



# **Mecanismos de Escalonamento e Descarte de Pacotes em Redes com Comutação de Pacotes**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa (asou@ua.pt)

DETI-UA, 2020/2021

# Mecanismos de escalonamento e descarte de pacotes

Considere-se uma rede com comutação de pacotes. Em cada interface de saída de cada ligação de um nó, existe uma fila de espera para condicionar temporariamente os pacotes a transmitir pela interface. Em cada interface de saída de cada ligação:

Disciplina de escalonamento: decide a ordem pela qual são transmitidos pela ligação os pacotes de diferentes fluxos que estão na fila de espera

- impõe assim diferentes atrasos médios (*average delays*) a diferentes fluxos ao definir a ordem de transmissão dos pacotes.

Método de descarte de pacotes: decide a forma como os pacotes dos diferentes fluxos são aceites na fila de espera quando a ligação está ocupada com a transmissão de outro pacote

- impõe assim diferentes taxas de perda de pacotes (*packet loss rates*) a diferentes fluxos ao definir que pacotes são descartados.

# Sumário do Módulo

## Primeira Parte:

- Caracterização das disciplinas de escalonamento de pacotes

## Segunda Parte:

- Disciplinas de escalonamento de pacotes: FIFO, com prioridades e que funcionam de forma rotativa

## Terceira Parte

- Disciplinas de escalonamento de pacotes que funcionam por aproximação ao sistema GPS

## Quarta Parte

- Métodos de descarte de pacotes
- Ilustração da combinação de disciplinas de escalonamento com métodos de descarte de pacote na arquitectura *DiffServ* do IETF.



# **Mecanismos de Escalonamento e Descarte de Pacotes em Redes com Comutação de Pacotes**

**Primeira parte:**

- **Caracterização das disciplinas de escalonamento de pacotes**

# Equidade das disciplinas de escalonamento

Quando uma ligação está congestionada (*i.e.*, a sua fila de espera não está vazia), o problema mais básico que se coloca à função de escalonamento é:

divisão de um recurso escasso por fluxos com iguais direitos mas com diferentes necessidades de utilização desse recurso.

Idealmente, a atribuição deve ser feita de acordo com o princípio de equidade *max-min*:

- Os recursos são atribuídos aos fluxos por ordem crescente de necessidade.
- A nenhum fluxo é atribuída uma quantidade de recursos maior do que a sua necessidade.
- A fluxos cuja necessidade não tenha sido satisfeita é atribuída uma igual quantidade de recursos.

# Equidade max-min com direitos iguais

Considere-se:

- um conjunto de fluxos  $1, 2, \dots, n$  com necessidades  $x_1, x_2, \dots, x_n$  e ordenados pelas suas necessidades ( $x_1 \leq x_2 \leq \dots \leq x_n$ );
- uma ligação com capacidade  $C$ .

A atribuição dos recursos da ligação é efetuada do seguinte modo:

- Inicialmente todos os fluxos têm direito a  $d = C/n$
- $d$  é menor que  $x_1$ ?
  - se sim, atribui-se  $d$  a todos os fluxos, i.e., aos fluxos  $1, 2, \dots, n$ ;
  - se não, atribui-se  $x_1$  ao fluxo 1 e os fluxos  $2, 3, \dots, n$  têm direito a  $d = d + (d - x_1)/(n - 1)$
- $d$  é menor que  $x_2$ ?
  - se sim, atribui-se  $d$  aos fluxos  $2, 3, \dots, n$ ;
  - se não, atribui-se  $x_2$  ao fluxo 2 e os fluxos  $3, 4, \dots, n$  têm direito a  $d = d + (d - x_2)/(n - 2)$
- E assim sucessivamente...

## Exemplo 1

Considere-se uma ligação com capacidade de 128 Mbps e 4 fluxos de tráfego de 8, 36, 48 e 128 Mbps. Determine que recursos são atribuídos a cada fluxo pelo princípio de equidade max-min quando todos os fluxos têm direitos iguais.

i) O fluxo 1 tem direito a  $d = 128/4 = 32$  Mbps.

Como o fluxo 1 gera menos que 32 Mbps, o fluxo 1 fica com 8 Mbps. Sobram  $32 - 8 = 24$  Mbps.

ii) O fluxo 2 tem direito a  $d = 32 + 24/3 = 40$  Mbps.

Como o fluxo 2 gera menos que 40 Mbps, o fluxo 2 fica com 36 Mbps. Sobram  $40 - 36 = 4$  Mbps.

ii) O fluxo 3 tem direito a  $d = 40 + 4/2 = 42$  Mb/s.

Como o fluxo 3 (e o fluxo 4) geram mais de 42 Mbps, os fluxos 3 e 4 ficam com 42 Mbps.

## Equidade max-min com direitos diferentes

São atribuídos pesos aos fluxos proporcionais aos seus direitos. A atribuição de recursos é feita de acordo com o princípio *weighted max-min* fair.

Neste caso:

- Os recursos são atribuídos aos fluxos por ordem crescente de necessidade, estando esta normalizada em relação ao peso.
- A nenhum fluxo é atribuído uma quantidade de recursos maior do que a sua necessidade.
- A fluxos cuja necessidade não tenha sido satisfeita é atribuída uma quantidade de recursos proporcional ao seu peso.



## Exemplo 2

Considere uma ligação com capacidade de 128 Mbps e 4 fluxos de tráfego de 8, 36, 48 e 128 Mbps. Determine que recursos são atribuídos a cada fluxo quando os fluxos têm pesos 1, 1, 3 e 3, respectivamente.

i) Fluxo 1 :  $1/(1+1+3+3) \times 128 = 16$  Mbps

Fluxo 2 :  $1/(1+1+3+3) \times 128 = 16$  Mbps

Fluxo 3 :  $3/(1+1+3+3) \times 128 = 48$  Mbps

Fluxo 4 :  $3/(1+1+3+3) \times 128 = 48$  Mbps

Atribui-se 8 Mbps ao fluxo 1 (<16 Mbps) e 48 Mbps ao fluxo 3.

Sobram  $(16 - 8) + (48 - 48) = 8$  Mbps.

ii) Fluxo 2 :  $16 + 1/(1+3) \times 8 = 18$  Mbps

Fluxo 4 :  $48 + 3/(1+3) \times 8 = 54$  Mbps

Atribui-se 18 Mbps ao fluxo 2 (<36 Mbps) e 54 Mbps ao fluxo 4 (<128 Mbps).

## Comparação dos Exemplos 1 e 2

Capacidade da ligação: 128 Mbps

Fluxos:	1	2	3	4
Débito de transmissão (Mbps):	8	36	48	128

---

Pesos:	1	1	1	1
Atribuição (Mbps):	8	36	42	42

---

Pesos:	1	1	3	3
Atribuição (Mbps):	8	18	48	54

---

- Quando os pesos são todos iguais, os fluxos 1 e 2 conseguem todo o seu débito porque são os fluxos com menor débito de transmissão
- Quando os fluxos 3 e 4 têm 3 vezes maior peso que os fluxos 1 e 2, conseguem maior débito enquanto que o fluxo 2 já não tem todo o seu débito de transmissão.

# Proteção nas disciplinas de escalonamento

Idealmente, a função de escalonamento deve procurar proteger os fluxos bem comportados dos fluxos mal comportados.

Um fluxo mal comportado é um fluxo que envia tráfego a uma taxa superior à taxa a que tem direito (de acordo com o princípio de atribuição de recursos em vigor).

Como veremos à frente:

- as disciplina de escalonamento do tipo FIFO ou com prioridades não protegem os fluxos bem comportados dos fluxos mal comportados;
- por exemplo, as disciplinas de escalonamento do tipo *round-robin* conseguem.

# Disciplinas de escalonamento

As disciplinas de escalonamento podem classificar-se em disciplinas com e sem conservação de trabalho (*work conserving*):

- numa disciplina com conservação de trabalho, a ligação só está inativa (i.e., não está a ser usada para transmitir pacotes) se não houver qualquer pacote à espera de ser transmitido;
- numa disciplina sem conservação de trabalho, a ligação pode estar inativa mesmo que haja pacotes na fila de espera.

Todas as disciplinas de escalonamento que iremos abordar são disciplinas com conservação de trabalho e são as seguintes:

- (1) por ordem de chegada (FIFO),
- (2) com base em prioridade estrita,
- (3) de uma forma rotativa (RR, WRR, DRR),
- (4) por aproximação ao sistema GPS (WFQ, SCFQ).

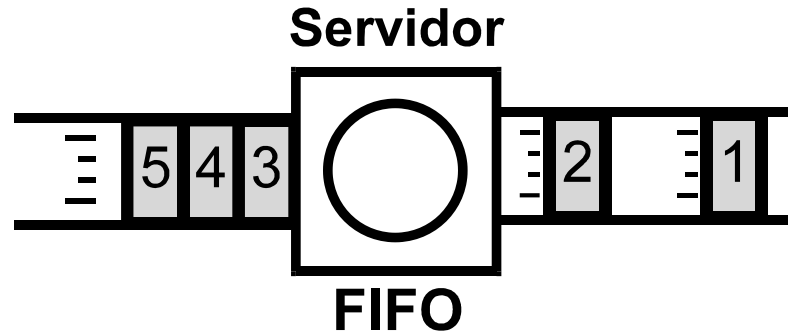


# **Mecanismos de Escalonamento e Descarte de Pacotes em Redes com Comutação de Pacotes**

**Segunda parte:**

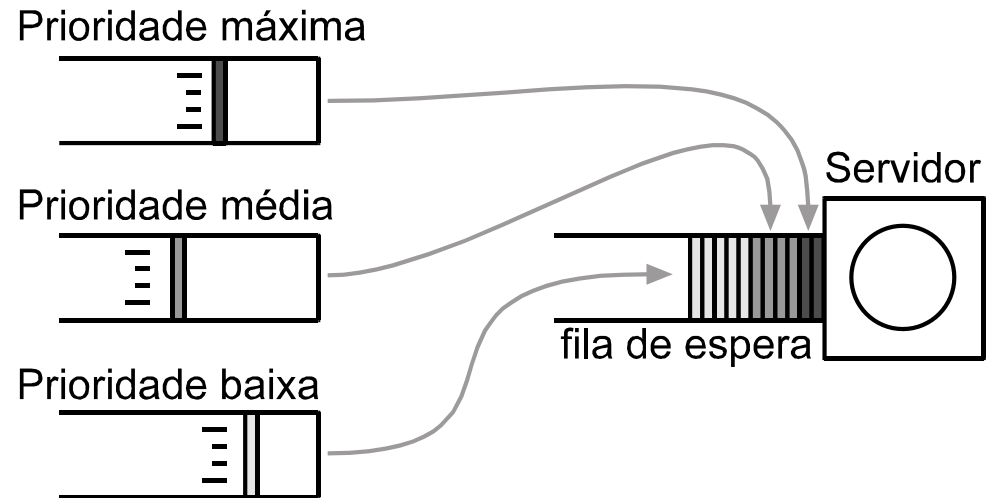
- **Disciplinas de escalonamento de pacotes: FIFO, com prioridades e que funcionam de forma rotativa**

# First-In-First-Out (FIFO)



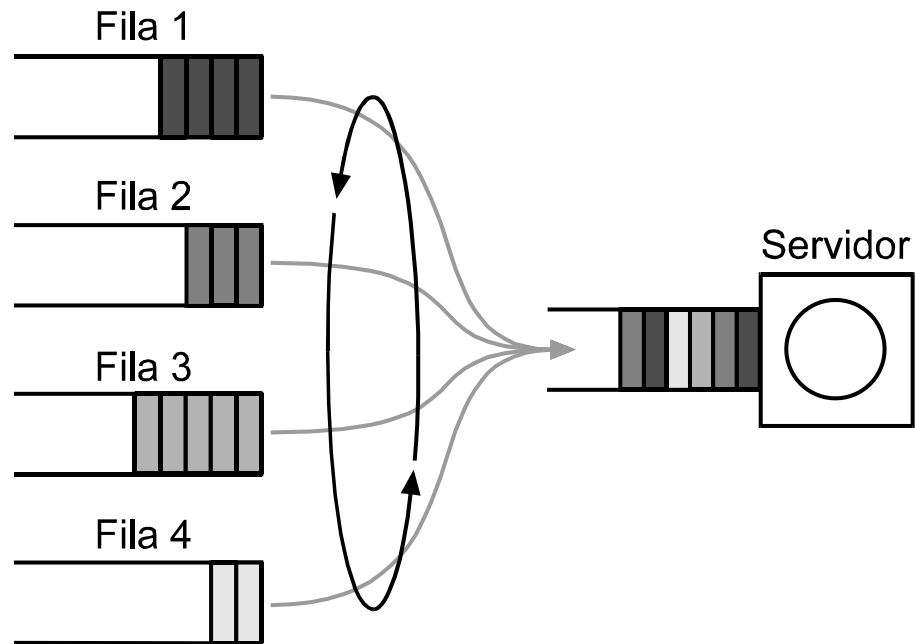
- Os pacotes de todos os fluxos são transmitidos pela sua ordem de chegada.
- Não envolve processamento de ordenação nem de classificação de pacotes.
- Não permite diferenciação de qualidade de serviço (o atraso médio na fila de espera é igual para os pacotes de todos os fluxos).
- Quando a fila de espera não está vazia, fluxos com  $n$  vezes mais tráfego recebem  $n$  vezes mais taxa de serviço pelo que os fluxos bem comportados não são protegidos.

## Prioridade Estrita



- Os pacotes classificados como de maior prioridade são sempre transmitidos antes dos pacotes de menor prioridade (os pacotes com a mesma prioridade são transmitidos com a disciplina FIFO).
- Não envolve processamento de ordenação.
- Envolve classificação dos pacotes de acordo com a prioridade.
- Permite diferenciação da qualidade de serviço (o atraso médio na fila de espera é menor para os pacotes de maior prioridade).
- Fluxos de pacotes de maior prioridade podem impedir que os fluxos de menor prioridade recebam qualquer serviço.

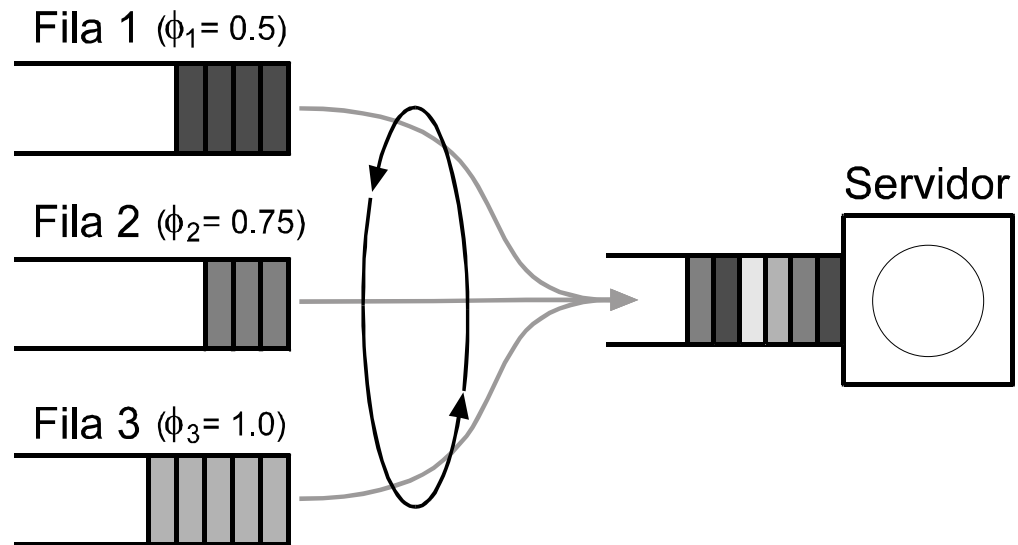
## Round Robin (RR)



- Existe uma fila por fluxo de pacotes e o algoritmo seleciona um pacote de cada fila não vazia de forma rotativa.
- Não permite diferenciação de qualidade de serviço.
- Ao contrário do FIFO, o RR serve o mesmo número de pacotes de todos os fluxos ativos (i.e. fluxos com pacotes na fila).
- Fluxos de pacotes maiores têm maior taxa de serviço.
- Protege os fluxos bem comportados (os fluxos mal comportados apenas penalizam o seu próprio atraso na fila de espera).

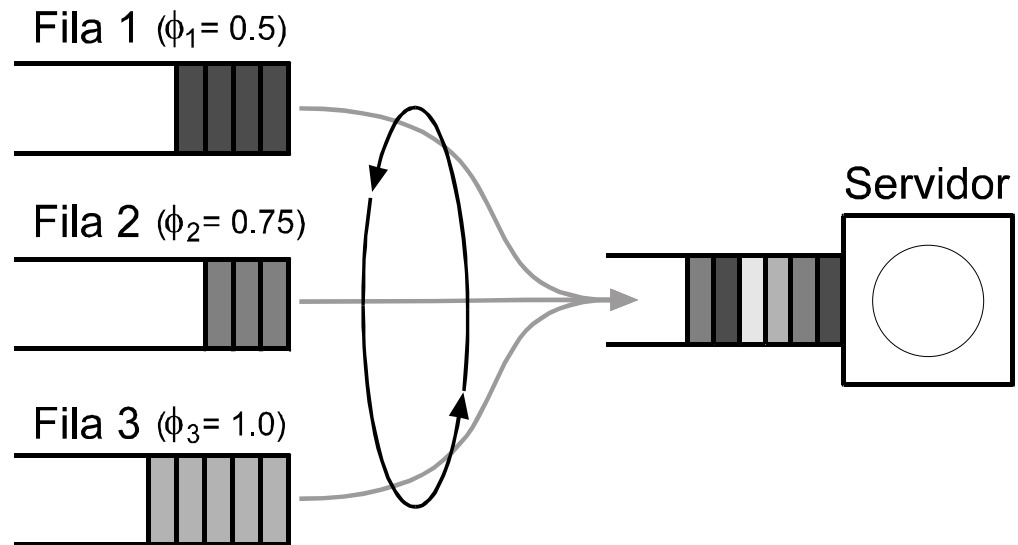


## Weighted Round Robin (WRR)



- É atribuído um peso  $\phi_i$  a cada fila de espera proporcional à taxa de serviço a proporcionar a cada fluxo em situação de congestão.
- Em cada ciclo, o WRR serve um número de pacotes de cada fila de espera tal que a soma dos seus tamanhos (em bytes) é proporcional ao peso da fila.
- É necessário conhecer a priori o comprimento médio dos pacotes.
- A ligação pode ficar demasiado tempo a servir cada fluxo de pacotes o que tem um impacto negativo no *jitter* introduzido pela ligação.

## Weighted Round Robin (WRR)



No exemplo da figura, se o comprimento médio (em Bytes) dos pacotes de cada fluxo for:

$$L_1 = 50, L_2 = 500, L_3 = 1500$$

Os pesos normalizados são:

$$\varphi_1 = 0.5/50 = 1/100 = 60/6000$$

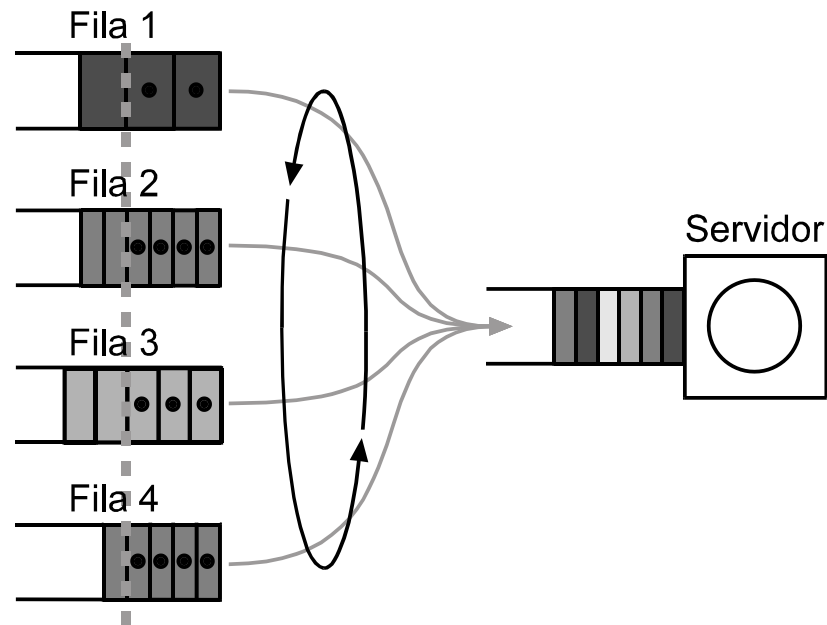
$$\varphi_2 = 0.75/500 = 3/2000 = 9/6000$$

$$\varphi_3 = 1/1500 = 4/6000$$

Número de pacotes por ciclo:

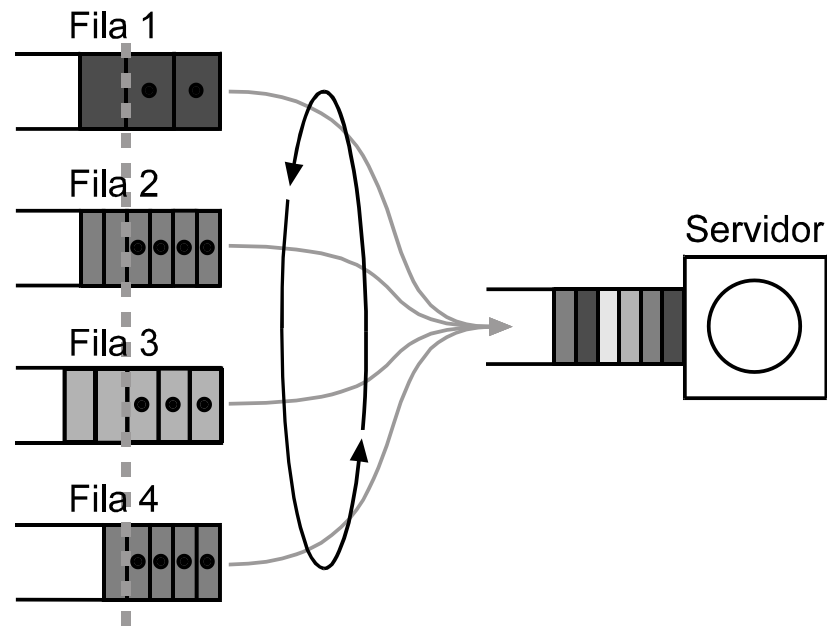
$$\Phi_1 = 60, \Phi_2 = 9, \Phi_3 = 4$$

## Deficit Round Robin (DRR)

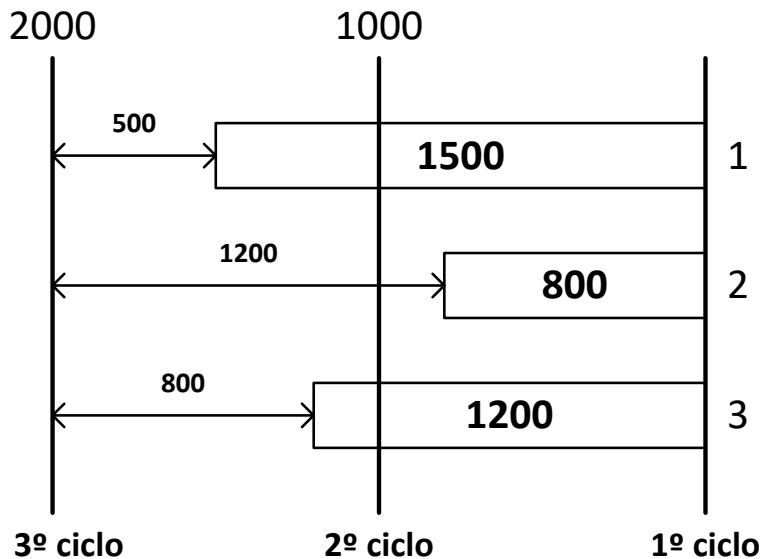


- Em cada ciclo, o DRR serve uma quantidade de bytes até um valor máximo designado por limiar.
- A diferença entre a quantidade servida e o limiar é contabilizada em forma de crédito para o ciclo seguinte.
- Quando uma fila está vazia, o crédito respetivo é colocado a zero.
- Se se considerarem limiares diferentes para as diferentes filas, a taxa de serviço de cada fluxo é proporcional ao limiar da sua fila de espera.
- Ao contrário do WRR, não é necessário saber o comprimento médio dos pacotes.

# Deficit Round Robin (DRR)



**limiar = 1000 bytes (para todos os fluxos)**



1º Ciclo:

- a) fila 1 não é servida, obtém crédito de 1000
- b) fila 2 é servida, obtém crédito de 200
- c) fila 3 não é servida, obtém crédito de 1000

2º Ciclo:

- a) fila 1 é servida, obtém crédito de 500
- b) fila 2 está vazia, fica com crédito a 0
- c) fila 3 é servida, obtém crédito de 800



# **Mecanismos de Escalonamento e Descarte de Pacotes em Redes com Comutação de Pacotes**

**Terceira parte:**

- **Disciplinas de escalonamento de pacotes que funcionam por aproximação ao sistema GPS**

# Generalized Processor Sharing (GPS)

modelo de fluídos



modelo de pacotes



- Algoritmo ideal que proporciona equidade perfeita, baseado num modelo de fluídos, em que o tráfego é considerado infinitamente divisível.
  - Exemplo: num dado instante, 50% da capacidade de uma ligação é utilizada por um fluxo e 50% por outro fluxo.
- Existe uma fila de espera por fluxo e é atribuído um peso  $\phi_i$  a cada fluxo.
- Quando um pacote chega a uma fila, se nenhum outro pacote da mesma fila estiver a ser transmitido, este começa imediatamente a ser transmitido, em paralelo com os pacotes das outras filas, a uma taxa de serviço proporcional ao seu peso.
- É um algoritmo impossível de realizar na prática, mas constitui uma boa base teórica para o desenvolvimento de outros algoritmos.

## Exemplo 3

Considere-se uma ligação de 64 Kbps com 2 filas de espera de pesos  $\phi_1 = 3$  e  $\phi_2 = 1$ , em que os 2 fluxos de pacotes são servidos pela disciplina de escalonamento ideal GPS. Chegam a esta ligação os seguintes pacotes:

- pacote 1 à fila 1 com 62 Bytes em  $t = 0$ ,
- pacote 1 à fila 2 com 32 Bytes em  $t = 4$  ms e
- pacote 2 à fila 1 com 18 Bytes em  $t = 6$  ms.

Determinar os instantes em que os pacotes são servidos (i.e., os instantes de tempo em que termina a transmissão de cada pacote).

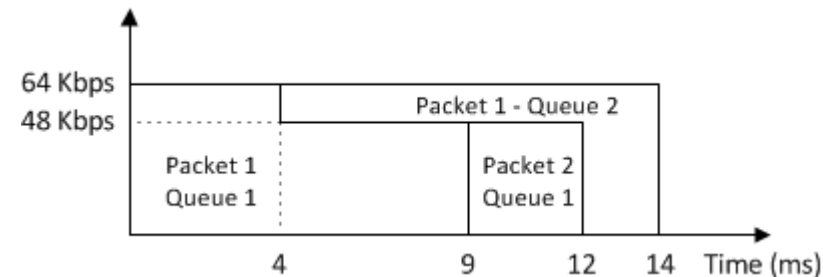
Ligação: 64 Kbps

2 filas de espera:  $\phi_1 = 3$  e  $\phi_2 = 1$

Chegam:

- pacote 1 à fila 1 com 62 Bytes ( $t = 0$ ),
- pacote 1 à fila 2 com 32 Bytes ( $t = 4$  ms) e
- pacote 2 à fila 1 com 18 Bytes ( $t = 6$  ms).

## Resolução do Exemplo 3



- O pacote 1 da fila 1 é servido inicialmente a 64 Kb/s. Em  $t = 4$  ms, foram servidos  $(64\text{Kb/s}) \times (4\text{ms}) = 256$  bits = 32 Bytes do pacote 1 da fila 1. A partir daqui, a fila 1 é servida a  $(3/4) \times 64$  Kb/s = 48 Kb/s e a fila 2 a  $(1/4) \times 64$  Kb/s = 16 Kb/s.
- Com estas taxas, o pacote 1 da fila 1 demora  $((62-32) \times 8) / (48\text{Kb/s}) = 5$  ms a finalizar a sua transmissão e o pacote 1 da fila 2 demora  $(32 \times 8) / (16\text{Kb/s}) = 16$  ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em  $t = 4 + 5 = 9$  ms. Neste instante, inicia-se a transmissão do pacote 2 da fila 1 porque chegou no instante  $t = 6$  ms.
- O pacote 2 da fila 1 demora  $(18 \times 8) / (48\text{Kb/s}) = 3$  ms a ser transmitido. Assim, o pacote 2 da fila 1 termina a sua transmissão em  $t = 9 + 3 = 12$  ms.
- A partir de  $t = 12$  ms, o pacote 1 da fila 2 é transmitido a 64 Kb/s. Como até este instante foram transmitidos  $(16\text{Kb/s}) \times (8\text{ms}) = 128$  bits = 16 Bytes, os restantes 16 Bytes demoram  $(16 \times 8) / (64\text{Kb/s}) = 2$  ms. Assim, o pacote 1 da fila 2 termina a transmissão em  $t = 12 + 2 = 14$  ms.



# Weighted Fair Queuing (WFQ)

É uma aproximação ao sistema GPS: o WFQ tenta servir os pacotes pela ordem em que terminariam de ser transmitidos no sistema GPS.

Sempre que chega um pacote a uma fila, é atribuído ao pacote um **Finish Number** ( $FN$ ) que indica a ordem pela qual ele será enviado relativamente aos outros pacotes.

**Round Number** ( $RN$ ) é uma variável real que cresce no tempo a uma taxa inversamente proporcional aos pesos dos fluxos ativos.

Num intervalo de tempo  $[\tau_i, \tau_{i+1})$  em que o número de fluxos ativos se mantenha constante:

$$RN(\tau_i + t) = RN(\tau_i) + \frac{1}{\sum_{j \text{ ativos}} \phi_j} t \quad t \in [\tau_i, \tau_{i+1})$$

O  $RN$  é processado sempre que o número de fluxos ativos se altera:

- quando um pacote chega de um fluxo que não tem pacotes no sistema;
- quando um pacote de um fluxo termina de ser transmitido e o fluxo não tem nenhum outro pacote na fila de espera.

Quando o pacote  $k$  com comprimento  $L_k$  pertencente à fila  $i$  chega, é-lhe atribuído o *finish number*  $FN_{i,k}$  dado por:

$$FN_{i,k} = \max(FN_{i,k-1}, RN) + \frac{L_k/C}{\phi_i}$$

## Self Clock Fair Queuing (SCFQ)

A principal desvantagem do WFQ é o peso computacional do cálculo do  $RN$ .

Por forma a evitar o cálculo do  $RN$  do WFQ, o SCFQ substitui este parâmetro pelo valor do  $FN$  do pacote que está a ser transmitido,  $FN_s$ , qualquer que seja o fluxo a que pertence.

Assim, quando o pacote  $k$  com comprimento  $L_k$  pertencente à fila  $i$  chega, é-lhe atribuído o *finish number*  $FN_{i,k}$  dado por:

$$FN_{i,k} = \max(FN_{i,k-1}, FN_s) + \frac{L_k}{\phi_i}$$

Não se utiliza o valor da capacidade da ligação ( $C$ ), uma vez que não é necessário saber o tempo que o pacote demoraria a ser servido no sistema GPS.

Apesar do SCFQ ser de muito menor complexidade que o WFQ, pode não ser tão justo para pequenos intervalos de tempo (i.e., não se aproxima tão bem ao GPS como o WFQ).

## Exemplo 4

Considere-se uma ligação de 64 Kbps com 2 filas de espera de pesos  $\phi_1 = 3$  e  $\phi_2 = 1$ . Chegam a esta ligação os seguintes pacotes:

- pacote 1 à fila 1 com 62 Bytes em  $t = 0$ ,
- pacote 1 à fila 2 com 32 Bytes em  $t = 4$  ms e
- pacote 2 à fila 1 com 18 Bytes em  $t = 6$  ms.

Determinar os instantes em que os pacotes são servidos (i.e., os instantes de tempo em que termina a transmissão de cada pacote) considerando que os 2 fluxos de pacotes são servidos por uma:

- (a) uma disciplina de escalonamento WFQ
- (b) uma disciplina de escalonamento SCFQ

## Exemplo 4 – resolução de (a)

Ligação: 64 Kbps

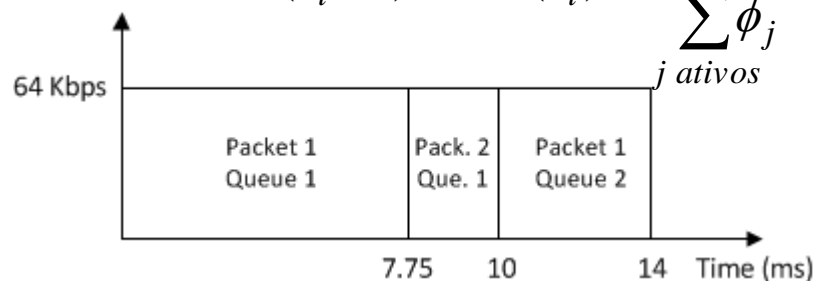
2 filas de espera:  $\phi_1 = 3$  e  $\phi_2 = 1$

Chegam:

- pacote 1 à fila 1 com 62 Bytes ( $t = 0$ ),
- pacote 1 à fila 2 com 32 Bytes ( $t = 4$  ms) e
- pacote 2 à fila 1 com 18 Bytes ( $t = 6$  ms).

$$FN_{i,k} = \max(FN_{i,k-1}, RN) + \frac{L_k/C}{\phi_i}$$

$$RN(\tau_i + t) = RN(\tau_i) + \frac{1}{\sum_{j \text{ ativos}} \phi_j} t$$



- Em  $t = 0$  ms,  $RN = 0$  e  $FN_{1,1} = 0 + (62 \times 8) / 64000 / 3 = 2.58 \times 10^{-3}$ . O pacote 1 da fila 1 é transmitido em  $(62 \times 8) / (64 \text{ Kb/s}) = 7.75$  ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em  $t = 0 + 7.75 = 7.75$  ms.
- Em  $t = 4$  ms :  $RN = 0 + (4 \times 10^{-3}) / 3 = 1.33 \times 10^{-3}$   
 $FN_{2,1} = 1.33 \times 10^{-3} + (32 \times 8) / 64000 / 1 = 5.33 \times 10^{-3}$
- Em  $t = 6$  ms :  $RN = 1.33 \times 10^{-3} + (6 \times 10^{-3} - 4 \times 10^{-3}) / 4 = 3.33 \times 10^{-3}$   
 $FN_{1,2} = \max(2.58 \times 10^{-3}, 3.33 \times 10^{-3}) + (18 \times 8) / 64000 / 3 = 4.08 \times 10^{-3}$
- Em  $t = 7.75$  ms, como  $FN_{1,2} < FN_{2,1}$ , o pacote 2 da fila 1 começa a ser transmitido. O pacote 2 da fila 1 é transmitido em  $(18 \times 8) / (64 \text{ Kb/s}) = 2.25$  ms. Assim, o pacote 2 da fila 1 termina a sua transmissão em  $t = 7.75 + 2.25 = 10$  ms.
- Em  $t = 10$  ms, o pacote 1 da fila 2 começa a ser transmitido. O pacote 1 da fila 2 é transmitido em  $(32 \times 8) / (64 \text{ Kb/s}) = 4$  ms. Assim, o pacote 1 da fila 2 termina a sua transmissão em  $t = 10 + 4 = 14$  ms.

## Exemplo 4 – resolução de (b)

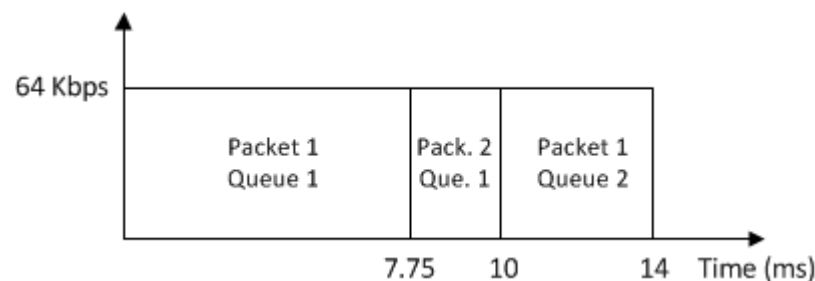
$$FN_{i,k} = \max(FN_{i,k-1}, FN_s) + \frac{L_k}{\phi_i}$$

Ligação: 64 Kbps

2 filas de espera:  $\phi_1 = 3$  e  $\phi_2 = 1$

Chegam:

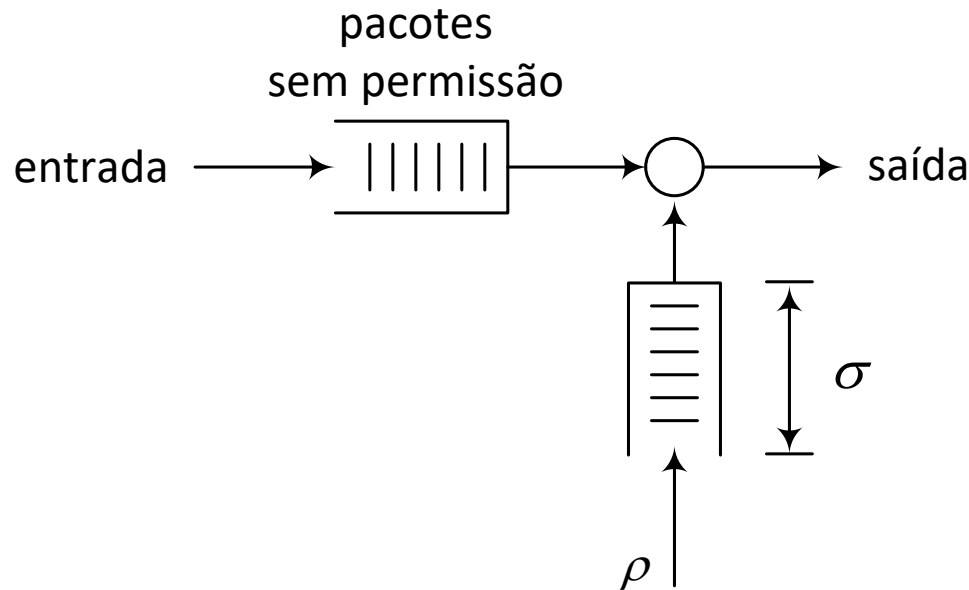
- pacote 1 à fila 1 com 62 Bytes ( $t = 0$ ),
- pacote 1 à fila 2 com 32 Bytes ( $t = 4$  ms) e
- pacote 2 à fila 1 com 18 Bytes ( $t = 6$  ms).



- Em  $t = 0$  ms,  $FN_{1,1} = 0 + (62 \times 8)/3 = 165.3$ . O pacote 1 da fila 1 é transmitido em  $(62 \times 8)/(64 \text{ Kb/s}) = 7.75$  ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em  $t = 0 + 7.75 = 7.75$  ms.
- Em  $t = 4$  ms :  $FN_{2,1} = 165.3 + (32 \times 8)/1 = 421.3$
- Em  $t = 6$  ms :  $FN_{1,2} = \max(165.3, 165.3) + (18 \times 8)/3 = 213.3$
- Em  $t = 7.75$  ms, como  $FN_{1,2} < FN_{2,1}$ , o pacote 2 da fila 1 começa a ser transmitido. O pacote 2 da fila 1 é transmitido em  $(18 \times 8)/(64 \text{ Kb/s}) = 2.25$  ms. Assim, o pacote 2 da fila 1 termina a sua transmissão em  $t = 7.75 + 2.25 = 10$  ms.
- Em  $t = 10$  ms, o pacote 1 da fila 2 começa a ser transmitido. O pacote 1 da fila 2 é transmitido em  $(32 \times 8)/(64 \text{ Kb/s}) = 4$  ms. Assim, o pacote 1 da fila 2 termina a sua transmissão em  $t = 10 + 4 = 14$  ms.

# Desempenho do GPS com controlo de taxa de transmissão por *leaky bucket*

O *Leaky Bucket* é um mecanismo de controlo de taxas de transmissão que permite impor um majorante ao tráfego gerado por um dado fluxo.



Se  $A_i(\tau, t)$  representar a quantidade de tráfego (em Bytes) do fluxo  $i$  que é submetido à rede no intervalo de tempo  $[\tau, t]$ , então:

$$A_i(\tau, t) \leq \sigma_i + \rho_i(t - \tau)$$

# Desempenho do GPS com controlo de taxa de transmissão por *leaky bucket*

Numa disciplina GPS, se designarmos por  $S_i(\tau, t)$  o tráfego (em Bytes) de um fluxo  $i$  que é servido num intervalo de tempo  $[\tau, t)$ , então:

$$S_i(\tau, t) \geq r_i(t - \tau) \quad \text{em que} \quad r_i = \frac{\phi_i}{\sum_j \phi_j} C$$

A quantidade máxima de tráfego em espera  $Q_{i,\max}(t)$  do fluxo  $i$ , desde um instante em que o fluxo não tinha tráfego no sistema ( $\tau = 0$ ) até um qualquer instante  $t$  é:

$$\begin{aligned} Q_{i,\max}(t) &= A_{i,\max}(0, t) - S_{i,\min}(0, t) \\ &= \sigma_i + \rho_i t - r_i t \\ &\leq \sigma_i \quad \Leftarrow \quad r_i \geq \rho_i \end{aligned}$$

O atraso máximo  $D_i$  é o tempo necessário para transmitir todo o tráfego em espera, que na pior das hipóteses é servido à taxa mínima de serviço  $r_i$ . Assim, se  $r_i \geq \rho_i$ , o atraso máximo de qualquer pacote do fluxo  $i$  é:

$$D_i = \frac{\sigma_i}{r_i}$$

# Desempenho do WFQ com controlo de taxa de transmissão por *leaky bucket*

Numa disciplina WFQ, o atraso máximo é maior que no GPS porque a informação é transmitida em pacotes.

Considere um fluxo  $i$  formatado por um *leaky bucket* com parâmetros  $\sigma_i$  e  $\rho_i$  que atravessa  $n$  ligações:

$C_j$  - capacidade da ligação  $j$

$r_i$  - largura de banda reservada para o fluxo  $i$  em todas as ligações ( $r_i \geq \rho_i$ )

$L_i$  - tamanho máximo dos pacotes do fluxo  $i$

$L_{\max}$  - tamanho máximo dos pacotes de todos os fluxos

Prova-se que o atraso máximo ( $D_i$ ) que os pacotes do fluxo  $i$  sofrem é:

$$D_i = \frac{\sigma_i + (n-1)L_i}{r_i} + \sum_{j=1}^n \frac{L_{\max}}{C_j} + \Gamma$$

em que  $\Gamma$  é o atraso total de propagação de todas as ligações.



## Exemplo 5

Considere um fluxo de pacotes de comprimento máximo de 200 Bytes formatado por um *leaky bucket* com parâmetros  $\sigma = 1000$  bytes e  $\rho = 150$  Kbps. O fluxo atravessa 8 ligações todas com capacidade 100 Mbps servidas por uma disciplina WFQ. O comprimento máximo dos pacotes de todos os fluxos é de 1500 bytes. O atraso de propagação total é 2 mseg. Qual a taxa (em Mbps) que é necessário reservar em todas as ligações para este fluxo, por forma a garantir um atraso máximo extremo-a-extremo de 20 mseg?

$$D_i = \frac{\sigma_i + (n-1)L_i}{r_i} + \sum_{j=1}^n \frac{L_{\max}}{C_j} + \Gamma$$

$$0.02 = \frac{1000 \times 8 + 7 \times 200 \times 8}{r} + 8 \times \frac{1500 \times 8}{100 \times 10^6} + 0.002$$

$$r = \frac{1000 \times 8 + 7 \times 200 \times 8}{0.018 - 8 \times \frac{1500 \times 8}{100 \times 10^6}} = 1127 \text{ Kbps} = 1.127 \text{ Mbps}$$



# **Mecanismos de Escalonamento e Descarte de Pacotes em Redes com Comutação de Pacotes**

**Quarta parte:**

- **Métodos de descarte de pacotes**
- **Ilustração da combinação de disciplinas de escalonamento com métodos de descarte de pacote na arquitectura DiffServ do IETF**

## **Métodos de Descarte de Pacotes**

Os métodos de descarte de pacotes podem ser classificados quanto a:

- Posição de descarte
- Prioridade de descarte
- Grau de agregação
- Descarte antecipado

# Métodos de Descarte de Pacotes

## *Posição de descarte*

- Cauda da fila – Normalmente usado por omissão; mais simples de implementar (o pacote não chega a entrar na fila).
  - Em muitos casos, a fila tem muitos pacotes pertencentes a poucos fluxos. Se o pacote que chega não pertence a nenhum desses fluxos, a estratégia não é justa.
- Posição aleatória – Escolhe-se aleatoriamente um pacote (entre todos os da fila + o novo) para ser eliminado (computacionalmente pesado).
  - Os fluxos com mais pacotes na fila são mais penalizados: estratégia mais justa.
- Cabeça da fila – Retira-se o pacote mais antigo da fila e aceita-se o que chegou (computacionalmente leve).
  - Tão bom como a posição aleatória em termos de justiça.
  - Útil quando o controle de fluxo é baseado em perdas de pacotes (porquê? lembrar controle de congestão do TCP!) 36

# Métodos de Descarte de Pacotes

## ***Prioridades de descarte***

- O emissor ou a rede (o policiador de um domínio DiffServ) podem marcar alguns pacotes com maior prioridade de descarte. Estes, em situação de congestionamento serão os primeiros a ser descartados.
- Quando um pacote é fragmentado e um dos fragmentos é descartado, os restantes fragmentos podem (e devem) também ser descartados pois deixam de ter qualquer utilidade.
  - Podia ser usado no protocolo IP? Relembrar utilização da flag '*more fragments*' e do campo *Fragment Offset*.
- Um método de descarte possível consiste em dar maior prioridade de descarte aos pacotes que passaram por menos ligações (*i.e.*, usaram menos recursos).
  - Este método não pode ser implementado no protocolo IP (porquê? relembrar utilização do campo TTL no IPv4)

# Métodos de Descarte de Pacotes

## ***Grau de agregação***

### Agregação de fluxos

- O método de descarte pode considerar os fluxos individualmente ou de forma agregada.
  - Na forma agregada, o método é aplicado a cada pacote do agregado, sem tomar em consideração o fluxo a que pertence.
  - Quanto mais fluxos forem agregados, menor a proteção entre os fluxos pertencentes ao mesmo agregado.

### Agregação da memória dedicada às filas de espera

- Se existe uma fila de espera por fluxo de pacotes e a memória é partilhada por todas as filas, consegue-se uma atribuição de memória *max-min fair* quando se descarta o último pacote da fila mais longa (*i.e.*, da fila com um maior número de pacotes).
  - Com o WFQ, isto corresponde a descartar o pacote com maior *Finish Number* de entre todos os fluxos.

# Métodos de Descarte de Pacotes

## ***Descarte antecipado***

Descarte quando a fila de espera está cheia:

- Quando a fila enche por um longo período (a ligação está muito congestionada), múltiplos pacotes são descartados provocando a reação simultânea de todas as ligações TCP afetadas; o tráfego tende a variar ciclicamente entre períodos de baixo débito e períodos de congestão.

Descarte antecipado (RED - *Random Early Discard*):

- Quando cada pacote chega à fila, ele é descartado com uma probabilidade proporcional à ocupação da fila; evita-se o sincronismo do controle de congestão das ligações TCP.
- Não proporciona diferenciação de qualidade de serviço.

Descarte antecipado pesado (WRED – *Weighted RED*):

- Atribuem-se diferentes probabilidades de descarte a pacotes pertencentes a diferentes fluxos (ou agregados de fluxos).
- Quanto menor a probabilidade de descarte, menor é a taxa de perda de pacotes que o fluxo (ou o agregado) sofre.

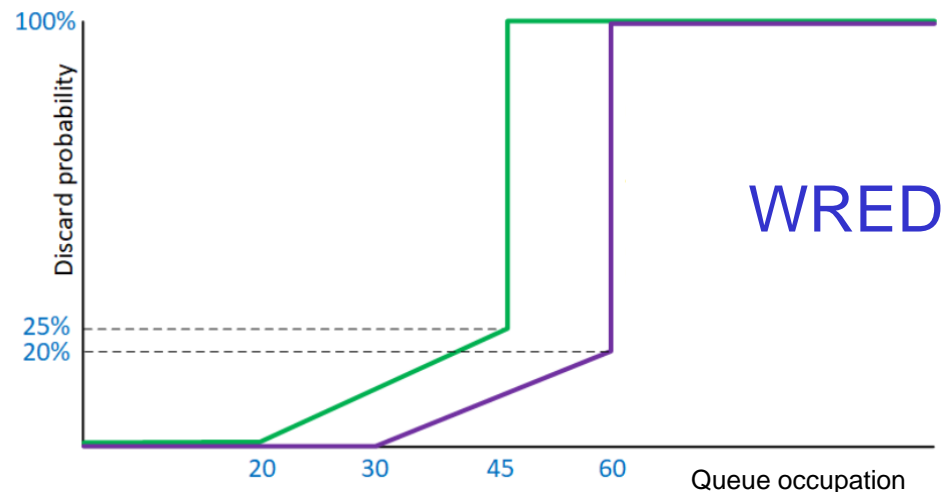
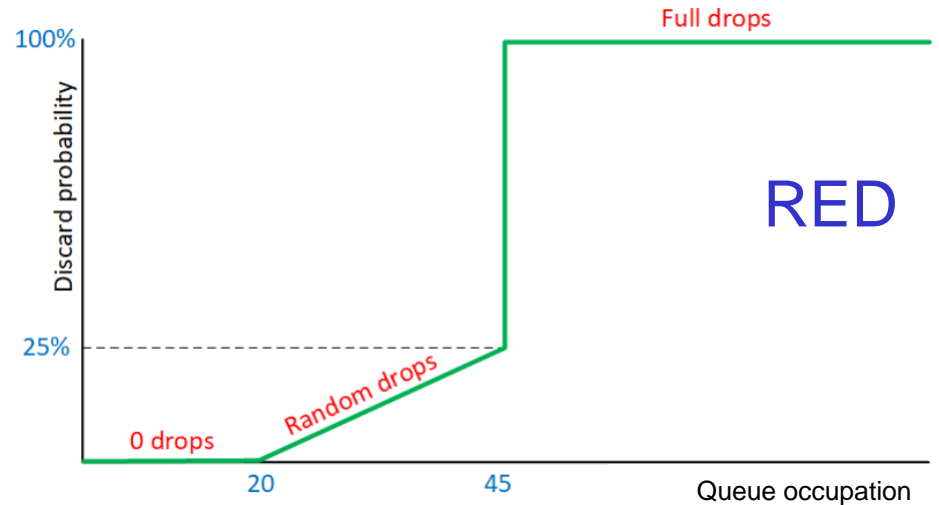
# RED e WRED

## No RED:

- Limite Mínimo ( $m$ ): quando um pacote chega e a ocupação da fila  $f$  é menor que o limite mínimo ( $f < m$ ), o pacote é sempre aceite na fila.
- Limite Máximo ( $M$ ): quando um pacote chega e a ocupação da fila  $f$  é maior que o limite máximo ( $f > M$ ), o pacote é sempre descartado.
- Mark Probability Denominator ( $MPD$ ): quando um pacote chega e a ocupação  $f$  está entre os limites mínimo e máximo ( $m \leq f \leq M$ ), o pacote é descartado com probabilidade  $(f-m)/(M-m) \times MPD$

## No WRED:

- São atribuídos diferentes valores de  $m$ ,  $M$  e  $MPD$  a diferentes fluxos (ou agregados de fluxos)





# Exemplo – Arquitectura *DiffServ*

## ***Classes de Serviço***

- *Default* (DE) → DSCP = 000000
  - serviço *best-effort* com uma única fila de espera do tipo FIFO
- *Expedited Forwarding* (EF) → DSCP = 101110
  - serviço tipo “linha alugada virtual”
  - disponibiliza controle de perdas, do atraso e da variância do atraso dentro de uma determinada largura de banda máxima
- *Assured Forwarding* (AF)
  - fornece uma Qualidade de Serviço relativa entre até 4 classes AF
  - em cada classe AF, pode haver até 3 níveis de precedência para descarte de pacotes (em caso de congestionamento)

<i>AF Codepoints</i>	AF1	AF2	AF3	AF4
<i>Low drop precedence</i>	001010	010010	011010	100010
<i>Medium drop precedence</i>	001100	010100	011100	100100
<i>High drop precedence</i>	001110	010110	011110	100110

# Possível Esquema de Escalonamento do DiffServ

