobre múltiplos ISP?

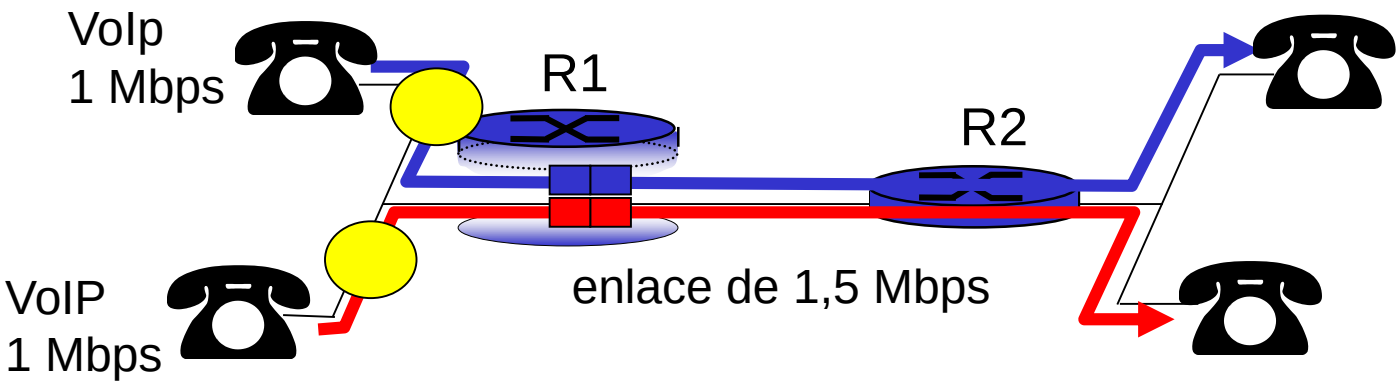
"A maior desvantagem do *DiffServ* é que, ao seu mais alto nível, pode ser considerada como uma solução técnica para um problema técnico que não existe."

Suporte à rede para multimídia

- QoS trabalhando conexão a conexão

Garantia de QoS por conexão

- *fato básico:* não é possível suportar demandas de tráfego acima da capacidade do enlace

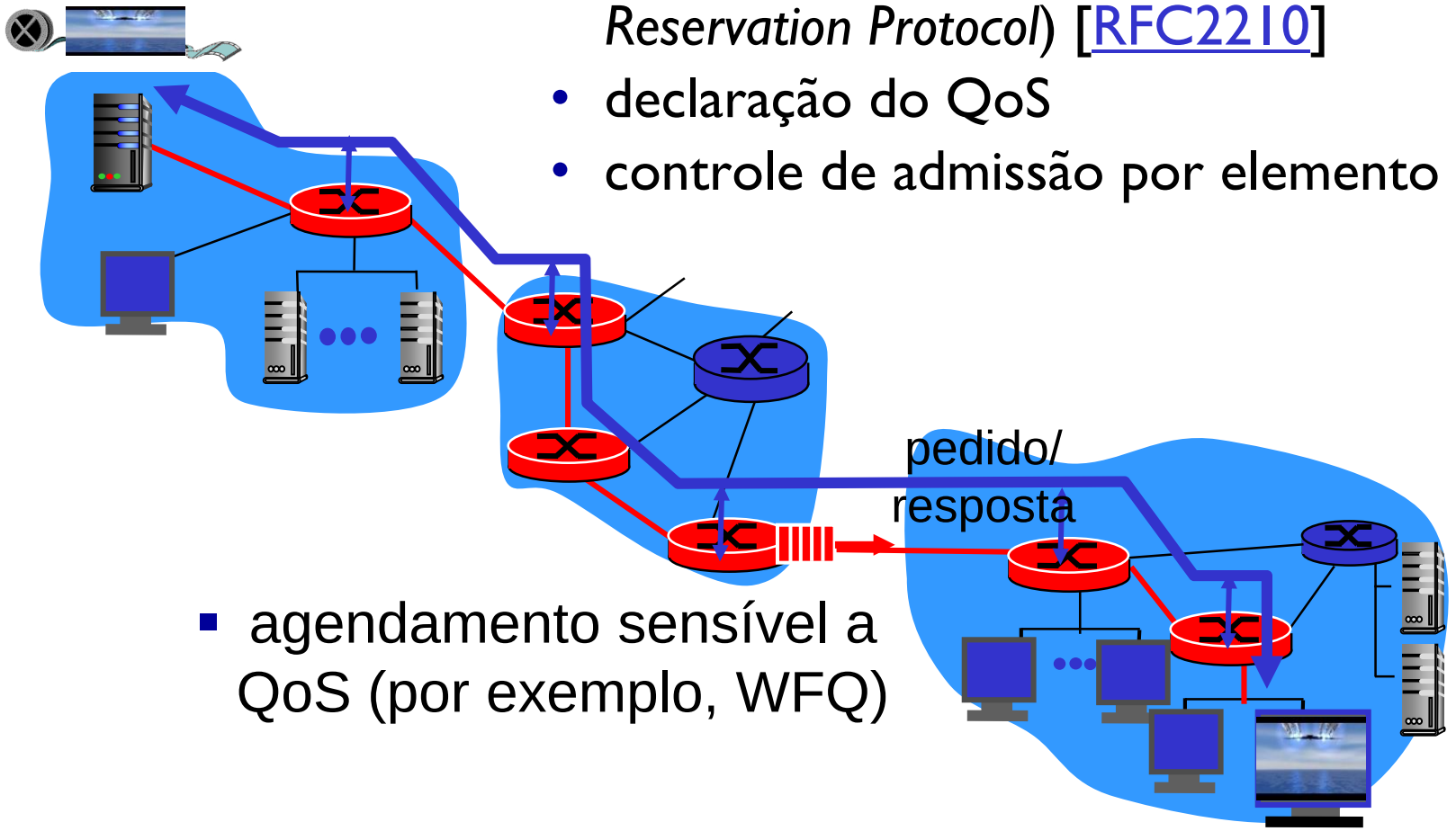


Princípio 4

admissão de chamada: fluxo declara suas necessidades, rede pode bloquear chamada (sinal de ocupado) se não consegue atendê-las

Cenário de garantia de QoS

- *reserva de recursos*
  - *call setup*, sinalização (RSVP - *Resource Reservation Protocol*) [RFC2210]
  - declaração do QoS
  - controle de admissão por elemento



- agendamento sensível a QoS (por exemplo, WFQ)

Problemas da garantia de QoS por conexão

- Complexo, caro, envolve todos os roteadores da conexão se comunicarem

- Muita pesquisa, mas pouca implementação: mecanismos da camada de aplicação mais dimensionamento adequado da rede já fornecem serviço "suficientemente bom"
- Assim, quem vai querer pagar? ☺