Escalonamento em Redes com Comutação de Pacotes

Desempenho e Dimensionamento de Redes

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

DETI-UA, 2017/2018

Escalonamento

Considere-se uma rede com comutação de pacotes. Em cada interface de saída de cada ligação:

Disciplina de escalonamento: decide a ordem pela qual os pacotes de diferentes fluxos são enviados

- a disciplina de escalonamento impõe diferentes atrasos 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 descartados quando a fila de espera da ligação está cheia.

- o método de descarte de pacotes impõe diferentes taxas de perda de pacotes a diferentes fluxos ao definir que pacotes são descartados.

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, ..., *n* com necessidades x_1, x_2, ..., x_n e ordenados pelas suas necessidades ($x_1 \leq x_2 \ldots \leq x_n$);
- 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* ao fluxos 1, 2, …, *n*;
  - se não, atribui-se x_1 ao fluxo 1 e o fluxo 2 tem direito a $d = d + (d - x_1)/(n - 1)$
- *d* é menor que x_2?
  - se sim, atribui-se *d* ao fluxos 2, 3, …, *n*;
  - se não, atribui-se x_2 ao fluxo 2 e o fluxo 3 tem direito a $d = d + (d - x_2)/(n - 2)$
- E assim sucessivamente...

Exemplo 1

Considere uma ligação com capacidade de 128 Kbps e 4 fluxos de tráfego de 8, 36, 48 e 128 Kbps. 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 Kbps.

Como o fluxo 1 gera menos que 32 Kb/s, o fluxo 1 fica com 8 Kbps. Sobram 32 – 8 = 24 Kbps.

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

Como o fluxo 2 gera menos que 40 Kbps, o fluxo 2 fica com 36 Kbps. Sobram 40 – 36 = 4 Kbps.

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

Como o fluxo 3 (e os restantes) gera mais de 42 Kbps, os fluxos 3 e 4 ficam com 42 Kbps.

Equidade max-min com pesos diferentes

São atribuídos pesos aos fluxos e 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 Kbps e 4 fluxos de tráfego de 8, 36, 48 e 128 Kbps. Determine que recursos são atribuídos a cada fluxo quando os fluxos têm pesos 1, 1, 3 e 3, respetivamente.

i) Fluxo 1 : $1/8 \times 128 = 16$ Kbps

Fluxo 2 : $1/8 \times 128 = 16$ Kbps

Fluxo 3 : $3/8 \times 128 = 48$ Kbps

Fluxo 4 : $3/8 \times 128 = 48$ Kbps

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

Sobram $(16 – 8) + (48 – 48) = 8$ Kbps.

ii) Fluxo 2 : $16 + 1/4 \times 8 = 18$ Kbps

Fluxo 4 : $48 + 3/4 \times 8 = 54$ Kbps

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

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, uma disciplina de escalonamento do tipo FIFO não consegue proteger os fluxos bem comportados mas um disciplina do tipo *round-robin* consegue.

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 está inativa apenas 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.

As disciplinas de escalonamento que iremos considerar são disciplinas com conservação de trabalho e são as seguintes:

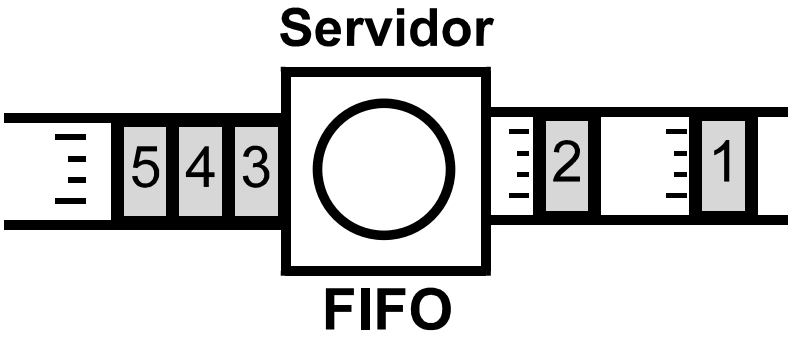
(1) por ordem de chegada (FIFO),

(2) com base em prioridade estrita,

(3) de uma forma rotativa (RR, WRR, DRR),

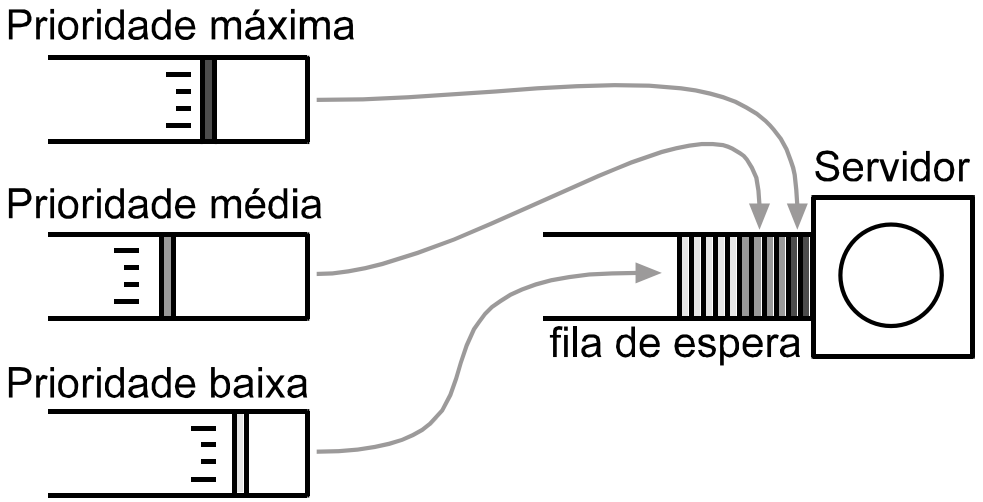
(4) por aproximação ao sistema GPS (WFQ, SCFQ).

First-In-First-Out (FIFO)



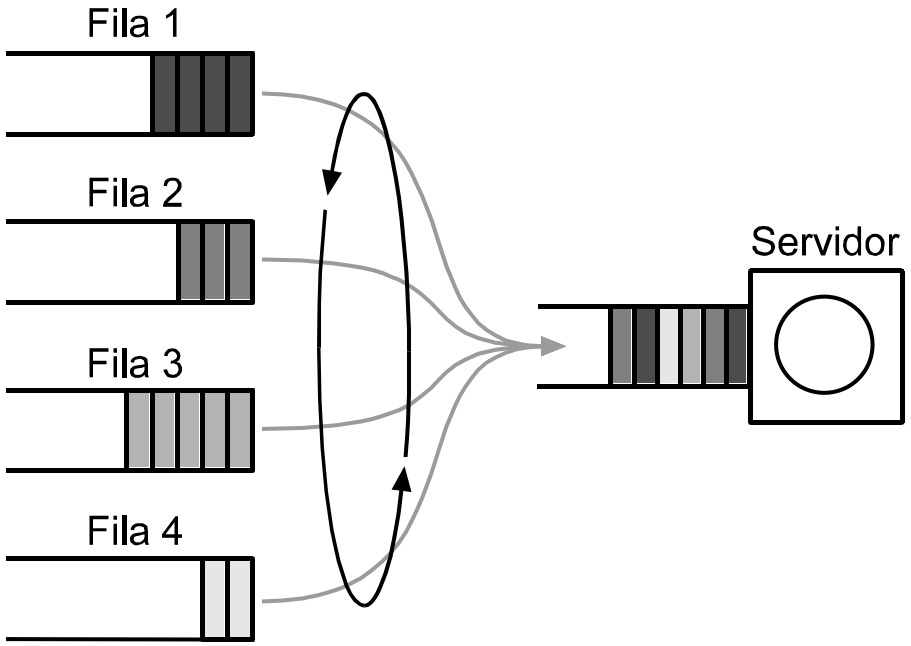
- Os pacotes de todos os fluxos são transmitidos por ordem de chegada.
- Não envolve processamento de ordenação nem de classificação de pacotes.
- Não permite diferenciação de qualidade de serviço (o atraso médio em fila de espera é igual para todos os fluxos).
- Quando a fila de espera não está vazia,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

Prioridade Estrita



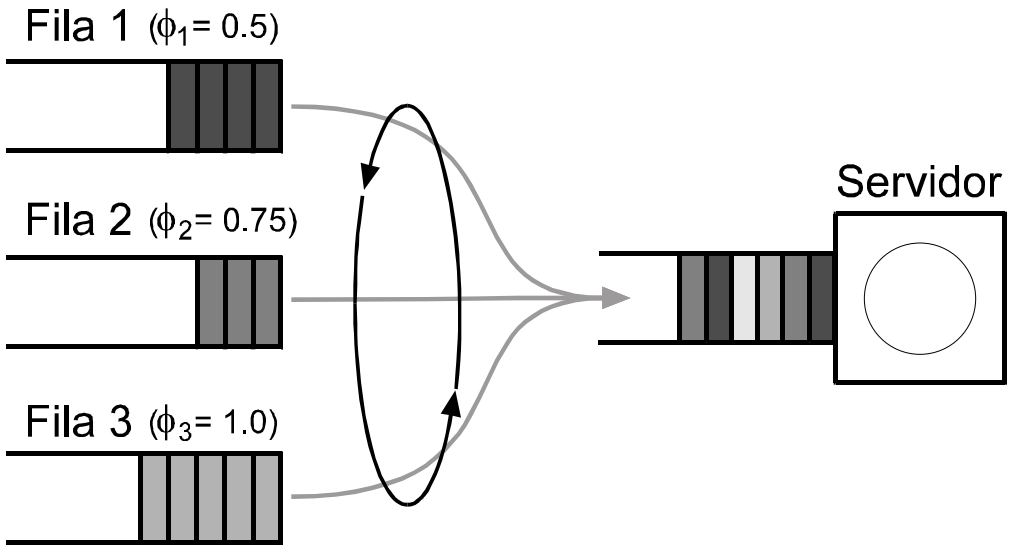
- Os pacotes classificados como de maior prioridade são sempre transmitidos antes dos pacotes de menor prioridade (os pacotes com a mesma prioridade são transmitidos com a disciplina FIFO).
- Não envolve processamento de ordenação.
- Envolve classificação dos pacotes de acordo com a prioridade.
- Permite diferenciação da qualidade de serviço (o atraso médio em 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)



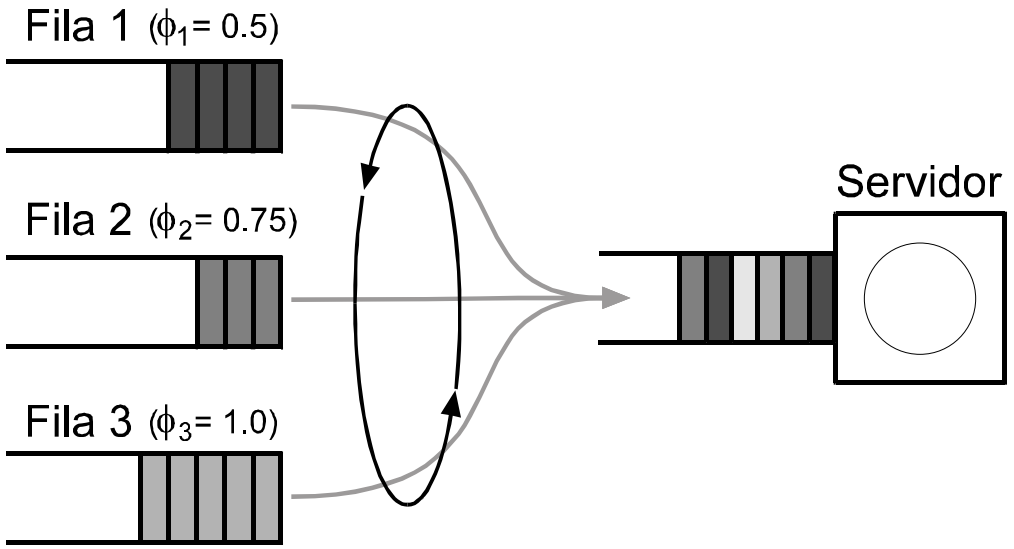
- Existe uma fila por fluxo de pacotes e o algoritmo seleciona um pacote de cada fila (não vazia) de forma rotativa.
- Não permite diferenciação de qualidade de serviço.
- Ao contrário do FIFO, serve o mesmo número de pacotes de todos os fluxos ativos.
- Beneficia os fluxos de pacotes maiores.
- Protege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

Weighted Round Robin (WRR)



- É atribuído um peso ϕ_i a cada fila de espera, peso este proporcional à taxa de serviço a proporcionar a cada fluxo.
- Em cada ciclo, serve um número de pacotes de cada fila 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:

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

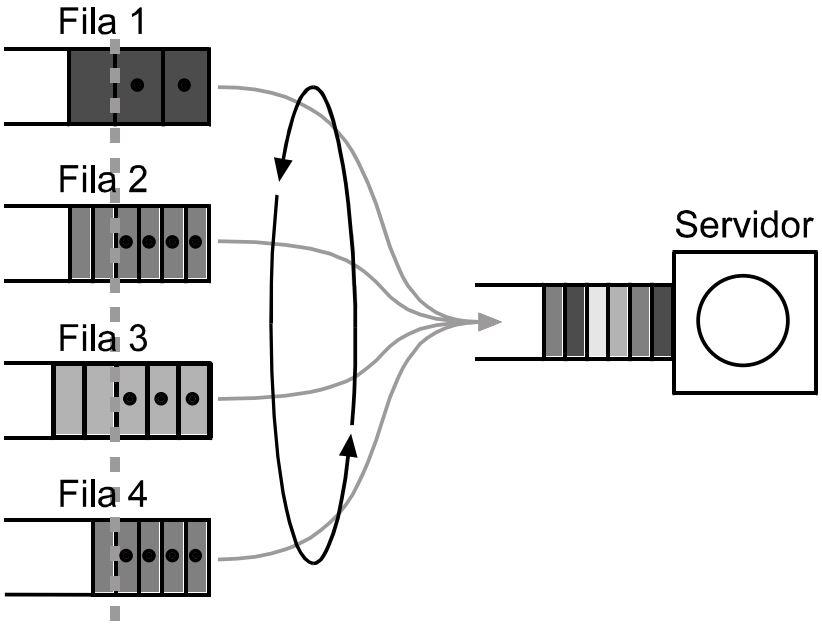
$\varphi_2 = 0.75/500 = 3/2000 = 9/6000$

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

Número de pacotes por ciclo:

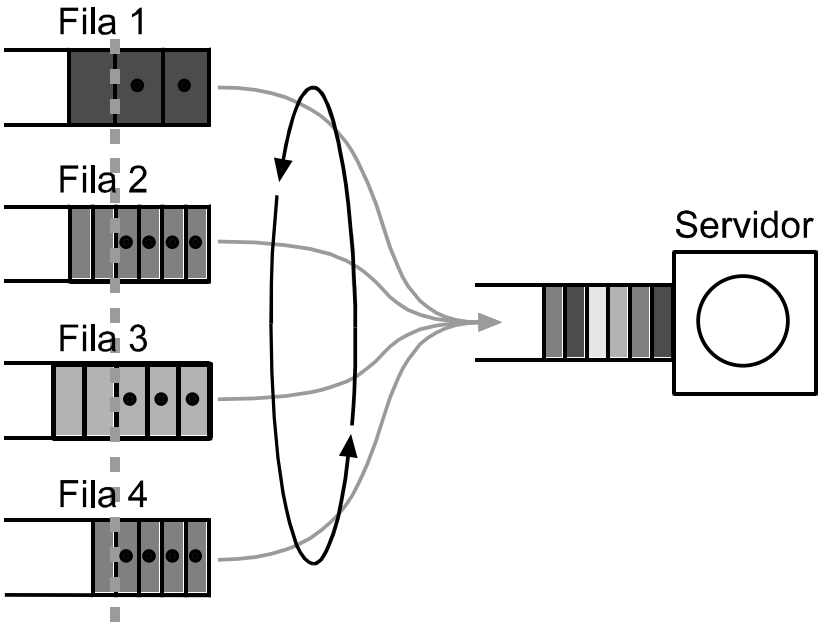
$\Phi_1 = 60,\ \Phi_2 = 9,\ \Phi_3 = 4$

Deficit Round Robin (DRR)

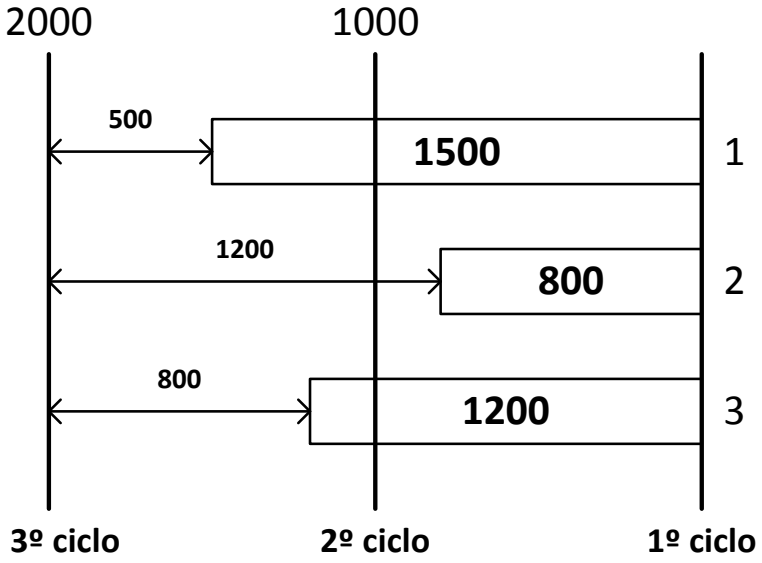


- Em cada ciclo, o algoritmo serve uma quantidade de bytes até um valor máximo designado por *limiar*.
- A diferença entre a quantidade servida e o limiar é contabilizada em forma de *crédito* para o ciclo seguinte.
- Quando uma fila está vazia, o crédito respetivo é colocado a zero.
- Se se considerarem limiares diferentes, 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)



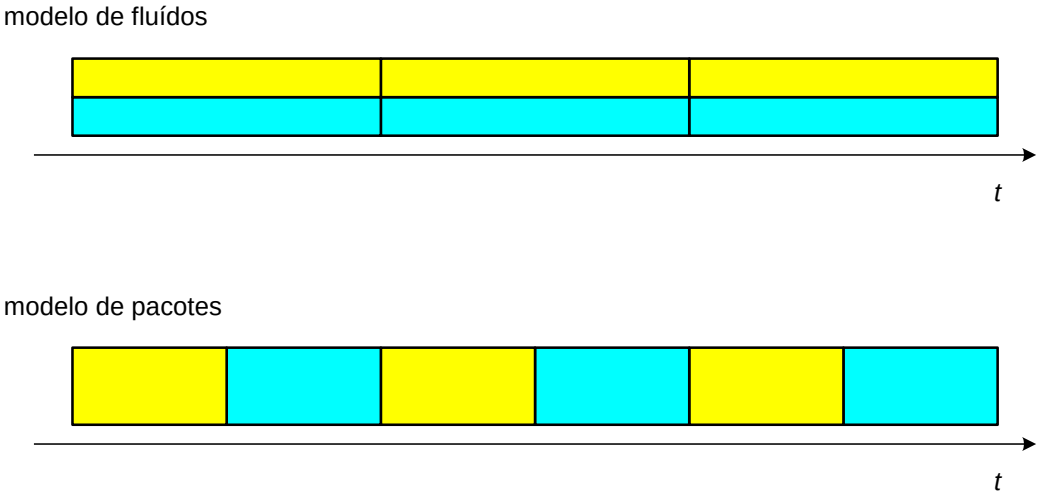
1º Ciclo:

a) 1 não é servido, obtém crédito de 1000
b) 2 é servido, obtém crédito de 200
c) 3 não é servido, obtém crédito de 1000

2º Ciclo:

a) 1 é servido, obtém crédito de 500
b) 2 fica com crédito a 0
c) 3 é servido, obtém crédito de 800

Generalized Processor Sharing (GPS)



- 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 largura de banda de uma ligação é utilizada por um fluxo, 30% por outro e 20% por outro.
- Existe uma fila de espera por fluxo e é atribuído um peso ϕ_i a cada fila.
- Assim que uma fila de espera recebe tráfego, este começa imediatamente a ser servido, em paralelo com o restante tráfego, a uma taxa proporcional ao seu peso.
- É um modelo impossível de realizar na prática, mas constitui uma boa base teórica para o desenvolvimento de outros algoritmos.

Generalized Processor Sharing (GPS)

Num sistema com *N* filas e em que a ligação tem capacidade *C* a taxa de serviço garantida a cada fila é:

$$r_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} C$$

Dado que é um sistema do tipo *work-conserving*, a taxa de serviço efetiva da fila *i* pode ser superior a r_i. Em particular, a percentagem de largura de banda atribuída a cada fila é, em cada instante:

$$\frac{\phi_i}{\sum_{j\ ativo} \phi_j} C$$

Um fluxo diz-se ativo (*backlogged*) quando tem pacotes no sistema (em transmissão e/ou na fila de espera).
Um fluxo ativo no intervalo $[\tau, t]$ com serviço nesse mesmo intervalo $S_i(\tau, t)$ obedece à condição

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j},\ j = 1,\ 2,\ \ldots,\ N$$

Weighted Fair Queuing (WFQ)

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

Sempre que chega um pacote a uma fila, é atribuído um *Finish Number* (*FN*) ao pacote 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:

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

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.

O $FN_{i,k}$ do pacote k com comprimento L_k pertencente à fila i é dado por:

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

Self Clock Fair Queuing (SCFQ)

Por forma a evitar o cálculo do *RN* do WFQ, o SCFQ substitui este parâmetro pelo valor do *FN* do pacote em serviço, FN_s, qualquer que seja o fluxo a que pertence.

Assim, o $FN_{i,k}$ do pacote *k* com comprimento L_k pertencente à fila *i* é dado por:

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

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 num outro sistema qualquer.

Apesar do SCFQ ser de muito menor complexidade que o WFQ, pode não ser justo para pequenos intervalos de tempo.

Exemplo 3

Considere um algoritmo de escalonamento com 2 filas de espera de pesos $\phi_1 = 3$ e $\phi_2 = 1$, servido por uma ligação de 64 Kbps.
Chegam a este sistema os seguintes pacotes:

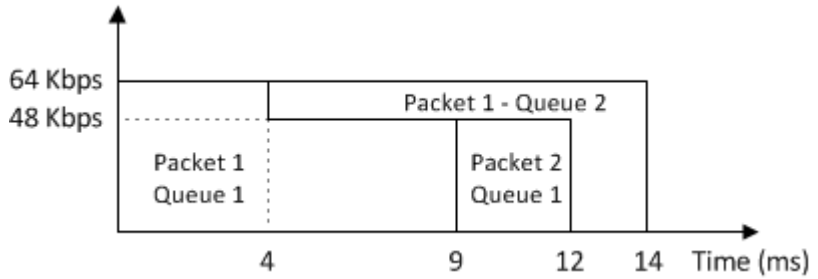
- pacote 1 à fila 1 com 62 Bytes em $t = 0$,
- pacote 1 à fila 2 com 32 Bytes em $t = 4$ ms e
- pacote 2 à fila 1 com 18 Bytes em $t = 6$ ms.

Determinar os instantes em que os pacotes são servidos considerando:

(a) uma disciplina de escalonamento GPS
(b) uma disciplina de escalonamento WFQ
(c) uma disciplina de escalonamento SCFQ

Exemplo 3 – resolução de (a)

2 filas de espera com pesos $\phi_1 = 3$ e $\phi_2 = 1$ com uma ligação de 64 Kbps. 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). Escalonamento GPS.



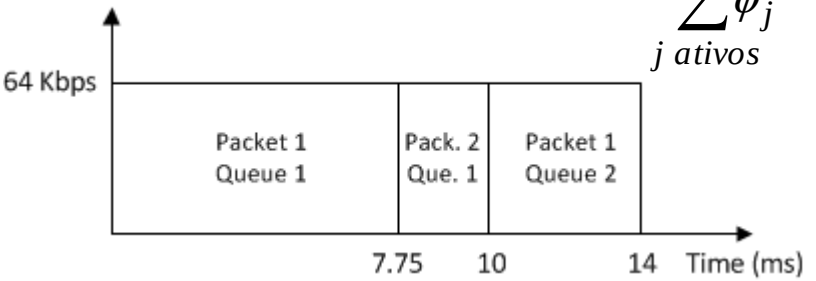
- O pacote 1 da fila 1 é servido inicialmente a 64 Kb/s. Em $t = 4$ ms, foram servidos (64Kb/s)×(4ms) = 256 bits = 32 Bytes do pacote 1 da fila 1. A partir daqui, a fila 1 é servida a (3/4)×64 Kb/s = 48 Kb/s e a fila 2 a (1/4)×64 Kb/s = 16 Kb/s.
- Com estas taxas, o pacote 1 da fila 1 demora (30×8)/(48Kb/s) = 5 ms a finalizar a sua transmissão e o pacote 1 da fila 2 demora (32×8)/(16Kb/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 (18×8)/(48Kb/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 (16Kb/s)×(8ms) = 128 bits = 16 Bytes, os restantes 16 Bytes demoram (16×8)/(64Kb/s) = 2 ms. Assim, o pacote 1 da fila 2 termina a transmissão em $t = 12 + 2 = 14$ ms.

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

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

Exemplo 3 – resolução de (b)

2 filas de espera com pesos $\phi_1 = 3$ e $\phi_2 = 1$ com uma ligação de 64 Kbps. 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). Escalonamento WFQ.

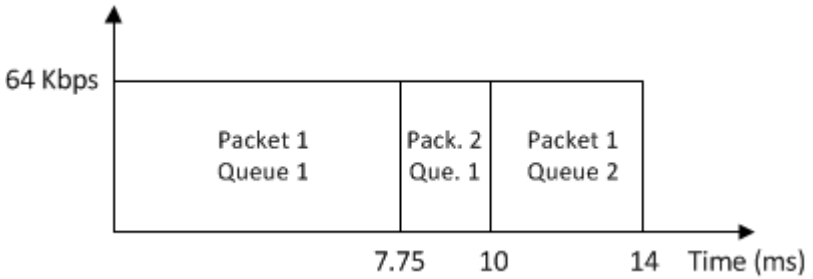


- Em $t = 0$ ms, $RN = 0$ e $FN_{1,1} = 0 + (62\times8)/64000/3 = 2.58\times10^{-3}$. O pacote 1 da fila 1 é transmitido em $(62\times8)/(64\text{Kb/s}) = 7.75$ ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em $t = 0 + 7.75 = 7.75$ ms.
- Em $t = 4$ ms : $RN = 0 + (4\times10^{-3})/3 = 1.33\times10^{-3}$
  $FN_{2,1} = 1.33\times10^{-3} + (32\times8)/64000/1 = 5.33\times10^{-3}$
- Em $t = 6$ ms : $RN = 1.33\times10^{-3} + (6\times10^{-3} - 4\times10^{-3})/4 = 3.33\times10^{-3}$
  $FN_{1,2} = \max(2.58\times10^{-3}, 3.33\times10^{-3}) + (18\times8)/64000/3 = 4.08\times10^{-3}$
- 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 $(18\times8)/(64\text{Kb/s}) = 2.25$ 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 $(32\times8)/(64\text{Kb/s}) = 4$ ms. Assim, o pacote 1 da fila 2 termina a sua transmissão em $t = 10 + 4 = 14$ ms.

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

Exemplo 3 – resolução de (c)

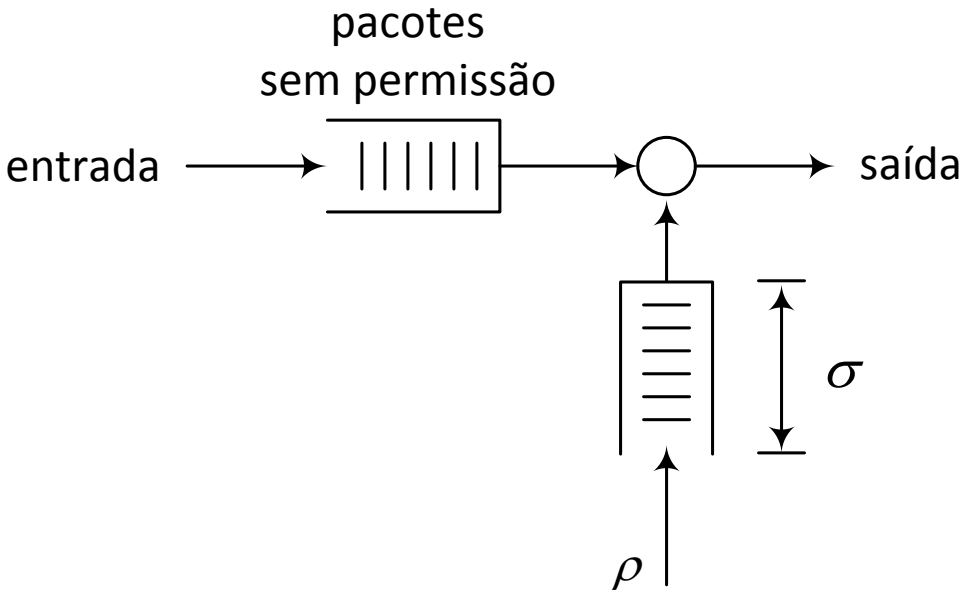
2 filas de espera com pesos $\phi_1 = 3$ e $\phi_2 = 1$ com uma ligação de 64 Kbps. Chegam: pacote 1 à fila 1 com 62 Bytes ($t = 0$), pacote 1 à fila 2 com 32 Bytes ($t = 4$ ms) e pacote 2 à fila 1 com 18 Bytes ($t = 6$ ms). Escalonamento SCFQ.



- Em $t = 0$ ms, $FN_{1,1} = 0 + (62\times8)/3 = 165.3$. O pacote 1 da fila 1 é transmitido em $(62\times8)/(64\text{Kb/s}) = 7.75$ ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em $t = 0 + 7.75 = 7.75$ ms.
- Em $t = 4$ ms : $FN_{2,1} = 165.3 + (32\times8)/1 = 421.3$
- Em $t = 6$ ms : $FN_{1,2} = \max(165.3, 165.3) + (18\times8)/3 = 213.3$
- 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 $(18\times8)/(64\text{Kb/s}) = 2.25$ 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 $(32\times8)/(64\text{Kb/s}) = 4$ 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 formatador de tráfego que permite impor um majorante ao tráfego gerado por um dado fluxo.



Se $A_i(\tau, t)$ representar a quantidade de tráfego do fluxo *i* que entra na 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 $S_i(\tau, t)$ o tráfego de um fluxo *i* servido num intervalo de tempo $[\tau, t)$, então:

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

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

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

O atraso máximo D_i será o tempo necessário para transmitir todo o tráfego em espera, que na pior das hipóteses será servido à taxa mínima de serviço r_i. Assim:

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

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 σ_i e ρ_i que atravessa *n* ligações ponto-a-ponto:

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:

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

Γ é o atraso total de propagação de todas as ligações

Exemplo 4

Considere um fluxo de pacotes de comprimento máximo de 200 Bytes formatado por um *leaky bucket* com parâmetros σ = 1000 bytes e ρ = 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 Kbps) que é necessário reservar em cada ligação para este fluxo, por forma a garantir um atraso máximo extremo-a-extremo de 20 mseg?

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

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

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

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? relembrar controlo de congestão do TCP!)

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 tratar os fluxos individualmente (mantendo o estado por fluxo) ou de forma agregada.
    - Na forma agregada, o método é aplicado a cada pacote, 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 WFQ isto corresponde a descartar o pacote com maior *Finish Number*.

Métodos de Descarte de Pacotes

Descarte antecipado

Descarte quando a fila de espera está cheia:

- Quando a fila está cheia por um largo período (a ligação está congestionada), múltiplos pacotes são descartados provocando a reação simultânea de todos 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 diretamente 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 (WRED – *Weighted* RED):

- Atribuem-se diferentes probabilidades a pacotes pertencentes a diferentes classes de serviço.

Exemplo – Arquitectura *DiffServ*

Classes de Serviço

- *Default* (DE) $\rightarrow$ DSCP = 000000
  - serviço *best-effort* com uma única fila de espera do tipo FIFO
- *Expedited Forwarding* (EF) $\rightarrow$ 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
  - em cada classe há 3 níveis de precedência para eliminação de pacotes em caso de congestionamento

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

Exemplo de Implementação